A multigrid method for elastic image registration with additional structural constraints

Inaugural-Dissertation

zur

Erlangung des Doktorgrades der

Mathematisch-Naturwissenschaftlichen Fakultät

der Heinrich-Heine Universität Düsseldorf

vorgelegt von

Lars Hömke

aus Düsseldorf

August 2006

Aus dem Mathematischen Institut
der Heinrich-Heine Universität Düsseldorf

Gedruckt mit der Genehmigung der
Mathematisch-Naturwissenschaftlichen Fakultät der Heinrich-Heine
Universität Düsseldorf

Referent: Prof. Dr. rer. nat. Kristian Witsch

Koreferent: Prof. Dr. rer. nat. Bernd Fischer (Universität zu Lübeck)

Tag der mündlichen Prüfung: 08.12.2006

Abstract

This thesis deals with the solution of a nonlinear inverse problem arising in digital image registration. In image registration one seeks to compute a transformation between two images such that they become more similar in some sense.

In the first part, we define the problem as the minimization of a regularized nonlinear least-squares functional, which measures the image difference and smoothness of the transformation. The nonlinear functional is linearized around a current approximation in order to obtain well-posed linear subproblems. The Hessian is replaced by an approximation that leads to an inexact Newton-type method, specifically a regularized Gauss-Newton method. A related gradient descent method is derived in the same framework for the purpose of comparison.

In the next part of this thesis we study geometric multigrid methods for the solution of the linear subproblems (inner iteration). The type of regularization employed leads to a system of elliptic partial differential equations. For the regularized Gauss-Newton method the differential operator contains jumping coefficients that cannot be adequately dealt with by standard geometric multigrid methods. Modifications of the multigrid components that improve multigrid convergence and allow for fast and efficient computation are proposed. In the outer iteration a trust region strategy is used and the whole procedure is embedded in a multiresolution framework. Extensive numerical results for the multigrid (inner iteration), outer iteration, and multiresolution framework are given.

In the last part a framework for the incorporation of additional structural constraints in the presented registration procedure is proposed. This framework is based on implicit representation of shapes via level sets. Representations for different types of shapes are discussed. Distance functionals of least-square type that can be easily plugged into the existing registration procedure are introduced. Examples for the different representations and the combination with image data are given.

Contents

List of Figures

List of Tables

List of Algorithms

CHAPTER 1

Introduction

Image registration is the process of computing a transformation between two images such that they become more similar. Other terms often found in the literature are warping, morphing, normalization, matching, and alignment. In general it is assumed that one image is a deformed version of the other one. There are two big classes of transformations that are considered: linear and non-linear ones. In the class of linear transformation one often makes a distinction between rigid and non-rigid transformations. Rigid transformations are volume preserving, i.e. only rotation and translation. Here we deal with non-linear transformations though in the application the computation of a non-linear transformation is almost always preceded by a linear transformation to compensate for global differences in shape, size, and orientation.

Image registration is a very active field of research, with applications in medical image analysis, target tracking, data fusion, recognition and satellite imagery. In the field of modern brain research image registration is a valuable tool. It allows for intra- and interindividual comparison of imaging data. In the intraindividual case image registration is necessary when data is from different imaging modalities, such as (functional) magnetic resonance imaging (MRI), positron emission tomography (PET) ,or histological sections, or the image data is acquired at different points in time. In the interindividual case the area of application is mainly the comparison of data from individual brains in a common reference space, e.g. for functional imaging studies or the construction of probabilistic atlas systems [5, 29, 33, 38, 45, 46]. Our applications are mainly from this area.

Since the goal of registration is to make the images more "similar" the modeling involves the minimization of a dissimilarity or distance functional. We differentiate between two different approaches to define such distance functionals. It can either be a function of the image pixels itself, intensity based, or a distance between landmarks, e.g. points, curves, or surfaces, that have to extracted in a preceding step. In the former case the information used in the registration process is given implicitly

by the whole image and in the latter explicit information about some parts of the image is used. Both approaches have there pros and cons which cannot be discussed at length here, but let us point out some of the most important arguments.

The landmark extraction step almost always involves manual user interaction. The matching of just some landmarks can often be more robust and precise especially when the images contain a lot of artefacts. Image information that is affected by artefacts does not have to be used and we can prescribe correspondences. On the other hand a perfect match of some landmarks does not necessarily yield a good match of the overall image, since no information from in-between is used. The result for the parts of the image not given by the landmarks is solely influenced by constraints on the solution, e.g. a smooth continuation of the transformation. Hence, when landmarks are not carefully chosen we end up with a mock precision. In case of complex transformations a lot of landmarks can be necessary to obtain satisfactory results. The result also depends on the appropriate localization of the extracted image landmarks. While landmarks are often easily defined in two-dimensional space an appropriate equivalent in three-dimensional space can often not be found. A one dimensional landmark in two-dimensional space is generally a curve in three-dimensional space. A contour (closed curve) in two-dimensional space is a surface in three-dimensional space. Defining a distance between landmarks (points) is easy, whereas defining a distance between curves and surfaces is more difficult due to the correspondence problem. With points the correspondence is uniquely defined. Between curves and surfaces a number of transformation lead to an identical match. Sometimes this problem is circumvented by discretizing curves and surfaces by a number of points. Yet, since the correspondence is not clear this leads to a mock precision, too. For an overview of landmark based techniques see e.g. [42].

In the intensity based approach the problem of establishing correspondence is implicitly taken care of. By implicitly we mean that we do not have to provide any information about correspondences a priori. Of course this does not entail that we always get the "correct" result. In the presence of artefacts the image distance might not be properly defined in all parts of the image. That can be dealt with by removing the artefacts in a preprocessing step, or by providing information about them to the registration algorithm [7, 28, 29]. Local minima can also lead to non desired correspondences. This is often due to locally ambiguous image information. Hence it would be advantageous to combine both approaches, but evidence of that in the literature is scarce. One example can be found in [18, 19] where an intensity based approach is combined with a one-to-one match of point landmarks. In [10] Cachier

uses a two step algorithm, where first the image difference and then the difference between geometric features in minimized by a closest point algorithm. In [13] the chamfer distance between representations of sulcal ribbons is minimized. Direct matching of the control points of B-splines is performed in [25].

We aim at solving the problem by minimizing a distance defined on the image intensities and on representations of geometric features simultaneously. We do not seek to establish a perfect correspondence on part of the geometric features, but let both representations "compete" in the minimization process. Both sides should augment or correct each other. The problem of reconciling both approaches lies in the fact that we have the implicit representation via image intensities on the one side and the explicit geometric representation on the other. The idea is to use implicit representations for the geometric features and define appropriate distance functions for them that can be plugged into an intensity based approach. We will first introduce an intensity based approach and later turn to geometric additional constraints.

The rest of this thesis is organized as follows. In the next chapter we give a more precise characterization of the modeling used here. Chapter 3 introduces two optimization approaches for the minimization of the non-linear functional proposed in chapter 2. In chapter 4 the numerical implementation is described. This includes the discretization, a multigrid solver for the linear equations derived in chapter 3, and a multiresolution framework. Numerical experiments for all aspects of the method are presented in chapter 5. The necessity of additional constraints for some registration problems is discussed in chapter 6 and a framework that makes use of implicit representations via level sets is presented. Results for the techniques proposed in chapter 6 are given in chapter 7. In the final chapter we offer conclusions and an outlook.

CHAPTER 2

Elastic image registration

In this chapter we introduce an intensity based approach to solving the registration problem, based on the minimization of the difference between gray values. Let us first state the image registration problem in general terms. Given are two images, a template T and a reference R. We assume that T and R can be represented by compactly supported functions

$$T, R : \Omega \subset \mathbb{R}^m \to \mathbb{R}. \tag{2.1}$$

Furthermore we assume that T is distorted by an invertible transformation Φ^{-1}. We search for the transformation

$$\Phi(u)(\cdot) : \ \mathbb{R}^m \ \to \ \mathbb{R}^m, \ \Phi(u)(x) : \ x \mapsto x - u(x), \ x \in \Omega. \tag{2.2}$$

In image processing one mostly finds $m = 2$ (images) or $m = 3$ (volume data sets).

Problem 2.1 (Image registration problem). *Find $u(x)$ such that the energy functional*

$$E_\alpha(u) := \mathcal{D}[T, R, \Omega; u] + \alpha \mathcal{R}[\Omega; u] : \mathbb{R}^m \to \mathbb{R} \tag{2.3}$$

is minimal with $\alpha \in \mathbb{R}^+$. $\mathcal{D}$ measures the disparity between $T(x - u(x))$ and $R(x)$ on the image domain Ω, and $\mathcal{R}$ imposes (smoothness) constraints on the solution u.

Generally this is an ill-posed problem, yet it becomes well-posed for some $\alpha > 0$. In the literature we find two main approaches to regularization. One is to regularize the functional directly [4, 16, 20, 31], as we have done here. The other is to linearize the non-linear functional and then regularize the linear subproblems [8, 12, 27, 30].

Several choices exist for $\mathcal{D}$ as well as for $\mathcal{R}$. A popular choice for $\mathcal{R}$ is the use of a bilinear form that leads to a partial differential operator. PDE based methods have become increasingly popular in the last couple of years. There is a whole

branch of PDE-based image processing methods, not only for registration, but also for image restoration, deconvolution, denoising, super resolution, inpainting, etc..

To measure the dissimilarity between R and T we use the integral (sum) of squared differences (SSD)

$$\mathcal{D}^{\text{SSD}} := \frac{1}{2}\int_{\Omega}(T(x-u(x))-R(x))^2\mathrm{d}x, \tag{2.4}$$

where we use integral in the continious and sum in the discrete context. In other words, the image distance is measured in the L_2-norm. The SSD can be used when identical structures are encoded by the same values in both images. If that is not the case other distance measures that relate the information contained in the images to each other have to be used, e.g. mutual information [36, 48].

As the regularizer $\mathcal{R}$ we chose $\mathcal{R}(u) = \ell[u,u]$ with the bilinear form

$$\ell[u,v] \;=\; \int_{\Omega}\Big(\lambda\mathrm{div}(u)\mathrm{div}(v) + 2\mu\sum_{i,j=1}^{m} e_{i,j}(u)e_{i,j}(v)\Big)\mathrm{d}x,$$

where

$$e_{i,j}(u) \;=\; \frac{1}{2}(\frac{\partial u_i}{\partial x_j}+\frac{\partial u_j}{\partial x_i}),$$

which is borrowed from the theory of linear elasticity for homogeneous and isotropic materials. It measures the elastic potential of the deformation u. The parameters $\lambda \geq 0$ and $\mu > 0$, the so-called Lamé constants, define the elastic properties of the underlying material. The elastic (Young's) modulus is given by $E = \frac{\mu(3\lambda+2\mu)}{\lambda+\mu}$. The first constant μ is the shear modulus and the λ is related to the bulk modulus $K = \lambda + \frac{3}{2}\mu$. The bilinear form ℓ can be written as $\ell[u,v] = \langle Lu, v\rangle$, where the canonical linear mapping L is given by the so-called *Navier-Lamé operator*

$$Lu \;=\; -\mu\Delta u - (\lambda+\mu)\nabla(\nabla u).$$

The symbol Δ is the vector valued Laplace operator and $\nabla(\nabla) = \mathrm{div}(\nabla)$. The choice of the regularizer is a modeling choice. The Navier-Lamé operator is frequently used for medical image registration problems since the deformation occurring in body tissues can be thought of being of elastic nature. In contrast to other popular regularizers, e.g. Δu, or $\Delta^2 u$, the Navier-Lamé operator couples the directions. The strength of the coupling depends on the choice of λ. For more information on the derivation see e.g. [41].

For our minimization problem we end up with a regularized nonlinear least squares functional that we tackle with a inexact Newton-type method, specifically a regularized Gauss-Newton method with a trust region approach. The functional is linearized around a current approximation which defines an iteration, the so called outer iteration. In each step of the outer iteration one normal equations which corresponds to a linear anisotropic subproblem is solved iteratively with a fast and efficient multigrid solver. The iteration in the solver is also called inner iteration. The Newton-type method is compared to a gradient descent method that is derived in the same framework. At some points we restrict ourselves to presentation of the 2d-case ($m = 2$) due to simplicity of presentation and space restrictions , but all techniques can be easily extended and are also used in 3d.

CHAPTER 3

Two optimization strategies for image registration

We present two optimization strategies for the solution of the minimization problem (2.3). The first method is a regularized gradient descent, and the second is Newton type method. The former is mainly introduced because it is a frequently used approach (c.f. [37] and references therein) and we need a reference method to illustrate the benefits of using higher order methods. Both methods are augmented with a trust region approach.

Since the functional is highly nonlinear a sequence of subproblems has to be solved. Each of these subproblems should have the following important features:

- it is well-posed and
- can be solved efficiently.

To this end E_α is linearized around a current approximation and the linearized model is used to construct an iterative method. The Taylor expansion of $E_\alpha(u)$ around $u^{(k)}$ is given by:

$$E_\alpha(u^{(k)} + v) \approx E_\alpha(u^{(k)}) + \langle J_{E_\alpha}(u^{(k)}), v\rangle + \frac{1}{2}\langle \mathrm{Hess}_{E_\alpha}(u^{(k)})v, v\rangle, \tag{3.1}$$

with $v = (u - u^{(k)})$ and $\langle\cdot,\cdot\rangle$ denoting the Euclidian scalar product. Using the abbreviation

$$T^{(k)}(x) = T(x - u^{(k)}(x))$$

for the transformed template image, the Jacobian and the Hessian of E_α at $u^{(k)}$ are given by

$$\begin{aligned} J_{E_\alpha}(u^{(k)}) &= J_\theta^t(u^{(k)})\theta(u^{(k)}) + \alpha L u^{(k)}, \\ \mathrm{Hess}_{E_\alpha}(u^{(k)}) &= J_\theta^t(u^{(k)})J_\theta(u^{(k)}) + S(u^{(k)}) + \alpha L, \end{aligned}$$

where

$$\theta(u^{(k)}) = T^{(k)}(x) - R(x)$$

is the difference between reference and transformed template. Since $J_\phi(u) = -I_m$, the $m \times m$ identity, we have

$$J_\theta^t(u^{(k)}) = J_T(\phi(u^{(k)})) \cdot J_\phi(u^{(k)}) = - \begin{pmatrix} \frac{\partial T^{(k)}(x)}{\partial x_1} \\ \vdots \\ \frac{\partial T^{(k)}(x)}{\partial x_m} \end{pmatrix}$$

and

$$J_\theta^t(u^{(k)}) J_\theta(u^{(k)}) = \begin{pmatrix} \frac{\partial T^{(k)}(x)}{\partial x_1}\frac{\partial T^{(k)}(x)}{\partial x_1} & \cdots & \frac{\partial T^{(k)}(x)}{\partial x_1}\frac{\partial T^{(k)}(x)}{\partial x_m} \\ \vdots & & \vdots \\ \frac{\partial T^{(k)}(x)}{\partial x_m}\frac{\partial T^{(k)}(x)}{\partial x_1} & \cdots & \frac{\partial T^{(k)}(x)}{\partial x_m}\frac{\partial T^{(k)}(x)}{\partial x_m} \end{pmatrix}.$$

Note that $J_\theta^t(u^{(k)}) J_\theta(u^{(k)})$ contains only first derivatives of the template image and is symmetric positive semi-definite. The term

$$S(u^{(k)}) = \theta(u^{(k)})\theta''(u^{(k)}) = \theta(u^{(k)})\nabla^2 T^{(k)}(x)$$

constitutes the more problematic part of $\mathrm{Hess}_{E_\alpha}(u^{(k)})$, which becomes small for small θ, i.e. well matched images. The Hessian $\mathrm{Hess}_{E_\alpha}(u^{(k)})$ is not necessarily regular. Even when it is regular it might have very small eigenvalues and hence be ill-conditioned. That leads to numerical problems.

We linearized the functional to deal with it nonlinearity. Consequently we have to solve a number of linear problems that are local approximations to the nonlinear functional. Replacing $u = u^{(k)} + v$ in equation (3.1) with $u^{(k+1)}$ defines an iteration

$$u^{(k+1)} = u^{(k)} + v, \tag{3.2}$$

where v is the solution to the normal equation

$$\langle J_{E_\alpha}(u^{(k)}), v\rangle + \frac{1}{2}\langle \mathrm{Hess}_{E_\alpha}(u^{(k)})v, v\rangle = 0. \qquad \forall v \in \mathcal{V}, \tag{3.3}$$

in a suitable function space $\mathcal{V}$. This iteration is also called outer iteration. As noted above $\mathrm{Hess}_{E_\alpha}(u^{(k)})$ is bound to cause problems for the solution of (3.3). Hence, it is usually replaced by a more well-behaved approximation. In the following we introduce two options, one that leads to a gradient descent method and one that leads to a Newton-type method.

3.1 Regularized gradient descent method

When we replace the data dependent and possibly ill-conditioned part of the Hessian, i.e. $J_\theta^t(u^{(k)})J_\theta(u^{(k)}) + S(u^{(k)})$ by a symmetric positive definite (spd) operator βB, $\beta \in \mathbb{R}_0^+$, we obtain a regularized gradient descent method with a trust region approach, i.e. the minimization problem

$$\min_{v\in\mathcal{V}} \left\{ \langle J_{E_\alpha}(u^{(k)}), v\rangle + \frac{\alpha}{2}\langle Lv, v\rangle + \frac{\beta}{2}\langle Bv, v\rangle \right\}. \tag{3.4}$$

The solutions to the corresponding quadratic normal equation

$$\langle J_{E_\alpha}(u^{(k)}), v\rangle + \frac{1}{2}\langle(\alpha L + \beta B)v, v\rangle = 0 \qquad \forall v \in \mathcal{V}, \tag{3.5}$$

are directions of steepest descent on E_α. The addition of βB can be interpreted as a trust region strategy. The linearization of E_α can only be trusted a region that depends on the nonlinearity of the functional at $u^{(k)}$. The solutions to the normal equation (3.4) are restricted to a trust region with a radius η_k that is measured in the norm induced by B.

Lemma 3.1. *The unconstrained minimization problem (3.4) is equivalent to the constrained minimization problem*

$$\min_{v\in\mathcal{V}} \left\{ \langle J_{E_\alpha}(u^{(k)}), v\rangle + \frac{\alpha}{2}\langle Lv, v\rangle \right\} \quad \textit{s.t. } ||v||_B^2 \leq \eta_k. \tag{3.6}$$

Proof. Using the method of Lagrange multipliers (3.6) can be rewritten as

$$\begin{aligned} & \langle J_{E_\alpha}(u^{(k)}), v\rangle + \tfrac{\alpha}{2}\langle Lv, v\rangle + \lambda||v||_B^2 \\ = \;& \langle J_{E_\alpha}(u^{(k)}), v\rangle + \tfrac{\alpha}{2}\langle Lv, v\rangle + \lambda\langle Bv, v\rangle. \end{aligned}$$

Setting the Lagrange multiplier λ to $\beta/2$ yields the desired result. □

The regularization parameter β directly determines the degree of regularization by B. The more regularization, the smaller the trust region radius. Another interpretation is that β^{-1} is the size of a time-step. Consider the time evolution equation

$$\partial u_t + \alpha L u + J_D(u) = 0 \quad , 0 \leq t \leq T,$$

where J_D is the Jacobian of the distance measure. When u^* is a stationary point ∂u_t vanishes and u^* is the solution. The semi-implicit time discretization of that

equation has the form

$$\frac{u^{(k+1)} - u^{(k)}}{\tau} + \alpha L u^{(k+1)} + J_D(u^{(k)}) = 0. \tag{3.7}$$

Lemma 3.2. *Let $B = I$, with I the identity, and $\beta = \tau^{-1}$, then (3.7) is equivalent to the classical formulation of the weak form (3.5) (see also proposition 3.1 on page 13).*

Proof.

$$\begin{aligned}
& (\alpha L + \tau^{-1} I)v &&= -J_{E_\alpha}(u^{(k)}) \\
\Leftrightarrow \quad & (\alpha L + \tau^{-1} I)(u^{(k+1)} - u^{(k)}) &&= -(J_\theta^t(u^{(k)})\theta(u^{(k)}) + \alpha L u^{(k)}) \\
\Leftrightarrow \quad & \alpha L u^{(k+1)} + \tau^{-1}(u^{(k+1)} - u^{(k)}) &&= -(J_\theta^t(u^{(k)})\theta(u^{(k)})) \\
\Leftrightarrow \quad & \tfrac{u^{(k+1)} - u^{(k)}}{\tau} + \alpha L u^{(k+1)} + J_D(u^{(k)}) &&= 0
\end{aligned}$$

□

If we do not replace B by I the size of the time step is measured in the norm $||\cdot||_B$. The smaller the time step τ the smaller the change that is to be expected. That is in accordance with what we have stated above about the influence of β.

3.2 Regularized Gauss-Newton method

In the last section we replaced all image dependent terms of the Hessian by a spd operator. Now we only replace $S(u^{(k)})$, the part that contains higher order derivatives of the template image, by a spd operator βB, $\beta \in \mathbb{R}_0^+$. That leads to the unconstrained minimization problem

$$\min_v \left\{ \langle J_{E_\alpha}(u^{(k)}), v\rangle + \frac{1}{2}\langle (J_\theta^t(u^{(k)}) J_\theta(u^{(k)}) + \alpha L + \beta B)v, v\rangle \right\}. \tag{3.8}$$

The corresponding quadrative normal equation to be solved is

$$\langle J_{E_\alpha}(u^{(k)}), v\rangle + \frac{1}{2}\langle (J_\theta^t(u^{(k)}) J_\theta(u^{(k)}) + \alpha L + \beta B)v, v\rangle = 0 \qquad \forall v \in \mathcal{V}, \tag{3.9}$$

and the solution v is a descent direction on E_α. Again the addition of βB can be interpreted as a trust region strategy.

Lemma 3.3. *The constrained minimization problem*

$$\min_v \left\{ \langle J_{E_\alpha}(u^{(k)}), v\rangle + \frac{1}{2}\langle (J_\theta^t(u^{(k)}) J_\theta(u^{(k)}) + \alpha L)v, v\rangle \right\}, \quad \textit{s.t.} ||v||_B \leq \eta_k, \tag{3.10}$$

is equivalent to the unconstrained minimization problem (3.8).

Proof. The proof is analogue to the proof for lemma 3.1. □

The combination of a Gauss-Newton method with a trust region strategy is often called Levenberg-Marquardt method.

3.3 Trust region approach

In lemmas 3.1 and 3.3 the link between the unconstrained and constrained minimization problems was shown. We can either interpret the addition of βB as an extra regularization of the unconstrained minimization problem, or as a constraint on the size of a trust region for a constraint minimization problem. We left open the question how the trust-region parameter β and the spd operator B are to be chosen.

3.3.1 Choice of the trust-region parameter β

The problem with the choice of β is that if the linear model is a good approximation to $E_\alpha(u)$ around $u^{(k)}$ we can take large steps, thereby reducing the number of outer iterations. If it is a bad approximation we should take small steps, since otherwise we do not compute sensible solutions, and the iteration might even diverge. The solution of the normal equation is the by far most expensive step. Hence, we have to find a reasonable compromise between efficency and robustness.

There are a number of strategies to control the trust region radius. For a new approximation $u^{(k+1)} = u^{(k)} + v$ we require that

$$E_\alpha(u^{(k+1)}) \leq E_\alpha(u^{(k)}) + \rho\langle f(u^{(k)}), v\rangle, \tag{3.11}$$

with some positive scalar ρ. The minimum permissable value for ρ is then given by

$$\rho_k = \frac{E_\alpha(u^{(k+1)}) - E_\alpha(u^{(k)})}{\langle f(u^{(k)}), v\rangle}.$$

This is also called the Armijo-Goldstein rule. It assesses the difference between the projected and computed solution. When (3.11) is fulfilled, i.e. $\rho_k \geq \rho$ we accept the update to the solution. When we have to reject the solution the trust region has to be made smaller which is identical to increasing the regularization. When ρ_k is much larger than ρ we can decrease the regularization. These considerations lead

to following update procedure for the parameter β_{k+1} and the new solution $u^{(k+1)}$:

$$(\beta_{k+1}, u^{(k+1)}) = \begin{cases} (2\beta_k, u^{(k)}) & \text{for } \rho_k < \rho_- \\ (\frac{1}{2}\beta_k, u^{(k)} + v) & \text{for } \rho_k > \rho_+ \\ (\beta_k, u^{(k)} + v) & \text{otherwise} \end{cases} \tag{3.12}$$

When $\rho_k < \rho_-$ the update is rejected and regularization is increased, i.e. the trust region is made smaller. In all other cases the update is accepted and if $\rho_k > \rho_+$ the trust region radius is increased. Typical values from literature for the thresholds are $\rho_- = 0.1$ and $\rho_+ = 0.5$ [24, 27].

Note that the trust region parameter is chosen a priori. The degree of regularization by B is fixed before the computation of v. To determine β_k information from the previous iteration step is used. That means we have to chose a reasonable value for β_0. There are two possible situations. If β_0 is to small v will be rejected until a proper degree of regularization is reached. When β_0 is chosen too conservative the update procedure (3.12) will decrease the regularization parameter during the iteration. The latter situation is the better one, since until the regularization parameter is adapted accordingly a number updates to the solution have already been computed and accepted. In the former situation the solution is identical to u_0 until the proper degree of regularization is reached. In both cases a value of β_0 that is far off the optimal choice will result in a large number of "unnecessary" iterations. There is no ideal way to chose β_0 because since it is an a priori choice you only find out if it has been a good choice after an expensive solve of the normal equation. For a more detailed discussion of trust-region methods see e.g. [15].

3.3.2 Choice of the trust region topology

Before we have always said that B should be spd, because we want $\alpha L + \beta B$, respectively $\alpha L + \beta B + J_\theta^t J_\theta$ to be spd. Then we a unique solution v exists. Clearly this property depends on the function space $\mathcal{V}$ (see the normal equations (3.5) and (3.9)). Let

$$V_0 := \overbrace{H_0^1(\Omega) \times \ldots \times H_0^1(\Omega)}^{m\text{-times}}.$$

Here the space $H^1(\Omega)$ is the Sobolev space of functions in $L^2(\Omega)$ with weak derivatives in $L^2(\Omega)$. $H_0^1(\Omega)$ is the subspace of $H^1(\Omega)$ consisting of the functions with zero boundary values, i.e. Dirichlet boundary conditions.

We can formulate the following two propositions about the uniqueness of the minimizer and the equations to be solved.

Proposition 3.1. *Let $\alpha, \beta \in \mathbb{R}^+$, and $L + B$ be a spd operator for all $v \in V_0$, then there exists a unique minimizer $u^* \in V_0$ of (3.4) that is characterized by the classical boundary value problem*

$$\begin{aligned} (\alpha L + \beta B)v &= -J_{E_\alpha}(u^{(k)}) \quad \text{for } x \in \Omega, \\ v &= 0 \qquad\qquad\quad \text{for } x \in \partial\Omega. \end{aligned} \tag{3.13}$$

Proposition 3.2. *Let $\alpha, \beta \in \mathbb{R}^+$, and $L + B$ be a spd operator, then there exists a unique minimizer $u^* \in V_0$ of (3.8) that is characterized by the classical boundary value problem*

$$\begin{aligned} (J_\theta^t(u^{(k)})J_\theta(u^{(k)}) + \alpha L + \beta B)v &= -J_{E_\alpha}(u^{(k)}) \quad \text{for } x \in \Omega, \\ v &= 0 \qquad\qquad\quad \text{for } x \in \partial\Omega. \end{aligned} \tag{3.14}$$

The proof of proposition 3.1 and 3.2 is a direct application of the fundamental lemma of the calculus of variation (see e.g. [41]). For the proof of proposition 3.2 it should be noted that $J_\theta^t J_\theta$ is symmetric positive semidefinite and with the assumption that $L + B$ is spd $J_\theta^t J_\theta + \alpha L + \beta B$ is spd, too.

Lets take a look at the different parts of the operators and their properties with respect to V_0. $J_\theta^t J_\theta$ is symmetric positive semidefinite. It cannot generally be spd because $T_{h_{x_i}}$ can be 0. The elliptic operator L is spd in V_0. Only one part has to be spd and the other can be only symmetric positive semidefinite, because a spd operator remains spd when we add something symmetric positive semidefinite. Thus, with Dirichlet boundary conditions B has to be symmetric positive semidefinite.

The simplest choice for B is the identity I. This is related to Tihkhonov regularization(see e.g. [6, 17]), since $\langle \beta I v, v\rangle = \beta ||v||_2^2$ (see also propositions 3.1 and 3.3). As shown in section 3.1 it is also a natural consequence of a semi-implicit time discretization for the gradient descent. Obviously I is spd, but it is not clear how a regularization by I relates to the original problem. We modeled our problem such that the solution should be "elastic" by the use of the Navier-Lame operator L. The topology of the trust region depends on the metric that is induced by the norm of the operator B. Depending on the balance of α and β the influence of B on the type of the solution increases. That leaves us with the problem that the solutions computed in each step of the outer iteration are not "elastic", and we cannot expect them to magically add up to reduce the penalty term in E_α. When β is to large the outer iteration can stop prematurely.

Another choice is to set $B := L$. Then the topology of trust region is induced by the regularizing term of the original problem. We look at both choices in more detail in section 5.2.2 where numerical experiments are presented.

The normal equation that has to be solved in each step of the outer iteration (3.2) of the regularized Gauss-Newton method is then

$$\langle \underbrace{J_{E_\alpha}(u^{(k)})}_{:=-f(u^{(k)})}, v\rangle + \frac{1}{2}\langle (\underbrace{J_\theta^t(u^{(k)})J_\theta(u^{(k)}) + (\overbrace{\alpha+\beta_k}^{:=\tilde{\alpha}_k})L}_{:=M})v, v\rangle, \quad \forall v \in V_0. \tag{3.15}$$

The abbreviations M, f, and $\tilde{\alpha}$, will be frequently used in the following sections.

A remark on Neumann boundary conditions

Let

$$V_1 := \overbrace{H^1(\Omega) \times \ldots \times H^1(\Omega)}^{m\text{-times}}$$

be the function space associated with Neumann boundary conditions. L is not spd because $\ell[u,u] \equiv 0$ for all constant functions $u_c = (c_1, c_2, \ldots, c_m)$, $\exists i : c_i \neq 0$. One option would be to chose a spd operator like I. Another option is to look at the operator $M := J_\theta^t(u^{(k)})J_\theta(u^{(k)}) + \tilde{\alpha}L$ and derive conditions under which it is spd. When

$$\ker(J_\theta^t(u^{(k)})J_\theta(u^{(k)})) \cap \ker(L) = 0$$

M would be spd. Since the constant functions u_c are the problem we look at $(J_\theta^t(u^{(k)})J_\theta(u^{(k)}))u_c = 0$. That leads to a system of m equations

$$T_{x_i}^k \left(\sum_{j=1}^{m} c_j T_{x_j}^k \right) = 0, \quad 1 \leq i \leq m.$$

The solutions are $T_{x_1}^k = \ldots = T_{x_m}^k = 0$ and $\sum_{j=1}^m c_j T_{x_j}^k = 0$. Note that the second term also holds when the first term is fulfilled and that it includes solutions where some of the $T_{x_j}^k$ are zero. Thus there exists a unique minimizer $u^* \in V_1$ of (3.8) if

$$\int_\Omega \left(c_1 T_{x_1}^k + \ldots + c_m T_{x_m}^k\right)^2 dx > 0, \tag{3.16}$$

holds. Note that we cannot formulate a similar statement for (3.4), because additional terms like $J_\theta^t(u^{(k)})J_\theta(u^{(k)})$ are not present. For the gradient descent with Neumann boundary conditions we have to chose an operator B that is spd.

In our implementation we always use Dirichlet boundary conditions, $u = 0, u \in \partial\Omega$, instead of Neumann boundary conditions, $\frac{\partial u}{\partial n} = 0, u \in \partial\Omega$, because of the data independent guaranteed invertibility of M for the Gauss-Newton method. When we compare the gradient descent with the Gauss-Newton method we always use the same boundary conditions and operator B.

CHAPTER 4

Numerical Implementation

In this chapter the discuss the discretization of the equations derived in the previous chapter. The principal concepts behind multigrid methods are introduced and modifications to the standard components that allow for the robust and efficient computation for equations with jumping coefficients are proposed. The structure of the outer iteration is discussed and a multiresolution framework in which the outer iteration is embedded is introduced.

4.1 Discretization

We regard the images as given on a domain $\Omega \subset \mathbb{R}^m$. For $m = 2$ that is usually the unit square $[0,1]^2$ and for $m = 3$ the unit cube $[0,1]^3$. The image data is given on a regular equidistant grid, with one pixel/voxel of a given value at each grid point. The discretization step in each dimension is given by $h_j = \frac{1}{N_j - 1}$, $1 \leq j \leq m$, where N_j is the number of pixels along dimension j, and $\mathbf{h} = (h_1, h_2, \ldots, h_m)$. Hence the discretization of $\Omega^m = [0,1]^m$ is given by

$$\begin{aligned} \Omega^m_{\mathbf{h}} &= \mathcal{G}^m_{\mathbf{h}} \cap \Omega^m, \\ \partial\Omega^m_{\mathbf{h}} &= \mathcal{G}^m_{\mathbf{h}} \cap \partial\Omega^m, \end{aligned} \tag{4.1}$$

with the grid

$$\mathcal{G}^m_{\mathbf{h}} := \{x_{i_1 i_2 \ldots i_m} | x_{i_1 i_2 \ldots i_m} := (i_1 h_1, i_2 h_2, \ldots, i_m h_m), i_j \in \mathbb{Z}\}. \tag{4.2}$$

The discrete images are then given by:

$$\begin{aligned} T_{\mathbf{h}} &:= \mathcal{G}^m_{\mathbf{h}} \cap T = \{T_{i_1 i_2 \ldots i_m}\}_{i_j=0}^{N_j} \in \mathbb{R}^{N_1 \times N_2 \times \ldots \times N_m} \\ R_{\mathbf{h}} &:= \mathcal{G}^m_{\mathbf{h}} \cap R = \{R_{i_1 i_2 \ldots i_m}\}_{i_j=0}^{N_j} \in \mathbb{R}^{N_1 \times N_2 \times \ldots \times N_m} \end{aligned}$$

The discretized version of u and $f := f(u)$ can then be expressed in the following form:

$$
\begin{aligned}
u_{\mathbf{h}} &:= \mathcal{G}^m_{\mathbf{h}} \cap u = \{u_{l,i_1 i_2 \ldots i_m}\}_{l=1,i_j=0}^{m,N_j} \in \mathbb{R}^{N_1 \times N_2 \times \ldots \times N_m \times m} \\
f_{\mathbf{h}} &:= \mathcal{G}^m_{\mathbf{h}} \cap f = \{f_{l,i_1 i_2 \ldots i_m}\}_{l=1,i_j=0}^{m,N_j} \in \mathbb{R}^{N_1 \times N_2 \times \ldots \times N_m \times m}
\end{aligned} \tag{4.3}
$$

The partial derivatives in the differential operator L are approximated by central finite differences of second order accuracy. The derivatives of the template image $T_{\mathbf{h}}$ are computed likewise. Let $u_{l,i_1 i_2 \ldots i_m + e_j} = u_{l,i_1 \ldots i_j + 1 \ldots i_m}$ then the approximations for the partial derivatives of $u_{l,i_1 i_2 \ldots i_m}$ and $T_{i_1 i_2 \ldots i_m}$ are:

$$
\begin{aligned}
\frac{\partial u_{l,i_1 i_2 \ldots i_m}}{\partial x_j} &= \frac{u_{l,i_1 i_2 \ldots i_m + e_j} - u_{l,i_1 i_2 \ldots i_m - e_j}}{2h_j} + \mathcal{O}(h_j^2), \\
\frac{\partial^2 u_{l,i_1 i_2 \ldots i_m}}{\partial x_j^2} &= \frac{u_{l,i_1 i_2 \ldots i_m - e_j} - 2u_{l,i_1 i_2 \ldots i_m} + u_{l,i_1 i_2 \ldots i_m + e_j}}{h_j^2} + \mathcal{O}(h_j^2), \\
\frac{\partial^2 u_{l,i_1 i_2 \ldots i_m}}{\partial x_j \partial x_k} &= \frac{1}{4h_j h_k} \Big(u_{l,i_1 i_2 \ldots i_m + e_j + e_k} - u_{l,i_1 i_2 \ldots i_m - e_j + e_k} \\
&\quad + u_{l,i_1 i_2 \ldots i_m - e_j - e_k} - u_{l,i_1 i_2 \ldots i_m + e_j - e_k} \Big) + \mathcal{O}(h_j h_k), \\
\frac{\partial T_{i_1 i_2 \ldots i_m}}{\partial x_j} &= \underbrace{\frac{T_{i_1 i_2 \ldots i_m + e_j} - T_{i_1 i_2 \ldots i_m - e_j}}{2h_j}}_{=: T_{h_{x_j}}} + \mathcal{O}(h_j^2).
\end{aligned}
$$

The use of different discretization steps for each dimension would cause undue complications in the subsequent exposition , e.g. very lengthy, unreadable formulas. For the ease of representation we henceforth assume that the discretization step is identical for all dimensions, i.e. $\forall h_j : h_j \equiv h$, $1 \leq j \leq m$. The formulas for varying discretization step can be easily derived.

With the above definitions the stencil representation of $M_h u_h = f_h$ for $m = 2$ is given in figure 4.1. The dashed line separates the equations for the the two dimensions. It can be easily seen that the equations are coupled by the mixed first order derivatives, and that the strength of the coupling is determined by λ.

So far we have expressed the discretization in the terminology of discrete differential operators, i.e. as grid functions, e.g. u_h, and grid operators, like the stencil representation above. The linearization that we performed in section 3 lead to a linear system of equations. The classical form to describe such a linear system of equations is with matrices and vectors. Both representations have their merits. The representation in form of grid functions and grid operators will be especially

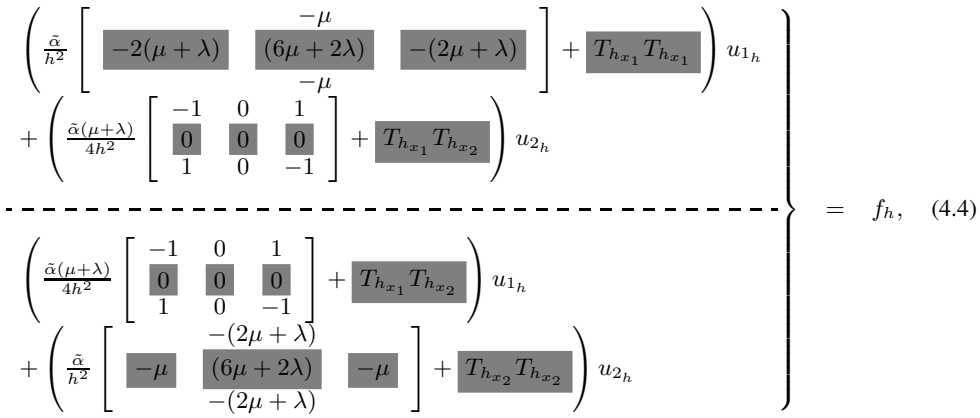


Figure 4.1: Stencil representation of $M_h u_h = f_h$ for $m = 2$. The dashed line separates the equations for the the two dimensions.

helpful for the introduction of the multigrid in the following chapter. The structure of the linear system can be more easily inferred from the matrix representation. Furthermore the matrix representation can be directly fed into standard solver for linear systems of equations. Before we give the matrix representation of the partial differential operator $M_h = \tilde{\alpha} L_h + J_{h_\theta}^t J_{h_\theta}$ (see equation (3.15)), we have to define some operators that allow us to construct the matrix efficiently.The **diag**-operator constructs matrices by filling diagonals with a given offset with fixed values.

Definition 4.1 (diag-operator). *Let* $A = diag[(b_0, \ldots, b_l)(d_0, \ldots, d_l), n] \in \mathbb{R}^{n \times n}$ *with values* b_i *on the diagonals with offset* d_i*, i.e.*

$$a_{ij} = \begin{cases} b_k & \text{if } i - j = d_k \\ 0 & \text{otherwise} \end{cases} . \tag{4.5}$$

For example, a tridiagonal matrix of size 5 with entries b_0 on the lower diagonal, b_1 on the main diagonal, and b_2 on the upper diagonal, is given by

$$\text{diag}[(b_0, b_1, b_2)(-1, 0, 1), 5] = \begin{pmatrix} b_1 & b_2 & 0 & 0 & 0 \\ b_0 & b_1 & b_2 & 0 & 0 \\ 0 & b_0 & b_1 & b_2 & 0 \\ 0 & 0 & b_0 & b_1 & b_2 \\ 0 & 0 & 0 & b_0 & b_1 \end{pmatrix} .$$

The Kronecker product allows us to easily construct block matrices.

Definition 4.2 (Direct matrix product (Kronecker product)). *Given two matrices $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{p\times q}$, the elements of the direct matrix product $C = A \otimes B \in \mathbb{R}^{mp\times nq}$ are defined by*

$$c_{\alpha\beta} = a_{ij}b_{kl}$$

with

$$\begin{aligned} \alpha &\equiv p(i-1)+k, \\ \beta &\equiv q(j-1)+l. \end{aligned}$$

Alternatively one can write

$$A \otimes B = \begin{pmatrix} a_{11}B & a_{12}B & \cdots & a_{1n}B \\ a_{21}B & a_{22}B & \cdots & a_{2n}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}B & a_{m2}B & \cdots & a_{mn}B \end{pmatrix}.$$

Together with the **diag**-operator we have a powerful tool for building block matrices of any kind. If, for example, we would like to construct a block diagonal matrix, with m identical blocks B we can simply write

$$C = \text{diag}[(1),(0),m] \otimes B.$$

The matrix B could be the discretized operator for a line in the image, for example. The matrix C would then be the discretized operator for m such lines.

The images and the solution are given as multi-dimensional grid functions (see equation (4.3)), but for the matrix representation we have to convert them to vectors. The **vec**-operator linearizes a m-dimensional grid function to a single vector with the identical number of elements. It is used to generate the solution and right hand side vectors.

Definition 4.3 (vec-operator). *Let $A = \{a_{i_1 i_2 \ldots i_m}\}_{i_1,i_2,\ldots,i_m}^{N_1,N_2,\ldots,N_m} \in \mathbb{R}^{N_1\times N_2\times\ldots\times N_m}$ be a m-dimensional matrix with a total of $N = \prod_{j=1}^{m} N_j$ elements. The mapping*

$$\begin{aligned} \mathit{vec} : \mathbb{R}^{N_1\times N_2\times\ldots\times N_m} &\to \mathbb{R}^N \\ A &\mapsto \mathit{vec}(A) \end{aligned}$$

is defined by the bijective index mapping ζ *with* $a_{i_1 i_2 \dots i_m} = vec(A)_{\zeta(i_1 i_2 \dots i_m)}$

$$\begin{aligned} \zeta : \mathbb{N}^m &\rightarrow \mathbb{N} \\ i_1 i_2 \dots i_m &\mapsto i_1 + \sum_{k=2}^{d} (i_k - 1) \prod_{l=1}^{k-1} N_k. \end{aligned}$$

Next we give the discrete form $M_h^{[m]}$, $m = 2, 3$, of M. Since Dirichlet boundary conditions, i.e. $\partial\Omega_h^m = 0$, are used, the boundary can be omitted. Hence the discretization matrix has size $N_1 - 2 \times \ldots \times N_m - 2$. We first define the auxiliary matrices

$$\begin{aligned} D_0^{N_l} &:= \text{diag}[(1), (0), N_l - 2], \\ D_1^{N_l} &:= \text{diag}[(1, 1), (-1, 1), N_l - 2], \\ D_2^{N_l} &:= \text{diag}[(-1, 1), (-1, 1), N_l - 2]. \end{aligned}$$

The Kronecker product $D_0 \otimes B$ creates a block diagonal matrix with blocks B. The matrices D_1, D_2 are used to shift coefficients up one dimension. While $(D_1^{N_2} \otimes D_0^{N_1})\,\text{vec}(X)$, $X \in \mathbb{R}^{N_1 \times N_2}$, acts on $x_{j-1,j}$ and $x_{j+1,j}$, $(D_0^{N_2} \otimes D_0^{N_1})\,\text{vec}(X)$ acts on $x_{j,j-1}$ and $x_{j,j+1}$.

With the help of these matrices we define the discrete version of the operator M for the two-dimensional ($m = 2$) and the three-dimensional ($m = 3$) case. For the ease of notation we henceforth use the substitution

$$A_h := J_{\theta_h}^t J_{\theta_h}$$

for the coefficients from the template image.
In the two-dimensional case the matrix has the form

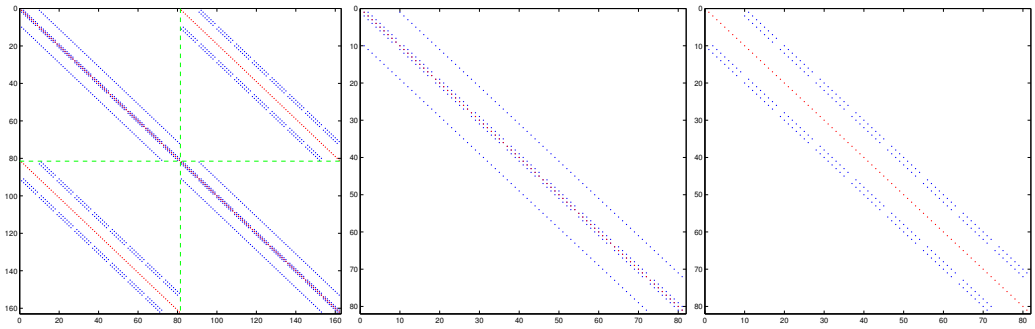
$$M_h^{[2]} = L_h^{[2]} + A_h^{[2]} = \underbrace{\left(\begin{array}{c|c} L_{11}^{[2]} & L_{12}^{[2]} \\ \hline L_{12}^{[2]} & L_{22}^{[2]} \end{array}\right)}_{=L_h^{[2]}} + \underbrace{\left(\begin{array}{c|c} A_{11} & A_{12} \\ \hline A_{21} & A_{22} \end{array}\right)}_{A_h^{[2]}} \tag{4.6}$$

where the individual blocks are given by

$$\begin{aligned} G_0 &= \frac{\tilde{\alpha}\mu}{h^2}\Big(D_0^{N_2} \otimes (4D_0^{N_1} - D_1^{N_1}) + D_1^{N_2} \otimes (-D_1^{N_1})\Big), \\ L_{11}^{[2]} &= G_0 + \frac{\tilde{\alpha}(\mu + \lambda)}{h^2}\Big(D_0^{N_2} \otimes (2D_0^{N_1} - D_1^{N_1})\Big), \end{aligned}$$

$$
\begin{aligned}
L_{22}^{[2]} &= G_0 + \frac{\tilde{\alpha}(\mu+\lambda)}{h^2}\Big(D_1^{N_2} \otimes (-D_1^{N_1}) + D_0^{N_2} \otimes 2D_0^{N_1}\Big), \\
L_{12}^{[2]} &= \frac{\tilde{\alpha}(\mu+\lambda)}{4h^2}\Big(- D_2^{N_2} \otimes D_2^{N_1}\Big), \\
A_{ij} &= \operatorname{diag}\left[\left(\operatorname{vec}\left(T_{h_{x_i}}\right)\right), (0), \prod_{l=1}^{n} N_l - 2\right] \\
&\quad \cdot \operatorname{diag}\left[\left(\operatorname{vec}\left(T_{h_{x_j}}\right)\right), (0), \prod_{l=1}^{n} N_l - 2\right].
\end{aligned}
$$

G_0 is the discretization of the Laplacian Δ. The auxiliary parts in $L_{11}^{[2]}$ and $L_{22}^{[2]}$ constitute an additional second order derivative in the respective direction. The block $L_{12}^{[2]}$ is identical for both equations and represent the mixed direction first order derivatives that couple the system of equations. The blocks A_{ij} are diagonal matrices with products of first order derivatives of the transformed template image as entries.



(a) $M_h^{[2]}$ (b) $M_{11}^{[2]} \& M_{22}^{[2]}$ (c) $M_{12}^{[2]}$

Figure 4.2: Sparsity pattern for $M_h^{[2]}$, with $N_l = 9$, $l = 1, 2$. The contributions of A_{ij} are marked red, all other entries are blue.

In the three-dimensional case the matrix has the form

$$
M_h^{[3]} = L_h^{[3]} + A_h^{[3]} = \underbrace{\left(\begin{array}{c|c|c} L_{11}^{[3]} & L_{12}^{[3]} & L_{13}^{[3]} \\ \hline L_{12}^{[3]} & L_{22}^{[3]} & L_{23}^{[3]} \\ \hline L_{13}^{[3]} & L_{23}^{[3]} & L_{33}^{[3]} \end{array}\right)}_{L_h^{[3]}} + \underbrace{\left(\begin{array}{c|c|c} A_{11} & A_{12} & A_{13} \\ \hline A_{12} & A_{22} & A_{23} \\ \hline A_{13} & A_{23} & A_{33} \end{array}\right)}_{A_h^{[3]}} \tag{4.7}
$$

where the individual blocks are given by

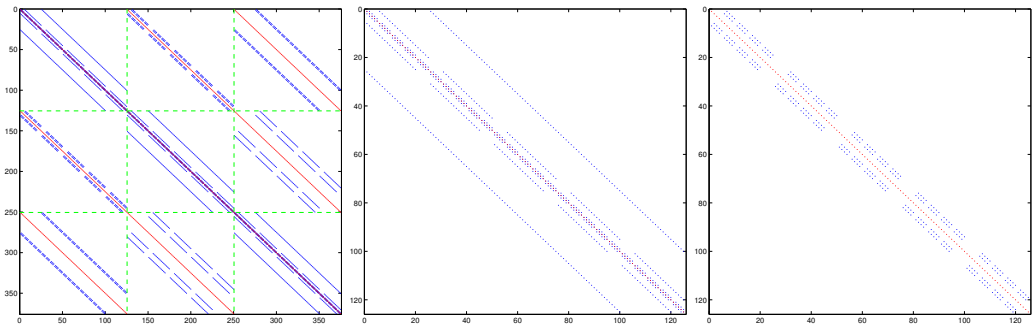
$$
G_1 = \frac{\tilde{\alpha}\mu}{h^2}\Big(D_0^{N_2} \otimes (6D_0^{N_1} - D_1^{N_1}) + D_1^{N_2} \otimes (-D_1^{N_1})\Big),
$$

$$
\begin{aligned}
G_2 &= D_0^{N_3} \otimes G_1 + \frac{\tilde{\alpha}\mu}{h^2}\Big(D_1^{N_3} \otimes (D_0^{N_2} \otimes (-D_0^{N_1})\Big) \\
L_{11}^{[3]} &= G_2 + \frac{\tilde{\alpha}(\mu+\lambda)}{h^2}\Big(D_0^{N_3} \otimes (D_0^{N_2} \otimes (2D_0^{N_1} - D_1^{N_1}))\Big), \\
L_{22}^{[3]} &= G_2 + \frac{\tilde{\alpha}(\mu+\lambda)}{h^2}\Big(D_0^{N_3} \otimes (D_0^{N_2} \otimes 2D_0^{N_1} + D_1^{N_2} \otimes (-D_0^{N_1})\Big), \\
L_{33}^{[3]} &= G_2 + \frac{\tilde{\alpha}(\mu+\lambda)}{h^2}\Big(D_0^{N_3} \otimes D_0^{N_2} \otimes 2D_0^{N_1} + D_1^{N_3} \otimes D_1^{N_2} \otimes (-D_0^{N_1})\Big),, \\
L_{12}^{[3]} &= \frac{\tilde{\alpha}(\mu+\lambda)}{4h^2}\Big(D_0^{N_3} \otimes D_2^{N_2} \otimes (-D_2^{N_1})\Big), \\
L_{13}^{[3]} &= \frac{\tilde{\alpha}(\mu+\lambda)}{4h^2}\Big(D_2^{N_3} \otimes D_0^{N_2} \otimes (-D_2^{N_1})\Big), \\
L_{23}^{[3]} &= \frac{\tilde{\alpha}(\mu+\lambda)}{4h^2}\Big(D_2^{N_3} \otimes (-D_2^{N_2}) \otimes D_0^{N_1}\Big).
\end{aligned}
$$

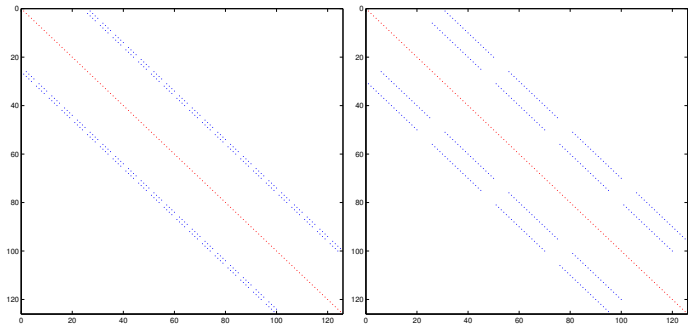
The definition of $M_h^{[3]}$ is analogous to the two-dimensional case. The discretization of the Laplacian Δ is given by G_2. For $L_{ii}^{[3]}$ second order derivatives in the respective dimensions are added. In contrast to $L_h^{[2]}$ there are three mixed derivatives. The matrices A_{ij} are constructed as in the two-dimensional case. The sparsity patterns for $M_h^{[2]}$ and $M_h^{[3]}$ are shown in figures 4.2 and 4.3. The contributions of A_{ij} are marked in red. Note that for all blocks except $M_{ii}^{[m]}$ the matrices A_{ij} are the only contributions to the diagonal of these blocks.

The condition number of L_h is in the order of $\mathcal{O}(h^{-2})$. The value λ plays an important role. It determines the degree of coupling and is, in terms of physics, related to the bulk modulus. Numerically large values of λ pose a problem since that leads a directional anisotropy in the diagonal blocks, $L_{ii}^{[m]}$, $1 \leq i \leq m$, and to large values in the off-diagonal blocks $L_{ij}^{[m]}$, $1 \leq i,j \leq m, i \neq j$. For our applications only moderate values of λ are necessary, because large values lack a proper physical motivation. Techniques for the theoretical investigation of the influence of λ, i.e. local Fourier analysis (LFA) for systems, are presented in [50]. A program called XLFA that performs LFA with various multigrid components is distributed with the book.

We have to deal with a type of anisotropy that is different from the one introduced by λ. This anisotropy is not global but local, the "jumping coefficients" introduced by A_h. All entries in the blocks A_{ij} are zero except for the diagonal. Furthermore we have $A_{ij} \neq 0$ only at places that correspond to regions where T_h changes locally, e.g. at edges. This leads to considerable jumps in the coefficients at these location, especially when $\tilde{\alpha}/h^2 \ll T_{h_{x_i}} T_{h_{x_j}}$, $i,j = 1,2,\ldots,m$. A theoretical analysis of the impact of these "jumping coefficients" is not possible due to



(a) $M_h^{[3]}$ (b) $M_{11}^{[3]} \& M_{22}^{[3]} \& M_{33}^{[3]}$ (c) $M_{12}^{[3]}$



(d) $M_{13}^{[3]}$ (e) $M_{21}^{[3]}$

Figure 4.3: Sparsity pattern for $M_h^{[3]}$, with $N_l = 5$, $l = 1, \ldots, 3$. The contributions of A_{ij} are marked red, all other entries are blue.

two main reasons. Firstly, standard techniques like local Fourier analysis cannot be used here. The power of local Fourier analysis stems from the fact that only one point has to be analyzed. That only works under the assumption that the operator is the same globally. When the variation is "reasonably smooth" one can still draw some conclusions from LFA. Obviously this is not the case here. A technique that takes into account the whole context, i.e. the whole operator, would be needed. That of course would be computationally expensive and amount to something like inverting the operator, e.g. compute the eigenvalues of M_h. Secondly if an efficient technique would exist we could only analyze multigrid behavior for one specific input because the structure and size of the anisotropies depends on the input data. It is well known that multigrid convergence degenerates in the presence of "jumping coefficients". In the following sections we will introduce multigrid components that deal with these problems and provide an analysis based on numerical experiments that demonstrates the influence of the coefficients and the improvements that can be achieved using different multigrid components.

4.2 Multigrid (inner iteration)

The main aim of this chapter is to propose a multigrid solver that solves the normal equation (3.15) efficiently. After a short introduction of the general concepts of multigrid solvers that can be skipped by the expert reader, we describe the components of this solver. We only consider geometric multigrid, and not algebraic multigrid (AMG) here. In some details, e.g. the definition of the grid transfer operators, we assume a vertex centered discretization as defined in section 4.1. For a more thorough introduction and further reading refer to [47].

4.2.1 The principles

Multigrid methods are based on two principles, the *smoothing principle* and the *coarsening principle*. The smoothing principle states that relaxation methods, like Gauss-Seidel, when applied to discrete elliptic problems have a strong smoothing effect on the error. That does not mean that the error becomes small, it just becomes smooth. In a short while we will explain what is meant by "smooth". The coarsening principle states that a smooth error term has a good approximation on a coarse grid. Obviously the computation on a coarse grid is far less expensive than a fine grid computation.

The error is a function of discrete variables $i_1, i_2, \ldots, i_m$. When we say "smooth" we mean that the high frequency components in Fourier expansion of the error become small. The Nyquist-Shannon sampling theorem tells us that only the low frequency components of the fine grid error are represented properly on a coarser grid. Specifically when we sample with step s and we have n discrete points in each dimension on the fine grid only the n/s smallest frequency components are represented properly on the coarse grid. All other components cause alias because they coincide with lower frequency components.

Coarsening does not only reduce the number of grid points for which we have to perform computations. Coarsening also leads to a spread in the spectrum by the same factor. Hence, the coarse error contains high frequency components for which we have a good smoother, the relaxation method.

So the basic idea is smooth the error on the fine grid, transfer the remaining error to a coarser grid, smooth it there, and so on. How this is done specifically is determined by the *multigrid components*. The components are

- smoother,

- coarsening strategy,
- coarse grid operator,
- transfer operator from fine to coarse grid (restriction),
- transfer operator from coarse to fine grid (prolongation, interpolation)
- cycle type.

These components are chosen in dependence to the problem to be solved. One can either tailor the components ones specific needs, or use components that might be not optimal with respect to efficiency, but solve a large class of problems robustly.

After this exposition of the key concepts we give a more formal definition of the two-grid cycle from which the multigrid cycle is build.

4.2.2 Two-grid cycle

Let $u_h^{(k)}$ be an approximation to the solution u_h of the discrete elliptic boundary value problem

$$M_h u_h = f_h. \tag{4.8}$$

The error of the approximation is

$$v_h^{(k)} := u_h - u_h^{(k)}$$

and the *defect* is denoted by

$$d_h^{(k)} := f_h - M_h u_h^{(k)}. \tag{4.9}$$

The defect equation

$$M_h v_h^{(k)} = d_h^{(k)} \tag{4.10}$$

is equivalent to the equation of original boundary value problem (4.8). We can define an iteration

$$u_h^{(k+1)} := u_h^{(k)} + \hat{v}_h^{(k)}. \tag{4.11}$$

Algorithm 1: Two grid cycle

1: **function** TWOGRIDCYCLE($M_h, f_h, u_h^{(k)}$)
2: ▷ M_h differential operator, f_h right hand side, $u_h^{(k)}$ start approximation
3: $d_h^{(k)} \leftarrow f_h - M_h u_h^{(k)}$ ▷ compute defect
4: $H \leftarrow 2h$
5: $d_H^{(k)} \leftarrow I_h^H d_h^{(k)}$ ▷ restrict defect
6: $\hat{v}_H^{(k)} =\leftarrow M_H^{-1} d_H^{(k)}$ ▷ solve on Ω_H^m
7: $\hat{v}_h^{(k)} \leftarrow I_H^h \hat{v}_H^{(k)}$ ▷ interpolate coarse grid correction
8: $u_h^{(k+1)} \leftarrow u_h^{(k)} + \hat{v}_h^{(k)}$ ▷ update solution
9: **end function**

by replacing M_h in (4.10) by an approximation $\hat{M}_h$, where $\hat{v}_h^{(k)}$ is the solution of

$$\hat{M}_h \hat{v}_h^{(k)} = d_h^{(k)}.$$

The key idea is to chose a coarse grid approximation M_H, $H > h$ as the approximation to M_h. The defect equation (4.10) then reads

$$M_H \hat{v}_H^{(k)} = d_H^{(k)}.$$

As noted above we need operators to transfer functions from fine to coarse grids (restriction) and vice versa (prolongation, interpolation). These operators are denoted by $I_{h_1}^{h_2}$ where the transfer is from a grid with discretization step h_1 to a grid with discretization step h_2, i.e.

$$I_{h_1}^{h_2} \text{ is a } \begin{cases} \text{restriction operator if } h_1 < h_2 \\ \text{prolongation (interpolation) operator if } h_1 > h_2 \end{cases}.$$

Thus the restricted defect and the prolongated coarse grid solution are

$$d_H^{(k)} = I_h^H d_h^{(k)} \qquad \text{and} \qquad \hat{v}_h^{(k)} = I_H^h \hat{v}_H^{(k)}.$$

A typical coarsening strategy is to double the discretization step, i.e. $H = 2h$. This strategy is often referred to as *standard coarsening* . When the update in the iteration (4.11) is computed from a coarse grid approximation of M_h the process is called *coarse grid correction scheme (CS)*. Algorithm 1 outlines all steps. Solving the boundary value problem on the next coarser grid still does not seem a much better prospect. To come from a two grid cycle to a *multigrid cycle* we have to apply the two-grid cycle recursively and use smoothing procedures to obtain smooth

defects for the next level.

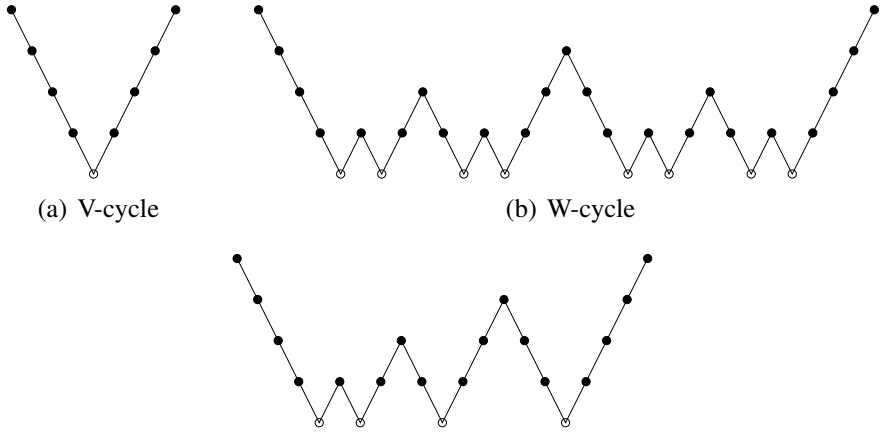
4.2.3 Multigrid cycle

From line 6 in algorithm 1 it is obvious that we need to recurse to a coarse grid where the direct exact solution of the defect equation is cheap.

Let Ω_h^m be as in (4.1). For the sake of simplicity of the presentation we assume that $N = 2^l + 1$. Using standard coarsening we can easily define a series of grids

$$(\Omega_{h_i}^m)_{i=1}^l = \{\Omega_{h_1}^m, \ldots, \Omega_{h_l}^m\}, \quad h_i = 1/2^i$$

where $\Omega_{h_l}^m$ is the finest grid and $\Omega_{h_1}^m$ corresponds to the coarsest grid with only one inner point. The index j is also called the *level*. Instead of only two levels like in the two-grid cycle there are now a maximum of l levels.



(a) V-cycle (b) W-cycle

(c) F-cycle

Figure 4.4: Common multigrid cycle types are illustrated. The open circle corresponds to the coarsest grid. The V- and the W-cycle have a constant cycle index of $\gamma = 1$ and $\gamma = 2$ respectively. In the F-cycle a combination is used, i.e. in the W-cycle the second recursive call is a V-cycle.

There are all kinds of different multigrid cycle types that are determined by the *cycle index* γ, the number of two-grid cycles applied on each level. The most common choices are a $\gamma = 1$ (*V-cycle*) or $\gamma = 2$ (*W-cycles*). A third frequently used option is the *F-cycle* which is obtained by varying γ depending on the level. The F-cycle is especially popular because it often has similar convergence rates as the W-cycle while being computationally cheaper and having a smaller memory footprint. The three mentioned cycle types are exemplified in figure 4.4 for a five

Algorithm 2: Multigrid algorithm

function MULTIGRID(M_h, f_h, u_h)
  ▷ M_h differential operator, f_h right hand side, u_h start approximation / solution
  if $h = \frac{1}{2}$ **then**
    $u_h \leftarrow (M_h f_h)^{-1}$
  else
    $u_h \leftarrow \mathcal{S}_h^{\nu_1}(M_h, f_h, u_h)$ ▷ S_h Gauss-Seidel relaxation, ν_1 presmoothing steps
    $d_h \leftarrow f_h - M_h u_h$ ▷ compute defect
    $H \leftarrow 2h$
    $d_H \leftarrow I_h^H d_h$ ▷ restrict defect
    $v_H \leftarrow$ MULTIGRID $(M_H, d_H, 0)$ ▷ recursive call to multigrid
    $v_h \leftarrow I_H^h v_H$ ▷ interpolate coarse grid correction
    $u_h \leftarrow u_h + v_h$ ▷ update solution
    $u_h \leftarrow \mathcal{S}_h^{\nu_2}(M_h, f_h, u_h)$ ▷ ν_2 postsmoothing step
  end if
end function

level case. It is easy to see that the F-cycle starts with $\gamma = 2$ and then continues with $\gamma = 1$ after the first of the two cycles.

A pseudo-code implementation of the multigrid cycle is given in algorithm 2. Note that we omit the iteration counter k and simply "overwrite" the existing values, since the old approximations are not needed anymore. The smoothing operator $\mathcal{S}_h^{\nu}$ can be any appropriate smoother. The natural number ν determines the number of iteration (smoothing steps) performed by the smoother. As we enter the recursion the right hand side f_h is the defect of the preceding level. This is only the multigrid cycle for coarse grid correction. There are other correction schemes, such as the *full approximation scheme (FAS)* for non-linear problems. Since we linearized our problem in the outer iteration the problem we have to solve with the multigrid (inner iteration) is a linear one and the use of the CS is sufficient. For linear problems CS and FAS yield the same result, FAS being the computationally more expensive scheme.

4.2.4 Restriction and prolongation operators

In this section we introduce some of the standard restriction and prolongation operators used in multigrid. The prolongation is also often referred to as interpolation instead. There are three standard restriction operators: injection, halfweighting, and fullweighting. The operator representations for the two-dimensional case in

the order of the above enumeration are

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad \frac{1}{10}\begin{bmatrix} 0 & 1 & 0 \\ 1 & 6 & 1 \\ 0 & 1 & 0 \end{bmatrix} \qquad \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}.$$

The operator is applied to the fine grid points which are also coarse grid points. These operators apply to standard coarsening with a equidistant vertex centered discretization. When other discretization schemes, e.g. cell-centered discretizations, are used the operators are different. The half- and fullweighting operators can be viewed as injection operators with prior smoothing (low-pass filtering) of the grid function. Remember the discussion about the high and low frequency components in section 4.2.1. In the light of that discussion the smoothing reduces the risk of alias. Injection can only be used when "enough" of the high frequency components have been eliminated by the smoother. Otherwise we see slower multigrid convergence or even divergence. In multigrid theory there is a rule that the sum of the orders of the restriction and interpolation operator should be larger than the order of the differential operator [26]. An interpolation operator has interpolation order k if it preserves polynomials of order $k-1$. Bilinear interpolation has order 2 and the transpose, the fullweighting operator, has also order 2. Injection has order 0 since the transpose does not even interpolate constant polynomials exactly. Thus for second order elliptic problems like the one considered here a combination of fullweighting and bilinear interpolation should be used.

The standard prolongation operator is

$$\left]\begin{matrix} \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ \frac{1}{2} & 1 & \frac{1}{2} \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \end{matrix}\right[.$$

The fact that it is a transfer operator from a coarse to a fine grid is indicated by the reversed parentheses. Again this operator applies to the equidistant vertex-centered discretization and is different for other discretizations. The coarse grid point that coincides with the fine grid point is left unchanged, the surrounding fine grid points receive a contribution depending on the neighborhood relation. This operator is the adjoint operator to the fullweighting operator, and with the equidistant grid is equivalent to bilinear interpolation.

In three-dimensional space we use trilinear interpolation as the prolongation operator. The transpose defines a three-dimensional version of the fullweighting

operator.

4.2.5 Multigrid components

As a starting point we use standard "out of the box" multigrid components. Since we deal with linear subproblems the correction scheme can be used. We use standard coarsening, i.e. $H = 2h$, and the coarse grid operator L_H is the compatible operator L_{2h}. Data is restricted by fullweighting and interpolated by bi/trilinear interpolation. The smoother is a pointwise coupled Gauss-Seidel relaxation method. Note that coupled relaxation is necessary because the centers of the off-diagonal stencils might be unequal to 0. We perform V-, F-, or W-cycles.

While this setup has good convergence rates for the equations from the gradient-descent method, convergence rates for the Gauss-Newton method will generally deteriorate due to the jumping coefficients (anisotropies) in the equations [47]. In chapter 5 numerical results that underline this fact are given. Next we introduce several changes in the standard components that improve the convergence of multigrid in the presence of jumping coefficients.

4.3 A multigrid method for anisotropic coefficients

In the last section the multigrid method has been introduced along with standard components. As noted earlier the standard components work well for smooth functions. It could for example be used for the normal equation (3.5) of the regularized gradient descent problem. In the presence of anisotropies multigrid convergence can deteriorate rapidly. This deterioration indicates that at least on of the principal principles of multigrid is violated.

The crucial building block of multigrid is the relation between the different grids. The three most important aspects are:

- The error has to be smoothed properly before it is transfered to the coarser grid.
- The restrictions and prolongation operators have to link fine and coarse grid in a proper way.
- The coarse grid operator has to be a good approximation of the fine grid operator.

With the modifications proposed in this section we touch all of these aspects. A different smoother, a coarse grid operator, an operator dependent prolongation operator, and a modification to the coarse grid correction step are proposed.

4.3.1 Coarse grid operator

Remember that the operator M_h consist of two parts: $\tilde{\alpha}L_h$ and $A_h = J_{\theta_h}^t J_{\theta_h}$. In case of L_h we use the compatible coarse grid operator $L_H = L_{2h}$. For A_h we have two options:

- restriction of the coefficients
- restriction of the template T_h, and computation of the coefficients on the coarse grid.

The latter option can be interpreted as a direct discretization of A on the coarse grid Ω_H. For the numerical experiments at the end of this paper we used the restriction of the coefficients, since it proved to give better results. Specifically we use full- or halfweighting with injection at the boundary.

A third alternative would be the Galerkin operator $I_h^H M_h I_H^h$. This would lead to full stencils, i.e. $3^m m^2$ coefficients for each grid point. All these coefficients have to be either stored or recomputed for each operator application. The variant proposed above needs far less coefficients to be stored. The compatible operator L_H is the same at all grid points and for A_H $m(m+1)/2$ coefficients are needed in each grid point. These storage issues are the reason why we avoided the use of the Galerkin operator.

4.3.2 Smoother

It is a well known fact that pointwise relaxation methods fail to smooth the error well for highly anisotropic problems. Generally strongly coupled components should be updated collectively [47]. In our problem the anisotropy depends on the input data and as a consequence location and direction of strongly coupled components are not known a priori. Thus, we use alternating Gauss-Seidel line relaxations, instead of pointwise relaxations. By line we mean a line on the discretized grid. In one smoothing step all lines along all dimensions are relaxed successively. Linewise relaxation results in a linear system of equations for each line. Note that when pointwise relaxations are used for a system of equations it is also necessary to solve a linear system when the equations are coupled. In our case this coupling

is introduced by the coefficients $T_{h_{x_i}}$, which leads to $m \times m$ linear system, where m is the number of equations, that has to be solved in each point relaxation.

When a whole line is relaxed simultaneously we do not only have to take into account the coupling between the elements of u_h in one point, but also the coupling between the grid points on the line. Due to the structure of the operator M_h the coefficient matrix of the resulting linear system is a block matrix $Q \in \mathbb{R}^{mn \times mn}$ with $m \times m$ blocks $Q_{ij} \in \mathbb{R}^{n \times n}$, $1 \leq i, j \leq m$, where n is the number of points on the line. The diagonal blocks are tridiagonal matrices. The off-diagonal blocks have non-zero entries, the coefficients $T_{h_{x_i}} T_{h_{x_j}}$, on the diagonal only. The structure of Q can be exploited to use a linear time solver to solve the system.

Definition 4.4 (Permutation matrix). *Let π be a permutation*

$$\pi : \{1, 2, \ldots, n\} \mapsto \{\pi(1), \pi(2), \ldots, \pi(n)\}$$

then the permutation matrix $P \in \mathbb{R}^{n \times n}$ is given by

$$P_\pi := \begin{bmatrix} e_{\pi(1)} \\ e_{\pi(2)} \\ \vdots \\ e_{\pi(n)} \end{bmatrix},$$

where e_i is the ith unit vector. For a matrix $Q \in \mathbb{R}^{n \times n}$, QP_π permutes the columns of Q, and $P_\pi Q$ permutes the rows of Q. The properties of permutation matrices include

$$\begin{aligned} P_\pi^t P_\pi &= I \\ P_\pi P_\sigma &= P_{\pi \circ \sigma} \\ P_\pi^{-1} &= P_{\pi^{-1}} \end{aligned}$$

With a proper permutations of the entries of Q a matrix with a single diagonal band of fixed width can be generated, which in turn allows for the efficient solution of the linear system.

Lemma 4.1. *Let $Q \in \mathbb{R}^{nm \times nm}$ be a block matrix with the tridiagonal submatrices $Q_{ij} \in \mathbb{R}^{n \times n}$, $1 \leq i, j \leq m$, and $b, x \in \mathbb{R}^{nm}$. Let the permutation π be defined as*

$$\begin{aligned} \pi : \{1, 2, \ldots, mn\} &\rightarrow \{\pi(1), \pi(2), \ldots, \pi(mn)\} \\ x &\mapsto ((xm - 1) \bmod mn) - 1 + \lfloor \frac{x - 1}{n} \rfloor. \end{aligned}$$

then

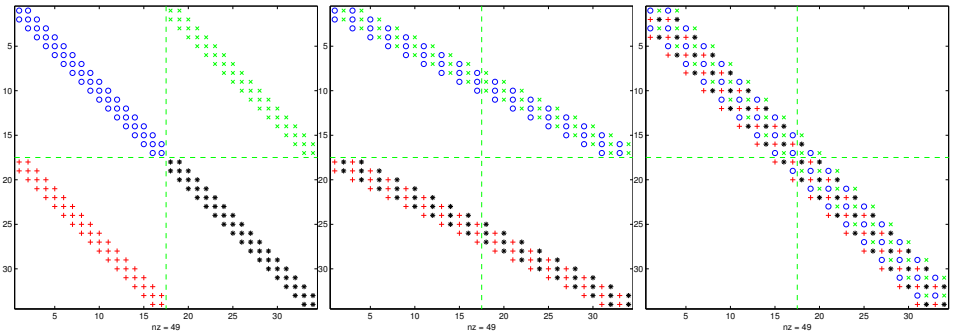
1. *solving*

$$Qx = b$$

is equivalent to solving

$$(P_\pi^t Q P_\pi) P_\pi^t x = P_\pi^t b,$$

2. *the matrix* $(P_\pi^t Q P_\pi)$ *has a single diagonal band of width* $3m$.



(a) Q (b) QP_π (c) $P_\pi^t(QP_\pi)$

Figure 4.5: Sparsity patterns for the matrix Q and the permuted version of it. Figure (a) shows the start configuration, (b) the configuration after the permutation of the columns, and (c) the result after the final permutation of the rows.

Proof. 1. This follows directly from the orthogonality of permutation matrices:

$$\begin{aligned} & (P_\pi^t Q P_\pi) P_\pi^t x &&= P_\pi^t b \\ \Leftrightarrow \quad & P_\pi (P_\pi^t Q P_\pi) P_\pi^t x &&= P_\pi P_\pi^t b \\ \Leftrightarrow \quad & Qx &&= b \end{aligned}$$

2. We do not give a thorough proof using matrix elements here, but rather explain the workings of the permutation π. The permutation π interleaves the columns and rows of the submatrices, such that the lth rows, or columns, of the blocks Q_{ij} are grouped together. It is easily verified that the term $((xm - 1) \bmod mn) - 1$ successively generates m sequences of the form

$1, m+1, 2m+1, \ldots, (n-1)m+1$. The term $\lfloor \frac{x-1}{n} \rfloor$ adds the block number minus 1, whereby interleaving the sequences.

The right side multiplication of Q with P_π groups the lth columns, $1 \leq l \leq n$ of Q_{ij} together, leading to m blocks of size $n \times nm$ with diagonals of width $3m$. The left multiplication with P_π^t exchanges the rows such that the lth rows of each block are grouped together. The final result is a $nm \times nm$ matrix with one diagonal band of width $3m$.

The rows of the solution vector x are exchanged ($P_\pi^t x$) to compensate for the columns permutation. The rows of the right hand side vector b are exchanged ($P_\pi^t b$) to compensate for the row permutation.

□

The permutation of the entries of Q is exemplified in figure 4.5. Figure 4.5(a) shows the initial configurations, figure 4.5(b) the configurations after the grouping of the columns, and figure 4.5(c) the final matrix. The system $(P_\pi^t Q P_\pi) P_\pi^t x = P_\pi^t b$ can be solved directly in time $\mathcal{O}(n)$ and space $\mathcal{O}(n)$ using a Band-Gauss solver. Hence, each point is relaxed in $\mathcal{O}(1)$ time, as it is the case for pointwise relaxation. Thus the asymptotic complexity of the multigrid is preserved. For $m = 3$ relaxation of whole planes is a common strategy. Usually a simple V-cycle with a 2D-multigrid solver is sufficient [47]. Note that the 2D-solver should also employ line relaxation. An alternative to plane relaxations are line relaxations in all directions which were a better tradeoff between convergence and computation time in our experiments.

Since this is essentially a Gauss-Seidel type relaxation method in the sense that already computed data is used for computations in the same sweep, the order in which the lines are relaxed can have an impact on the solution. From the abundance of possible orderings the most popular methods for pointwise Gauss-Seidel relaxations are lexicographical ordering and red-black ordering. For line relaxations there are also some typical orderings. There are two orthogonal strategies that can be used to construct a number of such orderings. Consider there is an inner loop and an outer loop. Then we can run through the directions in the outer loop and over the line indices in the inner loop or vice versa. The former strategy leads to so-called *alternating line relaxations*. When we first relax all lines with odd and then the ones with even line indices this is referred to as *zebratype line-relaxations*. In our calculations we iterate over the line indices in the inner loop and over the directions in the outer loop.

4.3.3 Operator dependent interpolation

The adjoint operator to the restriction by fullweighting is bilinear interpolation for $m = 2$, respectively trilinear interpolation for $m = 3$. The interpolated correction $v_h = I_H^h v_H$ solely depends on v_H. From the multigrid theory we know that this is a good choice when the coefficients of the operator, the solution, and the right side are smooth. Here, the operator M_h has jumping coefficients and does not fulfill these criteria. Depending on the discretization and knowledge about the underlying problem operator dependent interpolation operators can be derived [2, 14, 47, 51]. In the following we introduce a interpolation operator that is induced by the discretization of the differential operator M_h.

On the coarse grid we have solved

$$M_H v_H = \underbrace{I_h^H d_h}_{=r_H}.$$

Hence the interpolated correction v_h should be an approximate solution to the fine grid equation

$$M_h v_h = d_h.$$

In the following we replace h by a vector $\vec{h} = (h_1, h_2, \dots h_d)$ to indicate the discretization step in each dimension. We interpolate v_H by successively refining the grid along each dimension, i.e. in each refinement step the amount of grid points doubles. The missing grid points can then be computed by solving for all points on the new lines simultaneously. As in the case of the line relaxation that leads to band matrix that can be solved efficiently with a band matrix solver in $\mathcal{O}(n)$ steps and $\mathcal{O}(n)$ space where n is the number of grid points on the line. The corresponding equations are

$$M_{\vec{h}^i} v_{\vec{h}^i} = d_{\vec{h}^i}, \qquad \vec{h}^i = (h_1, \dots, h_i, H_{i+1}, \dots, H_d), \quad i = 1, \dots, d \tag{4.12}$$

The missing values for $M_{\vec{h}^i}$ and $d_{\vec{h}^i}$ can be obtained by applying restriction to the fine grid equivalents along all necessary directions:

$$\begin{aligned} M_{\vec{h}^i} &:= I_h^{\vec{h}^i} (J_{\theta_h}^t J_{\theta_h}) + \tilde{\alpha} L_{\vec{h}^i}, \\ d_{\vec{h}^i} &:= I_h^{\vec{h}^i} d_h. \end{aligned}$$

Figure 4.6 illustrates the procedure for the 2d-case. First the grid is interleaved in

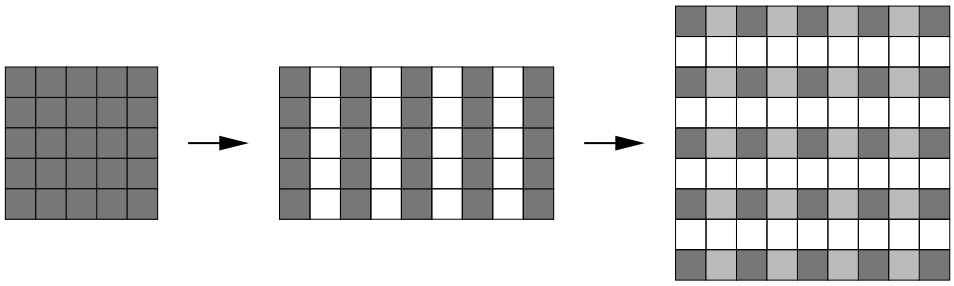


Figure 4.6: Illustration of the interpolation procedure for the 2d-case. First the grid is refined in one dimension. Then the unknown values [white boxes] are computed by solving a system of linear equations for each line. Afterwards the grid is refined in the other dimension and the missing values in the new grid are computed.

one dimension and the missing values (white boxes) are computed. Then the grid is refined in the other direction and the unknown values (white boxes) are computed from the values of the coarse grid solution (dark gray) and the values computed in the previous refinement step (light gray).

Note that the computational efficiency of the presented technique mainly stems from the fact that we have a simple coarsening and refinement strategy on a regular grid. The operator $M_{\vec{h}^i}$ and and the right side $d_{\vec{h}^i}$ can be easily computed. Furthermore the coefficient matrix of the resulting linear systems has a only m bands, where the width of the bands depends on the stencil width. Hence, as for the line relaxation, we can find fast solvers that preserve the asymptotic complexity of multigrid.

4.3.4 Operator dependent correction step

In this section we introduce a modification to the correction step. In the standard multigrid one computes $u_h \leftarrow u_h + I_H^h v_H = u_h + v_h$. Instead we multiply v_h by a factor τ that is optimal in a certain sense, i.e. line 12 in algorithm 2 (see page 28) is replaced by

$$u_h \leftarrow u_h + \tau v_h. \tag{4.13}$$

Here the parameter τ is chosen such that the underlying energy (3.8) becomes minimal, i.e. the following minimization problem

$$\min_{\tau\in\mathbb{R};\tau\geq 0} \Psi(\tau) \quad = \quad \min_{\tau\in\mathbb{R};\tau\geq 0} \left\{ \langle J_{E_\alpha}, (u+\tau v)\rangle + \frac{1}{2}\langle (J_\theta^t J_\theta + \tilde{\alpha} L)(u+\tau v), (u+\tau v)\rangle \right\}.$$

is to be solved. Ψ is a quadratic function and therefore it is differentiable. The necessary condition for a minimum is given by

$$\frac{\partial \Psi}{\partial \tau} = \langle J_{E_\alpha}, v\rangle + \langle (J_\theta^t J_\theta + \tilde{\alpha} L)(u + \tau v), v\rangle \stackrel{!}{=} 0.$$

With some elementary manipulations we yield:

$$\begin{aligned} \langle J_{E_\alpha}, v\rangle + \langle (J_\theta^t J_\theta + \tilde{\alpha} L)(u + \tau v), v\rangle &= 0 \\ \langle J_{E_\alpha} + (J_\theta^t J_\theta + \tilde{\alpha} L)u, v\rangle + \tau \langle (J_\theta^t J_\theta + \tilde{\alpha} L)v, v\rangle &= 0 \\ -\frac{\langle (J_\theta^t J_\theta + \tilde{\alpha} L)v, v\rangle}{\langle J_{E_\alpha} + (J_\theta^t J_\theta + \tilde{\alpha} L)u, v\rangle} &= \tau \end{aligned}$$

With $M = J_\theta^t J_\theta + \tilde{\alpha} L$ and $f = -J_{E_\alpha}$

$$\tau = -\frac{\langle Mu - f, v\rangle}{\langle Mv, v\rangle} = -\frac{\langle d(u), v\rangle}{\langle Mv, v\rangle} \tag{4.14}$$

fulfills the necessary condition. This minimization is analog to minimizing $||u^* - (u^{(k)} + \tau v)||_1$ where u^* is the exact solution. The value of τ can also be used to assess the "quality" of the interpolated solution with respect to the multigrid. In the optimal case this step should not be necessary and τ should always be 1. In section 5.1.3 this will be used to assess the influence of noise on the multigrid method proposed here.

4.4 Outer iteration

In the outer iteration (see equation (3.2) on page 8) the right hand side and in case of the Gauss-Newton method the new coefficients for the operator M_h have to be computed. The outer iteration is stopped if either the maximum number of steps ($k_{\max}$) is reached, the functional increases, or the change in two consecutive steps is smaller than some $\epsilon > 0$. Whether an update is actually performed depends on the parameter control strategy. Here we always use the procedure outlined in (3.12). The overall algorithm for the outer iteration is outlined in algorithm 3.

4.5 Multiresolution framework

In practice one often has to deal with large data sets and large deformations. Similar to the defect in multigrid the actual deformation can be decomposed into compo-

Algorithm 3: Pseudocode algorithm for the outer iteration. The iteration is stopped if either the maximum number of steps ($k_{\max}$) is reached, the functional increases, or the change in two consecutive steps is smaller than some $\epsilon > 0$.

function OUTERITERATION($T_h, R_h, u_h^{(0)}, k_{\max}$)
  $k \leftarrow 0$
  while ($k < k_{\max}$ AND $||u_h^{(k+1)} - u_h^{(k)}||_2^2 > \epsilon$ AND $E_\alpha(u_h^{(k)}) - E_\alpha(u_h^{(k+1)}) > 0$) **do**
    $f_h^{(k)} \leftarrow$ computeRHS($T_h, R_h, u_h^{(k)}$) ▷ compute right hand side
    $v_h \leftarrow M_h^{-1} f_h^{(k)}$ ▷ solve linear equation, e.g. with multigrid
    $k \leftarrow k + 1$
    $u_h^{(k+1)} \leftarrow u_h^{(k)} + v_h$ ▷ update may depend on parameter control strategy
  end while
end function

nents of different frequencies. Low frequency components correspond to smooth deformations and high frequency components to more local deformations. The goal must be to ensure that all these components are contained in the computed transformation. Note that here we make a semantical distinction between deformation and transformation. The deformation is an unknown quantity that is to be approximated by the computed transformation. Since we require the computed transformation to be smooth, we can expect local transformations to be small, while the more global smooth ones might be large. Actually we assume that the deformations are of that form by our modeling. Like the smoothers in the multigrid method the outer iteration is good at computing the high frequency (local) transformations, but meets problems when smooth large transformations have to be computed. The reason lies nonlinearity of the functional that is to be minimized. The linear model that results from linearization is only a good approximation in a limited region. Hence, the maximal length of the transformation vectors in the update computed in each step is limited. That not only leads to a large number of iterations for large deformations, but also to a danger of getting caught up in local minima.

The large transformations frequently apply to large objects in the data. Local details of different objects might look similar on fine grids. That in turn results in small forces in these regions because they are locally well registered. On coarser resolutions the details vanish, but the large objects are still visible. The low frequency components of the deformation are high frequency components on coarser resolutions. Hence, we can easily compute them there. The computed transformation can then be used as a start approximation at the next finer resolution. This strategy does not only aid robustness, but reduces the computational burden, too. The fact that we need less computations because we have less data is a nice side

Algorithm 4: Pseudocode algorithm for the multiresolution framework. The images are coarsened until some given minimal level $l_{\min}$ is reached. A solution is computed at that level, and interpolated to the next finer grid where the interpolated solution is used as a start approximation, and so on.

function MULTIRESOLUTION($T_{h_l}, R_{h_l}, u_{h_l}$,$l$)
    ▷ T_{h_l} template image, R_{h_l} reference image, u_{h_l} actual deformation (0 at
                                                                    ▷ start)
    if level== $l_{\min}$ **then**
        $u_{h_l} \leftarrow$OUTERITERATION $(T_{h_l}, R_{h_l}, u_{h_l}, k_{\max(l)})$
    else
        $T_{h_{l-1}} \leftarrow$ coarsen(T_{h_l})
        $R_{h_{l-1}} \leftarrow$ coarsen(R_{h_l})
        $u_{h_{l-1}} \leftarrow 0$
        MULTIRESOLUTION $(T_{h_{l-1}}, R_{h_{l-1}}, u_{h_{l-1}}$,$l-1)$
        $u_{h_l} \leftarrow$ interpolate($u_{h_{l-1}}$)
        OUTERITERATION $(T_{h_l}, R_{h_l}, u_{h_l}, k_{\max(l)})$
    end if
end function

effect of the multiresolution scheme. Its main purpose however is to avoid local minima.

More formally, we define a two resolution method that we apply recursively to obtain a multiresolution method. To transfer data from one resolution to another we can use the same transfer operators as is the multigrid algorithm. The images are restricted by means of fullweighting, and the solution is interpolated with the adjoint operator, i.e. bi(/tri)linear interpolation. In contrast to the multigrid we do not coarsen down to the smallest possible resolution. The size of the smallest image grid should be chosen with respect to the maximal expected transformation. The multiresolution approach used here is comparable to the *Full Multigrid Method (FMG)* known from multigrid applications (see e.g. [44]). The whole procedure is illustrated in algorithm 4.

CHAPTER 5

Numerical results

In this chapter we present numerical results for the multigrid (inner iteration), the outer iteration, and the multiresolution framework. For the multigrid h-independence, convergence rates, and the influence of noise are investigated. The part about the outer iteration deals with the effect of the number of inner iterations, the trust region topology, and the trust region control strategy. At the end the gradient descent and the Gauss-Newton method introduced in chapter 3 are compared. The chapter concludes with some observations about the multiresolution framework.

5.1 Multigrid

In this section we mainly focus on convergence rates for the multigrid, the inner iteration. For practical reasons the gray values in the images are scaled to $[0, 1]$, and all norms are divided by the number of data points. If not indicated otherwise

(a) Template (b) Reference (c) Transformation

Figure 5.1: Model problem: The template (a) is comprised of three nested squares with different gray values. In the reference (b) the squares are replaced by circles. The resulting transformation is shown in (c).

tests were performed with $\nu_1 = 2$, $\nu_2 = 1$, $\mu = 1$, and $\lambda = 1$. For the smoothers we used over-relaxation with $\omega = 1.3$. We have done F-cycles in all computations since that proved to be the best tradeoff between performance and convergence.

5.1.1 h-independence

One of the key properties of multigrid methods is that their convergence does not depend on the number of grid points. This property is also called h-independence. The data dependent anisotropic coefficients render a rigorous analysis impossible. Thus, we have to investigate h-independence empirically. To this end we determined the number of multigrid steps needed for the defect to drop below 10^{-8}. We cannot use a "real world" image here. In a "real world" image structures disappear when the image size is reduced, i.e. h is increased. Hence a wholly different problem, with a different type of anisotropies, would be solved for the various image sizes.

We used a synthetic images that are easily reproducible at different image resolu-

	$\tilde{\alpha} = 1$		$\tilde{\alpha} = 0.1$		$\tilde{\alpha} = 0.01$	
h	m	$\|\|d_h^m\|\|_2^2$	m	$\|\|d_h^m\|\|_2^2$	m	$\|\|d_h^m\|\|_2^2$
$1/128$	5	$1{,}748\,440 \cdot 10^{-10}$	6	$6{,}891\,06 \cdot 10^{-10}$	7	$7{,}020\,42 \cdot 10^{-09}$
$1/256$	5	$2{,}557\,180 \cdot 10^{-10}$	6	$1{,}896\,95 \cdot 10^{-09}$	8	$2{,}474\,33 \cdot 10^{-09}$
$1/512$	5	$2{,}777\,750 \cdot 10^{-10}$	6	$4{,}743\,39 \cdot 10^{-09}$	8	$4{,}068\,41 \cdot 10^{-09}$
$1/1024$	5	$3{,}189\,930 \cdot 10^{-10}$	6	$7{,}991\,38 \cdot 10^{-09}$	8	$5{,}672\,48 \cdot 10^{-09}$
$1/2048$	5	$3{,}711\,890 \cdot 10^{-10}$	7	$5{,}685\,02 \cdot 10^{-10}$	8	$7{,}144\,06 \cdot 10^{-09}$
	$\tilde{\alpha} = 0.001$		$\tilde{\alpha} = 0.0001$		$\tilde{\alpha} = 0.00001$	
h	m	$\|\|d_h^m\|\|_2^2$	m	$\|\|d_h^m\|\|_2^2$	m	$\|\|d_h^m\|\|_2^2$
$1/128$	7	$6{,}091\,81 \cdot 10^{-09}$	6	$4{,}544\,25 \cdot 10^{-09}$	5	$2{,}830\,160 \cdot 10^{-09}$
$1/256$	8	$3{,}020\,79 \cdot 10^{-09}$	7	$2{,}424\,38 \cdot 10^{-09}$	5	$6{,}455\,43 \cdot 10^{-09}$
$1/512$	8	$4{,}482\,29 \cdot 10^{-09}$	7	$3{,}470\,65 \cdot 10^{-09}$	5	$9{,}779\,99 \cdot 10^{-09}$
$1/1024$	8	$5{,}706\,06 \cdot 10^{-09}$	7	$4{,}424\,39 \cdot 10^{-09}$	6	$2{,}605\,61 \cdot 10^{-09}$
$1/2048$	8	$6{,}682\,02 \cdot 10^{-09}$	7	$5{,}294\,42 \cdot 10^{-09}$	6	$3{,}221\,24 \cdot 10^{-09}$

Table 5.1: Here the number of multigrid steps needed for the defect to drop below 10^{-8} for different values of h and $\tilde{\alpha}$ is displayed. The synthetic example in Figure 5.1 was used. The results show that the proposed multigrid converges independent of h.

tion. The template consists of three squares of decreasing size with different gray values that are all centered to the same spot (Figure 5.1(a)). The reference consists of discs of the same color and is a circle with the same colors and arrangement (Figure 5.1(b)). The results are displayed in table 5.1. Tests were performed for $\tilde{\alpha} = 10^{-i}$, $i = 0, \ldots, 5$, with line-relaxation, operator dependent interpolation, and operator dependent correction step. Similar results were obtained for other combinations.

5.1.2 Convergence rates

Convergence factors $\hat{q}^{(m)}$ where determined from the L_2-Norm of the defect using the formulas

$$q^{(m)} := \frac{||d_h^m||_2}{||d_h^{m-1}||_2}, \quad \hat{q}^{(m)} := \sqrt[m-m_0+1]{q^{(m)}q^{(m-1)}\dots q^{(m_0)}}, \tag{5.1}$$

where d_h^m is the defect after the m-th multigrid cycle [47]. Here we use $m_0 = 4$ and $m = 10$. The first three iterations are not included in the computation of the convergence rates, because the defect reduction in the first few steps is generally very good and does not reflect the asymptotic behavior.

$\tilde{\alpha}$	P/N/N	P/Y/N	P/N/Y	P/Y/Y	L/N/N	L/Y/N	L/N/Y	L/Y/Y
1	0.2410	0.1060	0.2365	0.0916	0.0276	0.0273	0.0248	0.0272
0.1	0.2454	0.1035	0.2420	0.1002	0.0374	0.0390	0.0372	0.0383
0.01	0.2445	0.1309	0.2418	0.1130	0.0713	0.0708	0.0528	0.0542
0.001	0.4450	0.4419	0.3241	0.3176	0.2254	0.2227	0.2083	0.1604
0.0001	0.5550	0.5538	0.5407	0.4571	0.3661	0.3561	0.3526	0.2842
0.00001	0.7202	0.7075	0.6991	0.6768	0.4418	0.4383	0.4401	0.4058
0.000001	0.7342	0.7238	0.7230	0.7082	0.4601	0.4538	0.4427	0.4287

Table 5.2: Convergence rates for different combinations of multigrid components, for a given starting $\tilde{\alpha}$. The first letter indicates whether pointwise (P) or linewise (L) relaxation has been used. The second letter indicates whether the operator dependent interpolation has been used (Y) or not (N). The third letter indicates whether the operator dependent correction has been used.

Computations are based on the registration of histological sections (512×512 pixels). The reference image has been distorted by an artificial transformation to obtain the template image (Figure 5.8, page 54).

Note that for the computation of the convergence only one step is performed in the outer iteration . Thus, we have $f(u^{(0)}) = -(J_\theta^t\theta + \alpha L u^{(0)})$, with $u^{(0)} \equiv 0$. As a consequence, only the value of $\tilde{\alpha}$ is important and not how it is composed. For subsequent outer iterations that is not the case due to the fact that the regularization on the right hand side then depends on the choice of α, i.e. one might observe different behavior for identical $\tilde{\alpha} = \alpha + \beta$ because the smoothness of the right hand side varies. For $\tilde{\alpha} = 0.01$ $||A||_\infty$ is approximately $\tilde{\alpha}||L||_\infty$. As $\tilde{\alpha}$ decreases the anisotropy increases and we generally expect worse convergence rates.

We tested many parameters for different combinations of multigrid components. We have selected some to illustrate the most important points. Pointwise relaxations is compared to linewise relaxation. For each relaxation type we

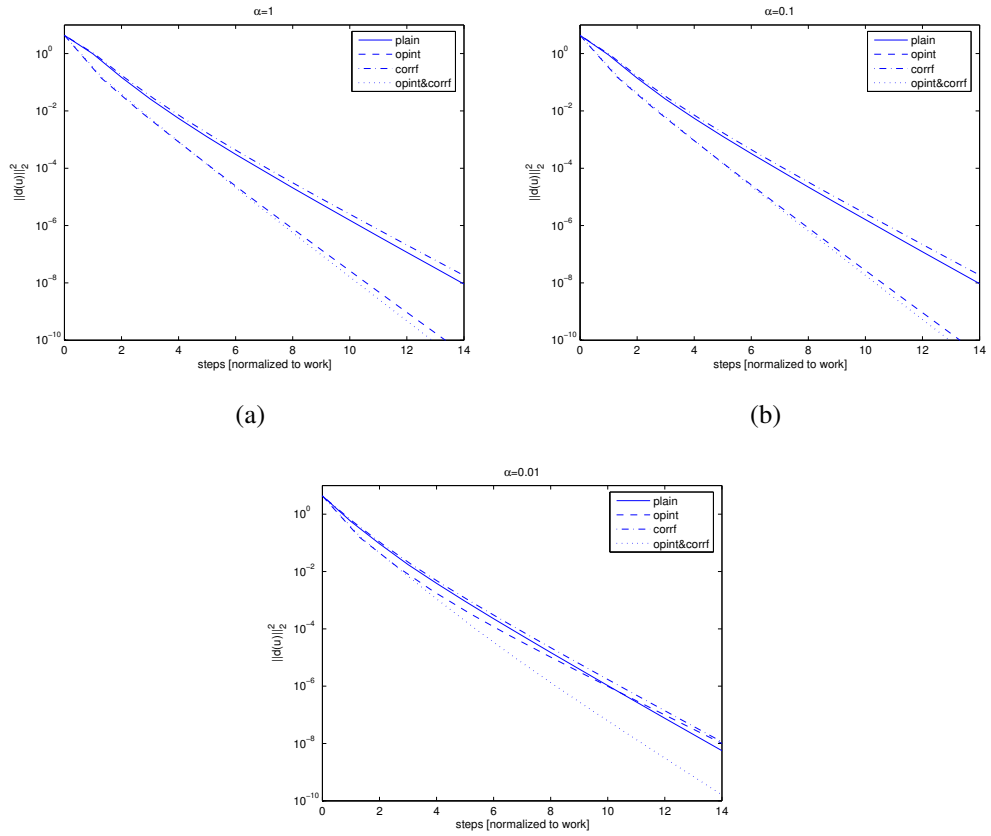


Figure 5.2: Convergence for pointwise relaxation with respect to the workload. Division by the cumulative weights of the multigrid components yields the actual number of steps.

present convergence rates with and without operator dependent interpolation (see section 4.3.3) and operator dependent correction step (see section 4.3.4). Operator dependent interpolation improves the convergence rates for moderate $\tilde{\alpha}$'s. With decreasing $\tilde{\alpha}$ this has to be combined with the operator dependent correction factor.

For a fair comparison the amount of additional computational effort has to be considered, too. To this end we assign a weight of 1.0 to the pointwise relaxation without any modifications. Linewise relaxation has a weight of 2.0. The weight of the operator dependent interpolation is 0.3 and that of the operator dependent correction factor is 0.05. These weights were determined based on timings of our implementation on a Pentium4. For different implementations or different architectures they might differ. We scaled the number of steps with these weights to make the defect reductions comparable. Figure 5.2 displays the defect reduction in the case of pointwise relaxation for $\tilde{\alpha} = 1, 0.1, 0.01$. A combination of operator dependent interpolation and correction factor yields the best performance. For

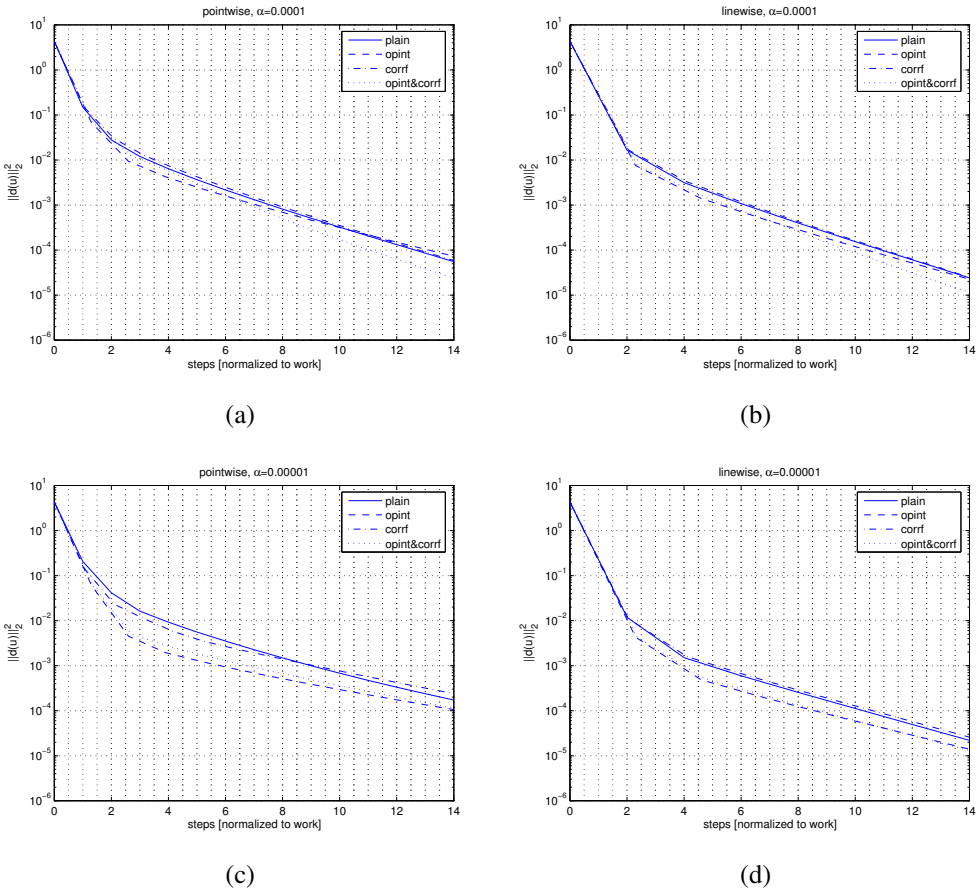


Figure 5.3: Comparison of the convergence with respect to the workload for small $\tilde{\alpha}$s (≤ 0.0001) between pointwise and linewise relaxation. With decreasing $\tilde{\alpha}$ linewise relaxation becomes superior to pointwise relaxation.

$\tilde{\alpha} = 1, 0.1$ operator dependent interpolation alone suffices, too. Figure 5.3 illustrates that linewise relaxation is superior to pointwise relaxation when the degree of anisotropy increases. The lower computational effort does no longer compensate for the lower convergence rates.

Note that the absolute convergence rates might differ for different input data, but the relative statements made above still hold.

5.1.3 Influence of noise

The term $J_\theta^t(u^{(k)})J_\theta(u^{(k)})$ depends on the transformed template $T^{(k)}$. Noise in the template image thus has an influence on the coefficients in the operator M_h. In the ideal case where images are free of noise, we have jumping coefficients at the edges in the images only. In the presence of noise we also have them in other places. The higher degree of local anisotropy is likely to affect the multigrid. A

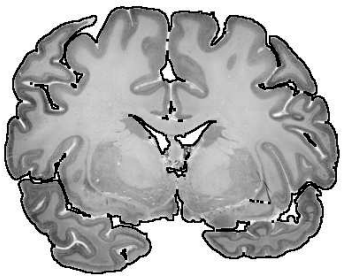
direct comparison of convergence rates is not possible, since the the right hand side, f, is affected, too.

With the operator dependent correction step introduced in section 4.3.4 we have another means to investigate the "quality" of the multigrid. In the ideal case the correction factor τ should be one. Any deviation indicates a problem in the transfer of information from one grid to the other, the approximation of the fine grid operator by the coarse grid operator, or the smoothing of the error. Usually it will not be one single effect and a separate analysis is not possible.

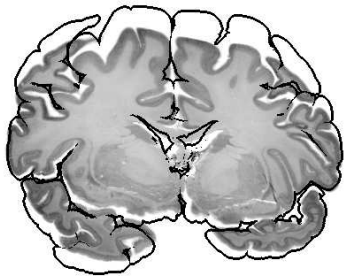
We used the following scheme to analyze the influence of noise. The reference image is a histological section of a human brain (Figure 5.4(a)). The deformed template with the contour of the reference is shown in figure 5.4(b)). We added gaussian and salt&pepper noise of various degrees $(0.0, 0.01, 0.05, 0.1, 0.5)$ to the template. In case of gaussian noise these values are the variance of the noise, and for the salt&pepper noise it is fraction of the affected pixels. Examples are shown in figure 5.4(c)- (d).

One outer iteration with ten V-cycles was performed for each image. The operator dependent correction factor τ was computed but not applied. Computations were done with both operator dependent interpolation on and off. The mean correction factor at each level is displayed in figure 5.4(e)- 5.4(f). The result for no noise (0.0) demonstrates the effect of the operator dependent interpolation. With operator dependent interpolation switched on the correction factors are close to one and vary only slightly. With added noise we see a significant increase in the correction factors when standard bilinear interpolation is used. Especially the coarser levels, i.e. the lower level numbers, are affected. When operator dependent interpolation is used the correction factors remain stable. This indicates that this type of interpolation provides a better link between the coarse and the fine grid, and explains why multigrid convergence is improved.

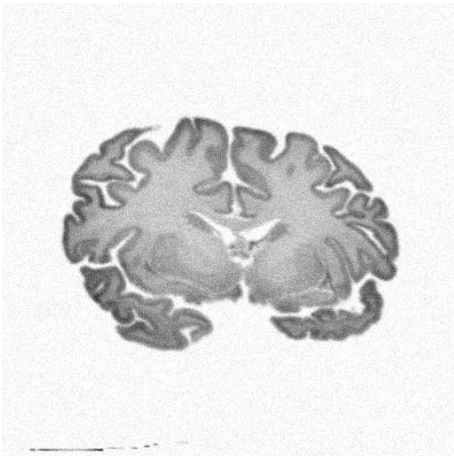
The influence of $J_\theta^t(u^{(k)})J_\theta(u^{(k)})$ on the registration is reduced by stronger regularization by means of α and β. Hence, regularization also reduces the influence of noise. In case of regularized gradient descent where the term $J_\theta^t(u^{(k)})J_\theta(u^{(k)})$ is not present, noise has no distinguishable effect on the multigrid. Though the right side f_h contains noise, the differential operator does not.



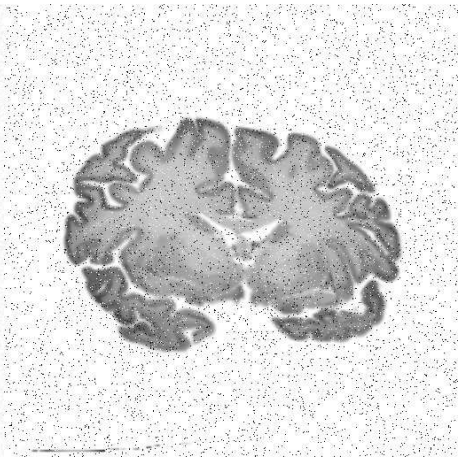
(a) Reference



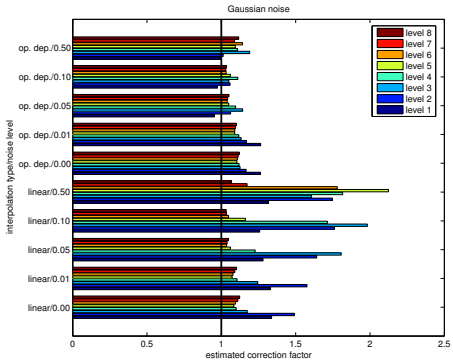
(b) Template



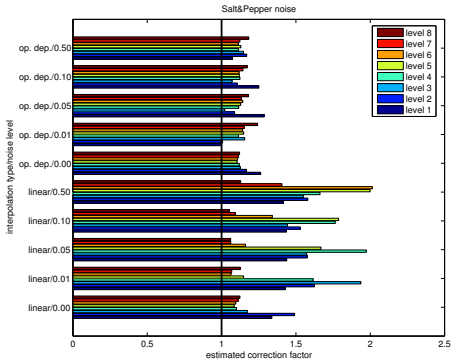
(c) Gaussian noise ($\sigma = 0.05$)



(d) Salt & Pepper noise (5%)



(e) Gaussian noise



(f) Salt & Pepper noise

Figure 5.4: Here the effect of noise on the multigrid is investigated. The operator dependent correction factor τ is used to assess the "quality". The factor is computed but not applied. In the optimal case it is one. Two types of noise, gaussian and salt&pepper noise are added to the template (b). Examples are shown in (c) and (d). The mean correction factors at each level of the multigrid, for different noise levels, with and without operator dependent interpolation, are displayed in (e) and (f). Ten V-cycles were used to compute the mean correction factor.

5.2 Outer iteration

In this section we investigate some aspects of the outer iteration. First we look at the effect of the number of multigrid cycles on the convergence in the outer iteration. Then we investigate two possible choices for the operator that determines the trust region topology. This is followed by comparison between the regularized gradient descent method (Section 3.1) and the regularized Gauss-Newton method (Section 3.2) follows.

5.2.1 Effect of the number of multigrid cycles

In the previous section we investigated the convergence of the inner iteration. The inner iteration is necessary because the direct solution of the normal equation is not feasible, due to the huge number of unknowns. It is also the computationally most expensive part of the whole registration algorithm. The solution from the inner iteration is just an approximation to the exact solution. The accuracy depends on the number of inner iterations (multigrid cycles). One would like to reduce the number of multigrid cycles to a minimum. Convergence in the outer iteration is measured by the decrease in the functional E_α (cf. equation (2.3)) which is composed of the image difference and the regularization term.

We investigated the influence of the number of inner iterations on the decrease of the functional E_α in the outer iteration. To this end we used the simple model problem that has already been used to show h-independence in section 5.1.1 (see figure 5.1). We did one to four F-cycles while not changing any of the other parameters and performed four steps in the outer iteration. The results are summarized in table 5.3. While there is a difference in the image distance and the value of E_α between one and two multigrid cycles, we do not see any substantial difference between two, three, and four multigrid cycles in these values. The defect at the end of the inner iteration decreases as expected with the number of multigrid cycles.

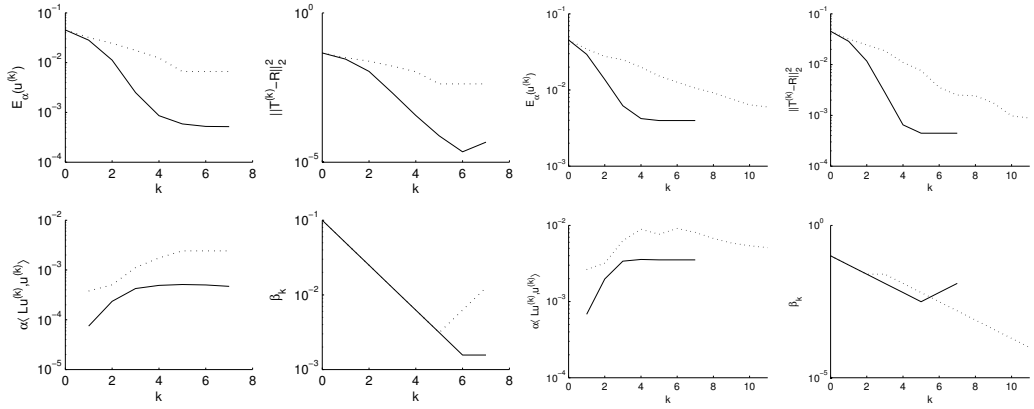
Of course we cannot conclude from this that we always get the best reduction in the outer iteration with just two inner iterations. But generally we can conclude that the approximation in the inner iteration does not have to be precise down to machine precision. The image is moved by one grid point if the length of u_h is h. Thus a change of say $h \cdot 10^{-4}$ in u_h leads to only very small changes in the image, and thus also in the image distance. We have to keep in mind that each update v is just a step into the direction of an approximated descent direction. Hence, we can also expect that small errors might be corrected in the next outer iteration.

k	1 MG-cycle		
	$\|\|T^{(k)} - R\|\|_2^2$	$E_\alpha(u^{(k)})$	$\|\|Mv - f(u^{(k)})\|\|_2^2$
0	$4{,}533\,800 \cdot 10^{-02}$	$4{,}533\,800 \cdot 10^{-02}$	$1{,}361\,880 \cdot 10^{-02}$
1	$2{,}127\,240 \cdot 10^{-02}$	$2{,}131\,000 \cdot 10^{-02}$	$8{,}496\,070 \cdot 10^{-03}$
2	$4{,}475\,590 \cdot 10^{-03}$	$4{,}557\,240 \cdot 10^{-03}$	$2{,}090\,040 \cdot 10^{-03}$
3	$9{,}085\,690 \cdot 10^{-04}$	$1{,}008\,210 \cdot 10^{-03}$	$1{,}142\,010 \cdot 10^{-03}$
4	$1{,}301\,490 \cdot 10^{-05}$	$1{,}146\,290 \cdot 10^{-04}$	$1{,}977\,190 \cdot 10^{-04}$
k	2 MG-cycles		
	$\|\|T^{(k)} - R\|\|_2^2$	$E_\alpha(u^{(k)})$	$\|\|Mv - f(u^{(k)})\|\|_2^2$
0	$4{,}533\,800 \cdot 10^{-02}$	$4{,}533\,800 \cdot 10^{-02}$	$1{,}913\,800 \cdot 10^{-03}$
1	$2{,}094\,060 \cdot 10^{-02}$	$2{,}097\,820 \cdot 10^{-02}$	$1{,}326\,570 \cdot 10^{-03}$
2	$4{,}466\,720 \cdot 10^{-03}$	$4{,}545\,920 \cdot 10^{-03}$	$3{,}054\,310 \cdot 10^{-04}$
3	$8{,}099\,220 \cdot 10^{-04}$	$9{,}064\,420 \cdot 10^{-04}$	$7{,}618\,570 \cdot 10^{-05}$
4	$4{,}650\,010 \cdot 10^{-06}$	$1{,}035\,440 \cdot 10^{-04}$	$2{,}617\,710 \cdot 10^{-05}$
k	3 MG-cycles		
	$\|\|T^{(k)} - R\|\|_2^2$	$E_\alpha(u^{(k)})$	$\|\|Mv - f(u^{(k)})\|\|_2^2$
0	$4{,}533\,800 \cdot 10^{-02}$	$4{,}533\,800 \cdot 10^{-02}$	$2{,}878\,430 \cdot 10^{-04}$
1	$2{,}090\,550 \cdot 10^{-02}$	$2{,}094\,300 \cdot 10^{-02}$	$2{,}925\,760 \cdot 10^{-04}$
2	$4{,}471\,130 \cdot 10^{-03}$	$4{,}550\,150 \cdot 10^{-03}$	$7{,}387\,920 \cdot 10^{-05}$
3	$8{,}056\,510 \cdot 10^{-04}$	$9{,}019\,490 \cdot 10^{-04}$	$1{,}574\,070 \cdot 10^{-05}$
4	$5{,}204\,780 \cdot 10^{-06}$	$1{,}037\,660 \cdot 10^{-04}$	$7{,}185\,780 \cdot 10^{-06}$
k	4 MG-cycles		
	$\|\|T^{(k)} - R\|\|_2^2$	$E_\alpha(u^{(k)})$	$\|\|Mv - f(u^{(k)})\|\|_2^2$
0	$4{,}533\,800 \cdot 10^{-02}$	$4{,}533\,800 \cdot 10^{-02}$	$4{,}487\,280 \cdot 10^{-05}$
1	$2{,}090\,060 \cdot 10^{-02}$	$2{,}093\,820 \cdot 10^{-02}$	$7{,}223\,900 \cdot 10^{-05}$
2	$4{,}472\,500 \cdot 10^{-03}$	$4{,}551\,500 \cdot 10^{-03}$	$2{,}072\,070 \cdot 10^{-05}$
3	$8{,}061\,560 \cdot 10^{-04}$	$9{,}024\,300 \cdot 10^{-04}$	$5{,}251\,340 \cdot 10^{-06}$
4	$5{,}596\,170 \cdot 10^{-06}$	$1{,}041\,340 \cdot 10^{-04}$	$2{,}768\,670 \cdot 10^{-06}$

Table 5.3: Normalized image distance, value of the functional E_α, and and norm of the defect at the end of the inner iteration for the model problem (see figure 5.1) for k outer iterations. The number of MG-cycles (inner iterations) is varied from one to four. While the defect decreases with each additional MG-cycle there is no substantial change in the image distance after two inner iterations.

5.2.2 Trust region topology

In section 3.3.2 we argued that it makes sense to use the same operator for the trust region as is already used for the regularization of the problem ($B := L$). In figure 5.5 we compare results for the outer iteration for two different choices of B: the Navier-Lamé operator L and the identity I. The values of $E_\alpha(u^{(k)})$, $||T^{(k)} - R||_2^2$, $\langle Lu^{(k)}, u^{(k)}\rangle$, and β_k are displayed. The curves for L are solid and the curves for I are dotted. We used the model problem from figure 5.1 for the computations. The outer iteration stops when the change in E_α is smaller than



(a) $\alpha = 10^{-3}, \beta_0 = 10^{-1}$ (b) $\alpha = 10^{-2}, \beta_0 = 10^{-1}$

Figure 5.5: Comparison of two different operators for the trust region topology. The value of the overall functional $E_\alpha(u^{(k)})$, the image difference $||T^{(k)} - R||_2^2$, the regularization term $\langle Lu^{(k)}, u^{(k)}\rangle$, and the trust region parameter β_k are plotted against the number of iterations. The dotted lines correspond to the results for I, the solid lines to the results for L. In (a) we use $\alpha = 10^{-3}$ and in (b) we use $\alpha = 10^{-2}$. The parameter β_0 is 10^{-1} in both cases.

some $\epsilon > 0$, when E_α increases, or when the update has been rejected three times. Generally the distance term $||T^{(k)} - R||_2^2$ decreases whereas the regularization term $\langle Lu^{(k)}, u^{(k)}\rangle$ increases during the iteration. The data on $\langle Lu^{(k)}, u^{(k)}\rangle$ in figure 5.5 indicates what we have already argued. The updates computed with I are less "elastic" and thus the increase in the penalty term $\langle Lu^{(k)}, u^{(k)}\rangle$ is larger than for L. The new aspect is that the decrease in the image distance is less, too. Thus the slower convergence of E_α cannot be attributed to some scaling effect in β due to the use of different operators, but is an effect of the choice of the trust region topology.

The strength of this effect depends on the ratio α/β_k. In figure 5.5(a) the iteration for $B = I$ is stopped because the update to the solution has been rejected three times. The increase of β that goes along with the rejection did not produce

acceptable solutions. The final value of E_α differs by a factor of 10. This is the result of both a larger penalty term and a slower decrease of the image distance.

In contrast for the choice of the parameters in figure 5.5(b) the outer iteration is able to "recover". Here α is larger and thus the role of B is diminished. The final decrease in $\langle Lu^{(k)}, u^{(k)} \rangle$ can be attributed to the fact that at that point the equation is dominated by α and β only plays a minor role. Nevertheless the overall convergence is considerably slower.

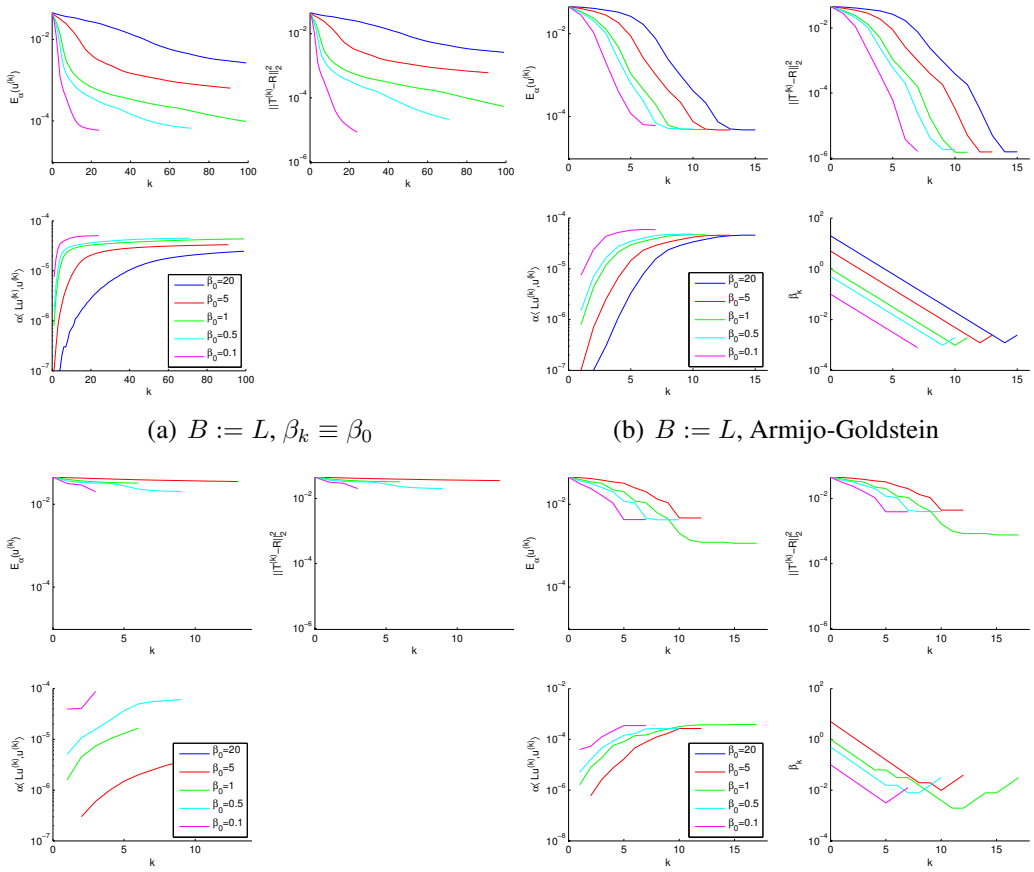
These examples show that with $B = L$ the outer iteration is more robust and the choice of β_0 is less difficult. It should be noted that multigrid convergence for $B = I$ is better especially when $\alpha \ll \beta$. But these are also the situations where we see the described effects in the outer iteration which outweigh the advantage of a faster convergence in the inner iteration by far. Additionally we have already demonstrated above that we do not need high accuracy of the inner iteration.

These are only results for the regularized Gauss-Newton method and they cannot be transfered to the regularized gradient descent method. The gradient descent methods lacks the term $J_\theta^t(u^{(k)})J_\theta(u^{(k)})$ which plays an important role in the Gauss-Newton method.

5.2.3 Trust region control strategy

Not only the topology of the trust region is important, but also the control of the parameter β_k. If β_k would remain constant the choice of β_0 would play a very important role for the outcome of the outer iteration. When $(\alpha + \beta_k)||L||_\infty \gg ||J_\theta^t(u^{(k)})J_\theta(u^{(k)})||_\infty$ the method behaves more like a gradient descent method, because the influence of $J_\theta^t(u^{(k)})J_\theta(u^{(k)})$ is small. As β_k becomes smaller the influence of $J_\theta^t(u^{(k)})J_\theta(u^{(k)})$ increases and the method behaves more like a Newton-type method. Gradient descent methods are globally convergent methods, with the disadvantage that convergence can be very slow. Newton-type methods are locally convergent methods, i.e. we can expect fast convergence near the solution. The downside is that when we are far off the solution the method might not converge at all. With an adaptive trust region strategy we seek to get the best of both worlds. The stability and global convergence of the gradient descent, and as we get close to the minimum the fast local convergence properties of the Newton-type method.

To analyze this behavior we used the model problem from figure 5.1 and performed registration on one resolution, with varying values for β_0. We used $B := L$ as well as $B := I$ and held β_k constant or used the Armijo-Goldstein rule (see equation (3.12). The results are displayed in figure 5.6. In 5.6(a) we see that when



(a) $B := L$, $\beta_k \equiv \beta_0$ (b) $B := L$, Armijo-Goldstein

(c) $B := I$, $\beta_k \equiv \beta_0$ (d) $B := I$, Armijo-Goldstein

Figure 5.6: Here we show results for the convergence of the outer iteration for the regularized Gauss-Newton method in dependence on the choice of β_0 and the parameter optimizations strategy. For the computations the model problem from figure 5.1 was used. The left column shows results for the case where β_k is kept constant. The right column shows results for the case where the Armijo-Goldstein rule is applied. In the upper row $B := L$ and in the lower row $B := I$.

β_k is held constant, the initial choice of β_0 has a tremendous influence on the result. For large value of β_0 like 20 and 5 convergence in the outer iteration is extremely slow. The trust region is very small and hence the step taken in each iteration is very small, too. After hundreds of iterations the outer iteration should still converge to a similar solution as for $\beta_0 = 1, 0.5, 0.1$.

When the Armijo-Goldstein rule is used to adapt the parameter we observe almost identical results for all start parameter choices (see figure 5.6(b)). Actually if we would shift the curves by the difference in the overall number of iterations the curves would look almost identical after the adaption phase in the beginning. We can conclude that a more conservative choice of β_0 costs us only a few iterations and does not determine the outcome and convergence speed to the extreme extend

it does when the parameter is fixed.

For the test with $B := I$ it can be said that the results for fixed β_k are extremely bad. The effect of the choice of the trust region topology (see also section 5.2.2) and the problems with the fixed parameter seem to combine. Note that the first row in all graphs is scaled to the same range on the y-axis. When the parameter is adapted the behavior is better, but the results are not consistent for all starting values. The result for $\beta_0 = 1$ is differs significantly from the ones for the other values.

We performed the same calculations for the gradient descent method. The results are shown in figure 5.7. The same starting values have been used, but we cannot expect similar results. Apart from the fact that convergence is expected to be slower, the equation is different and thus the influence of β_k might differ, too. The effect of the Armijo-Goldstein rule is similar. For $B := L$ the starting value does not play an important role, and for $B := I$ the results are not consistent. The overall convergence is, as expected, slower. The method seems to be more sensitive to the choice of β_k when $B := L$ which is indicated by the frequent adaption of the parameter. The band in which β_k is chosen is narrow. For $B := I$ that is not the case.

In the next section we discuss some of the further differences between the gradient descent and Gauss-Newton method on the basis of some registration results. Before we proceed we shortly discuss a strategy for a good initial guess of the start parameter β_0.

The choice of the start parameter β_0

With the choice of $B := L$ we can derive a strategy for an initial guess of β_0. The problem is that if we chose β_0 to large the influence of $J^t_{\theta_h}(u^{(0)})J_{\theta_h}(u^{(0)})$ on the minimization process is too small. Hence the information brought into the registration process by the term $J^t_{\theta_h}(u^{(0)})J_{\theta_h}(u^{(0)})$ is not utilized properly. When β_0 is too small the operator $M_h^{(0)}$ may be ill-conditioned depending on the choice of α. We show how β_0 can be chosen such that we obtain an upper bound for the condition of $M_h^{(0)}$:

$$\begin{aligned} \text{cond}_2(M_h^{(0)}) &\leq \frac{\tilde{\alpha}_0\lambda_{\max(L_h)} + ||J^t_{\theta_h}(u^{(0)})J_{\theta_h}(u^{(0)})||_2}{\tilde{\alpha}_0\lambda_{\min(L_h)}} \\ &= \text{cond}_2(L_h) + \frac{||J^t_{\theta_h}(u^{(0)})J_{\theta_h}(u^{(0)})||_2}{\tilde{\alpha}_0\lambda_{\min(L_h)}}. \end{aligned}$$

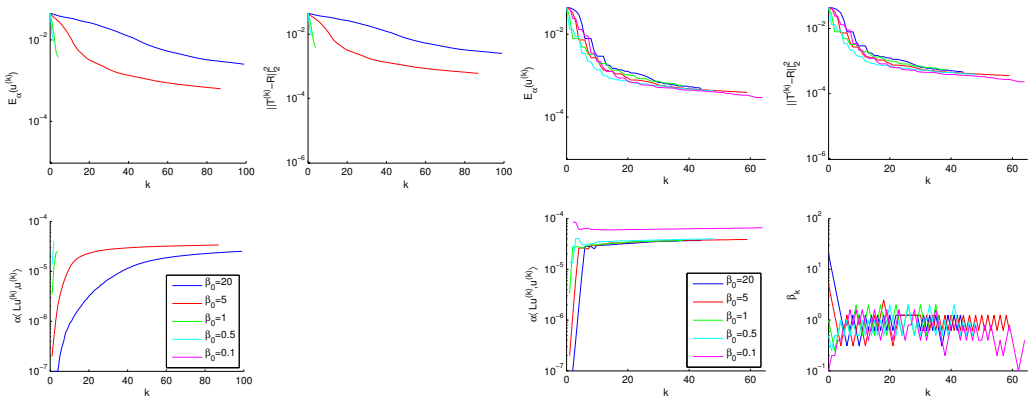
When we chose

$$\tilde{\alpha}_0 \geq \frac{||J^t_{\theta_h}(u^{(0)})J_{\theta_h}(u^{(0)})||_2}{||L_h||_2} \quad \Rightarrow \quad \beta_0 \geq \frac{||J^t_{\theta_h}(u^{(0)})J_{\theta_h}(u^{(0)})||_2}{||L_h||_2} - \alpha$$

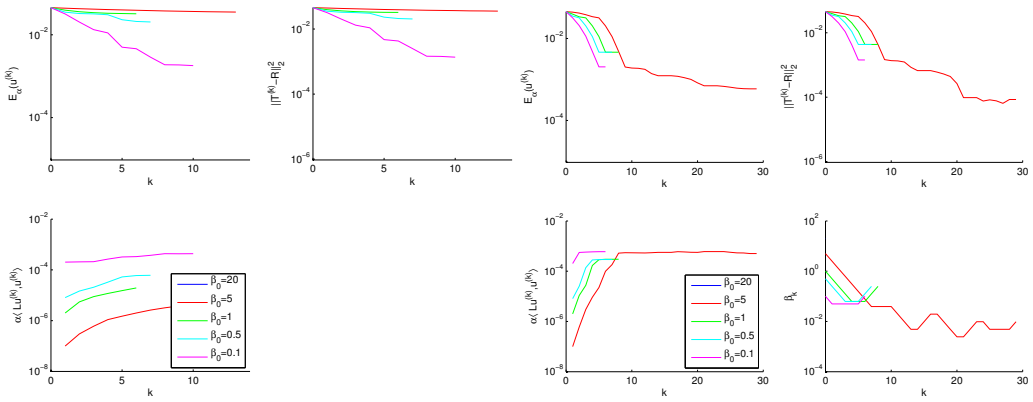
we obtain an upper bound for the condition of $M_h^{(0)}$,

$$\text{cond}_2(M_h^{(0)}) \leq 2\,\text{cond}_2(L_h).$$

In the application we can approximate $||\cdot||_2$ by $||\cdot||_\infty$ due to the fact the matrices are sparse.



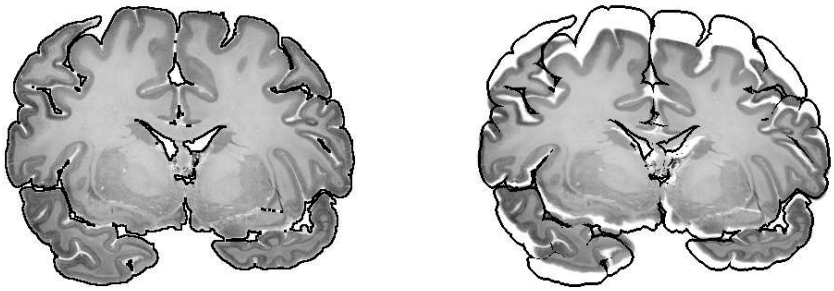
(a) $B := L$, $\beta_k \equiv \beta_0$ (b) $B := L$, Armijo-Goldstein



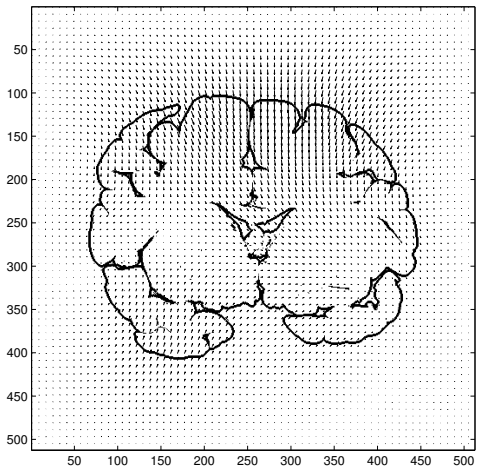
(c) $B := I$, $\beta_k \equiv \beta_0$ (d) $B := I$, Armijo-Goldstein

Figure 5.7: Here we show results for the convergence of the outer iteration for the regularized gradient descent method in dependence on the choice of β_0 and the parameter optimizations strategy. For the computations the model problem from figure 5.1 was used. The left column shows results for the case where β_k is kept constant. The right column shows results for the case where the Armijo-Goldstein rule is applied. In the upper row $B := L$ and in the lower row $B := I$.

5.2.4 Comparison of gradient descent and Gauss-Newton approach



(a) Reference (b) Template

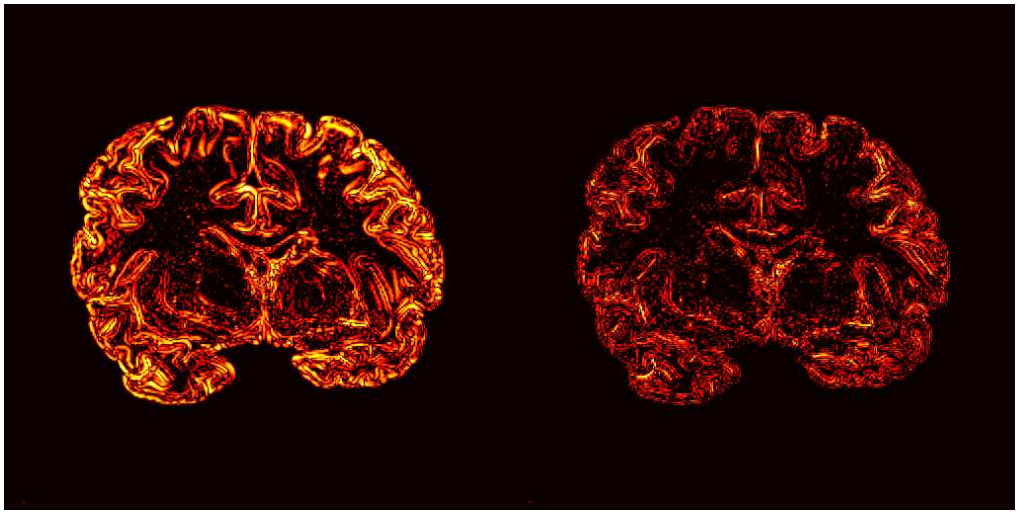


(c) Transformation

Figure 5.8: Example of registration of two histological sections ($m = 2$). The reference section (a) was deformed by an artificial transformation to obtain the template image (b). The calculated transformation is shown in (c).

In this section we to put forth some arguments in favor of the use of the regularized Gauss-Newton method. The normal equation of the regularized steepest descent method is easier to solve and therefore the cost of the inner iteration is less. The standard multigrid solver introduced in section 4.2.1 would suffice. We will demonstrate that convergence in the outer iteration is faster for the Newton-type method. Still one might argue that this advantage is nullified by the more expensive inner iteration. We will show that computation time is not the only important

aspect, and that the gradient descent does not yield satisfactory results in some situations. Theoretically after a large number of step both optimization strategies should produce the same result, i.e. if we neglect local minima and machine precision. Since both methods differ in their descent direction they take different paths towards the result. That means they could stop at different points when there is no "good" descent direction in the vicinity, i.e. we got trapped in a local minimum or the change in the functional is below a chosen threshold. The minimization process also stops when the increase in the regularization penalty exceeds the decrease in the distance functional. Due to the fact that the solutions computed in each step of the outer iteration are different for both methods the ratio between decrease in the distance function and increase in the regularization penalty is likely to be different, too. When the trust region parameter $\beta \to \infty$ both methods become equal since $||J_\theta^t J_\theta||((\alpha + \beta)||L||)^{-1} \to 0$, but then $||v|| \to 0$, too. That means for large β both methods are more likely to deliver the same result, but the number of outer iterations needed increases. Furthermore the information from $J_\theta^t J_\theta$ is not utilized the way it could be.



(a) (b)

Figure 5.9: In (a) the logarithm of the final difference after registration of the histological section from figure 5.8 with the gradient descent approach is shown. Figure (b) displays the logarithm of the difference after registration with the Gauss-Newton approach. Dark colors indicate small differences where as light colors indicate large differences. The result shows that the Gauss-Newton approach leads to more precise results, especially at edges with low contrast.

For the two-dimensional case we use a "real world" example, a histological section of a human brain. The section is colored for cell bodies and has been

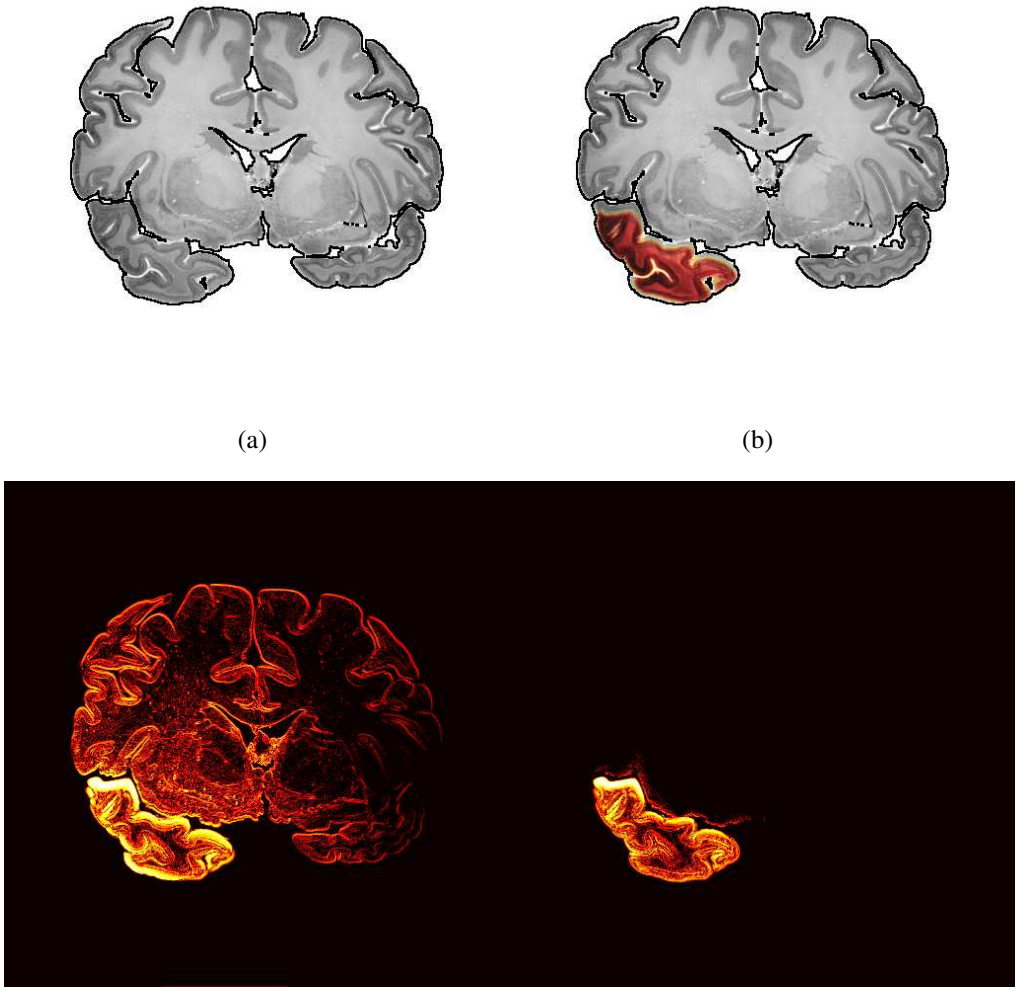
digitized with a flatbed scanner. The image size is 512×512. The reference is the original section (Figure 5.8(a)). The template is a artificially deformed version of the same section (Figure 5.8(b)). The contour of the reference is shown as a black line for orientation. The computed transformation is shown as a vector field in figure 5.8(c).

We performed registration with both methods, the regularized gradient descent and the regularized Gauss-Newton method. The final difference between the reference and transformed template is visualized in figure 5.9. The results for the gradient descent method is shown in figure 5.9(a), and the result for the Gauss-Newton method is shown in figure 5.9(b). To be able to see also minimal differences the logarithm of the pixelwise image difference is shown. Dark colors indicate small differences whereas light colors indicate large differences. Both images are scaled to the same maximal value.

The registration with the Gauss-Newton method is more precise. Compared to the gradient descent method the differences at the image edges are minimal. The large image errors for the gradient descent approach can especially be found in regions where the contrast is low, i.e. the gradient is weak. Information about the template image is only contained in the right side f_h in form of the image gradient for the gradient descent method. The image gradient drives the registration and the large gradients dominate the process. Thus edges with high gradient values are matched first and the low gradient edges are matched later in the process. The equations of the Gauss-Newton method incorporate information from the template image into the differential operator. This allows for good registration results in regions with low gradients and also more local transformations.

To demonstrate this use of local information we use a different template for the histological section example. Only a small part of the reference, one part on the left side, was transformed. In the three-dimensional case this part is attached to the rest of the brain, but when the sections are cut it might be torn off. This situation is simulated here. The reference is shown in figure 5.10(a). The same section with the deformed part colored, the template, is shown in figure 5.10(b). One iteration was performed. The logarithm of the image difference for the gradient descent method is shown in figure 5.10(c) and for the Gauss-Newton method in figure 5.10(d).

The gradient descent method results in a globally smooth transformation. That results in image differences in parts of the image that are actually identical. For the Gauss-Newton method we see a local transformation and a stronger reduction of the image difference after only one step. The transformations of the region in question are displayed in figure 5.11. After just one iteration we see a very

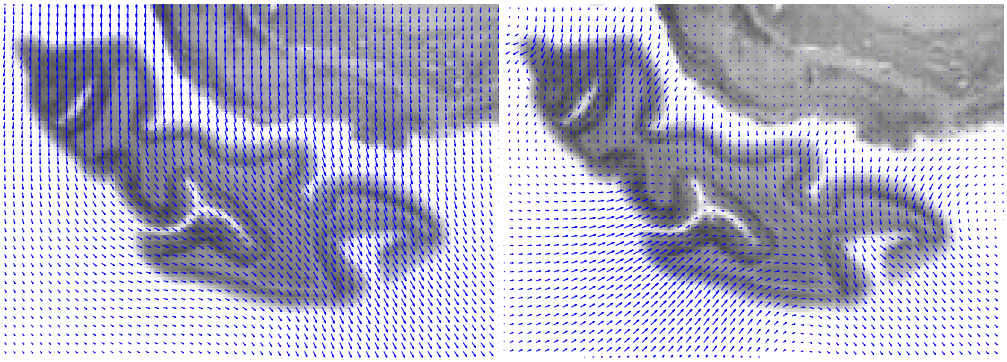


(a) (b)

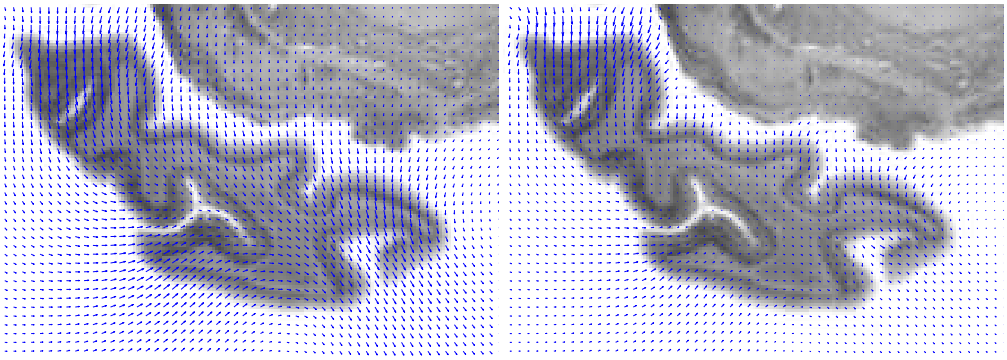
(c) (d)

Figure 5.10: The reference is shown in (a). The template differs from the reference only in the colored overlayed part in (b). The contour of the reference is indicated by a black line in both (a) and (b). In figure (c) the logarithm of the difference after one iteration with the regularized gradient descent approach is shown. In figure (d) the same is shown for the regularized Gauss-Newton method. Dark colors indicate small differences where as light colors indicate large differences. The result shows that the Newton-type method allows for more local transformations.

smooth transformation for the gradient descent method that extends into large parts of the identical portions of the template and reference image. In the Gauss-Newton method we already see some directions that are also present in the final transformation in both methods after just one iteration. The length of the transformation vectors declines rapidly in the direction of the not transformed parts of the histo-



(a) Gradient descent after one iteration (b) Gauss-Newton after one iteration



(c) Gradient descent, final result (d) Gauss-Newton, final result

Figure 5.11: Here a part of the transformations computed for the example in figure 5.10 are shown. The upper row shows the transformation after one iteration. The bottom row shows the final transformation. For the gradient descent the transformation extends into the part of the template section that is identical to the image. The results for the Gauss-Newton method are more localized. The difference is especially obvious after just one iteration.

logical section, whereas it is smooth in the directions where no additional image information is present, i.e. the white parts of the image. In case of the gradient descent the subsequent iterations correct the transformation. The image difference in the parts that are incorrectly moved results in forces in those regions in the subsequent iterations. Still the final result extends farther into regions it should not than with the Gauss-Newton method.

We conclude this section with 3D example from an application in neuroscience research. One would like to detect volume changes in MRI data of one individual over time. In case of neurodegenerative diseases volume change can be used to track and quantify the degenerative processes. The subjects are scanned multiple times in intervals of a couple of months. To estimate the change in volume the respective MRI data sets are registered to each other and the volume change in each voxel is determined from the transformation. One would like to not have to segment

the data prior to registration. By segmenting we mean removing everything that is not brain, i.e. skull, eyes, nerves, dura mater, etc.. The changes we would like to detect are very subtle. Hence an automatic segmentation is not sufficient and manual correction is needed. That takes both time and is observer dependent. The problem with not segmenting the brain is that some of the structures we would usually remove, especially the bones, have high gray value intensities. The gradient descent approaches mainly focus on the high gradients, yet we would like to detect changes in the brain and not in the position of the skull with respect to the brain, the eyes, or cavities. Hence gradient descent methods cannot be used in this context and we have to resort to the Gauss-Newton method.

We illustrate the points made above on a three-dimensional example. To avoid having to publish yet unpublished patient data we generated our own data set from two MRI data sets of a healthy control taken within a period of three months. To simulate a degeneration we took a segmented brain, eroded it, and placed it in the other data set. Registration was performed with both the gradient descent and the Gauss-Newton method. In figures 5.12- 5.15 renderings of the data with both transformation vectors and contour lines of the volume change are displayed for both methods. The vectors are projected on two orthogonal planes of the image data set. The length of the vectors is constant and the color indicates the magnitude. The colormap goes from blue over green, yellow, and orange, to red. All vectors with length greater than 0.2 of the voxel size are red. The contour lines are computed on the three-dimensional volume change data set computed from the transformation. The contours correspond to volume changes in the range of $\pm 5\%$. The colormap ranges from red to white for volume decrease and from white to red for volume increase.

For the gradient descent method (Figure 5.12) the transformation field is very smooth and is mainly influenced by the high gradient components of the data sets. The horizontal section cuts through the ear, where we some of the strongest deformations. When we compare that with the result for the Gauss-Newton method the difference is obvious. The transformation vectors within the brain are of lower magnitude and we see changes in the direction close to cavities where the brain was eroded. The more subtle differences for the Gauss-Newton method are not visible because we would have had to scale the them differently from the ones for the gradient descent method. The same thing can be observed in the contour plots in figures 5.14 and 5.15. The contour lines for the Gauss-Newton method are more localized. Close to the outer cortex boundary, where most of the erosion took place, they are close together, indicating that the local changes are better recognized.

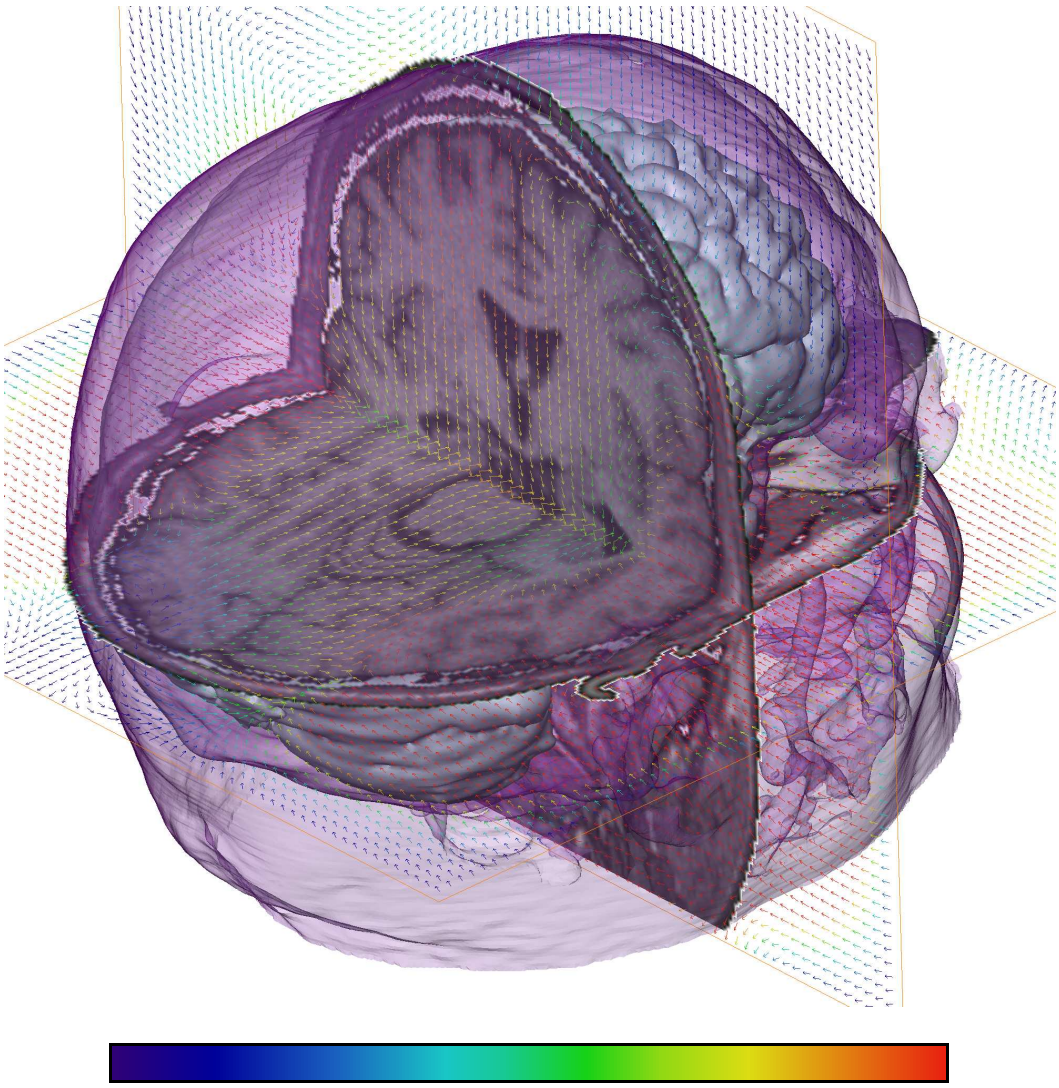


Figure 5.12: Volume rendering with transformation vectors for the gradient descent method. The transformation vectors are projected onto two orthogonal plains through the MRI data set. The surfaces of the skull (transparent) and of the brain (solid) are shown, too. The magnitude of the vectors is indicated by the coloring. The colormap goes from blue over green, yellow, and orange, to red. All vectors with a magnitude higher than 0.2 of the voxel length are red.

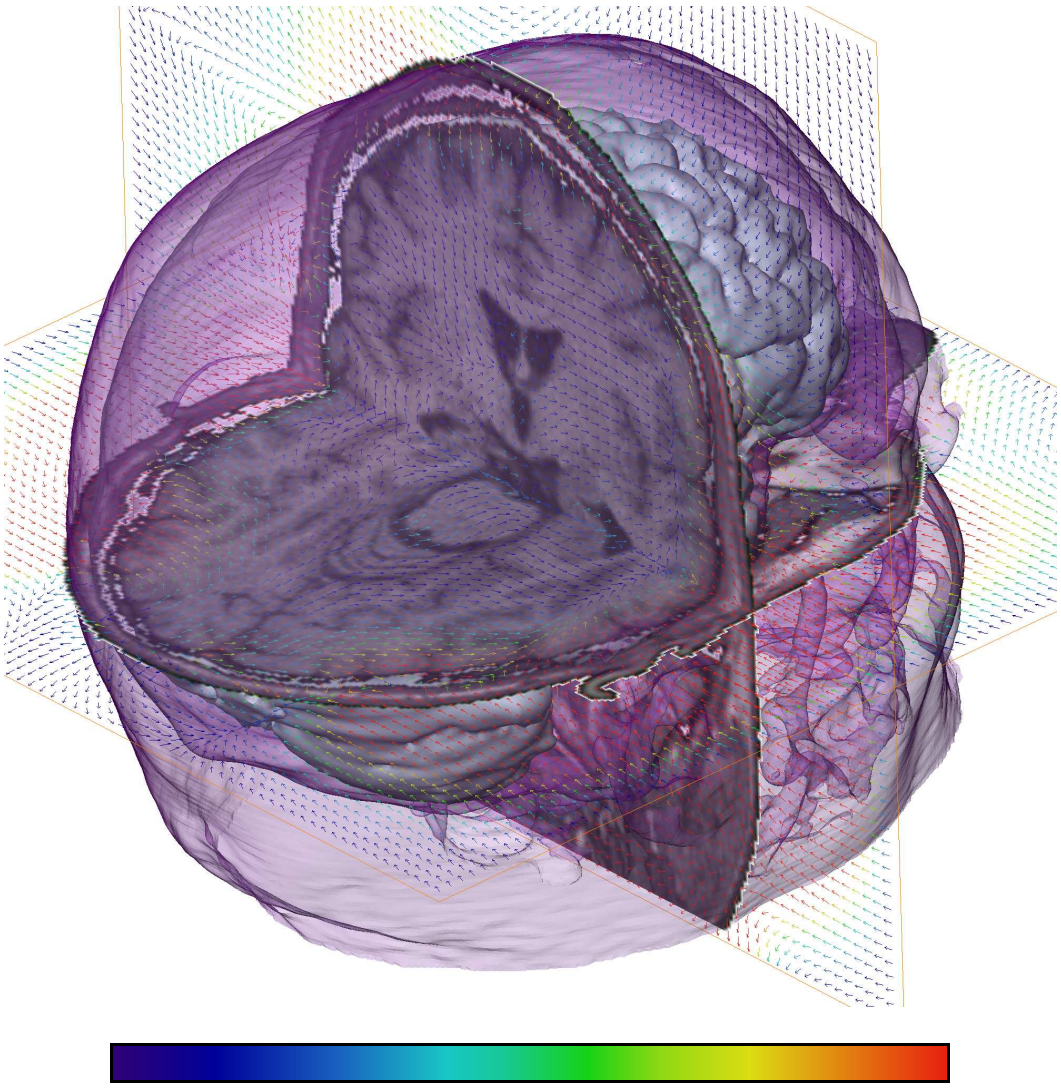


Figure 5.13: Volume rendering with transformation vectors for the Gauss-Newton method. The transformation vectors are projected onto two orthogonal plains through the MRI data set. The surfaces of the skull (transparent) and of the brain (solid) are shown, too. The magnitude of the vectors is indicated by the coloring. The colormap goes from blue over green, yellow, and orange, to red. All vectors with a magnitude higher than 0.2 of the voxel length are red.

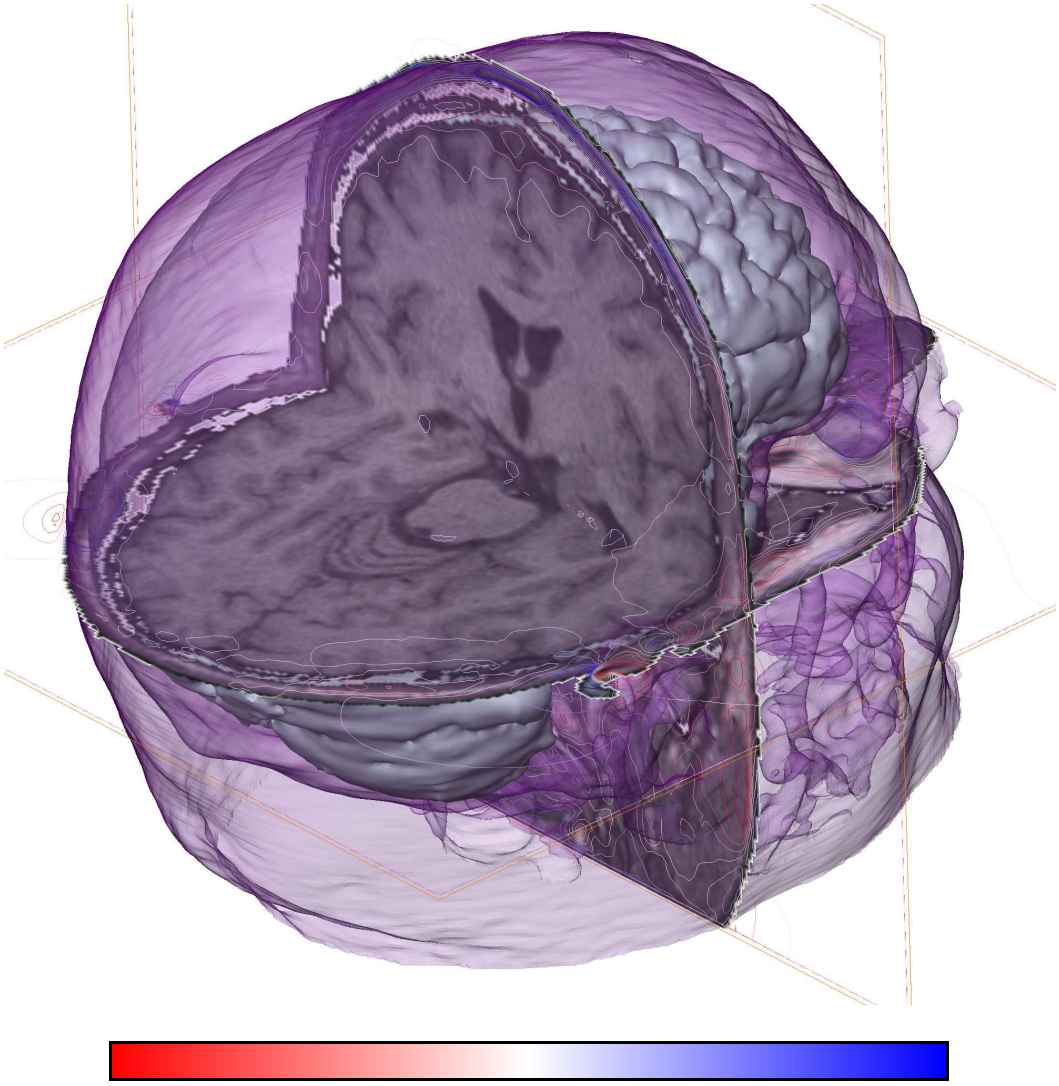


Figure 5.14: Volume rendering with contour lines for the gradient descent method. The contour lines are generated from a scalar volume change data set computed from the transformation. The contour lines are plotted on two orthogonal section through the MRI data set at ten levels in the range of $\pm 5\%$ volume change. The colormap ranges from red to white for volume decrease and white to blue for volume increase. The surfaces of the skull (transparent) and of the brain (solid) are shown, too.

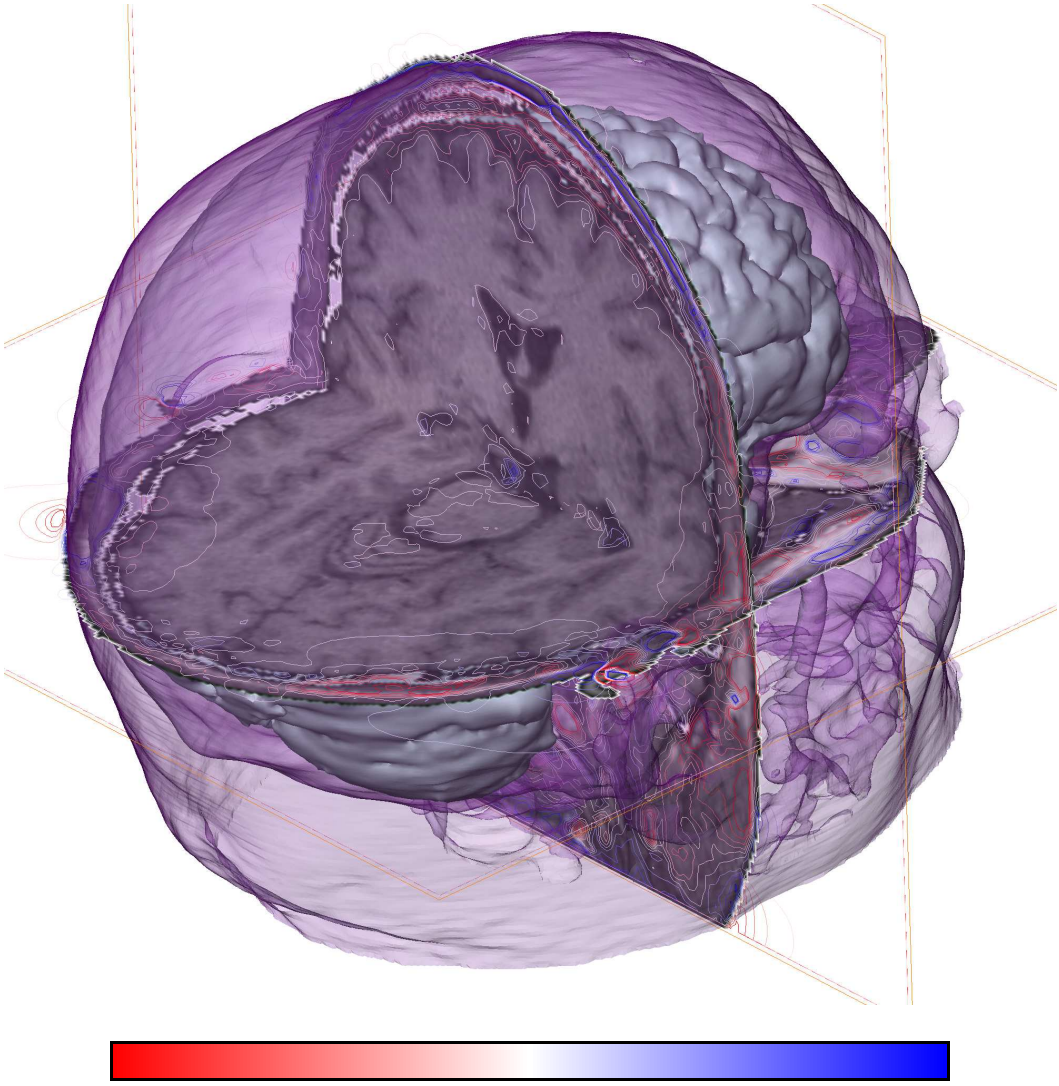


Figure 5.15: Volume rendering with contour lines for the Gauss-Newton method. The contour lines are generated from a scalar volume change data set computed from the transformation. The contour lines are plotted on two orthogonal section through the MRI data set at ten levels in the range of $\pm 5\%$ volume change. The colormap ranges from red to white for volume decrease and white to blue for volume increase. The surfaces of the skull (transparent) and of the brain (solid) are shown, too.

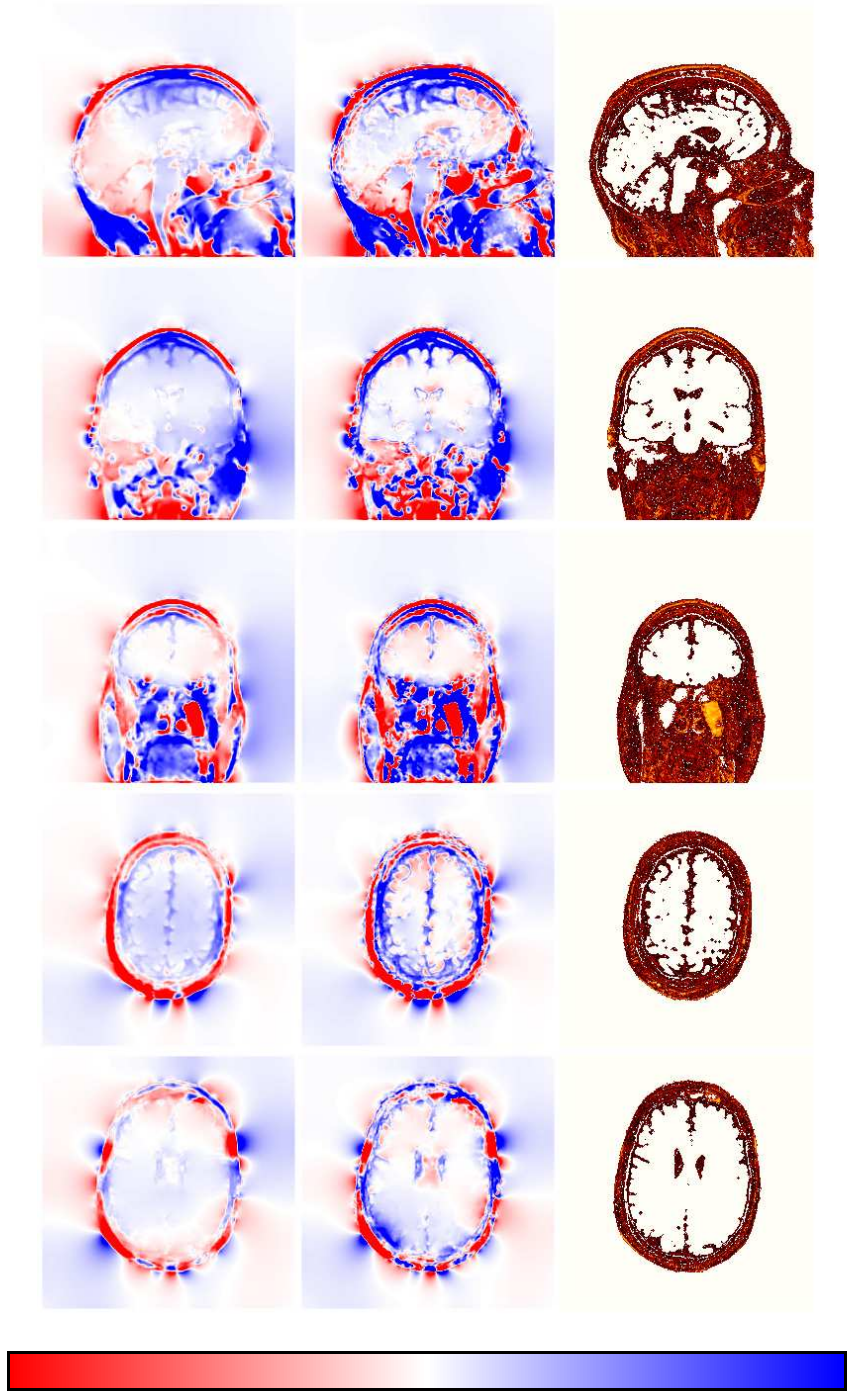


Figure 5.16: Volume changes on individual sections. In the left column results for the gradient descent method are displayed. In the middle column you find the results for the Gauss-Newton method. The logarithm of the voxelwise squared image difference between template and reference before registration is displayed in the right column. The colormap for the volume change ranges from red (decrease) over white (neutral) to blue (increase). The colormap is scaled to $\pm 5\%$ volume change. The image difference is colored according to a heat colormap. Dark colors indicate small differences and light colors greater differences. The parts of the volume that are identical are colored white for better contrast.

To further illustrate this point we show selected sections in figure 5.16. In the left column results for the gradient descent method are displayed. In the middle column you find the results for the Gauss-Newton method. The logarithm of the voxelwise squared image difference between template and reference before registration is displayed in the right column. The colormap for the volume change ranges from red (decrease) over white (neutral) to blue (increase). The colormap is scaled to $\pm 5\%$ volume change. The image difference is colored according to a heat colormap. Dark colors indicate small differences and light colors greater differences. The parts of the volume that are identical are colored white for better contrast. For the gradient descent method the detected volume changes are not as localized and sharp as for the Gauss-Newton method. The volume increase in the space between skull and brain is not captured well. In the images for the Gauss-Newton method the surface of the brain is clearly visible and we see a volume increase in the space around the brains, even in the depth of the sulci. On the surface of the brain itself we see a decrease in volume. In the images for the gradient descent method everything is looks more "hazy". All of this corroborates the observations made in the example with the histological section that in cases where the expected transformation are localized the Gauss-Newton method delivers the more precise results. Through the erosion only the outer surface of the brain is moved, not the rest. In regions where the gradients are small relative to the maximum gradients the Gauss-Newton method "does more". Even when the force f is small there is also information about the image edge in the operator M which is lacked in the gradient descent method. It should be noted that due to the construction of the example it cannot be expected that the erosion of the segmented brain is the only contributing factor to volume change. Before we inserted the brain into the skull of the second MRI data set that data set had to linearly registered to the other which incurs an interpolation of the data. Furthermore we did not perform a histogram equalization between the skulls. Hence gray value differences between the skulls and the cortico spinal fluid (CSF) of the two data sets contribute, too. In the actual application a number of preprocessing steps is necessary to ensure that interpolation artefacts, change in the shimming of the scanner coils, and other influences are held to a minimum. We did not do this here because we only wanted to demonstrate the difference between the two approaches.

5.3 Multiresolution framework

As discussed in section 4.5 the multiresolution approach allows for the robust and efficient computation of the transformation, by computing different components of the transformation on different levels of resolution. We illustrate this concept with a very simple model problem. The reference and template are shown in figure 5.17(a) and (b). The template and reference are shown at consecutive resolution levels in figure 5.17(c) and 5.17(d). To make the smoothing in the multiresolution framework, i.e. the loss of fine scale information, more obvious the blobs have been filled with a pattern at the finest resolution. We chose this simple example instead of a more complex one, because we need a problem that converges to approximately the same solution regardless of the number of different resolution levels. Images with a lot of fine scale information that require large transformations are not well suited because the registration algorithm tends to get trapped in local minima as discussed in section 4.5. Here we mainly want to illustrate that the multiresolution approach can greatly reduce computation time.

We registered the reference and template of the model problem and only varied the number of levels in the multiresolution approach. The outer iteration on one resolution level was stopped when the reduction in E_α from one step to the next was less than one percent of the start value. The computations were performed for one to five levels and two values for α, 0.01 and 0.001. The results are visualized in figure 5.18. The number of iterations is displayed on the x-axis and image difference is displayed in a logarithmic scale on the y-axis. The switch from one level to next finer level is indicated by a circle in the graphs. The computational effort for the computation of one step in the outer iteration differs from resolution level to resolution level. Hence, we normalized the number of iterations proportional to the number of grid points on the finest grid. A iteration on the second finest resolution is only counted as a fourth of an iteration and so on. This normalization allows us to assess the amount of computation time saved.

After switching to the next finer level we see an initial increase in the image difference. This is due to the fact that the next finer level contains new fine scale information that has not been considered for the transformation on the coarser resolution. These new details are not well matched, yet. The amount of the increase seems to be related to the resolution level. With increased resolution the initial increase becomes smaller.

There is another effect that can be seen in the data in figure 5.18. For four to six levels we do not gain efficency. The maximal number of levels is related to the

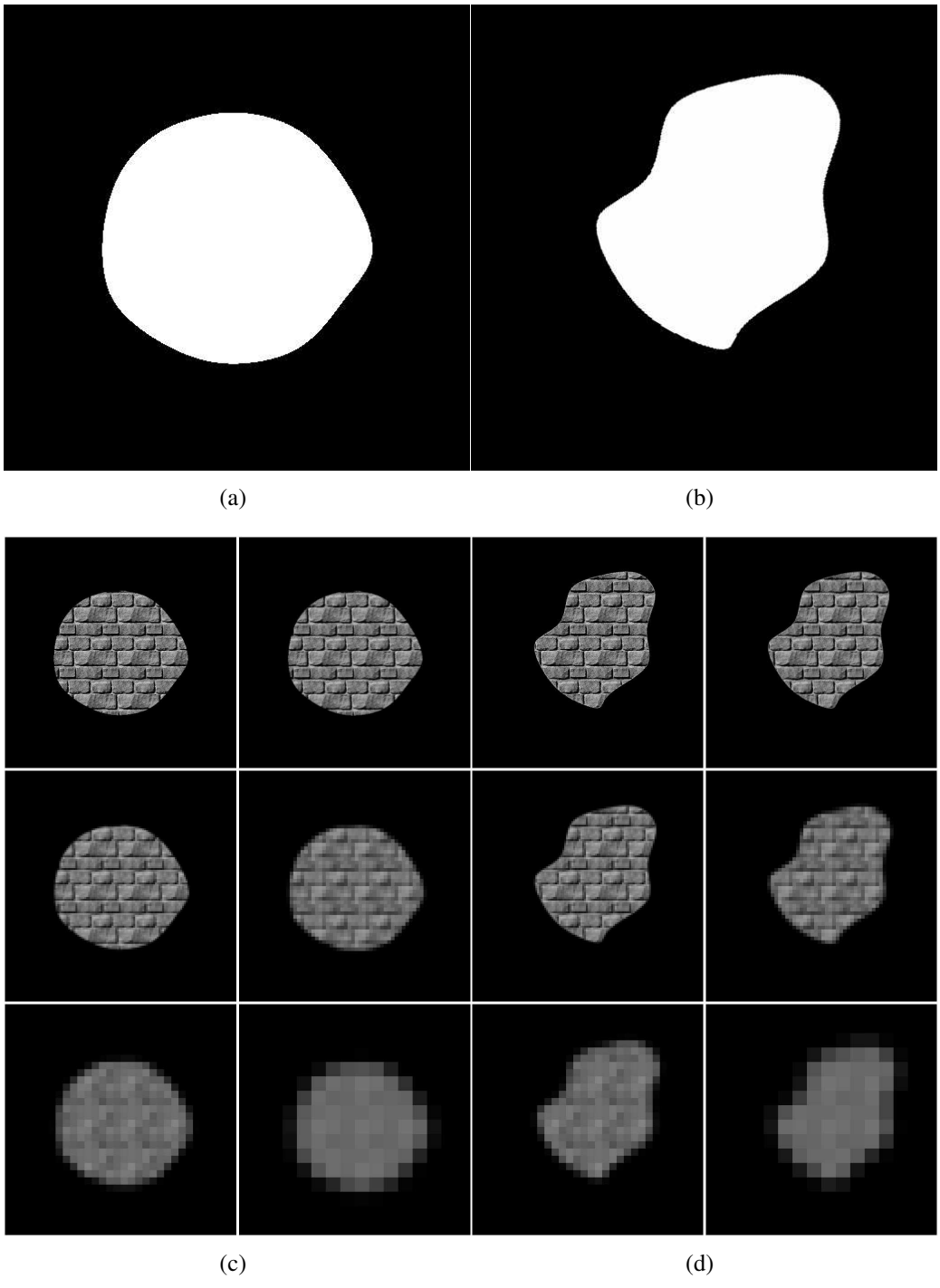
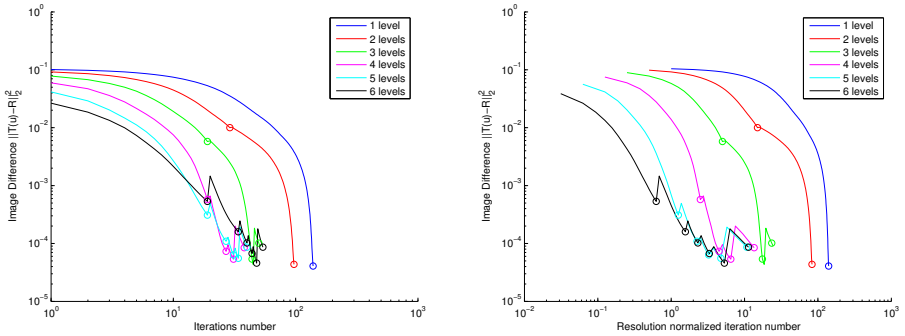


Figure 5.17: A artifical model problem with just a constant white blob is shown. The reference (a) and the template (b) are shown at different resolution levels in (c) and (d). To illustrate the smoothing from on resolution level to the next the blob has been filled with a pattern.



(a) $\alpha = 0.001$ (b) $\alpha = 0.001$

(c) $\alpha = 0.01$ (d) $\alpha = 0.01$

Figure 5.18: Registration was performed for the model problem in figure 5.17 with two values of α and for different number of levels in the multiresolution approach. The number of iterations in the right column is normalized by the amount of grid points on the resolution levels with respect to the finest resolution. The switch from one level to the next is marked by a circle.

maximal transformations we expect and the scale of information contained in the images. The registration is driven by features in the reference and template that are different. When we go down to a resolution there these difference are obscured, we cannot expect to gain anything in terms of efficency. In the present example we see that up to a certain image difference the normalized iterations needed differ for four to six levels, but then they coincide. This can be attributed to the fact that the information needed for this exactness is not present at lower resolutions.

When we compare the top and the bottom row in figure 5.18 we see that in some situations the multiresolution approach does not only save computation time, but also leads to more precise results. For $\alpha = 0.01$, the larger of the two values, the image difference is only reduced to the same amount when four to six levels are used. There is a clear dependency on the number of levels of resolution for the final result. When α is larger the influence of the penalty term in E_α is stronger. Hence,

the results indicate that the many small steps computed at the fine resolutions lead to transformations that are not that well in accordance with our model. That could have something to do with a point we discussed above. Since the Gauss-Newton method is a method with fast local convergence it is always better to be close to the solution. When we compute large transformation on coarse resolutions, where they are small transformations, we are closer to the solution on that resolution, than we are on a finer one.

CHAPTER 6

Shape constraints

In this chapter we introduce a framework for additional shape constraints that can be easily plugged in the registration method described in the previous chapters. The constraints introduced here are *soft constraints* in the sense that global penalty terms are used and the constraints are not enforced for each pixel like the volume preserving constraints in [23] or the landmark constraints in [18, 19].

6.1 Introduction

It cannot be guaranteed that the registration is always "successful", i.e. that the right correspondences are established, while sticking to the correct class of transformations. The reasons for that are manifold. Registration may fail because

1. the distance measure does not relate the template and reference in a proper way,
2. the actual transformation Φ is not within the class of transformations considered,
3. the optimization gets stuck in a local minimum,
4. the images do not contain the necessary information to establish correspondence.

These are probably the main four reasons. The first two are related to modeling. If for example corresponding objects in the template have gray values that differ considerably from the ones in the reference the squared distance of the images is a bad choice. We have to resign to multimodal distance measures like mutual information [36, 48] or morphological distance measures [16] . When we restrict the class of permissable transformation to the class of affine linear transformation, but the actual transformation Φ has nonlinear components, a "good" registration result cannot be expected.

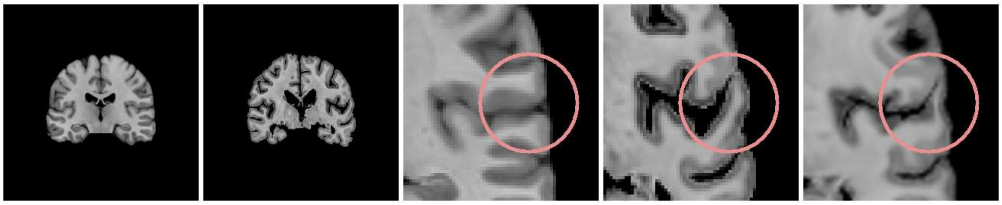
Even when the correct model is employed we still cannot be sure that registration will be successful. This brings us to the latter two reasons. When the model is correct the correct solution has to be global minimum of the functional to be minimized. We cannot determine that minimum analytically. Thus, iterative methods that evaluate the functional locally to find search directions are used. The risk of getting stuck in a local minimum is inherent, because once a stationary point is reached in the iteration there is no local search direction that would decrease the functional any further. There are strategies to make optimization algorithms more robust against such situations, but they can never be avoided completely. We discussed one of these strategies, multiresolution, earlier in section 4.5 and 5.3. We will now give an example how trying to avoid local minima via a multiresolution strategy can produce a situation described in item 4.

6.1.1 Motivational example

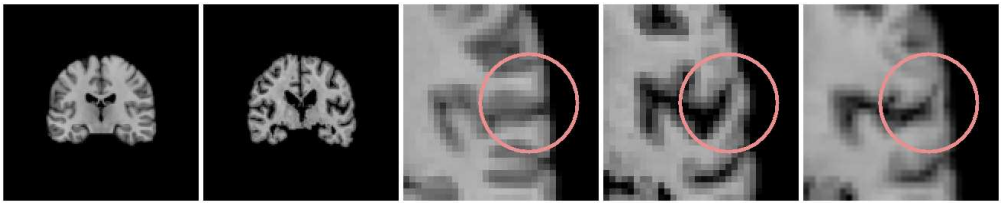
The idea behind multiresolution strategy is that computational burden and robustness can be improved by computing coarse components of the transformation on coarse grids. The robustness is achieved by getting rid of ambiguous fine scale information by smoothing and subsampling. Yet, sometimes things go awry and instead of removing ambiguous information we loose valuable structural information.

We discuss this concept of "loosing valuable structural information" on a specific example shown in figure 6.1. The first column shows the reference, a section of a T1-weighted MRI data set. The template in the second column is a section from a reconstructed postmortem brain that had been sectioned for microstructural examinations. The template has already been linearly registered to the reference, i.e. an affine linear transformation including scale, translation, rotation, and shear has been computed. The third and fourth column show a magnification of the same region in both reference and template. The rows correspond to different resolution levels in the multiresolution framework. The fifth column shows the result after nonlinear registration on each resolution. The transformation computed on one resolution was always used as a start approximation for the next finer resolution, as described in section 4.5.

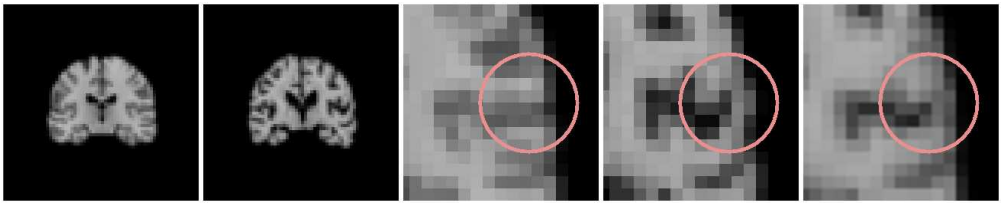
The problematic part is marked by a circle. The two "fingers" (gyri) in the reference belong to two separate brain structures, the parietal lobe (upper part) and the temporal lobe (lower part). The same is true for the template image. If we would overlay the reference and the template we would see that the gyrus in the



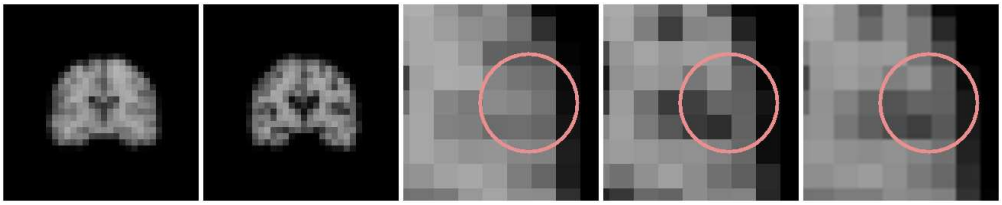
(a) Resolution 0



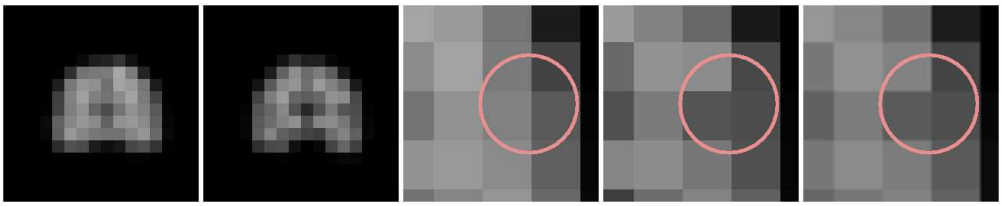
(b) Resolution 1



(c) Resolution 2



(d) Resolution 3



(e) Resolution 4

Figure 6.1: The first column shows the reference, a section of a T1-weighted MRI data set. The template in the second column is a section from a reconstructed postmortem brain that had been sectioned for microstructural examinations. The template has already been linearly registered to the reference, i.e. an affine linear transformation including scale, translation, rotation, and shear has been computed. The third and fourth column show a magnification of the same region in both reference and template. The rows correspond to different resolution levels in the multiresolution framework. The fifth column shows the result after nonlinear registration on each resolution. The transformation computed on one resolution was always used as a start approximation for the next finer resolution, as described in section 4.5.

template that belongs to the temporal lobe extends into the parietal lobe of the reference. This should not be the case after registration. To obtain such a result would require a strong local deformation. On the fine resolution the difference at the tip we would like to move downwards is small. Hence there are no strong forces that would move this part. The distance to the correct location that we infer from anatomical knowledge is too large. On coarser resolution that distance is smaller with respect to the number of pixels. The structure should be pulled along with other parts of the image due to the elastic penalty term, but on the lower resolution the surrounding structures are well matched already. Additionally, the pixels that belong to one of the structures are now mixed with pixels that belong to the other structure. Hence, the information that is needed to separate both is not present or too weak on the lower resolutions.

To sum up the situation. We have a complex structure, the human brain, that has a complex folding pattern. When overlayed the structures can be very similar locally, while not being very similar in a larger context. In the human brain that is mainly due to the fact that we have gray matter on the outside and white matter on the inside. Gray matter, respectively white matter, in one part of the brain cannot be distinguished from the one in another part of the brain. Strong local deformations lead to ambiguities that cannot be resolved on fine resolutions due to locality, and can also not be resolved on coarser resolution due to lack of information. Thus, we have to supply additional shape constraints to resolve ambiguities.

The above argument is not an argument against multiresolution strategies. They are useful, necessary, and lead to good results in many situations. The problem described above is a problem inherent to the image data used here, that can only be resolved by providing additional information. In the next section the possible representations of such additional information are discussed.

6.2 Representation of shape constraints

The additional shape constraints can be points (landmarks) and closed or open curves in two-dimensional space. In the three-dimensional space closed or open surfaces (surface patches) have to be added to that list. More generally one can say that the shape constraints are submanifolds of the image space.

Before we discuss how such shape constraints can be integrated into the registration process, we have to decide on how they will be represented. We can think of surfaces and curves as sets of pathwise connected points. Two methods for such

representations can be distinguished, namely parametric and implicit representations.

6.2.1 Parametric representations

Parametric representations, sometimes also called explicit representations, are given in form of a set of so-called parametric equations that express a set of quantities as explicit functions of a number of independent variables, the parameters. For example, the parametric representation of a circle in $\mathbb{R}^2$ is given by

$$\begin{aligned} x &= r\cos s, \\ y &= -r\sin s, \end{aligned}$$

with the angle $s \in [-\pi, \pi[$ and the radius $r \in \mathbb{R}_+$. When we add an equation for the third dimension we obtain a curve in $\mathbb{R}^3$. The equation

$$z = s$$

with $s \in \mathbb{R}$ produces a spiral pointing along the z-axis. When we express the equation for the z-direction in terms of a second parameter we obtain a surface, e.g.

$$z = t, \qquad t \in \mathbb{R}$$

defines a tube with unit radius along the z-axis.

The key property of parametric representations is that the points on the represented curve or surface are the result of the evaluation of explicit functions. The domain of the explicit functions is not necessarily, and will generally not be same as the range of these functions. In the simple example with the circle the parameter is an angle, i.e. a scalar in $\mathbb{R}$, while the circle is a set of points in $\mathbb{R}^2$. It is straightforward to produce all points that belong to the represented shape explicitly. Determining whether a specific point is an element of the represented shape can be extremely difficult. For implicit representations the opposite is the case.

6.2.2 Implicit representations

In implicit representations the shape is represented by the set of points that fulfill certain constraints given by a system of equations. The information is given only

implicitly not explicitly. That makes it very easy to determine whether a point belongs to the represented shape, simply by checking if all constraint equations are fulfilled. The circle from the above example can be represented by the equation

$$x^2 + y^2 - r^2 = 0,$$

with a given radius r. The same equation also produces the tube in three-dimensional space. Note that z is not used explicitly, it is given implicitly by the choice of the domain.

There is no straightforward way for producing all points that belong to the shape. A task which is easy for parametric representations. Essentially it is necessary to convert the implicit into a parametric (explicit) representation. The example of the circle demonstrated that there is not always a unique solution. Above we used polar coordinates, but we could also use Cartesian coordinates and express the circle by

$$\begin{aligned} x &= s, \\ y &= \pm\sqrt{r^2 - s^2}, \end{aligned}$$

with $s \in]-r, r]$. This formulation is closer to the implicit representation, yet it is also not as elegant. Things can easily become more complicated. The spiral along the z-axis for example is more difficult to express in terms of an implicit representation, because we do not have the rotation angle as a parameter. A square with sides of length r that is rotated by 45 degrees around the origin can be represented by the implicit equation

$$|x| + |y| - r = 0.$$

To express the same thing as a parametric representation with one free parameter we have to provide four different equations, one for each quadrant, e.g.

$$x(t) = \begin{cases} rt, & \text{for } t \in [0,1] \\ r(2-t), & \text{for } t \in [1,2] \\ -r(t-2), & \text{for } t \in [2,3] \\ -r(4-t), & \text{for } t \in [3,4] \end{cases} \qquad y(t) = \begin{cases} r(1-t), & \text{for } t \in [0,1] \\ r(1-t), & \text{for } t \in [1,2] \\ -r(t-3), & \text{for } t \in [2,3] \\ -r(3-t), & \text{for } t \in [3,4] \end{cases}.$$

We could also express the shape as a polygon with control points $(0, r)$, $(r, 0)$, $(0, -r)$, and $(-r, 0)$. The connections between the control points are represented

by a parametric line equation. Basically this is a finite element discretization of the shape boundary. This is a common form of parametric representation in computer graphics. These examples indicate that a closed explicit formula can only be given locally. This is also one of the main statements of the implicit function theorem which will be used later on.

6.2.3 Weighing the pros and cons

In computer graphics both parametric and implicit representations are used. Which one is used depends largely on the area of application. Parametric representations, like nonuniform rational B-splines (NURBS), are widely used in free form modeling, e.g. computer aided design (CAD). The main advantage of parametric representations is that the underlying control points can be locally refined and easily manipulated. Higher-order derivatives can be calculated analytically. Furthermore they are easy to render.

The main disadvantage of parametric representations is that collision detection and binary operations, such as intersection are very difficult and computationally expensive. Collision detection amounts to detection of intersections between two parametric representations in the image space. The parameter spaces of both shapes are likely to be totally different, i.e. the coordinate in image space for some fixed values of the parameters is different for both shapes. When we look for an intersection between a surface and a curve not even the arity of the explicit functions has to be identical. Two shapes intersect if the distance between them at some point in the image space is 0. The computation is expensive because in the worst case we have to check each segment between control points in one shape against each segment of the other shape, simply because we cannot compare the parameters directly. Merging two shapes with a parametric representation also requires "glueing" them together at the points of intersection. That means the actual shape generated by the intersection has to be extracted. That is difficult because the intersection has to be parameterized, which becomes especially difficult when changes in topology occur.

In the implicit representation both shapes are given in terms of coordinates in image space. The shape is represented by a set of points in the image space that is the solution to a system of equations. Hence the intersection of two shapes is the intersection of two such sets. We can even chose the underlying function such that it is the distance to the shape. Then it is very easy to compute the distance of a point on one shape to the other shape. It is just a function evaluation.

We will use implicit representations instead of parametric ones. The main reason for this choice is the difficulty in computing the distance between parametric representations. When we want the shapes to drive the registration, we have to be able to assess the dissimilarity between the template and the reference shapes. For parametric representations that could only be done efficiently under strong restrictions that are not desirable for our applications.

When both parametric representations are parameterized in the same way, e.g. NURBS with an identical number of control points, one could minimize the difference between corresponding control points. In that case one would have to track the control points during registration. The control points that have to be transformed are given in Lagrange coordinates in the template space. The transformation gives us the Lagrange coordinate for a given Euler coordinate, but not the other way around. Hence, the computation of the transformed points requires a local inversion of the transformation at these locations.

When we try to match shapes that are parameterized differently we have to deal with another problem apart from the difficult computation of dissimilarity. As the template parameterization is transformed situations can arise where we have to reparameterization the template shape. Since the computation of the distance between the shapes would involve information from between the control points, we have to be sure that that information remains valid. If for example B-splines are used that enforce C^2-differentiability at the control points the very oscillation artefacts that we seek to avoid by using polynoms with limited support, are likely to appear when control points are moved such that they are very close to each other. We could use linear interpolation between the control points instead. Under the assumption that the distances between control points are short it might be accurate enough, but when the distance between the control points increases during registration reparameterization would be necessary.

In [25], for example, the control points of B-spline representation are matched to each other. To this end corresponding representation are resampled such that they contain the same number of control points. The transformation is restricted to locally affine linear transformations. The authors admit though that they are not sure whether matching the control points is anatomically sensible.

6.3 Implicit representations via Level Sets

We will now give a more thorough characterization of implicit representations via implicit functions. An implicit function has to fulfill certain conditions to represent a manifold in $\mathbb{R}^n$. In the following we look in more detail at these conditions and introduce a special class of implicit functions that is well suited to represent such manifolds with the additional benefit of easy distance computation, the so-called *distance functions*.

6.3.1 Level sets and the implicit function theorem

Definition 6.1 (Level set). *Given a function*

$$\psi : \Omega \to \mathbb{R}^m, \quad \Omega \subset \mathbb{R}^n, \quad n \geq m$$

and a fixed $\upsilon \in \mathbb{R}^m$. The level set of ψ at level υ, or "υ level set of ψ", is given by

$$L_\psi(\upsilon) := \{\alpha \in \Omega | \psi(\alpha) = \upsilon\}.$$

The equation

$$\psi(\alpha) = \upsilon, \quad \alpha = (a_1, \ldots, a_n) \tag{6.1}$$

can also be written as a system of equations

$$\begin{array}{rcl} \psi_1(a_1, \ldots, a_n) & = & \upsilon_1 \\ \psi_2(a_1, \ldots, a_n) & = & \upsilon_2 \\ \vdots & \vdots & \vdots \\ \psi_m(a_1, \ldots, a_n) & = & \upsilon_m \end{array}$$

with functions $\psi_i : \Omega \to \mathbb{R}$, $1 \leq i \leq m$. A specific solution of equation (6.1) is a point in Ω, and the collection of all such points forms an object in Ω.

For the relationship between m, n in definition 6.1 three situations can be distinguished. When $n < m$ the system of equations is overdetermined and normally the system will have no solution. When $n = m$ and the system of equations has a solution that solution is unique when certain conditions are satisfied. The most interesting case for us is $n > m$. In this case the system of equations is underdetermined and has infinitely many solutions. Under certain conditions these solutions will form a manifold of dimension $n-m$. This is the content of the implicit function theorem. We give a version without proof that is adapted to our needs.

Theorem 6.1 (Implicit function theorem). *Let* $\psi : \Omega \to \mathbb{R}^m$, $\Omega \subset \mathbb{R}^n$, $n > m$ *and* $\psi(\alpha) = \upsilon$, $\alpha \in \Omega$ *and* $\upsilon \in \mathbb{R}^m$. *If*

$$\psi^{'}(\alpha) := J_\psi(\alpha) : \mathbb{R}^n \to \mathbb{R}^m$$

is onto, i.e. the Jacobian of ψ *has rank* m, *the following hold.*

1. *Near* α, $L_\psi(\upsilon)$ *the level set of* ψ *at level* υ, *is an* $(n-m)$*-dimensional submanifold of* Ω.
2. *The tangent space to* $L_\psi(\upsilon)$ *is perpendicular to the row vectors of the matrix*

$$J_\psi(\alpha) = \begin{pmatrix} \frac{\partial \psi_1}{\partial x_1}(\alpha) & \cdots & \frac{\partial \psi_1}{\partial x_n}(\alpha) \\ \vdots & \ddots & \vdots \\ \frac{\partial \psi_m}{\partial x_1}(\alpha) & \cdots & \frac{\partial \psi_m}{\partial x_n}(\alpha) \end{pmatrix}.$$

Item 1 states that the submanifold is described by the solution to a system of equations. Item 2 states that the corresponding tangent space is the null space of the Jacobian, or in other words the orthogonal complement of the linear subspace spanned by the Jacobian. Let $\mathcal{M}_{L_\psi(\upsilon)}$ be the submanifold described by $L_\psi(\upsilon)$, then its tangent space at a point α is

$$\mathcal{T}_\alpha \mathcal{M}_{L_\psi(\upsilon)} = \ker(J_\psi(\alpha)).$$

The dimension of the submanifold $\mathcal{M}_{L_\psi(\upsilon)}$ is the nullity, i.e. $\dim(\ker(J_\psi(\alpha)))$, of $J_\psi(\alpha)$. Each equation ψ_i, $1 \leq i \leq m$, eliminates one degree of freedom with respect to the possible directions of movement on the original manifold (the ambient space). This relation between a manifold and its submanifolds can be characterized by the codimension.

Definition 6.2 (Codimension). *If* W *is a manifold of dimension* n *and* V *is submanifold of* W *of dimension* k, *then the codimension of* V *is* $n-k$.

$$\textit{codim}(V) = n - k$$

Thus, the codimension of $\mathcal{M}_{L_\psi(\upsilon)}$ is m or the number of independent constraint equations ψ_i. Note that the space $\mathbb{R}^n$ is by definition a n-dimensional manifold. For $\mathbb{R}^n$ the shapes with the highest possible codimension of n are points. We first consider codimension one shapes, i.e. surfaces in three-dimensional space and curves

in two-dimensional space. Later we will turn our attention to shapes of higher codimension, or shapes with higher codimension boundaries. For codimension one, $m = 1$, we have the following version of the implicit function theorem.

Theorem 6.2 (Implicit function theorem for $m = 1$). *Let $\psi : \Omega \to \mathbb{R}$, $\Omega \subset \mathbb{R}^n$, and $\psi(\alpha) = \upsilon$ for some $\alpha = (a_1, \ldots, a_n) \in \Omega$ and $\upsilon \in \mathbb{R}$. If $\frac{\partial \psi}{\partial x_n}(x) \neq 0$ then the following hold.*

1. *There is a function $\zeta(x_1, \ldots, x_{n-1})$, defined near $(a_1, \ldots, a_{n-1}) \in \Omega \cap (\mathbb{R}^{n-1} \times \{a_n\})$, such that*

$$\psi(x_1, \ldots, x_{n-1}, \zeta(x_1, \ldots, x_{n-1})) = \upsilon.$$

2. *Near α the given equation has no solutions other than the ones described in 1.*

3. *Near α, $L_\psi(\upsilon)$ is a $(n-1)$-dimensional manifold, and its tangent plane at α is perpendicular to $\nabla\psi(\alpha)$.*

4. *The derivative of ζ at $(a_1, \ldots, a_{n-1})$ is given by*

$$\begin{aligned} \zeta'(a_1, \ldots, a_{n-1}) &= \left[\frac{\partial \zeta}{\partial x_1}, \ldots, \frac{\partial \zeta}{\partial x_{n-1}}\right](a_1, \ldots, a_{n-1}) \\ &= \left[-\frac{\frac{\partial \psi}{\partial x_1}(\alpha)}{\frac{\partial \psi}{\partial x_n}(\alpha)}, \ldots, -\frac{\frac{\partial \psi}{\partial x_{n-1}}(\alpha)}{\frac{\partial \psi}{\partial x_n}(\alpha)}\right] \end{aligned}$$

In short under the above conditions a_n can be locally expressed explicitly as a function of $(a_1, \ldots, a_{n-1})$. Moreover the above theorem states how and under which conditions the implicit function can be differentiated (implicit differentiation). Note that $\nabla\psi(\alpha)$ is the normal vector to the tangent plane. The direction of that vector will shortly play an important role.

6.3.2 Implicit representations for closed shapes

Now we can rethink what shapes in our images are and how they can be represented by implicit functions. Generally the shape is on the surface of an object. Let us first give a more formal definition of what is meant by objects.

Definition 6.3 (Connected object). *Let Ω be an open subset of $\mathbb{R}^n$ and let $\mathcal{B}$ be an open pathwise connected subset of Ω. Then we call $\mathcal{B}$ a connected object in Ω. Since $\mathcal{B}$ is a subset of $\mathbb{R}^n$ it is a n-dimensional manifold. The boundary $\partial\mathcal{B}$ is a $(n-1)$-dimensional closed manifold.*

The $(n-1)$-dimensional manifold from the above definition is our first shape. Later on we compose more complex shapes from this simple building block. Connected objects have an inside and an outside. We can build predicates based on the object itself, or on the boundary and a fixed but arbitrary point. A point $x \in \Omega$ is inside a connected object $\mathcal{B} \subset \Omega \subset \mathbb{R}^n$ if

1. $x \in \mathcal{B}$ or,
2. if there exists a connected path from x to a fixed but arbitrary point $x^* \in \mathcal{B}$ that does not include any point in $\partial\mathcal{B}$.

A point is outside if it is not inside. Predicate 1 is trivial. Predicate 2 utilizes information about the boundary of the object and one additional point inside the object. It is clear that the information of the boundary alone is not sufficient to tell whether a point is inside the object, because it is of lower dimension. Like in Flatland [1] where a two-dimensional being is visited by a sphere which it perceives as a circle varying in diameter as it moves through the two-dimensional plane. Some additional information is always needed to determine whether a point is in- or outside of the object defined by its boundary.

Note that the implicit function is defined on the whole of Ω and the interface $\partial\mathcal{B}$ is given by its zero level-set. We have not said how the implicit function is defined on the rest of Ω. That leaves us with the freedom to incorporate the concept of inside and outside into the implicit function. We simply encode the predicate as the value of the implicit function.

Idea 6.1. *Define the implicit function ψ of which $\partial\mathcal{B}$ is the level set at level 0, such that*

$$sign(\psi(x)) = \begin{cases} -1 & , if\ x \in \mathcal{B} \\ 0 & , if\ x \in \partial\mathcal{B} \\ 1 & , otherwise \end{cases}$$

A common choice for $\psi(x)$ is the signed distance of x from the level set at level 0. The sign is chosen as described above.

Definition 6.4 (distance function). *Let $\mathcal{B} \in \Omega \subset \mathbb{R}^n$ be a closed object, and $\partial\mathcal{B}$ its boundary, then the corresponding distance function $d(x; \partial\mathcal{B})$ is defined as*

$$\begin{aligned} d(x; \partial\mathcal{B}) \quad : \quad & \mathbb{R}^n \to \mathbb{R}_0^+ \\ & x \mapsto \min_{x_0 \in \partial\mathcal{B}} (||x - x_0||) \end{aligned}$$

Thus $d(x)$ is the distance to the closest point on $\partial\mathcal{B}$ in some norm $||\cdot||$.

If not indicated otherwise we use the euclidian norm $||\cdot||_2$. Note that if there is a unique closest point $x_0 \in \partial\mathcal{B}$ then $|\nabla d| = 1$ and ∇d is the normal. The definition of the signed distance functions is now straightforward. It is a combination of idea 6.1 and definition 6.4.

Definition 6.5 (Signed distance function). *Let $\mathcal{B} \in \Omega \subset \mathbb{R}^n$ be a closed object, and $\partial\mathcal{B}$ its boundary, then the corresponding signed distance function $\psi(x; \partial\mathcal{B}) : \mathbb{R}^n \to \mathbb{R}$ is an implicit function with $|\psi(x; \partial\mathcal{B})| = d(x; \partial\mathcal{B})$ for all $x \in \Omega$. Furthermore the sign indicates whether x is inside or outside of the object, i.e.*

$$\psi(x; \partial\mathcal{B}) = \begin{cases} -d(x; \partial\mathcal{B}) & , \text{if } x \in \mathcal{B} \\ 0 & , \text{if } x \in \partial\mathcal{B} \\ +d(x; \partial\mathcal{B}) & , \text{otherwise} \end{cases}$$

Again, if there is a unique closest point $x_0 \in \partial\mathcal{B}$ then $|\nabla\psi(x; \partial\mathcal{B})| = 1$. Remember that $\nabla\psi(x; \partial\mathcal{B})$ is the normal vector to the tangent plane. The signed distance function then is the distance to that tangent plane with an outward normal. From now on we assume that ψ is a signed distance function of a zero level set.

The advantage of the signed distance function is that we get the relation of a point x to the object and its boundary with just one function evaluation. The disadvantage is that we have to compute a function in $\mathbb{R}^n$ while the actual boundary of object, the shape we are interested in, is only $(n-1)$-dimensional. For the discretized version that means that we have to compute the distance function on the whole grid. Often we do not need to know the signed distance function on the whole domain, but only in a *narrow band* around the represented shape.

Definition 6.6 (Narrow band signed distance function). *Let $\mathcal{B} \in \Omega \subset \mathbb{R}^n$ be a closed object, $\partial\mathcal{B}$ its boundary, and $b \in \mathbb{R}$ the bandwidth, then the corresponding*

narrow band signed distance function is

$$\psi(x, b; \partial\mathcal{B}) = \begin{cases} -b & , \textit{if } \psi(x, \partial\mathcal{B}) < -b \\ b & , \textit{if } \psi(x, \partial\mathcal{B}) > b \\ \psi(x, \partial\mathcal{B}) & , \textit{otherwise} \end{cases} .$$

Note that $\nabla\psi(x, b; \partial\mathcal{B}) \equiv 0$ outside the band, and that the difference of two signed distance functions with the same bandwidth is 0 outside the band. The importance of the latter fact will become obvious, when we later define a distance functional with signed distance functions.

So far we have only defined the implicit function for the boundary of closed objects as defined in definition 6.3. In the next section we discuss how we can construct implicit functions for parts of these shapes and lower dimensional shapes.

6.3.3 Implicit representations for shapes of arbitrary codimension

Two main approaches for the construction of level set methods for objects of arbitrary codimension can be found in the literature. De Giorgi [22], Ambrosio and Soner [3] used one scalar valued function to represent codimension k objects, and Burchard et al. [9] used the intersection of k scalar valued functions to represent codimension k objects. The latter alternative is the natural consequence of theorem 6.1 (implicit function theorem). Both approaches have mainly been used in the evolution codimension $k > 1$ objects based on their configuration (geometrically based motion). In the present context a template level set is to be transformed in relation to a reference level set. The registration has to be driven by some force that decreases the distance between the reference and template representation. One of the arguments against the use of a single scalar valued function brought forth in [9] was that "the handling of topological changes does not carry over". They describe a phenomenon called "thickening" where the zero level set develops a non empty interior when curves try to merge. We do not face this problem here, because the functions that contribute to the level set representation of the template are not evolved separately, but are subject to the same transformation. Thus, we can and make use of both approaches, but we start with the representations via multiple scalar valued functions.

The following lemma about the relation of codimension and intersection gives some indication on how submanifolds with a higher codimension can be con-

structed from codimension one submanifolds.

Lemma 6.1 (Codimension and intersection). *Let V_1 and V_2 be submanifolds of a manifold $W \subset \mathbb{R}^n$. If V_1 has codimension k_1 and V_2 has codimension k_2, then*

$$\max(k_1, k_2) \leq \textit{codim}(V_1 \cap V_2) \leq k_1 + k_2.$$

Proof. Let V_1 be given by the zero level set of $\psi : \mathbb{R}^n \to \mathbb{R}^{k_1}$, and V_2 by the zero level set of $\phi : \mathbb{R}^n \to \mathbb{R}^{k_2}$, then $V_1 \cap V_2$ is given by the system of equations

$$\begin{array}{rcl} \psi_1(\alpha_1, \ldots, \alpha_n) & = & 0 \\ \vdots & \vdots & \vdots \\ \psi_{k_1}(\alpha_1, \ldots, \alpha_n) & = & 0 \\ \phi_1(\alpha_1, \ldots, \alpha_n) & = & 0 \\ \vdots & \vdots & \vdots \\ \phi_{k_2}(\alpha_1, \ldots, \alpha_n) & = & 0 \end{array}.$$

The codimension is the rank of the Jacobian of that system. Since J_ψ has rank k_1 and J_ϕ has rank k_2, the minimal rank of the Jacobian of the combined system of equations is $\max(k_1, k_2)$. The maximal rank is $k_1 + k_2$. □

If the two submanifolds from lemma 6.1 intersect transversally, the codimensions add exactly. Two submanifolds are said to intersect transversally if at every point of the intersection the separate tangent spaces at that point together generate the tangent space of the ambient manifold. To construct codimension two manifolds in $\mathbb{R}^2$ we can use the intersection of two lines. In the three-dimensional case we need the intersection of a line and a plane, where the line is the transversal intersection of two planes. Note that two curves in $\mathbb{R}^3$ are only transversal if they do not intersect at all. A circle in $\mathbb{R}^3$ is the intersection of a sphere (codimension one), and a plane (codimension one).

This construction principle can easily be applied to the level sets of implicit functions. Transversality translates into the rank of the Jacobian of ψ. The intersection of two submanifolds translates into the solution of a system of equations comprising their corresponding implicit equations, i.e. the zero level set of the new implicit function is the intersection of the zero level sets of implicit functions that correspond to codimension one manifolds.

Lemma 6.2. *Let $\psi_i, \ldots, \psi_m$, $\psi_i : \mathbb{R}^n \to \mathbb{R}$, be signed distance functions, and the*

combined implicit function defined by

$$\begin{aligned}\psi_{(1,\dots,m)} \quad : \quad & \mathbb{R}^n \to \mathbb{R}^m \\ & (x_1, \dots, x_m) \mapsto (\psi_1(x), \dots, \psi_m(x)),\end{aligned}$$

then the following hold:

1. *The level set of $\psi_{(1,\dots,m)}$ at level υ is then given by*

$$L_{\psi_{(1,\dots,m)}}(\upsilon) = \bigcap_{i=1}^{m} L_{\psi_i}(\upsilon_i).$$

2. *The codimension of $\mathcal{M}_{L_{\psi_{(1,\dots,m)}}}$ is rank$(J_{\psi_{(1,\dots,m)}})$.*
3. *The codimension of $\mathcal{M}_{L_{\psi_{(1,\dots,m)}}}$ is m if $\mathcal{M}_{L_{\psi_1}}, \dots, \mathcal{M}_{L_{\psi_m}}$ intersect transversally.*

Proof. Item 1 follows directly from the definition of the level sets (definition 6.1).

$$\begin{aligned}L_\psi(\upsilon) &= \{x | \psi(x) = (\psi_1(x), \dots, \psi_m(x)) = (\upsilon_1, \dots, \upsilon_m)\} \\ &= \{x | \psi_1(x) = \upsilon_1 \wedge \dots \wedge \psi_m(x) = \upsilon_m\} \\ &= \{x | \psi_1(x) = \upsilon_1\} \cap \dots \cap \{x | \psi_m(x) = \upsilon_m\} \\ &= \bigcap_{i=1}^{m} L_{\psi_i}(\upsilon_i)\end{aligned}$$

Item 2 follows directly from theorem 6.1. Item 3 follows from theorem 6.1 and definition 6.2. □

The zero level set of $\psi_{(1,\dots,m)}$ is the set of points that all zero level sets $L_{\psi_i}(0)$ have in common. Hence it is exactly what we are looking for, but $\psi_{(1,\dots,m)}$ is not a (signed) distance function. It maps to a vector of length m that contains the signed distance to each of the m $(n-1)$-dimensional manifolds used to create the new $(n-m)$-dimensional manifold.

The distance from $L_{\psi_{(1,\dots,m)}}(0)$ is simply given by $d(x; L_{\psi_{(1,\dots,m)}})$, but it is not possible to define a signed distance function like in section 6.3.2. There we dealt with codimension one objects, i.e. manifolds which normal space is one-dimensional. There for each point on the manifold there were always only up to two points with the same distance. One in the direction of the normal vector and one in the opposite direction. Manifolds of higher codimension do not have this property.

Lemma 6.3. *Let* $x_0 \in L_{\psi_{(1,\dots,m)}}$, $\upsilon \in \mathbb{R} \setminus \{0\}$, *and*

$$\mathcal{N}(x_0, \upsilon) = \{x |\, d(x, L_{\psi_{1,\dots,m}}) = \upsilon \wedge ||x - x_0|| = \upsilon\}$$

be the set of points with distance υ *to* x_0 *then the following hold:*

1. *For each* $x \in \mathcal{N}(x_0, \upsilon)$ *there exist* $\beta_i \in \mathbb{R}$, $1 \leq i \leq m$ *such that*

$$x = x_0 + \sum_{i=1}^{m} \beta_i \nabla\psi_i(x_0).$$

2. *For* $m = 1$, $|\mathcal{N}(x_0, \upsilon)| \leq 2$.

3. *For* $m > 1$, $|\mathcal{N}(x_0, \upsilon)| \leq \infty$.

Proof. 1. The point x has distance υ from the x_0. By definition connection vector has length υ and is orthogonal to the tangent space $\mathcal{T}_{x_0}\mathcal{M}_{\psi_{(1,\dots,m)}}$. Thus it has to be an element of the orthogonal complement which according to theorem 6.1 is span$\{\nabla\psi_1(x), \dots, \nabla\psi_m(x)\}$.

2. & 3. Note that $||x - x_0|| = \upsilon$ and item 1 are a necessary conditions for $d(x; L_{\psi_{(1,\dots,m)}}) = \upsilon$. Hence, only points that fulfill the former two are candidates for the latter. The expression $||x - x_0|| = \upsilon$ is a level set and represents a $(n-1)$-manifold. Specifically it is a $(n-1)$-sphere ($\mathbb{S}^{n-1}$) in the topological sense with the center x_0. The subspace that is spanned by $\nabla\psi_i(x_0)$, $1 \leq i \leq m$ is a m-dimensional manifold. The intersection of both is the set of permissable points. From lemma 6.1 we know that the codimension of the intersection is at least $\max(1, n-m) = n - m$ and at most $n - m + 1$.

   For $m = 1$ the $(n-1)$-sphere is intersected transversally by a line, i.e. the intersection are two points (0-manifolds) of distance υ from the origin x_0 in direction of $\nabla\psi_1(x_0)$.

   For $m > 1$ the intersection is a $(m-1)$-manifold with infinitely many points. For example, for $m = 2$ the $(n-1)$-sphere is intersected with a plane through x_0, and the result is a 1-sphere, i.e. a circle.

□

6.3.4 Implicit representations for non-closed shapes

So far we have only dealt with closed codimension one manifolds, i.e. boundaries of connected objects (see definition 6.3). These manifolds were always closed in the sense that you could take any direction along the manifold. We cannot model curve segments are surface patches with this technique. In the following we will discuss a technique to "cut" pieces out of the closed manifolds. That is equivalent to creating boundaries or barriers on the manifold.

A boundary on a $(n-1)$-dimensional manifold is a $(n-2)$-dimensional manifold. The codimension of the boundary is one less than that of the manifold. In two-dimensional space, for example, the ends of a curve segment are points. The curve has codimension one and the points have codimension two. In the last section we saw that such a lower dimensional manifold can be created by intersecting two $(n-1)$-dimensional manifolds. With same technique we can cut pieces out of a manifold. A manifold of equal dimension serves as a barrier, that is the boundary between the part of the original manifold that belongs to the sought for segment, and the rest. We can then define the segment as the part of the original manifold that is inside the "barrier" manifold.

Definition 6.7. *Let $\psi_1, \ldots, \psi_m : \mathbb{R}^n \to \mathbb{R}$, then the zero level set of the segment of the manifold given by L_{ψ_1} is*

$$L_{\psi_1|\psi_2,\ldots,\psi_m} = \left\{ x | \psi_1(x) = 0 \wedge \bigwedge_{i=2}^{m} \psi_i(x) < 0 \right\}.$$

A simple example in two dimensional space is shown in figure 6.2. The segment that is cut out of L_{ψ_1} is situated in the grey region defined by the area where $\psi_2 < 0$, i.e. the inside of the object defined by L_{ψ_2}. The boundaries of the curve segment are points, i.e. 0-manifolds, and are marked by black dots.

6.3.5 A unified implicit representation via an unsigned distance function

So far we have defined more complex objects via a number of implicit functions as proposed in [9]. That lead to vector valued functions. The other option would be to define a single scalar function that has the same zero level set [3, 22]. Note that we usually get the zero level set as an input and have to construct a proper implicit representation around it. The advantage of a single scalar function would be that we do not have to treat all sorts of special cases. We could imagine a number of

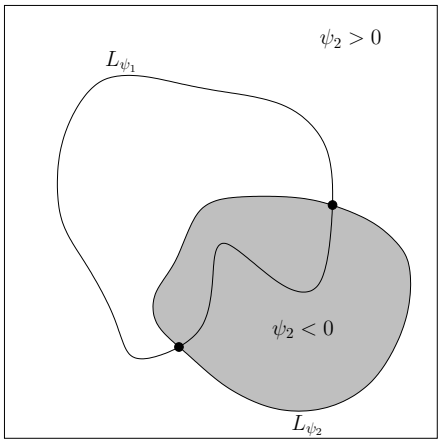


Figure 6.2: It is illustrated how an open curve is created from a closed one. The closed curve is given by the zero level set of ψ_1. The open curve is given by the segment that is restricted to the inside of a barrier manifold defined by the zero level set of ψ_2. The intersection of both level sets are points which are also the boundary of the open curve.

combinations between the sort of objects described in section 6.3.3 and 6.3.4, i.e. an object that is given by the level set

$$L_{\psi_1,\dots,\psi_j|\psi_{j+1},\dots,\psi_m} := \{x|\forall i \in [1,j] : \psi_i(x) = 0 \wedge \forall i \in [j+1,m] : \psi_l(x) < 0\} .$$

Not only that we have to deal with all those m functions but also we likely have to design a distinct distance functional for each situation.

As pointed out in section 6.3.3 it is not possible to define a signed distance function in such a case. It is not even possible for non-closed objects with codimension one. While we are able to define a signed distance function in a narrow band orthogonal to the segment, that is not possible for the higher codimension boundary, e.g. the end points of an open curve. Obviously we would also have to represent those, to be able to properly track them.

The simple idea is to use an unsigned distance function instead. The minimal distance of any point in the domain to the zero level set is well defined. Apart from the advantages of such an approach with respect to flexibility and reusability there are also some disadvantages that should not go unnoticed. Firstly, we loose any notion of inside and outside that is still present in the representation of non-closed objects in section 6.3.4. Secondly there is a "kink" in the function at the zero level set, i.e. the function is not differentiable exactly at the location of the represented shape. The remedy proposed by Ambrosia and Soner[3] and Giorgi[22] is to use the squared distance function instead. Another option frequently encountered is to replace the distance function with a differentiable approximation around zero [40].

For the definition of a distance functional it is only important that the function decreases monotonously towards the zero level set. On the discretized grid those kinks are smeared out by the discretization. In the worst case the derivative is zero, which can also happen when we use a smooth version of the distance function. Due to construction, the derivative close to the zero level set will nevertheless always be in $[-1, 1]$ modulo h. We can expect the method is well behaved in some sense, because the functional is regularized. Like for the images where the areas of constant gray values are moved along with the edges that drive the registration, the regularization will recover the correct solution even if some derivatives are not approximated correctly. In the classical level set method the regularization by curvature is used to obtain smooth viscosity solutions even in the presence of kinks in the implicit functions[40, 43].

6.4 Distance functions

The representations introduced in the previous section allow us to represent all kinds of possible shapes. Through the transformation of the representation we also transform the represented shape. In order for these representations to drive the registration process we have to define suitable distance functionals. In this section we propose such distance functionals.

6.4.1 Distance function for closed shapes

A straightforward approach would be to minimize the area inside ψ_T that is outside ψ_R and vice versa. This leads to the symmetric distance

$$\begin{aligned} \mathcal{D}_{\psi}^{\text{Area}} &= \int_{\Omega} H(\psi_T(x - u(x)))(1 - H(\psi_R(x)) \\ &\quad +(1 - H(\psi_T(x - u(x))))H(\psi_R(x))\mathrm{d}x \end{aligned} \tag{6.2}$$

where

$$H(x) = \begin{cases} 0, & \text{for } x < 0 \\ 1, & \text{otherwise} \end{cases}$$

is the Heaviside-function. This distance function has already been proposed in [35]. When ψ represents a closed object $1 - H(\psi)$ produces an image with a constant value of 1 for the interior and 0 for the outside. In that case the SSD distance (see

equation (2.4)) is component-wise equivalent to (6.2) because

$$\begin{aligned} & H_{\psi_T}(1 - H_{\psi_R}) + (1 - H_{\psi_T})H_{\psi_R} \\ \Leftrightarrow \quad & H_{\psi_T} - 2H_{\psi_T}H_{\psi_R} + H_{\psi_R} \\ \overset{H(x)=0\wedge 1}{\Leftrightarrow} \quad & (H_{\psi_T})^2 - 2H_{\psi_T}H_{\psi_R} + (H_{\psi_R})^2 \\ \Leftrightarrow \quad & (H_{\psi_R} - H_{\psi_T})^2 = ((1 - H_{\psi_T}) - (1 - H_{\psi_R}))^2 \end{aligned}$$

with $H_{\psi_T} := H(\psi_T(x - u(x)))$ and $H_{\psi_R} := H(\psi_R(x))$. The second step is only possible because H does only assume the values 0 and 1.The derivative of $\mathcal{D}_{\psi}^{\text{Area}}$ and $\mathcal{D}_{H(\psi)}^{\text{SSD}}$ with respect to u are

$$\begin{aligned} \frac{\partial \mathcal{D}_{\psi}^{\text{Area}}}{\partial u} &= -\int_{\Omega} \delta(\psi_T(x - u(x)))\nabla\psi_T(x - u(x))) \cdot \\ &\quad (2H(\psi_R(x)) - 1)\mathrm{d}x \qquad (6.3) \\ \frac{\partial \mathcal{D}_{H(\psi)}^{\text{SSD}}}{\partial u} &= -\int_{\Omega} \delta(\psi_T(x - u(x)))\nabla\psi_T(x - u(x))) \cdot \\ &\quad (H(\psi_T(x - u(x)) - H(\psi_R(x))))\mathrm{d}x. \qquad (6.4) \end{aligned}$$

In correspondence with chapter 3 this is also called the negative force. It is easily verified that both terms are equivalent up to a zero measure. For Computational purposes $\delta(x)$ and $H(x)$ have to be replaced by suitable approximations $\delta_a(x)$ and $H_a(x)$, with $\delta_a = H_a(x)$. This type of distance functional only makes sense when the template and reference overlap. The parts of the boundary of the template that are inside the reference are moved outwards until they coincide with the boundary of the reference. The parts of the template boundary that are on the outside of the reference are moved inwards. If there is no part of the reference inside the template this process continues until the area inside the template is zero. The reason for this behavior is that $-\partial\mathcal{D}_{\psi}^{\text{Area}}/\partial u$ and $-\partial\mathcal{D}_{\mathcal{H}(\psi)}^{\text{SSD}}/\partial u$ are only local descent directions. Moving the template in the direction of the reference would not change the distance functional until some part of them overlaps. Hence it is not a local descent direction. For these reasons one has also to be careful when narrow-band signed distance functions are used. Outside the band the difference in the distance is zero and the gradient, too. Thus, only the overlap of the narrow band around the zero level set is relevant for the computation of the forces.

6.4.2 Distance function for shapes of arbitrary codimension

One could argue that we could simply minimize the distance functional from section 6.4.1 for each of the m codimension one manifolds used to construct the codimension m manifold. Since all manifolds are subject to the same transformation the intersection is also subject to that transformation. The problem with that lies in the fact that we do not only want to match the object represented by this intersection, but also other objects, either represented by the zero level set of other implicit functions, or image data. The only thing we require of the codimension one manifolds used in the construction is that they intersect to produce the shape we would like to represent. In their single form they do not represent any object in the image. Hence minimizing the distance between those codimension one objects would lead to transformations around their intersection that have no relation to what we would like to achieve. It might only be meaningful in a small region around the intersection.

The difference between the intersection of the template and reference can be described by the integral

$$\mathcal{D}_{\psi}^{\text{Arbi}} := \frac{1}{2}\int_{\Omega}\left(\prod_{i=1}^{m}\delta(\psi_{T_i}(x-u(x))) - \prod_{i=1}^{m}\delta(\psi_{R_i}(x))\right)^2 \mathrm{d}x, \tag{6.5}$$

which is zero when the intersection of the ψ_{T_i} and the intersection of the ψ_{R_i} are identical and nonzero otherwise. The derivative with respect to u is given by

$$\begin{aligned}\frac{\partial \mathcal{D}_{\psi}^{\text{Arbi}}}{\partial u} \quad = \quad & \int_{\Omega}\left(\prod_{i=1}^{m}\delta(\psi_{T_i}(x-u(x))) - \prod_{i=1}^{m}\delta(\psi_{R_i}(x))\right)\cdot \\ & \left[-\sum_{i=1}^{m}\delta'(\psi_{T_i}(x-u(x)))\nabla\psi_{T_i}(x-u(x))\prod_{\substack{j=1\\ j\neq i}}\delta(\psi_{T_i}(x-u(x)))\right]\mathrm{d}x.\end{aligned} \tag{6.6}$$

The second term of (6.6) is a linear combination of the normal vectors of the template codimension one manifolds. The final product term restricts it to the intersection. Like in (6.3) this definition only makes sense when the representation of the template and the reference overlap. Again we have to work with an approximation δ_a. The support of δ_a should be chosen such that the representations overlap.

We have made extensive use of the fact that ψ is a distance function here. The

surface integral of ψ over the zero level set would usually be

$$\int_\Omega \delta(\psi(x))|\nabla\psi(x)|\mathrm{d}x. \tag{6.7}$$

When ψ is a distance function $|\nabla\psi| \equiv 1$ except for a few degenerate points. Hence, we can use the formulation

$$\int_\Omega \delta(\psi(x))\mathrm{d}x \tag{6.8}$$

instead. This trick is frequently used in the classical level set methods. The unit normal and curvature

$$\begin{aligned} N &:= \frac{\nabla\psi}{|\nabla\psi|}, \\ \kappa &:= \nabla\cdot\left(\frac{\nabla\psi}{|\nabla\psi|}\right), \end{aligned}$$

simplify to

$$\begin{aligned} N &:= \nabla\psi, \\ \kappa &:= \Delta\psi. \end{aligned}$$

As a result one does not have to deal with $|\nabla\psi|$ in the denominator which is always icky because the denominator may be zero. As ψ_T is transformed it might not be a distance function anymore. We are mainly interested in the direction of $\nabla\psi_T$, but when the gradients become too steep that is a problem and the implicit function has to be reinitialized to the zero level set.

6.4.3 Distance function for non-closed shapes

The surface integral over the non-closed object is given by

$$\int_\Omega \delta(\psi_1(x))\prod_{i=1}^{m} H(\psi_i(x))\mathrm{d}x.$$

The product term extracts the relevant part of the zero level set of ψ_1. Hence the distance between reference and template representation can be expressed by

$$\mathcal{D}_\psi^{\mathrm{Barrier}} := \frac{1}{2}\int_\Omega \left(\delta(\psi_{T_1}(x-u(x)))\prod_{i=2}^{m} H(\psi_{T_i}(x-u(x)))\right.$$

$$- \delta(\psi_{R_1}(x)) \prod_{i=2}^{m} H(\psi_{R_i}(x)) \Bigg)^2 \mathrm{d}x.$$

The associated force is then given by

$$\begin{aligned} \frac{\partial \mathcal{D}_\psi^{\text{Barrier}}}{\partial u} &= \int_\Omega \left(\delta(\psi_{T_1}(u)) \prod_{i=2}^{m} H(\psi_{T_i}(u)) - \delta(\psi_{R_1}) \prod_{i=2}^{m} H(\psi_{R_i}) \right) \cdot \\ &\quad \Bigg[-\delta'(\psi_{T_1}(u)) \nabla \psi_{T_1}(u) \prod_{i=2}^{m} H(\psi_{T_i}(u)) \\ &\quad -\delta(\psi_{T_1}(u)) \left(\sum_{i=2}^{m} \delta(\psi_{T_i}(u)) \nabla \psi_{T_i}(u) \prod_{\substack{j=2 \\ i \neq j}}^{m} H(\psi_{T_j}(u)) \right) \Bigg] \mathrm{d}x, \end{aligned}$$

where the x argument was omitted for the sake of readability.

The support of the approximation δ_a should be large enough for both representations to touch. Like for the objects with arbitrary codimension a separate registration of the ψ_{T_i} to the ψ_{R_i} is not an option because the functions with indices $2 \leq i \leq m$ only define the object boundary and do not represent any objects in the image. Furthermore the part of ψ_{T_1} and ψ_{R_1} that lies outside the barrier does not necessarily have any relation to an object in the image either.

6.4.4 Distance function for the unified approach

For the unified approach the surface integral is given by (6.8). The distance functional then is

$$\mathcal{D}_\psi^{\text{Unified}} = \int_\Omega \left(\delta(\psi_T(x - u(x))) - \delta(\psi_R(x)) \right)^2 \mathrm{d}x. \tag{6.9}$$

Although we have only one function for each representation we cannot minimize some area as in (6.2) because ψ_T and ψ_R are not signed distance functions. The derivative with respect to u is given by

$$\begin{aligned} \frac{\partial \mathcal{D}_\psi^{\text{Unified}}}{\partial u} &= -\delta'(\psi_T(x - u(x))) \nabla \psi_T(x - u(x)) \\ &\quad \cdot \left(\delta(\psi_T(x - u(x))) - \delta(\psi_R(x)) \right). \end{aligned} \tag{6.10}$$

As mentioned in section 6.3.5 has ψ_T to be a smooth approximation to the distance function that is differentiable around the interface. Concerning the approximation of δ the same as for the other distance functionals applies. From the formulation

some of the advantages of the unified approach become more obvious. We only need ψ_T and ψ_R in a region around the zero level set. Additionally the number of necessary function evaluations is less than for the other functionals. That leaves us with a smaller memory footprint and faster computation times. That all comes at the cost of the disadvantages already discussed in section 6.4.4.

6.4.5 Approximations for H and δ

There are a couple of popular approximations for $H(x)$ and consequently also $\delta(x) := \partial H(x)/\partial x$. One is the sigmoid or logistic function.

$$\begin{aligned} H_a^{\text{sig}}(x) &:= \frac{1}{1+e^{-x/a}} \\ \delta_a^{\text{sig}}(x) &:= \frac{a^{-1}e^{-x/a}}{(1+e^{-x/a})^2} \end{aligned}$$

A nice property is the fact that the derivatives δ_a^{sig} and $\delta_a^{\text{sig}'}$ can be expressed in terms of H_a^{sig}, i.e.

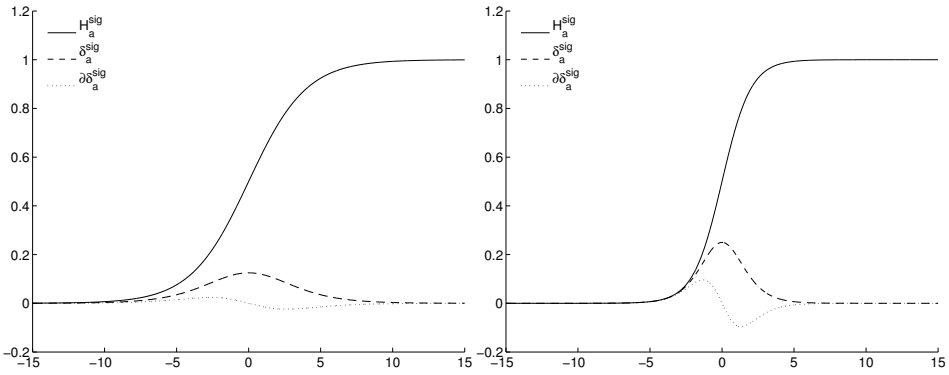
$$\begin{aligned} \delta_a^{\text{sig}}(x) &= a^{-1}H_a^{\text{sig}}(x)(1-H_a^{\text{sig}}(x)), \\ \delta_a^{\text{sig}'}(x) &= a^{-2}H_a^{\text{sig}}(x)(1-H_a^{\text{sig}}(x))(1-2H_a^{\text{sig}}(x)). \end{aligned}$$

Hence, we always only need one evaluation of the exponential function. The parameter s determines the slope of H_a^{sig} and thus also the slope of δ_a^{sig}.

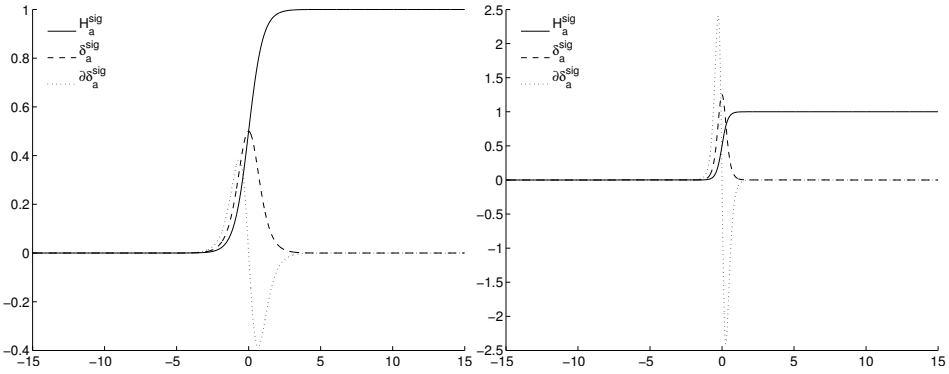
Another approximation that has been used e.g. in implicit active contours [11] is

$$\begin{aligned} H_a^{\text{atan}}(x) &:= \frac{1}{2}\left(1+\frac{2}{\pi}\arctan\left(\frac{x}{a}\right)\right), \\ \delta_a^{\text{atan}}(x) &:= \frac{a}{\pi(a^2+x^2)}. \end{aligned}$$

Both aforementioned approximations have global support and are C^∞. For computational and storage considerations we would like to use narrow band distance functions. That means we could use an approximation with compact support. Furthermore functions that are C^2 would be sufficient. In [52] the following approximation C^2 is proposed:



(a) $a = 2$ (b) $a = 1$



(c) $a = 0.5$ (d) $a = 0.2$

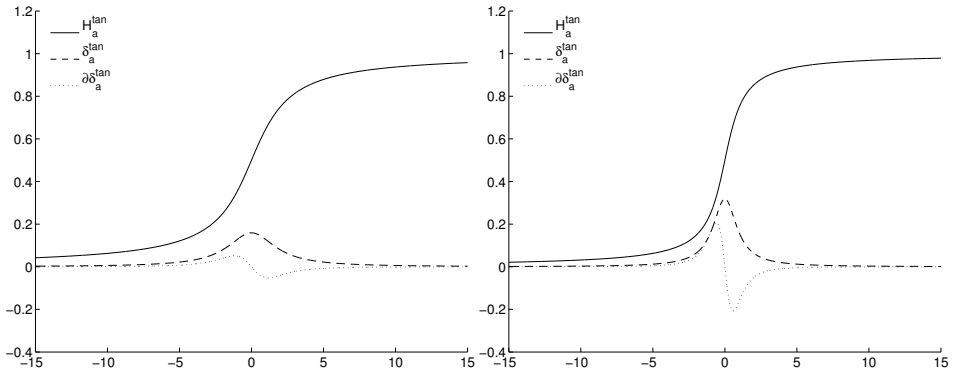
Figure 6.3: Plots of approximations via sigmoid functions for different values of the scaling parameter a. H_a^{sig} is plotted as a solid line, δ_a^{sig} as a dashed line, and $\delta_a^{\text{sig}'}$ as a dotted line.

$$H_a^{\sin}(x) := \begin{cases} \frac{1}{2}\left(1 + \frac{x}{a} + \frac{1}{\pi}\sin\left(\frac{\pi x}{a}\right)\right) & \text{, if } |x| \leq a \\ 1 & \text{, if } x > a \\ 0 & \text{, if } x < -a \end{cases},$$

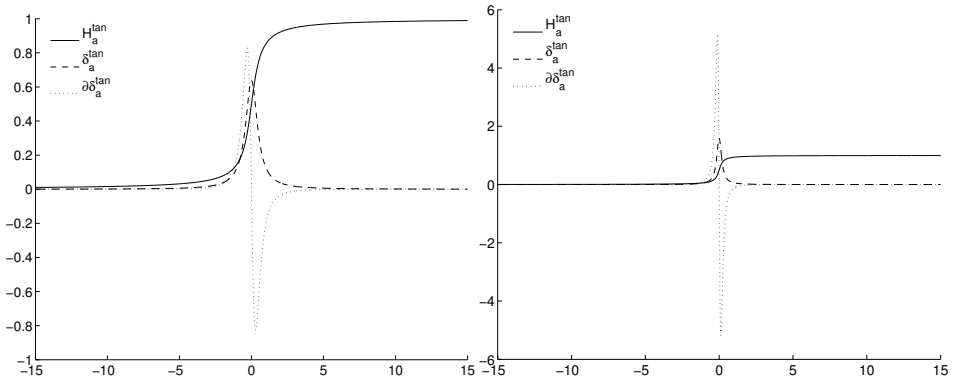
$$\delta_a^{\sin}(x) := \begin{cases} \frac{1}{2a}\left(1 + \cos\left(\frac{\pi x}{a}\right)\right) & \text{, if } |x| \leq a \\ 0 & \text{, otherwise} \end{cases}.$$

When we just need δ and derivatives thereof as e.g. in (6.10) we could also use radial basis functions with compact support. Usually the distance criterion is built into the radial basis function, i.e. a center and another point are the input arguments. In our case the distance is already given by the distance function ψ. In case of signed distance functions we have to use $|\psi|$. We use the radial basis functions proposed by Wendland in [49]. Specifically the C^2 function

$$\delta_a^{\text{Wend}}(x) = \frac{3}{2}(1 - |x|/a)_+^4(4|x|/a + 1),$$



(a) $a = 2$ (b) $a = 1$



(c) $a = 0.5$ (d) $a = 0.2$

Figure 6.4: Plots of approximations via H_a^{atan} for different values of the scaling parameter a. H_a^{atan} is plotted as a solid line, δ_a^{atan} as a dashed line, and $\delta_a^{\text{atan}\prime}$ as a dotted line.

where

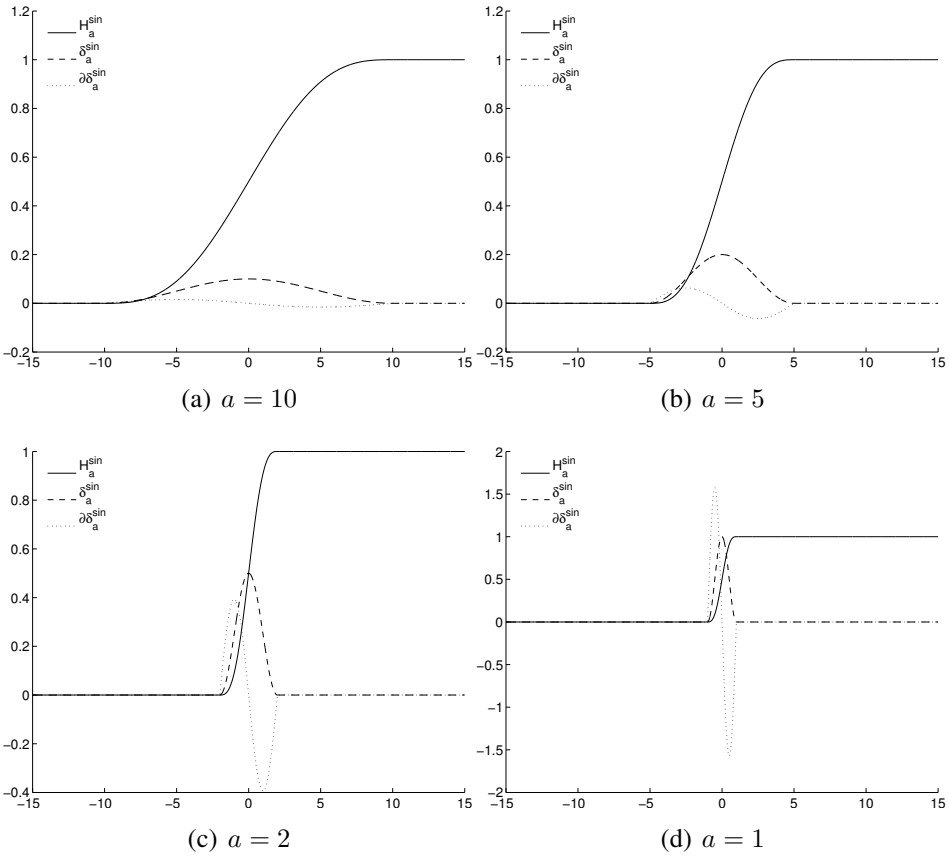
$$(y)_+ = \begin{cases} y & \text{, if } y > 0 \\ 0 & \text{otherwise} \end{cases} .$$

Obviously $\delta_a^{\text{Wend}} \equiv 0$ if $x > a$. Hence a is the maximal relevant distance to the zero level set and also the bandwidth of the narrow band distance function.

6.5 Integration into the registration process

With the distance functionals defined in the previous section we can now expand E_α (see equation (2.3)). We simply add the distance functionals for the additional constraints and obtain the new functional

$$E_\alpha^\psi(u) := \mathcal{D}^{\text{SSD}} + \sum_{i=1}^{l} \gamma_i \mathcal{D}_\psi^{[\text{Area}|\text{Arbi}|\text{Barrier}|\text{Unified}]} + \alpha\mathcal{R}. \tag{6.11}$$



(a) $a = 10$ (b) $a = 5$

(c) $a = 2$ (d) $a = 1$

Figure 6.5: Plots of approximations via sine and cosine functions for different values of the scaling parameter a. $H_a^{\sin}$ is plotted as a solid line, $\delta_a^{\sin}$ as a dashed line, and $\delta_a^{\sin\prime}$ as a dotted line.

The coefficients γ_i are weighting factors that determine the influence of each of the l structural constraints on the registration procedure. The derivation of the normal equations is done via Taylor expansion of the additional terms as in chapter 3. Note that all the proposed distance functional are quadratic and hence we can use the Gauss-Newton approach with all of them.

CHAPTER 7

Results for shape constraints

In this chapter we give examples for the various representations and distance functional introduced in the previous chapter. There is one exception, the closed shapes from section 6.4.1 are not considered here. That is because of the similarity of the distance functional to the original SSD functional used with images containing objects with constant gray value (see the results in section 5.3. At the end we revisit the motivational example from section 6.1.1 (see figure 6.1) and demonstrate the benefit of additional shape constraints there.

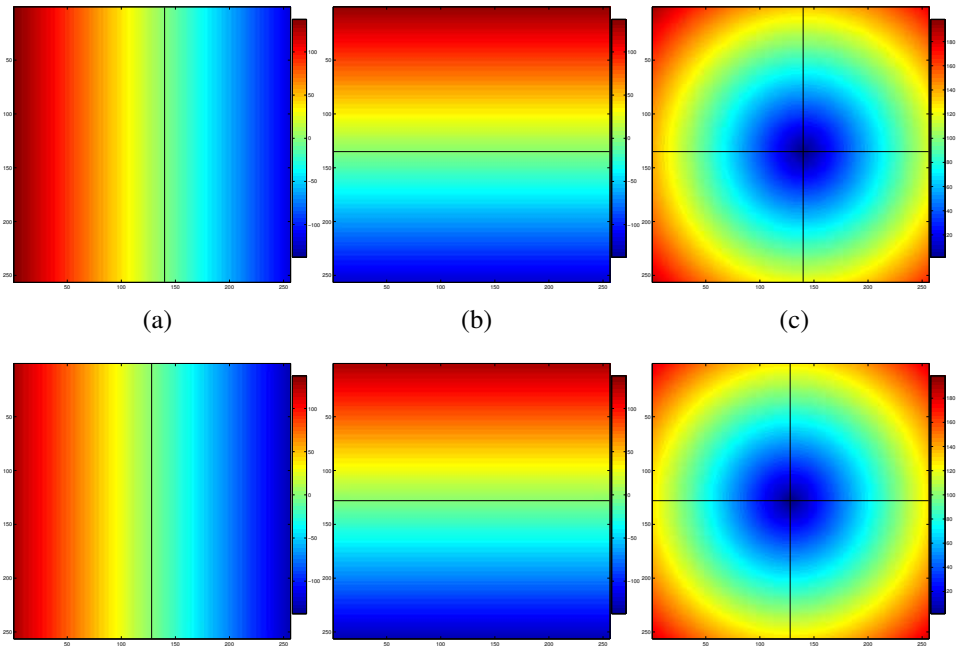
7.1 Shapes of arbitrary codimension

The simplest example for an object of arbitrary codimension is a point in two-dimensional space. Although this is a case that is easily covered by other techniques because the correspondence is trivial it is well suited to illustrate various aspects. In section 7.2 the same example will be used to illustrate the influence of the approximation parameter a. We use a 256×256 image. The template point is located at $(140, 135)$, and the reference point is located at $(128, 128)$.

When we use $\mathcal{D}_{\psi}^{\text{Aribtrary}}$ the points are modeled by two distance functions. The point is given by the intersection of their zero level sets. The functions are shown in the left and middle column of figure 7.1. The template functions are displayed in the top row and the reference functions are displayed in the bottom row. The zero level sets are indicated by a black line.

When we use $\mathcal{D}_{\psi}^{\text{Unified}}$ the points are modeled by a single distance function which is shown in the right column of figure 7.1. The zero level set which is just one point is indicated by the intersection of two black lines.

The resulting transformations are shown in figure 7.2. The template point is indicated by a black cross, the reference point by a gray cross. In the bottom row a magnification of the region around the points is shown. The approximation parameter a was $2/3$ and three levels in the multiresolution framework were used.



(a) (b) (c)

(d) (e) (f)

Figure 7.1: The top row shows the distance functions for the template point $(140, 135)$. The bottom row shows the distance functions for the reference point $(128, 128)$. The left and middle columns show the two distance functions used for $\mathcal{D}_\psi^{\text{Arbi}}$. The zero level set is indicated by a black line. In the right column the distance function used in $\mathcal{D}_\psi^{\text{Unified}}$ is displayed. The zero level set is indicated by the intersection of two black lines.

The representations of the shape constraints were treated like image data in the multiresolution framework. Since we deal with only one point we avoid the correspondence problem and the exact transformation u^* can be easily calculated from the location of the template and reference point.

$$\begin{aligned} 140 &= 128 - u_1^*(128,128)/h \;\Rightarrow\; u_1^*(128,128) = 12h \\ 135 &= 128 - u_2^*(128,128)/h \;\Rightarrow\; u_2^*(128,128) = 7h \end{aligned}$$

The computed results for the reference point location $(128, 128)$ are given in table 7.1 along with the relative error with respect to the expected transformation u^*. The relative error is in the order of 10^{-3}. Remember that we do not register the points but their representations and usually have no knowledge about the correspondence. Hence we have to resign to such simple examples for an analysis. In figure 7.3 the transformed zero level sets for $\mathcal{D}_\psi^{\text{Arbi}}$ are visualized. The zero level sets are at the interface from black to white. The left and the middle image show them separately. The two images are combined in the right image. The images also show the decay of the transformation away from the reference point. The strongest

	$u(128,128)/h$		relative error	
	u_1/h	u_2/h	$\lvert u_1 - u_1^*\rvert/u_1^*$	$\lvert u_2 - u_2^*\rvert/u_2^*$
$\mathcal{D}_\psi^{\text{Arbi}}$	11.9779	6.9939	$1{,}846 \cdot 10^{-3}$	$8{,}717 \cdot 10^{-4}$
$\mathcal{D}_\psi^{\text{Unified}}$	12.0221	7.0170	$1{,}839 \cdot 10^{-3}$	$2{,}426 \cdot 10^{-3}$

Table 7.1: Transformation vectors for point example at location of the reference point $(128, 128)$ for two representation strategies. The transformation vector normalized to image pixels is given in the left column. The relative error with respect to the actual transformation is given in the right column.

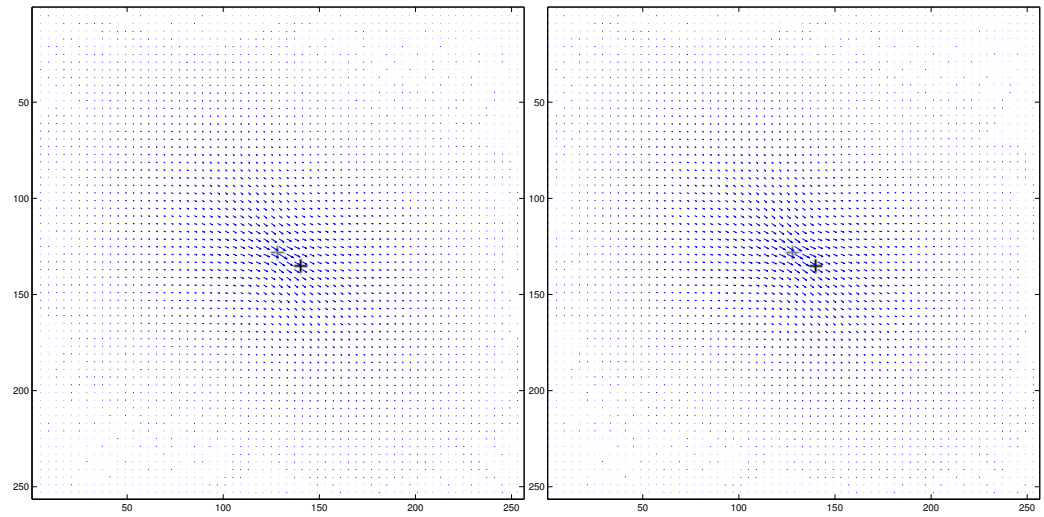
deformations are at the location of the reference point. In the next section we use the example from this section to investigate the influence of the approximation parameter a.

7.2 Influence of the approximation parameter a

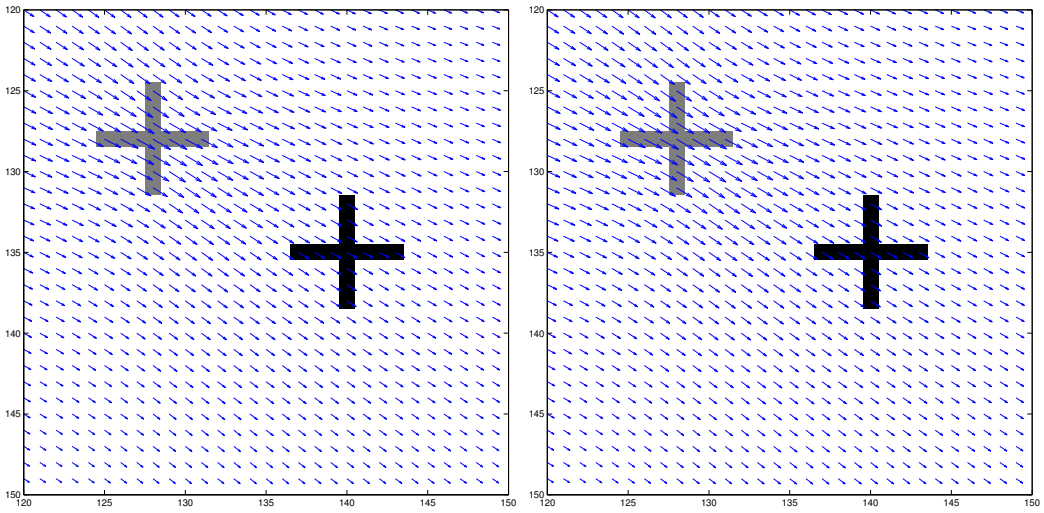
The parameter a stretches the approximations H_a and δ_a for $a > 1$ and does the adverse for $a < 1$. For approximations with compact support this is equivalent to varying the size of the support. We argued that a has to be chosen such that the representations of the reference and template overlap, because otherwise the template is not transformed in relation to the reference. Obviously it should not be too large because then the functions ψ are registered to each other and not the only the object they represent. We tested various choices of a for the point matching example from section 7.1 with $\mathcal{D}_\psi^{\text{Unified}}$. We chose this example because only one function is involved and the correspondence is known. This "simplicity" reduces the number of possible side effects. For the computations we used three levels in the multiresolution approach.

The results are displayed in figures 7.4 and 7.5. Figure 7.5 is a zoomed in version of figure 7.4. The values for a are 4, 1, and $\frac{2}{3}$. In the left column the length of transformation vectors, normalized to 1 with respect to the maximal length of the transformation vector, is shown. The right column displays the contour lines of the difference between the transformed template function ψ_T and the reference function ψ_R. A cross indicates the position of the reference point.

We start with the left column. The length of the transformation vectors decays away from the reference points. The blob has the shape of an ellipse where one principal axis is in the direction of the template point and the other orthogonal to it. This is what we expect since the isocontours of $\delta_a(\psi)$ are circles when ψ is a

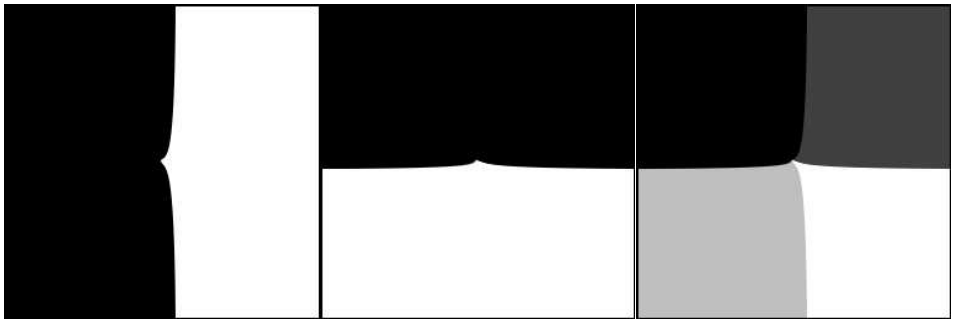


(a) Arbitrary (b) Unified



(c) Arbitrary (d) Unified

Figure 7.2: Results for the registration of a point. The transformation is displayed along with the template and reference point which are indicated by a cross. The black cross corresponds to the template point and the gray cross corresponds to the reference point. On the left is the result for $\mathcal{D}_{\psi}^{\mathrm{Arbi}}$ and on the right the result for $\mathcal{D}_{\psi}^{\mathrm{Unified}}$. The bottom row is a magnification of the relevant part of the images from the top row.



(a) (b) (c)

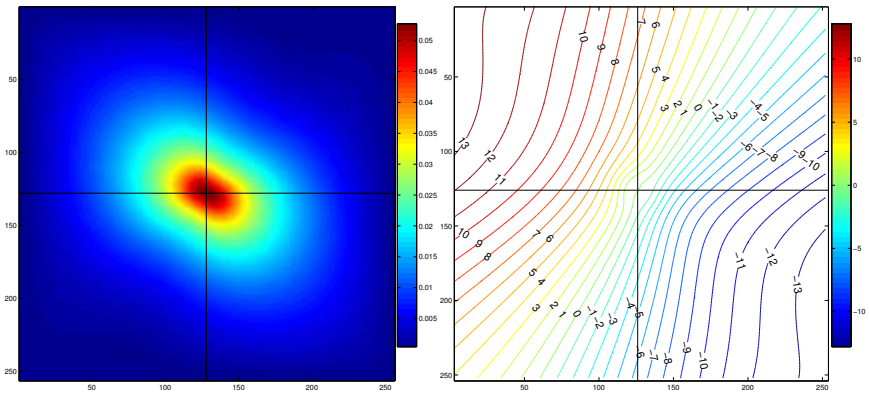
Figure 7.3: Transformed interface for the point example with $\mathcal{D}_{\psi}^{\mathrm{Arbi}}$. The left and middle image show the transformed zero level sets separately, and the right image shows a combined view. The zero level set is at the interface from black to white.

distance function. The rate of decay depends on the choice of a, because the rate of decay of the force away from the reference point also directly depends on the choice of a. As we increase a we also increase the area around the zero level set of ψ_T and ψ_R that is relevant for the registration process. A larger part of ψ_T is matched to ψ_R, where the influence depends on the value of δ_a.

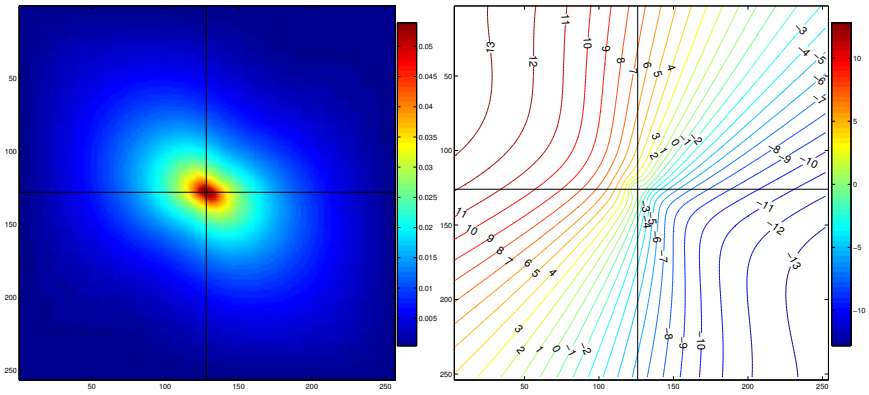
This effect is also visible in the contour plots in the right column. Since ψ_T and ψ_R correspond to different points in space they are only equal on the line that intersects the connection between the two points orthogonally at half the points distance. After registration the difference between ψ_T and ψ_R should be zero at least in the reference point. The difference around the reference point indicates how much of the surrounding region has been matched. For large a the isocontours of the difference should be more widely spaces than for small values of a. Away from the reference point the variation in the results for different a should diminish. The latter effect can be observed in figure 7.4. The former is more obvious in the zoomed in plots in figure 7.5. As the size of the region that is strongly transformed decreases the difference in the representing functions increases as indicated by the isocontours of the difference between ψ_R and the transformed ψ_T.

Here we have only shown for values of a for which the registration "worked". When a is chosen too small the representation of the template and reference do not overlap properly and the template representation is transformed subject to its own properties without any connection to the reference.

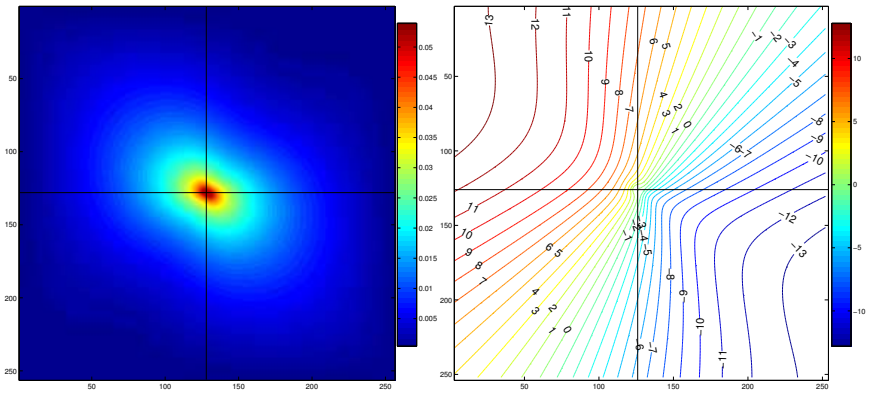
When a is chosen too large we see two effects. Firstly, as already indicated by the results shown in figure 7.4 and 7.5 the transformation is influenced by parts of the functions that are far away from their zero level sets. Secondly, the gradients are



(a) $a = 4$ (b) $a = 4$

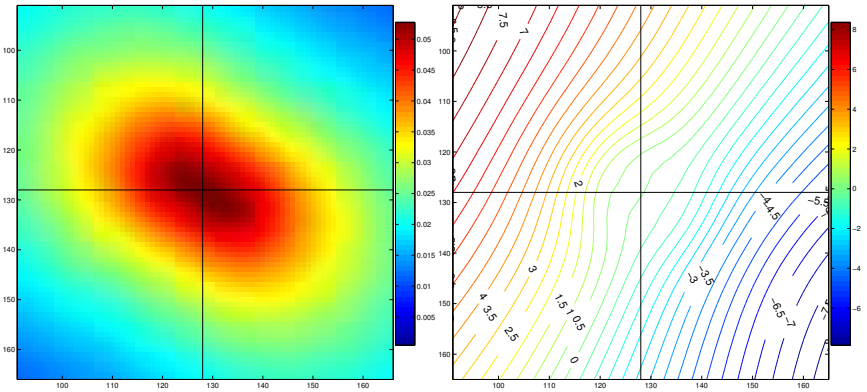


(c) $a = 1$ (d) $a = 1$

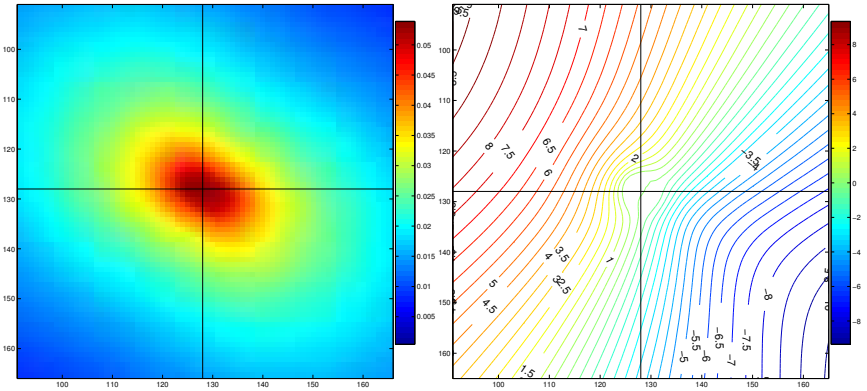


(e) $a = \frac{2}{3}$ (f) $a = \frac{2}{3}$

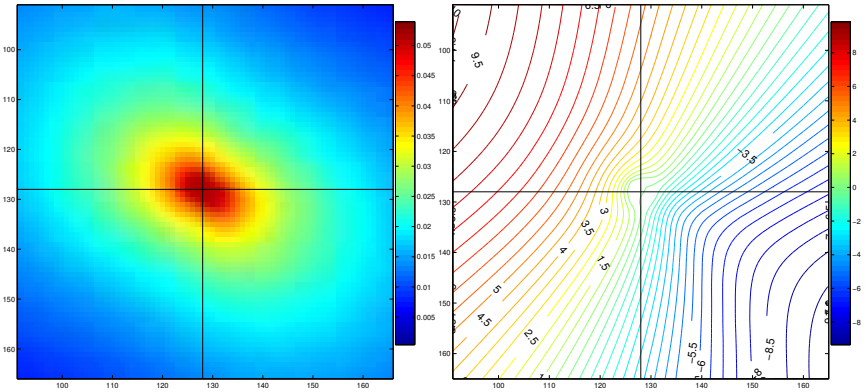
Figure 7.4: The influence of the approximation parameter a on the result is illustrated here. The left column displays the the length of the transformation vectors, normalized to 1 with respect to the maximal length, for the point matching example with $\mathcal{D}_{\psi}^{\text{Unfied}}$ with different values of a.In the right column the contour lines of the difference between the transformed template distance function ψ_T and the reference distance function ψ_R is shown. The position of the reference point is indicated by a cross. For higher values of a the region that influences the result is larger. Hence a larger region of the distance functions is well matched as indicated by the vector length and contour plots.



(a) $a = 4$ (b) $a = 4$



(c) $a = 1$ (d) $a = 1$



(e) $a = \frac{2}{3}$ (f) $a = \frac{2}{3}$

Figure 7.5: This figure is a zoomed in version of figure 7.4. It allows to better appreciate the differences for the various values of a. For a description please refer to figure 7.4.

less strong and the registration might stop prematurely. The transformation goes in the right direction, but its not large enough to match the representations well. When not only one, but also other objects and images data are registered, a value of a that is too large hinders the registration of these other objects and the image data, too.

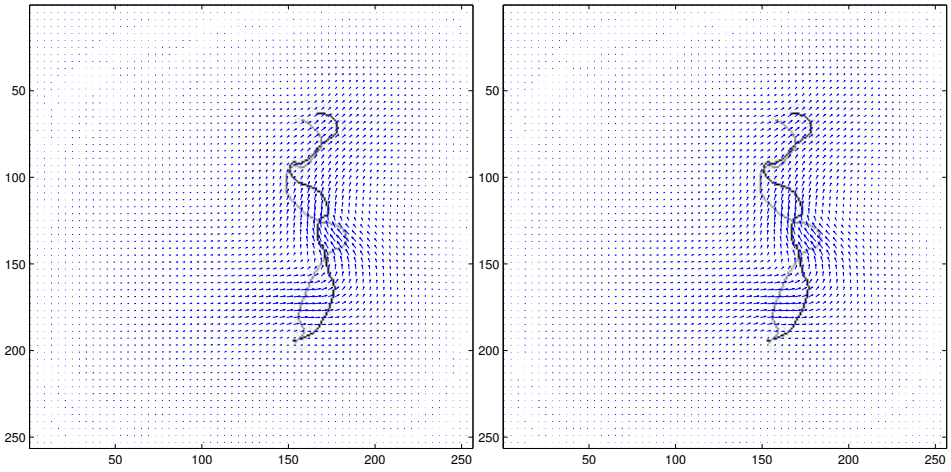
Hence, when a has to be chosen large because the distance between the represented objects is large, its value should be reduced as the distance decreases during registration. As a is decreased the registration focuses on the actual object. Varying a in the registration is a multiscale technique. An analogue that is sometimes used in the registration of images is the smoothing of the images with varying strength to first match coarse image components and then progress to finer features. This procedure can be used instead of or in conjunction with the multiresolution approach introduced in section 4.5.

7.3 Non-closed shapes

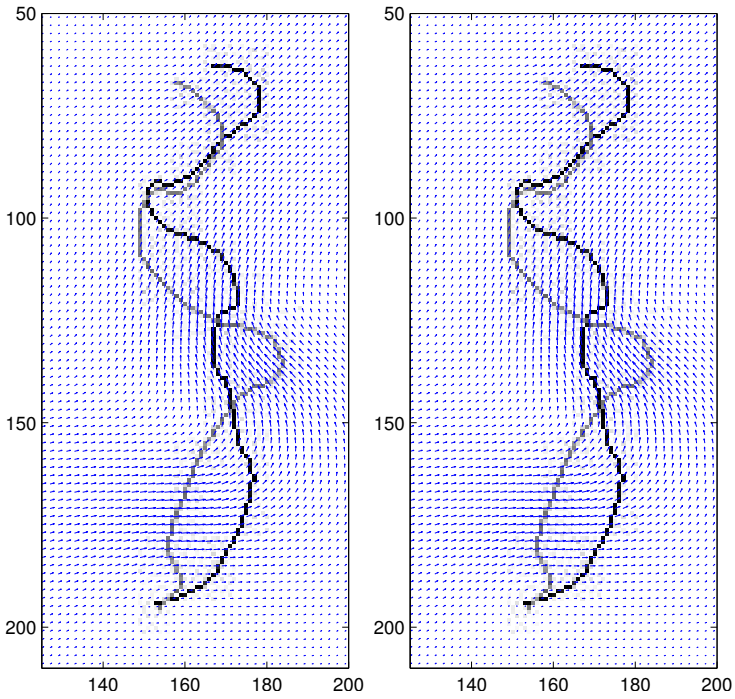
When we deal with non-closed shapes we have the choice between the barrier technique using additional "barrier functions" or we can use the unified representation. The simplest example for a non-closed shape is an open curve in two-dimensional space. The construction for the representation with barriers is already illustrated in figure 6.2. Hence, we do not show the corresponding distance functions here. The transformations for both representation strategies are displayed in figure 7.6. The black curve is the template curve and the gray curve is the reference curve. The bottom row is a magnification of the images in the top row in the region around the curves. The transformed template curve matches the reference curve almost exactly and and therefore an overlay with the recovered zero level set does not give any information. Hence, we abstained from such a visualization. Remember that the difference between the barrier and the unified representation is that that $\mathcal{D}_{\psi}^{\text{Barrier}}$ contains terms that explicitly enforce matching of the end points defined by the intersection of the zero level sets and that some notion of in- and outside in inherent. For the unified approach matching of the end points is due to regularity of the solution and there is not concept of in- and outside.

7.4 Combination of different shape constraints

We cannot only work with constraints of the same type but also use constraints of various types at the same time. We just have to add the appropriate functionals (see



(a) Barrier (b) Unified



(c) Barrier (d) Unified

Figure 7.6: Results for the registration of an open curve are shown. The transformation is displayed along with the template and reference curve. The black curve is the template curve and the gray curve is the reference curve. On the left is the result $\mathcal{D}_{\psi}^{\text{Barrier}}$ and on the right side the result for $\mathcal{D}_{\psi}^{\text{Unified}}$. The bottom row is a magnification of the relevant part of the images from the top row.

also equation (6.11)). Here we give an example of a combination of the constraints used in the sections above, the open curve and the point constraint. For the open curve we use $\mathcal{D}_\psi^{\text{Barrier}}$ and for the point constraint we use $\mathcal{D}_\psi^{\text{Unified}}$.

The computed transformation is displayed in figure 7.7(a) and figure 7.7(c) shows a magnified version of the relevant part. The black curve is the template curve and the gray curve is the reference curve. The position of the point constraint is indicated by a cross. The colors for template and reference are the same as for the curve. A contour difference plot like the ones used in section 7.2 is presented in figure 7.7(b) and a magnified version in figure 7.7(d).

When only one of the constraints was used the transformation extends smoothly into the space surrounding the transformed representation (see figure 7.4 and 7.6). In the combined case the transformation is influenced by the other constraint, too. In some situations the different representations even "compete". That becomes obvious when one compares figure 7.6(c) and 7.7(c). Where the point is close to the curve the transformation is clearly different from the one without the point. The point and the curve have to be moved in different directions. This is also indicated in the contour difference plots in figure 7.7(b),(d). While the zero contour still goes through the reference point the region towards the curve is clearly distorted by the transformation of the curve. That is not a problem since only the part close to the zero level set contributes to the registration process.

If however we would place the template point closer to the curve, registration in that region would be hindered by the conflicting directions in which the curve and the point have to be moved. Yet, this is not a problem specific to our approach, but a general problem in image registration. If locally discontinuous transformations are required one has to model the regularization term differently, e.g. use Total-Variation (TV) based regularization [21].

7.5 Combination of image data and shape constraints

The initial motivation for the development of the techniques presented above was the need for additional constraints on the registration of image data. Let us review the motivational example from section 6.1.1 (Figure 6.1). We performed the computations with the same parameters with the difference that this time we used an additional constraint for the critical region identified by the circle in figure 6.1. Here we used $\mathcal{D}_\psi^{\text{Barrier}}$. The results are displayed in figure 7.8. The leftmost column

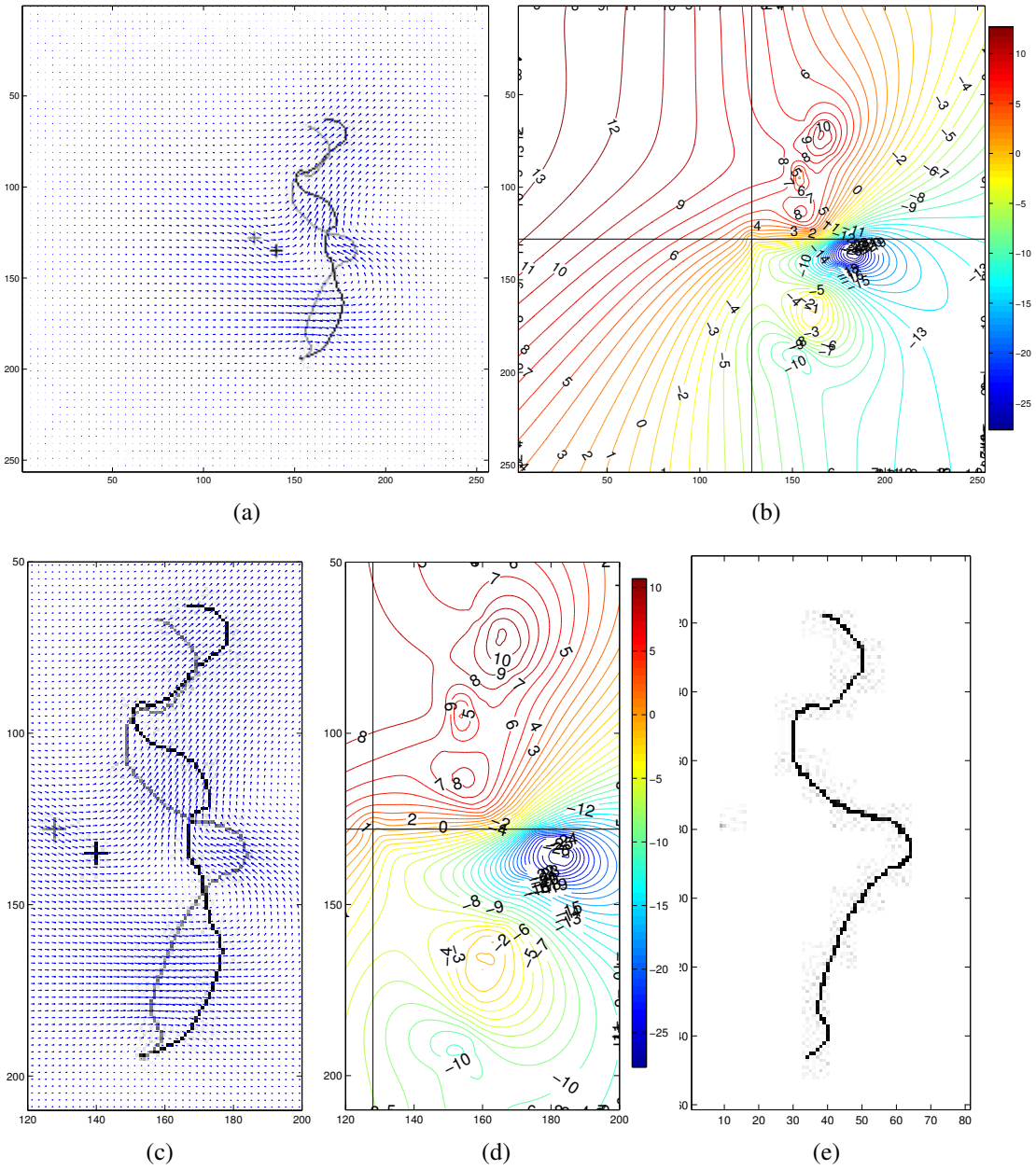
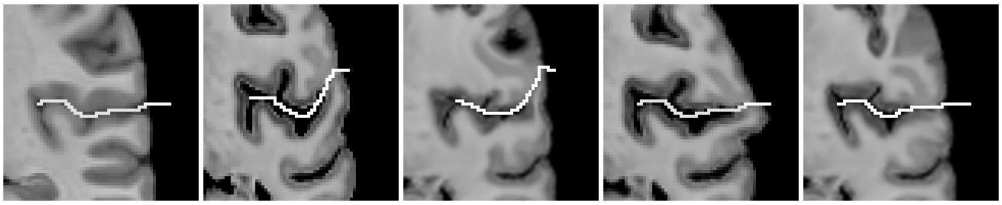


Figure 7.7: Result for the combination of two different types of constraints. It is a combination of the example in figure 7.6 and figure 7.4. For the point constraint the unified approach and for the open curve the barrier approach is used. The transformation is shown in (a). The black curve is the template curve and the light gray curve is the reference curve. Figure (b) shows the same contour difference plot used in section 7.2. The figures (c) and (e) are magnifications of the relevant parts of figures (a) and (b). The transformed template point and curve are shown in (e).

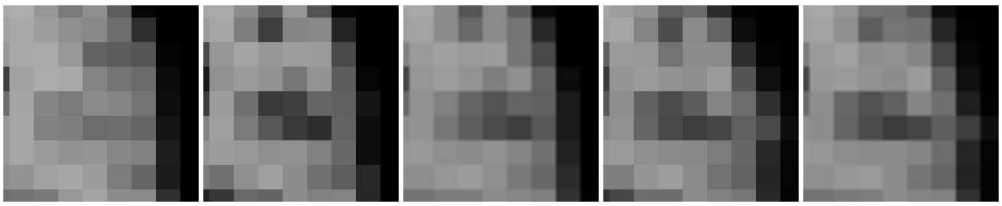
shows the reference, the second column the template. The zero level sets of the additional constraint are indicated by a white line. The result without the additional constraint is displayed in the middle column. Clearly the registration result is not what we would like to see. The result for just the additional constraint without the image data is displayed in the fourth column. The zero level sets are well matched, but the rest of the image is just "pulled along". In the rightmost column the result for the additional constraint in conjunction with the image data is shown. Now also the image features are matched. Remember that we cannot expect a perfect match since the images are slices from three-dimensional data sets and are brains of two different individuals. Nevertheless it is easier to visualize the effects in two dimension. The main issue was that a part of the temporal lobe was registered to the parietal lobe which is anatomically wrong. This problem has been resolved by the use of the additional constraint.



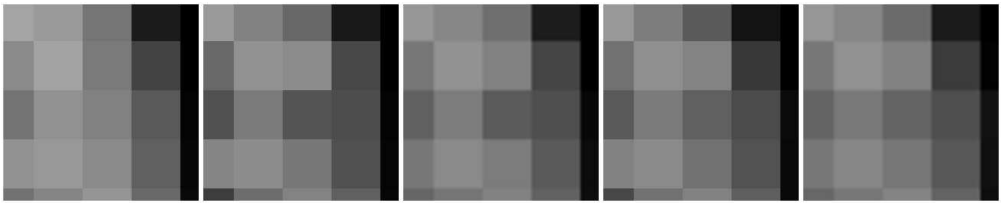
(a) Resolution 0

(b) Resolution 1

(c) Resolution 2



(d) Resolution 3



(e) Resolution 4

Figure 7.8: Final results for each stage in the multiresolution framework for the motivational example in figure 6.1. One of the critical parts is enlarged. The columns show from left to right, the reference image, the template image, the result when just the image difference is used, the result based just on the additional constraint, and the result based on the image difference and the additional constraint. On the finest resolution the chosen constraint is indicated by a white line.

CHAPTER 8

Conclusion

We introduced a inexact Newton method for the solution of the image registration problem based on image differences. The linearization of the non-linear functional results in a regularized Gauss-Newton method. To improve robustness and speed, and to facilitate the computation of large transformations a multiresolution framework is used. Regularization in each step of the outer iteration is controlled by a trust region strategy based on the Armijo-Goldstein rule. An efficient multigrid solver that deals with the jumping coefficients in the resulting system of linear equations has been proposed. In addition to the numerical investigation of the solver, we compared the regularized Gauss-Newton method with a related regularized gradient descent method. The only difference between the two methods are image dependent coefficients in the partial differential operator. We demonstrated that in certain situation the Gauss-Newton method is superior to the gradient descent method, and that that justifies the additional work that has to be invested.

In a second part we investigated the use of additional structural constraints on the registration process. These constraints were modeled with implicit functions which zero level sets represented the structural constraints. Distance functional that can be easily plugged into the existing intensity based minimization method have been proposed. The effectiveness has been shown in a number of examples.

There are a number of issues that have not been covered here. That especially concerns issues raised in the last part. We have said little about the efficient computation of distance functions, storage schemes, and reinitialization that might be necessary during registration. Extensive information on such issues can be found in [34, 39, 40, 43] and the references therein.

Another problem that will require further attention is the control of the influence of the additional constraints. In some situation when they are used to disambiguate they could possibly be switched of when one is close to solution. When they provide additional information that is not present in the images they should be kept active up to the end. One possibility might be to control the weighting parame-

ter with respect to the residual distance. A key problem is that size of the terms depends on the size and form of the constraint and the different representations scale differently. Thus, one probably has to tune the parameters relative the others. Proper experience is needed and problem dependent choices are inevitable. The goal of a control strategy should be to produce stable results over a large range of initial values.

APPENDIX A

Abbreviations and acronyms

AMG algebraic multigrid

CS correction scheme

CSF cortico spinal fluid

FAS full approximation scheme

FMG Full Multigrid Method

fMRI functional magnetic resonance imaging

GM gray matter

MRI magnetic resonance imaging

NURBS nonuniform rational B-splines

PET positron emission tomography

spd symmetric positive definite

WM white matter

CAD computer aided design

TV Total-Variation

SSD integral (sum) of squared differences

APPENDIX B

Nomenclature

Δ	vector valued Laplace operator	5
γ	cycle index	27
$\mathcal{D}$	distance functional	4
$\mathcal{R}$	regularization functional	4
$\mathcal{V}$	function space	8
$\Omega_{\mathbf{h}}^m$	discrete domain in m dimensions	16
d_h	defect (residual)	25
f	force [right hand side]	14
f_h	discrete force [right hand side]	17
H^1	Sobolev space of functions in L^2 with weak derivatives in L^2	12
H_0^1	subspace of H^1 consisting of the functions with zero boundary values	12
I_H^h	prolongation (interpolation) operator	26
I_h^H	restriction operator	26
L	Navier-Lamé operator [elliptic differential operator]	5
L^2	space of square integrable L^2-functions	12
L_h	discrete Navier-Lamé operator	18
M	partial differential operator $(\alpha+\beta)L + J_\theta^t J_\theta$	14
M_h	discrete partial differential operator $(\alpha+\beta)L_h + J_{h_\theta}^t J_{h_\theta}$	18
R	reference image	4

Bibliography

[1] Abbott EA. *Flatland: A romance of many dimensions*. Dover Publications, 1953, 6th edn.

[2] Alcouffe R, Brandt A, Dendy J, Painter J. The multigrid method for the diffusion equation with strongly discontinuous coefficients. *SIAM Journal on Scientific Computing* 1981; 2:430–454.

[3] Ambrosio L, Soner HM. Level set approach to mean curvature flow in arbitrary codimension. *Journal of Differential Geometry* 1996; 43(4):693–737.

[4] Amit Y. A nonlinear variational problem for image matching. *SIAM J. Sci. Comput.* 1994; 15:207–224.

[5] Amunts K, Zilles K. Advances in cytoarchitectonic mapping of the human cerebral cortex. *Anatomic Basis of Functional Magnetic Resonance Imaging* 2001; 11(2).

[6] Baumeister J. *Stable Solution of Inverse Problems*. Vieweg Advanced Lectures, 1987.

[7] Brett M, Leff A, Rorden C, Ashburner J. Spatial normalization of brain images with focal lesions using cost function masking. *Neuroimage* 2001; 14(2):486–500.

[8] Bro-Nielsen M, Gramkow C. Fast fluid registration of medical images. *Lecture Notes in Computer Science, Springer-Verlag* 1996; :267–276.

[9] Burchard P, Cheng LT, Merriman B, Osher S. Motion of curves in three spatial dimensions using a level set approach. *J. Comput. Phys.* 2001; 170(2):720–741. ISSN 0021-9991.

[10] Cachier P, Mangin JF, Pennec X, Riviere D, Papadopoulos-Orfanos D, Regis J, Ayache N. Multisubject non-rigid registration of brain MRI using intensity and geometric features. In W Niessen, M Viergever, eds., *MICCAI 2001*. Springer, 2001, vol. 2208 of *Lecture Notes in Computer Science*, 734–742.

[11] Chan TF, Vese LA. Active contours without edges. *IEEE Transactions on image processing* 2001; 10(2):266–277.

[12] Christensen G, Miller M, Vannier M, Grenander U. Individualizing neuroanatomical atlases using a massively parallel computer. *IEEE Computer* 1996; 29(1):32–38.

[13] Collins DL, Le Goualher G, Evans AC. Non-linear cerebral registration with sulcal constraints. In *MICCAI '98: Proceedings of the first international conference on medical image computing and computer-assisted intervention*. Springer, 1998, vol. 1496 of *Lecture Notes in Computer Science*, 974–984.

[14] Dendy J. Black box multigrid. *Journal of Computational Physics* 1982; 48:366–386.

[15] Dennis J, Schabel R. *Numerical methods for unconstrained optimization and nonlinear equations*. Prentice Hall, 1983.

[16] Droske M, Rumpf M. A variational approach to non-rigid morphological registration. *SIAM Appl. Math.* 2004; 64(2):668–687.

[17] Engl HW, Hanke M, Neubauer A. *Regularization of Inverse Problems*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1996.

[18] Fischer B, Modersitzki J. Combination of automatic non-rigid and landmark based registration: The best of both worlds. In M Sonka, J Fitzpatrick, eds., *Proceedings of the SPIE*. 2003, 1037–1048.

[19] Fischer B, Modersitzki J. Combining landmark and intensity driven registration. *PAMM* 2003; 3(1):32–35.

[20] Fischer B, Modersitzki J. Curvature based image registration. *JMIV* 2003; 18:81–85.

[21] Frohn-Schauf C, Henn S, Hömke L, Witsch K. Total variation based image registration. In XC Tai, KA Lie, T Chan, S Osher, eds., *Image processing based on partial differential equations*. Springer, 2006. Proceedings of the International conference on PDE-Based Image Processing and Related Inverse Problems, CMA, Oslo, August 8-12, 2005.

[22] Giorgi ED. Barriers, boundaries, motion of manifolds. Lectures held in Pavia, Italy 1994.

[23] Haber E, Modersitzki J. Numerical methods for volume preserving image registration. *Inverse Problems* 2004; 20:1621–1638.

[24] Hanke-Bourgeois M. *Grundlagen der Numerischen Mathematik und des Wissenschaftlichen Rechnens*. Teubner, Stuttgart · Leipzig · Wiesbaden, 2002.

[25] Hellier P, Barillot C. Coupling dense and landmark-based approaches for nonrigid registration. *IEEE Transactions on Medical Imaging* 2003; 22(2):217–227.

[26] Hemker P. On the order of prolongations and restrictions in multigrid procedures. *Journal of Computational and Applied Mathematics* 1990; 32(3):423–429.

[27] Henn S. A Levenberg-Marquardt scheme for nonlinear image registration. *BIT Numerical Mathematics* 2003; 43(4):743–759.

[28] Henn S, Hömke L, Witsch K. Lesion preserving image registration with applications to human brains. In CE Rasmussen, HH Bülthoff, B Schölkopf, MA Giese, eds., *Pattern Recognition: 26th DAGM Symposium, Tübingen, Germany*. Springer, 2004, vol. 3175 of *Lecture Notes in Computer Science*, 496–503.

[29] Henn S, Hömke L, Witsch K. *Mathematical Models for Registration and Applications to Medical Imaging*, Springer, 2006, vol. 10 of *Mathematics in Industry*. 3–26.

[30] Henn S, Witsch K. A multigrid-approach for minimizing a nonlinear functional for digital image matching. *Computing* 1999; 64(4):339–348.

[31] Henn S, Witsch K. Iterative multigrid regularization techniques for image matching. *SIAM J. Sci. Comput. (SISC)* 2001; 23(4):1077–1093.

[32] Hömke L. A multigrid method for anisotropic pdes in elastic image registration. *Numerical Linear Algebra with Applications* 2006; 13(2-3):215–229.

[33] Horwitz B, Amunts K, Bhattacharyya R, Patkin D, Jeffries K, Zilles K, Braun A. Activation of Broca' s area during the production of spoken and signed language: A combined cytoarchitectonic mapping and PET analysis. *Neuropsychologia* 2003; 41:1868–1876.

[34] Kimmel R, Bronstein M. *Numerical Geometry for Images*. Springer, 2004.

[35] Liao WH. *Mathematical techniques in object matching in computational anatomy: A new framework based on the level set method*. Ph.D. thesis, University of California, Los Angeles 2006.

[36] Maes F, Collignon A, Vandermeulen D, Marchal G, Suetens P. Multimodality image registration by maximization of mutual information. *IEEE transactions on Medical Imaging* 1997; 16(2):187–198.

[37] Modersitzki J. *Numerical methods for image registration*. Numerical Mathematics and Scientific Computation. Oxford University Press, 2004.

[38] Morosan P, Rademacher J, Schleicher A, Amunts K, Schormann T, Zilles K. Human primary auditory cortex: Cytoarchitectonic subdivisions and mapping into a reference system. *NeuroImage* 2001; 13:684–701.

[39] Osher S. *Geometric level set methods in imaging, vision, and graphics*. Springer, 2003.

[40] Osher S, Fedkiw RP. *Level Set Methods and Dynamic Implicit Surfaces*. Springer, 2002, 1st edn.

[41] Rektorys K. *Variational Methods in Mathematics, Science, and Engineering*. D.Reidel Publishing Company, Dordrecht-Holland/Boston-U.S.A., 1981.

[42] Rohr K. *Landmark-Based Image Analysis*. Kluwer Academic Publisher, Dordrecht Boston London, 2001.

[43] Sethian JA. *Level Set Methods and Fast Marching Methods*. Cambridge University Press, 1999, 2nd edn.

[44] Stüben K, Trottenberg U. Multigrid methods: Fundamental algorithms, model problem analysis and applications. *Lecture Notes in Mathematics, Springer Verlag* 1982; 960.

[45] Thompson P, Toga A. Anatomically driven strategies for high-dimensional brain image registration and pathology. *Brain Warping, Academic Press* 1998; :311–336.

[46] Toga A, Maziotta J. *Brain Mapping: The Methods, 2nd edition*. Academic Press, 2002.

[47] Trottenberg U, Oosterlee C, Schüller A. *Multigrid*. Academic Press, 2001.

[48] Wells W, Viola P. Alignement by maximization of mutual information. *International Journal of Computer Vision* 1997; 24(2):137–154.

[49] Wendland H. Piecewise polynomial, positive definite and compactly supported radial basis functions of minimal degree. *Advances in Computational Mathematics* 1995; 4(4):389–396.

[50] Wienands R, Joppich W. *Practical Fourier Analysis for Multigrid Methods*. CRC Press, 2004.

[51] Zeeuw PMD. Matrix-dependent prolongations and restrictions in a blackbox multigrid solver. *Journal of Computational and Applied Mathematics* 1990; 33(1):1–27.

[52] Zhao HK, Chan TF, Merriman B, Osher S. A variational level set approach to multiphase motion. *Journal of Computational Physics* 1996; 127:179–195.

Statement of originality

Die hier vorgelegte Disseration habe ich eigenständig und ohne unerlaubte Hilfe angefertigt. Die Dissertation wurde von der vorgelegten oder in ähnlicher Form noch bei keiner anderen Institution eingreicht. Ich habe bisher keine erfolglosen Promotionsversuche unternommen.

I do herewith declare that the material contained in this dissertation is an original work performed by me without illegitimate help. The material in this thesis has not been previously submitted for a degree in any University.

Düsseldorf, den 15.8.2006

(Lars Hömke)