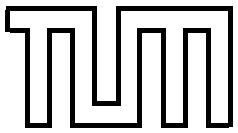


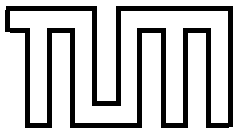
Technische Universität München
Fakultät für Informatik

Diploma Thesis in Computer Science

Non-rigid Registration Using Free-form Deformations

Loren Arthur Schwarz

In Collaboration with Siemens Corporate Research, Inc.
Princeton, New Jersey (USA)



Technische Universität München
Fakultät für Informatik

Diploma Thesis in Computer Science

Non-rigid Registration Using Free-form Deformations

Loren Arthur Schwarz

Director:	Prof. Nassir Navab, Ph.D. (TUM)
Supervisor:	Darko Zikic (TUM) Ali Khamene, Ph.D. (SCR)
Submission:	May 15, 2007

In Collaboration with Siemens Corporate Research, Inc.
Princeton, New Jersey (USA)

Ich versichere, dass ich diese Diplomarbeit selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 10. Mai 2007 Loren Arthur Schwarz

Abstract

A deformable registration approach for medical images of same dimensionality is proposed. The popular free-form deformations (FFD) setting is utilized to characterize deformations based on a grid of control points. B-splines serve the purpose of interpolating the dense deformation field from a given control point configuration. The central idea is to combine the FFD method with well-understood techniques from the context of variational deformable registration problems. In particular, an energy functional is employed that consists of image dissimilarity and regularization terms which are both functions of the free-form deformation control points. An iterative optimization approach is chosen that is inspired by methods used to solve the partial differential equations that arise in the variational registration realm. In a sense, computations that take place on a per-pixel basis in the variational approach are transferred to the coarse grid of control points, leading to a potential efficiency gain. In order to best account for the complex deformations that body tissue is typically exposed to, a multi-resolution strategy is used that increasingly refines the control point grid. The algorithm is implemented in C++ and can be utilized for 2D-2D and 3D-3D registration. Several known techniques to increase computational efficiency are incorporated and either linear or cubic B-splines can be used. An evaluation is provided based on ground-truth experiments with synthetic 2D images and real patient CT scans, demonstrating the effectiveness of the proposed algorithm and its practical applicability to medical problems.

Zusammenfassung

Im Rahmen dieser Diplomarbeit wird ein Verfahren zur deformierbaren Registrierung medizinischer Bilddaten gleicher Dimension vorgestellt. Der weit verbreitete Ansatz der Freiform-Deformation (FFD) wird verwendet, um Verformungen anhand eines Gitters von Kontrollpunkten zu beschreiben. Die Verschiebung für einzelne Pixel wird dabei mittels B-Spline-Funktionen aus der Position der Kontrollpunkte errechnet. Im Mittelpunkt steht die Idee, den FFD-Ansatz mit Methoden zu verbinden, die der variationellen deformierbaren Registrierung entstammen und in der Literatur ausführlich untersucht sind. Insbesondere wird ein Energiefunktional besprochen, das aus einem Bildähnlichkeitsmaß und einem Regularisierer besteht; beide Terme sind hierbei als Funktionen der Kontrollpunkte aufgestellt. Eine iterative Optimierungsstrategie wird eingesetzt, die dem Kontext der variationellen Registrierung entlehnt ist und dort zum Lösen der auftretenden partiellen Differentialgleichungen dient. In gewisser Hinsicht werden die Berechnungen, die im variationellen Ansatz pro Pixel durchgeführt werden, auf die Ebene des Kontrollpunktgitters übertragen, was eine Steigerung der Effizienz verspricht. Um die großen Verformungen bestmöglich zu rekonstruieren, die im Körper auftreten können, wird ein Multiskalenansatz gewählt, bei dem das Kontrollpunktgitter entsprechend verfeinert wird. Der Algorithmus wurde in C++ implementiert und kann zur 2D-2D- sowie 3D-3D-Registrierung eingesetzt werden. Es wurden mehrere bekannte Verfahren zur Effizienzsteigerung integriert und es besteht die Wahl zwischen linearen oder kubischen B-Splines. Die Ergebnisse einer Auswertung unter Einsatz von synthetischen 2D-Bildern sowie realen CT-Aufnahmen werden präsentiert und zeigen, dass der vorgestellte Algorithmus effektiv ist und für praktische Zwecke im medizinischen Bereich eingesetzt werden kann.

Acknowledgements

Many people have assisted me in preparing this thesis in various ways. Without their individual help I would probably still be where I am today (somewhere close to finishing my studies), but several things would simply have been different. First and foremost, without the friendly recommendation and the trust provided by Prof. Nassir Navab I would not have gone to Princeton for the exciting internship at Siemens Corporate Research. Without the original ideas and experience of Ali Khamene I would not have worked on a topic so challenging and fascinating as the present one. Without all the discussions that in parts reached out dangerously close to the borders of my horizons I would not have coped with many obstacles on the way. Without the uncomplicated, friendly and easy-going atmosphere I experienced while working with Ali I would not have had as much fun as I did. Without the other interns I would not have shared countless thoughts on simple things (nasty programming stuff) and less simple things (philosophy of mankind). Not having Fabrice Michel in one cubicle's reach would have meant many more days of headache caused by unsolvable mathematical problems and dozens of recitals less about French history, cuisine and tongue-twisters. And, of course, without the continuous, patient and illuminating support by Darko Zikic lots of small details and big pictures would not have found their intricate way to my awareness.

Contents

Part I.

Introduction and Overview

1. Introduction

Advanced computer technology is increasingly present in medical procedures and novel approaches that aim to improve diagnosis, intervention and medical workflow are evolving at a fast pace. Modern computer systems make complex techniques feasible that were out of reach just a few years ago. Moderate hardware costs additionally foster practical application of computerized methods in clinical environments. Hardly any type of computer aided medical technology does not rely on images whatsoever. Imaging modalities such as computed tomography (CT), magnetic resonance imaging (MRI) or positron emission tomography (PET) are widely used and literally provide physicians with invaluable insight.

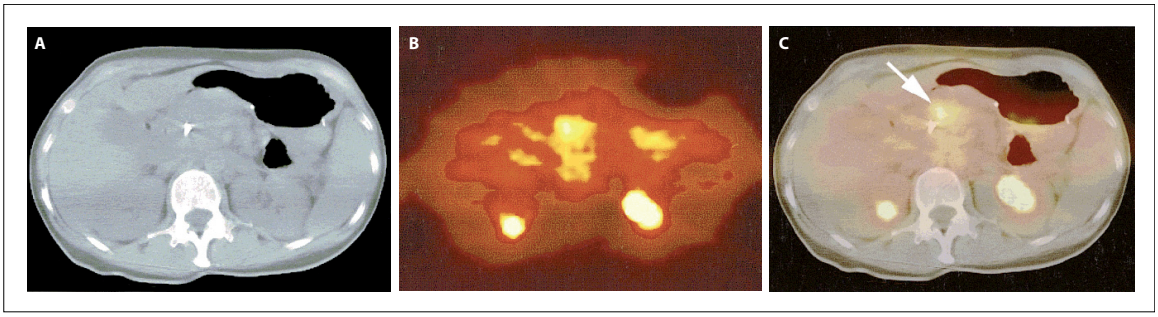
A typical scenario is, however, that the amount of data generated by medical imaging devices exceeds the available time and concentration potential of practitioners. In a way, the introduction of computerized technology therefore requires other new digital aids that assist humans in evaluating acquired data. Image registration is an important tool in this context that strives to automatically combine medical images of various sources in order to maximize the benefit for physicians. Usually the interest is focussed on one specific region or structure, such as a lesion or a tumor, and either its evolution over time or its appearance in different modalities is crucial. A valuable registration algorithm should make an immediate identification of these aspects possible without distracting the medical expert's attention to insignificant details.

This thesis describes the theoretical background and implementation of a registration algorithm. It is inspired by the work of various research groups and tries to combine advantageous approaches into one method. There are also added elements that extend previously existing ideas from a theoretical as well as from a practical point of view. A working system is available and is used to evaluate various aspects of the proposed algorithm based on synthetic and real medical data.

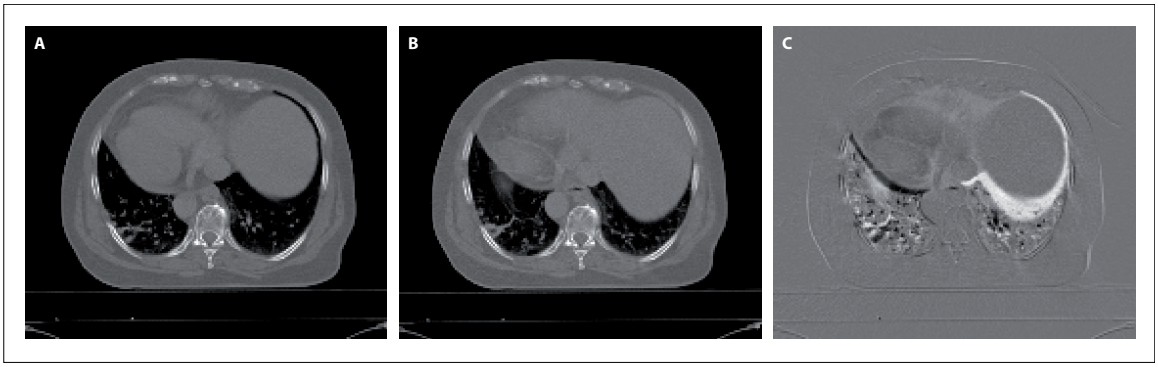
1.1. Motivation

While the human body is rigid to a certain degree based on the skeleton, soft tissue is deformable in a way that does not conform to any rigid approximation [3]. Rotations, scaling and translations, the only means of transformation available for rigid registration, are insufficient to adequately characterize natural soft tissue deformations. Yet at the same time soft tissue is most often the target of medical interest. While rigid registration methods are well-established, modeling non-rigid deformations in a meaningful way is still a challenge. Since practically any kind of movement is allowed in the realm of deformable registration, it is crucial to devise limits preventing deformations that are not plausible from a natural point of view.

Medical image registration and especially deformable registration is a recent discipline of active research. Despite their complexity, registration techniques are increasingly incorporated into devices and procedures that are utilized on a regular basis in clinical environments



(a) Patient with a history of pancreatic cancer, slices from CT (A) and FDG-PET scans (B). The difficulty of anatomically localizing the regions of highest marker uptake in the PET image is apparent. Registration and overlay of PET and CT scans (C) allows the expert to identify the region in doubt as pancreas, against a different suspicion. Images and remarks from [2].



(b) Corresponding slices in two CT scans taken successively at different breathing stages (A, B). The difference image before registration (C) illustrates the non-rigid character of deformations that occur naturally in a body. Deformable registration allows to precisely identify corresponding structures in the images.

Figure 1.1.: Examples of multi-modal and mono-modal medical registration.

where precision and reliability are vital. Attractive fields of application for registration methods can be found throughout the clinical track of events. Apart from diagnostic techniques, registration can be used to improve planning, execution and evaluation of surgical and radiotherapeutical procedures [19].

Registration is for example necessary to combine functional and anatomical images that are typically of complementary nature. For instance, PET imaging is used to illustrate functional properties through metabolism but hardly any anatomical structures are captured in PET images. On the other hand, CT scans can accurately depict anatomical structures, such as bone and tissue. In combination these two modalities can exceed their individual benefit, coupling functional and anatomical information. PET imaging is successfully used for carcinoma identification and treatment planning [2]. In order to precisely localize potential tumor tissue that is visible in a PET scan, it can be registered with a CT scan of the same body region. To account for movements induced e.g. by breathing between the PET and CT scans, a deformable registration approach is required. As Figure 1.1(a) demonstrates, a registered PET-CT image allows to both identify potential carcinoma tissue and to localize it anatomically.

Registration between different images of same modality is also important, for instance to verify treatment success based on pre- and post-interventional images of a patient. Moreover, growth monitoring on tumors can be facilitated by means of deformable registration applied to MR scans taken over longer periods of time [19]. Lung movement can be analyzed in order to be compensated for during radiotherapy. Another field of application for mono-modal registration can save time on manual segmentations performed by experts. A structure of interest, such as a lesion or a tumor, can be segmented once for a series of CT or MR scans that are mutually deformed by breating or patient motion. Deformable registration is then used to recover movements between the scans and to transform the single segmentation accordingly. The resulting artficial segmentations can then be used for the remaining scans in the series without the need to manually segment each scan.

1.2. Thesis Outline

The thesis is divided into five main building blocks that approach the topic from different angles. Each of these parts contains thematically related sections and all parts build upon each other. The outline of the thesis reads as follows:

I **Introduction and Overview**. After a few introductory words this part gives a general overview of the field of topics related to deformable medical image registration. The aim is to provide a grasp of the problem setting and to point out typical characteristics and challenges.

II **Background and Previous Work**. This part introduces background concepts that are the foundation for the deformable registration algorithm. Knowledge that is required to comprehend all subsequent parts is to be provided in a concise way. Moreover, related work done by other research groups is outlined so that this thesis can be put into context.

III **Deformable Registration Based on B-splines**. First a rather formal view of the proposed registration algorithm is given in this part. The description concentrates on high-level algorithmic elements, such as the objective, parameterization and the optimization strategy. Implementation details and issues such as efficiency are addressed thereafter.

IV **Evaluation**. Experiments and results are presented that illustrate important properties of the registration algorithm, such as the influence of specific parameters. Measurements are performed both in two and in three dimensions using synthetic images and medical patient data to demonstrate the applicability of the algorithm to typical real registration problems.

V **Summary and Conclusion**. The most important thoughts discussed in the thesis are summarized and closing remarks are given. Possible directions for future work on related topics are provided.

2. Problem Setting

The following sections take a first introductory look at crucial concepts related to image registration that will be recurrent throughout the thesis. After highlighting the nature of the data that medical imaging algorithms typically deal with, a short overview of different registration approaches will be given. The notion of free-form deformations is clarified and finally an overview of the registration approach proposed in this thesis is given.

2.1. Medical Images

In medical applications, input data typically originates from imaging modalities such as X-ray machines, CT or MRI scanners. Recently designed imaging systems natively generate digital images that can be directly used in computers for further processing steps [6]. Two-dimensional data sets as they are acquired for instance by X-ray or ultrasound machines are referred to as *images*, while three-dimensional data sets, such as CT scans, are called *volumes*. Individual elements of the two types of data sets are accordingly named *pixels* and *voxels*, respectively. For brevity reasons it is however customary to use the term *image* for two- and three-dimensional data sets when there is no risk of confusion.

Since most medical imaging devices do not measure visible light emission or reflection but artificially generate images based on various physical phenomena, medical images are generally intensity images. Scalar values within a range that is specific to different imaging modalities are assigned to locations in an image or volume. Such an intensity image can be formally viewed as a function $I : \Omega \to \Gamma,\ \mathbf{x} \mapsto I(\mathbf{x})$, where $\Omega \subset \mathbb{N}^d$ and d is the dimensionality (e.g. 2 or 3) [12]. The domain Ω is often defined to be a subset of $\mathbb{R}^d$ in the literature if images are treated as continuous quantities. In this thesis, however, images are from the beginning viewed as discrete objects along with an integer indexing scheme.

The codomain Γ depends on the modality and can be for example $[-1000, 3000] \subset \mathbb{Z}$ for CT data, compare [6]. In order to be able to deal with different modality images within the same algorithm it is often convenient to perform normalization by scaling Γ to $[0, 1] \subset \mathbb{R}$. Typical resolutions of medical imaging systems such as CT or MR are between 256 and 1024 samples per dimension, while usually the number of slices in a volume is less than the planar resolution per slice.

2.2. Optimization Problems

Registration is a typical example of an optimization problem. An optimization problem in general is any kind of problem where a particular solution from a set of candidate solutions is sought that is optimal in some sense. Typically real valued numbers are assigned to candidate solutions that measure their quality [12]. Special target functions have to be designed that perform this evaluation in a sense that fits the characteristics of a particular

problem. Since in most cases it is not feasible to check all possible solutions by enumerating them, a strategy has to be provided that states how to select candidate solutions. The set of all possible parameter changes that can be made in order to transform one candidate solution to a new one is called *search space*. The distinction between maximization and minimization problems basically states whether the optimal solution is supposed to have the highest or the lowest quality value, compared to all other candidate solutions.

Image registration is usually posed as a minimization problem. Candidate solutions are transformations that can be applied to one image in order to make it more similar to the other image. The target function generally contains a measure that quantifies this similarity, compare [19]. Many alternatives exist for optimization strategies, mainly depending on whether the target function is linear or not. Registration and in particular deformable registration problems typically involve nonlinear target functions, so that suitable solution strategies include gradient descent, complex algorithms such as the Levenberg-Marquardt method or special fix-point iteration techniques derived from solution methods for partial differential equations, as used for this work.

2.3. Registration Approaches

The general goal of finding a suitable alignment between two images can be achieved in numerous ways that depend on several characteristics of a particular registration problem. There are several terminologies – the image that is not changed during registration and to which the other image has to be registered is often called *reference* or *fixed image*. The second image that is made increasingly similar to the reference image is called *template* or *moving image*. In order to put in context the approach that is studied in depth for this work, it is helpful to briefly consider the principal ways of classification for registration algorithms along with fields of application. The main difference between various methods lies in the origin and the dimensionality of images to be registered, compare e.g. [19, 12]. Other important classification criteria include the way images are compared and different transformation models.

2.3.1. Mono-modal vs. Multi-modal

Each medical imaging method has physical characteristics that make it especially suitable for a certain kind of application. Every imaging modality also has some weaknesses that make image interpretation based on one single image difficult. Moreover, certain imaging technologies, such as computer tomography, have negative side effects on patients so that there is a limitation on utilization frequency. It can therefore be of great interest to acquire images of a certain body region using different imaging techniques and then to combine them for diagnosis. *Mono-modal* registration algorithms concentrate on aligning images originating from one and the same imaging modality. A practical example could be to register of a series of CT scans taken at different breathing stages of a patient. *Multi-modal* registration methods are used to register images acquired using different modalities. Such images usually have a totally different apperance although the same part of the body is shown. An example application is PET-CT registration that has been outlined before [3].

2.3.2. Dimensionality

2D-2D registration can be useful, for instance, when two X-ray images of the same patient taken at different times have to be compared. The multi-modal case of registering CT and MRI data is an example application for *3D-3D* registration. CT images offer a high contrast between bone structures and soft tissue, but different kinds of soft tissue are often hard to distinguish. This disadvantage is overcome by MRI images that allow visual separation of soft tissue. *2D-3D* registration can for instance take place between an X-ray image and an MRI volume . Registration is performed by generating 2D projections of the 3D volume which are then compared to the 2D image the volume is being registered to. Typical medical applications include specific intraoperative navigation techniques. A C-Arm can be used to acquire 2D fluoroscopy images during a surgery. These images are registered to pre-operatively acquired MRI scans so that the mapping between the visualized 3D volume and the current surgery process is facilitated for the surgeon [19].

2.3.3. Feature-based vs. Intensity-based

There are registration algorithms that utilize geometrical information in order to align two images. These so-called *feature-based* methods rely on point or shape correspondences between two images or volumes. Features can either be automatically derived from image characterstics, such as corners or contours of anatomical structures, or from markers with known positions [3]. Once corresponding points have been found, their locations in the two images can be used to reconstruct a spatial transformation. This transformation is then applied to one of the two images so that differences e.g. in scaling, rotation and translation between the two images are eliminated. *Intensity-based* methods treat images or volumes as whole entities. Instead of specific features, only pixel intensity values are considered in order to find the transformation of interest. Suitable similarity measures are crucial for a meaningful intensity-based comparison of two images or volumes.

2.3.4. Rigid vs. Deformable

Another classification criterion for registration algorithms is the type of transformation they use to map one image to the other image. For the rigid and affine cases the transformation is specified as a matrix that maps any point in one image to its appropriate position in the second image [19]. *Rigid* transformations can account for pure rotation and translation. *Affine* transformations extend the rigid approach to include stretching and skewing which increases registration flexibility. However, in many cases soft tissue is deformed in a more complicated fashion that can be represented by neither rigid nor affine transformations.

Deformable or *non-rigid* registration can account for much more general transformations compared to rigid or affine registration, at the cost of increased complexity. Potential deformations are allowed to encompass arbitrary movements of individual image pixels that are stored in so-called displacement fields. Finding a displacement field that optimally links two images together is, however, an ill-posed problem, since only certain movements are likely to occur in reality. A deformation is in general considered plausible if the pixel movement is smooth as to simulate natural elastic deformation. Such smoothness properties can be enforced in deformable registration algorithms by different means. For instance, a transformation model such as that of free-form deformations (FFD) can be chosen that

inherently generates smooth deformations [7]. In addition, a regularization strategy can be used that penalizes unlikely deformations and favors smooth candidates.

2.4. Free-form Deformations

The goal of free-form deformations is to provide a convenient means of modeling arbitrary deformations applied to objects. Although the foundations of FFD methods can be traced back to the area of computer aided design where geometric objects are manipulated [26], an application to images is also possible [17]. The general idea is to deform an image by manipulating a regular grid of control points that are distributed across the image at an arbitrary mesh resolution. Control points can be moved and the position of individual pixels between the control points is computed from the positions of surrounding control points. Techniques based on free-form deformations are attractive for several reasons. Apart from the smoothness properties that can be enforced using suitable basis functions, the control points can be placed at variable distances, giving a flexible way of controlling deformation precision. In addition, the concept of manipulating control points in order to deform an image can have an efficiency advantage over methods where deformations are computed on a per-pixel basis [25].

2.5. Method Overview

The deformable registration algorithm presented in this thesis is targeted at 2D-2D and 3D-3D registration problems. Although the algorithm in general can be applied to mono-modal as well as multi-modal registration, its current implementation makes the assumption that both images are acquired using the same modality. An intensity-based, non-rigid or deformable approach is chosen. Free-form deformations and B-spline basis functions are used to model non-rigid deformations.

An optimization strategy is implemented that is inspired by the general solution framework for variational deformable registration algorithms which is theoretically sound and well-understood. In the variational registration setting optimization takes place in a very high-dimensional space since deformations are modeled on a per-pixel basis. The appealing intuition that is exploited in this work is to elevate the variational optimization approach to the coarse grid of control points in order to increase computational efficiency. A regularization method that is often used in variational registration techniques is adapted to be applicable in the setting of free-form deformations. In particular, a link is created between regularization on dense deformation fields and a control point regularizer that provides comparable behavior at a significantly decreased computational complexity.

In order to cope with large data sets and to improve convergence properties of the algorithm, a multi-resolution strategy is used. A Gaussian pyramid is generated that contains resampled versions of the images at decreasing resolutions. Starting with the pair of images at the lowest resolution, registration is performed using a coarse grid of control points. The registration results are transferred from one resolution level to the next higher level and registration is run again, up to full resolution. This approach ensures that large deformations can be recovered early at a low resolution and more detailed deformations are accounted for at increasingly fine resolutions.

Part II.

Background and Previous Work

3. Background

Before more details on the proposed deformable registration algorithm are given, the most relevant background knowledge will be discussed in this part of the thesis. Concepts that are crucial to registration algorithms will be outlined, such as similarity measures, image warping approaches and displacement fields. The notion of deformation regularization is also introduced, a concept that is specifically used with non-rigid registration methods. Since the most appealing properties of FFD techniques are accounted for by the underlying B-splines, it is worthwile to introduce some fundamentals of spline theory. Finally an overview of related previous work will be given.

3.1. Similarity Measures

A similarity measure is a function that takes two input images as parameters and computes a numerical value that quantifies the extent to which the two images are similar. An ideal similarity function increases as the alignment of two images is improved, and has a peak value if the two images are optimally registered. Only if both images are acquired using the same imaging modality, relatively intuitive approaches are feasible. Such straightforward techniques rely on the fact that similar structures share similar intensity values in the two images [12]. A simple way to quantify image similarity is to consider the intensity difference for each pixel position in the two images. This idea leads to the similarity measure called *sum of squared differences* (SSD) that can be written as

$$\mathrm{SSD}(I_f, I_m) = \frac{1}{N} \sum_{\mathbf{x} \in \Omega} \left(I_f(\mathbf{x}) - I_m(T(\mathbf{x})) \right)^2 , \tag{3.1.1}$$

where N is the total number of pixels. I_f and I_m denote the fixed and moving images, and $T(\mathbf{x})$ is a transformation function that maps a voxel $\mathbf{x}$ to its new position. A slightly modified version of the SSD measure is also widely used and eliminates its quadratic behavior. The *sum of absolute differences* (SAD) is defined as

$$\mathrm{SAD}(I_f, I_m) = \frac{1}{N} \sum_{\mathbf{x} \in \Omega} \left| I_f(\mathbf{x}) - I_m(T(\mathbf{x})) \right| . \tag{3.1.2}$$

In the multi-modal case the assumption that similar anatomical structures have similar intensities is generally not valid [12]. More sophisticated similarity measures have to be introduced, one of which is *mutual information*. This measure is a representative example of the category of statistical similarity measures and has been successfully applied in many medical imaging methods [31]. The idea behind mutual information is to quantify how much information is shared between two images while not relying on intensities. The mutual

information (MI) of two images I_m and I_f is

$$\mathrm{MI}(I_f, I_m) = H(I_f) + H(I_m) - H(I_f, I_m), \tag{3.1.3}$$

where $H(I_f)$ denotes the entropy of the fixed image and $H(I_f, I_m)$ is the joint entropy of the two images. Describing the theory of the entropy measure is beyond the scope of this work, but informally speaking, the entropy of an image refers to the amount of information that it contains. Joint entropy is, as the name implies, the amount of information contained in two images together, compare e.g. [12]. The crucial property is that $H(I_f, I_m) \leq H(I_f)+H(I_m)$; if the two images are completely unrelated, their joint entropy equals the sum of individual entropies. Otherwise it decreases as the two images approach identity. Given this brief explanation, the mutual information measure can be summarized:

- $\mathrm{MI}(I_f, I_m) = 0$, if I_f and I_m are totally unrelated $[H(I_f, I_m) = H(I_f) + H(I_m)]$,
- $\mathrm{MI}(I_f, I_m) = H(I_f)$, if $I_f = I_m$ $[H(I_f, I_m) = H(I_f)]$,
- $\mathrm{MI}(I_f, I_m)$ is between the extreme values if I_f and I_m share some information.

As the registration algorithm described in this thesis is targeted at intra-modality registration, statistical similarity measures are not utlized. If an extension to multi-modal registration is required, there are no theoretical obstacles that prohibit to implement the mutual information measure in the algorithm. The current implementation, however, is based on the SSD and SAD measures.

3.2. Image Warping

No matter which type of registration algorithm is used, there is always the need to transform one image in a certain way in order to align it to the other image. Rigid and deformable registration methods perform different types of transformations, but in both paradigms there is a step that actually applies calculated transformations to an image. This procedure is called *image warping* and deserves a little attention.

Once a certain transformation has been computed (e.g. a global rotation or deformation), information is required on how to move each individual pixel in the image that is being transformed. This type of information is typically stored in a *displacement field* that relates the positions of pixels between the reference and template images [21]. A displacement field is formally a function $u : \Omega \rightarrow \mathbb{R}^d$ on the image domain Ω, where d is the dimensionality. An example of a displacement field can be seen in Figure 3.1 and a displacement field is used in the transformation function in Eq. (3.1.1). It is of the general form

$$T : \Omega \rightarrow \Omega; \quad T(\mathbf{x}) = \mathbf{x} + u(\mathbf{x}) \tag{3.2.1}$$

and transforms pixel coordinates $\mathbf{x}$ in the fixed image I_f to coordinates in the moving image I_m by means of an identity mapping and the corresponding value of the displacement field. Since any deformation is sufficiently characterized by a displacement field, it is the focal point for any registration algorithm. How the displacement field u is computed depends on the properties of particular applications. In this case, a B-spline based transformation function is employed for this purpose that will be addressed in subsequent sections.

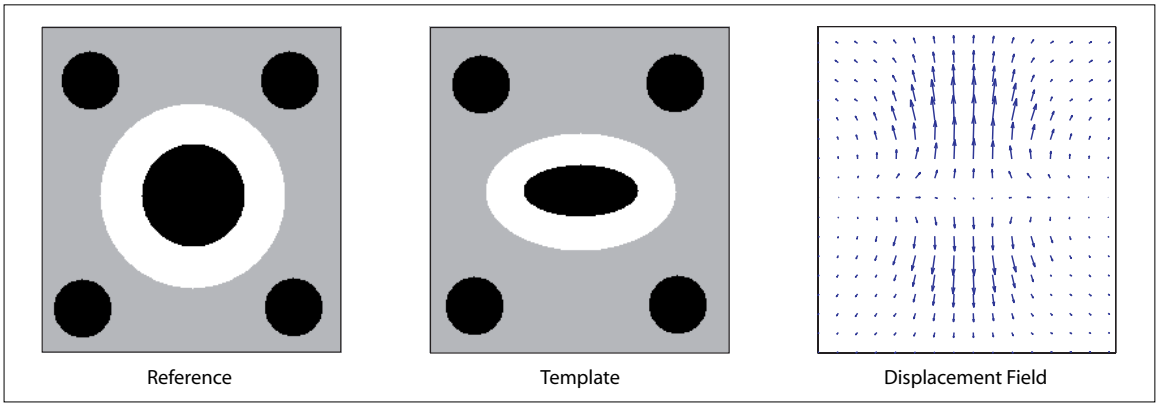


Figure 3.1.: The concept of displacement fields. A displacement field gives for every pixel position in the template image the direction and distance how it has to move in order to match the reference image. The displacement field is subsampled.

Given the definition of a displacement field, there are two principal ways of how image warping can be accomplished. One way is that for each position $\mathbf{x}$ of the template image, the corresponding intensity value is stored in the new image at the location $\mathbf{x}' = T(\mathbf{x})$:

$$I'_m(T(\mathbf{x})) \leftarrow I_m(\mathbf{x}),\ \forall \mathbf{x} \in \Omega. \tag{3.2.2}$$

This process is referred to as *forward* warping, since pixels are in a sense moved 'forward' from the coordinate frame of the old image to the new image. The problem with this intuitive approach is that the transformation function is generally neither injective nor surjective – due to the discrete nature of pixel images, non-integer values of the transformation function have to be rounded. As a result, not every pixel in the new image will be necessarily assigned a value and some pixels can be assigned several times (Figure 3.2).

The other option that eliminiates this problem is called *backward* warping. The main difference is that now for every pixel of the new image a coordinate in the original image is computed, where its intensity value originates from. Obviously this involves the inverse T^{-1} of the transformation function:

$$I'_m(\mathbf{x}) \leftarrow I_m(T^{-1}(\mathbf{x})),\ \forall \mathbf{x} \in \Omega. \tag{3.2.3}$$

In analogy to forward warping, it is possible that $T^{-1}(\mathbf{x})$ yields a non-integer value. However, in the case of backward warping an interpolation scheme on the original image can be used to obtain intensity values at coordinates between pixels. Bilinear or trilinear interpolation (for 2D and 3D) are generally reasonable choices. Unfortunately the inverse of the transformation function is often not trivial to obtain. However, in many cases an approximation such as the following can be used with acceptable results:

$$T^{-1}(\mathbf{x}) \approx \mathbf{x} - u(\mathbf{x}). \tag{3.2.4}$$

As a matter of definition, the behavior of the transformation function can also be treated as an inverse, so that an actual inversion would only be required for forward warping.

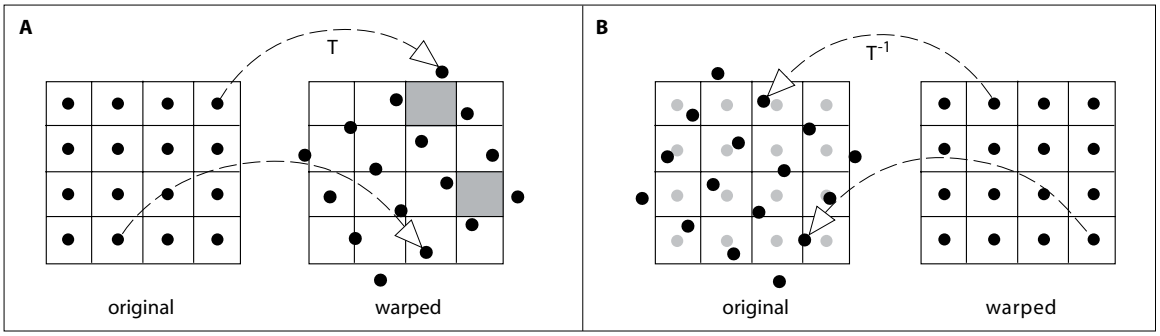


Figure 3.2.: Forward and backward image warping. In the case of foward warping (A), holes can occur in the warped image, marked in gray. Backward warping (B) eliminates this problem since intensities at locations that do not coincide with pixel coordinates can be obtained from the original image using an interpolation scheme.

3.3. Deformation Regularization

It has already been pointed out that not all types of deformations are physically plausible. Registration algorithms therefore are often regulated using a special technique that evaluates a given candidate deformation and penalizes it if it is considered "unregular". Although a physical model for regularity would give the most valid results, it is unfeasible to build such a model [19]. Practical regularization approaches usually rely on rather simple mathematical properties of deformations that are suitable from a conceptual point of view.

Most regularizers exploit the smoothness of displacement fields. A displacement field is considered smooth if it has no harsh jumps or if, in other words, direction and magnitude of displacements in a neighborhood change gradually. The idea of measuring a gradual change directly implies to use derivatives of the displacement fields for regularization. Two most widely used methods of this kind are the *diffusion* and *curvature* regularizers.

3.3.1. Diffusion

Diffusion regularization is physically motivated by the heat diffusion equation that describes how heat is distributed in a given medium over time [15, 21]. The analogy to smooth deformations is that local displacements are expected to spread over a certain region in a similar manner as heat from a static source is distributed over a cooler medium. The diffusion regularizer makes use of first order derivatives of a displacement field and is defined as

$$R_D(u) = \sum_{\mathbf{x}\in\Omega} \|\nabla u_x(\mathbf{x})\|^2 + \|\nabla u_y(\mathbf{x})\|^2 + \|\nabla u_z(\mathbf{x})\|^2, \tag{3.3.1}$$

where u_x is the x-component of the displacement field u, $\nabla = (\partial/\partial x, \partial/\partial y, \partial/\partial z)^\top$ is the gradient operator and $\|\cdot\|$ is the Euclidean vector norm. Such a penalty function is typically used in a global cost functional which is minimized during optimization. This way the "best" displacement field u is the one for which $R_D(u)$ is minimal.

3.3.2. Curvature

A similar regularization approach is the curvature regularizer that is based on second order derivatives [21]. The oddity of mathematical notation – which might fool the unsuspecting – allows to simply flip the gradient operator symbol ∇ to obtain the Laplace operator Δ. It essentially adds the unmixed second partial derivatives, $\Delta = \partial^2/\partial x^2 + \partial^2/\partial y^2 + \partial^2/\partial z^2$. The curvature regularization function can then be stated as

$$R_C(u) = \sum_{\mathbf{x}\in\Omega} \left(\Delta u_x(\mathbf{x})\right)^2 + \left(\Delta u_y(\mathbf{x})\right)^2 + \left(\Delta u_z(\mathbf{x})\right)^2. \tag{3.3.2}$$

The defining characteristic of the curvature regularizer is that it is invariant under affine transformations [7]. In other words, translations, rotations and scaling that can be necessary to register one image to the other are not penalized. As pointed out in [4], this aspect can highly reduce the sensitivity of the regularizer to the initial position of two images to be registered, reducing the impact of a missing or inaccurate rigid pre-registration.

3.4. Splines and B-Splines

The notion of *splines* originates from the field of industrial design at times long before the use of computers [8]. The term referred to simple mechanical tools used by designers to produce smooth curves in technical drawings. Especially in the shipbuilding industry flexible thin strips of wood or metal were used for this purpose. Heavy lead weights were placed at specific positions in a drawing and the elastic strips, the splines, were clamped in between the weights. Because of their material properties and the constraining force exerted by the weights the splines would take on smooth shapes. The designer was then able to connect the given points by tracing out the splines. Today the term spline is mainly associated with a certain type of mathematical function that is widely applied in computer science and graphics. Just as their mechanical counterparts, splines nowadays allow to create smooth shapes and surfaces. The principal idea of specifying and adjusting certain characteristic points, the control points, remains a central aspect of splines and makes them suitable to be applied in conjunction with free-form deformation techniques.

3.4.1. Parametric Curves

The principles of B-splines lie in the notion of parametric curves. There are generally several possibilities to mathematically define a curved shape. For instance, explicit or implicit functions can be used to model curves in an arbitrary number of dimensions. Parametric representations allow to model more general types of shapes and are therefore widely used in many fields [8]. A parametric representation is a mapping g from a parameter domain P, such as $[0, 1] \subset \mathbb{R}$, to a vector-valued codomain S, e.g. $\mathbb{R}^2$. The graph of a parametric representation is called a parametric curve and in general there can be several parametric representations that result in the same parametric curve. A simple example of a parametric curve is the unit circle

$$g : [0, 2\pi] \to \mathbb{R}^2, \quad g(t) = (\cos t, \sin t). \tag{3.4.1}$$

As the parameter t varies from 0 through 2π, the shape of the circle is traced out counter-clockwise. Getting back to the informal idea of splines, the desired setting is that the shape be determined by a set of control points. Moreover, polynomials are more favorable for parametric representations than trigonometric functions. The simplest example of a polynomial curve (of degree 1) is a straight line. Given two control points $\mathbf{c}_1, \mathbf{c}_2 \in \mathbb{R}^2$, a connecting line could be represented in a parametric form as

$$p : [0, 1] \to \mathbb{R}^2, \quad p(t) = (1 - t)\mathbf{c}_1 + t\mathbf{c}_2. \tag{3.4.2}$$

A weighted average of the two points is computed and as the parameter t increases from 0 to 1, the influence of the two points is gradually shifted from $\mathbf{c}_1$ to $\mathbf{c}_2$. This type of weighting, where the weighting factors are non-negative and add up to 1, is called a *convex combination* and can easily be generalized to more than two points, compare e.g. [18].

When performing calculations with a given set of points, it is desirable to use convex combinations, as this ensures that the result will always lie within the convex hull of the set of points[1]. The main advantage is increased numerical stability, since the output is guaranteed to be in the numerical range of the input points [18]. As will be shown in subsequent sections, convex combinations can be used to obtain parametric representations allowing to model curves and surfaces to an arbitrary degree of flexibility.

3.4.2. Bézier Curves

Extending the simple example of equation 3.4.2 to the case of three control points, $\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3 \in \mathbb{R}^2$, two convex combinations can be formed to create the connecting line segments:

$$p_{1,1}(t) = (1 - t)\mathbf{c}_1 + t\mathbf{c}_2, \quad p_{2,1}(t) = (1 - t)\mathbf{c}_2 + t\mathbf{c}_3. \tag{3.4.3}$$

The notation $p_{1,1}(t)$ is comprised of an index for one of the line segments and another index for the degree of the underlying polynomial. These two line segments can now be combined, in turn, in a convex combination,

$$p_{1,2}(t) = (1 - t)p_{1,1}(t) + tp_{2,1}(t) = (1 - t)^2\mathbf{c}_1 + 2t(1 - t)\mathbf{c}_2 + t^2\mathbf{c}_3. \tag{3.4.4}$$

The resulting parametric representation is clearly a polynomial of degree 2. This type of curve is called a quadratic Bézier curve [8, 18]. The scheme of repeated convex combinations can obviously be continued to an arbitrary depth. For instance, a cubic Bézier curve is constructed from four control points – three line segments are combined to two quadratic curve segments, which are finally combined to one cubic Bézier curve. An illustration of the construction for Bézier curves is shown in Figure 3.3.

A Bézier curve does not necessarily pass through all its control points; in other words, Bézier curves are not interpolating but approximating curves [8]. Moreover, the degree of a Bézier curve depends on the number of control points. As has been shown before, a quadratic Bézier curve is constructed using three control points and a cubic curve requires four thereof. A complex shape with many control points would therefore result in a Bézier curve of very high degree. One remedy for this issue that also leads the way to spline curves is to use *piecewise* Bézier curves. The idea is to model a complex shape by stitching together

[1] The convex hull of a set of points is, in fact, the set of all possible convex combinations of the points.

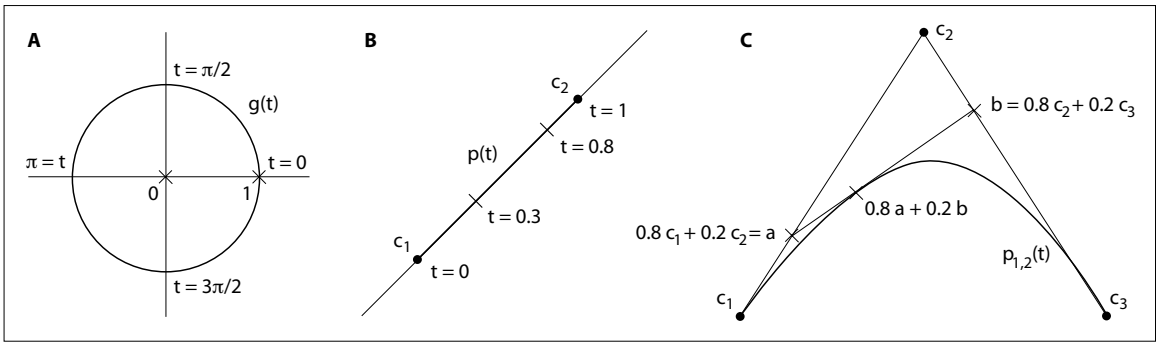


Figure 3.3.: Examples of parametric curves. A unit circle (A), a straight line connecting two control points (B) and a quadratic Bézier curve (C).

short curves of a fixed degree. For instance, if cubic Bézier curves are used for this purpose, every sequence of four consecutive control points is taken to generate a cubic Bézier curve. A still remaining problem is that there is in general no continuity at the joints between adjacent Bézier curves [18].

3.4.3. Spline Curves

A few slight modifications to the construction of Bézier curves finally lead to the desired concept of spline curves. While in Bézier curves convex combinations are always performed with the weights $(1-t)$ and t, a more general type of weighting is utilized for splines. The parameter t itself is not restricted to the range $[0,1]$ any more and the domains of adjacent curves are defined to overlap. In all, these modifications result in piecewise curves that fit together smoothly at the joints.

The range of the parameter t is allowed to be $[t_a, t_b]$ for any two real numbers $t_a < t_b$. Specific parameter values $t_a \leq t_i \leq t_b$, called the *knots*, are used to divide the range into subintervals. For a set of n control points $\{\mathbf{c}_1, \ldots, \mathbf{c}_n\}$ a total of $n+d-1$ knots are required, where d is the desired degree of the spline curve [18]. The i-th piece $p_{i,d}(t)$ of a spline curve $p_d(t)$ can then be defined in a recursive way with the base case

$$p_{i,0}(t) = \mathbf{c}_i \tag{3.4.5}$$

and recurrence

$$p_{i,d}(t) = \frac{t_{i+r} - t}{t_{i+r} - t_i} p_{i-1,d-1}(t) + \frac{t - t_i}{t_{i+r} - t_i} p_{i,d-1}(t), \tag{3.4.6}$$

where $r \in \{1, \ldots, d\}$ is the recursion depth. Notice that this parametric representation is still a convex combination since the weighting factors are non-negative and sum up to one. As Eq. (3.4.6) defines the i-th component of a piecewise spline curve, the total curve of degree d can be written as

$$p_d(t) = \begin{cases} p_{d+1,d}(t) & t \in [t_{d+1}, t_{d+2}], \\ p_{d+2,d}(t) & t \in [t_{d+2}, t_{d+3}], \\ \vdots & \vdots \\ p_{n,d}(t) & t \in [t_n, t_{n+1}]. \end{cases} \tag{3.4.7}$$

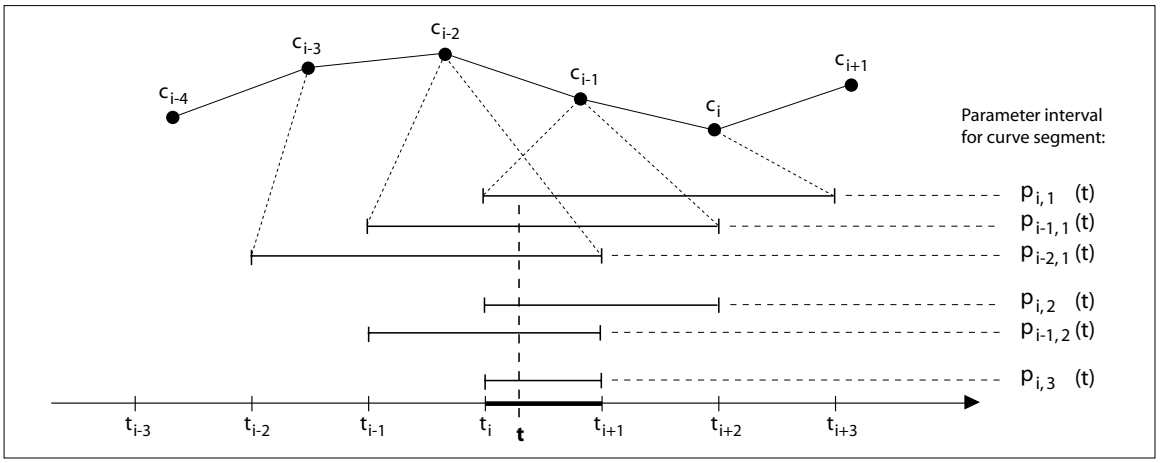


Figure 3.4.: Schematic view of the weighting scheme for spline curves. As opposed to Bézier curves where all subsegments are weighted in the same proportion, the weights are changed for spline curves. The weights for the two quadratic curve segments $p_{i-1,2}(t)$ and $p_{i,2}(t)$ are given by the relative position of t in the interval $[t_i, t_{i+1}]$. On the lowest level the weights for the control points are obtained from the relative position of t inside intervals of length 3.

A brief example illustrates the mechanism. The i-th piece of a spline curve of degree 3 is defined on the interval $[t_i, t_{i+1}]$ and is expressed according to Eq. (3.4.6) as

$$p_{i,3}(t) = \frac{t_{i+1} - t}{t_{i+1} - t_i} p_{i-1,2}(t) + \frac{t - t_i}{t_{i+1} - t_i} p_{i,2}(t), \tag{3.4.8}$$

where $r = 1$ on the first level of recursion. The cubic spline curve is a convex combination of two quadratic spline curves and the weights are computed by scaling the interval $[t_i, t_{i+1}]$ to $[0, 1]$. On the second level of recursion ($r = 2$), the two quadratic spline curves are then obtained according to

$$p_{i-1,2}(t) = \frac{t_{i+1} - t}{t_{i+1} - t_{i-1}} p_{i-2,1}(t) + \frac{t - t_{i-1}}{t_{i+1} - t_{i-1}} p_{i-1,1}(t) \tag{3.4.9}$$

$$p_{i,2}(t) = \frac{t_{i+2} - t}{t_{i+2} - t_i} p_{i-1,1}(t) + \frac{t - t_i}{t_{i+2} - t_i} p_{i,1}(t) \tag{3.4.10}$$

The two quadratic spline curves are each convex combinations of two linear spline curves. Although the parameter t still varies in the interval $[t_i, t_{i+1}]$, the weighting factors are now computed by scaling the intervals $[t_{i-1}, t_{i+1}]$ and $[t_i, t_{i+2}]$ to the range $[0, 1]$. The deepest level of recursion with $r = 3$ gives the following equations for the three different line segments:

$$p_{i-2,1}(t) = \frac{t_{i+1} - t}{t_{i+1} - t_{i-2}} \mathbf{c}_{i-3} + \frac{t - t_{i-2}}{t_{i+1} - t_{i-2}} \mathbf{c}_{i-2} \tag{3.4.11}$$

$$p_{i-1,1}(t) = \frac{t_{i+2} - t}{t_{i+2} - t_{i-1}} \mathbf{c}_{i-2} + \frac{t - t_{i-1}}{t_{i+2} - t_{i-1}} \mathbf{c}_{i-1} \tag{3.4.12}$$

$$p_{i,1}(t) = \frac{t_{i+3} - t}{t_{i+3} - t_i}\mathbf{c}_{i-1} + \frac{t - t_i}{t_{i+3} - t_i}\mathbf{c}_i, \tag{3.4.13}$$

where the line segment $p_{i-1,1}(t)$ is used by both quadratic spline curves. This weighting scheme for the curve segments is illustrated in Figure 3.4. The characteristic of overlap in the construction accounts for the favorable smoothness properties of spline curves. A cubic spline curve has continuous first and second derivatives (C^2 continuity), even at the joints between adjacent curve segments. In general, a spline curve of degree d has $d-1$ continuous derivatives [18], a significant advantage over a Bézier curve of same degree. Spline curves are also approximating curves that do not necessarily pass through all control points. They do, however, lie within the convex hull of all control points.

3.4.4. B-Splines

The great practical shortcoming of Eq. (3.4.7) is that the control points do not appear explicitly. Easy manipulation of a curve, however, requires the possibility to adjust its shape only through the control points. An algebraic transformation that can be found e.g. in [18] yields such a formulation with explicit control points $\mathbf{c}_i$. The first step is to rewrite Eq. (3.4.7) in a more concise way as

$$p_d(t) = \sum_{i=d+1}^{n} B_{i,0}(t)\ p_{i,d}(t), \qquad \text{with} \quad B_{i,0}(t) = \begin{cases} 1, & t_i \le t < t_{i+1} \\ 0, & \text{otherwise.} \end{cases} \tag{3.4.14}$$

This formulation introduces the appealing fact of being a linear combination of certain basis functions weighted by specific terms. Further simplifications that are not included in this discussion then yield the following notation for a spline curve of degree d:

$$p_d(t) = \sum_{i=1}^{n} B_{i,d}(t)\ \mathbf{c}_i, \tag{3.4.15}$$

where $B_{i,d}(t)$ are basis spline functions, called *B-splines*. They are defined in a recursive way based on $B_{i,0}(t)$ in Eq. (3.4.14):

$$B_{i,d}(t) = \frac{t - t_i}{t_{i+d} - t_i}B_{i,d-1}(t) + \frac{t_{i+d+1} - t}{t_{i+d+1} - t_{i+1}}B_{i+1,d-1}(t). \tag{3.4.16}$$

The B-splines are spline functions with real valued coefficients instead of control points. Similar to spline curves, a B-spline of degree 0 is a piecewise constant function, a B-spline of degree 1 a piecewise linear and a B-spline of degree 2 a piecewise quadratic polynomial. For instance, the first linear and quadratic B-splines can be derived from the recurrence as

$$B_{0,1}(t) = \begin{cases} t, & 0 \le t < 1 \\ 2 - t, & 1 \le t < 2 \\ 0, & \text{otherwise,} \end{cases} \qquad B_{0,2}(t) = \frac{1}{2}\begin{cases} t^2, & 0 \le t < 1 \\ -2t^2 + 6t - 3, & 1 \le t < 2 \\ (3-t)^2, & 2 \le t < 3 \\ 0, & \text{otherwise.} \end{cases} \tag{3.4.17}$$

Other B-splines and especially higher degree B-splines are stated in a similar way. A few B-splines of different degrees are shown in Figure 3.5.

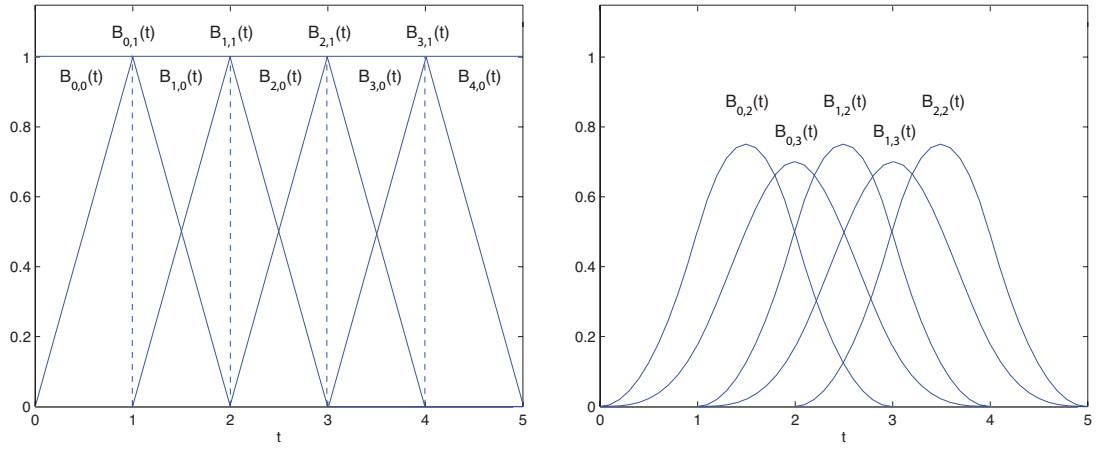


Figure 3.5.: The first few B-splines on a uniform knot sequence. Constant and linear (left), quadratic and cubic B-splines (right). All B-splines are translates of the first B-spline of each degree. In order to simplify notation, the individual B-splines are often split up into separate partial functions defined on the knot intervals. This way e.g. $B_{0,2}(t)$ becomes $B_2^0(t)$, $B_2^1(t)$ and $B_2^2(t)$.

No restricting assumptions were made about the knots t_i so far other than that they should be a non-decreasing sequence that subdivides the parameter interval $[t_a, t_b]$. A completely valid knot sequence could, for instance, include duplicate knots, i.e. multiple knots with the same value. As pointed out in [18], each time a knot is replicated decreases the number of continuous derivatives at the corresponding curve joint by one[2]. If the basic smoothness properties of a spline curve are sufficient, the spacing between all knots can be set to be identical, resulting in a *uniform* knot sequence of the form $t_k = k$, for all $k \in \mathbb{Z}, k \leq n$. Using a uniform knot sequence, all B-splines of one degree become translates of each other – any B-spline $B_{j,d}(t)$ can be written in terms of the first B-spline, $B_{j,d}(t) = B_{0,d}(t - j) =: B_d(t - j)$.

The piecewise definition of B-splines leads to the observation that most terms in the summation of equation 3.4.15 are zero when it is evaluated for one specific t. In fact, every B-spline of degree d is only non-zero for a parameter range of $d + 1$. To account for this property, the B-splines are split into separate functions for each piece they are defined on. The parameter range for each of these new functions is then assumed to be $[0, 1]$. Table 3.4.4 shows the resulting functions for the first linear, quadratic and cubic B-splines.

Using these new simplified basis functions and setting $i = \lfloor t \rfloor$, the spline curve equation 3.4.15 can be restated in the following way which will remain the formulation of choice throughout the thesis:

$$p_d(t) = \sum_{m=0}^{d} B_d^m(t - i)\ \mathbf{c}_{i+m}. \tag{3.4.18}$$

[2]The reason for this behavior can be seen when inspecting the B-splines on a non-uniform knot sequence.

Polynomial degree	**Simplified piecewise B-splines**
Linear	$B_2^0(t) = 1 - t$ $B_1^1(t) = t$
Quadratic	$B_2^0(t) = t^2/2$ $B_2^1(t) = (-2t^2 + 2t + 1)/2$ $B_2^2(t) = (t^2 - 2t + 1)/2$
Cubic	$B_3^0(t) = (-t^3 + 3t^2 - 3t + 1)/6$ $B_3^1(t) = (3t^3 - 6t^2 + 4)/6$ $B_3^2(t) = (-3t^3 + 3t^2 + 3t + 1)/6$ $B_3^3(t) = t^3/6$

Table 3.1.: Simplified piecewise first B-splines of degree 1, 2 and 3.

3.5. B-Spline Patches and Grids

The results derived so far about spline functions can be generalized to a multivariate setting. Having seen that a univariate spline function $p_d(t)$ with two-dimensional control points $\mathbf{c}_i$ generates a curve in 2D space, the focus is now shifted to bi- and trivariate spline functions which can be used to generate deformable meshes and grids. The general observation is that the dimensionality of control point space is decoupled from the number of parameters of a spline function. In fact, the spline function is simply applied to all components of the control point vectors individually.

As illustrated in [18], a bivariate spline function $p_d(u, v)$ can be obtained by replacing the coefficients of a univariate spline function $p_d(u)$ – i.e. its control points $\mathbf{c}_i$ – with a set of spline functions $c_d^{(i)}(v)$:

$$p_d(u, v) = \sum_{i=1}^{n_1} B_{i,d}(u)\ c_d^{(i)}(v). \tag{3.5.1}$$

These functions, in turn, are defined on a set of control points $\{\mathbf{c}(i, j)\}$ with $i = 1 \dots n_1$ and $j = 1 \dots n_2$ as

$$c_d^{(i)}(v) = \sum_{j=1}^{n_2} B_{j,d}(v)\ \mathbf{c}(i, j). \tag{3.5.2}$$

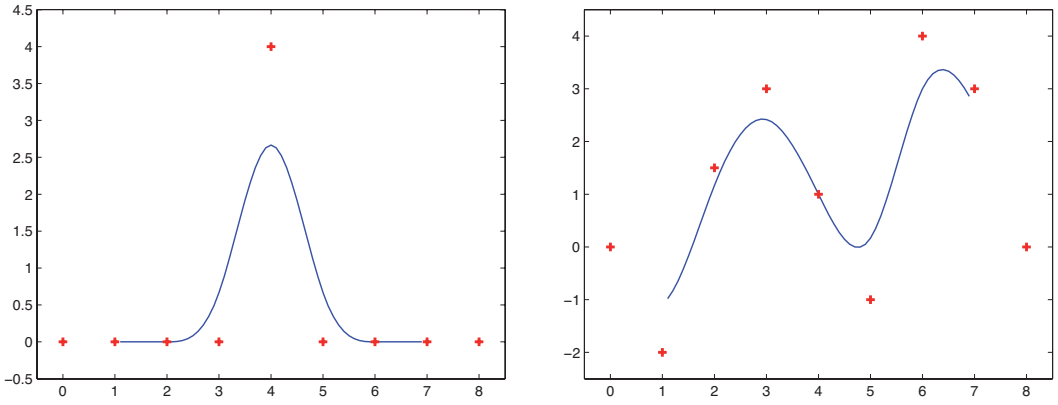
Combined the two equations yield

$$p_d(u, v) = \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} B_{i,d}(u) B_{j,d}(v)\ \mathbf{c}(i, j). \tag{3.5.3}$$

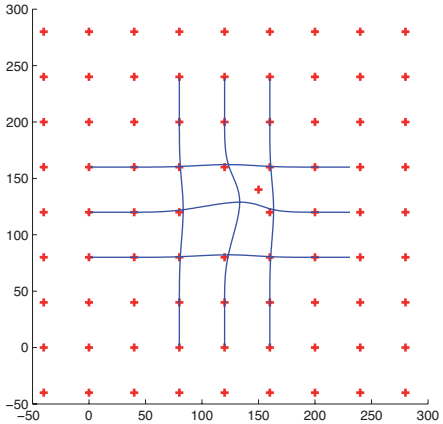
Using the notation introduced in the previous section and setting $i = \lfloor u \rfloor$, $j = \lfloor v \rfloor$, the combined equation can be rewritten as

$$p_d(u, v) = \sum_{m=0}^{d} \sum_{n=0}^{d} B_d^m(u - i) B_d^n(v - j)\ \mathbf{c}(i + m, j + n). \tag{3.5.4}$$

If the control points are two-dimensional vectors, the bivariate spline function in equation



(a) Univariate cubic spline curves trough a 1D sequence of control points that are not necessarily interpolated.



(b) A few bivariate spline curves from a patch on a 2D grid of control points.

Figure 3.6.: Examples of spline curves.

3.5.4 defines a planar 2D *patch* as a generalization of a 1D curve. Intuitively speaking, each of the two involved spline functions governs the behavior of the patch in one spatial dimension. Control points in a three-dimensional space can also be used with bivariate spline functions, resulting in a 2D *surface*, which can be visualized as a three-dimensional object. A trivariate spline function can be derived in a similar way as has been shown for bivariate spline functions. Given $k = \lfloor w \rfloor$ and a set of 3D control points $\{\mathbf{c}(i,j,k)\}$ a 3D *mesh grid* is defined as follows:

$$p_d(u,v,w) = \sum_{l=0}^{d} \sum_{m=0}^{d} \sum_{n=0}^{d} B_d^l(u-i) B_d^m(v-j) B_d^n(w-k)\ \mathbf{c}(i+l, j+m, k+n) \qquad (3.5.5)$$

As will be discussed in section 5, the registration approach based on free-form deformations will make use of B-spline patches in the two-dimensional case. For deformable registration in three dimensions B-spline mesh grids will be employed.

4. Previous Work

The following paragraphs aim to give an overview of related research that created the foundations for the work described in this thesis. A lot of theoretical and practical investigation has been performed in the field of variational deformable registration which is the basis for the optimization and regularization approach used in the proposed algorithm. On the other hand, free-form deformations have become popular more recently and most publications pursue a rather practical, application driven approach. To the best of our knowledge, a link between the variational and free-form deformations based registration approaches has not been established yet.

Pioneering work on a field closely related to the deformable registration problem was done by Horn and Schunck in 1980 [14]. Optical flow analysis deals with the reconstruction of pixel-wise movements between moving images in a sequence over time. Dense displacement fields are used to describe the motion between every pair of consecutive images. The variational approach is used in many optical flow reconstruction techniques. Apparently deformable image registration in the variational setting can be treated as a special case of optical flow reconstruction where only two images are involved.

Deformable registration has been addressed from the perspective of calculus of variations by Bajcsy and Kovacic [1]. Numerous approaches with various modifications, mainly related to the type of similarity measure and the regularization scheme, have been described ever since. Applications to many different medical problems have been proposed. A review of registration techniques can be found, for example, in [19]. Exhaustive methodological descriptions on variational deformable registration are provided by Modersitzki [21] and Hermosillo [13].

Attractive properties of both free-form deformations and spline-based approaches are the main reason why a lot of research has been conducted involving these techniques. Early applications of FFDs were mainly oriented towards computer-aided design and related disciplines. Sederberg and Parry first described the concept of deforming a geometric model by manipulating its surrounding space via a grid of control points [26]. An early work on deformation of intensity images rather than parametric surfaces was performed by Lee and Wolberg in the context of image morphing [17]. In their method a grid of control points is placed over a two-dimensional image and morphing constraints are introduced by a manual movement of control points.

The problem of image registration was first addressed using B-spline-FFDs by Szeleski and Coughlan [27]. They described a deformable registration algorithm that is very similar to the later work by Daniel Rueckert [25] which is most frequently cited as the reference for FFD-based image registration. Szeleski et al. presented their algorithm from the point of view of displacement fields and pointed out that a dense deformation field can be interpolated from a coarse grid of moving control points using B-splines. Rueckert et al. describe a more specialized method targeted at specific requirements of magnetic resonance mammography. Their algorithm uses a combination of rigid pre-registration and FFD-based deformable

registration. A global cost function is described that measures the intensity difference between the two images. A penalty term related to the smoothness of the transformation is also introduced but its computation takes place on the dense deformation field rather than on control point level. Optimization is performed by means of a general gradient descent approach and derivatives are approximated numerically.

Rohlfing and Maurer describe a similar B-spline-FFD based method that mainly involves a different type of smoothness measurement [23]. The presented incompressibility constraint is assumed to reflect physical properties of body tissue and is achieved by computing the Jacobian of the transformation function. Interestingly this regularization scheme is only calculated for the control points and not globally over the whole image. However, as opposed to the regularizer proposed in the scope of this thesis, the local incompressibility constraint is not related to the well-understood diffusion or curvature regularization methods.

An extension of the basic FFD-based registration framework that is, in parts, of great practical relevance is also given by Rohlfing et al. in [24]. Possibilities to maximize efficiency in the implementation of the complex and computationally intensive B-spline transformation function are explored. While the proposed implementation on highly parallelized shared-memory multiprocessor architectures is probably out of reach for clinical environments as of now, the other described aspects can also be used to significantly increase performance on conventional systems.

Another noteworthy and thorough investigation of FFD-based deformable registration is given by Kybic et al. [16]. Most notably, a comparison of linear, quadratic and cubic B-splines is presented with the conclusion that cubic B-splines can achieve best results with a performance drawback. Several optimization techniques, such as gradient descent, conjugate gradient and Levenberg-Marquardt are evaluated. From a theoretical point of view, they motivate the use of B-splines for the transformation model in a comparison with other candidates, such as polynomials, harmonic functions and wavelets. The compact support of B-splines and therefore the small number of basis functions that contribute to a particular control point is highlighted as the main advantage of the B-spline model.

An effort to generalize various FFD-based registration methods theoretically is made by Tustison et al. [29]. They demonstrate that the idea of characterizing deformations by means of a regular control point grid and an interpolation scheme, as with FFDs, can be treated as a special case of the finite elements method (FEM) for solution of partial differential equations. The control points in the FFD setting are viewed as an analogy to the nodes used for FEM discretization and the effect of B-splines is similar to that of shape functions in the FEM approach.

Part III.

Deformable Registration Based on B-splines

5. Algorithm Description

The algorithm developed in this work is a typical optimization algorithm that employs a target function to be minimized, a set of parameters that are manipulated in the course of the algorithm and an iterative optimization strategy. Viewed from a high level it follows typical steps that are also performed in other optimization algorithms. The details of this particular algorithm will be given in the following sections, but initially the basic steps are outlined:

- **Initialization**. A grid of control points is initialized so that the control points are evenly distributed across the volume. All control point displacements are set to zero. The reference and template volumes are scaled to the range of $[0, 1]$.
- **Iteration**. Based on the current control point configuration, a dense deformation field is computed and applied to the template image. The target cost function is evaluated based on the new warped image in comparison to the reference. An update for the control point displacements is computed by evaluating specific terms that are given by the optimization strategy. The computed update is applied to the control points and the next iteration begins.
- **Termination**. If the target function has reached a predefined threshold value or if it converges, the termination case is assumed to be reached and no more iterations are performed. The final configuration of the control points is returned as the allegedly optimal solution to the registration problem. Displacement fields and the warped template image are stored for further use.

5.1. Configuration

The general setting for the free-form deformation approach based on B-splines will now be described in a formal way. If no other claims are made, all explanations provided in subsequent sections refer to the three-dimensional case. However, the algorithm can easily be transferred to the two-dimensional setting, often by simply omitting the third coordinate in vector-valued computations.

5.1.1. Control Points

It is convenient to think of the control points as an analogy to displacement fields, just at a coarser grid. While a displacement field as defined in section 3.2 is a mapping $u : \Omega \to \mathbb{R}^d$ on the image domain, a control point grid can be interpreted as a mapping $\hat{u} : \Psi \to \Phi$, where $\Psi \subset \Omega$, $\Phi \subset \mathbb{R}^d$ and $d \in \{2, 3\}$. In the three-dimensional case the control point domain Ψ consists of all points in Ω that have a spacing of $(s_x, s_y, s_z)^\top$. This spacing can be set arbitrarily and determines the control point grid resolution.

Two sets of vectors ψ and φ are defined that enumerate the elements of Ψ and their images under $\hat{u}$. The total number of control points and thus the number of elements in ψ and φ is denoted by $N = n_x \times n_y \times n_z$. Any valid triplet of indices (i, j, k) gives an initial control point position $\psi(i, j, k)$ and the corresponding displacement $\varphi(i, j, k)$. In other words,

$$\hat{u}(\psi(i,j,k)) = \varphi(i,j,k) \quad \text{for } 0 \leq i < n_x, 0 \leq j < n_y, 0 \leq k < n_z. \tag{5.1.1}$$

The meaning of the slightly abusive use of multi-index notation in this case is obvious. An implicit rearrangement scheme simply remaps the three-dimensional index to the respective 1D index into ψ and φ. Components of one control point are written $\varphi_x(i, j, k)$, $\varphi_y(i, j, k)$ and $\varphi_z(i, j, k)$. Sometimes the vector of all x-components of the control points is addressed as φ_x and accordingly for the other directions.

As shown in Figure 5.1, the control points are positioned so that the image is surrounded outside its boundaries by one full row between control points on each side. For m_x, m_y, m_z being the size of the template image, this gives for the number of control points:

$$n_x = \left\lceil \frac{m_x}{s_x} \right\rceil + 3, \quad n_y = \left\lceil \frac{m_y}{s_y} \right\rceil + 3, \quad n_z = \left\lceil \frac{m_z}{s_z} \right\rceil + 3.$$

5.1.2. Suitable B-splines

The discussion will focus on linear and cubic B-splines since these two types are most suitable for FFD-based registration. Cubic B-splines provide C^2 smoothness properties and a reasonable efficiency, while higher order B-splines are hardly attractive because of their computational complexity. As pointed out in the evaluation section 7.2, linear B-splines in many cases do not exhibit significant disadvantages over cubic B-splines, although no smoothness properties are met. The use of linear B-splines can, however, strongly increase computation speed. Quadratic B-splines can account for marginally better results than linear B-splines at almost the computation time of cubic B-splines, so that quadratic B-splines are hardly used [16].

5.1.3. Displacement Field Generation

A dense deformation field that distributes the displacement of the control points to the voxels and that is used to warp the template image can be computed using a B-spline transformation function. This function takes as parameters a particular voxel $\mathbf{x} = (x, y, z)^\top$ and the current control point configuration φ and gives the displacement for the voxel after deformation. Evaluating this function for the whole image domain yields the displacement field. If all control points are at their initial positions, i.e. $\varphi_x = \varphi_y = \varphi_z = \mathbf{0}$, then the dense deformation field is zero everywhere. Otherwise, the displacement of any voxel is determined by the displacement of a certain number of control points around the voxel. For cubic B-splines the transformation function can be written according to [25, 17, 27] as

$$U_{\text{cubic}}(\mathbf{x}, \varphi) = \sum_{l=0}^{3} \sum_{m=0}^{3} \sum_{n=0}^{3} B_3^l(u) B_3^m(v) B_3^n(w) \cdot \varphi(i+l, j+m, k+n), \tag{5.1.2}$$

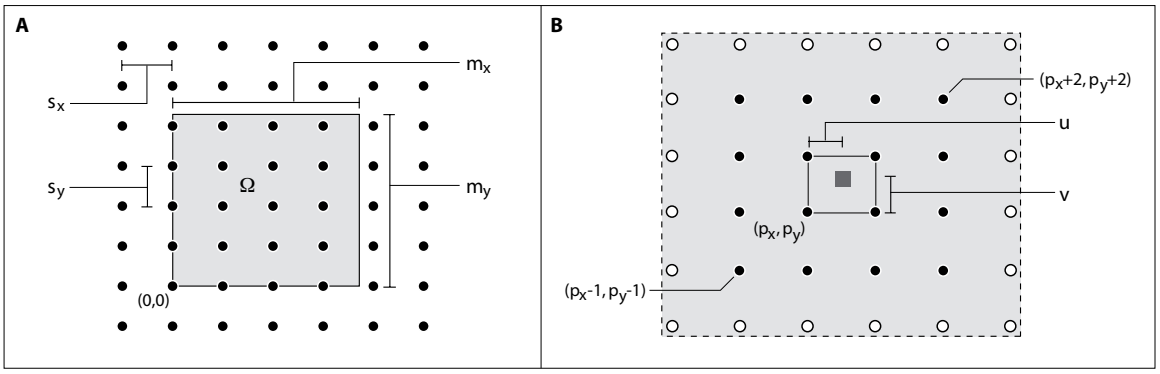


Figure 5.1.: Control point grid configuration for the two-dimensional case (A), the image is shown in gray. A magnified portion of the image (B) illustrating the computation of the transformation function. The voxel that is transformed is displayed as a solid box and all control points that are not included in the summation are marked in light gray. The set of control points in the summation is identical for all voxels inside the same box of surrounding control points. Only the weights for each of the control points change with u and v.

where $B_3^l(u)$ is the l-th cubic B-spline as listed in Table 3.1. Summation takes place over the control points in the neighborhood of voxel $\mathbf{x}$. This neighborhood consists of 16 control points in the 2D case and 64 control points in 3D, as illustrated in Figure 5.1. Let $\mathbf{p} = (p_x, p_y, p_z)^\top$ denote the control point that is closest to $\mathbf{x}$ such that

$$p_x = \left\lfloor \frac{x}{s_x} \right\rfloor, \quad p_y = \left\lfloor \frac{y}{s_y} \right\rfloor, \quad p_z = \left\lfloor \frac{z}{s_z} \right\rfloor.$$

Then the index of the basis control point is $(i, j, k) = (p_x - 1, p_y - 1, p_z - 1)$ and the last control point in the sum has the index $(p_x + 2, p_y + 2, p_z + 2)$. The parameters u, v, w are the fractional remainders of voxel coordinates between control points and represent the relative position of a voxel within its surrounding block of control points. Being arguments for piecewise uniform B-splines, they take on values between 0 and 1:

$$u = \frac{x}{s_x} - p_x, \quad v = \frac{y}{s_y} - p_y, \quad w = \frac{z}{s_z} - p_z.$$

The size of the control point neighborhood for a particular voxel explains the initial configuration of the control point grid. The rows and columns of control points outside the image boundaries are required so that the control point neighborhood is defined for every voxel in the image. This ensures that the transformation function can be computed everywhere, including the image boundaries. For linear B-splines the transformation function simplifies to

$$U_{\text{linear}}(\mathbf{x}, \varphi) = \sum_{l=0}^{1} \sum_{m=0}^{1} \sum_{n=0}^{1} B_1^l(u) B_1^m(v) B_1^n(w) \cdot \varphi(p_x + l, p_y + m, p_z + n), \tag{5.1.3}$$

which is a trilinear interpolation between the eight control points around any voxel $\mathbf{x}$.

5.2. Objective

The registration objective is posed as the problem of finding the optimal deformation that maps the template to the reference image. Since the deformation is parameterized by the control points, this aim can equivalently be formulated as that of finding the optimal control point configuration. Optimality is in this context defined by means of the target cost functional

$$E(\varphi) = S(\varphi) + \alpha R(\varphi), \tag{5.2.1}$$

which accomodates two competing goals. On the one hand, the dissimilarity term $S(\varphi)$ measures the intensity-based difference between the two volumes over all M voxels,

$$S(\varphi) = \frac{1}{M} \sum_{\mathbf{x} \in \Omega} \left(I_f(\mathbf{x}) - I_m(T(\mathbf{x}, \varphi)) \right)^2, \tag{5.2.2}$$

with a transformation function of the form $T(\mathbf{x}, \varphi) = \mathbf{x} + U_{\text{cubic}}(\mathbf{x}, \varphi)$. Obviously U_{linear} can be used equivalently in this context. The regularity term $R(\varphi)$ is designed to penalize control point displacements that potentially lead to naturally implausible deformations. A weighting factor $\alpha \in \mathbb{R}$ is introduced in Eq. (5.2.1) to govern the strength of regularization. The regularity term used in the proposed algorithm is essentially a diffusion regularizer applied to the control points,

$$R(\varphi) = \frac{1}{N} \sum_{i,j,k} \|\nabla \varphi_x(i,j,k)\|^2 + \|\nabla \varphi_y(i,j,k)\|^2 + \|\nabla \varphi_z(i,j,k)\|^2. \tag{5.2.3}$$

Here ∇ denotes a discrete approximation of the gradient operator based on central differences. Details on numerical differentiation are provided in section 6.3.

5.3. Optimization

In order to find the optimum of the cost functional in Eq. (5.2.1) one possible approach is to set its gradient to zero. This leads to the formulation

$$\nabla S(\varphi) = -\alpha \nabla R(\varphi), \tag{5.3.1}$$

which can be utilized to devise an iterative solution scheme. In the three-dimensional case this equation implicitly stands for three constraints, one for each spatial dimension. Considering for example the x-components, the gradients $\nabla S(\varphi_x)$ and $\nabla R(\varphi_x)$ are both $N \times 1$ vectors. For instance, the gradient of the dissimilarity term is

$$\nabla S(\varphi_x) = \left(\frac{\partial S(\varphi_x)}{\partial \varphi_x(0,0,0)}, \frac{\partial S(\varphi_x)}{\partial \varphi_x(1,0,0)}, \ldots, \frac{\partial S(\varphi_x)}{\partial \varphi_x(n_x, n_y, n_z)} \right)^\top, \tag{5.3.2}$$

with the partial derivatives

$$\frac{\partial S(\varphi_x)}{\partial \varphi_x(i,j,k)} = \frac{1}{M} \sum_{\mathbf{x} \in \Omega} \left(I_m(T(\mathbf{x}, \varphi)) - I_f(\mathbf{x}) \right) \cdot \frac{\partial}{\partial x} I_m(T(\mathbf{x}, \varphi)) \cdot \frac{\partial}{\partial \varphi_x(i,j,k)} T_x(\mathbf{x}, \varphi). \tag{5.3.3}$$

Since the last factor in the product, the partial derivative of the transformation function, is only non-zero in a specific region around the control point $\varphi(i,j,k)$, the summation does not have to be computed for the whole image. This fact introduces some potential for efficient implementation and will be addressed in more detail in section 6.5.2. The gradient of the regularity term for the x-component consists of the entries

$$\frac{\partial R(\varphi_x)}{\partial \varphi_x(i,j,k)} = -\frac{2}{N}\Delta\varphi_x(i,j,k) = -\frac{2}{N}\left(D_{xx}\varphi_x(i,j,k) + D_{yy}\varphi_x(i,j,k) + D_{zz}\varphi_x(i,j,k)\right), \tag{5.3.4}$$

where Δ in this context represents the discrete version of the Laplace operator. D_{xx} denotes a central difference approximation of the second unmixed partial derivative in the direction of x, compare section 6.3.

Deformable registration algorithms using the variational setting often introduce an appealing way of interpreting the optimization process. The state of minimal energy for Eq. 5.2.1 is then reached when two conceptual forces acting against each other are in an equilibrium. One of these forces is given by the gradient of the similarity term $\nabla S(\varphi_d)$ for $d \in \{x,y,z\}$, and is in this thesis denoted by $\mathbf{f}_d(\varphi)$. It „pulls“ pixels of one image towards a position that decreases the overall difference to the other image. The second force, determined by the regularizer, can be thought of as the stiffness of an elastic material that counteracts the effect of the former force. The gradient of the regularity term $\nabla R(\varphi_d)$ can be represented as a left-multiplication of the control point vector φ_d with a matrix $\mathbf{A}$ that discretizes the Laplace operator Δ.

In order to keep notation simple, the dimensionality index d is temporarily dropped for the following paragraphs. The reasoning still applies to all dimensions independently. Equation (5.3.1) can then be rewritten as

$$\mathbf{f}(\varphi) = -\alpha\mathbf{A}\varphi. \tag{5.3.5}$$

This system of equations can be approximated by applying a fix-point iteration method. In each iteration a new control point displacement $\varphi^{(t+1)}$ is obtained from the displacement in the previous iteration, $\varphi^{(t)}$, by solving the linear system of equations

$$-\alpha\mathbf{A}\varphi^{(t+1)} = \mathbf{f}(\varphi^{(t)}). \tag{5.3.6}$$

Stability of the iteration process can be improved by employing one of two possible modifications to this scheme. The simplest possibility is not to regularize the absolute new control point positions in each iteration, but only an incremental update $\hat{\varphi}$. The regularized update is then added to the existing displacement, $\varphi^{(t+1)} = \varphi^{(t)} + \hat{\varphi}^{(t+1)}$.

The second possible modification to the iteration scheme is referred to as *time-marching*. In addition to the regularization parameter α an additional weighting factor $\tau \in \mathbb{R}$ is introduced. Registration is interpreted as a process trough time and τ represents a discrete time step that is performed in each iteration. Depending on the choice of τ, convergence is reached in a smaller or greater number of iterations. Details on this behavior are given in section 7.1. The time-marching fix-point iteration scheme is stated as

$$\varphi^{(t+1)} - \tau\alpha\mathbf{A}\varphi^{(t+1)} = \varphi^{(t)} + \tau\mathbf{f}(\varphi^{(t)}), \tag{5.3.7}$$

which can be interpreted as a weighted sum of the trivial fix-point iteration $\varphi^{(t+1)} = \varphi^{(t)}$ and the general iteration scheme of Eq. (5.3.6). The influence of the latter part is damped depending on the choice of τ. In a more concise matrix notation the equation becomes

$$(\mathbf{I} - \tau\alpha\mathbf{A})\varphi^{(t+1)} = \varphi^{(t)} + \tau\mathbf{f}(\varphi^{(t)}). \tag{5.3.8}$$

Here $\mathbf{I}$ is an identity matrix of same size as $\mathbf{A}$. The resulting system of linear equations is only slightly different from the system that corresponds to the previous modification scheme and can be solved similarly, see section 6.3.2.

5.4. Regularization

Many deformable registration methods have been described that use control points but still regularize the displacement field, see e.g. [25]. However, there is a potential efficiency advantage that makes regularizing control point displacements attractive. Typically the number of control points is orders of magnitude less than the number of elements in a dense displacement field. In fact, the action of a dense diffusion regularizer in the FFD setting can be approximated with a differential operator acting on the control point displacements. This way regularization on dense deformation fields, as it is encountered in variational methods, can be put in relation with control point regularization.

The general diffusion regularization method has been introduced in section 3.3. Substituting the displacement field obtained from the B-spline transformation function $U_{\text{cubic}}(\mathbf{x}, \varphi)$ for the displacement field $u(\mathbf{x})$ in Eq. (3.3.1), the following dense deformation regularizer is obtained (the transformation function based on linear B-splines can also be used):

$$R_D(\varphi) = \sum_{\mathbf{x}\in\Omega} \|\nabla U^x_{\text{cubic}}(\mathbf{x}, \varphi)\|^2 + \|\nabla U^y_{\text{cubic}}(\mathbf{x}, \varphi)\|^2 + \|\nabla U^z_{\text{cubic}}(\mathbf{x}, \varphi)\|^2, \tag{5.4.1}$$

where the spatial components of the transformation function are indicated in superscript notation. The partial derivatives of $U_{\text{cubic}}(\mathbf{x}, \varphi)$ that appear in the gradients can be stated analytically and the solution for the direction of x is, for instance,

$$\frac{\partial}{\partial x} U^x_{\text{cubic}}(\mathbf{x}, \varphi) = \sum_{l=0}^{3}\sum_{m=0}^{3}\sum_{n=0}^{3} \left(\frac{d}{du} B^l_3(u)\right) B^m_3(v) B^n_3(w) \cdot \varphi_x(i+l, j+m, k+n). \tag{5.4.2}$$

The derivatives of the B-splines $B^l_3(u)$ are straightforward because of their nature as scalar polynomial functions of degree 3, see Table 3.1. In order to compute $d/dy\ U^x_{\text{cubic}}(\mathbf{x}, \varphi)$, the second B-spline function in the product is differentiated with respect to its parameter v. Obviously the rate of change of the transformation function in a specific spatial direction only depends on the rate of change of the B-spline that governs interpolation in that direction.

It is helpful to examine the weights that are attributed to the control points in Eq. (5.4.2). The weights that appear in the computation of the dense deformation field without any derivatives average the control points in the neighborhood of a voxel $\mathbf{x}$. As has been stated before, this neighborhood comprises a block of 64 control points for cubic B-splines in 3D. As soon as one of the B-spline terms appears in a derived form, the control points

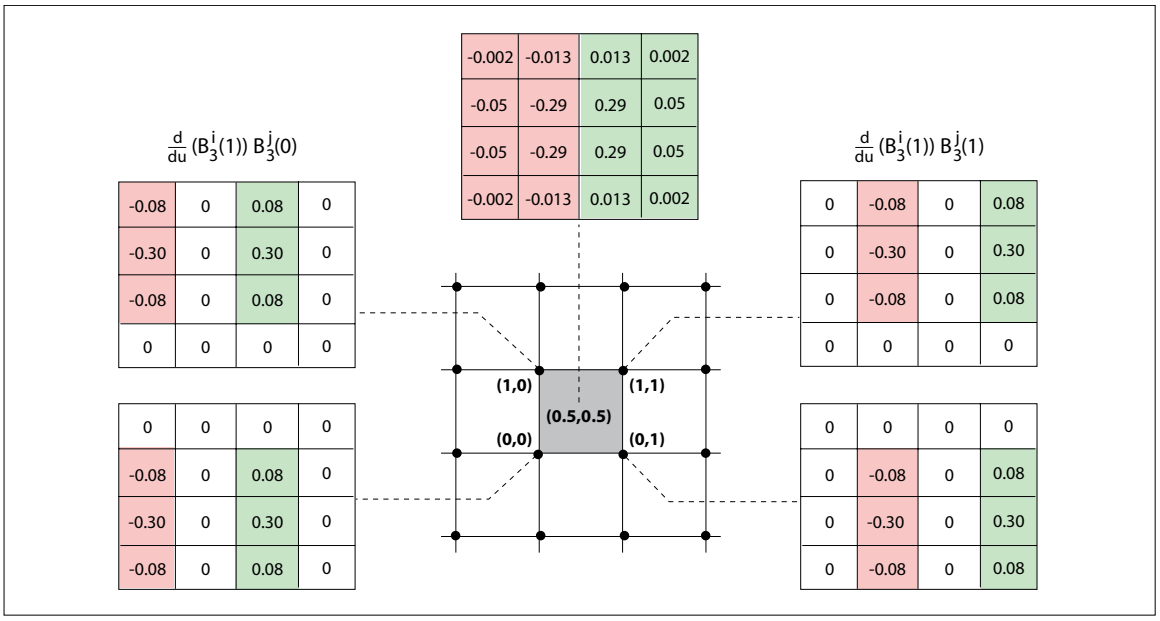


Figure 5.2.: Control point weighting scheme for a derivative in x of the B-spline transformation function. All voxels within the gray area have the same neighborhood of 16 control points in 2D. Five voxel positions for different values of (u, v) and the resulting weights for the 16 control points are given. In all cases the weighting kernels have a symmetrical structure with negative and positive values. Multiplying the control points with these kernels can be interpreted as a weighted finite difference approximation of a control point derivative.

in the neighborhood of $\mathbf{x}$ are weighted in a differential manner. Depending on the relative position of $\mathbf{x}$ within its block of 8 adjacent control points, one portion of the control points in the 64-neighborhood is weighted by negative values of same magnitude as the other control points. Figure 5.2 illustrates this observation in two dimensions for clarity. The weights that arise in the 2D version of Eq. (5.4.2) are indicated for five possible voxel positions that all share the same control point neighborhood. Apparently a weighted central difference approximation of a control point derivative in the direction of x is computed.

For linear B-splines the situation is obviously simpler since only 8 adjacent control points of any voxel constitute its neighborhood in 3D. In fact, for a derivative of the transformation function in any spatial direction, a simple finite difference between subsequent control points in that direction is computed. The reason for this behavior is that the derivatives of linear B-splines are constant functions, $d/dt\ B_1^0(t) = -1$ and $d/dt\ B_1^1(t) = 1$.

Together these observations lead to the idea of approximating the dense diffusion regularizer by means of a regularizer that uses only the control point displacements instead of repeatedly evaluating the derivative of the transformation function. The regularizer introduced in Eq. (5.2.3) exhibits exactly this behavior, given that a finite difference approximation of the gradient is employed. An experimental comparison of the dense deformation regularizer and its control point displacement counterpart is provided in section 7.4.

5.5. Multi-resolution Approach

Natural deformations that occur to tissue are typically comprised of deformations on several levels. Within a series of images that need to be registered there can be movement on a relatively large scale, for instance caused by breathing. At the same time there can also be a much smaller deformation caused by contraction of the heart muscle or by peristaltic activity. The tissue movements introduced by these effects can be related but also completely independent. In any case it is hard if not impossible to find one set of registration parameters that optimally accounts for all types of deformations within a series of images. Some kind of adaptation to movements of different magnitude has to be performed. The following sections describe the multi-resolution approach that is taken in this work.

5.5.1. Strategy

The main characteristic of the proposed algorithm that can be exploited to model deformations on different levels is the control point grid resolution [25]. The closer the control points are, the more sensitive to small deformations the registration process gets. Likewise, large deformations will hardly be reconstructed correctly since many control points have to move over relatively large distances. On the other hand, a coarse control point grid is likely not to capture deformations with an extent that is smaller than the control point distance. However, global deformations between the images to be registered can be effectively modeled with a comparably small number of control points. In addition, although implementation and efficiency discussions are postponed to later sections, it should be obvious already at this point that computational complexity increases with the control point grid resolution.

In order to combine the advantages of both a coarse and a fine control point grid it seems reasonable to divide the registration algorithm into pieces that operate on different resolutions. In an ideal case, an initial registration run on the coarsest resolution would result in a control point configuration that accounts for large deformations. Increasing the resolution and running registration again would then also capture small deformations. The final deformation would be a combination of the control point configurations from all resolution levels.

In practice not only the control point grid resolution needs to be adjusted from one level to the other, but also the image resolution. Using full resolution images with a coarse control point grid can inhibit the algorithm from correctly identifying large deformations, since too much image detail – and noise in particular – is provided. Therefore the whole multi-resolution procedure comprises the following steps:

- **Resampling**. The reference and template images are resampled to create a so-called Gaussian resolution pyramid. Essentially this means that a fixed number of image versions is created, each at half the resolution of the previous version. All images, including the original full-resolution versions, are kept temporarily.

- **Registration**. Starting with the pair of images at the lowest resolution, the full registration process as outlined in the previous section is performed. Given that the control point spacing is kept at a constant pixel value for all resolution levels, grid initialization automatically generates a coarse grid for low-resolution images.

- **Subdivision**. In some contexts this procedure is also referred to as *prolongation*. The control point configuration obtained as a registration result on one resolution level is used to generate a finer grid of control points to be used with the next higher resolution images. The registration and subdivision steps are repeated until the full-resolution images have been processed.

A realistic number of resolution levels for practical applications is 3 to 5, for input images at resolutions around 256^3.

5.5.2. Gaussian Resolution Pyramids

The concept of Gaussian resolution pyramids is so named because it involves creating low-pass filtered versions of images, a goal that can be achieved by means of Gaussian smoothing. A Gaussian pyramid consists of a number of levels, each of which is a copy of the original image at half the resolution of the previous level. Since just sampling every other pixel from an image to halve its resolution would violate the sampling theorem [15], a smoothing operation has to be performed before sampling. Starting with the original image $I^{(0)}$, the levels of a Gaussian pyramid are defined recursively as

$$I^{(t+1)}(i,j,k) = (w * I^{(t)})(2i, 2j, 2k), \tag{5.5.1}$$

where t is the level index, w is a suitable Gaussian smoothing kernel and $*$ denotes the discrete convolution operator (see section 6.4 for details). Figure 5.3 shows a Gaussian distribution along with the corresponding kernel and a schematic view of a resolution pyramid. The compression factor from one image to the next coarser level is 4 for the 2D case (2 in each dimension) and 8 in 3D. Most remarkably, storing all levels of a Gaussian resolution pyramid takes only 1/3 more space than is required for the original full resolution image in 2D, 1/7 more space is needed compared to an original volume in the 3D case [15].

5.5.3. Control Point Grid Subdivision

Once the registration process on a coarse level of the Gaussian pyramid is finished, a control point configuration is reached that reflects the deformations on this level. Before beginning registration on the next finer resolution level, the control point grid has to be subdivided so that there are twice as many control points. This is obvious, since the image resolution is doubled from one level to the next one and since the control point distance in pixels is to be kept constant for all levels. Initializing the control points on the new level to zero displacements, as is done on the coarsest level, is not possible because then the registration result from the previous level would be lost.

The new grid has to be constructed from the old one by keeping every other control point and by inserting a new control point between every pair on the coarse grid. A straightforward approach woule be to insert new control points by averaging their neighbors on the coarse grid. While this method certainly works, there is a more valid general algorithm that takes into account characteristics of the B-spline-FFD setting [9]. Using this method, exactly the same displacement field can be obtained from the subdivided control point grid as on the coarse grid, just at twice the resolution.

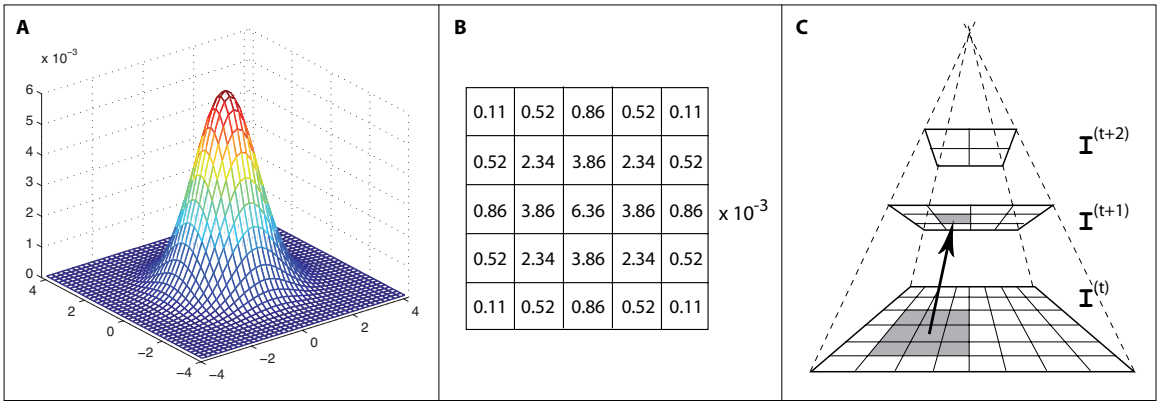


Figure 5.3.: Illustration of the concept of Gaussian pyramids. A two-dimensional Gaussian distribution $G_{0,\sigma}$ with $\sigma = 1$ (A) and a discrete convolution kernel (B) of size 5×5, obtained by sampling the Gaussian. Schematic view of a Gaussian pyramid (C); an intensity in the image on the middle level is computed as an average over a region on the lowest level weighted by the Gaussian kernel.

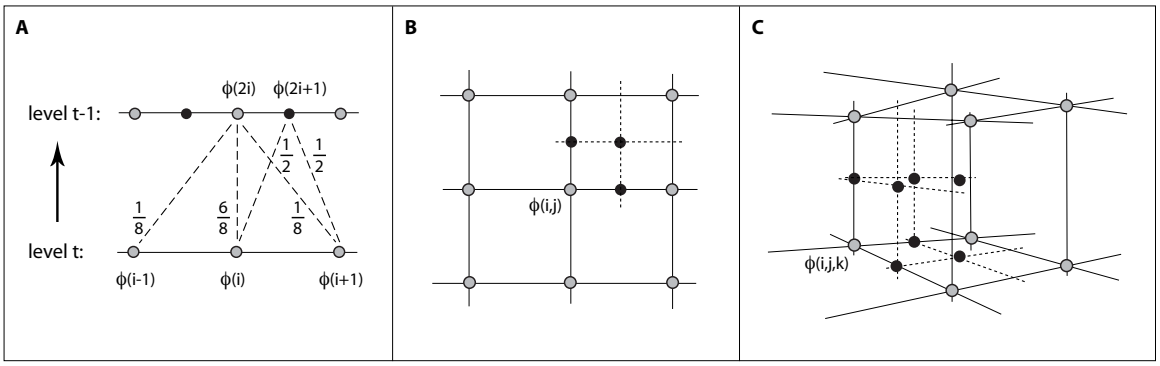


Figure 5.4.: Control point grid subdivison. In the one-dimensional case a new control point can either coincide with a control point on the coarse grid, or be between two old control points. The neighboring control points are weighted differently in the two situations (A). There are 3 configurations for non-coincident new control points (black) in the 2D case (B) and 7 configurations in 3D (C).

In one dimension there are two possible configurations for new control points. A control point can either be placed in the subdivided grid at a position that corresponds to a control point on the coarse grid, or between two control points. Let $\varphi^{(t)}$ denote the control points of pyramid level t and let $t-1$ be the next finer level (to be consistent with the notation used for resampled images). Then the two subdivision rules in 1D are

$$\varphi^{(t-1)}(2i) = \frac{1}{8}\varphi^{(t)}(i-1) + \frac{6}{8}\varphi^{(t)}(i) + \frac{1}{8}\varphi^{(t)}(i+1), \tag{5.5.2}$$

$$\varphi^{(t-1)}(2i+1) = \frac{1}{2}\varphi^{(t)}(i) + \frac{1}{2}\varphi^{(t)}(i+1). \tag{5.5.3}$$

In 2D and 3D the weighting scheme is analogous, there is simply a greater number of distinct configurations for new control points. Figure 5.4 illustrates these configurations. In 3D, a new control point that does not coincide with a control point on the coarse grid can be between old control points in any combination of the three spatial axes x, y and z.

Let $\mathbf{p} = (1/8, 6/8, 1/8)^\top$ and $\mathbf{q} = (0, 1/2, 1/2)^\top$ denote vectors containing the weights in Eqs. (5.5.2) and (5.5.3). Then the weights in $\mathbf{q}$ are used for the directions in which a new control point is between two control points on the coarser grid. Control points in the other directions are weighted by $\mathbf{p}$. For instance, in the computation of control point $\varphi^{(t-1)}(2i, 2j+1, 2k+1)$ the neighboring old control points in the y and z directions would be weighted by $\mathbf{q}$ and the weights in $\mathbf{p}$ would be applied to the neighbors in the x direction. Being related to the B-spline transformation function, the subdivision rule can be expressed in a general tensor product form. A few configurations for the 3D case are:

$$\varphi^{(t-1)}(2i, 2j, 2k) = \sum_{l,m,n=0}^{2} \mathbf{p}_l\ \mathbf{p}_m\ \mathbf{p}_n\ \varphi^{(t)}(i+l-1, j+m-1, k+n-1),$$

$$\varphi^{(t-1)}(2i+1, 2j+1, 2k+1) = \sum_{l,m,n=0}^{2} \mathbf{q}_l\ \mathbf{q}_m\ \mathbf{q}_n\ \varphi^{(t)}(i+l-1, j+m-1, k+n-1),$$

$$\varphi^{(t-1)}(2i+1, 2j, 2k) = \sum_{l,m,n=0}^{2} \mathbf{q}_l\ \mathbf{p}_m\ \mathbf{p}_n\ \varphi^{(t)}(i+l-1, j+m-1, k+n-1),$$

$$\varphi^{(t-1)}(2i, 2j+1, 2k+1) = \sum_{l,m,n=0}^{2} \mathbf{p}_l\ \mathbf{q}_m\ \mathbf{q}_n\ \varphi^{(t)}(i+l-1, j+m-1, k+n-1).$$

The extension to the remaining four configurations is straightforward.

6. Implementation Details

Having stated the deformable registration algorithm in a formal way, the following sections are intended to emphasize implementation aspects. The C++ programming language was used for implementation, mainly for the high performance native code that can be generated. Given that optimization takes place in a very high-dimensional space and two- or three-dimensional image data sets are processed, the significance of an efficient implementation cannot be overestimated. Several parts of the propsed algorithm give rise to measures that increase efficiency but that are non-trivial and therefore worth being mentioned.

6.1. Data Structures

Two classes represent the fundamental data structures that many entities of the algorithm are based on. The generic classes `Image<T>` and `Volume<T>` internally use one-dimensional arrays of the specified type `T` and provide access and modification methods for the 2D and 3D case. In addition, many image processing and vector routines are implemented in these classes, such as gradient computation or component-wise arithmetic operations. Thanks to their general character these classes can be used for images and volumes, but also for displacement fields, control point grids and other entities of similar structure. The classes `MultiresImage` and `MultiresVolume` essentially represent Gaussian resolution pyramids for 2D and 3D. Each level of a Gaussian pyramid is internally represented as an instance of `Image<double>` and `Volume<float>`, respectively. While in the 2D case double precision floating point variables are utilized for higher precision, memory constraints suggest to use single precision floats for 3D.

6.2. Application Model

As the algorithm involves a considerable amount of conceptual entities, such as images or control points, significant use of object-orientation is made. In the current state of development the focus is on the algorithm itself so that a graphical user interface is not provided. All executables can be run from the command line while supplying specific parameter files. In order to create one consistent code and application package, the programs for 2D and 3D registration are based on a common structure and objects that do not depend on dimensionality are factorized into a shared library. To effectively specify the system structure it seems reasonable to employ techniques from the domain of software engineering. In particular, the static (object) structure is displayed in a UML class diagram (Figure 6.1). Annotations are provided in the caption for quick reference.

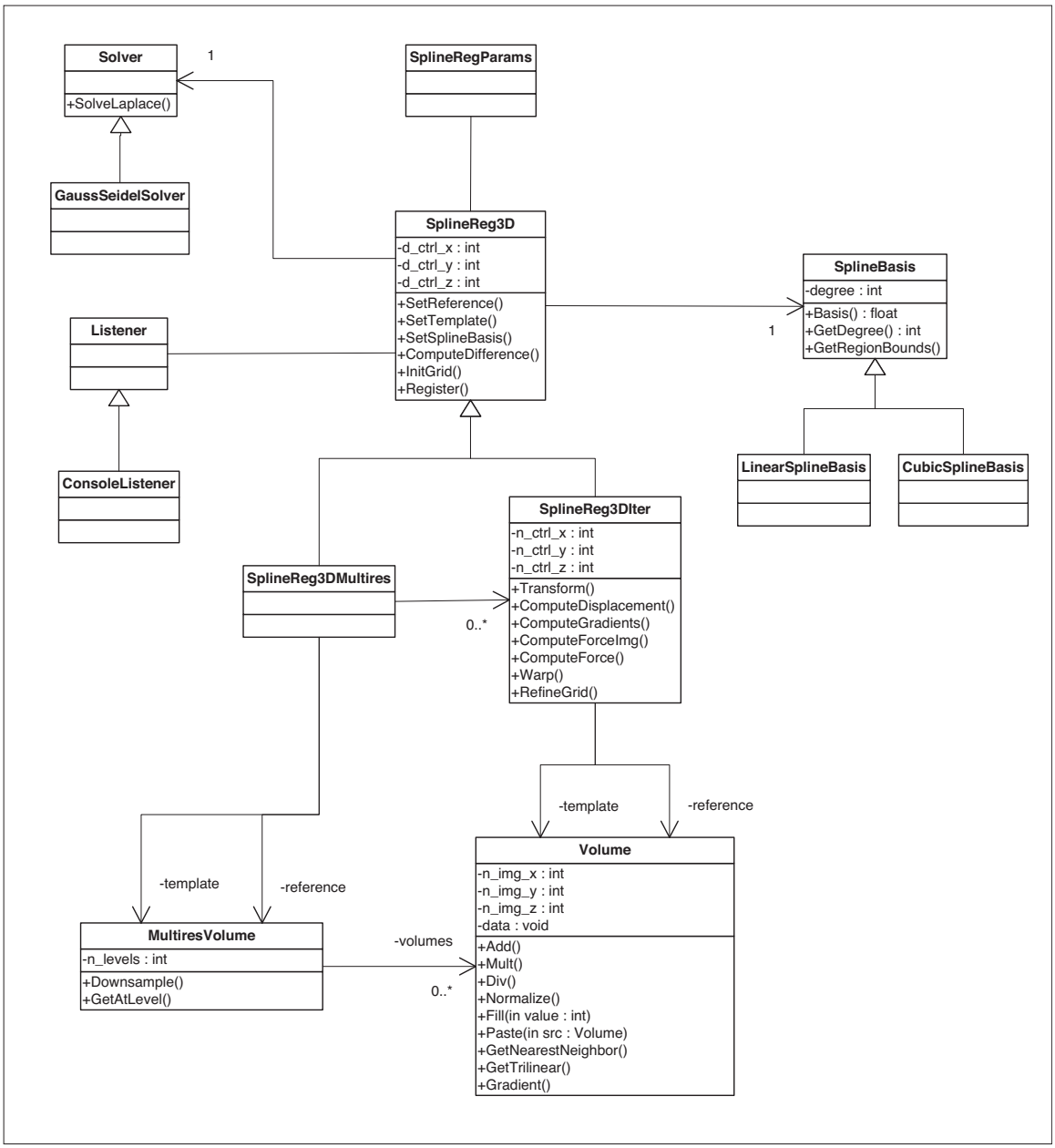


Figure 6.1.: UML class diagram of the system structure for the 3D case. The class `SplineReg3D` models main, abstract algorithmic elements such as the control point grid. The derived classes `SplineReg3DIter` and `SplineReg3DMultires` are specializations that implement details of the iterative optimization process and the multi-resolution approach. Exchangeable elements of the algorithm, such as the linear system solver and the B-splines are introduced as specializations of abstract base classes (e.g. `GaussSeidelSolver` or `CubicSplineBasis`). The `Listener` is used for program output and can be inherited to add a graphical user interface.

6.3. Numerical Techniques

Since all numerical techniques used in this thesis are related to the control points, it seems worthwhile to recall their somewhat ambiguous definition. In the three-dimensional case the control points are in a sense a displacement field $\hat{u} : \Psi \subset \Omega \rightarrow \mathbb{R}^3$, sampled with a spacing of $(s_x, s_y, s_z)^\top$. The notation used in this thesis denotes with φ the set of all N control point displacements, i.e. the 3-vectors $\hat{u}(\mathbf{x})$ for all $\mathbf{x} \in \Psi$. The $N \times 1$ vectors φ_x, φ_y and φ_z contain the respective components for all control points.

6.3.1. Differentiation

At several stages in the proposed algorithm discrete approximations to derivatives are used, for instance in conjunction with gradient and Laplace operators that are applied to the control points. Using the central difference method [22, 28], discrete partial derivatives of the control point displacements φ at the position (i, j, k) can be defined for the three spatial directions according to

$$D_x\varphi_x(i,j,k) := \frac{1}{2s_x}(\varphi_x(i+1,j,k) - \varphi_x(i-1,j,k)), \tag{6.3.1}$$

$$D_y\varphi_x(i,j,k) := \frac{1}{2s_y}(\varphi_x(i,j+1,k) - \varphi_x(i,j-1,k)), \tag{6.3.2}$$

$$D_z\varphi_x(i,j,k) := \frac{1}{2s_z}(\varphi_x(i,j,k+1) - \varphi_x(i,j,k-1)). \tag{6.3.3}$$

The second partial derivatives can be constructed as central difference approximations of the first partial derivatives, for instance

$$\begin{aligned} D_{xx}\varphi_x(i,j,k) &:= \frac{1}{2s_x}\left(D_x\varphi_x(i+1,j,k) - D_x\varphi_x(i-1,j,k)\right) && (6.3.4)\\ &= \frac{1}{2s_x}\left(\frac{1}{2s_x}(\varphi_x(i+2,j,k) - \varphi_x(i,j,k)) - \frac{1}{2s_x}(\varphi_x(i,j,k) - \varphi_x(i-2,j,k)\right)\\ &= \frac{1}{4s_x^2}\left(\varphi_x(i-2,j,k) - 2\varphi_x(i,j,k) + \varphi_x(i+2,j,k)\right). && (6.3.5) \end{aligned}$$

Using a denominator of 2 instead of 4 in the last expression allows to „contract“ the differentiation distance, so that adjacent control points are used in the difference instead of control points with a spacing of 2. These definitions and their analogous extensions to the other partial first and second derivatives can be used to define discrete versions of the gradient and Laplace operators:

$$\nabla\varphi_x(i,j,k) := (D_x\varphi_x(i,j,k), D_y\varphi_x(i,j,k), D_z\varphi_x(i,j,k))^\top, \tag{6.3.6}$$

$$\Delta\varphi_x(i,j,k) := D_{xx}\varphi_x(i,j,k) + D_{yy}\varphi_x(i,j,k) + D_{zz}\varphi_x(i,j,k). \tag{6.3.7}$$

6.3.2. Linear System Solver

The iterative optimization strategy involves solving linear systems of equations in each iteration. The exact properties of these systems depend on the type of stability modification scheme that is used. In both cases a matrix $\mathbf{A}$ is used as a discrete approximation of the

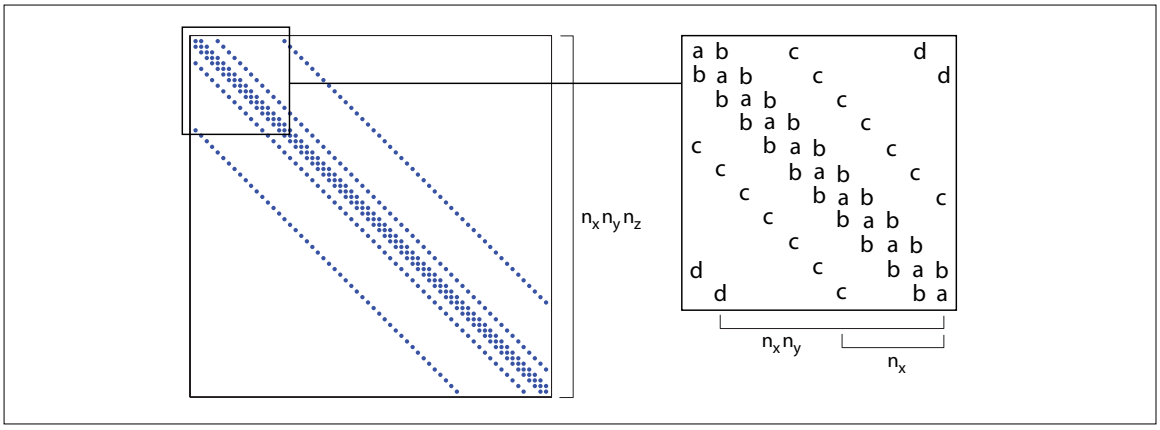


Figure 6.2.: Structure of matrix **A** representing the discretized Laplace operator. On the left side the matrix shape for a total of 64 control points (4 in each spatial dimension) is shown. It is a sparse diagonal band matrix where all values off the dotted diagonals are zero. The remaining values are $a = (2/s_x^2+2/s_y^2+2/s_z^2)$, $b = -1/s_x^2$, $c = -1/s_y^2$ and $d = -1/s_z^2$ with a common factor of α/N.

Laplace operator. It is supposed to apply the following point-wise operation to all control points simultaneously:

$$-\frac{2}{N}\Delta\varphi_x(i,j,k) = \frac{1}{N}\left(2\varphi_x(i,j,k)\,[\frac{1}{s_x^2}+\frac{1}{s_y^2}+\frac{1}{s_z^2}] - \frac{1}{s_x^2}\,[\varphi_x(i-1,j,k)+\varphi_x(i+1,j,k)]\right.$$
$$\left.-\frac{1}{s_y^2}\,[\varphi_x(i,j-1,k)+\varphi_x(i,j+1,k)] - \frac{1}{s_z^2}\,[\varphi_x(i,j,k-1)+\varphi_x(i,j,k+1)]\right). \quad (6.3.8)$$

The structure of **A** is now obvious, assuming that the vectors φ_x, φ_y and φ_y enumerate the control points in a row-column-slice order. In other words, starting from control point $\varphi(i,j,k)$ the next control point in the direction of x is stored at the following location, the next control point in y is n_x vector entries away and in z the spacing is $n_x n_y$ elements. The matrix **A** has a typical sparse pattern that is illustrated in Figure 6.2. For the time-marching situation an identity matrix is added to **A**.

In order to solve the system of linear equations in each iteration of the optimization process, the theoretical approach is to compute $\mathbf{A}^{-1}$ and to multiply it with $\mathbf{f}(\varphi^{(t)})$. However, the practical value of this concept is relatively low. A reasonable number of control points for an image of 256^3 voxels could be 26^3, resulting in a matrix **A** of size 17576^2. Stating such a matrix explicitly, especially if it is sparse, is for efficiency reasons hardly a suitable approach. Instead, the linear system of equations is solved approximately using the Gauss-Seidel method, adapted to exploit the sparsity pattern of **A**.

The Gauss-Seidel method is a general fix-point iteration technique [22]. For a system of linear equations $A\mathbf{x} = \mathbf{b}$ with $A \in \mathbb{R}^{n\times n}$ and $\mathbf{b}, \mathbf{x} \in \mathbb{R}^n$, the idea is to solve for the elements of $\mathbf{x}$ one equation at a time until convergence. The value of the i-th element at iteration

$t+1$ is obtained as

$$\mathbf{x}_i^{(t+1)} = \frac{1}{a_{ii}} \left(\mathbf{b}_i - \sum_{j<i} a_{ij} \mathbf{x}_j^{(t+1)} - \sum_{j>i} a_{ij} \mathbf{x}_j^{(t)} \right), \tag{6.3.9}$$

where a_{ij} are the elements of A. In each iteration all previously computed entries of $\mathbf{x}^{(t+1)}$ are used as well as the elements of $\mathbf{x}^{(t)}$ that are yet to be updated [22]. This way only one instance of $\mathbf{x}$ is kept in memory. For the sparse matrix case the summations can also be limited to the indices corresponding to the six non-zero elements off the main diagonal of $\mathbf{A}$. The necessary condition for convergence of the Gauss-Seidel method, namely that A is strictly diagonally dominant, is obviously fulfilled.

6.4. Image Filtering

Image filtering is a technique that is used to accomplish various tasks related to digital imaging, such as derivative approximation or smoothing. It can be approached from a signal processing point of view, then leading to frequency domain filtering, or from a spatial domain perspective. Since the theory behind filtering is not to be discussed in this context, the following sections will concentrate on spatial domain filters that are used in the implementation of the proposed algorithm.

6.4.1. Discrete Convolution

In the spatial domain, image filtering is typically performed by means of discrete convolution or correlation. These two techniques involve a filter kernel that is moved across the image to be filtered [15, 28]. At each image location an average of the neighborhood intensities is calculated, where the weighting factors are given by the entries of the kernel. Convolution and correlation only differ in the orientation of the kernel which is for convolution flipped by 180 degrees with respect to the image. More formally, for a kernel w with a side length of r and an image I, discrete convolution can be written as

$$(w * I)(x, y, z) = \sum_{i=-c}^{c} \sum_{j=-c}^{c} \sum_{k=-c}^{c} w(i, j, k) \cdot I(x - i, y - j, z - k), \tag{6.4.1}$$

where $c = \lfloor r/2 \rfloor$. To obtain the value in the filtered image at location (x, y, z), the kernel is centered around that position. If for some values of (x, y, z) the kernel exceeds the image boundaries, it is customary to either pad the image with zeros or to mirror it along its boundaries, compare e.g. [15].

6.4.2. Gaussian Smoothing

Gaussian smoothing is employed at several stages of the registration algorithm, e.g. in the multi-resolution approach when the resolution pyramid is generated. A convolution kernel for Gaussian smoothing is obtained by sampling the Gaussian distribution $G_{0,\sigma}$ at a specific resolution and by truncating values below a threshold. Usually this threshold is reached at around 5 times the standard deviation σ from the mean of the distribution [28]. This

property also allows to specify suitable values for the sampling resolution if a desired kernel size in pixels is given. Figure 5.3 shows a Gaussian distribution in two dimensions along with the resulting filter kernel of size 5×5.

6.4.3. Sobel Filter

The Sobel filter is a discrete differential operator used to compute approximations of image intensity gradients [11]. For instance, the gradient of the warped image $I_m(T(\mathbf{x}, \varphi))$ is calculated in each iteration of the registration algorithm. For a general image I the two components of the gradient in the two-dimensional case can be obtained by means of the 3×3 Sobel filter kernel as

$$\frac{1}{8}\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * I \qquad \text{and} \qquad \frac{1}{8}\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} * I, \tag{6.4.2}$$

where $*$ denotes the discrete convolution operator [15]. The Sobel kernel, in fact, can be seen as a combination of a smoothing operation with derivative approximation since for each row or column in a 2D image not only adjacent pixels in the respective direction are considered [11]. Neighboring pixels in the perpendicular direction are also included in the computation at half the weight of the pixels on the principal direction of derivation. In the three-dimensional case there are three kernels of size $3 \times 3 \times 3$ which are defined analogously.

6.4.4. Recursive Filters

The aforementioned filters are finite impulse response (FIR) filters that have a discrete filter kernel of a specific size. However, the Gaussian distribution that Gaussian kernels approximate has an infinite extent that is cut off in order to generate a finite kernel [28]. Infinite impulse response (IIR) filters typically do not use finite kernels but employ a recursive way of approximating the true impulse response. Most importantly, the computational complexity of FIR filters depends on the size of the convolution kernel. For the Gaussian filter this quantity is related to the standard deviation σ of the desired Gaussian which, in turn, is an important parameter to adjust smoothing strength. While a convolution with a kernel of size k takes in one dimension k multiplications and additions per image location, IIR Gaussian filters can be implemented to be in their complexity independent of σ.

Several such recursive filter algorithms have been described in the literature, for instance the method by Deriche [5]. The approach taken for the proposed registration algorithm is based on work by Young and van Vliet [32, 30]. They propose recursive implementations of Gaussian and derivative filters that address several issues of the Deriche filter, such as its complex kind of definition. While the derivation of the algorithms by Young et al. is beyond the scope of this thesis, the way they are applied in practice is as follows. Starting with the Gaussian smoothing operation, filtering in 2D or 3D is split up into successive filtering steps in each of the dimensions. A set of weights b, b_0, b_1, b_2, b_3 is defined that can be found in the original publication. A forward (F) and a backward (B) pass is applied to every line in the image, for each dimension. Letting $I_{\text{old}}(x), I_{\text{new}}(x)$ denote the image to be

filtered and the output image, restricted to one dimension, the two passes can be written

$$\text{F: } I_{\text{tmp}}(x) = bI_{\text{old}}(x) + \left(b_1 I_{\text{tmp}}(x-1) + b_2 I_{\text{tmp}}(x-2) + b_3 I_{\text{tmp}}(x-3)\right)/b_0, \tag{6.4.3}$$

$$\text{B: } I_{\text{new}}(x) = bI_{\text{tmp}}(x) + \left(b_1 I_{\text{new}}(x+1) + b_2 I_{\text{new}}(x+2) + b_3 I_{\text{new}}(x+3)\right)/b_0. \tag{6.4.4}$$

Here $I_{\text{tmp}}(x)$ denotes an image used for intermediate storage. Obviously boundary conditions have to be observed, so for instance zero values can be assumed outside the image boundaries. A slight modification of the forward pass while leaving the backward pass unchanged gives a derivative filter [30]. The modified forward pass is

$$\begin{aligned}\text{F: } I_{\text{tmp}}(x) = (b/2)[I_{\text{old}}(x+1) - I_{\text{old}}(x-1)] + (b_1 I_{\text{tmp}}(x-1) \\ + b_2 I_{\text{tmp}}(x-2) + b_3 I_{\text{tmp}}(x-3))/b_0.\end{aligned} \tag{6.4.5}$$

This filter still possesses smoothing properties and is in this sense similar to the Sobel operator which also combines modest smoothing with derivative approximation.

6.5. Force Computation

The notion that registration is achieved when certain antagonistic forces are in an equilibrium has been introduced in section 5.3. A little more light shall be cast on that aspect at this point. It has become customary in the literature to refer of the gradient of the dissimilarity term $\nabla S(\varphi)$ as „the force“, while the second force, the gradient of the regularity term $\nabla R(\varphi)$ is not named in particular [21, 33]. Adapting this convention, the force is written $\mathbf{f}(\varphi)$ and can be computed separately for the three spatial directions, $\mathbf{f}_x(\varphi)$, $\mathbf{f}_y(\varphi)$ and $\mathbf{f}_z(\varphi)$. In order to illustrate the motivation behind this nomenclature, the corresponding equation is restated, for instance for the x-component:

$$\mathbf{f}_x(\varphi)[i,j,k] = \frac{1}{M}\sum_{\mathbf{x}\in\Omega} \underbrace{\left(I_m(T(\mathbf{x},\varphi)) - I_f(\mathbf{x})\right)}_{\text{Intensity Difference}} \cdot \underbrace{\frac{\partial}{\partial x} I_m(T(\mathbf{x},\varphi))}_{\text{Image Gradient}} \cdot \underbrace{\frac{\partial}{\partial \varphi_x(i,j,k)} T_x(\mathbf{x},\varphi)}_{\text{Smoothing Kernel}}. \tag{6.5.1}$$

The three major quantities involved in the summation are indicated. The force in the direction of x can be evaluated for all control points $\varphi(i,j,k)$, and for each of these control points a sum over $\mathbf{x} \in \Omega$ is performed. The first quantity is simply the intensity difference image between the fixed and the moving image. The spatial gradient of the moving image can be computed using filtering techniques, such as the Sobel filter, described above. Of greater interest is the third quantity which is named *smoothing kernel* for reasons that will become apparent shortly.

6.5.1. Image Level Force

In the context of variational deformable registration, where the interpretation with forces originates from, the force term is identical to Eq. (6.5.1) except for the smoothing kernel. This term is not present in methods that do not use a control point grid (compare e.g. [33]). However, the part of the force without the smoothing kernel also plays a role in the free-form deformations framework. It is therefore separately referred to as *image level force*

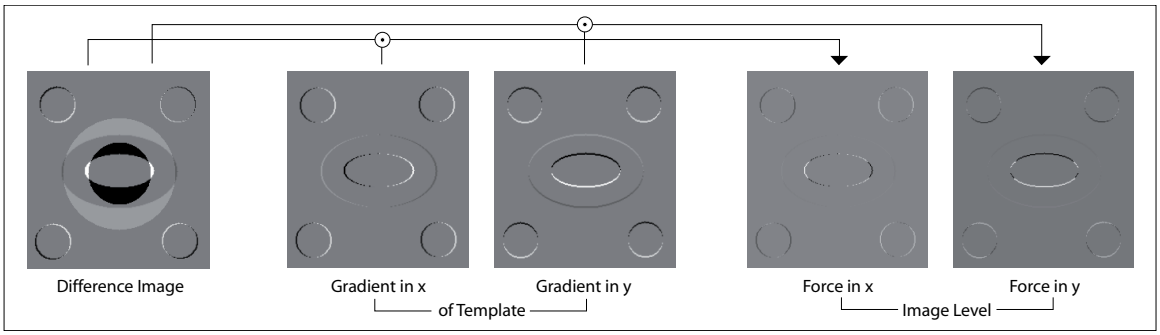


Figure 6.3.: Computation of image level force. The images are intermediate results obtained after five iterations of the registration algorithm applied to the reference and template images shown in Figure 3.1. The components of the gradient of the warped template are multiplied point-wise with the difference image to yield the components of the image level force.

since it can be evaluated for each voxel in the image domain. It gives the direction and distance how each voxel is to be moved based on the difference between the reference and the template and the gradient of the template. For the x-component the image level force can be stated as

$$\mathbf{f}_x^{\text{img}}(\varphi)[\mathbf{x}] = (I_m(T(\mathbf{x}, \varphi))) - I_f(\mathbf{x})) \cdot \frac{\partial}{\partial x} I_m(T(\mathbf{x}, \varphi)) \tag{6.5.2}$$

Obviously the image level force is a displacement field that is not regularized. In contrast to a regularized displacement field that is desirable for image registration, it typically has harsh local differences between displacement vectors. Figure 6.3 illustrates the computation of the image level force. In practice the difference image and the image level force are first calculated individually for all $\mathbf{x}$ and then multiplied and added elementwise to compute Eq. (6.5.1).

6.5.2. Control Point Force

The term that is referred to as smoothing kernel is the x-component of the partial derivative of the B-spline transformation function $T(\varphi, \mathbf{x})$ with respect to a particular control point $\varphi(i, j, k)$. Informally speaking, this partial derivative gives the rate of change of the dense deformation field for the case that one of the control points is moved. Intuitively it seems that moving one single control point should only locally affect the displacement field. In fact, it has been shown before that any given point on a spline curve is only influenced by a fixed number of control points[1]. In this sense there is a local neighborhood around each control point covering all voxels that are influenced by this control point. For the control point at original position $\psi(i, j, k)$ this neighborhood can be defined as

$$L(\psi(i, j, k)) = \{\mathbf{x} \in \Omega \quad | \quad |\mathbf{x}_d - \psi_d(i, j, k)| \leq \lambda s_d\}, \quad d \in \{x, y, z\}, \tag{6.5.3}$$

[1]This number is 2 for linear and 4 for cubic B-splines in each spatial direction.

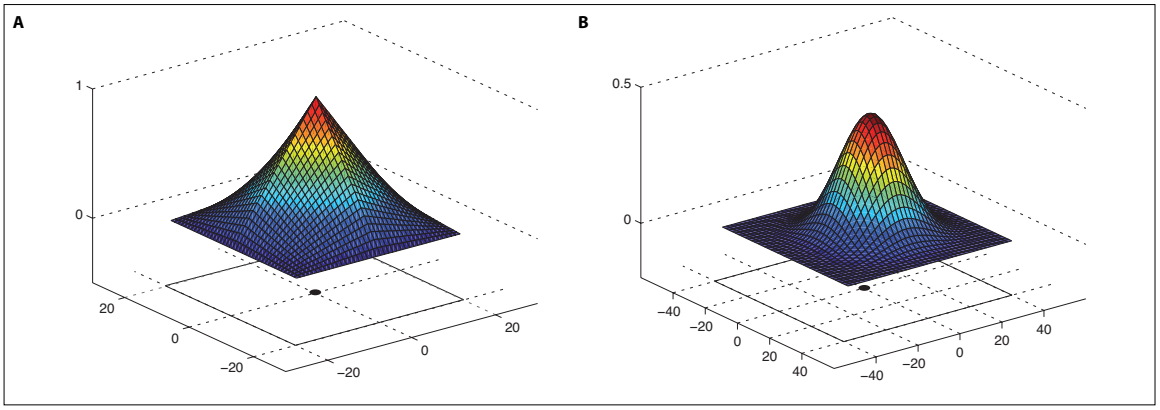


Figure 6.4.: Smoothing kernels $h_{\text{linear}}(\mathbf{x})$ and $h_{\text{cubic}}(\mathbf{x})$ in 2D. The kernels are essentially the product of linear and cubic B-splines, respectively. Both kernels are centered around a particular control point that is highlighted. In the case of linear B-splines (A), the kernel covers an image area between two control points in each direction and four control points for cubic B-splines.

where $\lambda = 1$ for linear and $\lambda = 2$ for cubic B-splines. Using this definition, the partial derivative of the transformation function with respect to one control point can be solved analytically to yield

$$\frac{\partial}{\partial \varphi_x(i,j,k)} T_x(\mathbf{x}, \varphi) = \begin{cases} B_3^a(u) B_3^b(v) B_3^c(w) & \text{for } \mathbf{x} \in L(\psi(i,j,k)) \\ 0 & \text{for } \mathbf{x} \notin L(\psi(i,j,k)) \end{cases}, \tag{6.5.4}$$

with $a = |p_x - i - 1|$, $b = |p_y - j - 1|$, $c = |p_z - k - 1|$ and u, v, w as defined in Eq. (5.1.2). Since this derivative takes on identical values in the neighborhood of any control point $\varphi(i,j,k)$, general partial functions independent of a particular control point can be defined:

$$h_{\text{linear}} : [0, 2s_x] \times [0, 2s_y] \times [0, 2s_z] \to \mathbb{R}, \quad h_{\text{linear}}(\mathbf{x}) = B_1^{p_x}(u) B_1^{p_y}(v) B_1^{p_z}(w), \tag{6.5.5}$$

$$h_{\text{cubic}} : [0, 4s_x] \times [0, 4s_y] \times [0, 4s_z] \to \mathbb{R}, \quad h_{\text{cubic}}(\mathbf{x}) = B_3^{p_x}(u) B_3^{p_y}(v) B_3^{p_z}(w). \tag{6.5.6}$$

Here p_x, p_y, p_z again denote the principal control point that is closest to $\mathbf{x}$. As Figure 6.4 illustrates in the 2D case, these two functions have the typical shape of smoothing kernels, suggesting several efficient techniques for force computation.

A straightforward improvement over simply implementing the force equations is to precompute the kernels before registration. The kernels only depend on the control point distances s_x, s_y, s_z that are known in advance. During registration the precomputed kernel is placed ontop of the image level force for each control point, and all values under the kernel are weighted and summed up. Alternatively, the image level force can be convolved with the precomputed smoothing kernel. The convolution result is then sampled at the control point positions to obtain the control point force, for instance:

$$\mathbf{f}_x(\varphi)[i,j,k] = (h_{\text{cubic}} * \mathbf{f}_x^{\text{img}}(\varphi))\ [\psi(i,j,k)]. \tag{6.5.7}$$

There is an obvious computational overhead in this approach – the filter mask is moved

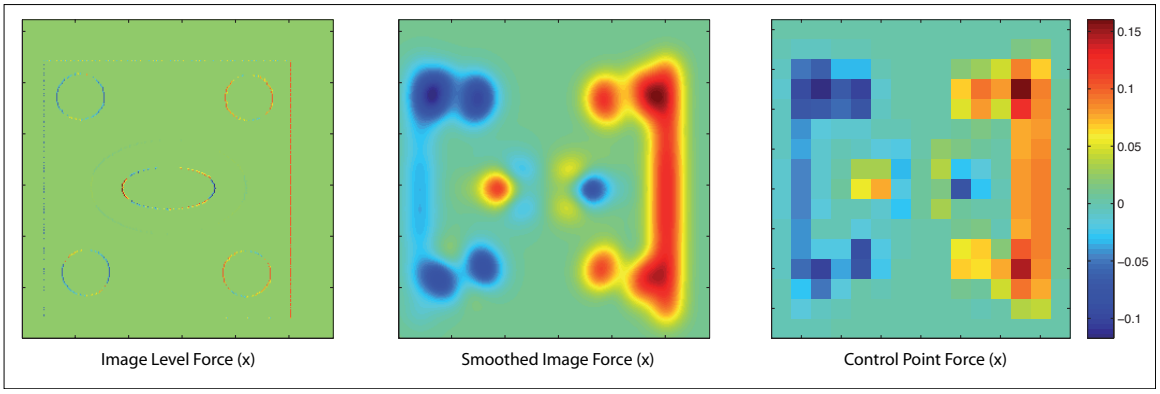


Figure 6.5.: Computation of control point force. The force acting on a specific control point is determined by the underlying image level forces in a region around the control point. The central image is the result of convolving the image force in the direction of x (left) with the cubic B-spline kernel (Figure 6.4). Sampling the smoothed image force at the grid knots gives the respective component of the control point force (right).

across the whole image domain for convolution, while only the values at the control points are required. However, this overhead can be cancelled out if an efficient filtering technique is used, such as recursive filtering, compare section 6.4.

6.6. Precomputing B-Spline Coefficients

The most demanding part of the registration algorithm from a computational complexity point of view is the generation of the dense deformation field based on a control point configuration. This step consists of computing weighted averages of control point displacements in a neighborhood around each image pixel. Depending on the order of B-splines that are used this neighborhood is comprised of the image area between two or four control points in each dimension. Rohlfing et al. [24] describe an interesting way how displacement field generation can be implemented efficiently. For convenient reference the B-spline transformation function that is to be implemented efficiently is restated at this point. For a given control point displacement φ, the displacement of a voxel $\mathbf{x}$ is given by

$$U_{\text{cubic}}(\mathbf{x}, \varphi) = \sum_{l=0}^{3} \sum_{m=0}^{3} \sum_{n=0}^{3} B_3^l(u) B_3^m(v) B_3^n(w) \cdot \varphi(i+l, j+m, k+n), \tag{6.6.1}$$

where (i, j, k) is the index of the control point cell surrounding $\mathbf{x}$ and (u, v, w) is the relative position of $\mathbf{x}$ within the cell. While this equation has to be evaluated for the whole image domain, there are redundancies in a straightforward implementation of the equation. The axes of the control point grid are by definition parallel to the axes of the reference image. As a result, the sequence of values i and u when moving horizontally through an image is identical for all rows. These values can be precomputed once before registration and reused for each row. Consequently also the B-spline basis function values $B_0(u)$, $B_1(u)$, $B_2(u)$ and

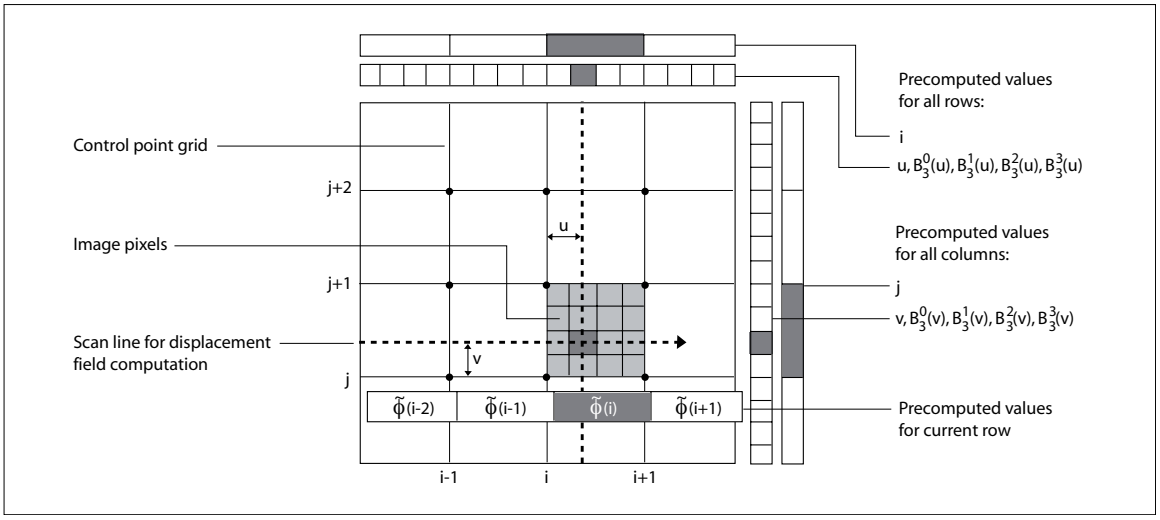


Figure 6.6.: Efficient implementation of displacement field generation based on precomputing B-spline coefficients. The 2D case is illustrated but the 3D case is a straightforward extension. All pixels within the light gray box share the indices i, j determining their corresponding set of control points. When moving along the scan line, v is constant as well as the respective B-splines. These values can be precomputed and are valid for all scan lines. The values $\tilde{\varphi}(i)$ are only valid for the current scan line and are therefore precomputed for every row.

$B_3(u)$ can be precomputed and stored in lookup tables. Moreover, the sequence of values j, v and the corresponding basis functions is identical for all vertical columns. A similar observation can be made for k and w and the slices of an image.

Having precomputed these values, another source of redundant calculations can be addressed. For any row through a 3D image, the control point indices i, j, k are constant inside one control point cell. When the next horizontally adjacent control point cell is reached, only i is incremented by one. In addition, the relative offsets of a pixel v, w in the y and z directions do not change on a row within a control point cell. In a horizontal direction only u changes from pixel to pixel. Therefore also $B_3^m(v)$ and $B_3^n(w)$as well as their products with the control points are constant for each row inside a control point cell. These observations suggest to split Eq. (6.6.1) into a part $\tilde{\varphi}$ that is constant within one control point cell, and the remaining parts of the equation that change. The constant part is then computed once for each control point cell i in a row:

$$\tilde{\varphi}(i) = \sum_{m=0}^{3} \sum_{n=0}^{3} B_3^m(v) B_3^n(w) \cdot \varphi(i, j+m, k+n). \tag{6.6.2}$$

The whole transformation function that is evaluated for every image pixel simplifies to

$$U_{\text{cubic}}(\mathbf{v}, \varphi) = \sum_{l=0}^{3} B_3^l(u) \cdot \tilde{\varphi}(i+l). \tag{6.6.3}$$

For voxels in adjacent control point cells three out of four addends in the sum are identical, suggesting to precompute the values $\tilde{\varphi}$ for each image row. Although the performance gain from these transformation is most significant for the 3D case, since then two summations can be factorized out, an analogous transformation can also be applied to the 2D case. Figure 6.6 illustrates the procedure of computing the displacement field along a scan line.

6.7. System Overview

Having presented all major components of the proposed registration algorithm from a theoretical point of view and having illustrated specific implementation issues, an overview of the whole system is now given. The UML activity diagram shown in Figure 6.7 is probably the most concise tool for this purpose.

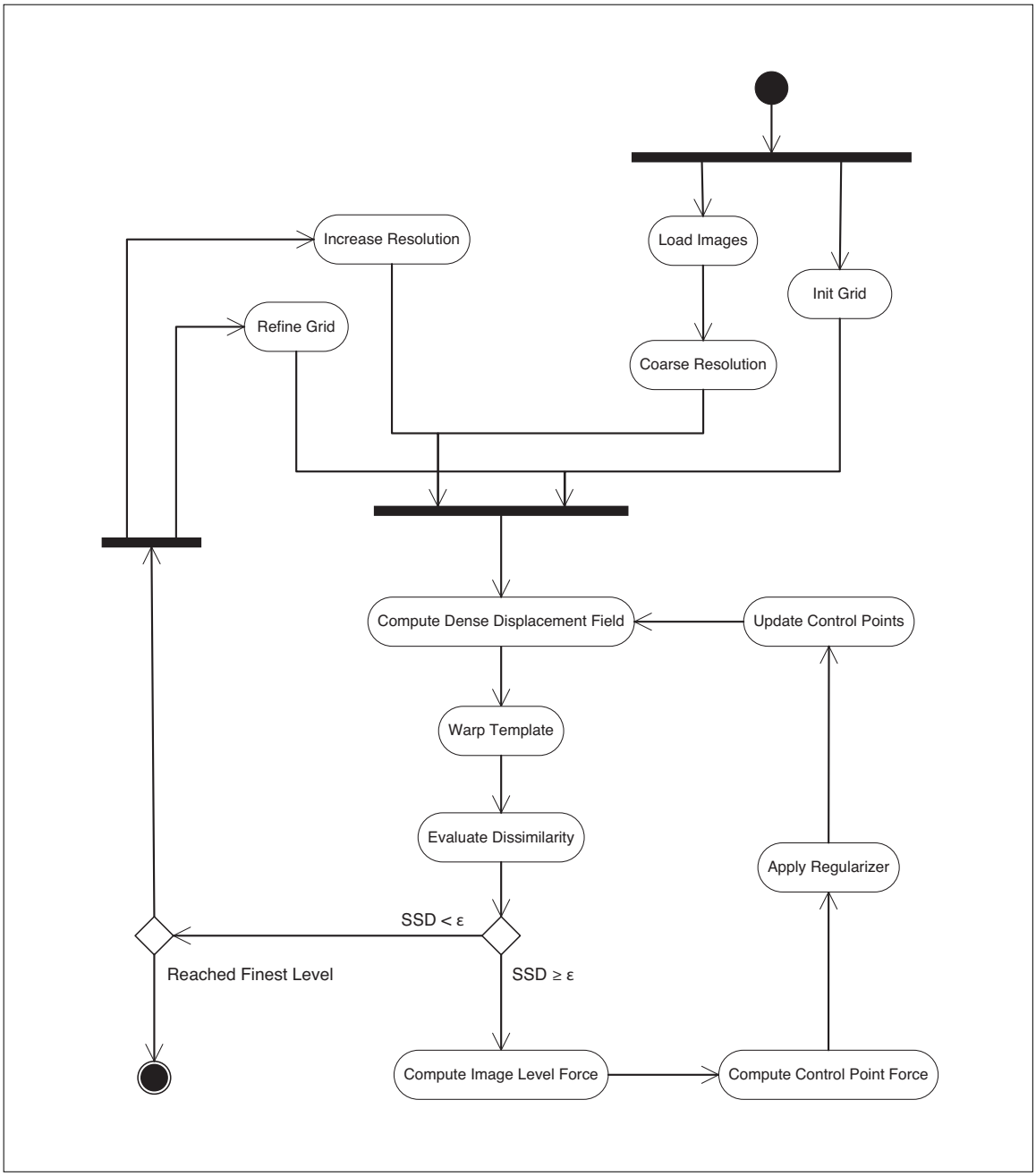


Figure 6.7.: UML activity diagram for the main components of the deformable registration algorithm.

Part IV.

Evaluation

7. Synthetic Data

A number of experiments have been conducted in order to evaluate the proposed registration algorithm. Measurements on synthetic data sets are used to demonstrate the effectiveness and performance of the algorithm. Crucial properties of the algorithm such as the influence of registration parameters are evaluated. The important intensity bias phenomenon caused by the choice of similarity measure is addressed as well. In addition, medical data sets are utilized to illustrate the applicability of the registration method to typical problems in the clinical setting. Ground truth data is incorporated both into the synthetic and the medical data experiments for objective evaluation. All experiments are performed on a 2.4 GHz Intel Core 2 Duo system equipped with 2 GB of memory and running Windows XP.

7.1. Registration Parameters

The most important parameters of the proposed registration algorithm are the following:

- choice of linear or cubic B-spline basis functions,
- control point grid spacing,
- regularization strength parameter α,
- time marching step size parameter τ.

A study of the first two parameters is given in subsequent sections since evaluation is performed in conjunction with the synthetic ground truth experiments. The parameters α and τ mainly influence the optimization process and convergence behavior of the registration algorithm. Getting back to the intuition of forces that counteract each other during registration, α determines the rigidity of the control point grid against the image force. A relatively high value for α results in a comparably rigid control point grid and therefore regularization is strong. Control points that are in a spatial neighborhood of each other move together and reduce the local impact of the image force. On the other hand, if a low value is chosen for α, then the control points react more directly to the image force and move more freely.

The ability to adjust regularization strength in the aforementioned sense is of great importance for achieving reasonable registration results. With regard to the ill-posedness of the deformable registration problem, regularization prohibits the algorithm from only reducing image dissimilarity. If no regularization is performed, the dissimilarity between the reference and template image can often be almost completely removed, although in that case the process should more precisely be called *morphing* than *registration*. Especially if the multi-resolution strategy is employed, variable control point rigidity on the inidividual resolution levels is valuable. On a coarse resolution level regularization can be set to be rather strong, since global deformations are recovered on coarse levels. On a finer level,

global deformations are assumed to be already reconstructed and regularization can be relaxed in order to capture more detailed, local deformations.

Unfortunately the regularization parameter α is coupled with convergence properties for the fix-point approach where only control point updates are regularized. As Figure 7.1 demonstrates, changing α has a clear influence on regularization strength, as desired. In the case of weak regularization ($\alpha = 6$), the control point configuration after registration has large local differences in control point displacements. The control points clearly follow the shape of the reference image (the same images as in Figure 3.1 are used). For the higher values $\alpha = 24$ and $\alpha = 42$ the control point displacements are rather smoothly distributed over all control points. As can be also seen from the figure, convergence behavior changes with α, an aspect that is generally not desired. Strong regularization is in this way always tied to slow convergence and for low values of α optimization can become unstable, begin to oscillate (Figure 7.1) or even to diverge.

The time-marching fix-point iteration approach is a remedy for this problem since convergence is controlled with a dedicated parameter, the time step size τ. Regularization strength can now be adjusted to a larger degree of freedom using α than for the previous iteration approach. If τ is set to a relatively low value, intutively speaking small steps are performed in each iteration. A high value of τ results in larger steps that are taken, leading to faster convergence, while leaving the rigidity properties of the control point grid unchanged. This behavior is illustrated in Figure 7.2 – different values of τ influence the duration until convergence but the general shape of the dissimilarity graph remains similar. Most importantly, as can be seen in the case for $\tau = 34$, stability of the optimization process is not adversely affected by a large step size parameter.

7.2. Ground Truth Experiments

The general idea of ground truth experiments is that some sort of data is used which is known to be correct in some sense. Typically this data is chosen so that the output of the algorithm to be evaluated can match it. The precision and error introduced by the algorithm can be assessed by comparing results with the ground truth. Ground truth for the 2D experiments is provided by deforming a synthetic image using a known control point configuration. The resulting deformed image is treated as the reference for a series of measurements while the template image is given by the original, undeformed image. Registration accuracy can be assessed by comparing the displacement field obtained by the registration algorithm with the ground truth displacement computed from the known control point configuration.

The synthetic image that is used for the experiments, shown in Figure 7.3, is a checker board of size 300×300 pixels with random intensities. A control point configuration that is sinusoidal in the x and y directions and that is based on a fixed spacing is used to generate the ground truth displacement field and the artificial reference image. Registration is performed for various initial control point spacings ranging from 5 to 50 pixels at increments of 5. All measurements are repeated for linear and cubic B-splines using otherwise identical parameters, allowing to make several observations.

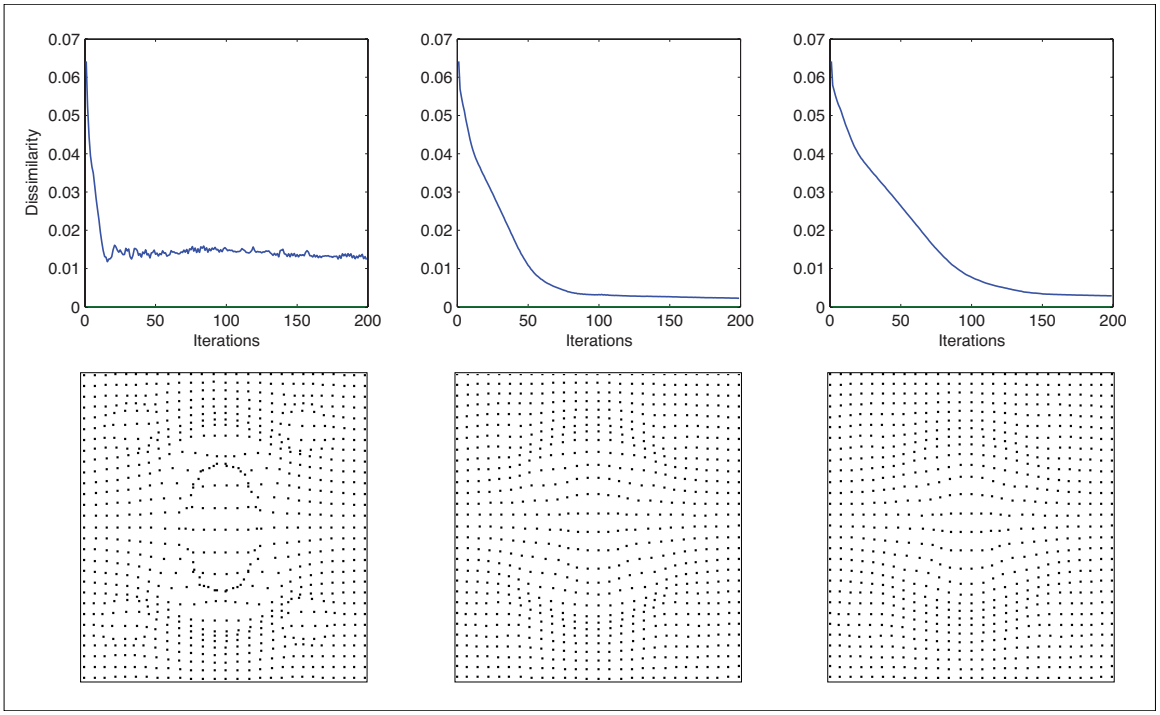


Figure 7.1.: Fix-point iteration with regularization of control point update. SSD dissimilarity measure over the course of 200 iterations (top row) for $\alpha = 6$, $\alpha = 24$ and $\alpha = 42$ (left to right). Corresponding control point distributions after registration for reference and template images shown in Figure 3.1.

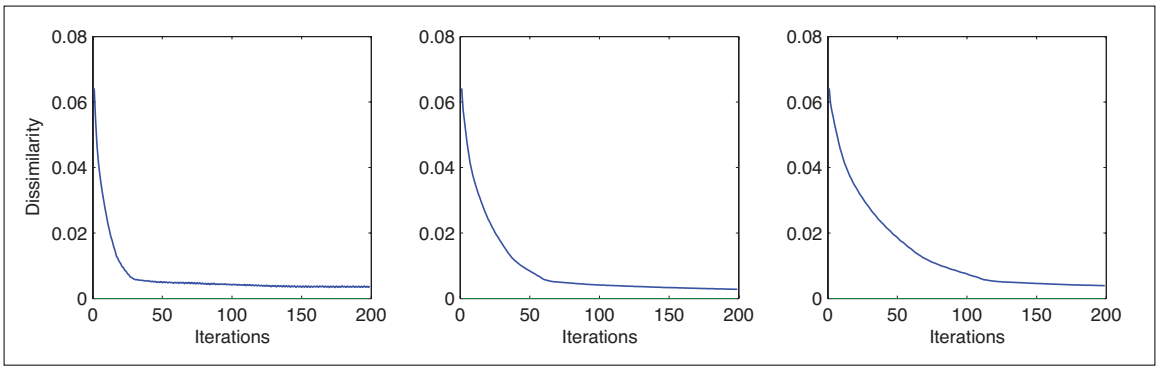


Figure 7.2.: Fix-point iteration using time-marching method. SSD dissimilarity measure over the course of 200 iterations (top row) for constant α and $\tau = 34$, $\tau = 15$ and $\tau = 8$ (left to right). Varying τ only influences convergence behavior without changing regularization strength.

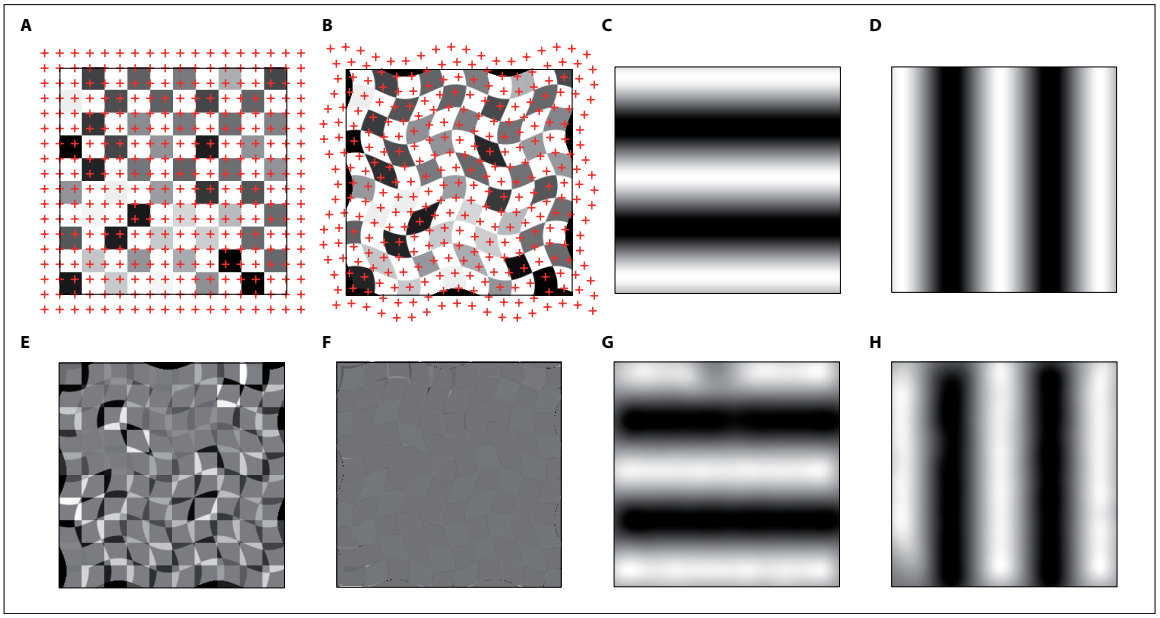


Figure 7.3.: Ground truth experiment for evaluation of registration quality. Fixed control point grid and original synthetic image (A), known control point configuration and deformed image (B). Resulting ground-truth displacement field (C, D). Difference images before and after registration (E, F), components of the reconstructed displacement field (G, H).

7.2.1. Dissimiarlity after Registration

Registration based on both linear and cubic B-splines can yield comparably low SSD dissimilarity values after registration for moderate control point spacings. As can be seen in Figure 7.4, control point spacings between 5 and 20 pixels give similar results for both types of B-splines. For larger control point spacings, cubic B-splines show better tolerance and only start to produce significantly worse registration results at spacings around 35 pixels. This aspect can be explained by the smoothness properties of displacement fields generated using cubic B-splines. The deformation shape between control points follows that of a cubic polynomial, while for linear B-splines straight lines are the basis. As in the context of interpolation, linear interpolation requires a larger number of control points than higher degree interpolation to achieve comparable results. However, linear B-splines are attractive nonetheless for practical applications because of their higher computational efficiency (see section 7.2.4). Since in practice typically relatively small control point spacings are used in order to capture small image details, the lack of smoothness inherent in linear B-splines can often be neglected.

7.2.2. Magnitude of Difference

In order to assess registration quality by comparing displacement fields reconstructed during registration with the ground truth displacements, it is necessary to devise a suitable similarity metric. Several measures have been proposed in the literature for this purpose, most

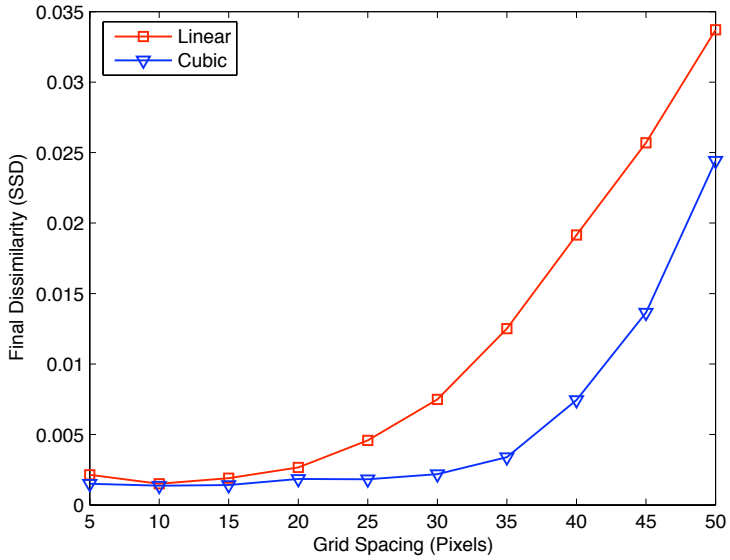


Figure 7.4.: SSD dissimilarity after registration for linear and cubic B-splines and for different control point grid resolutions. Linear B-splines yield final dissimilarity values that are comparable to those achieved using cubic B-splines for moderate control point spacings up to 20 pixels.

notably in the context of optical flow reconstruction [20] where displacement fields are used to describe changes between moving images in a sequence. A straightforward approach is to average the magnitude of difference between all vectors in a reconstructed displacement field and their corresponding vectors in the ground truth displacement field. For a vector $\mathbf{c}$ in the ground truth and a reconstructed vector $\mathbf{r}$, the simple magnitude of difference is

$$e_{\text{mod}}(\mathbf{c}, \mathbf{r}) = \|\mathbf{c} - \mathbf{r}\|. \tag{7.2.1}$$

The measure that can be used to compare two displacement fields u_c and u_r consisting of M vectors can then be stated as

$$E_{\text{mod}}(u_c, u_r) = \frac{1}{M} \sum_{\mathbf{x} \in \Omega} e_{\text{mod}}(u_c(\mathbf{x}), u_r(\mathbf{x})). \tag{7.2.2}$$

While this measure gives meaningful values in the unit of pixels, it does not take into account how strong a particular displacement is in the ground truth. Obviously, if a very small displacement is missed to a certain degree, the effect on the warped image can be less severe than if a large displacement is incorrectly reconstructed. Furthermore, a relative measure can be of advantage that gives a percentage to which the vectors in a reconstructed displacement field match the length of those in the ground truth on average.

This aim can be approached by normalizing e_{mod} with the magnitude of the ground truth vector $\mathbf{c}$. The error is then a percentage relative to the length of the correct vector. A remaining problem is that errors in very small displacements become disproportionally

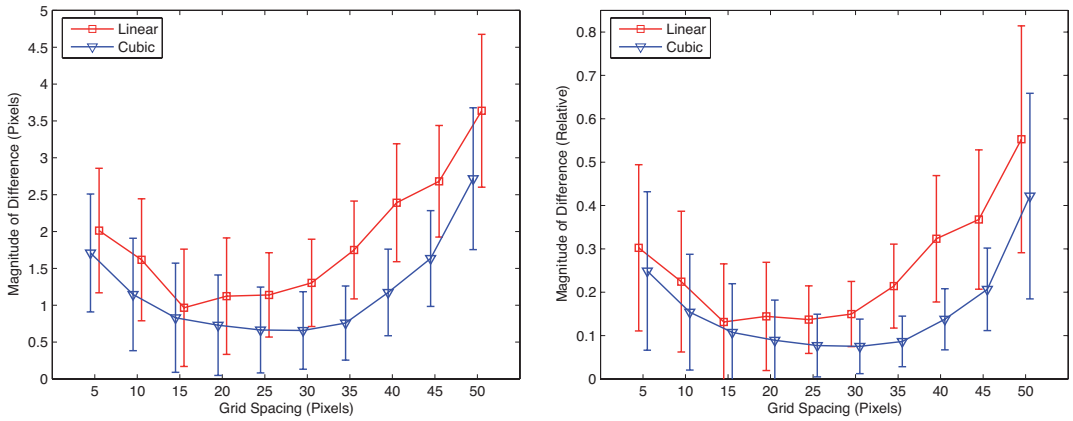


Figure 7.5.: Magnitude of vector difference between ground truth and reconstructed displacement fields. The absolute measure E_{mod} in pixels is shown (left) and the normalized relative measure E_{nmd} (right) with a significance threshold $T = 0.5$. Best values are achieved for cubic B-splines and control point spacings between 15 and 35 pixels with average difference vector magnitudes of less than one pixel or less than 10 percent of difference to the ground truth. Data points represent mean values, standard deviations are indicated as vertical bars.

prominent on average, since the error is now a relative value. A remedy pointed out in [20] is to use a significance threshold T that determines which displacements are to be disregarded because of their marginal length. The normalized magnitude of difference measure can be stated as

$$e_{\text{nmd}}(\mathbf{c},\mathbf{r}) = \begin{cases} \frac{\|\mathbf{c}-\mathbf{r}\|}{\|\mathbf{c}\|} & \text{if } \|\mathbf{c}\| \geq T \\ \frac{\|\mathbf{r}\|-T}{T} & \text{if } \|\mathbf{c}\| < T \text{ and } \|\mathbf{r}\| \geq T \\ 0 & \text{if } \|\mathbf{c}\| \leq T \text{ and } \|\mathbf{r}\| \leq T \end{cases} . \tag{7.2.3}$$

The averaged measure E_{nmd} for two displacement fields is defined as for E_{mod}. Figure 7.5 gives the statistics for a series of experiments with the ground truth data set described above. The simple and normalized magnitude of difference measures have been used for comparison. While the general shape of the graphs is almost identical, the relative measure exhibits a slower increase for larger grid spacings as compared to the absolute measure. Moreover, the standard deviations using the normalized measure are smaller for grid spacings between 15 and 45 pixels. A reason for this behavior is the significance threshold, set to $T = 0.5$ pixels. As expected, it accounts for the marginal importance of errors in regions of small displacements.

7.2.3. Angular Error

Another measure that is suitable for displacement field comparison described in [20] is called angular error. Being simple and intuitive, this measure can be used in conjunction with the normalized magnitude of difference for additional insight. The angular error is defined as the directional difference between the two vectors $\mathbf{c}$ and $\mathbf{r}$, where the former is the ground

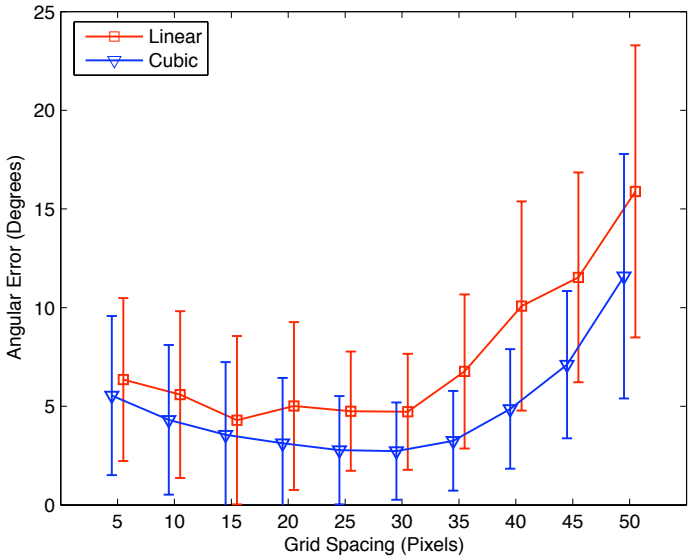


Figure 7.6.: Angular error between ground truth and reconstructed displacement fields. Best values for E_{ang} are achieved for cubic B-splines and control point spacings between 15 and 35 pixels with average angular errors around 4 degrees. Data points represent mean values, standard deviations are indicated as vertical bars.

truth and the latter is the reconstruction:

$$e_{\text{ang}}(\mathbf{c}, \mathbf{r}) = \cos^{-1}(\|\mathbf{c}\| \cdot \|\mathbf{r}\|). \tag{7.2.4}$$

Similar to the previous measures, e_{ang} is evaluated for all pairs of vectors at corresponding locations in the ground truth and the reconstructed displacement field in order to compute the average angular error, denoted by E_{ang}. It is customary to add a third coordinate to each vector in 2D and a fourth coordinate in 3D with a constant small value such as $\delta = 1$. As pointed out in [20], the influence of angular discrepancies for vectors with a magnitude less than δ is decreased this way. For strong displacements with a larger magnitude the additional coordinate is negligible, increasing the proportional influence of angular errors at large displacements.

Once again, experiments are repeated for different control point grid spacings and for linear and cubic B-splines. As Figure 7.6 illustrates, best results are achieved using cubic B-splines and a control point spacing between 15 and 35 pixels, where the mean angular error is around 4 degrees. Given the fact that the synthetic images used for experiments contain regions of constant intensities, offering less „grip“ for the registration algorithm, these values are competitive.

7.2.4. Processing Time

The main advantage of linear B-splines is their lower computational complexity that can significantly increase registration speed. Using the efficient B-spline coefficient precomputation

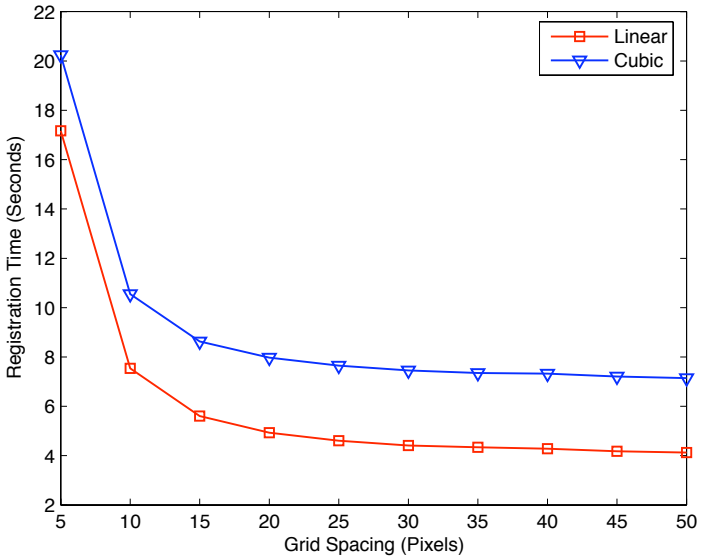


Figure 7.7.: Computation time for linear and cubic B-splines and for different control point grid resolutions. Images are of size 300×300 pixels. The same, sufficient number of iterations (200) is performed in all measurements for comparability. Linear B-splines offer better computational efficiency.

technique proposed in [24] and the machine described above, the difference in registration time between linear and cubic B-splines is reduced to an additive constant independent of the control point spacing. Figure 7.7 shows that in the 2D case this constant is around 3 seconds. Total registration durations for linear B-splines and the data described above are between 4 and 7 seconds. Shorter times are achieved on coarser control point grids. It can also be deduced from the figure that control point spacings below 15 pixels yield a disproportionate increase in registration time.

7.3. Intensity Bias in Force Computation

As has been pointed out in earlier sections, the force $\mathbf{f}_d(\varphi)$ acting on control points (for each dimension $d \in \{x, y, z\}$) is computed by smoothing and subsampling the image level force $\mathbf{f}_d^{\text{img}}(\varphi)$. This force is determined by the type of dissmilarity measure used in the overall registration energy functional. The discussion so far assumed that the sum of squared differences measure (SSD) is employed but in practice there is a significant shortcoming associated with this measure. The phenomenon referred to as *intensity bias* is addressed e.g. in [33] and can result in an unexpected registration behavior for specific situations. A closer look at the definitions is suitable for a better understanding. The SSD dissimilarity term is

$$S_{\text{SSD}}(\varphi) = \sum_{\mathbf{x} \in \Omega} \left(I_f(\mathbf{x}) - I_m(T(\mathbf{x}, \varphi)) \right)^2 , \tag{7.3.1}$$

which gives the following form for the image level force in x (for details refer to section 5.3):

$$\mathbf{f}_x^{\text{img}}(\varphi)[\mathbf{x}] = (I_m(T(\mathbf{x},\varphi))) - I_f(\mathbf{x})) \cdot \frac{\partial}{\partial x} I_m(T(\mathbf{x},\varphi)), \quad \forall \mathbf{x} \in \Omega. \tag{7.3.2}$$

Taking into account the y and z components that are defined accordingly, apparently the image level force for a given position $\mathbf{x}$ is characterized by the gradient of the warped moving image I_m and the intensity difference between the fixed and the moving image at that location. In consequence, forces are strongest at locations with a large gradient magnitude and a high intensity difference. However, the assumption that is implied by this fact is not generally valid for registration problems – bright objects in front of a dark background do not necessarily deform more severely than darker objects [33]. There is hardly ever such a link between imaged intensities and tissue deformation properties.

A simple modification can be performed in order to diminish the intensity bias of force computation. The sum of absolute differences (SAD) measure gives significantly better results, as compared to SSD, while still being mathematically sound as a similarity measure. The SAD based dissimilarity term is stated as

$$S_{\text{SAD}}(\varphi) = \sum_{\mathbf{x}\in\Omega} \left| I_f(\mathbf{x}) - I_m(T(\mathbf{x},\varphi)) \right|, \tag{7.3.3}$$

where typically an approximation of the standard norm $|\cdot|$ is used to ensure differentiability, such as $|x| := \sqrt{x^2 + \epsilon^2}$. The resulting slightly modified image level force then becomes:

$$\mathbf{f}_x^{\text{img}}(\varphi)[\mathbf{x}] = \text{sign}\left(I_m(T(\mathbf{x},\varphi))) - I_f(\mathbf{x})\right) \cdot \frac{\partial}{\partial x} I_m(T(\mathbf{x},\varphi)). \tag{7.3.4}$$

The force at a given location $\mathbf{x}$ is now only dependent on the magnitude of the gradient at that location. The intensity difference only determines the direction of the force vector by means of the signum function. Although more elaborate solutions to the intensity bias problem have been discussed in the literature (see e.g. [33]), simply exchanging the similarity measure to SAD is attractive and gives a reasonable improvement.

Figure 7.9 gives a comparison for force computation based on the SSD and SAD measures. The dissimilarity is in both cases evaluated and plotted as sum of squared differences for comparability. The dissimilarity curve does not drop as low for SSD as for SAD caused by the intensity bias phenomenon. On the other hand, the regularization energy increases more significantly for SAD and converges at a higher level. Obviously the more complex control point configuration that is introduced by the SAD-forces has a higher irregularity in the sense of diffusion regularization.

The sample difference images in the figure have been obtained after specific numbers of iterations that are indicated. Intensity-biased behavior is clearly visible for both SSD-based and SAD-based force computation. The innermost circle, which is black on white background in the reference and template images, is registered first in both cases while the larger circle, white on gray, remains almost undeformed. Once the regions of highest intensity difference have been registered, the other parts are accounted for. However, for the most intensity biased SSD-based forces the larger circles are not even registered after 500 iterations. In the case of SAD forces there is no change any more after approximately 300 iterations and the difference vanishes almost completely.

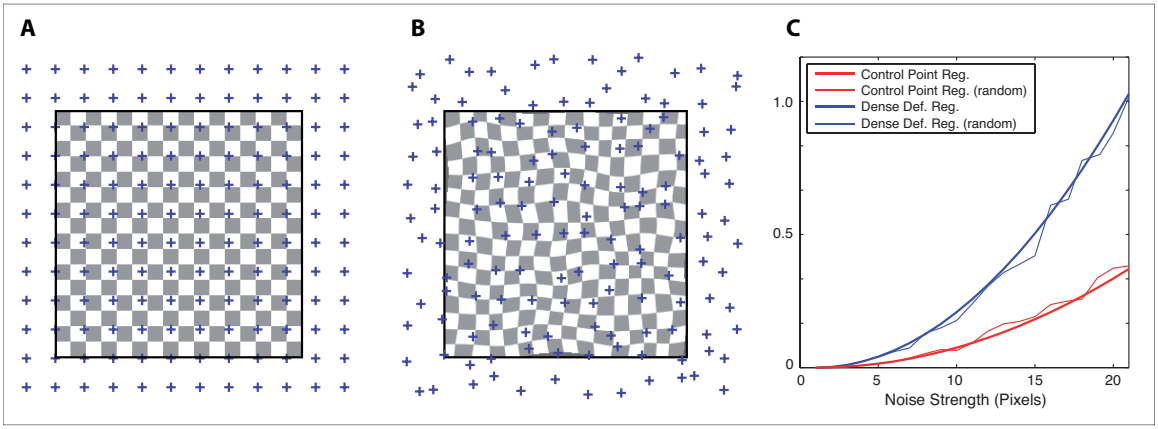


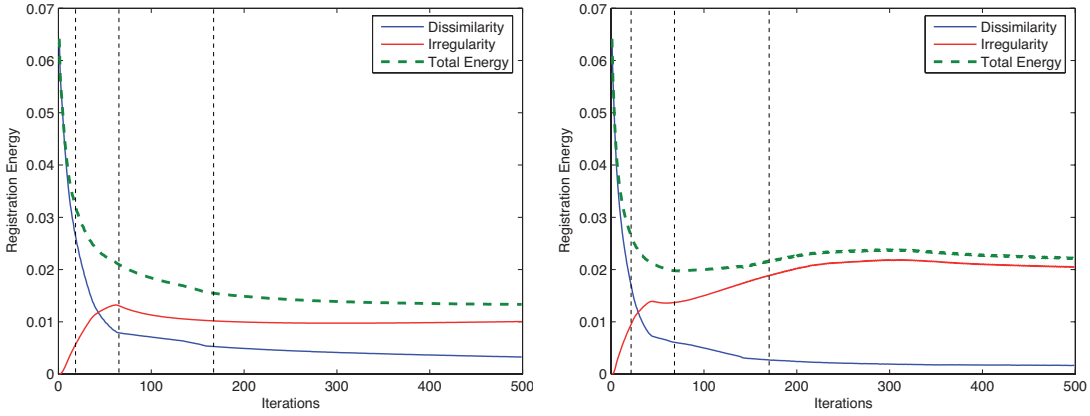
Figure 7.8.: Comparison of regularization on control points and on the dense deformation field. Initial control point configuration (A) and final situation with artificial random displacements at maximal amplitudes of 20 pixels (B). Relative development of diffusion regularization energies on control points and on the associated dense deformation fields for increasing random displacements (C). Without regard to scaling, both measures increase monotonically.

It is also noteworthy that the upper parts of both large circles seem to move more quickly to the registered position. The reason is mainly that the deformed large ovals are not perfecly centered with respect to the large circles in the reference image, leading to the assymetric registration behavior. Moreover, four small circles in the corners act as anchors, keeping the corners relatively rigid and counteracting fast movements of the large circles.

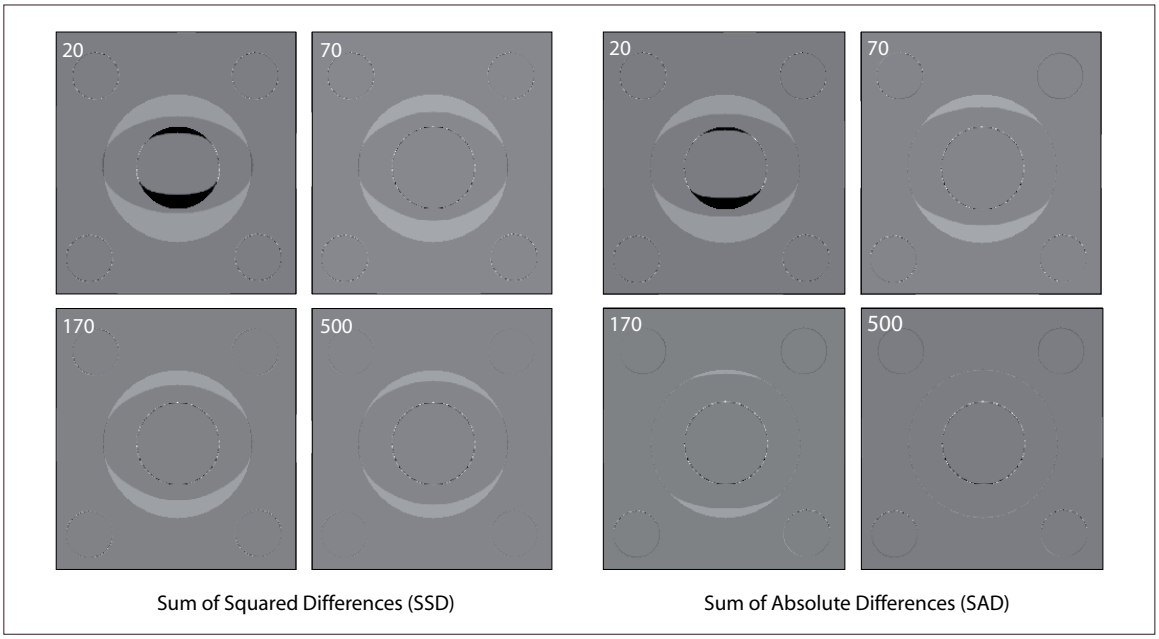
7.4. Control Point Regularization

The similarity of regularization on control points to a dense deformation field regularizer has been addressed in previous sections and is now illustrated experimentally. In order to compare the behavior of both regularization approaches, artificial control point configurations are evaluated that are increasingly „irregular“. Starting from an ideal control point grid with a uniform spacing of 20 pixels, random control point displacements are created with maximum amplitudes that increase from 0 to 20 pixels. For each configuration the dense deformation field is computed and the regularization energy is evaluated for the displacement field (Eq. 3.3.1) and for the control point displacements (Eq. 5.2.3).

Figure 7.8 shows the inital control point configuration and the situation for random noise with a maximum amplitude of 20 pixels. The graph depicts the development of both regularity measures for the case that one random seed is used with increasing amplitudes and the case that a new random displacement is used for each amplitude. The introduced irregularity is clearly captured by both regularization approaches. No claims are made that one measure is a bound for the other one, since scaling is ignored in this experiment. What can be concluded, however, is that both regularizers react in an analogous way to irregularities in the control points and in the resulting deformation field. Both measures increase monotonically for an increasing artificial irregularity.



(a) Registration energy during 500 iterations using forces based on SSD (left) and SAD (right).



(b) Difference images after indicated number of iterations.

Figure 7.9.: Illustration of intensity bias phenomenon for force computation based on SSD and SAD dissimilarity measures. The graphs show the progress of registration energy and its components, the difference images are obtained by interrupting registration at the positions indicated by vertical dashed lines in the graphs. Intensity-biased behavior is noticeable for both similarity measures as the innermost circle (black on white in the original images, compare Figure 3.1) is registered almost instantaneously after less than 70 iterations. Registering the larger circle (white on gray in the originals) takes disproportionally more iterations and is not even achieved after 500 iterations for SSD.

8. Medical Data

A data set of 11 CT scans of one patient is used where each scan is acquired at a different breathing stage. The scans all contain a thorax region approximately from the shoulders to the abdominal area. A rigid pre-registration that is typically necessary before applying a deformable registration algorithm is implicitly given since all scans were performed within close temporal intervals. The images are of size $256 \times 256 \times 142$ voxels. To utilize as much of the information contained in the 11 images as possible for evaluation, pairs of scans are mutually registered. Neglecting reciprocal registrations, such as A $\rightarrow B$ and $B \rightarrow A$, the data set can be divided into 55 pairs of images and as many independent registrations.

8.1. Visual Assessment

Judging the quality and especially the significance of a registration result on medical data is not a straightforward task. While the dissimilarity measure and the regularization energy can give hints on registration success, these numerical values generally have no physical or even medical justification. Moreover, as has already been pointed out, a low dissimilarity after registration cannot be taken alone for an indicator of successful registration. Visual assessment can give valuable insight into the effect of deformable registration. Especially medical experts can often more easily judge upon a registration outcome by means of their experienced eye than based on similarity measures and statistics. Apart from inspecting the warped template after registration for unplausible deformations, the change between the difference images before and after registration can be taken into account. A few examples of slices and difference images are shown in Figure 8.1. All major non-rigid deformations that are due to breathing are apparently completely compensated by the registration algorithm. The remaining structures in the difference images are in parts too small for the chosen grid spacing on the finest resolution level or are caused by interpolation artefacts.

8.2. Ground Truth Experiments

Since a more quantitative assessment of registration success is desirable, ground truth data is generated. A tumor on the left side of the patient's chest is manually segmented in all 11 available CT scans using a graph-cuts based algorithm, resulting in binary segmentation masks for each image. The dense deformation fields from the registration series that link each pair of images over different breathing stages are then applied to the appropriate segmentation masks. For instance, the deformation field obtained from registering image 1 to image 6 is applied to the segmentation mask corresponding to image 1. The resulting deformed segmentation is compared to the ground truth, the manual segmentation for image 6. In an ideal case these two segmentations overlap perfectly.

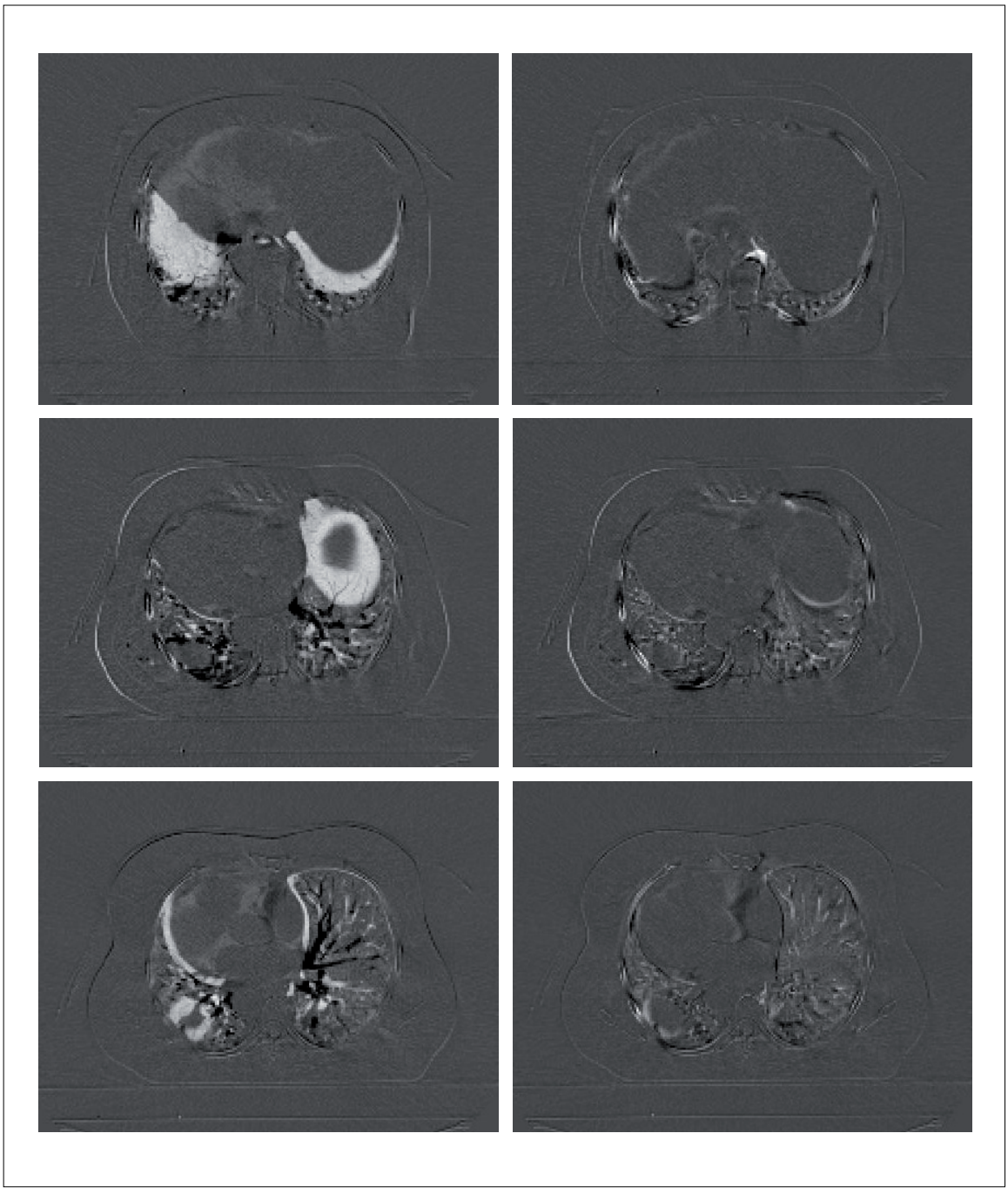


Figure 8.1.: Difference slice images before (left) and after (right) registering two of the CT scans across breathing stages. The white areas in the left images are due to vertical movement of organs (in the direction of z in the images) caused by breathing. These movements are completely compensated by registration.

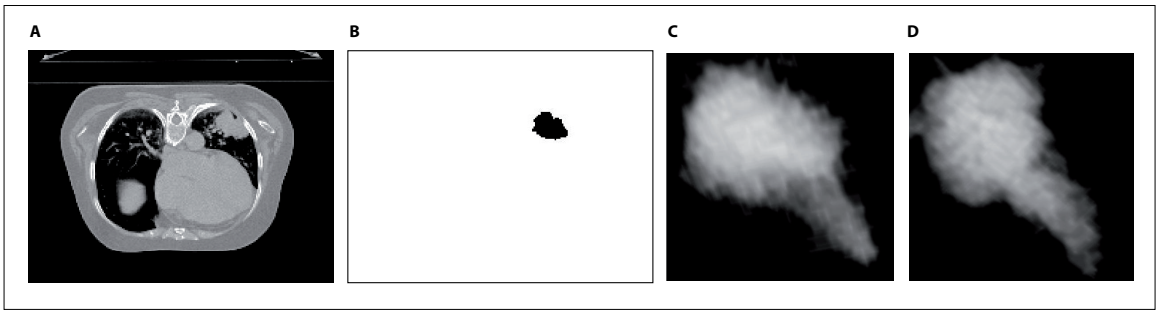


Figure 8.2.: Sample slice through one of the available CT scans (A), corresponding slice in manual ground truth segmentation (B). The binary segmentation mask is inverted. Volume renderings of two out of 11 manual tumor segmentations (C, D) illustrating the non-rigid deformation the tissue is exposed to by breathing.

8.2.1. Sensitivity and Specificity

To measure the degree of overlap that is achieved, the deformed and ground-truth binary segmentation masks are evaluated from a classification theory point of view. Interpreting voxels that are segmented as tumor in a segmentation mask as classified *positive* and the remaining voxels as *negative*, allows to apply the concepts of sensitivity and specificity. In considering pairs of voxels between a deformed and a ground truth segmentation mask, the true positive (tp), false positive (fp), true negative (tn) and false negative (fn) voxels can be counted. Following [10], sensitivity is then written informally as

$$\text{Sensitivity} = \frac{\text{tp}}{\text{tp} + \text{fn}} \tag{8.2.1}$$

and gives the fraction of voxels in the ground-truth tumor region that are correctly matched in the reconstructed segmentation. A perfect overlap results in a sensitivity value of 1. Specificity can be stated according to [10] as

$$\text{Specificity} = \frac{\text{tn}}{\text{tn} + \text{fp}} \tag{8.2.2}$$

and represents the fraction of correctly identified non-tumor voxels. Both measures are evaluated on a per-voxel basis and statistics over all 55 correspondences in the registration series are collected. Since on average the tumor regions cover only 0.04% of the whole volume, specificity is practically 1 in all experiments. The sensitivity mean is at 0.879 with a standard deviation of 0.045 and a median at 0.894. Minimal and maximal sensitivity values among the data set are 0.769 and 0.937, respectively. The average sensitivity is 10% lower for similar experiments performed without employing the deformations obtained by registration.

8.2.2. Processing Time

A typical registration run for the described 3D images using linear B-splines takes 70 to 150 seconds on the machine described above, depending on various parameter settings. Table

Level	Image size	Control Points	t/Iter. (linear)	t/Iter. (cubic)
2	$64 \times 64 \times 36$	900	$0.12s$	$0.29s$
1	$128 \times 128 \times 71$	3,840	$0.95s$	$2.43s$
0	$256 \times 256 \times 142$	22,707	$6.55s$	$19.69s$

Table 8.1.: Image dimensions for three resolution levels, control point count and computation durations per iteration for linear and cubic B-splines.

8.1 gives an overview of registration durations per iteration from the perspective of the multi-resolution setting. From level to level the duration increases by a factor of 8 since the number of control points is doubled in each spatial direction (neglecting control points outside the image boundaries). Computation based on cubic B-splines takes approximately 2.5 times longer than for linear B-splines, independent of control point spacing and image size. A similar observation has been described for the 2D situation. Other characteristics of the muli-resolution approach are evaluated in the following section.

8.3. Multi-resolution Setting

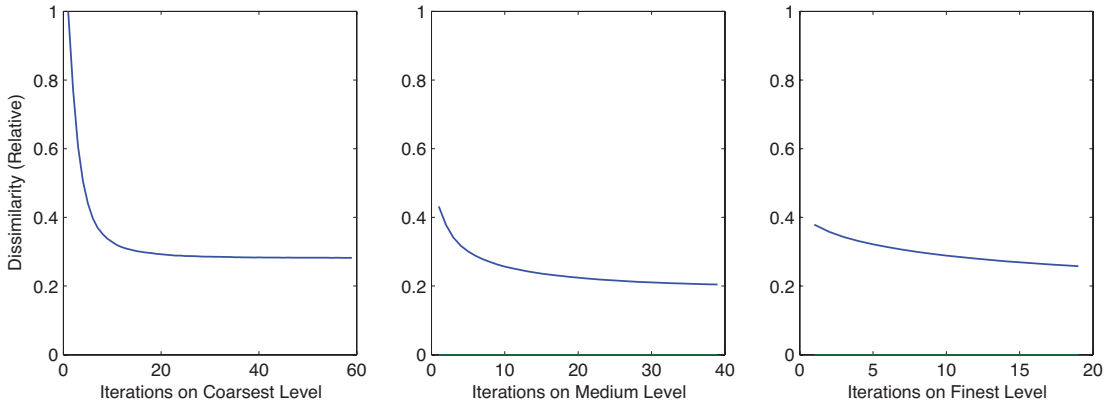
The multi-resolution approach described in section 5.5 is most effectively employed in the three-dimensional case since computation on full resolution 3D images is orders of magnitude more time consuming than in 2D. Several suitable decisions have to be taken in order to take full advantage of a multi-resolution registration strategy. First of all, a control point grid spacing has to be set. It seems most intuitive to use the same spacing in pixels for all levels so that in effect the number of control points is exactly doubled from one level to the next finer level. Spacings of 10 pixels per spatial dimension have proven to provide a reasonable compromise between performance and precision.

In addition, the registration parameters α and τ have to be determined for each level. In this case there is no reason to keep identical values for different resolutions. In fact, the registration behavior is typically so different on each level that these two parameters offer a valuable tool for control. As has been pointed out before, a rather strong regularization (a high value of α) is desirable on a coarse resolution in order to ensure a smooth global deformation. On the finest level regularization can be decreased significantly so that small structures can be deformed more flexibly. Similar reasoning applies to the step size parameter τ – whenever regularization is set to be strong, larger steps can be performed without negatively affecting convergence behavior. More caution (smaller values for τ) is necessary if regularization is decreased, since then the image forces have a more direct influence on deformations.

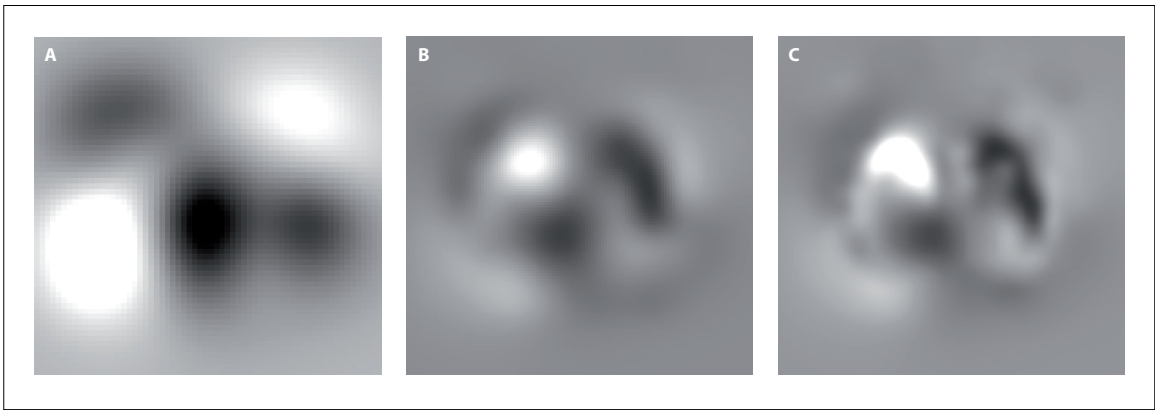
In practice the number of iterations to be performed is also manually adjusted for each resolution level. For the 3D images in the data set described above, three resolution levels are used. The resulting image sizes and processing durations can be seen in Table 8.1. Since in each resolution level the number of control points is doubled in each spatial dimension, one iteration takes 8 times longer from one level to the next finer level. The largest number of iterations is therefore allowed on the coarsest level and only a few iterations are performed on the full resolution images. Such a setting also complies with the assumption stated

previously that the global deformation between two images is generally recovered on the coarse resolution levels.

Figure 8.3 illustrates typical behavior observed for a registration between two scans from the thorax CT data set. The largest decline in dissimilarity is achieved on the coarsest level. The dissimilarity measure drops quickly and converges at a value around 30% of the initial dissimilarity on that level. The decrease in dissimilarity is less harsh for the intermediate and full resolutions, which can be explained by the fact that the most significant deformations are already accounted for on the coarsest level.



(a) Dissimilarity graphs for the 3 resolution levels. Original SSD values are scaled to fractions of the initial dissimilarity on each level. For the intermediate and fine levels the graphs do not start at 1, indicating the dissimilarity improvement achieved by transferring previous registration results from the coarser resolutions. The number of performed iterations is highest on the coarsest level.



(b) Sample slices through the x-components of the displacement fields obtained after registration on each of the 3 resolution levels. Intensities are contrast-enhanced for clarity in print. The displacement fields have the size of the images on the respective level, i.e. $32 \times 32 \times 36$, $64 \times 64 \times 71$ and $256 \times 256 \times 142$ voxels (left to right). While the displacement reconstructed on the coarsest level (A) only shows general, global deformations, the shape of the patient in the original scans is more clearly visible in the finer displacement fields (B, C). The added local detail achieved by a few iterations of registration on full resolution (C) is apparent.

Figure 8.3.: Illustration of typical multi-resolution registration behavior on 3D data.

Part V.

Summary and Conclusion

9. Conclusion

Having presented theoretical and practical aspects of the work accomplished for this thesis and having provided a thorough evaluation of achievable results, it seems worthwhile at this point to recapitulate the crucial points of the method. A few ideas for possible future work are also given, followed by closing remarks.

9.1. Summary

An algorithm for deformable registration has been described that combines aspects from two different and popular approaches. On the one hand, the conceptual framework of free-form deformations is employed which allows to model flexible deformations by controlling a limited number of points, instead of considering every image pixel individually. On the other hand, a numerical solution strategy is used that originates from the field of variational deformable registration based on dense deformation fields, which seemingly has no direct link to the free-form deformations setting.

This connection has been established for both entities that are involved in the optimization technique. In a sense, the procedure that is performed on a per-pixel basis for variational methods is transferred to the control points used in the context of free-form deformations. The analogy of the image force from the variational setting is the control point force, and the link between the two is given by the formulation of the dissimilarity term. It turns out that the control point force is a weighted average over the underlying image force. Computation can this way be achieved by means of a spatial smoothing filter.

A connection between the dense deformation and free-form deformation approaches can also be devised for the second crucial entity involved, the regularization term. Regularization that is introduced as a remedy to the ill-posedness of the deformable registration problem can be performed using the penalty schemes often employed in the variational realm, while being solely applied to the control points. Depending on the order of B-splines that are used, a dense deformation regularization based on diffusion can be seen as a weighted discrete derivative approximation of the control points. It has been demonstrated that the regularization energy evaluated on the control point displacements behaves analogously to a full regularization on dense deformation fields. This way, when using e.g. a diffusion regularizer for the control points, an energy is minimized in the optimization process that inherently also minimizes the dense deformation field diffusion regularization energy.

The proposed method has been implemented for the two-dimensional and the three-dimensional setting while incorporating methods to increase computational efficiency. Experiments on synthetic data were provided to demonstrate general properties of the registration approach and to evaluate its general effectiveness and performance. The applicability of the algorithm to typical real-life registration problems has been illustrated using medical patient data.

9.2. Future Work

The work described in this thesis gives plenty of opportunities for continued research and improvement. To name just a few, the following issues can be addressed:

- **Similarity measures**. While the reference implementation described in this thesis is based on the sum of squared differences (SSD) and the sum of absolute differences (SAD) as similarity measures, other metrics can be integrated. Given that the aforementioned measures are both most suitable for the mono-modal registration situation, interesting new options for similarity metrics can be found in the domain of statistical similarity measures. For instance, the mutual information metric (MI) mentioned in an introductory section is often used for multi-modality registration, since it judges image similarity without relying directly on image intensities.

- **Regularization terms**. Given the link that has been created between the domains of free-form deformable and variational registration, regularization terms that have been devised for variational registration can be incorporated into the algorithm described in this thesis. The diffusion regularizer has been used for the present implementation, but the performance of other choices such as curvature or linear elasticity is an interesting question.

- **Optimization strategies**. The optimization strategies implemented are both popular modifications of the standard fix-point iteration approach used to solve a quasi-linearized version of the partial differential equation obtained by the energy functional comprising dissimilarity and regularization terms. Other solution strategies that are applied for variational deformable registration can be tested for their suitability in the presented framework.

- **Visualization**. Since the current implementation was created for pure feasibility evaluation reasons, visual presentation was neglected. Having demonstrated the effectiveness of the proposed algorithm, more attention can be devoted to presentation and visualization. Although text-based output along with resulting image data can provide important information on algorithm behavior, a pictorial feedback during registration progress can help to gain deeper insight and is certainly worth the implementation effort.

- **Graphical user interface**. Obviously the current way of running the registration algorithm is not suitable for use by non-computer experts. Parameter files have to be created according to a specific scheme and the command prompt is used for execution. A graphical user interface (GUI) can be implemented to facilitate the use of the program for future applications. Given the modular structure of the program, a user interface can be added with relative ease using popular systems for user interface programming (e.g. FLTK, Qt).

- **General efficiency improvements**. Although the performance that is achievable with the current implementation is competitive for both 2D and 3D data, there is always a potential for efficiency gains. An aspect that could be investigated, for instance, is gradient computation only for the fixed image. This way the spatial gradient

is not computed in each iteration for the warped moving image. This approximation is a heuristic but reasonable approach that can increase computation speed. Furthermore, code libraries optimized for performance can be exploited for certain image processing or memory management tasks which are currently implemented in a rather straightforward fashion.

- **Further practical evaluation**. Finally additional practical tests can be performed, based on specific prerequisites and necessities of particular medical fields of application. Since different types of medical images have diverse characteristics, shortcomings of a registration algorithm can only be revealed by further extensive studies.

9.3. After Registration

Medical image registration has grown over the past 20 years from a rather minor and very specific field of imaging applications into a subdiscipline in itself that is increasingly approached as a stand-alone field of research at conferences and workshops and in the literature [12]. While the need for image registration clearly has a clinical origin, the vast majority of related publications focus on theoretical research on registration methodology – at a risk of neglecting questions regarding the applicability in clinical practice. As viewed by Maintz et al. in [19], two important questions arise at the point where research on medical image registration concludes in many cases:

„*How valid is the registration?*“ and „*How to use the registration?*“

The former question has obviously been addressed in this thesis by means of several experimental studies and most publications on registration algorithms typically provide validation information. Yet at the same time, the interest in comprehensive validation frameworks that combine various criteria such as precision, stability, robustness and time performance has only started to grow recently [19]. The lack of proper and thorough validation is considered by Maintz et al. to be a barrier that prevents many registration approaches from being meaningfully applied in the clinical setting.

The second question of how to use a computed registration is also believed to be often underestimated [12, 19]. An answer to this question demands a clear definition of a clinical need, as well as an interdisciplinary approach that links registration methods with segmentation and visualization techniques. Publications on registration methods often point the way to potential fields of application, and practically oriented methods for visualization assume their input to be registered images. However, combining methods from the two sides into one approach of high clinical relevance is believed to be a field of research that, according to Maintz et al., deserves more attention on its own.

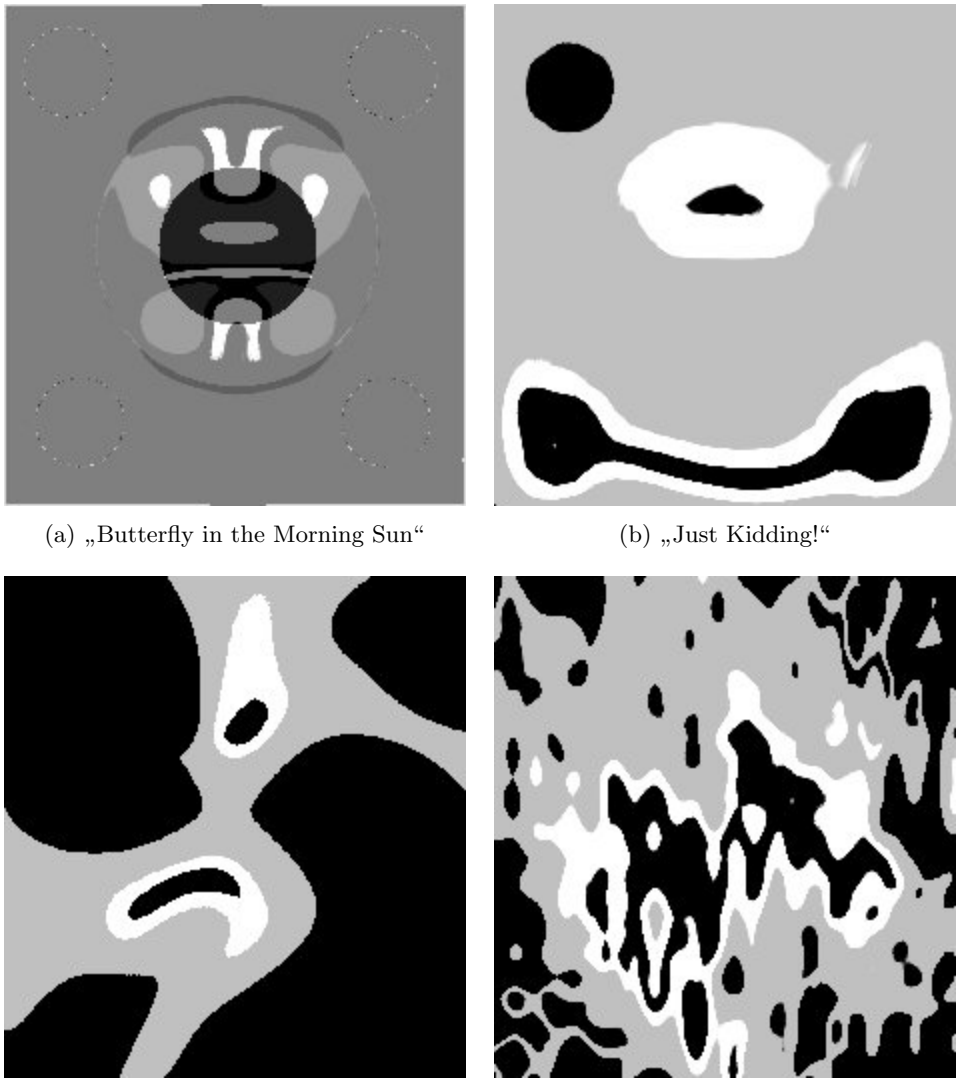
And then again, no matter how advanced computer aided medical procedures will become in the close future, there will always remain one constant: the need for experts from all involved domains, such as physicians, computer scientists and physicists. For how tempted one might be to attribute „intelligence“ to computers, their contribution will always be repetitive in character and require human capabilities for any advance whatsoever.

Part VI.

Appendix

A. Art

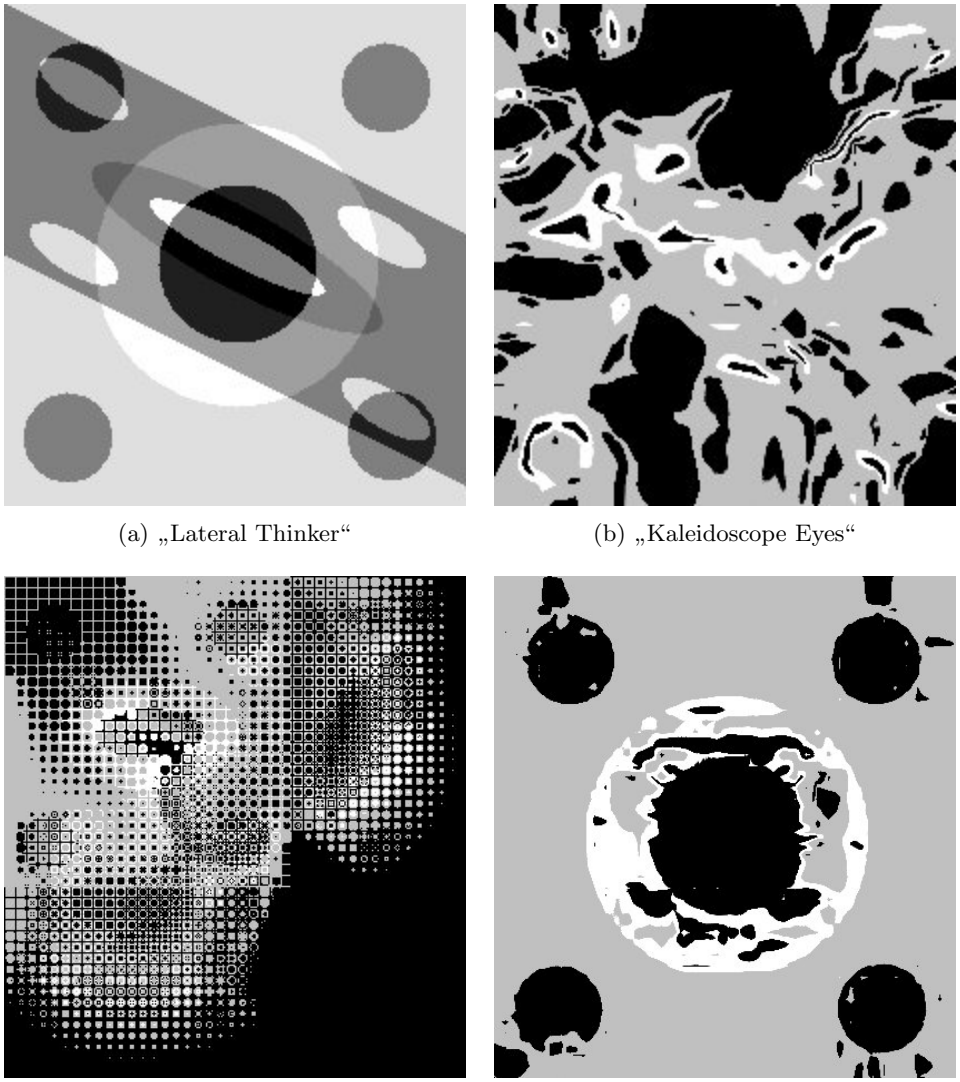
Those who dare to doubt that computers have an artistic side shall be branded fools by the following humble collection of exhibits, obtained from the images shown in Figure 3.1.



(a) „Butterfly in the Morning Sun“ (b) „Just Kidding!“

(c) „Miss Registration“ (d) „Melting Pot“

Figure A.1.: A creative little programming mistake, interspersed with slightly unorthodox parameters, and suddenly a registration algorithm breaks out of its cage of pudicity to unfold its beautiful potential.



(a) „Lateral Thinker“ (b) „Kaleidoscope Eyes“

(c) „Hommage à Lichtenstein“ (d) „Good Vibrations“

Figure A.2.: Additional artificial yet artsy illustrations. Exhibits are of size 231×251 pixels, created using the technique of opaque intensity on background.

List of Figures

Bibliography

[1] Ruzena Bajcsy and Stane Kovacic. Muliresolution elastic matching. *Computer Vision, Graphics and Image Processing*, 46:1–21, 1989.

[2] Thomas Beyer, David W. Townsend, Tony Brun, and Paul E. Kinahan. A combined PET/CT scanner for clinical oncology. *Journal of Nuclear Medicine*, 41:1369–1379, 2000.

[3] W. R. Crum, T. Hartkens, and D. L. Hill. Non-rigid image registration: theory and practice. *The British Journal of Radiology*, 77:140–153, 2004.

[4] E. D'Agostino, J. Modersitzki, F. Maes, and Vandermeulen D. Free-form registration using mutual information and curvature regularization. *Workshop on Biomedical Image Registration*, LNCS 2717:11–20, 2003.

[5] R Deriche. Fast algorithms for low-level vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:78–87, 1990.

[6] Martin Dugas and Karin Schmidt. *Medizinische Informatik und Bioinformatik*. Springer, 2003.

[7] Bernd Fischer and Jan Modersitzki. Curvature based image registration. *Journal of Mathematical Imaging and Vision*, 18:81–85, 2003.

[8] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. *Computer Graphics*. Addison Wesley, 1997.

[9] David R. Forsey and Richard H. Bartels. Hierarchical b-spline refinement. *ACM Computer Graphics*, 22/4:205–212, 1988.

[10] Alan S. Go, J. Ben Davoren, Michael G. Shlipak, Stephen W. Bent, Leslee L. Subak, and Terrie Mendelson. *Evidence-based Medicine – A Framework for Clinical Practice*. McGraw-Hill, 1998.

[11] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Prentice Hall, 2002.

[12] Joseph V. Hajnal, Derek L. G. Hill, and David J. Hawkes. *Medical Image Registration*. CRC Press, 2001.

[13] Gerardo Hermosillo Valadez. *Variational Methods for Multimodal Image Matching*. PhD thesis, Universite de Nice – Sophia Antipolis, 2002.

[14] Berthold K. P. Horn and Brian G. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

[15] Bernd Jähne. *Digitale Bildverarbeitung*. Springer, 2005.

[16] Jan Kybic and Michael Unser. Fast parametric elastic image registration. *IEEE Transactions on Image Processing*, 12/11:1427–1442, 2003.

[17] Seungyong Lee, George Wolberg, Kyung-Yong Chwa, and Sung Yong Shin. Image metamorphosis with scattered feature constraints. *IEEE Transactions on Visualization and Computer Graphics*, 2/4:337–354, 1996.

[18] Tom Lyche and Knut Morken. Spline methods draft. http://heim.ifi.uio.no/~tom/, 2006.

[19] J. B. Antoine Maintz and Max A. Viergever. A survey of medical image registration. *Medical Image Analysis*, 2/1:1–36, 1998.

[20] B. McCane, K. Novins, D. Crannitch, and B. Galvin. On benchmarking optical flow. *Computer Vision and Image Understanding*, 84:126–143, 2001.

[21] Jan Modersitzki. *Numerical Methods for Image Registration*. Oxford University Press, 2004.

[22] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C – The Art of Scientific Computing*. Cambridge University Press, 1992.

[23] Torsten Rohlfing and Calvin R. Maurer. Intensity-based non-rigid registration using adaptive multilevel free-form deformation with an incompressibility constraint. *Proceedings of MICCAI 2001*, 2208:111–119, 2001.

[24] Torsten Rohlfing and Calvin R. Maurer. Nonrigid image registration in shared-memory multiprocessor environments with application to brains, breasts and bees. *IEEE Transactions on Information Technology in Biomedicine*, 7/1:16–25, 2003.

[25] Daniel Rueckert, L. I. Sonoda, and C. Hayes. Nonrigid registration using free-form deformations: Application to breast MR images. *IEEE Transactions on Medical Imaging*, 18/8:712–721, 1999.

[26] Thomas W. Sederberg and Scott R. Parry. Free-form deformation of solid geometric models. *ACM Siggraph*, 20/4:151–160, 1986.

[27] Richard Szeleski and James Coughlan. Spline-based image registration. *DEC Cambridge Research Labs Technical Report Series*, 94/1, 1994.

[28] Emanuele Trucco and Alessandro Verri. *Introductory Techniques for 3-D Computer Vision*. Prentice Hall, 1998.

[29] Nicholas J. Tustison, Brian B. Avants, Tessa A. Sundaram, Jeffrey T. Duda, and James C. Gee. A generalization of free-form deformation image registration within the ITK finite element framework. *Proc. International Workshop on Biomedical Imaging*, 4057:238–246, 2006.

[30] Lucas J. van Vliet, Ian T. Young, and Piet W. Verbeek. Recursive gaussian derivative filters. *Proceedings International Conference on Pattern Recognition*, 1:509–514, 1998.

[31] William M. Wells, Paul Viola, Hideki Atsumi, Shin Nakajima, and Ron Kikinis. Multi-modal volume registration by maximization of mutual information. *Medical Image Analysis*, 1:35–51, 1996.

[32] Ian T. Young and Lucas J. van Vliet. Recursive implementation of the gaussian filter. *Signal Processing*, 44:139–151, 1995.

[33] Darko Zikic, Ali Khamene, and Nassir Navab. Intensity-unbiased force computation for variational motion estimation. In Adrien Bartoli, Nassir Navab, and Vincent Lepetit, editors, *DEFORM'06 Workshop on Image Registration in Deformable Environments*, pages 61–70, 2006.