



Graz University of Technology
Institute of Computer Graphics und Vision

Master Thesis

NON-LINEAR 3D-VOLUME REGISTRATION
USING A LOCAL AFFINE MODEL

Stefan Kluckner

Graz, Austria, October 2006

Thesis supervisors

Prof. Dipl.-Ing. Dr. techn. Horst Bischof

Dipl.-Ing. Martin Urschler

Kurzfassung

Die Masterarbeit beschäftigt sich mit der Implementierung einer nicht-linearen Registrierungsmethode für medizinische CT Volumendaten unter der Verwendung eines lokal affinen Modells.

Unter dem Begriff Registrierung versteht man die Bestimmung einer geometrischen Transformation, die ein Objekt in einem Bild auf dasselbe oder ein ähnliches Objekt in einem anderen Bild ausrichtet. Die Dimension des Bildes und die Art der Transformation legen die Anzahl der zu optimierenden Parameter fest. Eine nicht-lineare Registrierung von Bildern ist ein hochdimensionales Optimierungsproblem.

Im Rahmen dieser Arbeit wird eine nicht-lineare Registrierung von 3D CT Daten von der Lunge zu unterschiedlichen Atmungszuständen implementiert. Viele Ansätze erlauben keine genaue Registrierung von kleinen Gefäßen, wie sie etwa in der Lunge vorhanden sind. Ein lokal affines Registrierungsmodell berechnet in jeder Nachbarschaft eines Bildpunktes die zugehörigen Transformationsparameter. Als Gesamtes wird eine globale Deformation mit lokalen Verschiebungen berechnet, welche eine genaue Registrierung von kleinen Gefäßen erlaubt. Zusätzlich ermöglicht das vorgeschlagene Modell eine Kompensation von Intensitätswert-Variationen. Die vorgestellte Methode wird anhand von synthetischen und realen Datensätzen evaluiert.

Stichworte. nicht-lineare Registrierung, mono-modal, optischer Fluss, lokal affines Modell, Intensitätsvariationen, medizinische Bildanalyse

Abstract

This master thesis deals with the implementation of a non-linear registration method for medical CT volume data by means of a local affine model.

The term registration describes a determination of a geometrical transformation, which aligns an object in an image with the same or a similar object in another image. The dimension of the images and the type of the transformation specify the number of parameters to be optimised. A non-linear registration of images is a high-dimensional optimisation problem. In the context of this thesis a non-linear registration algorithm of three-dimensional CT data is implemented to register the lungs at different respiratory states. Several approaches do not obtain an exact registration of small structures, such as are for instance contained within the lung. A local affine registration model computes the associated transformation parameters in each neighbourhood of a voxel. A global deformation with local displacements is computed, which makes an exact registration of small structures possible. Additionally, the suggested model can compensate intensity variations. The presented methods are evaluated on synthetic and real data sets.

Keywords. non-linear registration, mono-modal, optical flow, locally affine model, intensity variations, medical computer vision

Contents

1	Introduction	1
1.1	Motivation	1
1.2	Problem Description and Aim of the Work	3
1.3	Structure of the Master Thesis	3
2	Medical Image Registration	5
2.1	Introduction	5
2.2	Medical Imaging	5
2.3	Image Registration	9
2.3.1	Types of Image Registration	10
2.3.2	Modalities	10
2.3.3	Components of Image Registration	11
2.4	Related Work	19
2.4.1	Spline-based Registration	20
2.4.2	Optical Flow Based Registration	21
2.4.3	Elastic Registration	22
2.4.4	Fluid- and Diffusion-Based Registration	23
2.4.5	Biomechanical Registration	23
3	Elastic Registration in the Presence of Intensity Variations	25
3.1	Introduction	25
3.2	Optical Flow	25
3.3	Local Affine Model	28
3.4	Intensity Variations	30
3.5	Smoothness	31
3.6	Choosing the Model of Transformation	32
3.7	Computation of the Derivatives	33
3.8	Computing the Inverse	35
3.9	Implementation Details	37
3.10	Summary	40
4	Improvements	43
4.1	Introduction	43
4.2	Improving the Inverse Computations	43

4.2.1	Computing the Inverse with LU Decomposition	43
4.2.2	Estimating the Condition Number	44
4.2.3	Computing the Inverse with Cholesky Decomposition	45
4.3	Reducing the number of Inverse Computations	46
4.3.1	A Sub-Sampling Scheme	46
4.3.2	Interpolation of Missing Parameters	47
4.3.3	Decomposing the Affine Matrix	48
4.3.4	Composing the Resulting Affine Matrix	50
4.4	Replacing the Iterative Smoothing	51
5	Experiments and Results	53
5.1	Introduction	53
5.2	Data	53
5.3	Evaluation Methods	55
5.3.1	Intensity Differences	55
5.3.2	Displacement Vector Differences	57
5.3.3	Normalised Mutual Information	57
5.4	Experiments on Improvements	57
5.4.1	SVD vs. LU and Cholesky decomposition	58
5.4.2	Experiments on Sub-Sampling	60
5.4.3	Experiments on Replacing the Iterative Smoothing	62
5.5	Experiments on Number of Iterations	64
5.6	The Comparison	69
5.6.1	Results on Synthetic Data	70
5.6.2	Results on Real Data	72
5.6.3	Results on Real Data including Intensity Variations	74
6	Summary and Discussion	85
6.1	Conclusion	85
6.2	Future Work	86
A	Acronyms and Symbols	87
B	Evaluation Framework	89
B.1	Implementation	89
B.2	Configuration Files	90
B.3	Result Files	91
B.4	Extension	92
C	Outline of the B-Spline Deformable Registration Algorithm	93
	Bibliography	95

List of Figures

1.1	Registration of three dimensional data sets at different respiratory states. . .	2
1.2	Location and structure of the lung.	2
2.1	Thorax images.	6
2.2	Sample slices of real CT-scan of sheep lungs.	7
2.3	Sample slices of real CT-scans of the humans lungs	8
2.4	Various MRT Images of the authors inside.	8
2.5	Sonogram and fMRI images.	9
2.6	Components of a registration process.	11
2.7	Examples of different types of transformations.	13
2.8	The principle of a hierarchical multi-scale approach.	18
2.9	Integer and sub-pixel raster.	18
2.10	Interpolation methods	19
3.1	1D Gaussian distribution.	35
3.2	Different resolutions of an image at inhalation and exhalation state.	38
3.3	Convolution with and without padding.	39
4.1	Sub-sampling the image in 3 dimensions.	46
4.2	Interpolation scheme.	47
4.3	The principle of the smoothing scheme.	51
5.1	Examples of input images.	55
5.2	Comparison of difference fusion images.	56
5.3	Synthetic Displacement fields	57
5.4	Evaluation of the time consumption using LU/Cholesky decomposition and SVD	59
5.5	Black box evaluation of the time consumption, RMSE and NMI using LU/C- holesky decomposition and SVD.	60
5.6	Axial sample slices of difference fusion images before and after registration using <i>Model 1</i>	60
5.7	Axial sample slices of difference fusion images before and after registration using <i>Model 4</i>	61
5.8	Diagram of the time consumption of no, decomposed and direct interpolation.	62
5.9	Diagram of the computed RMSE and NMI similarity measurements of no, de- composed and direct interpolation.	63

5.10	Sagittal sample slices of difference fusion images before and after registration.	63
5.11	Time consumption of iterative and Gaussian smoothing.	65
5.12	RMSE similarity measurements of iterative and Gaussian smoothing with varying σ	65
5.13	Sample slices of resulting displacement fields for <i>Model 2</i> using various σ . . .	66
5.14	Axial sample slices of resulting difference images.	66
5.15	Computation times and similarity measurements on experiments for varying number of outer and smoothing loops.	68
5.16	Convergence behaviour of the sum of displacements in each iteration and level.	69
5.17	Evaluation results on $128 \times 128 \times 128$ <i>S10</i> data set.	71
5.18	Visual results on $128 \times 128 \times 128$ <i>S10</i> data set.	72
5.19	Evaluation results on $128 \times 128 \times 128$ <i>S25</i> data set.	74
5.20	Visual results on $128 \times 128 \times 128$ <i>S25</i> data set.	75
5.21	Time consumption on real data sets.	76
5.22	RMSE and NMI comparison using a real $256 \times 256 \times 256$ data set.	76
5.23	Moving, fixed, warped images and the displacement field in different views . .	77
5.24	Computed displacement field in vector representation.	78
5.25	Sagittal sample slices of difference fusion images before and after registration using various algorithms.	78
5.26	Visual image comparisons before and after registration of data set <i>Intensity5</i> .	79
5.27	Graphical evaluation results of several methods performed on data set <i>Intensity5</i> .	81
5.28	Visual image comparisons before and after registration of data set <i>Intensity2</i> .	82
5.29	Time consumption and NMI measurements on data set <i>Intensity2</i>	83
5.30	An illustration of the displacement fields for data set <i>Intensity2</i> and <i>Intensity5</i> .	83
B.1	The organisation of the parameters configuration file.	90
B.2	The organisation of the processing list configuration file.	91

List of Tables

2.1	Substance densities.	6
3.1	First-order derivative kernels.	34
5.1	Evaluation of the time consumption using LU/Cholesky decomposition and SVD.	58
5.2	Black box evaluation of the time consumption, RMSE and NMI using LU/Cholesky decomposition and SVD.	59
5.3	Evaluation of the time consumption of no, decomposed and direct interpolation.	61
5.4	RMSE and NMI similarity measurements of no, decomposed and direct interpolation.	62
5.5	Evaluation of the time consumption of iterative and Gaussian smoothing.	64
5.6	RMSE similarity measurements of iterative and Gaussian smoothing.	64
5.7	RMSE similarity measurements of iterative and Gaussian smoothing with varying σ	67
5.8	Evaluation of time consumption depending on the number of smoothing iterations.	67
5.9	Computation times and similarity measurements on experiments for varying number of outer loops.	68
5.10	Convergence behaviour of the sum of displacements in each iteration and level.	69
5.11	Methods and their parameters of the comparison.	70
5.12	A comparison of different registration methods by time consumption and similarity measurements on data set <i>S10</i>	70
5.13	A comparison of different registration methods by time consumption and similarity measurements on data set <i>S25</i>	73
5.14	A comparison of different registration methods by time consumption and similarity measurements on data set <i>Real</i>	73
5.15	A comparison of different registration methods by time consumption and similarity measurements on data set <i>Real</i>	80
5.16	Evaluation results of various methods performed on data set <i>Intensity5</i>	80
5.17	Evaluation results of various methods performed on data set <i>Intensity2</i>	81

Chapter 1

Introduction

1.1 Motivation

The medical and healthcare sector is a rapidly growing sector since humans are thinking about their quality of life. Modern technologies in medicine, such as imaging based modalities make it possible to visualise functional and anatomical information of the human body. Because of that, radiologists and physicians are reliably supported in their diagnosis of diseases. Medical imaging techniques also provide important data for research, science and education.

Thanks to the improvement of the information technology, medical images are available in digital form and they can be archived in large databases. Medical imaging systems produce a large amount of raw data. Special image processing steps are necessary to alleviate specified diagnosis and treatment of diseases from different image modalities such as CT, MRI, US, X-ray, etc..

Depending on the modality the image processing tasks differ significantly. An important image processing task is the extraction of relevant information from raw data. Another challenge is to bring images, generated by the same or multiple, modalities into spatial alignment, that means that each point in an image has a known corresponding point in a second image. These images can be taken at different times, views or modalities. In literature this is referred to as image registration. Monitoring tissue motion or growth is one of the main applications of image registration. Fusing information of multiple images, comparing anatomical features to an atlas or supporting image guided surgery are other uses.

In this work we concentrate on the registration of images differing by respiratory motion. Figure 1.1 shows a sample registration case for three dimensional data at two different respiratory states. The process of registration aligns the moving image at a time t and the fixed image at a time $t - 1$. This alignment represents respiratory motion, e.g. from inhalation to exhalation. As a result, the transformation between the two images is computed. In our case we wish to model the respiratory motion of the lungs. The diaphragm controls primarily the respiration in the human body. During exhalation the diaphragm relaxes and when the diaphragm contracts fresh air can stream into the lungs. To a smaller extent, the ribcage muscles also contribute to respiration. Figure 1.2(a) shows a scheme containing the location¹ of the lungs and the diaphragm. The pulmonary alveoli in the lungs are responsible for

¹Image is taken from <http://www.delta-education.com>

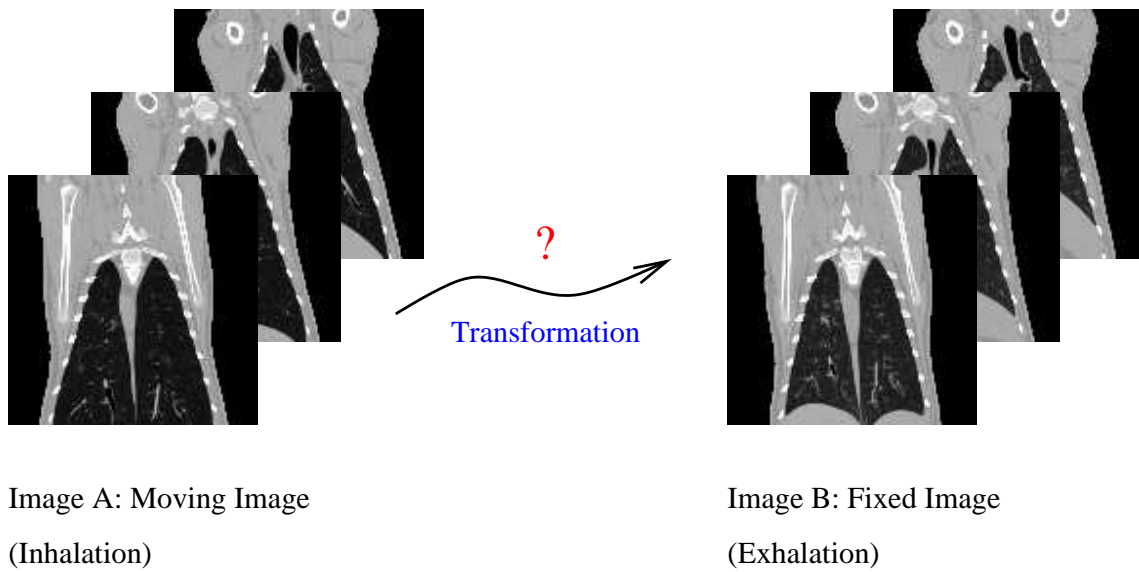


Figure 1.1: Registration of three dimensional data sets at different respiratory states.

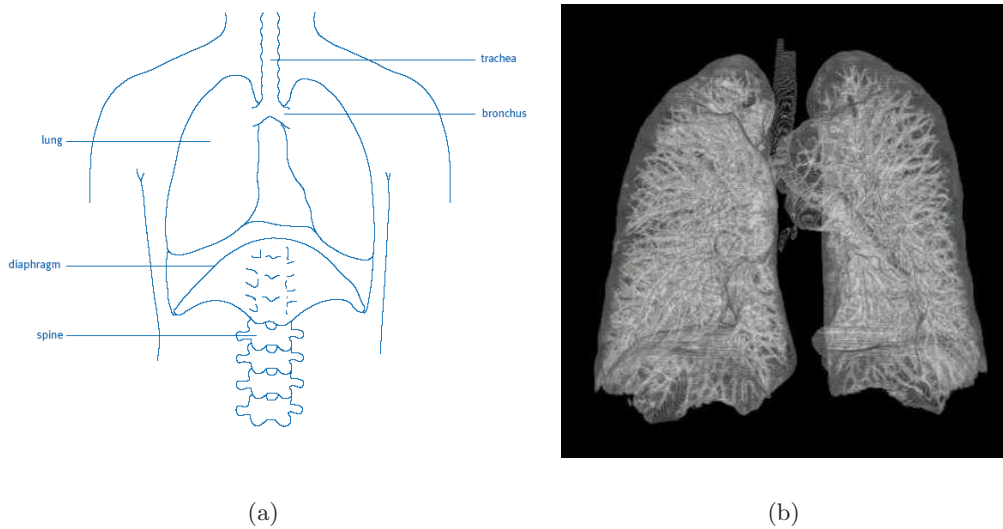


Figure 1.2: (a) Location of the lung and diaphragm, (b) reconstructed structure within the lung from a CT image.

the gas exchange with the blood. Due to the small size of the alveoli, the dendritic blood vessels and the complex deformation, which is limited to the ribcage and controlled by the diaphragm, respiratory motion cannot be modelled by a simple linear transformation. In Figure 1.2(b) a reconstruction² of the small structures within the lungs from CT images is shown.

In order to perform registration in the presence of respiratory motion differences, we need a non-linear registration approach, which handles large motion caused by the diaphragm and

²Image is taken from <http://en.wikipedia.org/wiki/Lung>

which can model small deformations to align capillary structures such as alveoli.

1.2 Problem Description and Aim of the Work

Non-linear registration of two images requires an optimisation procedure of the transformation parameters in a high-dimensional search space. This search space represents the according degrees of freedom (DoF). In the majority of cases, a high-dimensional optimisation procedure turns out to be a difficult problem.

At the present time a general purpose, efficient and simple to set-up non-linear registration algorithm does not exist. Due to this fact, the topic of non-linear registration is a very active area of research. The growing interest in the development of new algorithms provided a large amount of publications. This master thesis, therefore, reviews important non-linear registration methods to get an overview.

Periaswamy et al. [36] proposed a promising registration method, which deals with a compensation of intensity variations and gets by with few parameters to set-up. The main goal of this work is to implement this method based on a hierarchical optical flow method.

The first task is to extend an available *Matlab* implementation³ of this approach to three dimensions. Medical CT images with a dimension of $128 \times 128 \times 128$ serve as input data.

The *Matlab* prototype will point out the issues of a further *C++* implementation. The Insight Segmentation and Registration Toolkit [32] makes it possible to apply already implemented image processing units to the *C++* code in a modular fashion.

The obtained results of the *C++* implementation should be compared to existing non-linear registration algorithms. Another task is to improve the runtime if necessary.

An implementation of a highly automated framework will support the comparison of the results of the algorithms. This evaluation framework generates statistical measurements on the resulting warped images and displacement fields. A description and implementation details regarding the evaluation framework are included in Appendix B. In this master thesis, the algorithms are evaluated on several kinds of data, using intra-modality CT-scans of sheep and human lungs.

1.3 Structure of the Master Thesis

The master thesis is separated into 6 chapters. This introducing chapter gives a motivation of performing registration on medical image data sets and it describes the aims of the thesis. Chapter 2 illustrates roughly how to generate medical images. Common used imaging techniques are explained. The main components of a general registration process are also given in this chapter. The related work section reviews important registration approaches and includes a literature survey for non-linear registration methods.

In Chapter 3 Periaswamy's registration approach using a local affine model is outlined. It includes details about the local affine model, intensity variations and the applied smoothness constraint. A description on how to compute the spatial and temporal derivatives and inverse computations completes the theoretical part of this chapter. A short summary and

³The code can be found on <http://www.cs.dartmouth.edu/farid/research/registration.html>

information on implementation details round off the description of the suggested approach. Some improvements of Periaswamy's method are described in Chapter 4. Enhancements for an efficient computation of the inverse of matrices and further computation time saving methods are shown. The details of decomposing a general affine matrix in rotation, scaling, shearing and translation components are explained in a detailed way.

Chapter 5 describes the accomplished experiments and the obtained results. Quantitative and qualitative results are shown for synthetic and real CT datasets and for images including intensity variations.

Chapter 6 finally discusses the results and experiments of this master thesis, summarises the contributions and gives an outlook on further work.

Chapter 2

Medical Image Registration

2.1 Introduction

In the recent years the use of radiological images in healthcare and medicine has shown a strong increase. The widespread development of various medical imaging techniques makes it possible to generate images of human internals, with high resolution and accurate information about the function and growth of organs. Several imaging modalities support physicians in diagnosis and treatment. Besides, they provide important information for science and research.

Section 2.2 gives a short overview of the most widespread medical imaging methods which provide input data for image processing.

In Section 2.3 different types of registration methods are specified. Furthermore an illustration of necessary components for the medical image registration process is given.

Section 2.4 goes into the details of several approaches of non-linear registration, accompanied by a literature survey, where fundamental registration techniques are presented.

2.2 Medical Imaging

Medical imaging makes it possible to examine diseases e.g. of human patients and supports diagnosis. Imaging is used to explore the anatomy so that physiological and biochemical processes can be studied. New, improved imaging techniques allow to visualise processes in organs such as in the brain or the lungs. The following non-invasive methods produce information in form of images of living organisms.

Radiography

Radiography is a widespread technique with a high diagnostic yield, which uses X-rays and a photographic film to create medical images [19]. The internal structure of a subject can be displayed by means of the effect that different organs absorb the energy of the rays differently. The attenuated X-rays expose a film or digital receptors. Figure 2.1(b) shows a radiograph of the thorax. In the medical field the most common usage for radiography is the detection of fractures of bones, mammography and dental medicine. An important advantage of radiography is its ease-of-use and its relatively low cost.

Computed Tomography

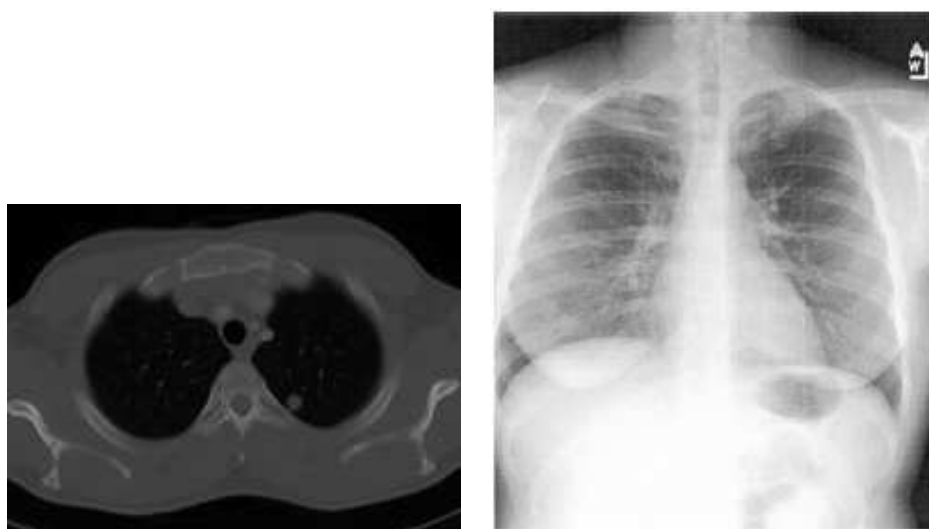
Computed Tomography (CT) is a specialised X-ray imaging technique [19]. By rotating an X-ray source along a single axis, the X-ray detectors distributed around the patient collect raw data from multiple angles. In this way two-dimensional (2D) X-ray images can be generated by solving an inverse problem. Figure 2.1(a) illustrates a reconstructed slice of the human body. A three-dimensional (3D) image can be created by taking a series of images traversing along this axis. The CT technique avoids the major drawback of radiography, where anatomical structures shadow themselves.

The newest CT scanner generation, multi-slice helical scanners, offers a fast imaging technique to display organs with or without injected contrast agents. Modern scanners already include principle methods of image processing.

In CT scans, each pixel is assigned to a numerical value, which characterises the relative density of a substance. This number is compared to the attenuation value of water and displayed on a scale of arbitrary units named Hounsfield units (HU) after Sir Godfrey Hounsfield. Table 2.1 (examples are taken from [19]), highlights the substance densities in HU.

Air	-1000
Fat	-50
Water	0
Soft Tissue	+40
Calculus	+100 ... +400
Bone	+1000

Table 2.1: Substance densities in HU



(a) An axial slice of a thorax CT-scan

(b) A thorax radiograph

Figure 2.1: Thorax images, taken from [19].

In this work we evaluate the implemented registration algorithm on 3D-CT scans. These CT images display human and sheep lungs. The scans of humans consist of a native and a contrast enhanced image, shown in Figure 2.3, which make it possible to evaluate the results on intensity variations.

The native images showing the sheep's lung are taken at two respiratory states. Experiments on synthetic data are based on the sheep data. The details of the synthetic data generation are explained in Chapter 5.

Figure 2.2 illustrates sample slices of real data of the sheep's lung at the two respiratory states in axial, sagittal and coronal view.

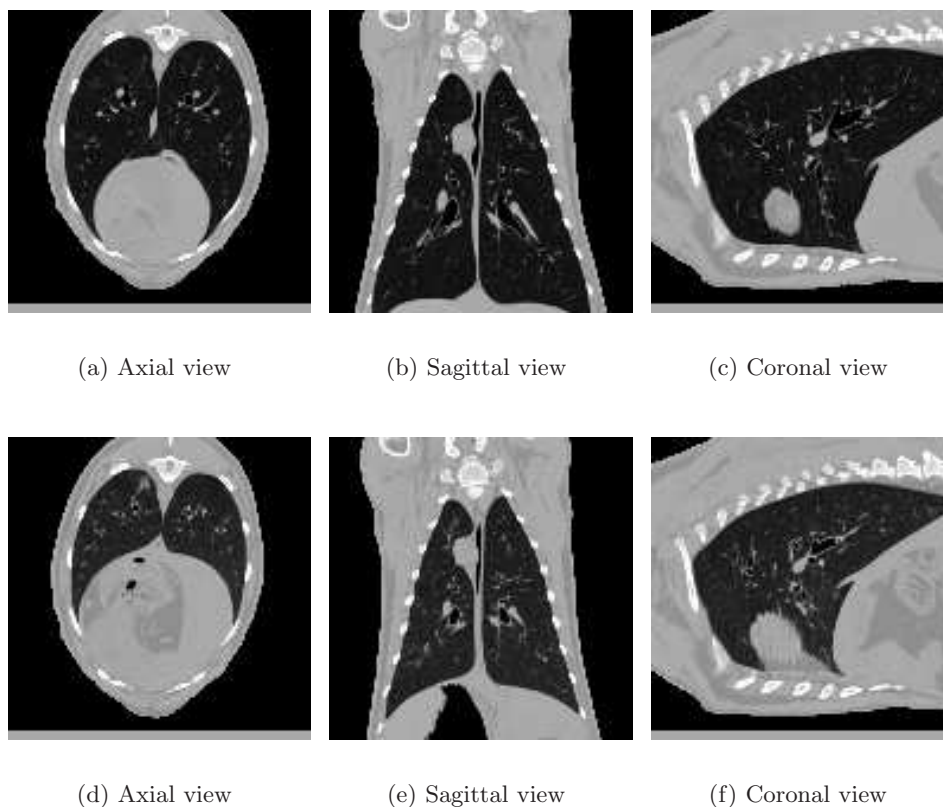


Figure 2.2: Sample slices of real CT-scan of sheep lungs at different views and respiratory states: (a)-(c) inhalation and (d)-(f) exhalation.

Magnetic Resonance Imaging

Instead of X-rays magnetic resonance imaging (MRI) [19], also referred to as nuclear magnetic resonance (NMR) or magnetic resonance tomography (MRT), uses a strong magnetic field to generate medical images. The magnetic field causes the protons to align in one direction. Another high frequency electro-magnetic field deranges the alignment. This alternating process produces a resonance signal which is used to generate a 2D or 3D image.

The images visualise pathological or physiological alterations of tissues. MRI is a very expensive modality. However, an important advantage is, that the MRI operates on electro-magnetic

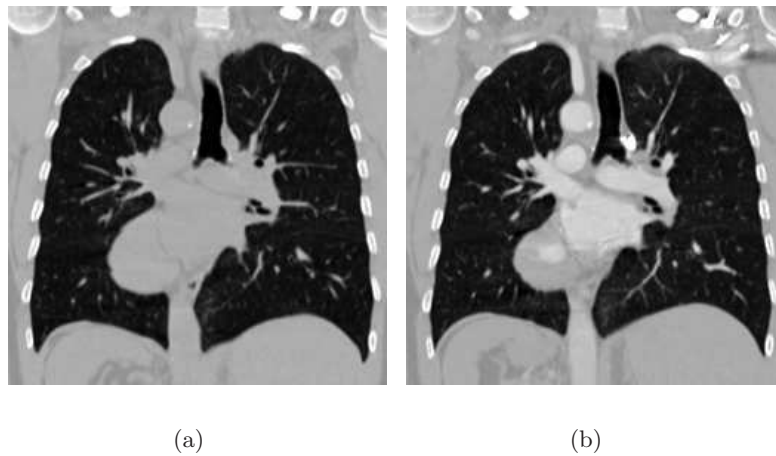
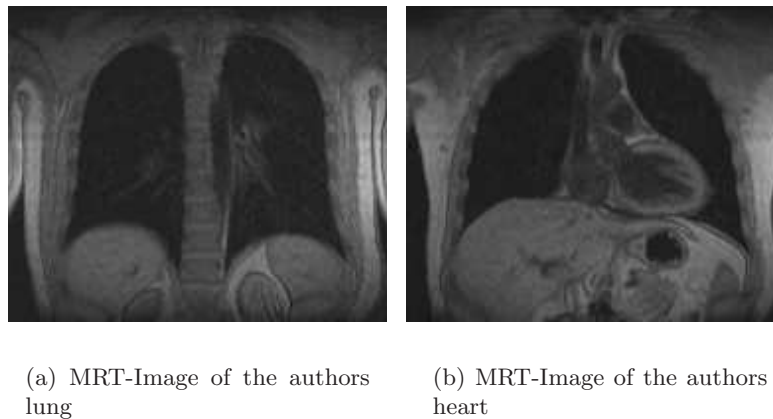


Figure 2.3: Sample slices of real CT-scans of the humans lungs in sagittal view: (a) A native scan and (b) a scan including contrast agents.

waves, which are harmless for the human body as opposed to X-ray used in radiography or CT. Only for humans with cardiac pacemaker or a metal joint it is obviously a dangerous modality due to the high magnetic field.

Figure 2.4 illustrates two MRI images of the upper part of the author's body.



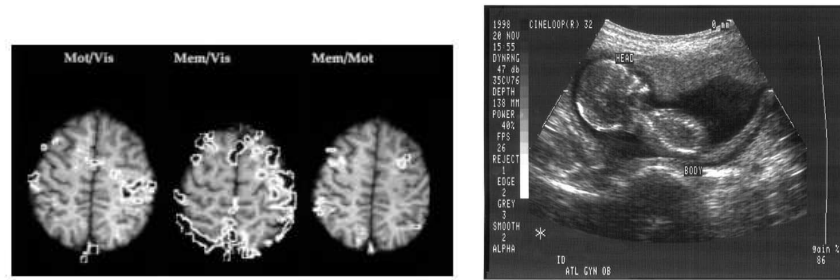
(a) MRT-Image of the authors lung

(b) MRT-Image of the authors heart

Figure 2.4: Various MRT Images of the authors inside.

Functional Magnet Resonance Tomography

Functional MRI (fMRI) measures signal changes in the brain. Increased neural activity causes an increased demand for oxygen. The proportion of oxygenated and de-oxygenated hemoglobin in the brain vessels makes it possible to generate high resolution 3D images by means of the blood oxygen level-dependent (BOLD) effect. Details of this technique are given in [19]. Figure 2.5(a) shows a series of fMRI images taken during a study of learning ability and its relation to short-term memory activity.



(a) Different fMRI images during a study, taken from [19]

(b) Sonogram of a baby in its mother's womb

Figure 2.5: (a) fMRI images and (b) a sonogram.

Sonography

Sonography [19] is an ultra sound (US) based diagnostic imaging technique used to visualise muscles and internal organs. It examines their size, structure and any pathological lesions. Sonography is real-time suitable, completely harmless to the patient and therefore commonly used during pregnancy. A US signal is emitted and the reflected signals are measured and used to reconstruct an image. Recent techniques provide 3D volume information or image blood flows using the Doppler effect. A sonogram¹ of a baby in its mother's womb is shown in Figure 2.5(b).

Positron Emission Tomography

Positron emission tomography (PET) [19] is a nuclear medical imaging technique that produces a 3D image or map of functional processes in the human body.

Single Photon Emission Computed Tomography

The radioactive gamma rays are responsible for the image generation in Single Photon Emission CT (SPECT) [19]. This diagnostic technique takes images of living organisms. It demands an injection of a small dose rate of radionuclides before an image acquisition and can visualise the function of organs. SPECT provides 2D or 3D images by using gamma cameras.

2.3 Image Registration

As mentioned in the introduction of Chapter 1, registration is the process of determining the correspondence between all points in two or more images. In our case we use two images, where image A denotes the fixed image and image B is the moving image, respectively.

Depending on the various imaging techniques and their characteristics, different registration approaches are obviously needed. The most general application of image registration is the alignment of medical images. Aligning medical images of the brain or the bones allows to

¹taken from <http://de.wikipedia.org/wiki/Sonografie>

model e.g. growth over time. Periodic deformations of tissue, resulting from cardiac or respiratory cycles, can be modelled as well. Changes in the position of the subject, while taking images of the same modality at particular times or with different modalities, can be determined with registration methods. For all types of medical image registration the accuracy of the results is an important factor.

The following sections give an overview of general terms and components used in medical image registration. Details can be found in [29], [55] and [9].

2.3.1 Types of Image Registration

There exist several types of image registration methods. An image-to-image alignment such as intra-subject and inter-subject registration, determines a one-to-one mapping between the images in a way that identical anatomical points are mapped together. Intra-subject registration is used to match medical images taken at different times to monitor the growth of e.g. the bones or the brain and changes in tissues. In general, intra-subject registration allows to align multi-modal images to correct spatial distortions from one subject.

Inter-subject registration captures anatomical shapes, provides correspondences for statistical shape models or makes it possible to align an image with a detailed anatomical atlas. The removal of individual anatomical features from volume data is a challenge in inter-subject registration.

A determination of an one-to-one mapping between e.g. images stored on a computer and the patient in a real operating room is called image-to-physical-space registration, where identical anatomical features are mapped together. Correspondences between images and physical landmarks are identified with 3D tracking techniques. A combination of image-to-image and image-to-physical-space registration methods is also possible.

2.3.2 Modalities

When images are acquired by different medical imaging techniques, the registration process is called multi-modal as opposed to mono-modal, where the images have the same modality. Examples of mono-modality are e.g. CT-CT, MRI-MRI, . . .

If the images modalities of the registration are e.g. MRI-CT, MRI-PET, SPECT-CT the task is multi-modal. The complex relation between the different unknown intensity distributions of the input images forms a problem in multi-modal registration. In that case, the similarity measurements are often based on statistical approaches or information theoretical techniques. A dynamic modality denotes a recording of MRI or CT perfusion series. Cine modality describes temporal MRI sequences as an example. The image dimension has a large influence on the computation time of the registration process. Common dimensions of modalities are 2D, 3D or 4D, which use a 3D modality with a decoupled temporal component.

In general, an alignment is performed for images of equal dimension, but there also exist approaches to align 2D to 3D and 3D to 4D images or vice versa.

In this work we concentrate on the mono-modal registration of 3D-CT to 3D-CT data.

2.3.3 Components of Image Registration

A registration process consists of several principle parts. The following sections highlight commonly used transformations, interpolation methods, registration bases and optimisations strategies. Figure 2.6 illustrates the interaction of these components. The re-sampling component represents the warping of the moving image with the estimated transformation parameters.

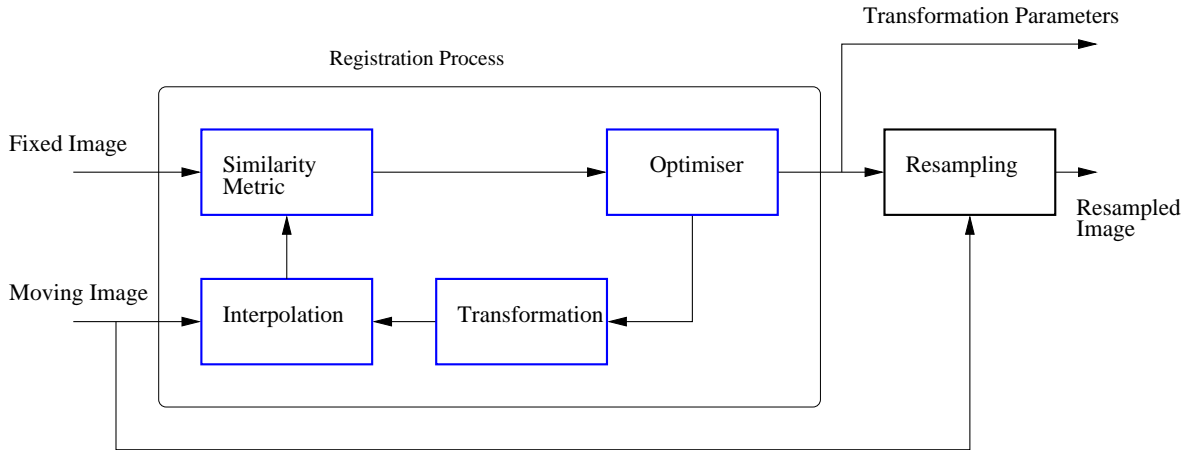


Figure 2.6: Components of a registration process.

Transformation

A transformation T is defined to transform points from one image space to another. The transformation implies the nature of expected motion that has to be modelled. The transformation models global, local motion or growing. Here, global motion denotes a transformation that is applied to the entire image domain, while local means that for determined sub-sections of the image space the applied transformations are varying. The following points give a review of the natures of transformations.

- *Rigid Body Transformation*

A rigid body transformation compensates global patient repositioning. It is used for simple tasks such as brain and bone registration. Six DoF describe a rotation R and a translation $t = (t_x, t_y, t_z)$. Three parameters define the rotation, which can be represented by e.g. Quaternions, Euler angles or a general rotation matrix. Equation 2.1 shows a transformation of the point $p(x, y, z)$ to $p(x', y', z')$. Rigid body transformation preserves distances, straightness of lines, planarity of surfaces and non-zero angles between straight lines.

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = R \begin{pmatrix} x \\ y \\ z \end{pmatrix} + t \quad (2.1)$$

- *Affine Transformation*

The affine transformation preserves lines and parallelism and includes rotation, translation, shearing and scaling. In homogeneous coordinates the affine transformation can be expressed as a composition of rigid, scaling and shearing parameters. In Equation 2.2 the parameters m_1 to m_9 represent the affine and m_{10} , m_{11} and m_{12} the translation parameters. The affine transformation has an overall number of 12 DoF.

$$T_{affine} = T_{shear} \cdot T_{scale} \cdot T_{rotation} \cdot T_{translation} = \begin{pmatrix} m_1 & m_2 & m_3 & m_{10} \\ m_4 & m_5 & m_6 & m_{11} \\ m_7 & m_8 & m_9 & m_{12} \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (2.2)$$

- *Projective and Perspective Transformation*

The projective transformation extends the affine transformation matrix with additional parameters p_1, p_2 and p_3 . This transformation preserves straightness of lines, but no parallelism. The homogeneous transformation matrix is given in Equation 2.3.

$$T_{projective} = T_{shear} \cdot T_{scale} \cdot T_{rotation} \cdot T_{translation} = \begin{pmatrix} m_1 & m_2 & m_3 & m_{10} \\ m_4 & m_5 & m_6 & m_{11} \\ m_7 & m_8 & m_9 & m_{12} \\ p_1 & p_2 & p_3 & \alpha \end{pmatrix} \quad (2.3)$$

The perspective transform consists of a subset of projective transformations and describes the image formation for many modalities like photography, X-ray projection, microscopy or endoscopy.

- *Non-linear or Curved Transformation*

An accurate modelling of complex tissue deformations requires a high number of DoF. Non-linear transformation compensates additional local transformations and does not preserve straightness of lines nor parallelism. The literature survey in Section 2.4 highlights the most common registration approaches using non-linear transformations based on polynomials, splines and physics-derived models.

Figure 2.7 demonstrates the results of applying different transformations to a 2D square.

Registration Basis

The registration methods can be classified by their basis. Extrinsic methods are based on objects (fiducial markers) attached to the subject, while intrinsic methods use only image information of the subject. Non-image based registration, as a third registration basis, uses a calibrated coordinate system of the involved imaging systems and suffers under the assumption that the subject remains motionless between the imaging phases.

In the extrinsic case the objects are well recognisable in all modalities so that a fast registration without the need for optimisation methods is possible. Invasive marker objects, such as screws, allow a highly accurate registration result, but there is a need of placing the

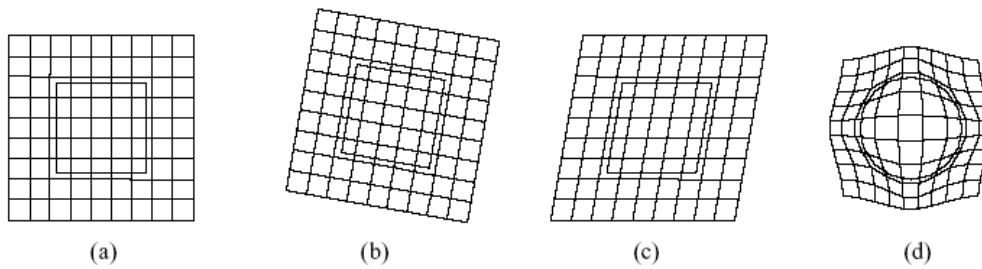


Figure 2.7: Examples of different types of transformations in 2D case: (a) identity transformation, (b) rigid transformation, (c) affine transformation and (d) non-linear transformation, taken from [18].

markers inside the body before image acquisition. In contrast, non-invasive skin mounted markers can also reach a fast alignment of images without an invasion.

An intrinsic method uses an alignment based on the image content. The intrinsic approaches can be divided into landmark-, segmentation- and voxel-property based registration methods. Landmarks are extracted from geometrical or anatomical features. Features can be found in a segmentation process by user interaction or automatic identification. In contrast to landmark based registration, segmentation based methods use high-order segmented structures such as curves, surfaces or volumes.

Voxel-property based methods work directly on the image intensity values. These methods are also called intensity-based. Methods working on the complete image content are of great interest, because of their flexibility and capability to automate the registration process.

In opposition to voxel-property based measurement, which allows non-linear transformations, the extrinsic-, landmark- and segmentation based methods are often limited to rigid or affine transformations.

In literature a various number of suggested approaches exists. In [28] a survey of proposed approaches based on the mentioned registration bases is given.

Similarity Metrics

To compare intensity differences, Fourier domain computations, cross-correlation calculations or methods based on information theoretical background are common important similarity measurements. In this thesis, only the voxel-based similarity measurements are outlined. Depending on the type of modalities, which are used, different approaches are adopted to the registration process. In case of mono-modality the voxel information is compared directly between the two images, while a multi-modal registration process has to cope with the involved relationship of intensity values between these images.

The next sections describe the most common similarity measurements used in image registration such as intensity differences, correlation techniques, ratio image uniformity and information theoretic approaches.

- *Intensity Difference*

Commonly used, simple approaches measuring the similarity are the sum of squared difference (**SSD**) and the sum of absolute difference (**SAD**). In an overlapping domain Ω in image A and image B, the sum of squared or absolute differences is computed and normalised by the number of voxels. Equations 2.4 and 2.5 show the formulas to compute the **SSD** and the **SAD**.

$$SSD = \frac{1}{N} \sum_{x \in \Omega} (A(x) - T(B(x)))^2 \quad (2.4)$$

$$SAD = \frac{1}{N} \sum_{x \in \Omega} |A(x) - T(B(x))| \quad (2.5)$$

The sum of squared differences is very sensitive to a small domain Ω with large intensity differences between the two images A and B. This sensitivity must be considered for images with contrast agents. This measurements assume that after alignment, the images differ only by Gaussian distributed noise. If the difference between the images is Gaussian noise, the **SSD** gives the optimal measurement. By using the sum of absolute differences the effect of outlier differences can be reduced.

- *Correlation Coefficient*

If there exists a linear relationship between the voxel intensity values, the correlation coefficient (**CC**) gives the optimal measurement for similarity:

$$CC = \frac{\sum_{x \in \Omega} (A(x) - \bar{A}) (T(B(x)) - \bar{B})}{\sqrt{\sum_{x \in \Omega} (A(x) - \bar{A})^2 \sum_{x \in \Omega} (T(B(x)) - \bar{B})^2}} \quad (2.6)$$

where \bar{A} and \bar{B} represent the mean intensity values computed over the overlapping domain Ω in the images A and B, respectively. The normalized cross correlation (**nCC**) can be computed using Equation 2.7. These similarity measurements are applied to inter-modality registration [28]. The normalized correlation value always ranges between -1 and $+1$. If the two sub-images in the spatial neighbourhood Ω are identical representing maximal alignment, the value of **nCC** is $+1$.

$$nCC = \frac{\sum_{x \in \Omega} A(x)T(B(x))}{\sqrt{\sum_{x \in \Omega} A(x)^2 \sum_{x \in \Omega} T(B(x))^2}} \quad (2.7)$$

- *Ratio Image Uniformity*

The ratio image uniformity (**RIU**) approach, suggested by Woods [53] derives a ratio image computed from image A and B. This idea is also referred to as the method of variance of intensity ratio. The transformation parameters are iteratively determined by maximising the uniformity of the derived ratio image, which is quantified as the normalised standard deviation of the voxel values in the ratio image. By using Equation 2.8 the ratio image can be computed, \bar{R} is determined by Equation 2.9 and 2.10

calculates the resulting RIU of the images A and B for one iteration.

$$R(x) = \frac{A(x)}{T(B(x))} \quad \forall x \in \Omega \quad (2.8)$$

$$\bar{R} = \frac{1}{N} \sum_{x \in \Omega} R(x) \quad (2.9)$$

$$RIU = \frac{\sqrt{\frac{1}{N} \sum_{x \in \Omega} (R(x) - \bar{R})^2}}{\bar{R}} \quad (2.10)$$

The RIU approach is applied to inter-modality registration processes such as MRI-PET image alignment.

- *Information Theoretical Approaches*

These approaches are based on the the Shannon-Wiener Entropy, which has been suggested in the 1940s for the first time. It was used for information measurements in communication theory. In the one-dimensional (1D) case Equation 2.11 can be used to compute the entropy H , which gives a number of average information content, covered by the set of i symbols, whose probabilities are given by p_i .

$$H = \sum_i p_i \log p_i \quad (2.11)$$

In our case, we have two images to be aligned. The joint entropy measures the amount of shared information in the images A and B. By maximising the amount of this shared information between image A and B over a space of suitable transformations, the images can be aligned. At each voxel position there are two symbols for an estimate of the transformation. If the two images are unrelated, the joint entropy results from the sum of entropies of the images A and B like $H(A, B) \leq H(A) + H(B)$. In [44], the joint entropy is derived from a normalised 2D histogram, which describes the probability density function (PDF) of the intensity values of image A against the values of image B. The joint entropy is given in Equation 2.12, where $p(a, b)$ describes the value at location (a, b) in the histogram.

$$H(A, B) = \sum_a \sum_b p(a, b) \log p(a, b) \quad (2.12)$$

The accuracy of quantification of the PDF is given by the bin number. This number tiles the intensity value space into bins. Studholme [44] and Collignon [10] proposed joint entropy similarity measurements in multi-modal image registration, where the joint entropy gets iteratively minimized in the registration process to align the two images. The mutual information (MI) was simultaneously proposed by Viola and Wells [50] and Collignon and Maes [27], where the overlap problem is dealt with. Details on the overlap

problem can be found in the referred papers. The MI is defined as:

$$I(A, B) = H(A) + H(B) - H(A, B) = \sum_a \sum_b p(a, b) \log \frac{p(a, b)}{p_a(a)p_b(a)} \quad (2.13)$$

where $p_a(a)$ and $p_b(b)$ represent the marginal probability distributions, respectively. The MI is maximized by optimal alignment, if the joint entropy $H(A, B)$ is minimized and the marginal entropies $H(A)$ and $H(B)$ are maximized. MI improves the overlap problem, but does not solve it. Studholme et al. [45] proposed a normalised mutual information (NMI) to overcome the sensitivity of mutual information to change in image overlap (Equation 2.14). The NMI is more robust for inter-modality registration in which the overlap volume changes substantially.

$$I(A, B) = \frac{H(A) + H(B)}{H(A, B)} \quad (2.14)$$

By computing the similarity measures directly from intensity values of the voxels, the transformation parameters can be estimated. This may be performed either in a direct or an iterative fashion, where the latter way leads to the necessity for using elaborate optimisation algorithms.

Optimisation

The optimisation process tries to find the optimal transformation parameters, which maximises the alignment with respect to the similarity measure of the images A and B. This similarity measure is defined by a function, which depends on the two images. The dimension of the function is given by the DoF of the transformation. As mentioned in [28] the transformation parameters can either be computed directly i.e. determined in an explicit fashion from available data or searched for i.e. determined by finding an optimum of some objective function defined on the parameter space.

In non-linear registration the parameter search space is almost always very high dimensional. The following sections give an overview of optimisation strategies to find the optimal transformation parameters, which register two images in a best way, such as closed form solutions, iterative search methods and hierarchical approaches.

- *Closed Form Solution*

Referring to [18], only two suggested algorithms use closed form solutions to determine the transformation parameters. The first algorithm is the *Procrustes* method, which is based on point correspondences. The extracted corresponding set of 3D points of two images, so called fiducial markers, allows to compute a rigid body or affine transformation that aligns the sets of points. In practise the singular value decomposition (SVD) [38] is used to compute the transformation matrix. This matrix is applied to any point in the moving image.

A second method proposed by Friston et al. [17] uses statistical parametric maps (SPM) and assumes that the image A and B have nearly same intensity value ranges and the

transformation to establish the correspondences is small. This method minimises the sum of squares between the two images following non-linear spatial deformations and transformations of the voxel intensity values. The spatial and intensity transformations are obtained by using the least squares method. Direct methods only provide a solution for simple rigid cases.

- *Iterative Search*

Voxel-based registration algorithms require an iterative strategy to find optimal transformation parameters. An initial estimate of transformation parameters is refined by trial and error. Through the iteration process and by computing the similarity between the images A and B in each iteration, the transformation parameters get estimated and refined. As long as the similarity measurement improves, i.e. the optimisation process has not yet converged or a stopping condition such as e.g. a maximum number of iterations or maximum optimisation time has been met, the process searches for improved transformation parameters. To find optimal parameters an objective function has to be defined.

The main goal of optimisation is to find an optimum in a fast and accurate way, consuming as few memory as possible. However, in practice, these three tasks can not all be fulfilled simultaneously, so a trade-off between the three mentioned criteria has to be found.

In literature there exists a large number of different multi-dimensional optimisation approaches, which makes it possible to search for the best aligning parameters. Plum's survey [37] outlines various optimisation strategies in voxel-based registration. Frequently used deterministic optimisation techniques [38] range from Powell's method to approaches that are time and memory efficient, such as the Broyden-Fletcher-Goldfarb-Shanno (BFGS) optimiser [24]. Fundamental gradient descent and quasi-Newton methods are based on the computation of derivatives of the objective function, whereas Powell's method goes without this. Powell's method estimates a single parameter in each turn. In contrast the downhill-simplex method optimises all parameters simultaneously.

Stochastic techniques like Genetic Algorithms (GA) and Simulated Annealing (SA) are only used in combination with deterministic methods, which tend to get stuck in local extrema. The stochastic approaches avoid local extrema and support the optimisation process in finding a more global solution of the parameters, however they are computationally much more expensive. In [38] an outline of the implementation of these techniques is given.

- *Hierarchical Multi-resolution Approach*

As mentioned above, the parameter space is very high dimensional in non-linear registration. Besides getting stuck in a local extrema, optimisation strategies sometimes converge to an undesired optimum. To overcome this problem, a multi-resolution approach is frequently used in practice, where the images are down-sampled to coarser resolutions. Figure 2.8 shows the principle of a multi-resolution approach.

In [37], several references are given, that take use of a multi-resolution approach.

Gaussian pyramids are widespread in its application to build levels of different resolutions. For each level of resolution from coarse up to the finest level, the registration process is repeated. The estimated transformation parameters in a coarse level provide the input for the next level. The hierarchical multi-resolution approach speeds up the optimisation process by fast computation of initial solutions at coarse levels. The estimated parameters support the search direction in finer resolution levels. A coarse-to-fine strategy makes it possible to model large motions, if the resulting parameters of the coarse level are up-sampled to the finer level.

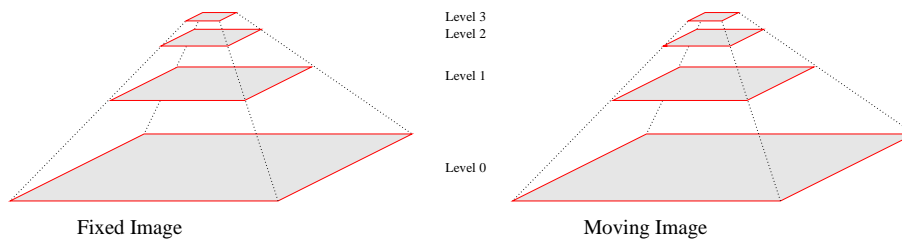


Figure 2.8: The principle of a hierarchical multi-scale approach.

Interpolation

Through an estimated transformation T , a new point given by the coordinates x', y', z' can be computed from the point x, y, z . The obtained positions are mostly not integer numbers, which fit into the discrete raster of an image. An interpolation method maps intensity values that are given in the sub-pixel area to the integer raster. Figure 2.9 illustrates the integer raster (black lines) and the blue sub-pixel grid.

Commonly used interpolation methods such as nearest neighbour (NN), linear and bi-cubic methods are explained in [43]. In Figure 2.10(a) the principle of NN interpolation is shown.

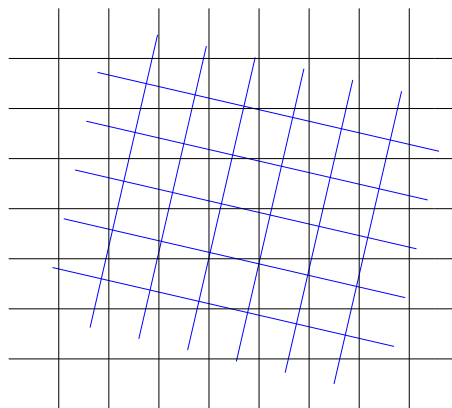


Figure 2.9: Integer raster (black lines) and sub-pixel raster (blue lines).

The value of the nearest neighbour point in the sub-pixel grid (blue lines) is assigned to the according intersected grid point into the integer raster (black lines). In contrast, the linear interpolation sets a linear weighted intensity value of the 4 surrounding points of the sub-pixel raster (shown in Figure 2.10(b)).

Several voxel-based registration algorithms transform the moving image iteratively to reach an overall alignment with the fixed image. During the registration process, interpolation is frequently performed. Therefore, interpolation is an important issue for efficiency and accuracy. The survey [37] also refers to more complex interpolation approaches that are used in image registration. However in practise due to runtime constraints mainly linear interpolation is performed.

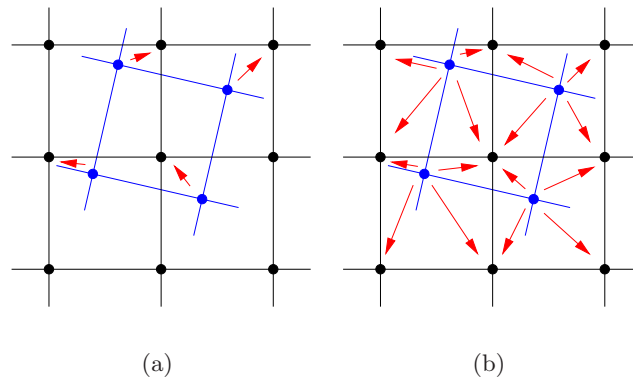


Figure 2.10: Two important interpolation methods: (a) nearest neighbour and (b) linear interpolation.

2.4 Related Work

The previous sections have given an overview on components that are important to set-up a general registration process. In medical image registration current research is very active. The diversity of complex motions in the human body needs different methods to achieve practical results.

In practice e.g. rigid registration methods are only appropriate for bone and brain alignment. Particularly, the number of approaches for an alignment using transformations that are described by a large number of DoF is rapidly increasing.

The transformation T can be regarded as a deformation field that records the 3D displacement vector at each voxel in image B needed to align it with the corresponding position in image A.

In [18] rigid registration algorithms are also referred to. A point based method such as the *Procrustes* analysis aligns two given sets of extracted corresponding points.

Surface fitting approaches, such as the “head-hat” algorithm, was proposed by Pelizzari et al. [34]. This algorithm is applied to align MRI and PET images of the brain.

Besl and McKay [4] proposed the iterative closest point (ICP) algorithm for the registration of 3D shapes. ICP is widely used to align surfaces extracted from medical images.

3D rigid registration is based on a transformation including 6 DoF. In each case three parameters describe rotation and translation to describe the transformation between corresponding points in image A and B. The affine transformation is an extension to the rigid model with the 6 DoF. By adding additional DoF this linear model can be extended to a non-linear trans-

formation model.

However, rigid registration methods cannot handle complex tissue deformation such as e.g. in lungs, breast and liver. Consequently, there is a need for non-linear methods to achieve good results in e.g. modelling respiratory and cardiac motion.

The following part of the thesis gives a review of fundamental approaches in non-linear registration. Non-linear registration methods consist of a transformation, a similarity measure and an optimisation process, which provides the optimal set of transformation parameters with respect to the similarity measure. Non-linear registration can be formulated in a mathematical way as

$$\operatorname{argmax}_d (d (A(x), T(B(x)))),$$

where the optimiser maximises the similarity criterion d between the image A and the moving image B, which is deformed by the currently estimated transformation.

The transformation T is given as $T : (x, y, z) \rightarrow (x', y', z')$, where the location (x', y', z') in image B represents the corresponding location (x, y, z) in image A.

In [52] non-linear transformations are represented by high-order polynomials with third, fourth and fifth order, leading to 60, 105 and 168 DoF, respectively. In the image registration survey of Zitova [55], global and local mapping models are distinguished. Global transformation models with low order polynomials cannot align local deformations. Second or third order polynomials are used for satellite image alignment in combination with a least square fit. In the medical area, the modelling of local deformations is essential. The idea of increasing the polynomials order of the transformation model introduces artefacts [18] instead of a more local alignment. As mentioned in [18], i.a. Goshtasby proposed a method based on high-order polynomials, but these models are rarely adopted to non-linear medical image registration.

2.4.1 Spline-based Registration

This type of non-linear registration employs the concept of splines to model a deformation. Originally, splines are long flexible strips of wood or metal and they were used to model the surfaces of ships and planes. These splines were bent by attaching different weights along its length.

This technique is used for interpolation or approximation of scattered image data. A smooth transformation is obtained from a set of points in image B and the corresponding points in image A. These points can be either manually or automatically determined landmarks or selected control points.

Rohr et al. [39] suggested an elastic image registration method based on a set of corresponding anatomical landmarks and approximating thin-plate splines (TPS) [7]. This approach extracts several types of landmarks in a semi-automatic manner by means of a 3D differential operator. They extended the TPS approach [7] to an approximation scheme to incorporate isotropic as well as anisotropic landmark errors. TPS are defined as a linear combination of radial basis functions (RBF). By means of the landmarks or control points and the RBF, the TPS interpolation estimates a dense and a globally smoothed deformation field. The disadvantage of a TPS interpolation is, that the arbitrary distribution of the control points perform a global influence on the deformation, since the RBF have infinite support.

By using intensity-based techniques, Rueckert et al. presented in [40] another approach for non-linear registration which was applied to contrast enhanced breast MRI. The registration method is based on a global and a local motion model. The global model describes the overall rigid motion of the breast, whereas the local model specifies local deformations. Local deformations are modelled by free form deformations (FFD) which are estimated over a defined regular grid of control points. Rueckert et al. have shown that their algorithm achieves better results on breast MRI than simple rigid or affine registration methods. This algorithm is outlined in more detail in Appendix C.

In [16] a locally affine registration approach is proposed, that matches two surfaces described by points. Each point of the first surface is deformed by a local affine transformation. An additional constraint arranges for smoothness of the deformation along the 3D surface.

In 2004, Xie et al. [54] proposed a multi-level non-linear registration approach based on hierarchical B-splines. A major problem of traditional B-spline registration is the choice of the number of control points. Few control points achieves only a rough global registration result. Using a large number of control points, the deformation may exhibit local oscillations and it significantly increases computation time. An adaptive local refinement of the hierarchical B-splines overcomes this problem.

Similar to [54], Schnabel et al. presented an approach based on B-splines, that uses a non-uniform grid to control the FFD deformations. They introduced a status of each control point in a dense grid. Active control points are allowed to move during the registration process, whereas passive control points remain fixed. A segmentation step before registration provides the information for assigning the status to each control point.

Mattes et al. [30] proposed an algorithm for PET-CT registration of the chest by using a rigid body transformation in combination with localized cubic B-splines to model the motion between the two image. A mutual information based similarity criterion is applied to this inter-modality registration process. The deformation is defined on a regular grid and is parameterised by several coefficients. Together with a spline-based representation of images and a histogram estimate, the deformation model results in a closed-form expressions for the similarity criterion and gradient calculations.

2.4.2 Optical Flow Based Registration

The optical flow (OF) approach is based on the work of Horn and Schunk [21] and originates from physics. In computer vision OF approaches estimate the motion between two frames that are taken at time t and $t + \Delta t$. In the case of image registration we can use this approach to model tissue motion and deformation. Assuming intensity value constancy of particular points in two frames $f(x, y, z, t)$ and $f(x + dx, y + dy, z + dz, t + dt)$, the objective of OF is to determine dx , dy and dz such that

$$f(x, y, z, t) = f(x + dx, y + dy, z + dz, t + dt). \quad (2.15)$$

By means of Taylor series expansion and ignoring high order terms the fundamental optical flow equation is denoted as:

$$\nabla f \vec{v} + \Delta f = 0 \quad (2.16)$$

Δf describes the temporal difference between the two frames, ∇f describes the spatial derivatives of the moving image and \vec{v} is the motion between the two frames.

To use OF in practice a smoothness regularisation must be applied to the motion field v . In the survey of Beuchemin et al. [5] different OF techniques are explained. In our case only the differential optical flow method is of importance. Mentioned approaches like frequency based, correlation based, multiple motion and temporal refinement methods are referred to for the sake of completeness.

The introductory section of Chapter 3 covers the details of the differential optical flow in conjunction with the approach of Periaswamy [36].

The differential method computes a motion from spatial and temporal derivatives of images. The assumption of continuity of the image in time and space domain allows to compute a global or a local model. The global method needs the mentioned smoothness constraint to get a dense, smooth motion field over the complete image region. The local methods find the best fit for \vec{v} by means of simple optimisation in their local neighbourhood. In Barron et al. [3] a quantitative evaluation of several OF methods is given.

2.4.3 Elastic Registration

In elastic registration the estimation of the deformation field is reduced to a search for optimal parameters, instead of using a parametric mapping function such as in spline- and polynomial-based registration. In 1989, Bajcsy et al. [2] proposed the idea of elastic registration. The moving image B can be viewed as being composed of an elastic material, such as rubber. Through the physical process of stretching and bending, the image B gets aligned with image A. Two forces guide the registration process. The external force takes effect on the elastic material from outside and the counteracting internal force controls not to lose the equilibrium state of the material's shape. The Navier linear elastic partial differential equation (PDE) (Equation 2.17) describes the procedure of finding a solution at equilibrium with a minimum energy state. The deformation field is represented by u , ∇ denotes the gradient operator and ∇^2 the Laplace operator. λ and μ are elasticity constants. The external force f , which may be based on distances or voxel similarity measurements governs the registration process.

$$\mu \nabla^2 u(x, y, z) + (\lambda + \mu) \nabla(\nabla u(x, y, z)) + f(x, y, z) = 0 \quad (2.17)$$

In [47] the external force depends on the distance between surfaces of anatomical structures. Davatzikos et al. [12] used curved boundary structures for similarity measurements and optimisation.

In contrast, Peckar et al. [33] suggested an incorporation of elastic deformation and the prescribed deformation field instead of the external force.

Large localised deformations can not be estimated since the deformation energy caused by stress increases proportionally with the strength of the deformation. Fluid registration handles this problem by relaxing this constraint over time.

2.4.4 Fluid- and Diffusion-Based Registration

The concept of a viscous fluid model controls the image deformation in this type of registration method. The moving image B is modelled as a thick fluid that flows out to match image A. Fluid registration is mainly used in areas such as inter-subject registration, where large deformations and large degrees of variability occur. A comparison of fluid based methods is given in [51], where numerical experiments highlight the results in computational effort and accuracy for several PDE solvers. The fluid registration model is based on the Navier-Stokes PDE (Equation 2.18), which is similar to the elastic registration approach. In contrast, the velocity field v is used instead of the displacement field u . The mathematical relationship between u and v is given in Equation 2.19.

$$\mu \nabla^2 v(x, y, z) + (\lambda + \mu) \nabla(\nabla v(x, y, z)) + f(x, y, z) = 0 \quad (2.18)$$

$$v(x, y, z, t) = \frac{\partial u(x, y, z, t)}{\partial t} + v(x, y, z, t) \nabla u(x, y, z, t) \quad (2.19)$$

Bro-Nielsen et al. [8] proposed a fast implementation for solving the PDE in Equation 2.18. Before that, solving this equation with a conventional numerical method such as successive over-relaxation (SOR) [38] lead to long computation times.

In [8] the PDE is solved by deriving a convolution filter from the eigenfunctions of the linear elasticity operator under the assumption that the viscosity of the fluid is nearly constant. In their work it has also been shown that this solution is similar to a regularisation by convolution with a Gaussian kernel as in the diffusion based registration technique proposed by Thirion [46]. This work presents the concept of diffusing models to perform image registration. The main idea of this algorithm (in literature known as *Demons* algorithm) is to consider the objects boundaries in the images as membranes.

The *Demons* algorithm, included in the ITK [32], is based on a normalised optical flow concept. In a first step, forces are computed through OF. These forces then produce an overall deformation field. A Gaussian filtering step provides the regularisation of the deformation field. Pennec et al. [35] applied the *Demons* algorithm to non-linear 3D ultra-sound image registration.

2.4.5 Biomechanical Registration

Edwards et al. [14] proposed a three-component registration model to simulate the properties of rigid structures such as a bone, elastic structures such as the white, gray matter or other soft tissue and fluid structures such as the cerebrospinal fluid. This approach uses a simple finite elements method (FEM) to model deformations. FEM is an analysis technique that solves partial differential equations, which describe complex systems and processes that are spatially distributed. In [14] a triangulation of the images allows a labelling of each node to one of the three physical properties. By deforming the mesh, an energy objective function can be optimised. Several energy terms for constraining the deformations were suggested. The distances between selected landmarks provide a basis for the energy objective function. Severe disadvantages of FEM based registration are its high computation times and the need for prior segmentation.

Chapter 3

Elastic Registration in the Presence of Intensity Variations

3.1 Introduction

In [36], Periaswamy et al. suggested a general purpose registration approach. This algorithm models a deformation in a locally affine and globally smooth fashion. In our case, we need a registration technique, that copes with small, local deformations, like in the case of the lung's alveolus, and that can model large deformations caused by respiration. The presented non-feature based registration approach affords an alignment of images with intensity variation, which is relevant if images are taken with contrast agents. The algorithm is intended for the alignment of mono-modal images. This Chapter presents the details of Periaswamy's registration approach. At first, Section 3.2 outlines the principle of the OF approach. The following sections provide information on how to compute a locally affine model (3.3) with intensity variations (3.4) and on how to gain a globally smooth deformation field (3.5). Further sections provide details on computing the derivatives (3.7), computing the inverse of a matrix (3.8) and additional details (3.9). A summary concludes this chapter giving information about observed advantages and disadvantages of the implementation and it presents an outlook on further improvements.

3.2 Optical Flow

Periaswamy's suggested registration approach is based on a differential optical flow method. Horn and Schunk [21] presented global techniques to determine motion from OF, while the work of Lucas and Kanade [26] deals with a local method. Global methods estimate a dense motion field and local methods provide robustness under noise [3].

Referring to the previous chapter, an OF estimate assumes an intensity constancy in the concerned frames t and $t + dt$. A second assumption, the velocity smoothness constraint, signifies that nearby points move in a similar direction.

There exist several techniques to solve the OF problem dealing with these two assumptions. Details of these methods can be found in the specified papers [21], [26] and [3].

A differential flow or motion estimate is based on the computation of the spatial and temporal

derivatives directly from the intensity values of the images. The following equation assumes the two continuous images $f(x, y, z, t)$ and $f(\hat{x}, \hat{y}, \hat{z}, \hat{t})$ as identical, if

$$f(x, y, z, t) = f(\hat{x}, \hat{y}, \hat{z}, \hat{t}) \quad (3.1)$$

where $f(x, y, z, t)$ represents the fixed image and $f(\hat{x}, \hat{y}, \hat{z}, \hat{t})$ stands for the moving image. We can denote the term $f(\hat{x}, \hat{y}, \hat{z}, \hat{t})$ as $f(x + dx, y + dy, z + dz, t + dt)$ by introducing the following definitions: $\hat{x} = x + dx$, $\hat{y} = y + dy$, $\hat{z} = z + dz$ and $\hat{t} = t + dt$.

Using a Taylor series expansion, $f(x + dx, y + dy, z + dz, t + dt)$ can be expressed as a function of location and time:

$$f(x + dx, y + dy, z + dz, t + dt) = f(x, y, z, t) + \frac{\partial f}{\partial x} dx + \frac{\partial f}{\partial y} dy + \frac{\partial f}{\partial z} dz + \frac{\partial f}{\partial t} dt + O(\partial^2) \quad (3.2)$$

Ignoring the high order terms $O(\partial^2)$ of the series expansion, applying it to Equation 3.2 and simplifying the result leads to

$$\frac{\partial f}{\partial x} \cdot \frac{dx}{dt} + \frac{\partial f}{\partial y} \cdot \frac{dy}{dt} + \frac{\partial f}{\partial z} \cdot \frac{dz}{dt} + \frac{\partial f}{\partial t} = 0. \quad (3.3)$$

By multiplying with dt and performing further simplification, the previous equation can be written as

$$(\nabla f)^T \vec{v} + f_t = 0, \quad (3.4)$$

where $\vec{v} = (dx, dy, dz)^T$ denotes the motion or velocity vector and $\nabla f = (f_x, f_y, f_z)^T$ stands for the spatial derivatives of the image $f(x, y, z, t)$ in each of the x , y and z directions. The variable f_t is the temporal derivative. The derived result 3.4 is called the optical flow constraint equation or gradient constraint equation. Following the paper [3] there exist three main approaches to solve the OF constraint equation. In [21], they defined the dissimilarity between two frames as a combination of the gradient constraint and a smoothness term. The additional global smoothness term constrains the velocity or motion field. Over a domain V the function ϵ_1 can be minimized to estimate the motion field $\vec{v} = (v_1, v_2, v_3)^T$. The variable λ reflects the influence of the additional smoothness term.

$$\epsilon_1 = \int_V \left((\nabla f)^T \cdot \vec{v} + f_t \right)^2 + \lambda^2 \left(\|\nabla v_1\|^2 + \|\nabla v_2\|^2 + \|\nabla v_3\|^2 \right)^2 dx dy dz \quad (3.5)$$

In the work of Barron [3], they also give a description of Uras's et al. method, where the solution is based on a second order technique. The images are divided into smaller sub-images. For each sub-image the best estimate for the local velocity field is chosen considering the Hessian matrix.

The third approach is based on the work of Lucas and Kanade [26]. They suggested to minimise a locally defined error function. (Equation 3.6). In each local region Ω a closed

form solution produces the velocity field \vec{v} , analytically.

$$\epsilon_2 = \sum_{x,y,z \in \Omega} W(x,y,z)^2 \left((\nabla f)^T \cdot \vec{v} + f_t \right)^2 \quad (3.6)$$

The variable $W(x,y,z)$ denotes a windowing function, which controls the influence of the constraint depending on the location in this region Ω .

In the following paragraphs, we concentrate on the derivation of this error function ϵ_2 to implicate Periaswamy's idea.

Starting from the papers [48] and [41], the time term is skipped for convenience and a formal redefinition of the image notation is established with $f_{moving}(\alpha) = f(\hat{x}, \hat{y}, \hat{z}, t + 1)$ and $f_{fixed}(\alpha) = f(x, y, z, t)$. The vector \vec{v} defines the correspondences as a displacement at each location of the $f_{moving}(\alpha + v)$ and the fixed image $f_{fixed}(\alpha)$.

An error function ϵ in a squared difference fashion can be expressed as

$$\epsilon = \sum_{x,y,z \in \Omega} W(x,y,z)^2 (f_{moving}(\alpha + v) - f_{fixed}(\alpha))^2 \quad (3.7)$$

Truncating the high order terms in Equation 3.2 and using the new formal definition yields the following approximation in a compact manner

$$f_{moving}(\alpha + v) = f_{moving}(\alpha) + \nabla f \cdot \vec{v} \quad (3.8)$$

Certainly, ∇f denotes the spatial derivatives of the moving image. Employing the previous equation to the error function given in Equation 3.7 results in

$$\epsilon = \sum_{x,y,z \in \Omega} W(x,y,z)^2 (f_{moving}(\alpha) + \nabla f \cdot \vec{v} - f_{fixed}(\alpha))^2 \quad (3.9)$$

Comparing Equation 3.9 to Equation 3.6, it is obvious that the temporal derivative f_t is the difference of $f_{moving}(\alpha)$ and $f_{fixed}(\alpha)$. This quadratic error function can be minimised analytically with respect to \vec{v} . Differentiating this error function in Equation 3.9 with respect to \vec{v} and setting the result equal to zero produces

$$\sum_{x,y,z \in \Omega} W^2 (f_{moving}(\alpha) - f_{fixed}(\alpha) + \nabla f \cdot \vec{v}) \nabla f = 0 \quad (3.10)$$

Assuming a constant motion within the domain Ω the following equation can be written

$$\left(\sum_{x,y,z \in \Omega} W^2 \nabla f \nabla f^T \right) \vec{v} = \sum_{x,y,z \in \Omega} W^2 (f_{moving}(\alpha) - f_{fixed}(\alpha)) \nabla f \quad (3.11)$$

by using $(\nabla f \cdot \vec{v}) \nabla f = (\nabla f \nabla f^T) \vec{v}$.

A simplification with

$$G = \sum_{x,y,z \in \Omega} W^2 \nabla f \nabla f^T \quad (3.12)$$

and

$$\vec{w} = \sum_{x,y,z \in \Omega} W^2 (f_{moving}(\alpha) - f_{fixed}(\alpha)) \nabla f \quad (3.13)$$

offers a linear equation system (Equation 3.14), which can easily be solved for \vec{v} by computing the inverse of G and a multiplication with the right hand-side term \vec{w} . In the 3D case the solution of this linear system gives a 1×3 displacement vector as a result.

$$G\vec{v} = \vec{w} \quad (3.14)$$

The outer product of $\nabla f \nabla f^T$ results in a symmetric matrix with a dimension of 3×3 . Sampling all locations in x, y and z directions within the region Ω and the element-wise summation of the resulting matrices, the matrix G can be computed as follows

$$G = \begin{pmatrix} \sum f_x f_x & \sum f_x f_y & \sum f_x f_z \\ \sum f_y f_x & \sum f_y f_y & \sum f_y f_z \\ \sum f_z f_x & \sum f_z f_y & \sum f_z f_z \end{pmatrix} \quad (3.15)$$

The matrix G is also known as the structure tensor [22]. The previous description has given an outline on how to compute a displacement vector in a small region Ω . The optical flow technique based on Lucas and Kanade was suggested for tracking feature points in images. Applying this technique to image registration, a 3D sliding mask makes it possible to compute a displacement vector for each traversed voxel. At each location in the moving image the corresponding region in the fixed image is tracked. The size of the sliding mask represents the spatial neighbourhood region Ω . The following sections explain the ideas of Periaswamy's approach utilising the preceding solution in a similar way.

3.3 Local Affine Model

As mentioned before, registration is the process of aligning two images. As a result a deformation field representing the transformation T is computed. In this work we adopt the same notation as in [36]. The principle of the algorithm is a modelling of the motion between a fixed and a moving image by means of a locally applied affine transformation. As defined in the previous section, $f(x, y, z, t)$ represents the fixed image and $f(\hat{x}, \hat{y}, \hat{z}, \hat{t})$ the moving image. Periaswamy et al. introduced a local affine model in their work. The relation of \hat{x} to x , \hat{y} to y and \hat{z} to z can be described by the affine transformation.

Assuming that the intensity values are nearly constant in a small spatial neighbourhood, Equation 3.16 explains the transformation model. Within the differential notation the variable t denotes the temporal parameter. The parameters x, y and z cover all voxel locations of the corresponding image.

$$\begin{aligned} f(x, y, z, t) = f(m_1 x + m_2 y + m_3 z + m_{10}, \\ m_4 x + m_5 y + m_6 z + m_{11}, \\ m_7 x + m_8 y + m_9 z + m_{12}, t - 1) \end{aligned} \quad (3.16)$$

In Equation 3.16, the variables $m_1 \dots m_9$ represent the affine parameters, while m_{10} , m_{11} and m_{12} are the translation parameters. The affine transformation with 12 DoF allows shearing, scaling, translation, and rotation in any direction. For each small spatial neighbourhood Ω of traversed voxels the appropriate parameters can be estimated by minimizing a quadratic error function.

$$E(\vec{m}) = \sum_{x,y,z \in \Omega} [f(x, y, z, t) - f(m_1x + m_2y + m_3z + m_{10}, \\ m_4x + m_5y + m_6z + m_{11}, \\ m_7x + m_8y + m_9z + m_{12}, t - 1)]^2 \quad (3.17)$$

This error function, given in Equation 3.17, is based on the similarity measure SSD presented in Chapter 2.

Computing the unknown parameters for all locations x , y and z in the images yields a non-linear deformation model. Because of the non-linearity in its unknowns $\vec{m} = (m_1 \dots m_{12})$ the equation can not be minimized analytically. By means of the Taylor series expansion and truncating after the first order term the non-linear quadratic error function can be approximated through a linear error function

$$E(\vec{m}) \approx \sum_{x,y,z \in \Omega} [f(x, y, z, t) - (f(x, y, z, t) + \\ + (m_1x + m_2y + m_3z + m_{10} - x)f_x(x, y, z, t) \\ + (m_4x + m_5y + m_6z + m_{11} - y)f_y(x, y, z, t) \\ + (m_7x + m_8y + m_9z + m_{12} - z)f_z(x, y, z, t) \\ - f_t(x, y, z, t))]^2 \quad (3.18)$$

where now f_x , f_y , f_z and f_t represent the derivatives of the moving image $f(x, y, z, t)$, as mentioned in the paper [36]. In Section 3.7 it is explained how to compute the spatial and the temporal gradients from a combination of moving and fixed images. To make the derivation convenient we first assume the computation of the gradients of the moving image. Further simplification of Equation 3.18 reduces the error function to Equation 3.19. This approximation results in a quadratic function with linear unknowns \vec{m} .

$$E(\vec{m}) \approx \sum_{x,y,z \in \Omega} [f_t(x, y, z, t) \\ - (m_1x + m_2y + m_3z + m_{10} - x)f_x(x, y, z, t) \\ - (m_4x + m_5y + m_6z + m_{11} - y)f_y(x, y, z, t) \\ - (m_7x + m_8y + m_9z + m_{12} - z)f_z(x, y, z, t)]^2 \quad (3.19)$$

If a vector form notation is introduced, Equation 3.19 can be compactly written as

$$E(\vec{m}) = \sum_{x,y,z \in \Omega} [k - \vec{c}^T \vec{m}]^2 \quad (3.20)$$

where the scalar k denotes

$$k = f_t + xf_x + yf_y + zf_z \quad (3.21)$$

and the vector \vec{c} expresses

$$\vec{c} = (xf_x \ yf_x \ zf_x \ xf_y \ yf_y \ zf_y \ xf_z \ yf_z \ zf_z \ f_x \ f_y \ f_z)^T. \quad (3.22)$$

Due to linearity of the equation in its unknown parameters $\vec{m} = (m_1 \dots m_{12})$, the parameters can be estimated from an analytic minimization by differentiation with respect to them and setting this error function equal to zero:

$$\frac{dE(\vec{m})}{d\vec{m}} = \sum_{x,y,z \in \Omega} -2\vec{c} [k - \vec{c}^T \vec{m}] \stackrel{!}{=} 0. \quad (3.23)$$

Solving for the parameters \vec{m} results in

$$\vec{m} = \left[\sum_{x,y,z \in \Omega} \vec{c}\vec{c}^T \right]^{-1} \left[\sum_{x,y,z \in \Omega} \vec{c}k \right] \quad (3.24)$$

As a result of this derivation we use the compact Equation 3.24 to calculate the locally affine model analytically from a small spatial neighbourhood. The estimated parameters \vec{m} for each location x , y and z and their corresponding neighbourhood Ω allow a dense mapping between moving and fixed image. Mathematically, the term $\left[\sum_{x,y,z \in \Omega} \vec{c}\vec{c}^T \right]$ forms a 12×12 matrix. This matrix is usually invertible assuming that the spatial neighbourhood is large enough and includes sufficient image content. The second term $\left[\sum_{x,y,z \in \Omega} \vec{c}k \right]$ results in a column vector with a dimension of 1×12 .

3.4 Intensity Variations

Only if a brightness constancy constraint is fulfilled, the derivation of the model, given in the previous section 3.3 is valid. If the intensity values of the images change between different images, Periaswamy suggests to introduce two additional parameters m_{13} and m_{14} , that compensate local contrast and brightness changes [31]. The extended form of Equation 3.16, including the varying spatial parameters yields

$$\begin{aligned} m_{13}f(x, y, z, t) + m_{14} &= f(m_1x + m_2y + m_3z + m_{10}, \\ & m_4x + m_5y + m_6z + m_{11}, \\ & m_7x + m_8y + m_9z + m_{12}, t - 1) \end{aligned} \quad (3.25)$$

Due to the linearity of the additional parameters the derivation is similar to the one given in Section 3.3. Using the Taylor series expansion and ignoring higher order terms, the approxi-

mated error function results in

$$E(\vec{m}) = \sum_{x,y,z \in \Omega} [k - \vec{c}^T \vec{m}]^2 \quad (3.26)$$

The scalar k given in 3.27 and the vector \vec{c} given in 3.28 extend the definitions of k and \vec{c} of the pure affine model presented in the Equations 3.21 and 3.22, respectively. f describes the intensity value of the moving image at location (x, y, z) .

$$k = f_t - f + x f_x + y f_y + z f_z \quad (3.27)$$

$$\vec{c} = (x f_x \quad y f_x \quad z f_x \quad x f_y \quad y f_y \quad z f_y \quad x f_z \quad y f_z \quad z f_z \quad f_x \quad f_y \quad f_z \quad -f \quad -1)^T \quad (3.28)$$

Minimization of this error function is accomplished, as before, by differentiating, setting the result equal to zero and solving for the parameters \vec{m} . The solution takes the same form as in 3.24, with k and \vec{c} defined as above. Intensity variations are typically a significant source of error in differential motion estimation. The addition of contrast and brightness terms allows to accurately register images in the presence of local intensity variations.

An intensity modulation is one way to explain the mapping between the two grey-value ranges of the images. The next section highlights an alternative method.

3.5 Smoothness

The derivation in Section 3.4 assumes that the affine, brightness and contrast parameters do not change within the spatial neighbourhood. There is a natural trade-off in choosing the size of this neighbourhood Ω . A larger area makes it more likely that the term $\left[\sum_{x,y,z \in \Omega} \vec{c} \vec{c}^T \right]$ in Equation 3.24 will be invertible. A small neighbourhood preserves the assumption of constant parameters in this area. To avoid finding the trade-off between the parameters constancy assumption and the ability to invert the term, a smoothness constraint is introduced [21]. The error function, given in Equation 3.26, is extended with an additional term $E_s(\vec{m})$, which denotes this constraint. The term $E_b(\vec{m}) = [k - \vec{c}^T \vec{m}]^2$ now represents the part of Equation 3.26 without the summation. The vectors \vec{m} and \vec{c} have the same form as described in the Equations 3.27 and 3.28, respectively. The resulting error function is given as follows in Equation 3.29.

$$E(\vec{m}) = E_b(\vec{m}) + E_s(\vec{m}) \quad (3.29)$$

The introduction of a smoothness constraint term $E_s(\vec{m})$ leads to a quadratic equation given in 3.30, where λ_i represents a positive constant value, that controls the relative weight given to the smoothness constraint on each element of the parameter vector \vec{m} .

$$E_s(\vec{m}) = \sum_{i=1}^{14} \lambda_i \left[\left(\frac{\partial m_i}{\partial x} \right)^2 + \left(\frac{\partial m_i}{\partial y} \right)^2 + \left(\frac{\partial m_i}{\partial z} \right)^2 \right] \quad (3.30)$$

The complete error function, given in Equation 3.29, is solved for the minimising parameters $\vec{m} = (m_1 \dots m_{14})$. Differentiation with respect to the parameters \vec{m} and setting the error

function equal to zero leads to:

$$\frac{dE(\vec{m})}{d\vec{m}} = \frac{dE_b(\vec{m})}{d\vec{m}} + \frac{dE_s(\vec{m})}{d\vec{m}} \stackrel{!}{=} 0 \quad (3.31)$$

In Equation 3.32, the derivative of the first term $E_b(\vec{m})$ leads to a similar result as given in 3.23.

$$\frac{dE_b(\vec{m})}{d\vec{m}} = -2\vec{c}[k - \vec{c}^T \vec{m}] \quad (3.32)$$

The derivative of $E_s(\vec{m})$ is computed by expressing the partial derivatives, $\frac{\partial m_i}{\partial x}$, $\frac{\partial m_i}{\partial y}$ and $\frac{\partial m_i}{\partial z}$ with discrete approximations [21]. The differentiation results in

$$\frac{dE_s(\vec{m})}{d\vec{m}} = 2L(\overline{\vec{m}} - \vec{m}) \quad (3.33)$$

where $\overline{\vec{m}}$ represents the mean values of each parameter over a small spatial neighbourhood. In case of a full affine model with a contrast and a brightness parameter estimate, the variable L denotes a 14×14 identity matrix multiplied with the positive constant value vector $\vec{\lambda} = (\lambda_1 \dots \lambda_{14})^T$. Applying the Equations 3.32 and 3.33 to 3.31, setting this equation equal to zero and solving for the parameters \vec{m} for each location in x , y and z direction, yields a linear system, which is intractable to solve. As mentioned in [36] an iterative scheme is employed to solve the resulting Equation 3.34 for the parameters \vec{m} .

$$\vec{m}^{(j+1)} = (\vec{c}\vec{c}^T + L)^{-1} (\vec{c}k + L\overline{\vec{m}}^{(j)}) \quad (3.34)$$

For each iteration j , $\vec{m}^{(j+1)}$ is estimated from the current $\overline{\vec{m}}^{(j)}$. The initial estimate for the parameters $\vec{m}^{(0)}$ is computed from the closed-form solution explained in Section 3.4.

Applying the proposed smoothness constraint to the algorithm results in a dense locally affine but globally smooth deformation field. The drawback is that the minimization is no longer analytic, but the iterative minimization scheme converges relatively quick and stable as referred to in the paper.

3.6 Choosing the Model of Transformation

The previous derivation of the fully local affine model estimates the parameters for scaling, shearing, rotation, translation, contrast and brightness. As mentioned before, this model demands an inversion of a 14×14 matrix at each voxel location. A simplification done by restricting the complete vector \vec{c} , given in 3.28, and an accordant computation of the scalar k , denoted in 3.27, leads to a less complex model of transformation. We reproduced these models from the given *Matlab* implementation, because the theory is unquoted in the paper of Periaswamy.

The basic model, which includes only translation, can be derived from the theory as stated in Section 3.2. The complete vector \vec{c} simplifies to:

$$\vec{c} = (f_x \ f_y \ f_z)^T \quad (3.35)$$

and the scalar k (given in Equation 3.27) is now computed as follows:

$$k = f_t \quad (3.36)$$

Hence, the computation of the parameters, using Equation 3.26, results in the 1×3 translation vector. This model equals to the standard local approach presented in 1981 by Lucas and Kanade [26]. In the course of this work we call this simple model as *Model 1*.

An introduction of the contrast and brightness parameters expresses an initial model

$$\begin{aligned} m_{13}f(x, y, z, t) + m_{14} &= f(x + m_{10}, \\ & y + m_{11}, \\ & z + m_{12}, t - 1) \end{aligned} \quad (3.37)$$

Further derivation also results in Equation 3.26, where this time \vec{c} and k are denoted as

$$k = f_t - f \quad (3.38)$$

and

$$\vec{c} = (f_x \ f_y \ f_z \ -f \ -1)^T. \quad (3.39)$$

This restricted form of a transformation model is named *Model 2*. This model uses translation as transformation and handles intensity variations in the images.

Model 3 omits the contrast and brightness parameter estimation. The following \vec{c} and k , denoted in Equation 3.41 and 3.40 result in the parameter vector \vec{m} by inverting a 9×9 matrix.

$$k = f_t + xf_x + yf_y + zf_z \quad (3.40)$$

$$\vec{c} = (xf_x \ yf_x \ zf_x \ xf_y \ yf_y \ zf_y \ xf_z \ yf_z \ zf_z \ f_x \ f_y \ f_z)^T \quad (3.41)$$

The resulting parameters \vec{m} with a dimension of 1×9 describes a pure affine transformation. The complete model, including the 14 estimated affine, contrast and brightness parameters is called *Model 4*. The influence of the restriction on the computation time and the registration results are described in Section 3.10. Chapter 5 illustrates quantitative results comparing these models on image data.

3.7 Computation of the Derivatives

In Periaswamy [36] the computation of the spatial and temporal derivatives is characterised as a critical step. These derivatives of discretely sampled images are often computed as differences between neighbouring intensity values, which lead to substantial errors in the gradient approximations. These errors arise if continuously defined derivatives are applied to discrete images. Periaswamy et al. suggest specifically designed filters for multi-dimensional differentiation for computation of the derivatives [15].

According to this paper, there exists a fair amount of work on the design of discrete differentiators in literature. Many approaches base on an approximation of the derivative of a continuous sinc function. These methods suffer from the need of large filter sizes to achieve

accurate results.

Another adequate approach is the sampling of the Gaussian derivatives. This provides good approximations of the gradients and is less computationally expensive than sinc functions. These two primary methods are not well suited for the computation of derivatives of multi-dimensional images. The proposed, so-called Simoncelli-filters feature the property of separability, which allows differentiation in arbitrary directions by means of linear combinations of axis derivatives. In addition filters are optimally equivariant to rotations, thereby reducing the rotation dependent approximation error.

1D separable filters efficiently convolve the 3D image in each according dimension.

The computation of the derivatives of a discretely sampled signal requires an interpolation step to approximate continuity. The derivative of this continuous signal is then re-sampled at the points of the original sampling lattice. Details on the interpolation methods, needed to design these filters can be found in [15].

Table 3.1 gives a set of first-order derivative kernels of various taps. d_0 represents the values of the interpolator and d_1 denotes the derivatives of the interpolator values.

3-tap	d_0	0.223755	0.552490	0.223755			
	d_1	-0.453014	0	0.453014			
4-tap	d_0	0.092645	0.407355	0.407355	0.092645		
	d_1	-0.236506	-0.267576	0.267576	0.236506		
5-tap	d_0	0.036420	0.248972	0.429217	0.248972	0.036420	
	d_1	-0.108415	-0.280353	0	0.280353	0.108415	
6-tap	d_0	0.013846	0.135816	0.350337	0.350337	0.135816	0.013846
	d_1	-0.046266	-0.203121	-0.158152	0.158152	0.203121	0.046266

Table 3.1: First-order derivative kernels of various taps given in pairs of the interpolator d_0 and its derivatives d_1 . Taken from [15].

Periaswamy suggests a 2-tap filter to smooth the temporal derivative f_t . In our implementation we subtract the fixed image from the moving image and convolve the result with an efficient 1D Gaussian filtering method in each direction. The Gaussian filter mask is expressed in Equation 3.42, where β represents the location in the corresponding dimension. The filter mask gets normalised to a maximal value of 1 by dividing through the term $1/(\sqrt{2\pi}\sigma)$. Figure 3.1 illustrates the normalised 1D Gaussian distribution with $\sigma = 1$. The filtering causes a smooth and a continuous change of the intensity values over the complete image domain.

$$G(\beta) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{\beta^2}{2\sigma^2}} \quad (3.42)$$

As proposed in Periaswamy's work [36], we also apply the Simoncelli-filters to our implementation. The spatial derivatives f_x , f_y and f_z are approximated using a 3-tap filter. As described in the previous optical flow section (3.2), the spatial derivatives are approximated directly from the moving image.

Periswamy et al. used a symmetric definition of the error function $E(\vec{m})$ (Equation 3.17) in their *Matlab* code. We adopt this method for our implementation. Following the unpublished paper of Birchfield [6] in conjunction with Kanade-Lucas-Tomasi tracking [48],[41],

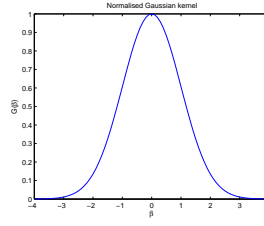


Figure 3.1: The normalised 1D Gaussian distribution with $\sigma = 1$.

the derivatives are now computed including both images. The detailed derivation of this approximation can be found in [6]. The symmetric error function is redefined as follows

$$E(\vec{m}) = \sum_{x,y,z \in \Omega} [f(x - m_{10}, y - m_{11}, z - m_{12}, t) - f(x + m_{10}, y + m_{11}, z + m_{12}, t - 1)]^2 \quad (3.43)$$

The error function, expressed in Equation 3.43, is made symmetrical with respect to the two input windows, but it equals to Equation 3.17. The redefinition of the computation of spatial derivatives ∇f results in Equation 3.44. In contrast to the paper, the provided implementation ∇f involves the constant factor 1/2 directly.

$$\nabla f = \begin{pmatrix} f_x \\ f_y \\ f_z \end{pmatrix} = \begin{pmatrix} \frac{\partial}{\partial x} \frac{1}{2} (f_{moving} + f_{fixed}) \\ \frac{\partial}{\partial y} \frac{1}{2} (f_{moving} + f_{fixed}) \\ \frac{\partial}{\partial z} \frac{1}{2} (f_{moving} + f_{fixed}) \end{pmatrix} \quad (3.44)$$

This method of computing the derivatives with a symmetrical definition promises a more accurate estimate of the translation parameters m_{10} , m_{11} and m_{12} . This symmetric scheme is also used for the more complex models *Model 2-4*.

3.8 Computing the Inverse

Starting from Equation 3.24, it is obvious that a matrix inversion is needed to compute a closed form solution for the parameter vector \vec{m} . This section explains a robust attempt of computing the matrix inverse and verifying its invertibility. Checking the singularity of the matrix is important, because the intensity constancy assumption holding.

Equation 3.24 computes the initial estimate for the parameters $\vec{m} = (m_1 \dots m_{14})$. This equation consists of a simple matrix multiplication with a vector. For the sake of convenience a short notation is introduced, which is shown in Equation 3.45.

$$\vec{m} = \left[\sum_{x,y,z \in \Omega} \vec{c}\vec{c}^T \right]^{-1} \left[\sum_{x,y,z \in \Omega} \vec{c}k \right] = G^{-1} \cdot \vec{w} \quad (3.45)$$

As noted previously, the inverse computation of the matrix $G = \left[\sum_{x,y,z \in \Omega} \vec{c}\vec{c}^T \right]$ is needed.

The vector \vec{b} represents the second term $\vec{w} = \left[\sum_{x,y,z \in \Omega} \vec{c}k \right]$. The resulting parameter vector \vec{m} has a dimension of 1×14 in the case of the fully affine model with brightness and contrast

estimation.

In numerics literature [38],[20] there exist various methods to solve linear equation systems or inverse computations. Periaswamy et al. applied the SVD [38] to invert the matrix G in their *Matlab* implementation. We adopt this robust technique to our implementation.

As a simple starting position we suppose the matrix G to be a square matrix and Equation 3.45 to be a linear system.

The SVD offers a powerful technique for dealing with matrices that are either singular or else numerically very close to singular. If the determinant $\det(G)$ of a matrix G equals to 0, G is called singular. A singular matrix is not invertible.

In the following, a brief outline of the mathematical background of the SVD approach is given, details can be found in [38].

As illustrated in Equation 3.46, the SVD decomposes the matrix G into three matrices U , W and V^T . We introduce a variable N , which denotes the dimension of the matrix G . In the case of a square matrix G , the three decomposed matrices have the same dimension ($N \times N$).

$$G = U \cdot W \cdot V^T \quad (3.46)$$

The matrices U and V are orthogonal (Equation 3.47), which imposes that their inverses are equal to their transposes.

$$U^T U = V^T V = \mathbf{I} \quad (3.47)$$

where \mathbf{I} denotes the identity matrix.

The matrix W is a diagonal matrix, which contains the singular values ω_j of G :

$$W = \begin{pmatrix} \omega_1 & & & \\ & \omega_2 & & \\ & & \dots & \\ & & & \omega_N \end{pmatrix} \quad (3.48)$$

So its inverse is the diagonal matrix whose elements are the reciprocals of the elements ω_j . Note that the inversion of the matrix G can therefore easily be written as

$$G^{-1} = V \cdot \left[\text{diag} \left(\frac{1}{\omega_j} \right) \right] \cdot U^T. \quad (3.49)$$

The SVD offers a simple technique to invert square matrices. There is only a problem, if one of the diagonal elements ω_j equals to zero or is so small that its value is dominated by round-off error. If more than one of the diagonal elements have this problem, then the matrix G has a higher degree of singularity.

In [20] the degree of singularity is referred to as the condition number $\kappa(G)$ of a matrix G . This condition number is defined as the ratio of the largest diagonal element in the matrix W to the smallest. A matrix G is singular (not invertible), if its condition number is infinite. The condition number can be expressed as

$$\kappa(G) = \|G^{-1}\| \cdot \|G\|. \quad (3.50)$$

If the l_2 -Norm is used, $\kappa(G)$ can be computed as follows:

$$\kappa(G) = \frac{\max(\omega_j)}{\min(\omega_j)} \quad (3.51)$$

where max and min of ω_j with $j = 1 \dots N$ denote the largest and smallest singular values of G , respectively.

We use the built-in function of the numerics library *VNL*¹ to compute the SVD. The implementation of the SVD is based on Fortran routines. The SVD method makes it possible to compute the exact value of the condition number of the matrix G . If the matrix G is invertible, which is the case if the condition number exceeds a given precision threshold, the linear equation system (Equation 3.45) is solved for the parameters \vec{m} .

3.9 Implementation Details

The previous section explained the theory for the implementation of the proposed registration approach. In their work Periaswamy et al. suggested a multi-resolution technique to handle large motions. They applied a 5-tap Simoncelli-filter to construct a four-level Gaussian pyramid. In our implementation we use recursive Gaussian filters [13] to perform the coarse-to-fine strategy for the two images A and B. This method reduces the computational effort of building the pyramids. The smoothing, the computation of the derivatives and the filtering using Gaussian filters are performed with a constant number of operations per output point, independent of the filter size.

In opposition to this, the computational effort of constructing the pyramids by convolving with normal Gaussian filters is directly proportional to the filter mask size. The method approximates the Gaussian kernel with a filter bank depending on the scale of the Gaussian kernel.

Going from fine to coarse levels of the pyramid, the resolution is halved in three dimensions. Figure 3.2 illustrates extracted slices from an exhalation and an inhalation image in each level of the down-sampled resolution. These samples have an original dimension of 256×256 in x and y direction. We apply the down-sampling until a given minimal image size *minSize* at coarsest level is reached. In this case we get four levels, with a minimal dimension of 32×32 . Starting from the original and the minimal image resolution the number of levels *NbLevels* can be calculated by using Equation 3.52. We choose the minimum of three resolution values of the images in each direction as the original size *minOriginalSize* to compute the number of levels.

$$NbLevels = \left\lceil \lg \frac{minOriginalSize}{minSize} + 1 \right\rceil \quad (3.52)$$

Before generating the levels of different resolutions for the image A and B, a padding is applied to both images to overcome the problem of boundary effects. We assume a minimal padding size in each direction, so that a convolution at the boundaries of the original sized images is possible. In the implementation the size for the spatial neighbourhood is chosen as $\Omega = 5 \times 5 \times 5$. Periaswamy determined this neighbourhood size empirically and it is a

¹ *VNL* is included in the ITK package [32]

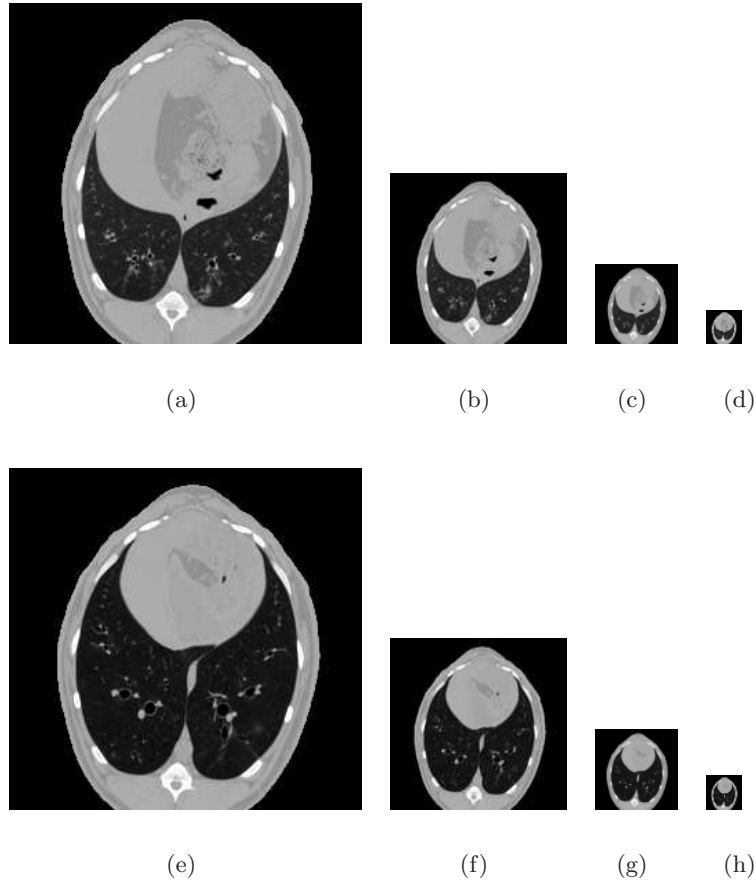


Figure 3.2: (a)-(e) Different resolutions of an image at inhalation state, (f)-(j) Different resolutions of an image at exhalation state.

trade-off between holding the estimated parameters constancy and the feasibility of inverting the resulting term $\sum_{x,y,z \in \Omega} \vec{c}\vec{c}^T$ within Ω .

Therefore a padding of more than 2 voxels on each boundary of the images is necessary to compute the parameters \vec{m} in the coarsest level. The padded areas are filled with constant intensity values. In our case the filling with intensity value equals to zero, representing a content of no information in these areas.

Figure 3.3 illustrates the principle of convolution with and without a padding. The illustration is simplified to the 2D case. Considering the unpadded case, the convolution of the original image grid (blue grid) with the filter mask (red grid box) leads to undefined areas. An additional padding (green grid) makes it possible to compute the appropriate filtered values at the boundaries of the original images.

The padded moving and fixed images act as input with the increased image resolution.

Skipping the details, the presented method can be briefly summarised in Algorithm 1.

Starting at the coarsest level of the pyramid, the parameters for each location in the images are estimated from the fixed and moving image using the proposed algorithm. Extracting the translation parameters m_{10} , m_{11} and m_{12} yields a deformation field, which represents the overall motion between the two images.

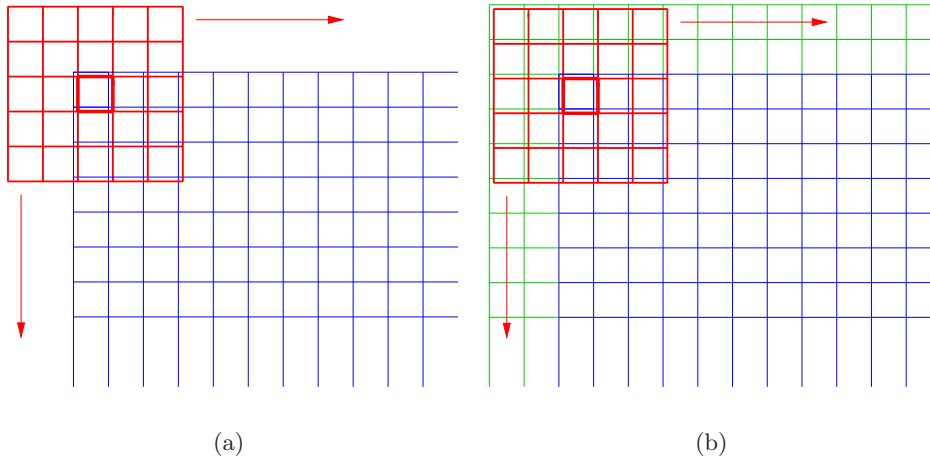


Figure 3.3: A convolution (a) with padding and (b) without padding.

The remaining affine parameters are not applied to the moving image, because the indices equal to zero in the center of each spatial neighbourhood. The contrast and brightness parameters are skipped as well.

A warping, which includes these two parameters m_{13} and m_{14} , obviously performs like an intensity transformation. However, the interest lies on modelling a deformation.

Using this deformation field for warping leads to a new resulting moving image, which is more similar to the fixed image. Periaswamy et al. recommended a four times repetition, the so called outer loops, at each level of the multi-resolution approach to improve the final deformation field.

The computed deformations at each level and each iteration are accumulated yielding a single field of transformations. As mentioned in the previous chapter, the coarse-to-fine strategy allows to register images with large motions. Going up to the next finer resolution level, the intermediate deformation field is up-sampled to the appropriate image size and applied to the moving image as a pre-warping.

As mentioned in [36], the smoothness constraint converges relatively fast after 10 iterations (inner loops).

In each inner loop a given filter mask averages the parameters in the small spatial neighbourhood. In the paper, a 3×3 kernel is suggested to compute \bar{m} in the 2D case. The given kernel (see 3.53) increases the influence of the 4-connected neighbourhood voxels. In our case, we simplify the averaging for the reason of efficiency to a simple mean computation of all surrounding neighbours. Hence, in 3D we compute the mean over 26 neighbourhood voxels.

$$kernel = \frac{1}{20} \begin{pmatrix} 1 & 4 & 1 \\ 4 & 0 & 4 \\ 1 & 4 & 1 \end{pmatrix} \quad (3.53)$$

If *Model 4* is used, the values for $\vec{\lambda} = (\lambda_1 \dots \lambda_{14})^T$ in the matrix L are set to a value of 10^{11} as given in the paper. Considering the *Models 1-3*, the dimension of L is reduced, following

the dimension of the vector \vec{c} and the scalar k . The matrix L controls the influence of the global smoothness constraint on the locally computed parameters.

Initial experiments with the *Matlab* code have confirmed, that the choice of the parameters $\vec{\lambda}$ and the kernel is not critical. In this work, the parameters of L are not changed for the presented experiments in Chapter 5.

Algorithm 1 Outline of Periaswamy's algorithm

1. Load the two input images
Image A: fixed Image
Image B: moving Image
 2. Choose one of the transformation models
Model 1: Translation
Model 2: Translation and Brightness/Contrast Estimation
Model 3: Affine Transformation
Model 4: Affine Transformation and Brightness/Contrast Estimation
 3. Apply a padding to the two input images according to spatial neighbourhood size
 4. Compute the levels of the pyramid for the coarse-to-fine strategy
 5. For each level of the pyramid
 - Compute the raw transformation parameters \vec{m} according to Equation 3.24
 - Iteratively smooth the transformation parameters \vec{m} by using Equation 3.34
 - Extract the deformation field from the computed parameters
 - Up-sample the deformation field
 - Warp the moving image according to the deformation field and store the warped image as new moving image
-

3.10 Summary

Periaswamy et al. presented in their work an optical flow based registration algorithm, which makes it possible to register medical images using a locally affine, but globally smooth transformation model.

As a first step, we have executed the existing *Matlab* code using 2D images as input, which produces slightly faster results than the presented values in the paper. Obviously, this is caused by the constant improvements in computer hardware.

Furthermore, we tried to extend the provided *Matlab* implementation to three dimensions. The extension of the code to three dimensions is straight forward. As input, we used a pair of $128 \times 128 \times 128$ images of sheep lungs. The various parameters are set as given in the paper.

The executions of the extended implementation lead to an enormous time and memory consumption. The resulting warped moving image is achieved after approximately 15 hours,

where the memory consumption increased to more than 16 *GByte*.

As a result, the time and memory consumption of the *Matlab* execution are not feasible in practise.

Our next attempt was to implement the proposed algorithm in *C++* to speed-up the registration process. The reduction of computation time and improved memory management take effect on the implementation. Concrete results are given in Chapter 5.

As mentioned previously, the test runs are accomplished with a pair of $128 \times 128 \times 128$ images to achieve a relatively fast qualitative measurement on how the algorithm behaves.

In practice, a registration of an image dimension larger than $256 \times 256 \times 256$ is of interest. The large number of inverse computations and the extremely high memory consumption assert that the proposed algorithm is not applicable in real-world registration problems. We need at least one computation of an inverse matrix for each voxel in the image by means of the SVD depending on the number of iterations and levels. A matrix inversion precedes the estimate of the condition number $\kappa(G)$.

Another major problem of the algorithm is the high memory consumption, which arises from the smoothing iterations. In an inner loop, all estimated parameters are needed to average the parameters. The obtained result of each voxel requires a second identical structure to store the smoothed parameters. These two structures necessitate memory consumption of $2 \times 128^3 \times 4 \text{ Byte} \times 14 \approx 230 \text{ MByte}$, using $128 \times 128 \times 128$ images, single precision with float-type variables and *Model 4* with 14 parameters. Providing images with a dimension of $256 \times 256 \times 256$ to the algorithm, this value increases to approximately 1.8 *GByte* and that is beyond the requirements of a currently configured personal computer. The following Chapter 4 addresses some algorithm improvements in order to deal with these problems.

Chapter 4

Improvements

4.1 Introduction

Concluding the previous chapter, the proposed registration algorithm is inefficient in time and memory consumption. These drawbacks prevent an application in practical image registration.

Periaswamy provided a *Matlab* implementation, which uses the **SVD** for inverse computations. The necessity to estimate the condition number for all voxels increases the computation time dramatically. Computing the inverse of the matrices with a dimension of 12×12 or 14×14 is computationally expensive.

The suggested iterative smoothing demands high memory requirements, since the inverses of all voxels have to be stored in an additional data structure to conserve computation time. As mentioned, the smoothing is an iterative approach, there is the need of setting up a sensitive number of iterations.

The following sections give an overview on how to reduce computation time by speeding up the estimation of the condition number and the inverse computation (4.2). In Section 4.3 a sub-sampling method is suggested to decrease the number of inverse computations. Attention is given to the interpolation of parameters in an affine matrix. Section 4.4 describes a replacement of the iterative smoothing by a simple Gaussian regularisation.

4.2 Improving the Inverse Computations

Computing the outer product of two vectors with the same dimension yields a quadratic matrix. In numerics literature, e.g. [20], there exist various approaches to invert quadratic matrices. If the matrix G fulfils appropriate conditions, the computation of the inverse can be accomplished in a reduced effort of mathematical operations. Therefore, the following sections give an idea on how to replace the **SVD** under these circumstances with the LU- and Cholesky decomposition.

4.2.1 Computing the Inverse with LU Decomposition

In linear numerics, the LU factorisation decomposes a given matrix G into a product of a lower and an upper triangular matrix of the same dimension. The details are given in [20].

The decomposition can be expressed as

$$G = LU \quad (4.1)$$

where L is the lower and U is the upper triangular matrix. For example, a 4×4 matrix G , L and U are expressed as

$$L = \begin{pmatrix} l_{11} & 0 & 0 & 0 \\ l_{21} & l_{22} & 0 & 0 \\ l_{31} & l_{32} & l_{33} & 0 \\ l_{41} & l_{42} & l_{43} & l_{44} \end{pmatrix} \quad U = \begin{pmatrix} u_{11} & u_{12} & u_{13} & u_{14} \\ 0 & u_{22} & u_{23} & u_{24} \\ 0 & 0 & u_{33} & u_{34} \\ 0 & 0 & 0 & u_{44} \end{pmatrix}. \quad (4.2)$$

If we have to solve a linear equation system of the form $Gx = b$, this equation can be denoted as

$$Ax = (LU)x = L(Ux) = b \quad (4.3)$$

Resulting from this consideration, the system can easily be solved by using a forward and a backward substitution step. The decomposition of G into L and U is performed by Crout's algorithm, which uses a row-wise permutation of the matrix G to compute an unit upper triangular matrix.

Performing Crout's method the elements get stored in one matrix, which makes it possible to decompose in-line and efficiently.

$$U = \begin{pmatrix} u_{11} & u_{12} & u_{13} & u_{14} \\ l_{21} & u_{22} & u_{23} & u_{24} \\ l_{31} & l_{32} & u_{33} & u_{34} \\ l_{41} & l_{42} & l_{43} & u_{44} \end{pmatrix} \quad (4.4)$$

We employ an implementation of the LU decomposition given in [38] to our algorithm. The following section describes how to benefit from the factorisation for estimating the condition number.

4.2.2 Estimating the Condition Number

As mentioned in the previous chapter, the condition number can be estimated from a matrix G . It gives a measurement of how close a matrix is to being rank deficient. The condition number $\kappa(G)$ is the product of the norm of the matrix G and the norm of its inverse. Equation 4.5 shows how to compute the exact value of the condition number. A high condition number results from an ill-conditioned matrix, while a number near to one results from a well-conditioned matrix.

$$\kappa(G) = \|G\| \|G^{-1}\| \quad (4.5)$$

The condition number can also be considered as the ratio of the largest to the smallest singular value of the matrix [20]. The SVD makes it possible to compute the exact value of the condition number. In our case, it is sufficient to approximate this value.

Following [20], an estimate of the condition for the term $\|G^{-1}\|$ can be computed by using the LU decomposition. In general, an ill-conditioned matrix G reflects its condition in the

factorised upper and lower triangular matrix.

If another given matrix H is triangular and the max-norm is used, the condition number can be estimated as denoted in Equation 4.6. We use the diagonal elements u_{jj} where $j = 1 \dots N$ of H to estimate a condition number of it. $N \times N$ represents the dimension of H .

$$\kappa(H) \geq \frac{\max_j (u_{jj})}{\min_j (u_{jj})} \quad (4.6)$$

We assume that the LU decomposition of an invertible matrix G yields two non-singular matrices L and U . Considering the condition number of the upper triangular matrix, we decide whether the matrix is singular or not. If the reciprocal value of the computed condition number is used, a well-conditioned matrix has a condition number near to 1. Ill-conditioned matrices give a reciprocal $\kappa(H)$ near to zero. This estimate may fail in some special cases, but we can verify the invertibility in a few computation steps.

4.2.3 Computing the Inverse with Cholesky Decomposition

If the square matrix G is symmetric and additionally positive definite, G has a more efficient, triangular decomposition. The positive definite property denotes that all eigenvalues of G are positive, while symmetric means that $g_{ij} = g_{ji}$ for $i, j = 1, \dots, N$. Considering the term $(\vec{c}\vec{c}^T + L)$ in the iterative smoothing scheme (Equation 4.7), L denotes a strictly diagonally dominant matrix.

$$\vec{m}^{(j+1)} = (\vec{c}\vec{c}^T + L)^{-1} (\vec{c}k + L\vec{m}^{(j)}) \quad (4.7)$$

A strictly diagonally dominant matrix H fulfils the condition $\|h_{ii}\| > \sum_{i \neq j} h_{ij}$. We can consequently assume that the term $(\vec{c}\vec{c}^T + L)$ results in a symmetric and diagonally dominant matrix. As shown in [20], these two properties enforce positive definiteness.

Both, symmetry and positive definiteness, make it possible to apply the Cholesky factorisation. In theory, this decomposition is about a factor of 2 faster than alternative methods solving linear equation systems. In LU decomposition the symmetry property is ignored.

Instead of a LU-decomposition in a lower and an upper triangular matrix L and U , the Cholesky method constructs just a lower triangular matrix L . The relation between L and L^T is given as $L_{ij}^T = L_{ji}$. The decomposition can be written as

$$A = L \cdot L^T. \quad (4.8)$$

A straightforward implementation [38] uses

$$L_{ii} = \sqrt{\left(a_{ii} - \sum_{k=1}^{i-1} L_{ik}^2\right)} \text{ and } L_{ji} = \frac{1}{L_{ii}} \left(a_{ij} - \sum_{k=1}^{i-1} L_{ik}L_{jk}\right) \quad (4.9)$$

where $j = i+1, i+2, \dots, N$, to decompose the matrix G into the Cholesky factor L . Once the matrix is decomposed, a back-substitution step makes it possible to solve a linear equation system.

We apply the Cholesky decomposition to the iterative smoothing for solving the Equation 4.7 with respect to the smoothed parameters \vec{m} .

Experiments on using SVD, LU- and Cholesky decomposition are given in Chapter 5.

4.3 Reducing the number of Inverse Computations

The application of LU- and Cholesky decomposition instead of SVD and the approximation of the reciprocal condition number, give an appreciable speed-up. However, the problem is still the large number of inverse computations, which keeps the registration time high. The following section outlines the main idea on how to apply a sub-sampling scheme to get a further speed-up of Periaswamy's approach.

4.3.1 A Sub-Sampling Scheme

In general, the idea is to reduce the number of voxels to be traversed. The ability of the suggested Periaswamy approach to register local details must not suffer from the sub-sampling scheme.

Liu et al. [25] addressed the problem of finding the trade-off between the accuracy and efficiency, respectively. They presented different approaches towards the goal of experimental studies on accuracy vs. efficiency trade-offs. The computational complexity of an optical flow algorithm is proportional to the according image size.

An intuitive idea improving the speed is to sub-sample the images. A sub-sampling runs the risk of under-sampling below the Nyquist frequency, resulting in aliasing effects. In our case, we use a low-pass filtering in each level of the multi-resolution approach, anyway.

By visiting only every second voxel in each dimension, the number of inverse computations can theoretically be reduced to an eighth of the overall number of voxels. If we use a spatial neighbourhood with a size of $5 \times 5 \times 5$, we still have an overlap between two proximate neighbourhoods.

Figure 4.1 explains the idea of the sub-sampling scheme. In the first iteration (4.1(a)), we start to estimate the parameters at position $(0,0,0)$. In an important second iteration (4.1(b)), this starting position is increased by an offset of 1 in each dimension. Displacing the position of the image iterator is necessary to avoid an overall systematic error. By means

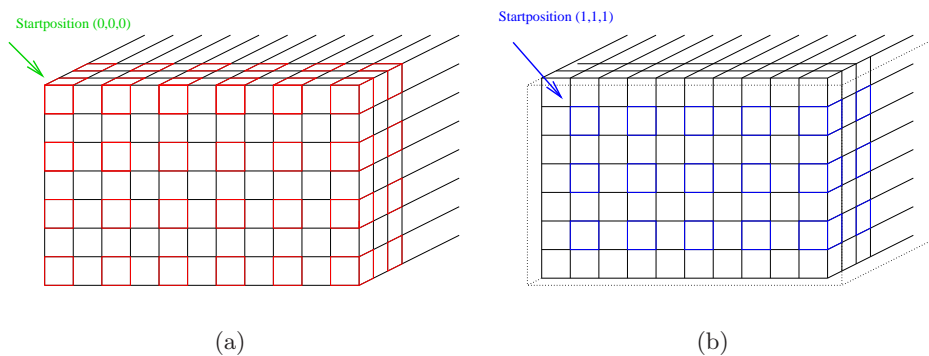


Figure 4.1: Sub-sampling the image in 3 dimensions: (a) first iteration without an offset, (b) second iteration with an offset of 1 in each dimension.

of this scheme, we can reduce the number of matrix inversions to $\frac{1}{8}$. In a second step, we need to estimate the missing parameters to preserve the originally presented transformation model in Chapter 3. Therefore, the next sub-section outlines an idea on how to interpolate the missing parameters.

4.3.2 Interpolation of Missing Parameters

The remaining, not traversed voxels have to be interpolated in its parameters. This interpolation can be performed by involving the computed parameters of the proximate neighbourhood. In case of *Model 1* and *Model 2*, which use only translation with and without intensity variation, it is intuitive how to interpolate these values. A simple translation can be linearly interpolated. The contrast and the brightness parameters represent a continuous intensity variation over the complete image. Additionally, these two values are independent of the affine transformation parameters. We apply a separate linear interpolation to achieve the missing brightness and contrast parameters, under the assumption of a continuous change of these two parameters.

From these ideas, we introduce an interpolation scheme. In each level of the pyramid, the sub-sampling method is executed on the according full image size. We apply an in-line interpolation scheme, that does not require additional memory.

Figure 4.2(a) illustrates the first step of the interpolation. The figure shows a $3 \times 3 \times 3$ extracted neighbourhood of the image structure. The blue cubes represent the locations of current computed parameters. The parameters of the red cube are interpolated from the parameters of the eight surrounding blue cubes.

In a second step, we interpolate the missing parameters of the green cubes consulting the appropriate four blue cubes in the next neighbourhood. The principle is shown in Figure 4.2(b). For the y -direction we apply an interpolation step given in Figure 4.2(c). The required values of the magenta cube are computed by an interpolation of the parameters of two blue cubes. The remaining x and z directions are proceeded in an according way. This

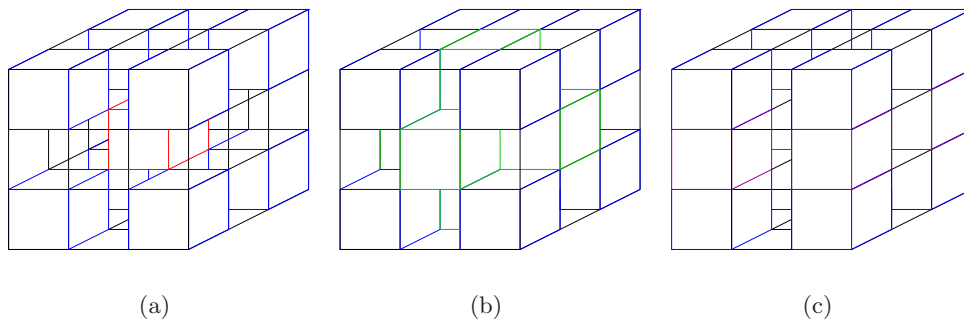


Figure 4.2: Interpolation scheme: (a) interpolation in the first iteration and (b)-(c) the according interpolation of the second step.

interpolation scheme can be performed in two steps with no additional memory requirements. The aim of this interpolation is to reduce the high time consumption of the proposed registration approach in Chapter 3.

So far, we have only highlighted the interpolation in case of *Model 1* and *Model 2*. The more complex *Model 3* and *Model 4* include affine transformations. As mentioned in the previous chapter, a raw affine matrix in 3D is defined by 9 DoF, which include parameters for shearing, scaling and rotation in any direction. For further explanations, we introduce a 3×3 matrix P_k , which includes the estimated parameters $m_1 \dots m_9$. The index k of P denotes the k -th neighbour of the current visited voxel. The matrix $P_{interpolated}$ includes the resulting interpolated affine parameters.

The interpolation of rotations is a commonly known problem in computer graphics. The decomposition of an affine transformation into scaling, shearing and rotation is outlined in Section 4.3.3. We want to decompose an affine matrix to interpolate the rotation correctly in its involved parameters. This decomposition makes it possible to compare the effect of linearly interpolated affine parameters to correct rotation-interpolated matrices.

4.3.3 Decomposing the Affine Matrix

A correct estimate of a mean orientation from a set of orientations needs a decomposition of the 3×3 matrix P_k into scaling, shearing and rotation. We use an algorithm given in [1] for the decomposition. The algorithm produces a sequence of transformations, so that a concatenation will result into the matrix G . This method is constrained to a non-zero determinant of the original matrix P_k .

P_k is decomposed by first extracting the scaling factors and then the shearing parameters, leaving a raw rotation matrix. The rotation matrix is broken down into 3 components representing the rotation in each dimension. Furthermore, we get the 3 scaling factors s_x , s_y and s_z . The shearing component consists of $shear_{xy}$, $shear_{xz}$ and $shear_{yz}$. The decomposition is outlined in Algorithm 2, where P'_i represents the i -th row and P'_{ik} the element at the position (i, k) of the matrix P' .

After pure decomposition, we can apply a linear interpolation to the scaling factors. If there are e.g. $k = 8$ surrounding voxels in the $3 \times 3 \times 3$ neighbourhood, we can compute a linear interpolation using Equation 4.10.

$$P_{sc} = \underbrace{\begin{pmatrix} s_{xr} & 0 & 0 \\ 0 & s_{yr} & 0 \\ 0 & 0 & s_{zr} \end{pmatrix}}_{\text{resulting scaling matrix}} = \frac{1}{8} \left(\underbrace{\begin{pmatrix} s_{x1} & 0 & 0 \\ 0 & s_{y1} & 0 \\ 0 & 0 & s_{z1} \end{pmatrix}}_{\text{scaling matrix k=1}} + \dots + \underbrace{\begin{pmatrix} s_{x8} & 0 & 0 \\ 0 & s_{y8} & 0 \\ 0 & 0 & s_{z8} \end{pmatrix}}_{\text{scaling matrix k=8}} \right) \quad (4.10)$$

The shearing matrices $P_{sh1} = \begin{pmatrix} 1 & 0 & 0 \\ s_{xy} & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$, $P_{sh2} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ s_{xz} & 0 & 1 \end{pmatrix}$ and

$P_{sh3} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & s_{yz} & 1 \end{pmatrix}$ are interpolated in a similar way. The idea of a linear interpolation

of translation, shearing and scale is plausible, while the interpolation of rotation is more complex. A pure rotation matrix includes parameters that are not independent. In a final

Algorithm 2 Outline of the Matrix Decomposition

 For each neighbourhood voxel of current location

1. Define a matrix $P' = P_k$
2. Compute $s_x = \|P'_1\|_2$ and normalise P'_1 by s_x
3. $\text{shear}_{xy} = P'_1 \cdot P'_2$
4. Orthogonalise P'_2 by computing $P'_2 = P'_2 - \text{shear}_{xy}P'_1$
5. Compute $s_y = \|P'_2\|_2$ and normalise P'_2 , shear_{xy} by s_y
6. $\text{shear}_{xz} = P'_1 \cdot P'_3$
7. Orthogonalise P'_3 by computing $P'_3 = P'_3 - \text{shear}_{xz}P'_1$
8. $\text{shear}_{yz} = P'_2 \cdot P'_3$
9. Orthogonalise P'_3 by computing $P'_3 = P'_3 - \text{shear}_{yz}P'_2$
10. Compute $s_z = \|P'_3\|_2$ and normalise P'_3 by s_z
11. Normalise shear_{yz} and shear_{xz} by s_z
12. Compute the determinate $\det(P')$ of P'
13. If $\det(P') = -1$, negate the matrix P' and s_x, s_y, s_z
14. Compute the rotation angle β about the y-axis by $\arcsin(-P'_{13})$
15. If $\cos(\beta) \neq 0$, compute $\alpha = \text{atan2}(P'_{23}, P'_{33})$ and $\gamma = \text{atan2}(P'_{12}, P'_{11})$
 else compute $\alpha = \text{atan2}(-P'_{31}, P'_{22})$ and $\gamma = 0$

End for

Interpolate the decomposed parameters

step, we have to estimate an interpolated rotation matrix from the neighbouring matrices. Interpolation of rotations is often referred to as a problem in computer graphics. In animation there is a need for finding a smooth in-between of animation sequences [42]. In the early days of computer graphics, there were few publications concerning the topic of interpolating rotations.

In Shoemake's work [42], the Euler angle coordinates are described as an approach to achieve interpolation. Euler angles coordinates specify orientation of three independent rotations about the x , y and z axis. In computer graphics, these orientations are often noted as "yaw", "pitch" and "roll" factors. The application of Euler angles must follow a pre-defined order, depending on the coordinate system. As a consequence, rotation in space is not commutative. For each of the Euler angles a corresponding 3×3 rotation matrix is defined. The rotation P_{rotX} about the x-axis is given by the angle α . β defines the rotation about the y-axis (P_{rotY}) and γ expresses the rotation about the z-axis (P_{rotZ}). In matrix

notation these matrices are constructed as follows:

$$P_{rotX} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{pmatrix}, P_{rotY} = \begin{pmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{pmatrix} \quad (4.11)$$

$$P_{rotZ} = \begin{pmatrix} \cos(\gamma) & -\sin(\gamma) & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (4.12)$$

The overall rotation P_{rot} is obtained by multiplying the three individual matrices in a given order. In our case we define an order for the coordinate system as x-y-z. Following this definition, we can compute P_{rot} by $P_{rot} = P_{rotX}P_{rotY}P_{rotZ}$.

In Dam et al. [11], approaches of interpolation based on quaternions, Euler angles or rotation matrices are reviewed. Quaternions use generalised complex numbers to represent rotations. The theory of quaternions is discussed in [11].

The mentioned work refers to the problem of the gimbal lock, where a specific sequence of rotations causes a loss of one degree of freedom in rotations. In the case of our work, we assume only small angle differences between the surrounding neighbour matrices, so that the occurrence of gimbal locks can be avoided.

After we have found a method to extract Euler angles, we estimate the mean angles α , β and γ each in a linear way from the neighbour angles. From this, we construct the rotation matrices by inserting the interpolated corresponding Euler angles into 4.11 and 4.12. The following section illustrates the construction of a resulting matrix including the interpolated parameters.

4.3.4 Composing the Resulting Affine Matrix

Following the description in [1], the decomposition is done in a certain sequence. The resulting interpolated affine matrix $P_{interpolated}$ is composed according to the decomposition order. Multiplying the individual matrices which represent rotations, shearings and scale leads to a combined matrix as shown in Equation 4.13.

$$P_{interpolated} = P_{sc}P_{sh1}P_{sh2}P_{sh3}P_{rotX}P_{rotY}P_{rotZ} \quad (4.13)$$

The new affine matrix $P_{interpolated}$ includes the new interpolated parameters m_1 to m_9 . Besides, the 3 translation and brightness/contrast parameters are set into the associated voxel location of the data structure.

In this section we have presented a simple method for Euler angle interpolation by decomposing and composing affine matrices. Interpolation based on Euler angles may not be definitively correct but due to assuming small rotations, we have found a comparable method to achieve a linear interpolation of affine parameters. In Chapter 5, a comparison of results on direct and decomposed linear interpolation is given.

4.4 Replacing the Iterative Smoothing

As mentioned in Chapter 2, Bro-Nielson et al. [8] proposed a method to solve the linear PDE by convolving with a specific derived filter. Applying this linear elastic filter gives a stable way of modelling fluid deformation. Thirion [46] proposed an approximated solution of this method to perform the regularisation. The determined displacements are low-pass filtered by convolving with a Gaussian filter mask.

Because of high time and memory consumption, the idea is to replace the iterative smoothing with a Gaussian filtering. Periaswamy [36] suggested a globally smooth displacement field through the combination of local deformations and a global smoothing.

The local deformations with no regularisation, result from the raw parameter computation. From this, we can extract all raw translation parameters. These parameters construct an overall deformation field, which includes a 3D displacement vector for each voxel in each dimension. We decompose the overall deformation field into three components, representing the translation in x , y and z direction. Then, a Gaussian kernel convolves the three separated fields, independently. After convolution, the smoothed translations in each dimension are combined to form one resulting deformation field. Figure 4.3 illustrates the principle of this smoothing scheme. In each dimension, the convolution is performed with a constant standard

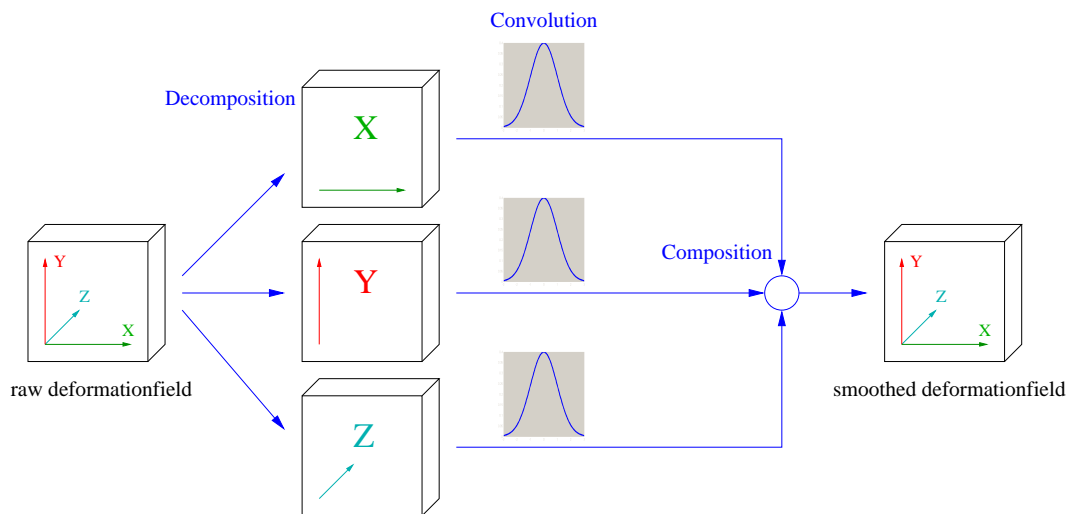


Figure 4.3: The principle of the smoothing scheme.

deviation σ , controlling the variability of the Gaussian distribution. This way of regularisation avoids the tuning of the number of iterative smoothing steps. Instead, the standard deviation of the Gaussian filter controls the global influence on the local deformations. This parameter is obviously of great importance concerning the global smoothness and accurate local registration. Chapter 5 investigates the influence of σ on the registration results.

In this section, an idea has been given on how to replace the iterative smoothing proposed in [36]. Applying a similar regularisation as presented in [46] and [8], reduces the problem of extensive time and memory consumption. The convolution with a Gaussian filter mask can be performed fast and efficiently. An additionally introduced parameter replaces the choice of the number of iterative steps.

Chapter 5

Experiments and Results

5.1 Introduction

This chapter presents the performed experiments and results. Section 5.2 gives an overview of the provided data. A description of a synthetic transformation on original data is also given in this section. In Section 5.3, the methods of evaluation are explained. The following sections illustrate the performed experiments and obtained evaluation results.

First, we compare the results of the suggested algorithm in [36] to the results of our improvements. These experiments are performed on synthetically transformed images. In Section 5.4, the results are also given for different transformation models and varying numbers of iterations. Section 5.6.2 shows the ability of aligning real data sets. Results of our implementation are compared to standard registration algorithms such as the *Demons* algorithm [46] and a B-spline based method [40], as well. Registration results on real CT images including intensity variations are given in Section 5.6.3. All experiments in this chapter are performed on a 64 bit AMD OpteronTM with 8 GByte RAM. For visualisation of difference images, we use a colour fusion representation. The colour coding is similar to the colour spectrum. Magenta to blue represents small intensity differences, whereas orange to red colours indicate large differences.

5.2 Data

In this thesis, the provided data sets are 3D and intra-modal images of soft tissue organs like lung and liver. Mainly, we evaluate our implementation on a pair of CT sheep images. These images, including the thorax, were acquired at two respiratory states. The provided raw images have a dimension of $256 \times 256 \times 256$ with a voxel dimension of $1.12 \text{ mm} \times 1.12 \text{ mm} \times 1.2 \text{ mm}$.

Figure 5.2 includes sample slices of real 3D data sets in different views for the two states of inhalation and exhalation. To reduce computation and memory consumption, these data sets are down-sampled to a dimension of $128 \times 128 \times 128$ with a voxel dimension of $2.34 \text{ mm} \times 2.34 \text{ mm} \times 2.4 \text{ mm}$. We show experimental results on both dimensions if computation is possible.

In case of real data, the evaluation of the registration can only be done on intensity difference

measurements. To get accurate information on the performance, we need a comparison of a priori known displacement to the computed. For this reason, we generate synthetic data sets with known displacement fields.

The synthetic data sets are produced by deforming the original inhalation image related to a real respiration. A non-linear transformation [49] $T : P(x, y, z) = Q(T(x, y, z))$ approximates the respiration. The movement of the diaphragm is simulated by a transformation in vertical direction, described by $t_{vertical}$. Values of 35 mm, 25 mm and 10 mm are used in this work for $t_{vertical}$. A 2D Gaussian distribution depending on x and y position determines the translation in negative z -direction. The resulting displacement vector maps each point $(x, y, z)^T$ to $(x, y, z')^T$. μ_x and μ_y correspond to the centre of gravity of the diaphragm.

$$z' = z - t_{vertical} e^{-\frac{(x-\mu_x)^2 + (y-\mu_y)^2}{2\sigma^2}} \quad (5.1)$$

The simulation of rib cage behaviour during respiration is performed by a radial, centre directed translation t_{inward} that maps each point $(x, y, z)^T$ to $(x', y', z)^T$.

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \mu_x \\ \mu_y \end{pmatrix} + t' \cdot \frac{\vec{c}}{|\vec{c}|} \quad (5.2)$$

where

$$\vec{c} = \begin{pmatrix} x - \mu_x \\ y - \mu_y \end{pmatrix} \text{ and } t' = |\vec{c}| - t_{inward} \cdot \left(1 - e^{-\frac{(x-\mu_x)^2 + (y-\mu_y)^2}{2\sigma^2}} \right) \quad (5.3)$$

The combination of vertical and inward transformation leads to a resulting displacement field T , which makes a qualitative comparison of the different registration results possible. We apply this synthetic transformation on the image representing the inhalation. Due to transformation, the resulting image shows a simulated exhalation.

Figure 5.2 displays axial sample slices of real inhalation, real exhalation and synthetically produced exhalation in case of 25 mm vertical, 10 mm inward and 35 mm vertical, 10 mm inward transformation, respectively. For convenience of further experiments, we define the names of the most often used data sets as follows:

- Data set *S10* includes a real inhalation image as moving image and the synthetic transformed inhalation image with $t_{vertical} = 10$ mm, $t_{inward} = 10$ mm as fixed image.
- Data set *S25* includes a real inhalation image as moving image and the synthetic transformed inhalation image with $t_{vertical} = 25$ mm, $t_{inward} = 10$ mm as fixed image.
- Data set *S35* includes a real inhalation image as moving image and the synthetic transformed inhalation image with $t_{vertical} = 35$ mm, $t_{inward} = 10$ mm as fixed image.
- Data set *Real* includes a real inhalation image as moving image and a real exhalation as fixed image.

In Figure 5.2, various difference fusion images restricted to the range of -1000 HU ... 100 HU are shown. Several synthetic displacement fields are displayed in Figure 5.2. These displace-

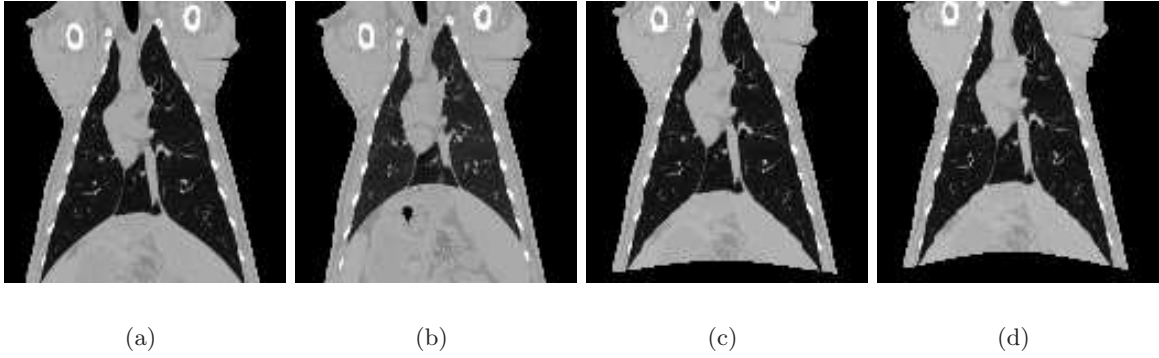


Figure 5.1: Examples of input images (sagittal view slice 128): (a) Real inhalation, (b) Real exhalation, (c) Synthetic exhalation, $t_{vertical} = 25 \text{ mm}$, $t_{inward} = 10 \text{ mm}$ and (d) Synthetic exhalation, $t_{vertical} = 35 \text{ mm}$, $t_{inward} = 10 \text{ mm}$.

ment fields are presented in a magnitude colour fashion, where the colour represents the magnitude of each displacement vector.

5.3 Evaluation Methods

This section describes the methods of evaluation. In this thesis the quality of registration is measured by means of intensity and displacement differences and normalised mutual information. Of course, we also use visual examinations as an important measurement. A comparison of the time consumption supports the evaluation of the algorithms in their registration quality.

5.3.1 Intensity Differences

A common used measurement for determining the quality of the results is the comparison of difference intensity values. This measurement is based on root mean squared error (RMSE) computations between the pair of images before and after the registration process. Additionally, we compute the standard deviation (STD) and the maximum value of the intensity differences. If f_1 and f_2 denote the two images of a data set we can compute the RMSE as follows

$$RMSE_{intensity} = \sqrt{\frac{1}{n} \sum_n (f_1 - f_2)^2} \quad (5.4)$$

where n represents the number of traversed voxels. The STD of the intensity differences can be expressed as

$$STD_{intensity} = \frac{1}{n-1} \left[\sum_n (f_1 - f_2)^2 - \frac{1}{n} \left(\sum_n (f_1 - f_2) \right)^2 \right] \quad (5.5)$$

We adopt the $RMSE_{intensity}$ computations as a common measurement to compare the pair of images before and after registration. A registration process decreases both, the value of

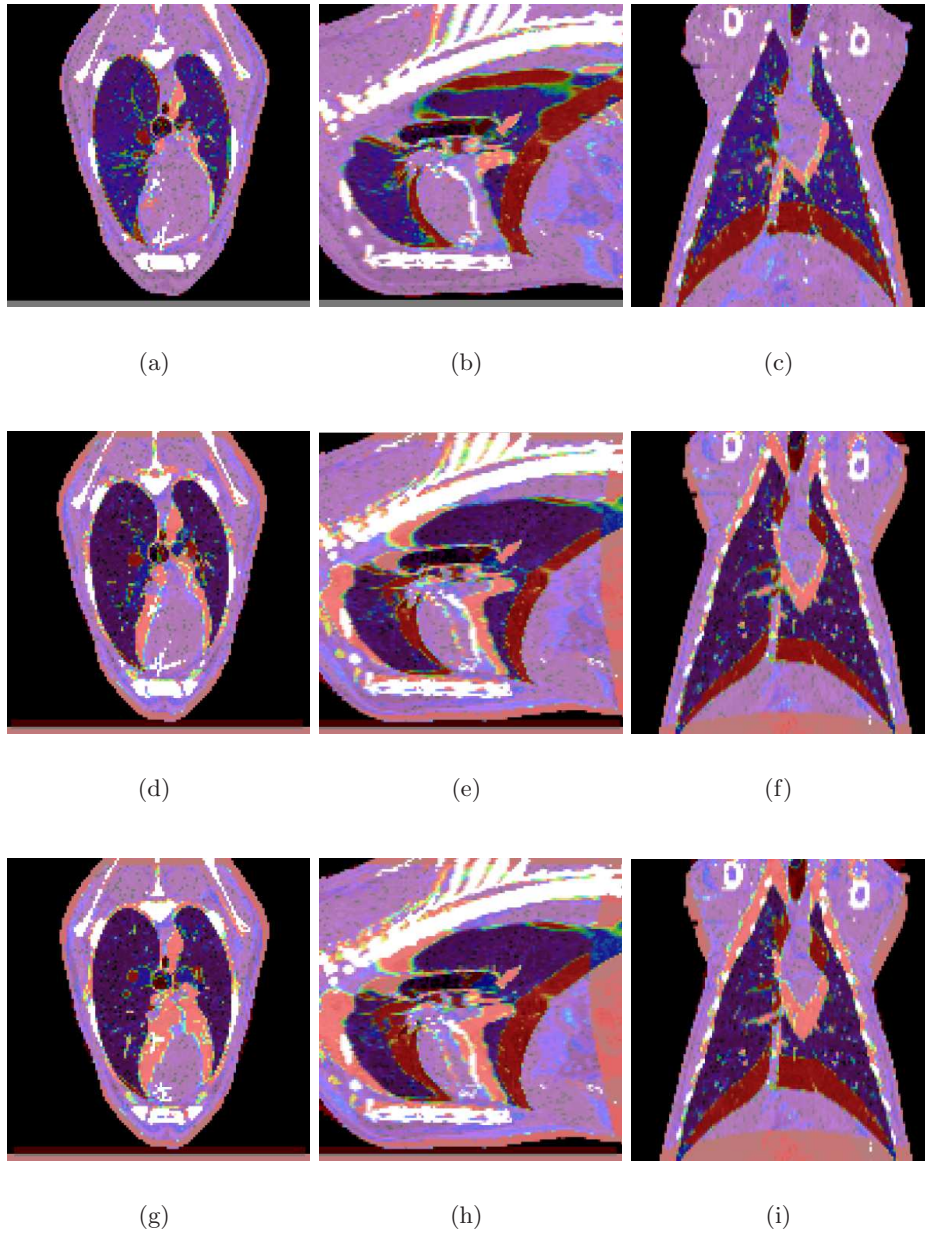


Figure 5.2: Difference fusion images: (a) real images axial, (b) real images coronal, (c) real images sagittal, (d) synthetic data set S_{25} axial, (e) synthetic data set S_{25} coronal, (f) synthetic data set S_{25} sagittal and (d) synthetic data set S_{35} axial, (e) synthetic data set S_{35} coronal, (f) Synthetic data set S_{35} sagittal.

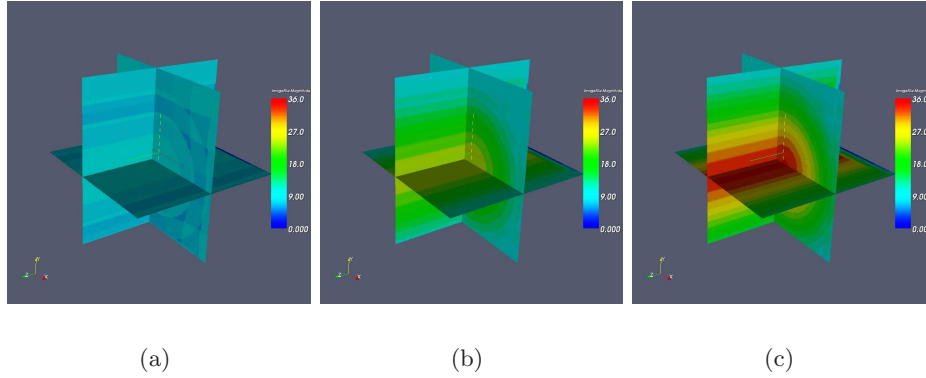


Figure 5.3: Synthetic displacement fields (a) $t_{vertical} = 10 \text{ mm}$, $t_{inward} = 10 \text{ mm}$, (b) $t_{vertical} = 25 \text{ mm}$, $t_{inward} = 10 \text{ mm}$ and (c) $t_{vertical} = 35 \text{ mm}$, $t_{inward} = 10 \text{ mm}$.

the RMSE and the STD.

5.3.2 Displacement Vector Differences

By using synthetic data, it is possible to measure the RMSE of the displacement field differences. For each voxel in the synthetic data, there exists a corresponding 3D displacement vector in the displacement field. The RMSE is calculated between the known deformation field and the resulting deformation field of registration. The RMSE is defined as

$$RMSE_{df} = \sqrt{\frac{1}{n} \sum_n \|\vec{DF}_{synthetic} - \vec{DF}_{calculated}\|^2} \quad (5.6)$$

where $\vec{DF}_{synthetic} = (dx_1, dy_1, dz_1)^T$ and $\vec{DF}_{calculated} = (dx_2, dy_2, dz_2)^T$ describe the directions of motion for each voxel and n the total number of image voxels.

5.3.3 Normalised Mutual Information

As described in Chapter 2, NMI provides an important basis for similarity measurements. We apply the NMI computation to a pair of images. Besides, this information theoretic approach can handle the evaluation of the registration results on images including intensity variations. In case of the evaluation in this work, we take 255 bins for the NMI computation. Starting from this number, the according histogram is generated by using the overall images information. The joint entropy and the two independent marginal entropies compose the NMI. In case of our evaluations, a rising of the similarity results in an ascending value of the NMI. If two image are equal, the value of the NMI is one.

5.4 Experiments on Improvements

In this section, the aim is to evaluate the suggested algorithm and our improved methods. First of all, we show the influence on time consumption by applying the LU and the Cholesky

decomposition instead of the SVD. The results on the sub-sampling method are evaluated by means of the synthetic transformation. As mentioned before, this makes it possible to compare the computed displacement vectors to the synthetic ones. Experiments on various data sets show the significant influence of decomposing the parameters in contrast to direct linear interpolation. As a last step, we give qualitative results for the replaced iterative smoothing.

5.4.1 SVD vs. LU and Cholesky decomposition

The evaluation of the time consumption using SVD or LU/Cholesky decomposition turns up to be sophisticated. Considering the algorithm as a black box leads to a poor conclusion of the resulting time consumption, because it is not possible to determine the real number of inverse computations. Therefore, we attach an additionally independent condition number check before estimating the raw parameters. Assuming that both methods provide an identical number of computed inverses, these methods can be compared directly in their time consumption.

We investigate the time consumption on data set *S25*. Both images have the dimension of $128 \times 128 \times 128$.

The first experiment is accomplished with a standard set of parameters: The multi-resolution approach consists of four levels, where the coarsest level (*Level 3*) has a dimension of $16 \times 16 \times 16$. The number of outer loops in each level is determined by using Equation 5.7 in our implementation.

$$\text{NoOfOuterLoop} = \text{current Level} \times \text{Parameter 1} + \text{Parameter 2} \quad (5.7)$$

Parameter 1 and *Parameter 2* are set equal to 1. At finest level (*Level 0*), the algorithm performs 1 iteration for transformation parameter estimation. The number of smoothing steps is set to a constant value of 10.

In Table 5.1, the obtained values of time consumption are given using the SVD and LU/Cholesky decomposition for different transformation models. The significant evaluation

Model	Time [s]	Time [s]
	SVD	LU/Cholesky
Model 1	740.46	615.95
Model 2	1283.79	1159.41
Model 3	1875.18	1290.60
Model 4	2245.01	1542.67

Table 5.1: Evaluation of the time consumption using LU/Cholesky decomposition and SVD on $128 \times 128 \times 128$ images for different transformation models.

results of the time consumption are illustrated in Figure 5.4.1. This experiment shows the obtained speed-up, if the SVD is replaced by a combination of LU/Cholesky decomposition. As described in Chapter 4, we apply the LU decomposition to estimate the raw parameters \vec{m} . The method of Cholesky decomposition is introduced to solve the inverse in case of the

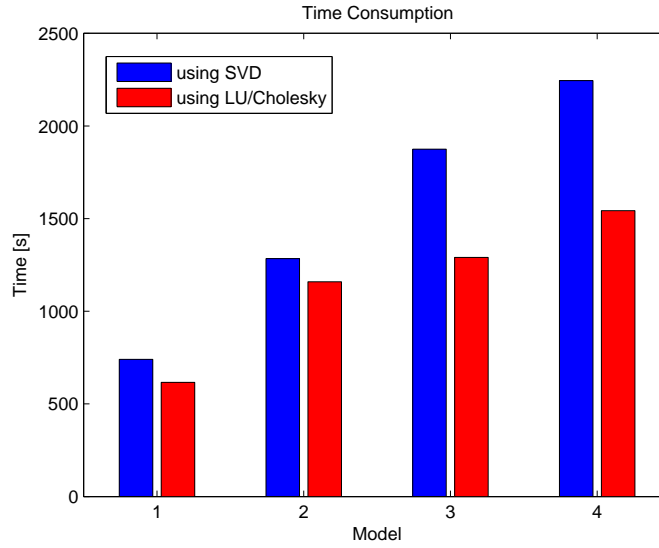


Figure 5.4: Evaluation of the time consumption using LU/Cholesky decomposition and SVD on $128 \times 128 \times 128$ images for different transformation models.

smoothing iterations.

However, this experiment shows only an improvement of time. It is not representative for the real time consumption in practise, because it is not necessary to calculate the condition number in an additionally step.

In a second experiment we compare the algorithms in their time consumption as a black box on the basis of achieved results. The results are compared by RMSE and NMI evaluations. Table 5.2 summarises the achieved computation times and gives a comparison of the similarity measurements. Figure 5.5 illustrates these evaluation values in a diagram. The

Model	Time [s] SVD	RMSE [HU] SVD	NMI	Time [s] LU/Chol.	RMSE [HU] LU/Chol.	NMI
before	-	55.224	0.243	-	55.224	0.243
Model 1	714.28	42.889	0.473	657.59	42.888	0.473
Model 2	1187.20	26.024	0.669	1141.99	25.999	0.669
Model 3	1605.43	43.215	0.465	1042.72	43.214	0.465
Model 4	1911.83	23.382	0.714	1228.49	23.380	0.714

Table 5.2: Black box evaluation of the time consumption, RMSE and NMI using LU/Cholesky decomposition and SVD on $128 \times 128 \times 128$ images for different transformation models.

Figures 5.4.1 and 5.7 display axial sample slices (slice 80) of data sets before and after registration using the transformation model *Model 1* and *Model 4*, respectively.

These experiments have shown that we obtain an acceptable speed-up in computation time with a constant quality of registration results. Considering the different models, we conclude that the models including brightness and contrast estimation, yield more accurate registration results. Furthermore, the costly matrix inversions in case of *Model 3* and *Model 4* produce a

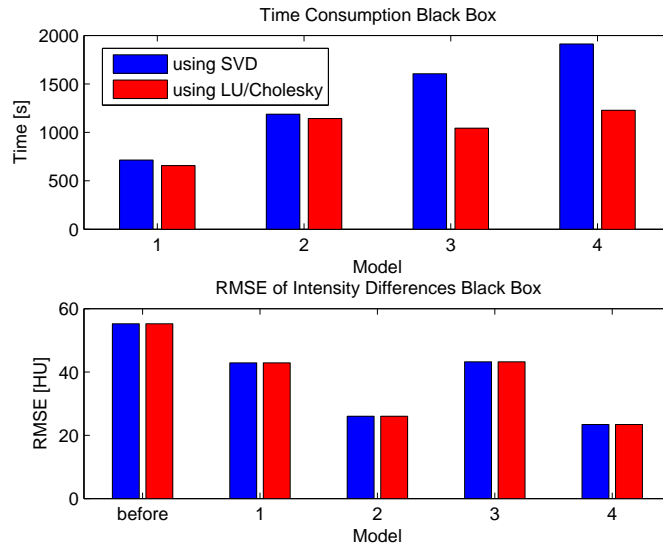


Figure 5.5: Black box evaluation of the time consumption, RMSE and NMI using LU/Cholesky decomposition and SVD on $128 \times 128 \times 128$ images for different transformation models.

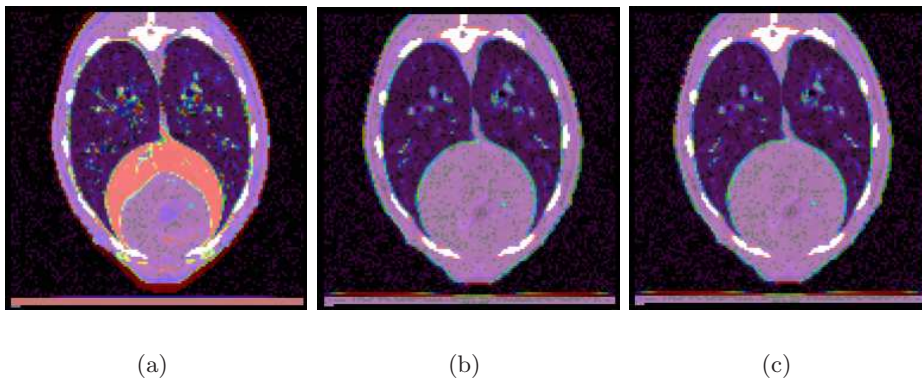


Figure 5.6: Axial sample slices of difference fusion images: (a) before Registration, (b) after Registration using *Model 1* with SVD, (c) after Registration using *Model 1* with LU/Cholesky decomposition.

strong increase of computation time.

5.4.2 Experiments on Sub-Sampling

As illustrated in the previous chapter, we apply a sub-sampling to the proposed algorithm. This time reducing improvement is evaluated on data set *S35*.

First we have to investigate the influence of correct interpolation of the parameters. We assume an Euler angle interpolation as sufficient, because there are only small angle changes in the $3 \times 3 \times 3$ neighbourhood. Then we illustrate that direct linear interpolation between the parameters will achieve similar results. Table 5.3 summarises the time consumption of the algorithms using no, decomposed and direct interpolation. The direct interpolation denotes

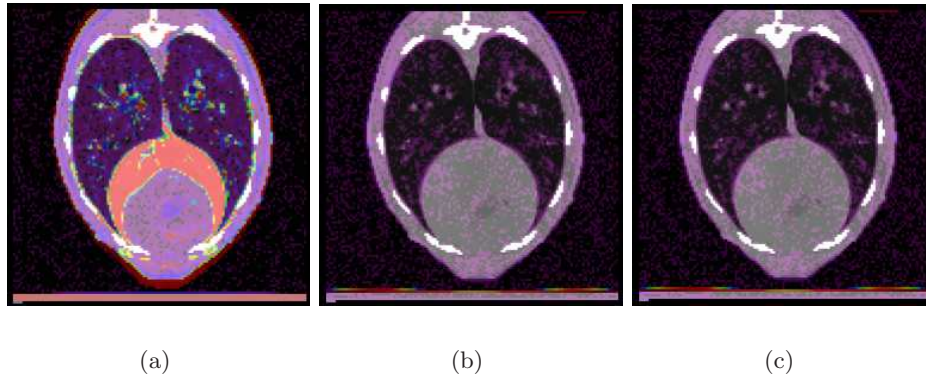


Figure 5.7: Axial sample slices of difference fusion images: (a) before Registration, (b) after Registration using *Model 4* with SVD, (c) after Registration using *Model 4* with LU/Cholesky decomposition.

an interpolation of the parameters in the neighbourhood without decomposition. To avoid a systematic error, we set *Parameter 2* to two iterations at finest level of resolution. Again, we perform all experiments using a four-level multi-resolution strategy.

Model	Time [s] no interpolation	Time [s] decomposed interpolation	Time [s] direct interpolation
Model 1	626.76	662.86	320.04
Model 2	1117.75	876.97	437.76
Model 3	1028.09	1005.24	377.45
Model 4	1235.06	922.00	387.70

Table 5.3: Evaluation of the time consumption of no, decomposed and direct interpolation on $128 \times 128 \times 128$ images for different transformation models.

Considering the plots (Figure 5.8) of the values given in Table 5.3, the time consumption of no interpolation is similar to the decomposed interpolation. Assessing the performed similarity measurements for RMSE and NMI shown in Table 5.4, we conclude that direct interpolation gives an acceptable speed-up for all transformation models without a significant loss of registration quality. The plot shown in Figure 5.9, represents the direct comparison of the computed similarity measurements. Figure 5.10 illustrates the visual registration results for *Model 2* using no, decomposed and direct interpolation. The performed experiments in this section have shown that we can reduce computation time by applying a direct interpolation of transformation parameter. An Euler angle interpolation by a decomposition achieves similar results as direct interpolation. This sub-sampling scheme introduces a negligible loss of quality in the registration results.

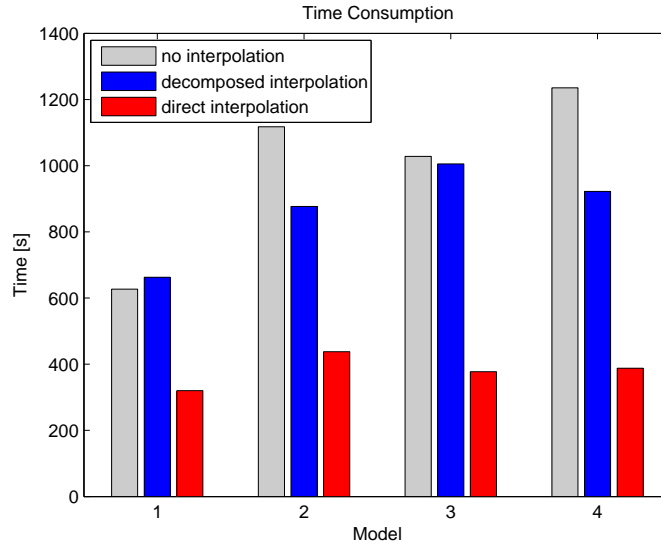


Figure 5.8: Diagram of the time consumption of no, decomposed and direct interpolation on $128 \times 128 \times 128$ images for different transformation models.

Model	RMSE [HU]	NMI	RMSE [HU]	NMI	RMSE [HU]	NMI
	no interpolation		decomposed interpolation		direct interpolation	
before	56.719	0.222	56.719	0.222	56.719	0.222
Model 1	43.245	0.473	42.132	0.479	42.132	0.479
Model 2	26.881	0.658	28.358	0.639	28.358	0.639
Model 3	43.627	0.464	42.213	0.475	42.213	0.475
Model 4	24.317	0.701	27.020	0.664	27.009	0.664

Table 5.4: RMSE and NMI similarity measurements of no, decomposed and direct interpolation on $128 \times 128 \times 128$ images for different transformation models.

5.4.3 Experiments on Replacing the Iterative Smoothing

This section highlights the results on replacing the iterative smoothing. Concluding the previous sections, the iterative smoothing suffers from high computation time and requires a large amount of memory. As explained in Chapter 4, this smoothing is replaced by a simple Gaussian smoothing procedure.

The first experiment compares the obtained computation times of the iterative smoothing to the results of using a Gaussian smoothing instead. Additionally, the times are given with and without our presented sub-sampling method. In case of replacing the iterative smoothing by Gaussian convolution, it is sufficient to interpolate only the translation values in each dimension, which involves an additional speed-up.

For this experiment we apply a $\sigma = 2.5$ to the Gaussian smoothing procedure. All experiments in this section are performed on data set *S35*. The number of outer loops in each level is computed according to Equation 5.7 with *Parameter 1* = 1 and *Parameter 2* = 2.

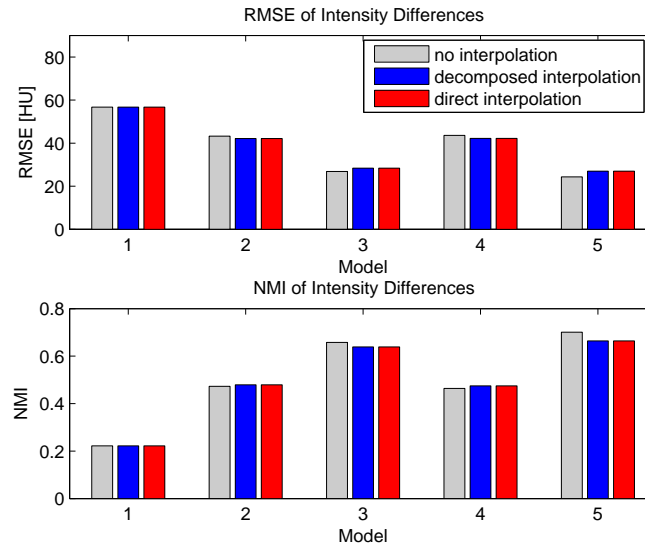


Figure 5.9: Diagram of the computed RMSE and NMI similarity measurements of no, decomposed and direct interpolation on $128 \times 128 \times 128$ images for different transformation models.

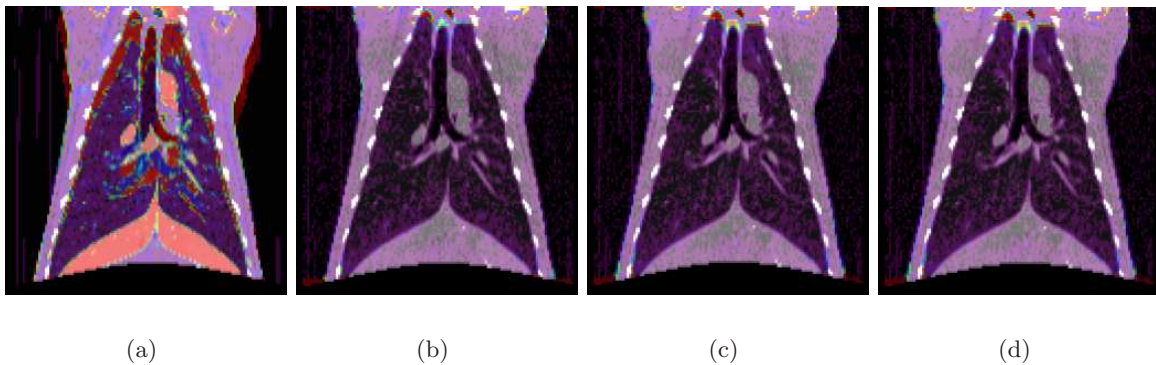


Figure 5.10: Sagittal sample slices of difference fusion images: (a) before Registration, (b) after Registration using *Model 2* without interpolation, (c) after Registration using *Model 2* with decomposed interpolation and (d) after Registration using *Model 2* with direct interpolation.

The results of the time measurement are given in Table 5.5. Table 5.6 expresses the according similarity measurements before and after registration in terms of RMSE. In Figure 5.11 these values are shown in demonstrative charts. In a second experiment, we study the influence of varying Gaussian distributions on the results. These results are compared to registration results using an iterative smoothing without interpolation by means of RMSE and NMI. Considering *Model 2*, the result on RMSE of intensity differences and respectively the RMSE of displacement differences are illustrated in Figure 5.12. The according values altogether are given in Table 5.7 for *Model 2*.

A visualisation of sample slices of the different resulting displacement fields for *Model 2* using various σ in comparison to the synthetic displacement field are displayed in Figure 5.13.

Model	Time [s]	Time [s]	Time [s]	Time [s]
	iterative smoothing, no interpolation	iterative smoothing, direct interpolation	Gaussian smoothing, no interpolation	Gaussian smoothing, direct interpolation
Model 1	626.76	662.86	174.47	108.27
Model 2	1117.75	876.97	215.394	122.13
Model 3	1028.09	1005.24	456.405	142.79
Model 4	1235.06	922.00	555.597	155.529

Table 5.5: Evaluation of the time consumption of iterative and Gaussian smoothing on $128 \times 128 \times 128$ images for different transformation models.

Model	RMSE [HU]	RMSE [HU]	RMSE [HU]	RMSE [HU]
	iterative smoothing, no interpolation	iterative smoothing, direct interpolation	Gaussian smoothing, no interpolation	Gaussian smoothing, direct interpolation
before	56.719	56.719	56.719	56.719
Model 1	43.245	42.132	45.738	45.444
Model 2	26.881	28.358	35.473	33.703
Model 3	43.627	42.213	46.107	46.013
Model 4	24.317	27.020	34.480	32.575

Table 5.6: RMSE similarity measurements of iterative and Gaussian smoothing on $128 \times 128 \times 128$ images for different transformation models.

Figure 5.14 shows resulting sample slices before and after registration using *Model 2*. To summarise these experiments, we can say that we achieve an enormous speed-up in computation time by replacing the iterative smoothing. A choice of $\sigma = 3 \dots 5$ achieves acceptable registration results. The simple Gaussian smoothing introduces an observable smearing effect at intensity gradients, because all locations of the deformation field are smoothed equally.

5.5 Experiments on Number of Iterations

Intuitively, the choice of number of iterations affects the computation time and the quality of registration. In this section we therefore perform several experiments varying the number of iterations.

The experiments are performed on synthetic data using data set *S35* with a four level multi-resolution strategy. The first experiment demonstrates the influence on computation time and RMSE for different numbers of smoothing iterations. Additionally, the RMSE between a synthetic and a resulting displacement field is computed. In this experiment we use the transformation model *Model 3* and one iteration at finest level of resolution. Additionally, we apply our suggested sub-sampling scheme. Table 5.8 includes the results of this experiment. Figure 5.15(a) illustrates a graphical data interpretation. The first experiment

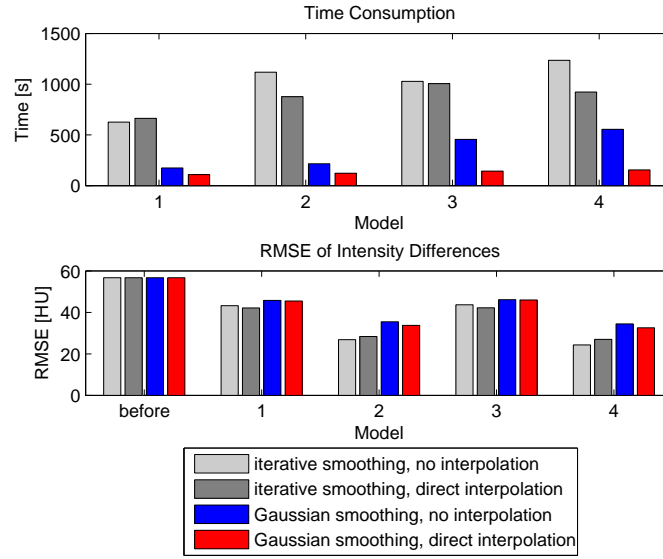


Figure 5.11: Time consumption of iterative and Gaussian smoothing on $128 \times 128 \times 128$ images for different transformation models.

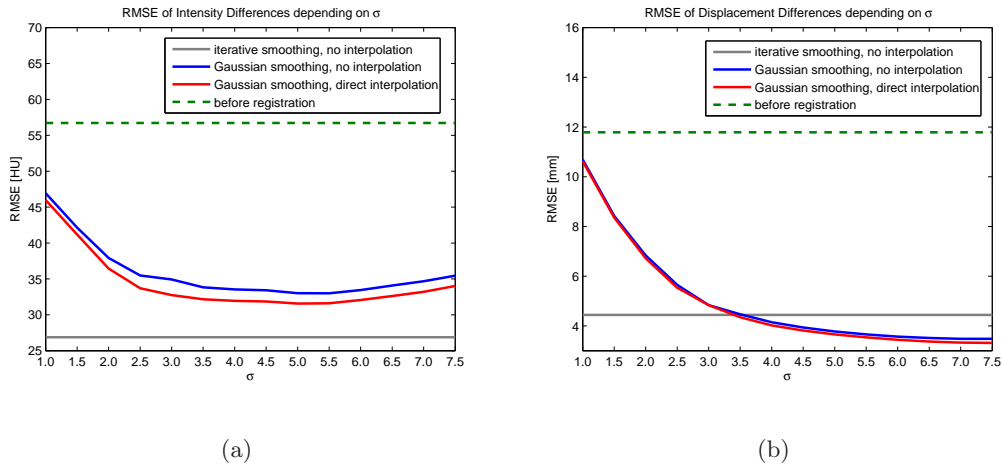


Figure 5.12: RMSE similarity measurements of iterative and Gaussian smoothing on $128 \times 128 \times 128$ images with varying σ (a) RMSE of intensity differences and (b) RMSE of displacement differences.

exemplifies that the number of smoothing iterations does not increase the computation time dramatically, because there is no need to compute inverses in each smoothing iteration. On the other hand, the prior computed inverses have to be stored in a data structure, which demands high memory requirements. This experiment illustrates that 10 to 20 smoothing iterations give the best registration results. We have to choose a trade-off between the accuracy and time consumption of the registration process.

The second experiment compares the results of different numbers of iterations at the finest level of resolution provided by the multi-resolution approach. Following Equation 5.7, the

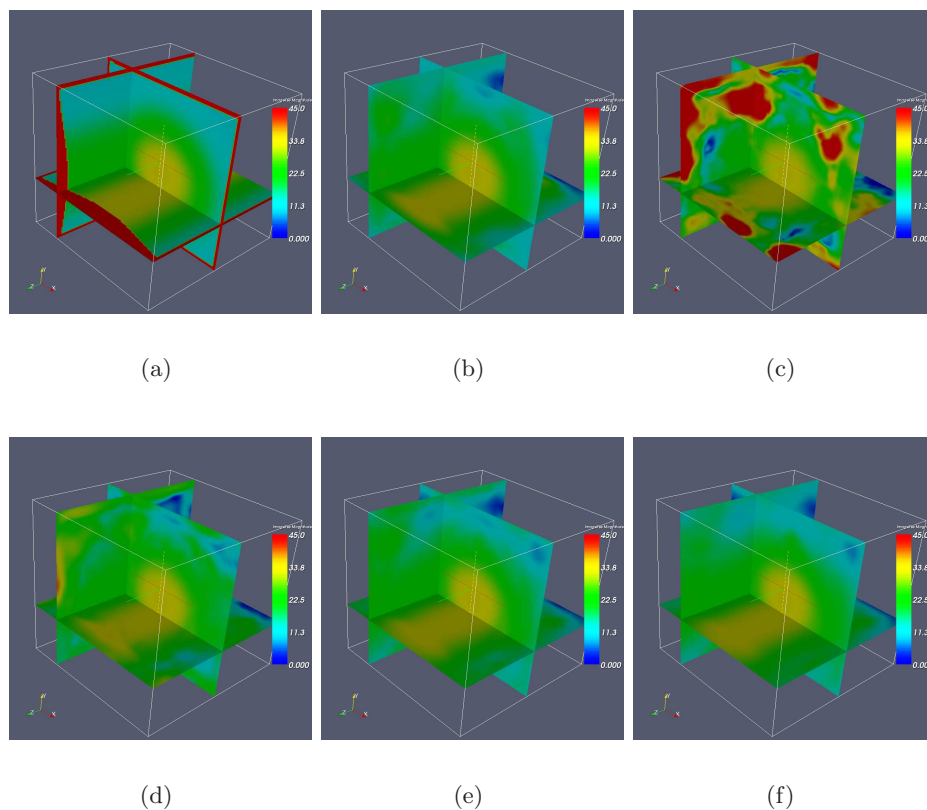


Figure 5.13: Sample slices of resulting displacement fields for *Model 2* using various σ , (a) synthetic, (b) original Periaswamy, and Gaussian smoothing with (c) $\sigma = 1.0$, (d) $\sigma = 2.0$, (e) $\sigma = 3.0$ and (f) $\sigma = 4.0$.

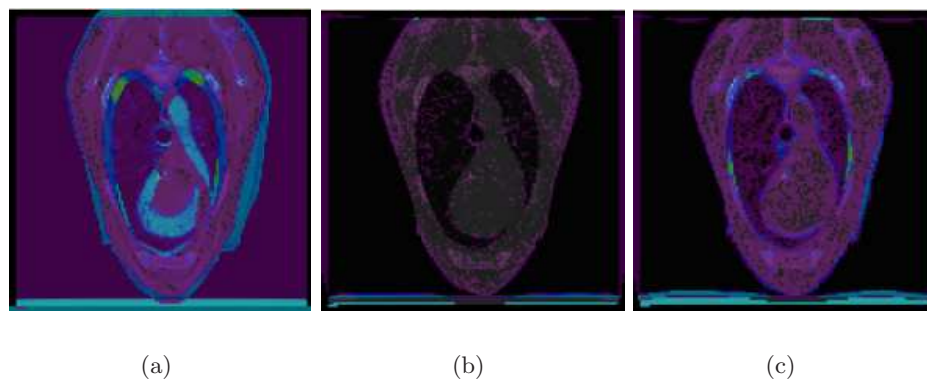


Figure 5.14: Axial sample slices of resulting difference images using *Model 2*: (a) before registration, (b) after registration with original Periaswamy and (c) a method using the Gaussian smoothing ($\sigma = 3.0$).

Model	RMSE [HU]	RMSE [mm]	σ	RMSE [HU]	RMSE [HU]	RMSE [mm]	RMSE [mm]
	it. sg., no int.			G. sg., no int.	G. sg. dir. int.	G. sg. no int.	G. sg dir. int.
before	56.719	11.785		56.719	56.719	11.785	11.785
Model 2	26.881	4.443	1.0	46.928	45.979	10.689	10.621
	26.881	4.443	1.5	42.136	41.181	8.424	8.355
	26.881	4.443	2.0	37.919	36.440	6.834	6.718
	26.881	4.443	2.5	35.473	33.703	5.647	5.532
	26.881	4.443	3.0	34.919	32.755	4.834	4.835
	26.881	4.443	3.5	33.826	32.169	4.466	4.347
	26.881	4.443	4.0	33.533	31.940	4.146	4.020
	26.881	4.443	4.5	33.425	31.857	3.933	3.811
	26.881	4.443	5.0	33.010	31.570	3.773	3.658
	26.881	4.443	5.5	32.995	31.609	3.658	3.538
	26.881	4.443	6.0	33.445	32.066	3.568	3.440
	26.881	4.443	6.5	34.072	32.614	3.512	3.371
	26.881	4.443	7.0	34.681	33.196	3.481	3.326
	26.881	4.443	7.5	35.455	34.023	3.481	3.314

Table 5.7: RMSE similarity measurements of iterative and Gaussian smoothing on $128 \times 128 \times 128$ images with varying σ for *Model 2*.

Number of smoothing Iterations	Time [s]	RMSE [HU]	RMSE [mm]
before	0	56.719	11.785
1	191.82	53.918	11.482
5	264.32	46.191	6.798
10	361.17	42.213	4.931
15	460.04	39.760	4.098
20	563.44	38.193	3.340
25	656.51	37.598	3.632

Table 5.8: Evaluation of time consumption depending on the number of smoothing iterations.

algorithm performs the number of iterations given by *Parameter 2* at finest *Level 0*. We use 10 smoothing iterations for the transformation model *Model 2* and the direct interpolation sub-sampling scheme. The values of the resulting computation times, the RMSE for intensity and displacement difference computation are given in Table 5.9. The chart, shown in Figure 5.15(b), depicts the computed values. The second experiment shows, that the number of outer loops accounts for high computation time. A choice of 3 or 4 loops reduces the RMSE of the intensity differences in fact, nevertheless the error of the displacement differences increases which results from the warping of the displacement field after each outer loop with a fast linear interpolation.

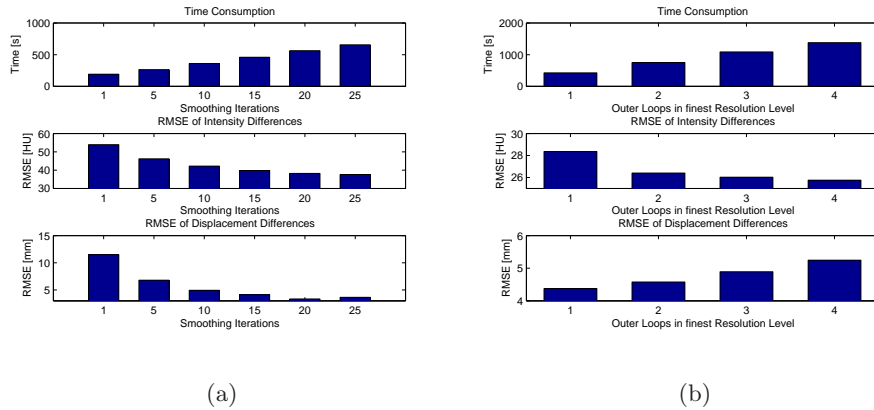


Figure 5.15: Computation times, RMSE and NMI similarity measurements on experiments (a) using *Model 3* on $128 \times 128 \times 128$ images for varying the number of smoothing iterations and (b) using *Model 2* on $128 \times 128 \times 128$ images for varying the number of outer loops.

Number of Outer Loops in finest Level	Time [s]	RMSE [HU]	RMSE [mm]
before	-	56.718	11.785
1	424.51	28.358	4.374
2	750.85	26.396	4.577
3	1085.84	26.022	4.892
4	1379.83	25.752	5.246

Table 5.9: Computation times, RMSE and NMI similarity measurements on experiments using *Model 2* on $128 \times 128 \times 128$ images for varying the number of outer loops.

In a third experiment, the convergence behaviour of the proposed algorithm is illustrated. Therefore, we compute on each iteration and level a normalised sum of the estimated displacements vectors of the current displacement field. Equation 5.7 yields the number of iterations in each according level. The computation of this measurement is expressed in Equation 5.8, where n denotes the number of visited voxels according to the current level and DF_i is the displacing vector in each 3D direction.

$$sum_{current} = \frac{1}{n} \sum_{i=1}^n \|DF_i\|_2 \quad (5.8)$$

Again, 10 smoothing iterations are used for regularisation. Table 5.10 contains the resulting values of the convergence behaviour for two and four iterations at finest level of resolution. The number of corresponding iterations of coarser level are determined using Equation 5.7. Figure 5.16 shows these values in a plot.

The third experiment demonstrates the convergence behaviour of the displacement fields. In the optimal case, the values converge to zero in each level, while in the real case the boundary effects avoid the decrease to zero.

Iterations	2 iterations at finest Level 0				4 iterations at finest Level 0			
	Level 3	Level 2	Level 1	Level 0	Level 3	Level 2	Level 1	Level 0
1	5.637	1.937	0.514	0.151	5.637	2.034	0.503	0.141
2	1.867	0.895	0.367	0.084	1.867	0.944	0.385	0.084
3	1.035	0.758	0.266	-	1.035	0.776	0.294	0.056
4	0.848	0.701	-	-	0.848	0.707	0.271	0.043
5	0.746	-	-	-	0.746	0.672	0.220	-
6	-	-	-	-	0.673	0.648	-	-
7	-	-	-	-	0.620	-	-	-

Table 5.10: Convergence behaviour of the sum of displacements in each iteration and level of the registration process.

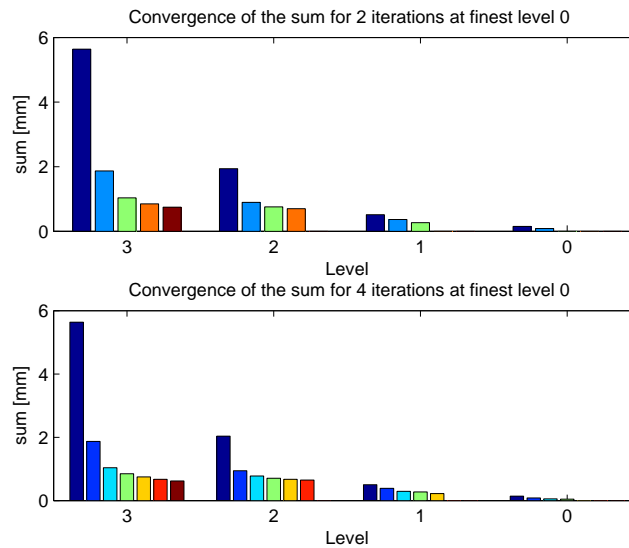


Figure 5.16: Convergence behaviour of the sum of displacements in each iteration and level of the registration process.

5.6 The Comparison

So far, the evaluations have demonstrated how to improve the Periaswamy approach. In this section we compare the suggested approach to our improvements by using various data sets and different set-ups of parameters. The performance evaluation includes time, NMI, intensity and displacement difference measurements. In order to get a feeling of the behaviour compared to standard registration algorithms, we include the described B-Spline registration method (Appendix C) and the standard *Demons* algorithm in the comparison. To additionally reduce computation time and to avoid re-sampling interpolation effects, we assume that it is sufficient to perform a constant number of outer loops at each level. We evaluate the results on the following methods and parameters as given in 5.11. The following sections describe the results on synthetic and real data for these methods.

Name	Method	Model	Number of outer loops	Number of smoothing iterations	Number of levels
PO1	original Periaswamy	4	2	15	4
PO2	original Periaswamy	2	2	15	3
PI1	with sub-sampling	2	2	15	3
PI2	with sub-sampling	4	2	10	4
PG1	with Gaussian smoothing	2	1	$\sigma = 3.0$	3
PG2	with Gaussian smoothing	2	1	$\sigma = 4.5$	4
BS1	6591 parameters	-	-	-	2
BS2	10125 parameters	-	-	-	3
Demons1	50/20 iterations	-	-	-	4
Demons2	80/30 iterations	-	-	-	4

Table 5.11: Methods and their parameters of comparison.

5.6.1 Results on Synthetic Data

The first experiments evaluate the various methods (given in Table 5.11) on *S10* with a dimension of $128 \times 128 \times 128$. Table 5.12 includes the achieved evaluation results. Figure 5.17 shows plots for a variety of these results. Figure 5.18 compares the visually obtained results

Method	Time [s]	RMSE [HU]	STD [HU]	max. Int. [HU]	RMSE [mm]	max. DF-Vec. [mm]	NMI
before		52.133	40.604	4095	4.505	10.000	0.284
P01	2618.67	14.276	13.787	3902	2.023	17.568	0.802
P02	1160.83	19.188	18.302	2257	1.704	10.500	0.750
PI1	655.02	19.215	18.229	2320	1.610	9.787	0.739
PI2	615.00	16.961	16.268	2730	2.497	25.172	0.780
PG1	64.75	21.852	20.546	1965	2.194	16.571	0.706
PG2	167.89	20.558	19.497	2566	2.673	20.978	0.743
BS1	785.00	31.624	27.857	2678	2.668	9.946	0.616
BS2	2638.00	33.762	29.450	2831	2.766	9.946	0.600
Demons1	52.07	9.558	9.231	1686	1.707	15.100	0.873
Demons2	79.00	8.970	8.694	1807	1.385	10.617	0.889

Table 5.12: A comparison of different registration methods by time consumption and similarity measurements on data set *S10* with a dimension of $128 \times 128 \times 128$.

before and after registration. The displacement fields are shown in a magnitude representation. By means of the *top* command, we observe the memory consumption during runtime. The original Periaswamy’s approach (*PO1* and *PO2*) reaches a memory consumption of more than 5 *GByte*. Therefore, we abstain from an evaluation of $256 \times 256 \times 256$ data sets for these two methods. The memory consumptions of the remaining methods are below 2 *GByte*.

In a second experiment, the methods, denoted in Table 5.11, are evaluated on the *S25* data

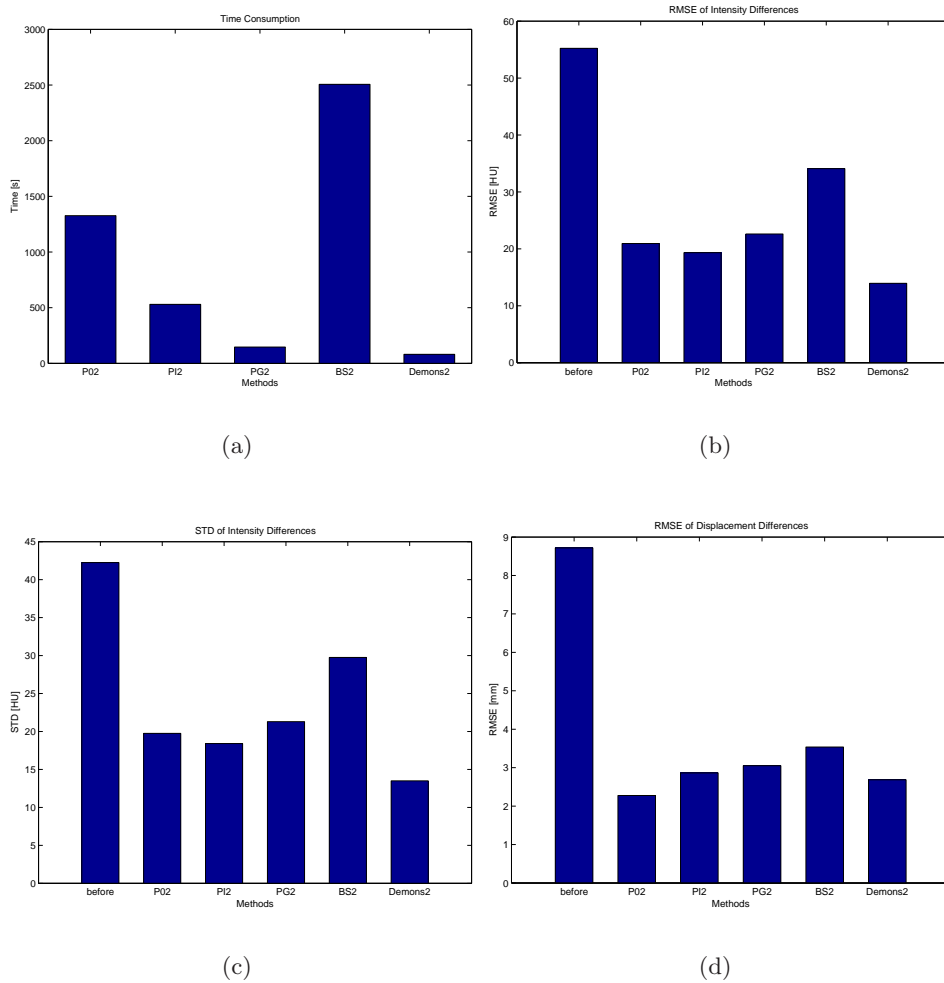


Figure 5.17: Evaluation results on $128 \times 128 \times 128$ *S10* data set: (a) time consumption, (b) RMSE of intensity differences, (c) STD of intensity differences and (d) RMSE of displacement field differences.

set, again the images have a dimension of $128 \times 128 \times 128$. Table 5.13 contains the achieved values. The representative results, such as time consumption, intensity and displacement field difference measurements are summarised in Figure 5.19. Figure 5.20 illustrates the visual results before and after registration by means of axial viewed difference fusion images and sample slices of the displacement fields in magnitude representation. Considering the similarity measurements before and after registration, the experimental results on synthetic data have shown the ability to register the synthetic data sets. The *Demons* algorithm performs an accurate and fast registration. The locally affine based methods with and without improvements reach similar acceptable results with an increased time consumption. The registration technique using FFD with a different number of parameters for optimisation needs a high computation time and yields a less accurate registration result.

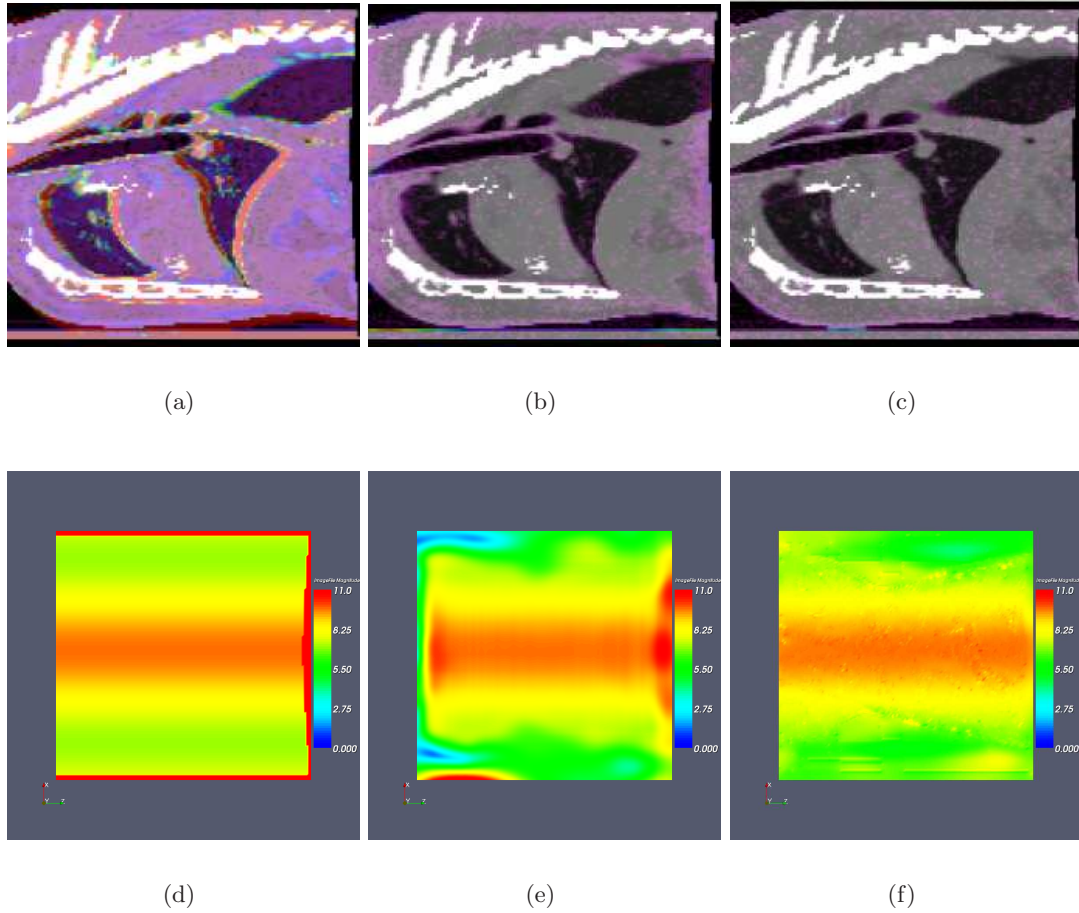


Figure 5.18: Visual results on $128 \times 128 \times 128$ *S10* data set: coronal sample slices of the difference fusion images (a) before registration, (b) after registration with method *P01*, (c) after registration with method *Demons1* and according slices of (d) the synthetic, (e) the computed *P01* and (f) the *Demons1* displacement field.

5.6.2 Results on Real Data

The previous section has given several results on data sets including a synthetic transformation. In the following section, we demonstrate the performance of several methods on real data sets. The data set *Real* with the dimensions $128 \times 128 \times 128$ and $256 \times 256 \times 256$ is applied to the registration process.

Table 5.14 displays the evaluation results on the $128 \times 128 \times 128$ data set, while Table 5.15 gives the values for the images with a dimension of $256 \times 256 \times 256$. In Figure 5.21, the computation times are given in a combined plot. Comparisons on RMSE of intensity difference and NMI computations for a choice of algorithms are given in Figure 5.22. In Figure 5.23 the moving, the fixed and the warped images in axial, coronal and sagittal views with the according sample slice of the displacement are illustrated. Figure 5.24 shows a resulting displacement field computed with method *P11*. Sagittal sample slices are shown in Figure 5.25. This figure illustrates the visual registration results before and after registration for

Method	Time [s]	RMSE [HU]	STD [HU]	max Int. [HU]	RMSE [mm]	max. DF-Vec. [mm]	NMI
before	-	55.223	42.242	4095	8.718	25.000	0.242
P01	2263.66	16.795	16.114	3674	2.430	23.986	0.774
P02	1326.90	20.933	19.750	2647	2.276	23.978	0.707
PI1	529.66	21.526	20.175	2797	2.149	23.080	0.695
PI2	530.37	19.336	18.407	2985	2.864	24.877	0.750
PG1	61.16	26.444	24.054	2451	3.096	24.033	0.613
PG2	146.61	22.612	21.301	3239	3.052	22.221	0.719
BS1	799.00	32.259	28.245	2556	3.652	27.251	0.608
BS2	2506.00	34.097	29.753	2556	3.538	27.251	0.617
Demons1	52.39	13.835	13.347	2985	2.813	13.927	0.855
Demons2	81.12	13.954	13.497	2984	2.685	12.431	0.865

Table 5.13: A comparison of different registration methods by time consumption and similarity measurements on data set *S25* with a dimension of $128 \times 128 \times 128$.

Method	Time [s]	RMSE [HU]	STD [HU]	max Int. [HU]	NMI
before	-	48.967	39.526	3085	0.360
PO1	2655.84	44.823	36.543	2467	0.451
PO2	1917.57	44.588	36.403	2390	0.446
PI1	642.03	44.898	36.636	2428	0.437
PI2	606.60	44.776	36.510	2356	0.447
PG1	66.53	46.184	37.483	4195	0.424
PG2	167.18	45.753	37.155	4195	0.437
BS1	1134.00	52.066	40.208	4195	0.385
BS2	3266.00	54.741	41.364	4195	0.376
Demons1	52.75	46.972	38.542	4195	0.473
Demons2	80.75	47.106	38.623	4195	0.474

Table 5.14: A comparison of different registration methods by time consumption and similarity measurements on data set *Real* with a dimension of $128 \times 128 \times 128$.

different methods on the $256 \times 256 \times 256$ data set. These experiments have compared the accuracy and time consumption of registration on real sheep data sets. Disregarding the time consumption, the various methods decrease the RMSE and the STD of the intensity differences. The B-spline approach performs slightly worse. Considering the NMI measurement, the registration processes obtain an expected increase of these values. Due to the high memory consumption, the methods *PO1* and *PO2* are not computed on the $256 \times 256 \times 256$ data set. We estimate the memory consumption on more than 30 *GByte*. The estimated time consumption lies within the range of *BS2*.

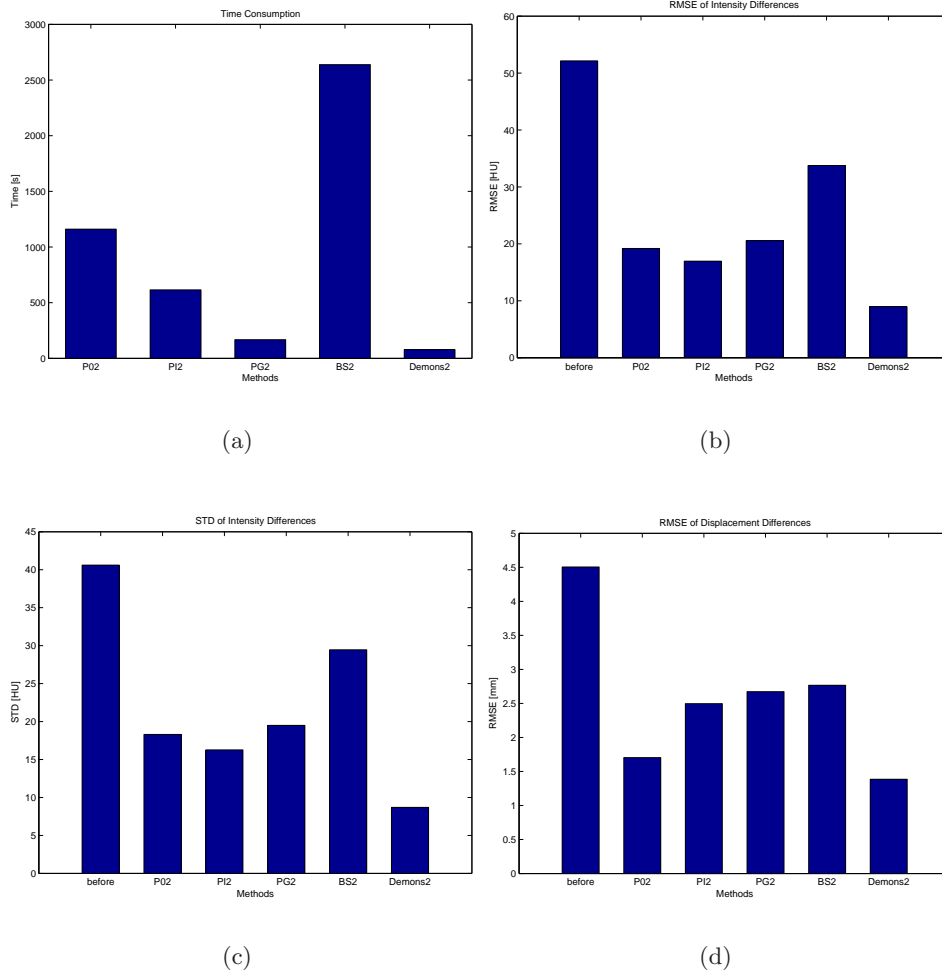


Figure 5.19: Evaluation results on $128 \times 128 \times 128$ S25 data set: (a) time consumption, (b) RMSE of intensity differences, (c) STD of intensity differences and (d) RMSE of displacement field differences.

5.6.3 Results on Real Data including Intensity Variations

In this section, real data sets with intensity variations are used as input for the various methods defined in Table 5.11. The images including contrast agents are used as fixed image, the native scan as moving image, respectively. We perform the evaluations on two data sets (data sets *Intensity5* and *Intensity2*). The provided raw images have a dimension of $256 \times 256 \times 256$ with a voxel dimension of $1.25 \text{ mm} \times 1.25 \text{ mm} \times 1.29 \text{ mm}$ and $1.1 \text{ mm} \times 1.1 \text{ mm} \times 1.2 \text{ mm}$, respectively. Both data sets are down-sampled to a dimension of $128 \times 128 \times 128$. The difference fusion images in the first row of Figure 5.26 illustrate the data set *Intensity5* before registration in axial, coronal and sagittal views with a dimension of $128 \times 128 \times 128$. This data set includes a large complex deformation.

The remaining difference images present the registration result after the registration process using the methods *PI1* and *Demons1*. Figure 5.27 contains the time consumption and NMI chart comparison for a choice of several methods. Table 5.16 summarises the evaluation

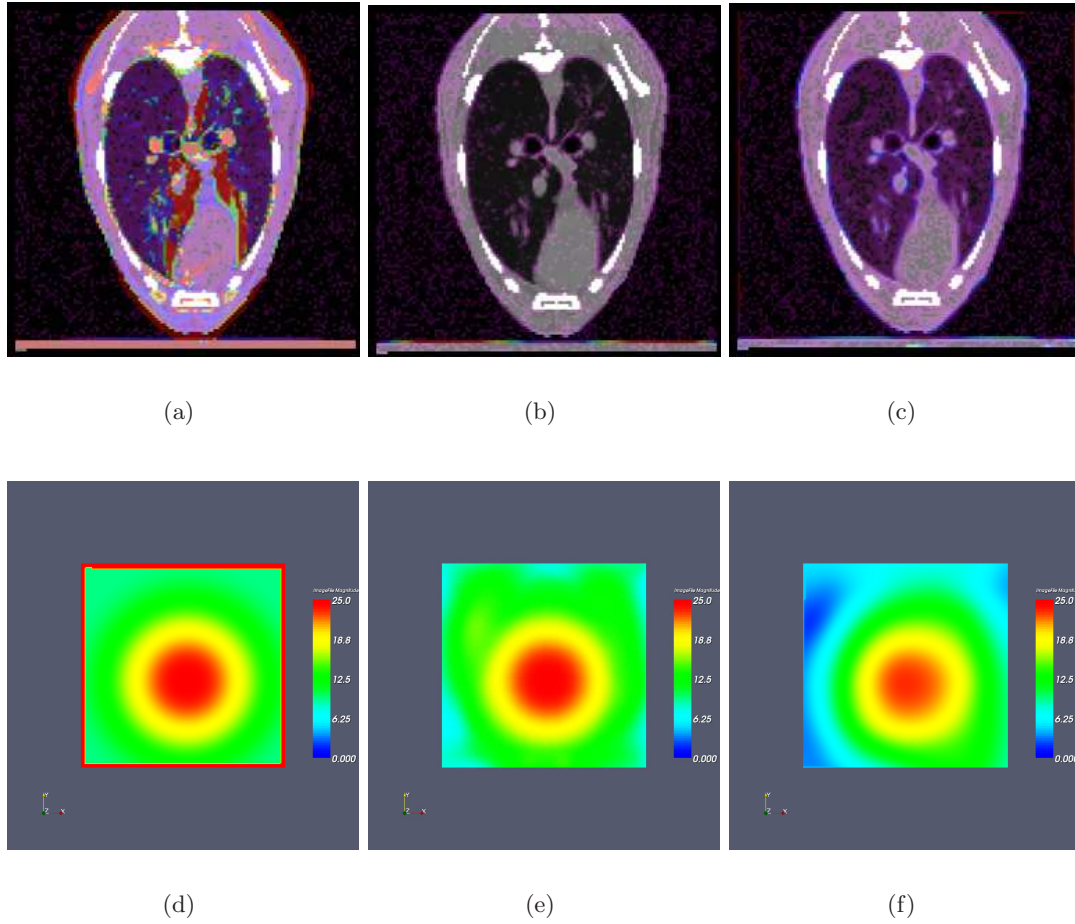


Figure 5.20: Visual results on $128 \times 128 \times 128$ *S25* data set: axial sample slices of the difference fusion images (a) before registration, (b) after registration with method *P01*, (c) after registration with method *BS1* and according slices of (d) the synthetic, (e) the computed *P01* and (f) the *BS1* displacement field.

results for all given methods in Table 5.11.

The second experiment evaluates the given methods on data set *Intensity2* with a dimension of $128 \times 128 \times 128$ and $256 \times 256 \times 256$. The first column in Figure 5.28 presents sample slices in coronal and sagittal view of this data set in image fusion difference representation before registration. Again, the contrast enhanced image serves as fixed image. The images in the second and third row represent the registration results using the methods *PI1* and *Demons1* on the $256 \times 256 \times 256$ data set. Table 5.17 includes the evaluation results on data set *Intensity2* for a dimension $128 \times 128 \times 128$ and $256 \times 256 \times 256$. Figure 5.29 directly compares the time consumption and NMI measurements for the choice of several methods on both dimensions. Figure 5.30 presents two computed displacement fields with the provided data sets *Intensity2* and *Intensity5*, respectively. In this section, the experiments have highlighted the registration of images including intensity variations. We achieve an acceptable decrease of the RMSE and an increase of NMI on both data sets. The visual results show the ability of registration of the rib cage motion. After registration, the structures within the lungs still include high

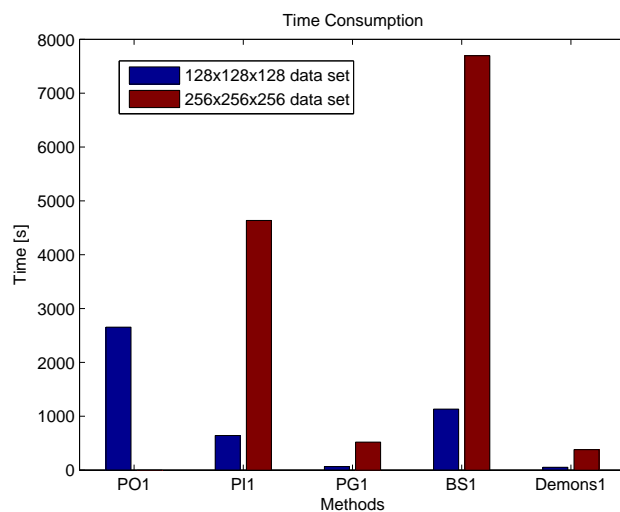


Figure 5.21: Time consumption on the $128 \times 128 \times 128$ and the $256 \times 256 \times 256$ *Real* data set.

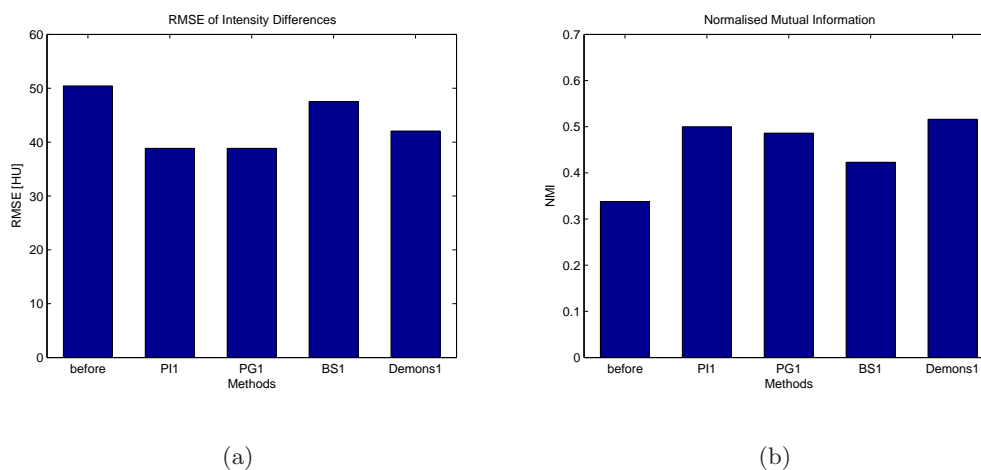


Figure 5.22: Evaluation results based on RMSE and NMI computations for a choice of algorithms with applied $256 \times 256 \times 256$ *Real* data set.

intensity differences in case of data set *Intensity5*. A registration of data set *Intensity2* using the improved implementations of the Periaswamy's approach gives suitable visual results.

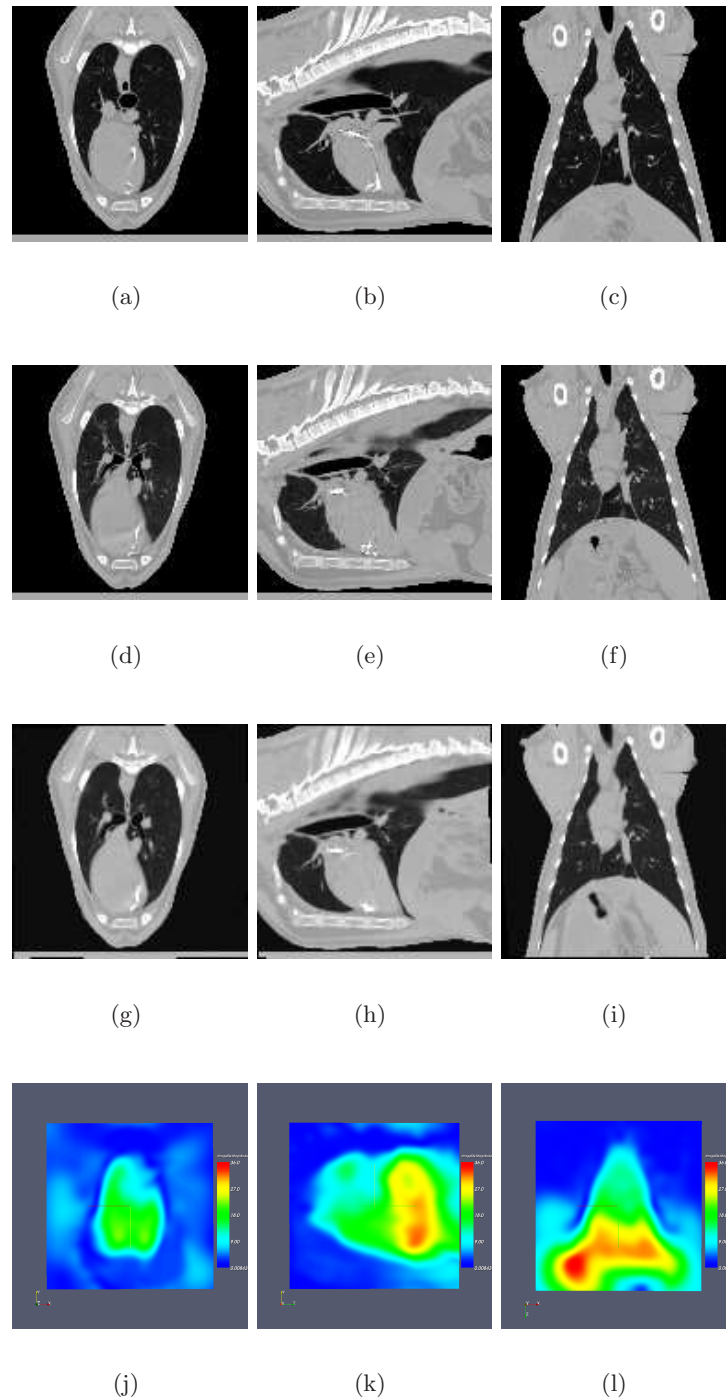


Figure 5.23: Sample slices of axial (first column), coronal (second column) and sagittal (third column) views of (a)-(c) the moving image, (d)-(f) the fixed image, (g)-(i) the warped image and (j)-(l) the according displacement field slices using method *PI1*.

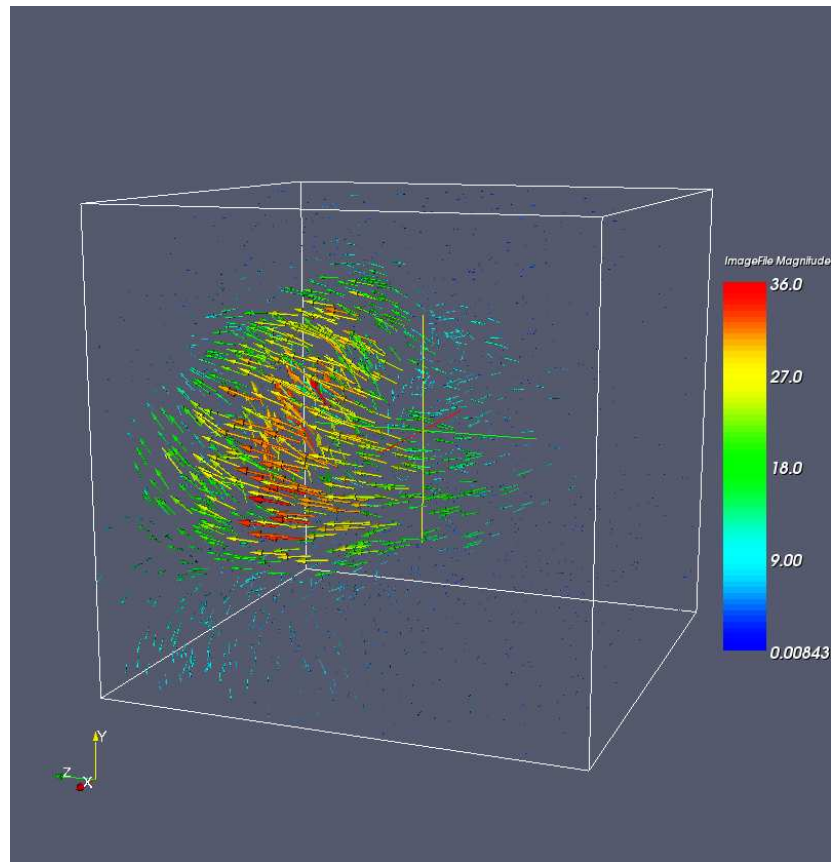


Figure 5.24: Computed displacement field in vector representation using method *PI1* applied on the $128 \times 128 \times 128$ *Real* data set.

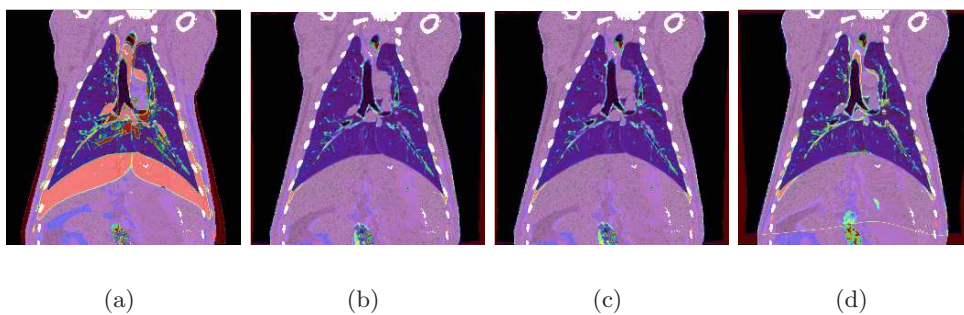


Figure 5.25: Sagittal sample slices of difference fusion images on real $256 \times 256 \times 256$ data sets: (a) before Registration, (b) after Registration using *PI2*, (c) after Registration using *PG2* and (d) after Registration using *BS2*

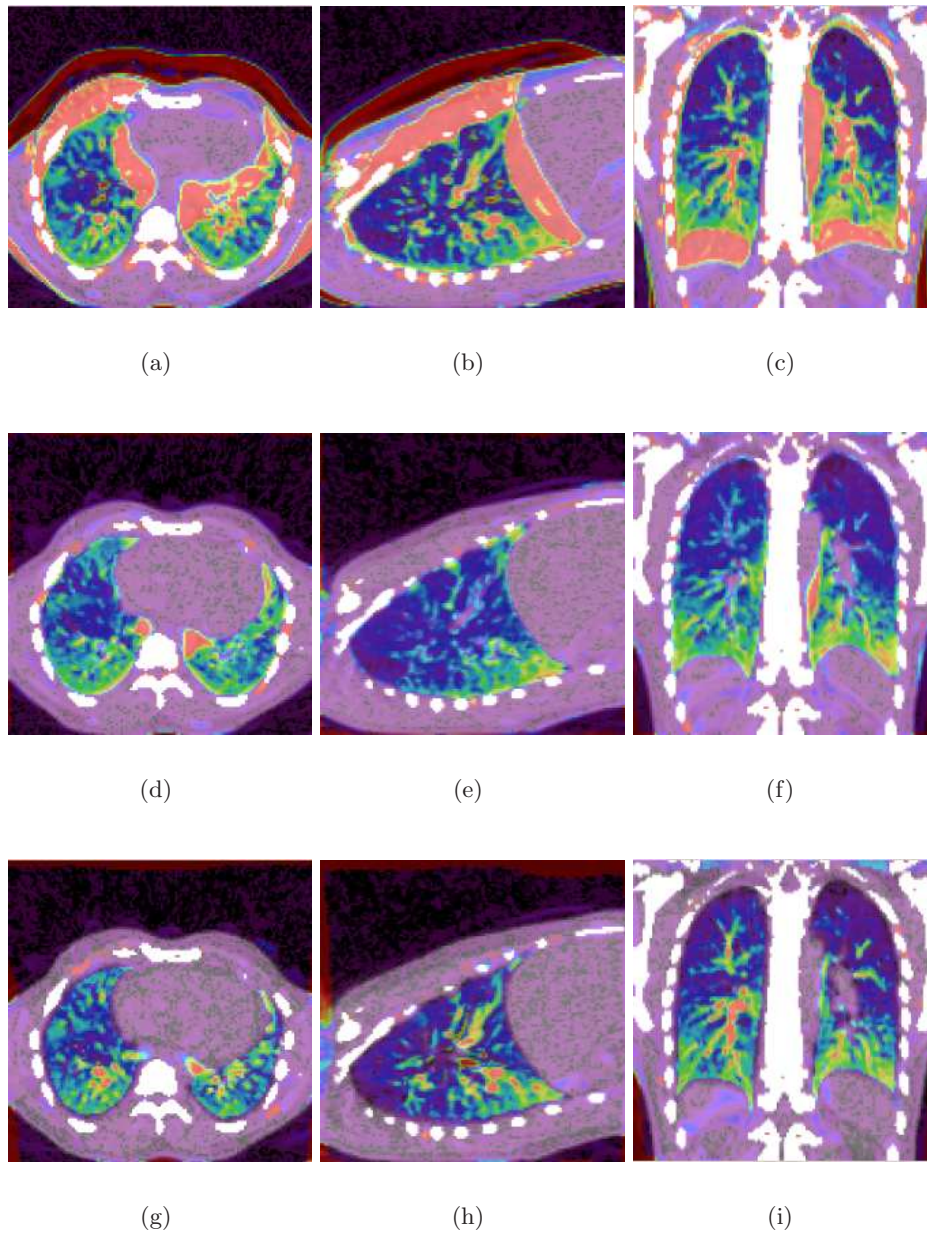


Figure 5.26: Visual image fusion difference images before and after registration of data set *Intensity5*: (a)-(c) axial, coronal and sagittal view before registration, (d)-(f) views after registration using method *PO1* and (g)-(i) views after registration using *Demons1*.

Method	Time [s]	RMSE [HU]	STD [HU]	NMI
before		50.435	40.162	0.338
PI1	4635.76	38.849	32.590	0.500
PI2	5142.83	39.973	33.431	0.484
PG1	521.87	38.838	32.506	0.486
PG2	1315.87	40.891	33.932	0.480
BS1	7695.00	47.535	37.843	0.423
BS2	23990.00	48.197	38.792	0.402
Demons1	379.89	42.059	35.515	0.516
Demons2	574.35	41.900	35.422	0.518

Table 5.15: A comparison of different registration methods by time consumption and similarity measurements on data set *Real* with a dimension of $256 \times 256 \times 256$.

Method	Time [s]	RMSE [HU]	STD [HU]	NMI
before		66.912	43.453	0.190
PO1	1640.87	49.048	37.585	0.384
PO2	2257.58	47.031	36.536	0.411
PI1	527.87	48.221	37.140	0.399
PI2	522.37	51.530	38.822	0.361
PG1	62.97	50.116	37.909	0.378
PG2	149.76	49.039	37.506	0.391
BS1	535.00	65.374	42.834	0.299
BS2	2951.00	70.779	43.085	0.290
Demons1	53.05	49.247	39.340	0.449
Demons2	80.60	47.966	38.715	0.463

Table 5.16: Evaluation results of various methods performed on data set *Intensity5* with a dimension of $128 \times 128 \times 128$.

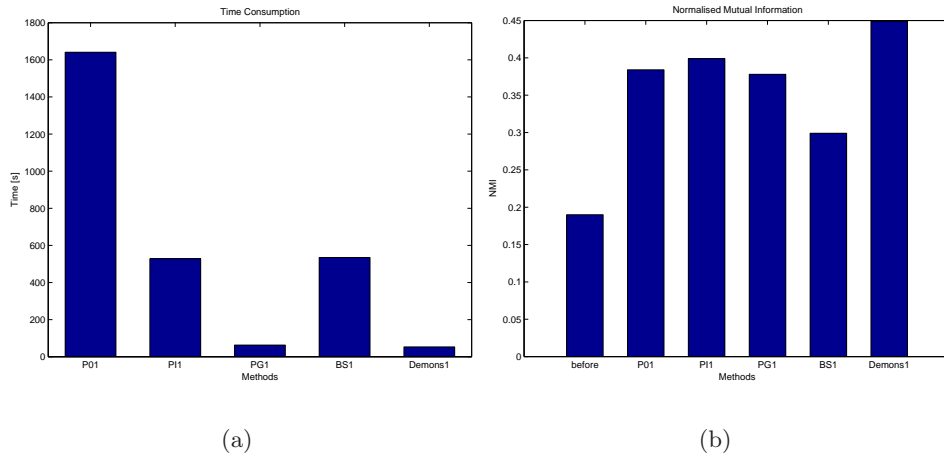


Figure 5.27: Graphical evaluation results of various methods performed on data set *Intensity5* with a dimension of $128 \times 128 \times 128$ for (a) time consumption and (b) NMI measurements.

Method	Time	RMSE	STD	NMI	Time	RMSE	STD	NMI
	[s]	[HU]	[HU]		[s]	[HU]	[HU]	
	$128 \times 128 \times 128$				$256 \times 256 \times 256$			
before		45.494	35.780	0.408		47.635	35.726	0.363
PO1	1906.18	33.733	27.955	0.552	-	-	-	-
PO2	2532.78	32.778	27.301	0.557	-	-	-	-
PI1	602.02	33.158	27.562	0.552	4640.54	31.265	25.106	0.531
PI2	642.56	34.510	28.480	0.540	4289.88	32.041	25.647	0.520
PG1	70.97	34.649	28.567	0.545	538.17	31.506	25.273	0.530
PG2	169.52	33.782	27.922	0.544	1377.38	31.684	25.368	0.522
BS1	1342.00	44.319	34.365	0.460	7561.00	42.462	32.565	0.441
BS2	2772.00	44.852	34.580	0.462	22210.00	41.988	32.211	0.449
Demons1	54.27	27.449	24.848	0.666	397.51	26.634	23.490	0.627
Demons2	83.83	27.381	24.798	0.667	627.43	26.567	23.454	0.632

Table 5.17: Evaluation results of various methods performed on data set *Intensity2* with a dimension of $128 \times 128 \times 128$ and $256 \times 256 \times 256$.

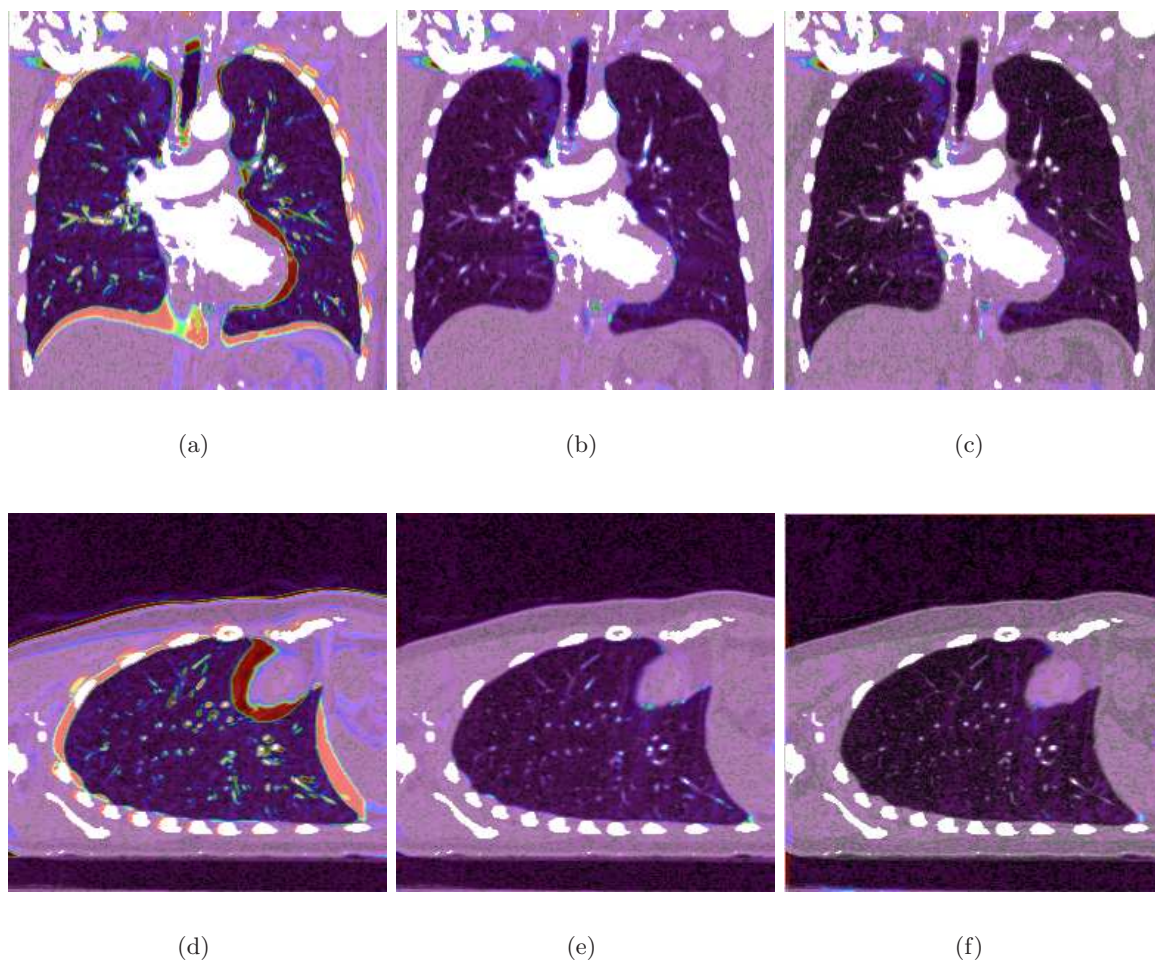


Figure 5.28: Visual image fusion difference images before and after registration of data set *Intensity2*: (a) coronal and (d) sagittal view before registration, (b),(e) views after registration using method *P11* and (c),(f) views after registration using *Demons1*.

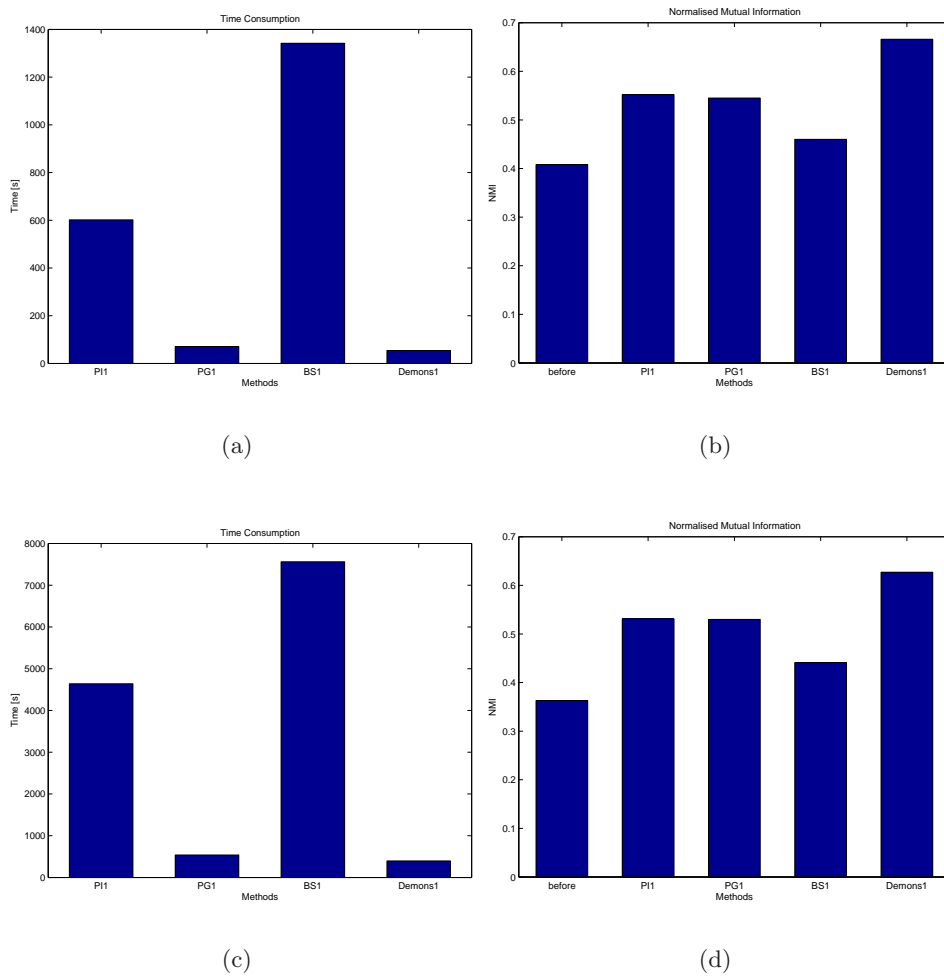


Figure 5.29: Direct comparison of the time consumption and the NMI measurements on data set *Intensity2* with a dimension of $128 \times 128 \times 128$ and $256 \times 256 \times 256$.

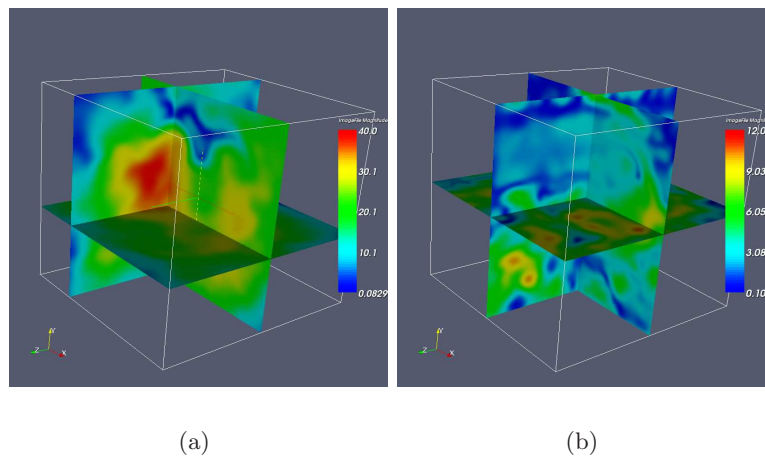


Figure 5.30: An illustration of the computed displacement fields for data set (a) *Intensity5* using method *PO1* and (b) *Intensity2* using method *PI1*.

Chapter 6

Summary and Discussion

6.1 Conclusion

This thesis has presented a non-linear registration approach for medical images. According to Periaswamy's approach, the essential implementation details have been highlighted. As proposed, this method makes it possible to generate a local affine, but globally smooth deformation.

We have found out, that this originally suggested approach is not applicable in practice for medical images. This approach suffers from high time and large memory consumption. Modern imaging techniques provide data with resolutions such as $512 \times 512 \times 512$ and higher. Inherently, the experimental results on $256 \times 256 \times 256$ images are conditionally acceptable. On the one hand, the number of smoothing iterations increases the computation time of the registration process, on the other hand the outer loops in each level introduce an error if interpolation in the re-sampling procedure is inaccurate.

The closed-form solutions for each spatial neighbourhood avoids an elaborate set-up of an iterative optimisation strategy. Otherwise the proposed algorithm gets by on a few parameters to tune. Furthermore, we have shown experiments on different transformation models. The models, including the brightness and contrast parameter estimates, achieve more accurate results.

To reduce the high computation time, we have realised a sub-sampling scheme. The interpolation between an affine matrix is complex, therefore a method of decomposition in its components, such as rotation, scaling, shearing and translation has been described. Comparing the methods of direct and decomposed interpolation, we have pointed out in our experiments that both methods perform equally. The sub-sampling scheme using direct interpolation obtains a significant speed-up. As a second improvement, we have replaced the intense memory consuming iterative smoothing, which performs the regularisation, by an efficient Gaussian smoothing. In general the choice of the parameter σ is intractable. By applying the sub-sampling scheme and the Gaussian smoothing with a $\sigma = 3 \dots 5$ to the algorithm, the method yields acceptable registration results in time consumption and accuracy. Additionally, numerical techniques for faster inverse matrix computation have been adapted to the given matrix properties. Comparing our algorithm to the *Demons* approach, we presented a reasonable and plausible registration method, because *Demons* fails under certain

conditions. However, *Demons* performs much better in terms of computation speed. In contrast to the B-spline approach, which suffers from an elongate computation time and a complex optimiser set-up, our improved method yields a constantly acceptable result with unchanged parameter settings on images with a dimension up to $256 \times 256 \times 256$. The implemented algorithm also obtains suitable results on data sets containing a moderate deformation with intensity variations.

6.2 Future Work

Some ideas for future work are:

- The idea of closed-form solutions in a spatial neighbourhood for a local displacement can be used to prior set-up another optimising strategy. To reduce the number of inverse computation, this idea can only be applied to specified location. One possibility is to perform displacement computation on a discrete grid or on extracted feature points. An interpolation technique, using B-Splines, would generate a dense deformation field between these grid points. A TPS interpolation can cope the interpolation of the non-uniform distributed feature points.
- Our presented Gaussian smoothing suffers from the problem of smearing edges. An edge-conserving method would avoid these effects. Diffusion based filtering techniques would prevent edge artefacts in our applied regularisation.
- Since we have used an Euler interpolation scheme, a further consideration of affine parameter interpolation would prove our experimental results.
- A masking of the images into fore- and background would give a reduction of the number of voxels to traverse. A smoothing routine and a subsequent region growing method with seed points may provide this mask in general.

Appendix A

Acronyms and Symbols

List of Acronyms

1D	one-dimensional
2D	two-dimensional
3D	three-dimensional
4D	four-dimensional
BFGS	Broyden-Fletcher-Goldfarb-Shanno
BOLD	blood oxygen level-dependent
CT	Computed Tomography
CC	correlation coefficient
DoF	degrees of freedom
FFD	free form deformations
FEM	finite elements method
fMRI	Functional MRI
GA	Genetic Algorithms
HU	Hounsfield units
ICP	iterative closest point
MI	mutual information
MRI	magnetic resonance imaging
MRT	magnetic resonance tomography
MSE	mean squared error
nCC	normalized cross correlation
NMI	normalised mutual information
NMR	nuclear magnetic resonance
NN	nearest neighbour
OF	optical flow
PDE	partial differential equation
PDF	probability density function
PET	Positron emission tomography
RBF	radial basis functions
RIU	ratio image uniformity

RMSE	root mean squared error
SA	Simulated Annealing
SAD	sum of absolute difference
SOR	successive over-relaxation
SPECT	Single Photon Emission CT
SPM	statistical parametric maps
SSD	sum of squared difference
STD	standard deviation
SVD	singular value decomposition
TPS	thin-plate splines
US	ultra sound

Appendix B

Evaluation Framework

The idea of an evaluation framework is intended to evaluate various registration algorithms automatically. Starting from a given pair of input images, the evaluation of different sets of parameters offers a direct comparison of several statistical measurements.

The framework is easy to handle, freely configurable and extensible.

B.1 Implementation

The evaluation framework is developed in *Python*, a powerful script programming language. The specified evaluation and execution parameters are summarised in separated configuration files. We specify the requirements as follows:

Data Sets

To investigate the results on synthetic and real data, the evaluation framework handles both kind of data. If synthetic data is employed, the second image is generated by using a given synthetic transformation. The type of transformation and its parameters are configurable.

File Names and Paths

The file names and their location on the system are specified in the configuration file. This file includes both, file names of the input images and the execution names of transformation, registration and evaluation routines. The names of resulting evaluation files are also declared.

Processing List

A processing list is given in the configuration file. An unique name identifies each execution of the registration process. Additionally, the type of registration and its parameters are given.

Registration Parameters

A second configuration file includes the representing parameters of the various registration types. This file gives the standard parameter values for orientation. The order of the specified parameters determines the sequence of parameters, which are set-up in the processing list.

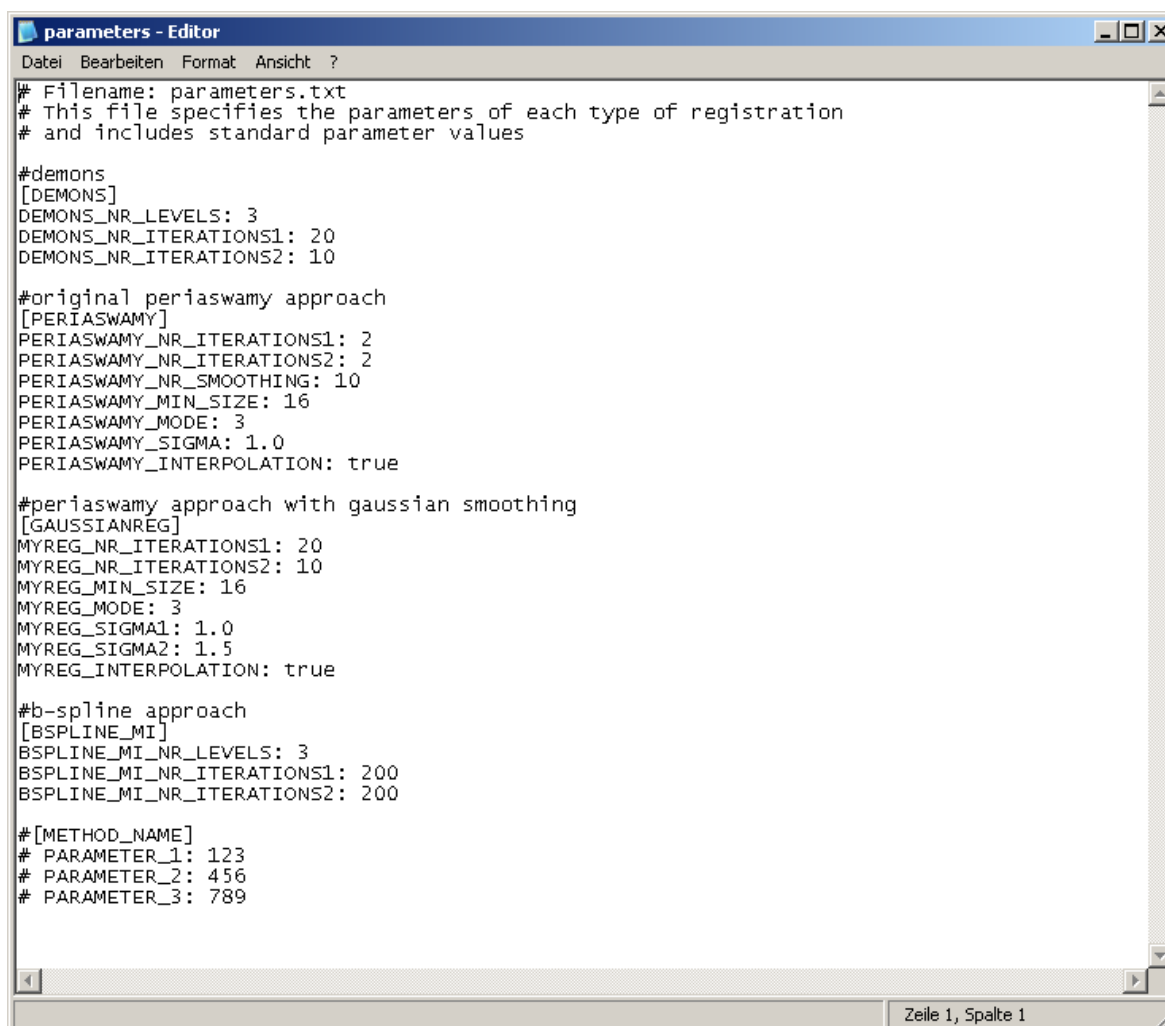
Evaluation Methods

In our case, there exist three different evaluation methods. The first method is based on intensity difference computation. Second, the time consumption is written in a separated result file. If the registration is performed on synthetic data, the output data is evaluated on displacement difference measurements as a third method.

B.2 Configuration Files

As mentioned before, the implementation of the evaluation framework takes two separate configuration files as input. The first configuration file includes the essential parameters of each type of registration method. This file has to be modified, if new types of registration are added.

The organisation of this file is illustrated in Figure B.1. The processing list configuration



```

# Filename: parameters.txt
# This file specifies the parameters of each type of registration
# and includes standard parameter values

#demons
[DEMONS]
DEMONS_NR_LEVELS: 3
DEMONS_NR_ITERATIONS1: 20
DEMONS_NR_ITERATIONS2: 10

#original periaswamy approach
[PERIASWAMY]
PERIASWAMY_NR_ITERATIONS1: 2
PERIASWAMY_NR_ITERATIONS2: 2
PERIASWAMY_NR_SMOOTHING: 10
PERIASWAMY_MIN_SIZE: 16
PERIASWAMY_MODE: 3
PERIASWAMY_SIGMA: 1.0
PERIASWAMY_INTERPOLATION: true

#periaswamy approach with gaussian smoothing
[GAUSSIANREG]
MYREG_NR_ITERATIONS1: 20
MYREG_NR_ITERATIONS2: 10
MYREG_MIN_SIZE: 16
MYREG_MODE: 3
MYREG_SIGMA1: 1.0
MYREG_SIGMA2: 1.5
MYREG_INTERPOLATION: true

#b-spline approach
[BSPLINE_MI]
BSPLINE_MI_NR_LEVELS: 3
BSPLINE_MI_NR_ITERATIONS1: 200
BSPLINE_MI_NR_ITERATIONS2: 200

#[METHOD_NAME]
# PARAMETER_1: 123
# PARAMETER_2: 456
# PARAMETER_3: 789

```

Figure B.1: The organisation of the parameters configuration file of the evaluation framework.

file includes the input image, the type of synthetic transformation and its parameters. Fur-

thermore, the execution names and their directory paths on the system are specified. The processing list defines the type of registration and the values for the set-up of the parameters. Figure B.2 shows the bodywork of this file.



```
# Processing List for Evaluation
# Filename: queue_eval.txt
# This file specifies the processing list
# and includes the execution parameters

# performs a registration on real data
[Images]
fixedImage: exhalation_image.hdr
movingImage: inhalation_image.hdr
synthetic_x:
synthetic_y:
inputImagePath: c:/images/
outputImagePath: c:/resultimages/

# # performs a registration on real data
# [Images]
# fixedImage: inhalation_image.hdr
# movingImage:
# synthetic_x: 25
# synthetic_y: 10
# inputImagePath: c:/images/
# outputImagePath: c:/resultimages/

[ProcessingList]
# Standard Name: ProcessName, registration method, param1, param2, param3, ..
Method1: GAUSSIANREG,2,2,16,3,1.0,3.0,true
Method2: PERIASWAMY,2,1,10,32,2,1.0,false
Method3: DEMONS,3,60,30
Method3: BSPLINE_MI,4,200,200

[FileNames]
IntensityResults: results_intensity.csv
DisplacementField: results_displacements.csv
ComputationTime: results_time.csv
ResultFilePath: c:/resultfiles/
ConfigFileName: config
ConfigFileNameExtension: .txt

[Parameters]
writeDisplacements: true

[Execution]
TransformationPath: c:/work/synthetic_bin/
Transformation: TransformVolume
CommandLineInterfacePath: c:/work/plugins/test/
CommandLineInterface: PluginsTest
EvaluationToolPath: c:/work/evaluate_bin/
EvaluationTool: ComparewarpedImages
```

Figure B.2: The organisation of the processing list configuration file of the evaluation framework.

B.3 Result Files

The evaluation framework produces *CSV* result files as an output. The specified result files include information about computation time and intensity similarity measurements. If

synthetic data is used, a third evaluation on displacement difference is performed. Depending on the implementation, the time consumption results are directly generated from the *C++* code or by means of the *Python* implementation.

The similarity measurement result files are produced from the implemented evaluation tools. These tools take the images and displacements, before and after registration, as input. The *CSV* format files can easily be processed in *Matlab* or *Microsoft Excel* for further graphical interpretation.

B.4 Extension

To extend this evaluation frame with additional registration algorithms, the parameter configuration file has to be modified. As a second step, the common *C++* interface, which implements the various registration algorithm, must be adapted.

Appendix C

Outline of the B-Spline Deformable Registration Algorithm

Rueckert et al. presented in [40] a new approach for non-linear registration of contrast enhanced breast MRI. The registration method is based on a global and a local motion model. The global model describes the overall rigid motion of the breast, whereas the local model specifies local deformations. Local deformations are modelled by FFD. The overall transformation T , which relates fixed and moving images is defined as

$$T(x, y, z) = T_{global}(x, y, z) + T_{local}(x, y, z). \quad (C.1)$$

This combined transformation consists of a global model and a local model.

Global Model

Rueckert suggested an affine registration model with 12 DoF for three dimensional data sets.

$$T_{global}(x, y, z) = \begin{pmatrix} \Theta_{11} & \Theta_{12} & \Theta_{13} \\ \Theta_{21} & \Theta_{22} & \Theta_{23} \\ \Theta_{31} & \Theta_{32} & \Theta_{33} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} \Theta_{14} \\ \Theta_{24} \\ \Theta_{34} \end{pmatrix} \quad (C.2)$$

Local Model

The local motion model is described by means of a free form deformation model based on cubic B-Splines. An underlying mesh of uniformly-arranged control points allows to deform the object. Definition of a free form deformation:

$$\begin{aligned} \text{Image Domain } \Omega &= \{(x, y, z) | 0 \leq x < X, 0 \leq y < Y, 0 \leq z < Z\} \\ \text{Grid Resolution} & n_x \times n_y \times n_z \\ \text{Uniform Grid Spacing} & \delta \\ \text{Control Points} & \phi_{i,j,k} \end{aligned}$$

$$T_{local}(x, y, z) = \sum_{l=0}^3 \sum_{m=0}^3 \sum_{n=0}^3 B_l(u) B_m(v) B_n(w) \phi_{i+l, j+m, k+n} \quad (C.3)$$

where

$$i = \left\lfloor \frac{x}{n_x} \right\rfloor - 1, \quad j = \left\lfloor \frac{y}{n_y} \right\rfloor - 1, \quad k = \left\lfloor \frac{z}{n_z} \right\rfloor - 1,$$

$$u = \frac{x}{n_x} - \left\lfloor \frac{x}{n_x} \right\rfloor, \quad v = \frac{y}{n_y} - \left\lfloor \frac{y}{n_y} \right\rfloor, \quad w = \frac{z}{n_z} - \left\lfloor \frac{z}{n_z} \right\rfloor$$

and where $B_l(u), B_m(v), B_n(w)$ represent the $\{l, m, n\}$ -th basis functions of the B-Splines [23]

$$B_0(u) = \frac{(1-u)^3}{6}$$

$$B_1(u) = \frac{3u^3 - 6u^2 + 4}{6}$$

$$B_2(u) = \frac{-3u^3 + 3u^2 + 3u + 1}{6}$$

$$B_3(u) = \frac{u^3}{6}$$

In order to make the optimization process more robust and efficient in time, a multi-resolution approach is used. The resolution of control points is increased in each multi-resolution step. Optimization starts with a low resolution to rapidly obtain a rough estimate of the overall local deformation. For each resolution an increasing number of control points is defined to refine the estimated deformation from the previous level. The resulting additional transformation parameters are calculated by a B-Spline subdivision algorithm.

For each resolution a new T_{local}^l is calculated. The overall local transformation is defined as

$$T_{local}(x, y, z) = \sum_{l=1}^L T_{local}^l(x, y, z) \quad (C.4)$$

Rueckert et al. proposed NMI to measure the degree of similarity in each registration step. They suggested a simple gradient descent techniques for the optimization of the similarity cost functions for the local and global transformation models, respectively. Both the global and the local transformation model are optimized by a regular step gradient descent optimization strategy.

Bibliography

- [1] Arvo, J. (1991). *Graphics Gems II*. Academic Press, Inc., Boston, MA, USA.
- [2] Bajcsy, R. and Kovacic, S. (1989). Multiresolution Elastic Matching. *Computer Vision, Graphics and Image Processing*, 46(1):1–21.
- [3] Barron, J., Fleet, D., Beauchemin, S., and Burkitt, T. (1992). Performance Of Optical Flow Techniques. *Proceedings IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 92:236–242.
- [4] Besl, P. and McKay, N. (1992). A Method for Registration of 3-D Shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256.
- [5] Beauchemin, S.S. and Barron, J.L. (1995). The computation of optical flow. *ACM Computing Surveys*, 27:433–467.
- [6] Birchfield, S. (1997). Derivation of Kanade-Lucas-Tomasi tracking equation, unpublished notes. citeseer.ist.psu.edu/birchfield97derivation.html.
- [7] Bookstein, F. (1989). Principal Warps: Thin-Plate Splines and the Decomposition of Deformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(6):567–585.
- [8] Bro-Nielsen, M. (1996). Fast fluid registration of medical images. In *4th International Conference Visualization in Biomedical Computing VBC 1996*, pages 267–276.
- [9] Brown, L. G. (1992). A Survey of Image Registration Techniques. *ACM Computing Surveys*, 24(4):325–376.
- [10] Collignon, A., Maes, F., Delaere, D., Vandermeulen, D., Seutens, P., and Marchal, G. (1995). Automated multimodality image registration using information theory. In *Information Processing Medical Imaging: Proceeding of 14th International Conference IPMI 95*, pages 263–274.
- [11] Dam, E. B., Koch, M., and Lillholm, M. (1998). Quaternions, interpolation and animation. Technical Report DIKU-TR-98/5, Department of Computer Science, University of Copenhagen.
- [12] Davatzikos, C. (1996). Image Registration Based on Boundary Mapping. *IEEE Transactions on Medical Imaging*, 15(1):112–115.
- [13] Deriche, R. (1993). Recursively Implementing the Gaussian and its Derivatives. Technical Report 1893, Institut National de Recherche en Informatique et en Automatique (INRIA), 2004 routes des Lucioles, B.P. 93, 06902 Sophia-Antipolis, France.
- [14] Edwards, P. J., Hill, D. L. G., Little, J. A., and Hawkes, D. J. (1998). A three-component deformation model for image-guided surgery. *Medical Image Analysis*, 2(4):355–367.

- [15] Farid, H. and Simoncelli, E. P. (1997). Optimally Rotation-Equivariant Directional Derivative Kernels. In *Proceedings of the 7th International Conference on Computer Analysis of Images and Patterns, Kiel, Germany*, pages 207–214. Springer.
- [16] Feldmar, J. and Ayache, N. (1996). Rigid, affine and locally affine registration of free-form surfaces. *International Journal of Computer Vision*, 18(2):99–119.
- [17] Friston, K., Ashburner, J., Frith, C., Poline, J., Heather, J. D., and Frackowiak, R. (1995). Spatial registration and normalization of images. *Human Brain Mapping*, 2:165–189.
- [18] Hajnal, J., Hill, D., and Hawkes, D. J., editors (2001). *Medical Image Registration*. CRC Press, Boca Raton.
- [19] Hendee, W. R., editor (2002). *Medical Imaging Physics*. Wiley-Liss.
- [20] Higham, N. J. (2002). *Accuracy and Stability of Numerical Algorithms*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, second edition.
- [21] Horn, B.K.P and Schunck, B.G. (1981). Determining optical flow. *Artificial Intelligence*, 17(1-2):185–203.
- [22] Jahne, B. (1993). *Spatio-Temporal Image Processing: Theory and Scientific Applications*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.
- [23] Lee, S., Wolberg, G., Chwa, K.-Y., and Shin, S. Y. (1996). Image metamorphosis with scattered feature constraints. *IEEE Transactions on Visualization and Computer Graphics*, 2:337–354.
- [24] Liu, D. C. and Nocedal, J. (1989). On the limited memory BFGS method for large-scale optimization. *Math. Programming*, 45:503–528.
- [25] Liu, H., Hong, T.-H., Herman, M., Camus, T., and Chellappa, R. (1998). Accuracy vs efficiency trade-offs in optical flow algorithms. *Computer Vision and Image Understanding: CVIU*, 72(3):271–286.
- [26] Lucas, B. and Kanade, T. (1981). An Iterative Image Registration Technique with an Application to Stereo Vision. In *Proc 7th International Conference on Artificial Intelligence*, pages 674–679.
- [27] Maes, F., Collignon, A., Vandermeulen, D., Marchal, G., and Suetens, P. (1997). Multi-modality Image Registration by Maximization of Mutual Information. *IEEE Transactions on Medical Imaging*, 16(2):187–198.
- [28] Maintz, J. and Viergever, M. (1996/97). An Overview of Medical Image Registration Methods. In *Symposium of the Belgian Hospital Physicists Association (SBPH/BVZF)*, volume 12, pages V:1–22.
- [29] Maintz, J. and Viergever, M. (1998). A Survey of Medical Image Registration. *Medical Image Analysis*, 2(1):1–36.

- [30] Mattes, D., Haynor, D. R., Vesselle, H., Lewellen, T. K., and Eubank, W. (2003). PET-CT Image Registration in the Chest Using Free-form Deformations. *IEEE Transactions on Medical Imaging*, 22(1):120–128.
- [31] Negahdaripour, S. and Yu, C.-H. (1993). A generalized brightness change model for computing optical flow. In *Computer Vision, Proceedings, Fourth International Conference on Computer Vision*, pages 2–11.
- [32] NLM (2006). National Library of Medicine, Insight Segmentation and Registration Toolkit (ITK). <http://www.itk.org>.
- [33] Peckar, W., Schnorr, C., Rohr, K., and Stiehl, H. S. (1997). Two-Step Parameter-Free Elastic Image Registration with Prescribed Point Displacements. In *Proceedings 9th International Conference on Image Analysis and Processing (ICIAP '97)*, pages 527–534, Florence, Italy.
- [34] Pelizzari, C. A., Chen, G. T. Y., Spelbring, D. R., Weichselbaum, R. R., and Chen, C.-T. (1989). Accurate three-dimensional registration of CT, PET, and/or MR images of the brain. *Journal of Computer Assisted Tomography*, 13:20–26.
- [35] Pennec, X., Cachier, P., and Ayache, N. (1999). Understanding the demon’s algorithm: 3d non-rigid registration by gradient descent. In *Proceedings of the 2nd International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pages 597–605.
- [36] Periaswamy, S. and Farid, H. (2003). Elastic registration in the presence of intensity variations. *IEEE Transactions on Medical Imaging*, 22(7):865–874.
- [37] Pluim, J., Maintz, J., and Viergever, M. (2003). Mutual-Information-Based Registration of Medical Images: A Survey. *IEEE Transactions on Medical Imaging*, 22(8):986–1004.
- [38] Press, W. H., Flannery, B. P., Teukolsky, S. A., and Vetterling, W. T. (1993). *Numerical Recipes in C*. Cambridge University Press, Cambridge, England, 2nd edition.
- [39] Rohr, K., Stiehl, H. S., Sprengel, R., Buzug, T. M., Weese, J., and Kuhn, M. H. (2001). Landmark-Based Elastic Registration Using Approximating Thin-Plate Splines. *IEEE Transactions on Medical Imaging*, 20(6):526–534.
- [40] Rueckert, D., Sonoda, L. I., Hayes, C., Hill, D. L. G., Leach, M. O., and Hawkes, D. J. (1999). Nonrigid Registration Using Free-Form Deformations: Application to Breast MR Images. *IEEE Transactions on Medical Imaging*, 18(8):712–721.
- [41] Shi, J. and Tomasi, C. (1994). Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, Seattle.
- [42] Shoemake, K. (1985). Animating rotation with quaternion curves. In *SIGGRAPH '85: Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, pages 245–254. ACM Press.

- [43] Sonka, M., Hlavac, V., and Boyle, R. (1999). *Image Processing, Analysis and Machine Vision*. Brooks/Cole Publishing Company, Pacific Grove, CA, USA, 2nd edition.
- [44] Studholme, C., Hill, D., and Hawkes, D. (1995). Multiresolution voxel similarity measures for MR-PET registration. In *Information Processing Medical Imaging: Proceeding of 14th International Conference IPMI 95*, pages 287–298.
- [45] Studholme, C., Hill, D., and Hawkes, D. (1999). An overlap invariant entropy measure of 3D medical image alignment. *Pattern Recognition*, 32(1):71–86.
- [46] Thirion, J.-P. (1998). Image matching as a diffusion process: An analogy with Maxwell’s demons. *Medical Image Analysis*, 2(3):243–260.
- [47] Thompson, P. and Toga, A. W. (1996). A Surface-Based Technique for Warping Three-Dimensional Images of the Brain. *IEEE Transactions on Medical Imaging*, 15(4):402–417.
- [48] Tomasi, C. and Kanade, T. (1991). Detection and tracking of point features. Technical Report CMU-CS-91-132, Carnegie Mellon University.
- [49] Urschler, M. and Bischof, H. (2005). Assessing breathing motion by shape matching of lung and diaphragm surfaces. In *SPIE Symposium on Medical Imaging 2005: Physiology, Function, and Structure from Medical Images*, volume 5746, pages 440–452.
- [50] Viola, P. and Wells III, W. (1997). Alignment by Maximization of Mutual Information. *International Journal on Computer Vision*, 24(2):137–154.
- [51] Wollny, G. and Kruggel, F. (2002). Computational Cost of Nonrigid Registration Algorithms Based on Fluid Dynamics. *IEEE Transactions on Medical Imaging*, 21(8):946–952.
- [52] Woods, R. P., Grafton, S. T., Watson, J. D. G., Sicotte, N. L., and Mazziotta, J. C. (1998). Automated Image Registration: II. Intersubject Validation of Linear and Nonlinear Models. *Journal of Computer Assisted Tomography*, 22(1):153–165.
- [53] Woods, R. P., Mazziotta, J. C., and Cherry, S. R. (1993). MRI-PET Registration with Automated Algorithm. *Journal of Computer Assisted Tomography*, 17(4):536–546.
- [54] Xie, Z. and Farin, G. (2004). Image Registration Using Hierarchical B-Splines. *IEEE Transactions on Visualization and Computer Graphics*, 10(1):85–94.
- [55] Zitova, B. and Flusser, J. (2003). Image registration methods: A survey. *Image and Vision Computing*, 21(11):977–1000.