Comparison of distance measures in Multimodal registration of medical images

Master thesis in applied and computational mathematics

Jon Kirkebø Nyfløtt

Institute of Mathematics
University of Bergen
Norway

June 14, 2011

Acknowledgments

Before I started working on this thesis I really had no idea what I was getting myself into. The actual work invested in this eclipses any pre-existing idea I had of how much time this would take. Never before have I had the possibility of submerging myself this deeply in one subject, and I guess I will never again get such a chance.

First I would like to thank Antonella Zanna Munthe-Kaas as my mentor. She has provided me with interesting tasks and ideas and helped me lay out the framework of the thesis, as well as leading me in the right directions whenever I was stuck. I would also like to thank Erlend Hodneland, who has been working on a similar project as myself, and who has been helpful whenever I had questions.

I would also like to thank Erik Natvig, for reading some of the earlier drafts and helping me with spelling and consistency.

Most importantly, I wish to thank Gry, for endless encouragements, patience and love. I can not imagine how tough this last year would have been without her. I dedicate this to you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In my project I have worked with a number of problems related to *Image Registration*. From the start I have worked with registering brain images using a software package called FAIR [18], and specifically with looking at the importance of choosing distance measures suited for the task. Last fall I spent some time working on a project with the Image Processing Group here at UiB related to Forskningsdagane, a science fair for kids. We made a program that registered pictures of faces to animals, celebrities, cartoon figures etc.

1.1 FAIR

I have based most of the registration on a software package called FAIR. FAIR has been developed by Jan Modersitzki, and is an extensive collection of image processing (in particular image registration) tools. The book FAIR - Flexible Algorithms for Image Registration [18] explains the framework used in the software and most of the theory behind it. I have decided to use the same formalism as Modersitzki uses in the book whenever possible.

One of the important concepts in the model used in FAIR is that it is a continuous setting; all the variables are functionals. There are multiple reasons for choosing a continuous model, both philosophically; since an image is a model of a continuous setting and thus should be continuous, and practical; since our transformed image most likely will not align with a fixed, discrete pixel grid. In addition this approach is more efficient. To obtain this continuous model we use interpolation as discussed in Section 3.1.

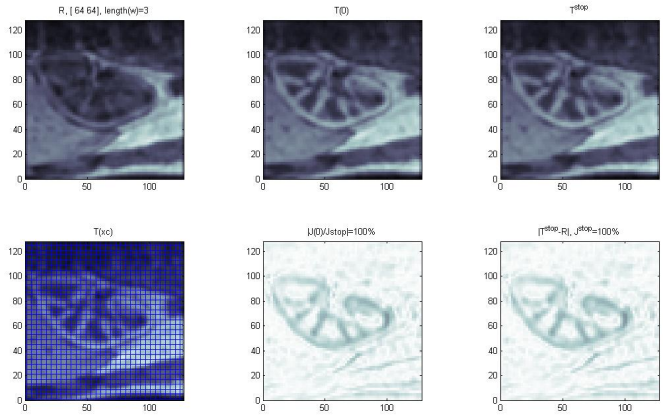
Getting familiar with the software I have used and the theory behind it has been a bigger undertaking than I imagined it would be. The multilayered structure of the software and the, in my opinion, somewhat lacking commentary made it difficult to follow the process at times. I have certainly learned the value of good comments and clarity.

Despite this I really appreciate the opportunity of starting with a complete registration package. Without it I would surely not have gotten as far as I have.

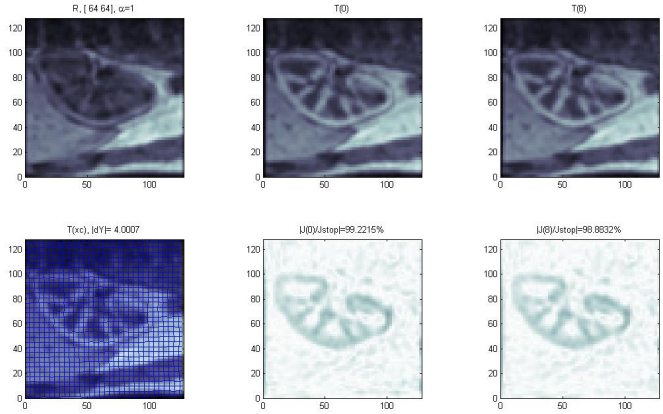
Many of the results I show will be images generated by the FAIR software. The basic layout of a registration result in shown in Figure 1.1.

1.2 Image processing

In mathematics, we may view a digital image as a two- or three-dimensional function, $\mathcal{T}(x)$, where the points x are the spatial coordinates and the function value $\mathcal{T}(x_0)$ is the intensity value at coordinates x_0. For graytone images, which encompasses all images I have been working on, $\mathcal{T}(x_0) \in \mathbb{R}$. The choice of viewing an image as a function opens up a lot of possibilities for manipulating the image or for extracting information out of it. We can for instance find edges in the image by looking at the gradients, remove noise by filtering the Fourier spectrum of the

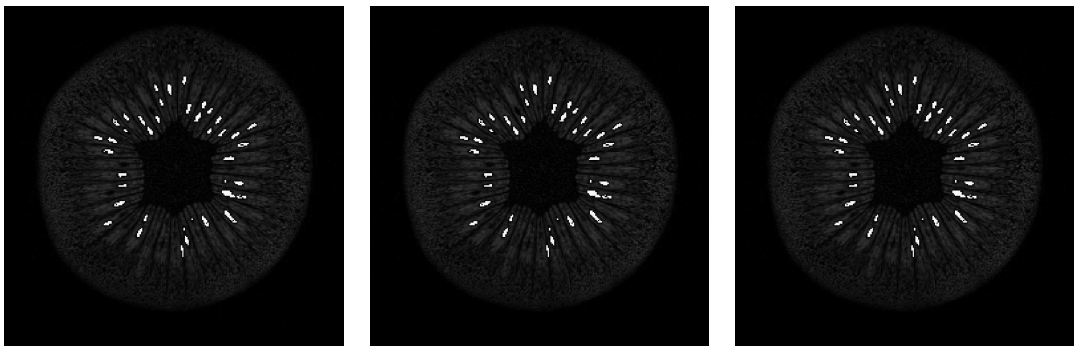


(a) *Start of registration*



(b) *End of registration*

Figure 1.1: *Basic FAIR graphical output. In the top left corner of the output is the reference image. Top middle shown the template image interpolated at the starting guess. The template at the current grid is shown in the upper right corner. The lower row shows the original template overlayed with the current grid and the difference between reference and template at starting guess and registered grid, respectively.*



(a) **(b)** **(c)**

Figure 1.2: *Example of image processing. Three slices of an MRI taken of a Kiwi-fruit. The seeds have been segmented out using a image processing method known as k-means.*

image and increase contrast by manipulating the distribution of intensity values (histogram) just to name a few applications.

We think of the images we view on a computer as a representation of a real object. As such this is a digital or discrete representation of a continuous function. We use some special terms when discussing computer representations of images:

Definition 1.2.1 (Digital images)**.** A **digital image** is a collection of pixels or voxels, where

- A **pixel** (picture element) is a discrete unit with information about position and intensity value in a two-dimensional image. The position is a 2-dimensional vector, $x \in \Omega \subset \mathbb{R}^2$. The **intensity values** are given by a function $\mathcal{T}(x) : \Omega \to \mathbb{R}$, where $\Omega \in \mathbb{R}^2$ is the image domain.
- A **voxel** (volume element) is the equivalent of a pixel for $x \in \mathbb{R}^3$.

An example of image processing that is of special interest to us is the applying of transformations to the image. We may express these transformations as

$$\mathcal{T}[y](x) = \mathcal{T}(y(x)), \tag{1.2.1}$$

where the function $y : \mathbb{R}^d \to \mathbb{R}^d$ takes the coordinates x and maps them to $y(x)$. Transformations will be discussed further in Section 3.2.

1.2.1 Image registration

The need for Image registration often comes up in image processing when dealing with multiple images. In short, image registration is the process of aligning two or more images of the same object or scene [10]. Example of fields where image registration is important range from astronomy, remote sensing, robotics, biology and criminology. There are a variety of methods developed for registration, both between fields and within medical image registration (see for instance [3, 30]).

We start by choosing one of the images to be a *reference image*, $\mathcal{R}$, and the other(s) as *templates image(s)*, $\mathcal{T}$. Now, if one knew exactly how the images related to each others and the transformation taking us from $\mathcal{R}$ to $\mathcal{T}$, all that would be needed to do was apply the inverse of the transformation and we would have a perfect match. In practice, however, we never know the exact transformation. Furthermore, the existance of such a transformation might not be true. Often the images are taken at different times or with different settings such that intensity values might not correspond from $\mathcal{R}$ to $\mathcal{T}$.

The problem turns into one of finding a transformation that takes us *close enough* to the target image, going through an iterative process of transforming our template step by step and checking how close we are to the target. There are a number of important tools needed to do image registration, and some of them will be discussed in this chapter. Most of the theory in this chapter is taken from the framework laid out in [18]. Wherever I have used other sources or made additions on my own this will be clarified.

In the FAIR registration model the mathematical model is based mainly on the *distance measure* and the *regularization*. In short, to make the registration problem as compact as possible, we create an objective function, $\mathcal{J}$, that we want to minimize. This function may be composed of different parts that we want to use in our registration, but in the FAIR framework it will consist of a sum of the distance measure, $\mathcal{D}$, and the regularizer, $\mathcal{S}$. These will both be functions of the template image, the reference image and the transformation y. The objective function with these terms be expressed as

$$\mathcal{J}[y] = \mathcal{D}[\mathcal{T}[y], \mathcal{R}] + \alpha \mathcal{S}[y - y^{\text{ref}}]. \tag{1.2.2}$$

Here α is a constant determining the amount of regularization. y^{ref} is the grid at the start of the current registration.

I have mostly focused on the distance measure in my work, including testing the ones in the FAIR package and implementing a new one. These will all be discussed in Section 3.4.

In Chapter 2 I will introduce the project data I have been working on and the motivation behind the methods I have been using. I will also introduce some history and theory on the subject of medical imaging.

Chapters 4 and 5 will show some examples of the registration results; first on synthetic data then on the real medical images, before a conclusion is provided in Chapter 6.

Finally, Chapter 7 includes an appendix with additional figures and some pieces of the code I have used.

A note on notation

I have tried to use the same notation as that used in FAIR for my thesis. As such, $\mathcal{R}$, $\mathcal{T}$, etc. refers to the continuous data and `Tc`, `Rc`, etc. refers to discretized data. For transformations and grids, x usually refers to the original data grid, whereas y refers to the transformed grid.

Chapter 2

Multimodal Image registration

In this project I have mainly looked at problems regarding image registration of *medical images*. In medical practice images of different modalities can provide vital information that can be used in clinical practice or medical science [10]. The history of medical imaging goes as far back as the late 19th century, when Wilhelm Conrad Röntgen discovered that certain wavelengths (that we now call X-rays) in the electromagnetic spectrum can penetrate solid material and thus create an image showing internal structure [10]. Today we have a multitude of different technologies to obtain images of the anatomy, from the early X-rays to more recent discoveries; such as magnetic resonance imaging (MRI), computer tomography (CT), positron emission tomography (PET) and ultrasound (US). Sometimes we want to use images of different modalities from the same anatomy or images taken at different times or of different patients. To compares these it helps to have them relatively similar, and in order to get similar images we need to align them. The process of aligning images is often referred to as image registration.

2.1 Image modalities

Most of the medical images we rely on today are the result of fairly recent advances in physics, especially in the field known as particle physics. For instance, the discovery of the natural phenomenon known as Nuclear Magnetic Resonance was first published by I.I. Rabi in 1938 [23]. This later led to the development of the technology behind Magnetic Resonance Imaging (MRI). Computer Tomography (CT) is based on X-rays.

2.1.1 Magnetic Resonance

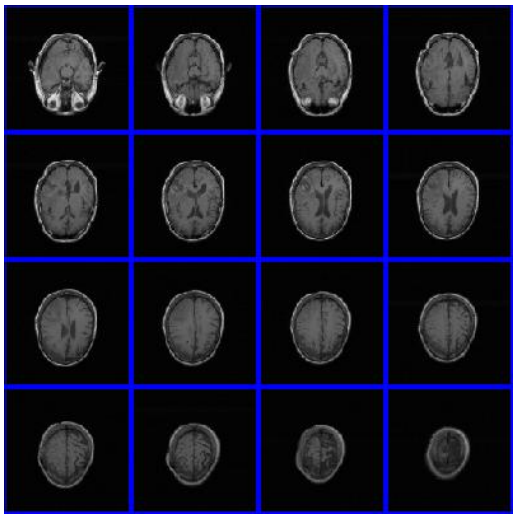
Magnetic Resonance Imaging is based on the magnetic moments of particles in the body. The magnetic moment comes from the fact that some particles have a non-zero *spin*. The spin is an intrinsic property of subatomic particles' quantum states; a form of angular momentum. All matter is composed of elementary particles, such as quarks and electrons, known as fermions. These all have spin $s = \frac{1}{2}$ and thus a magnetic moment. Composite particles, such as the protons and neutrons, take the sum of the spins of their composing fermions. A proton, composed of two up and one down quark, has a resulting spin of $\frac{1}{2}$ [16].

Of special interest to us is the ^{1}H-isotope. The reason for this interest is that it is the most common molecule in the human body. It also produces a stronger signal than most other stable nuclei [1]. Pathologies will often change the composition of molecules in the affected area, leading to a change in observable signal (intensities in the image).

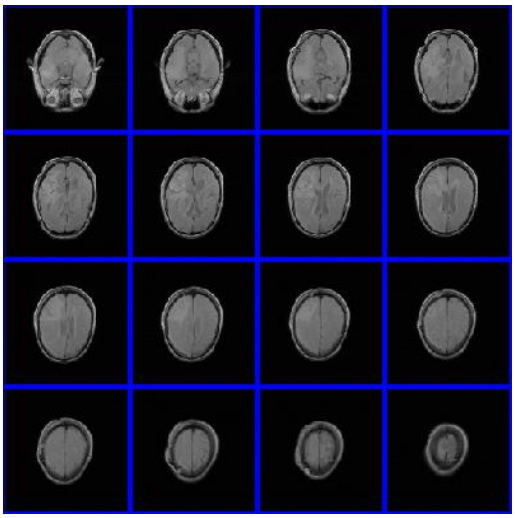
Because of this spin we get a magnetic moment,

$$\mu = \gamma \mathbf{I}, \tag{2.1.1}$$

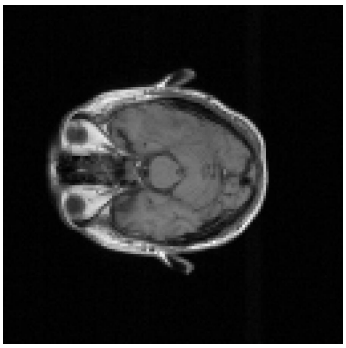
where γ is the gyromagnetic ratio. The gyromagnetic ratio is dependent on the nucleus.



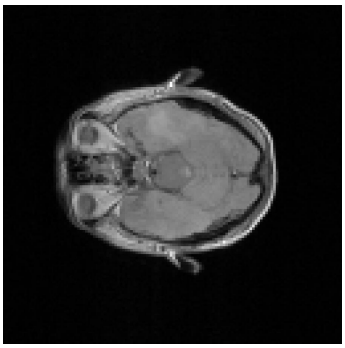
(a) *T1-weighted MRI montage*



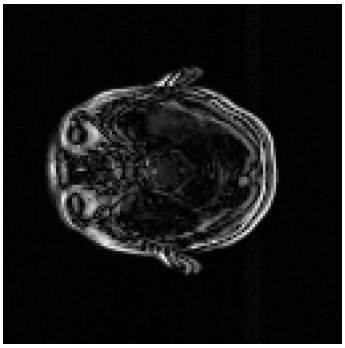
(b) *PD-weighted MRI montage*



(c) *Slice of T1 data*



(d) *Slice of PD data*



(e) *Difference T1 and PD*

Figure 2.1: *Example of unregistered medical images.*

Weighting	TR	TE	Description
T_1	Short (< 750 ms)	Short (< 40 ms)	Can be run very fast due to short repetition time. Usually involves the introduction of a contrast agent such as gadolinium, with images taken before and after injection. T_1 images give good contrast between gray and white matter.
T_2	Long (> 1500 ms)	Long (> 75 ms)	Good sequence for detecting pathologies.
PD	Long (> 2000 ms)	Short(< 20 ms)	Minimizes the contribution from both T_1 and T_2. Intensity based on number of protons in an area (Proton Density).

Table 2.1: Different versions of MRI.

Outside the influence of a magnetic field, all possible orientations of the spin have the same energy. Because of this, the number of atoms in each state will be approximately equal at equilibrium. Placing the nucleus in an electric field will influence the balance between the states and they will no longer have the same energy. The resulting energy difference between the two states results in a bias towards the lower energy state. A convenient way to picture this is to think of the spins as aligning with the magnetic field.

The factor

$$\nu = \frac{\gamma B}{2\pi} \tag{2.1.2}$$

is of importance; it is the frequency of electromagnetic radiation required to shift the quantum spin state known as the resonance frequency. The frequencies needed for shifting a hydrogen nuclei in a magnetic field are in the order of

$$\nu = \frac{\gamma}{2\pi} \approx \frac{26.75 \times 10^7 \,\mathrm{T}^{-1}\,\mathrm{s}^{-1}}{2\pi} = 4.2 \times 10^7 \,\mathrm{Hz}\,\mathrm{T}^{-1}, \tag{2.1.3}$$

or about 42 MHz per Tesla, which lies in the radiofrequency part of the electromagnetic spectrum.

In Magnetic Resonance Imaging (MRI) the patient is exposed to a magnetic field B_0 in the range of about 1 to 7 Tesla. This will give a net polarization parallel to the field, with a net magnetization M_0. We define the direction of the field as the z-axis, so that $M_{z,eq} = M_0$. The patient is then subjected to a Radio Frequency (RF) pulse, either perpendicular or antiparallel to the field, which can tip the net magnetization either 90 or 180 degrees. This will change the value of M_z, either to 0 (the net magnetization lies solely in the xy-plane) or to $-M_{z,eq}$, depending on the angle between the field and the RF pulse.

The choice of RF pulse sequences defines the type of MRI we get. A common form is what is known as the spin-echo sequence. Starting with a 90° pulse, the magnetization vector is shifted to the xy-plane, and the net magnetization starts shifting back to the original direction. After a certain time t a 180° pulse rotates the magnetization vector in the x-plane. This pulse is called a refocusing pulse, leading to a signal echo at time $2t = TE$. Typically, additional 180° pulses are applied, with a repetition time TR. This method was first described in [14]. Different TE and TR give us different intensity values, some of the more common ones are listed in Table 2.1.

Since the energy of this radiation is a lot lower than that required to ionize the molecules in the body and the intensity is too low to produce significant heating, the radio frequencies are not expected to do any damage. This, combined with high sensitivity, makes MRI a great tool for diagnosing certain types of pathologies.

2.1.2 Computed Tomography

Radiography, the use of radiation to create images of internal structure, had its advent with the discovery of X-rays at the start of the nineteenth century. For many decades the images had the disadvantage of being projections of the three dimensional object they were depicting, thus providing no depth information [27]. This was improved upon by the invention of computer-assisted tomography (CT or CAT) scanners in the 1970s. A fan-shaped slice of a X-ray beam is rotated about the patient and recorded by an arc of detectors on the opposite side. The resulting image shows an entire slice of the body.

Any radiation sent through a medium will have its intensity reduced at a factor depending on the material. This factor is known as the *attenuation coefficient*. Since the body is composed of different materials with different attenuation coefficients, the photon count measured going through the medium will be depending on the materials inside. Using these numbers as the intensity values gives us an image showing the makeup of the medium. See e.g. [15] for a more detailed discussion on Computed Tomography.

2.1.3 Position Emission Tomography

The nucleus of an atom is composed of nucleons; protons and neutrons. The number of protons give us the atomic number, and thus the element. The number of neutrons may vary, and the total number of nucleons decides the isotope of the element. Some isotopes are inheritable unstable. These are radioactive, and will eventually decay into a more stable form. The decay can be in multiple forms, but all have in common that they change the number of nucleons.

One way the nuclei can decay is known as *position emission* or *beta-plus decay*. A proton (p) in the nucleus is converted into a neutron (n) and a position (e^+). An example is the decay of Carbon-11, or ${}^1 1C$. Carbon is the sixth element, and thus has 6 protons. The isotope number tells us that there are 5 neutrons. ${}^{11}C$ decays solely through position emission,

$$ {}^{11}C \rightarrow {}^{11}\ B + e^+ + \nu, \tag{2.1.4} $$

where ν is an undetected neutrino [20].

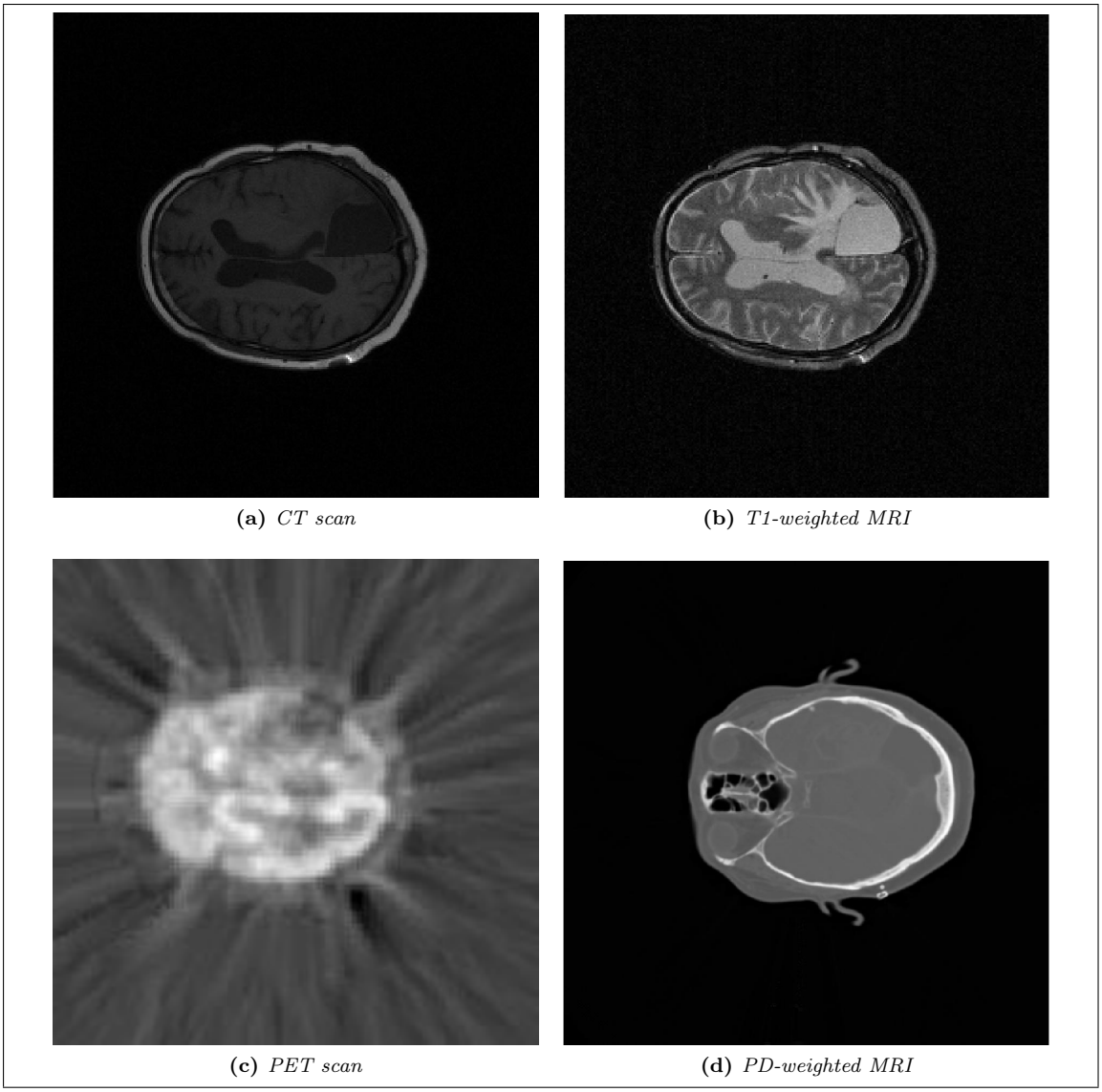
The emission of such position is the basis of *Position emission tomography*, or PET; another form of tomography used in medical imaging. The position emitted in the decay has a very short lifetime when emitted inside an electron rich material such as those found withing the human body. The position's energy will rapidly diminish in interactions with electrons (usually between 10^{-1} and 10^{-2} cm), and once it's energy is sufficiently reduced it will combine with an electron and form a positronium. This state lasts a very short time (about 10^{-10} seconds) before it annihilates; the masses of the positron and electron are converted into energy in the form of electromagnetic radiation (photons). The energy released is given by Einstein's famous formula,

$$ E = mc^2 = (m_e + m_{e^+})c^2, \tag{2.1.5} $$

where m_e and m_{e^+} are the masses of the electron and position, respectively, and c is the speed of light in vacuum (approximately 3×10^8 m/s). To conserve momentum and energy, the annihilation needs to produce a set of two photons are produced, traveling in exactly opposite directions with equal energy [20].

This property is the foundation of PET: A raioactive isotope is injected into the patient. This isotope will be incorporated with a biologically active molecule (typically fluorodeoxyglucose (FDG), a sugar). The molecules will then be concentrated in the tissue of interest before the scan takes place. Annihilations of positions emitted from the resulting decay inside the body will produce photons that can be detected once they escape the body, and the location of the emission can be calculated using the time of flight of the photons and the knowledge that the emission happened at a point on a straight line between the detections.

PET scans are often taken in combination with a CT scan, resulting in an image with good anatomical information from the CT scan as well as information about metabolism from the PET scan.



(a) *CT scan* **(b)** *T1-weighted MRI*

(c) *PET scan* **(d)** *PD-weighted MRI*

Figure 2.2: *Four brain images taken with different modalities.*

Example 2.1.1 (Example of modalities)**.** Figure 2.2 shows one slice from each of four images taken of brains using four different modalities. Note the differences in contrast and information.

2.1.4 Other image modalities

Other image modalities used in medical imaging include fMRI, ultrasound, diffusion tensor images and thermography. I have not worked much with images taken with these modalities, but the methods discussed in this thesis should be viable for these modalities as well.

2.2 Datasets

I have looked at many different datasets in the course of this project. Most of the registration has, however, been done on two datasets. The first is an MRI of a patients abdomen, focusing on the kidneys. A contrast agent (omniscan) has been administrated to observe the function of the kidneys. The dataset consists of 20 images (each of size 256-by-256-by-20) taken at different

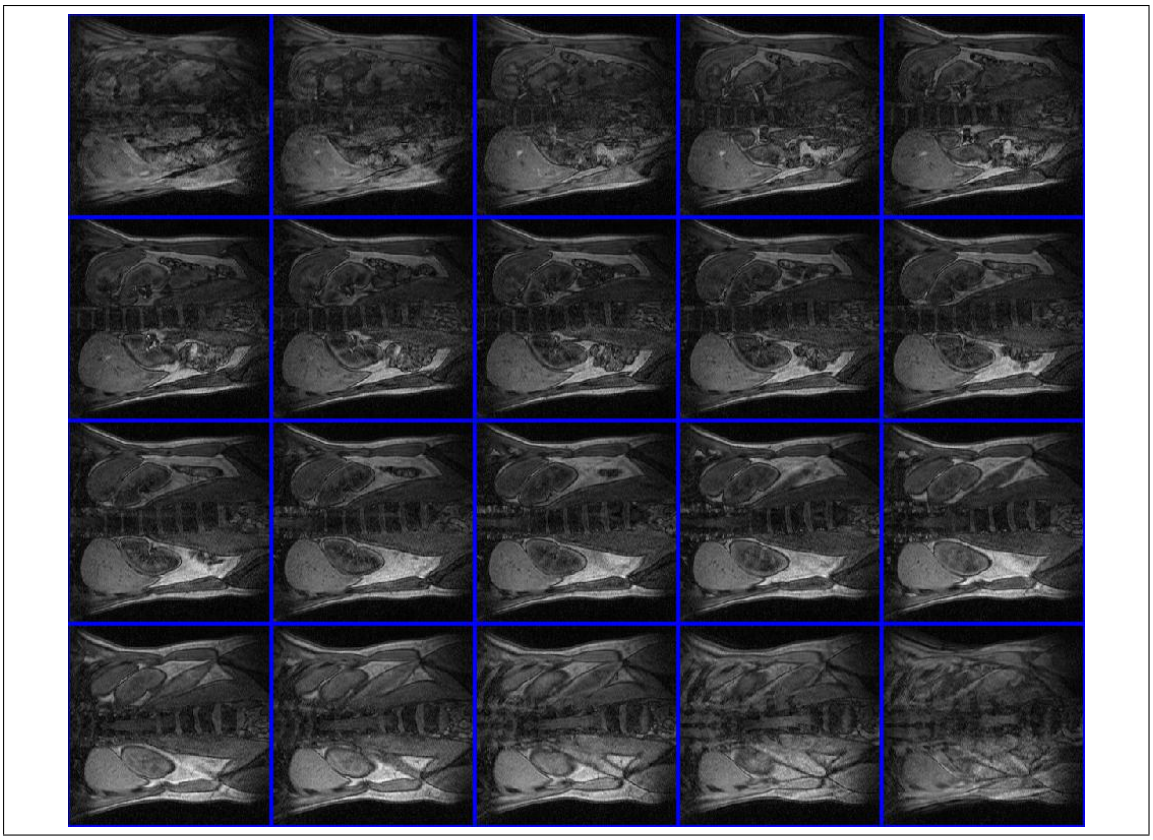


Figure 2.3: *One time slice of MRI data of kidney (abdomen).*

times. Due to respiration and patient movement, the position of structures in the different image do not correspond between images. Our task is to register the administrated images so that the displacement between images is minimal. An additional problem is the addition of a contrast agent, making a direct intensity based comparision of the images inadequate. In most of the tests I have chosen to use a cropped version (interpolated on a 128-by-128-by-20 grid) of the image showing only one of the kidneys. This is to avoid running out of memory and also to improve performance of the registration. Figure 2.3 shows a montage of the first time slice of the full dataset. Figure 2.4 shows one slice of the cropped dataset through the time steps. Both images are displayed using the `viewImage`-script from FAIR set to `'imgmontage'`.

The second dataset I have looked at is from a project at Haukeland University Hospital looking at the aging brain. Images have been taken of 100 patients over a period of many years. Each patients dataset consists of images of different modalities, such as MR images with different weighs, CT scans and PET scans. These have different spatial resolution, different alignment and display different types of information. Figure 2.5 displays a montage of slices through the head of different modalities for one patient with all the images scaled to the same dimensions (128-by-128-by-16).

2.3 Difficulties surrounding multimodal registration

When registering images the perhaps most intuitive way of measuring the similarity is by looking at the intensity values of the images. For multiple images taken with the same equipment with the same setting this might work very well, but for our discussion this is not the case. Multimodal images display different information about the object, as Figure 2.5 clearly shows. As such, methods that work well for single modality registration might not work in multimodal registration.

The differences between single- and multimodal registration is mostly reflected in the choice

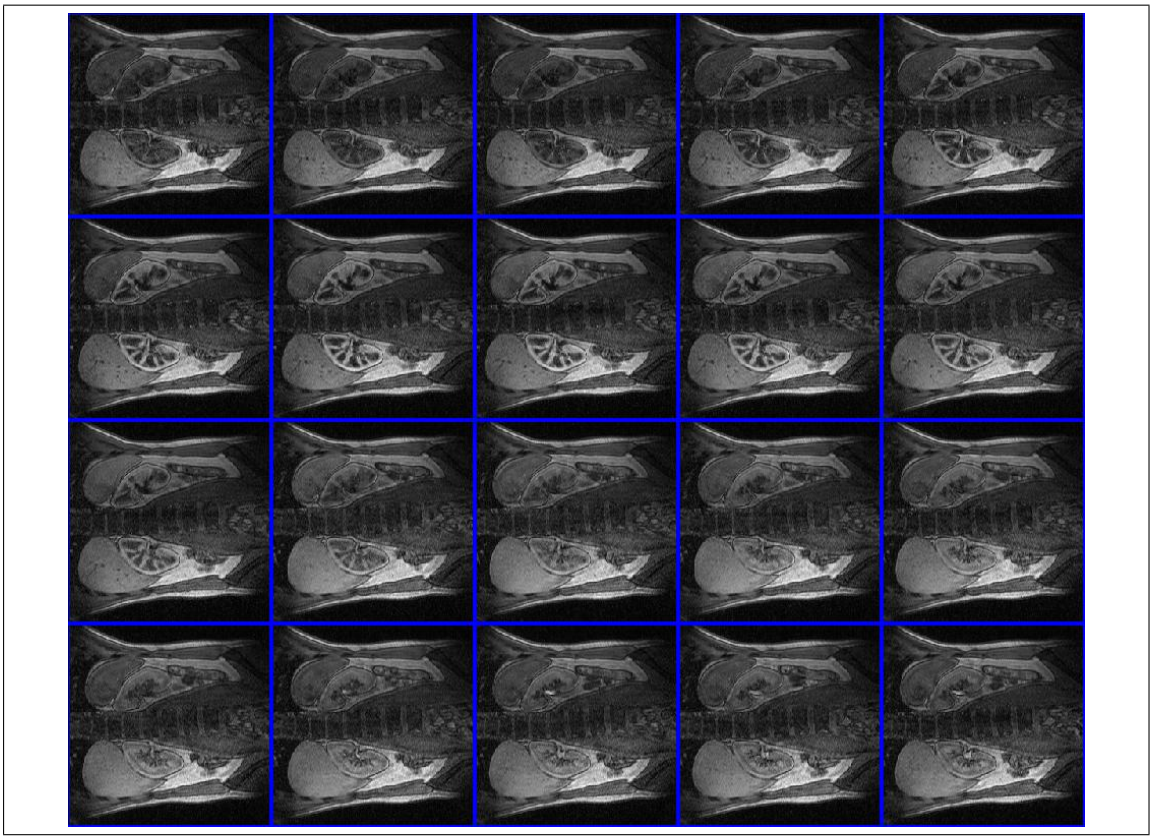
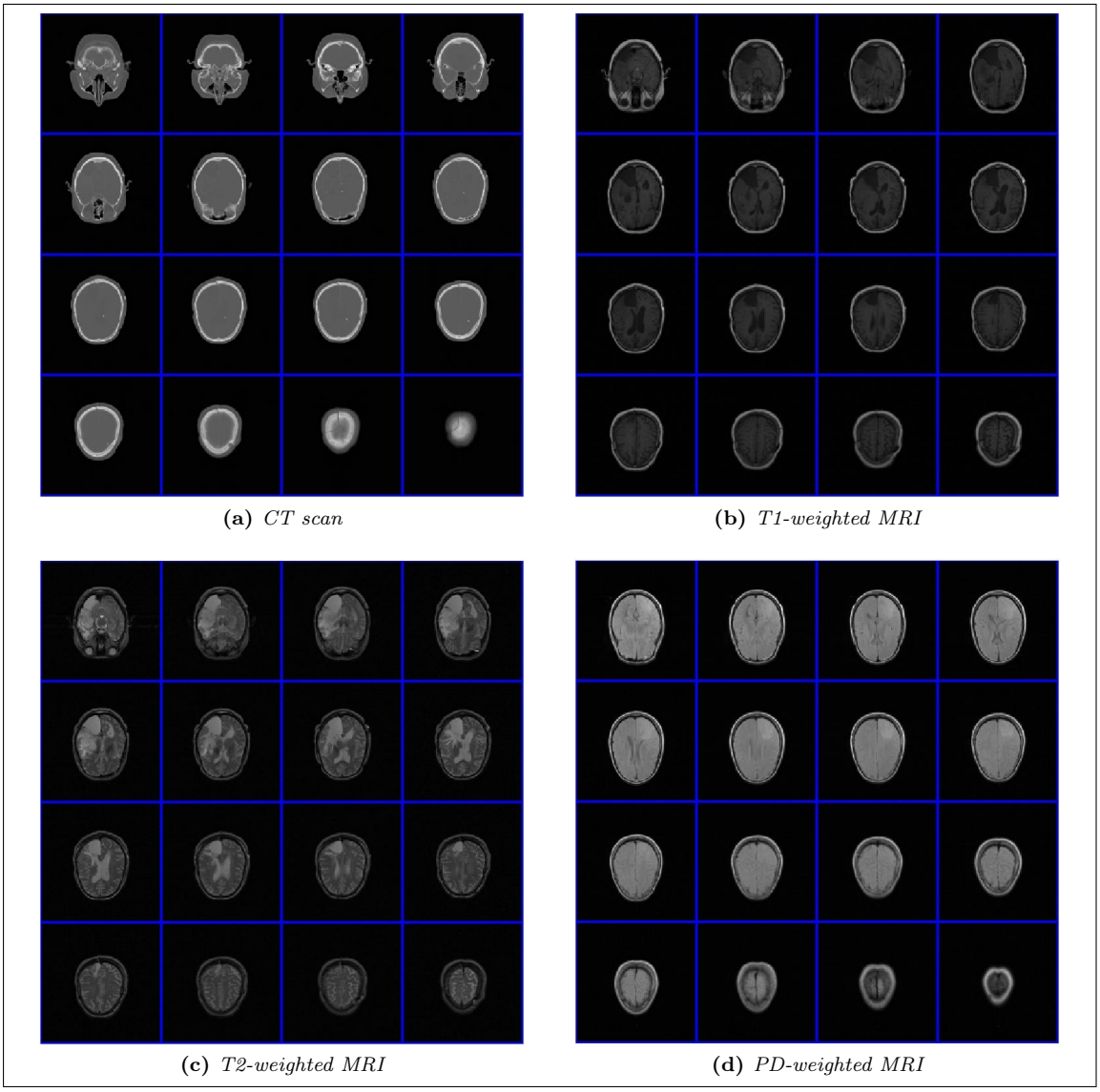


Figure 2.4: *One slice of MRI data of kidney (abdomen). Note the change in contrast due to the contrast agent which makes registration with some distance measures unviable.*



(a) *CT scan*

(b) *T1-weighted MRI*

(c) *T2-weighted MRI*

(d) *PD-weighted MRI*

Figure 2.5: *Image montages showing slices of four images of a brain taken with different modalities.*

of the distance measure seen in Equation 1.2.2.

Chapter 3

Background and Theory

The process of registering the images in FAIR can be laid out as a collection of steps. We start by finding a *multilevel representation* of the images consisting of a set of *grids* of different sizes. After choosing the necessary parameters (e.g. distance measure, regularization method and number of iterations) for the registration, a *parametric preregistration* is performed on the coarsest grid. When a sufficiently good parametric transformation has been found, the non-parametric registration starts, going from the coarsest grid to the finest. After each step the current registered grid is prolonged to fit the next level of registration and is then used as the next starting guess.

3.1 Interpolation

Interpolating is the process of constructing new data points within the range of the old data. In image registration interpolation is needed to reconstruct the image from a set of data and a grid. There are a variety of methods used in interpolation, one of the simplest and most used being *next neighbour* interpolation. Other methods include *linear* and *spline* interpolation.

3.1.1 Grids and cells

All transformations and changes in images are done via grids (see Figure 3.1 for an example), such that the actual raw data is preserved. Our data, `dataT`, will be related to points

$$x_j = \left[x_j^1, \ldots, x_j^d\right] \in \mathbb{R}^d, \quad j = 1, \ldots, n, \tag{3.1.1}$$

where d is the dimension of the data, usually 2 or 3, and n is the number of elements in `dataT`. These points can be thought of as the centers of cells in a grid that is laid over our image domain $\Omega = (\omega^1, \omega^2) \times \cdots \times (\omega^{2d-1}, \omega^{2d}) \subset \mathbb{R}^d$. Ω refers to the physical coordinates of the data. The coordinate system is usually chosen so that $\omega^1 = \omega^3 = \cdots = \omega^{2d-1} = 0$.

The grid is a way of partitioning the image domain into congruent cells. For data of size $m = \left[m^1, \ldots, m^d\right]$ we define

$$h^i = \left(\omega^{2i} - \omega^{2i-1}\right)/m^i, \qquad h = \left[h^1, \ldots, h^d\right], \tag{3.1.2}$$

$$\xi_j^i = \omega^{2i-1} + (j - 0.5)\, h^i, \qquad \xi^i = \left[\xi_1^i, \ldots, \xi_{m^i}\right] \in \mathbb{R}^{m^i}. \tag{3.1.3}$$

In this notation the index i refers to dimension and $j = \left[j^1, \ldots, j^d\right]$ refers to the data points. Specifically, the collection of points

$$x_j = \left[\xi_{j^1}^1, \ldots, \xi_{j^d}^d\right], \quad j^i = 1, \ldots, m^i, \quad i = 1, \ldots, d, \tag{3.1.4}$$

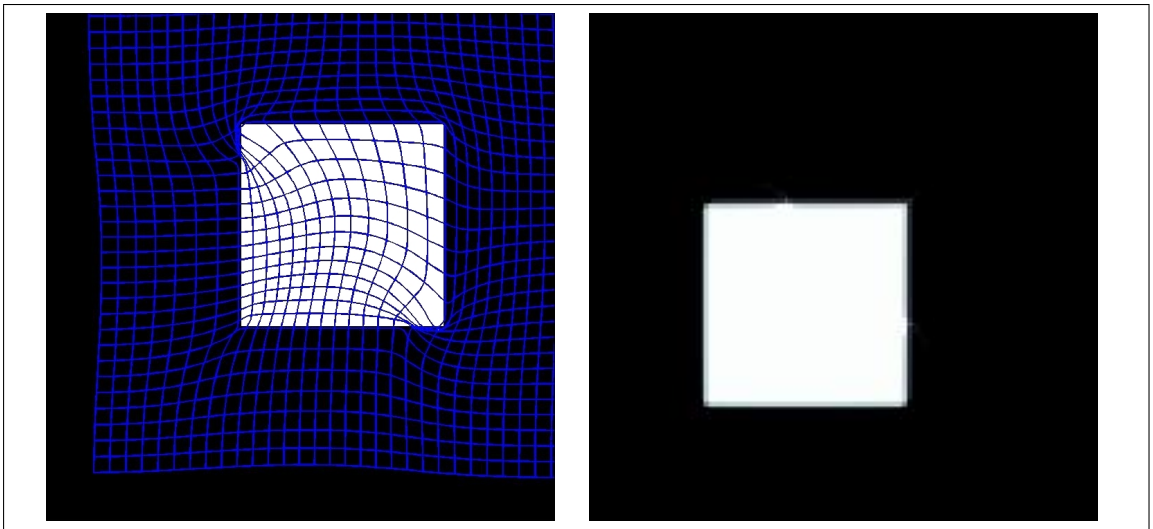


Figure 3.1: *Example of an image interpolated on a transformed grid. Figure 3.1a shows the image with the overlaid grid and Figure 3.1b shows the resulting interpolated image.*

is called a *cell-centered* grid. The intervals

$$\text{cell}_j = \left\{ x \in \mathbb{R}^d \,|\, -h^i/2 < x^i - \xi_j^i < h^i/2 \right\} \tag{3.1.5}$$

are *cells* with cell-centers x_j^c. h^i is the length of a cell in dimension i.

There are two other types of grids used in FAIR, mainly used in the approximation of derivatives. These are called *nodal* and *staggered* grids. In the $1D$-case when approximating the derivatives of a function u at a cell-centered grid point one can use the central finite-differences,

$$\partial u(x_j^c) = \frac{u(x_j^c + 0.5h) - u(k_j^c - 0.5h)}{h} + \mathcal{O}(h^2), \tag{3.1.6}$$

where h is the cell size, $(\omega_2 - \omega_1)/m$. See Figure 3.2. This requires knowledge about the function u on the nodal grid $\texttt{xc}^n$ to compute the approximation of the derivative on the cell-centered grid $\texttt{xc}^c$,

$$x_j^n = \omega^1 + jh, \quad j = 0, \ldots, n, \quad \text{and} \quad x_j^c = \omega^1 + (j - 0.5)h, \quad j = 1, \ldots, n. \tag{3.1.7}$$

With $u_j = u(jh)$ this gives us

$$\partial u(x_j^c) \approx (u_j - u_{j-1})/h. \tag{3.1.8}$$

The third grid type, the staggered grid, are required in the 2– and $3D$ approximations. In the $2D$ case we define the grids as $x_j = [j^1 h^1, j^2 h^2]$ and

$$x_j^{s,1} = x_{j^1, j^2 - 0.5}, \quad j^1 = 0, \ldots, m^1, \quad j^2 = 1, \ldots, m^2, \tag{3.1.9}$$

$$x_j^{s,2} = x_{j^1 - 0.5, j^2}, \quad j^1 = 1, \ldots, m^1, \quad j^2 = 0, \ldots, m^2. \tag{3.1.10}$$

The approximations of the derivative of $u = [u^1, u^2]$ follows as

$$\partial_1 u^1(x_{j^1 - 0.5, j^2 - 0.5}) \approx (u^1_{j^1, j^2 - 0.5} - u^1_{j^1 - 1, j^2 - 0.5})/h^1, \tag{3.1.11}$$

$$\partial_2 u^2(x_{j^1 - 0.5, j^2 - 0.5}) \approx (u^2_{j^1 - 0.5, j^2} - u^2_{j^1 - 0.5, j^2 - 1})/h^2, \tag{3.1.12}$$

$$\partial_2 u^1(x_{j^1, j^2}) \approx (u^1_{j^1, j^2 + 0.5} - u^1_{j^1, j^2 - 0.5})/h^2, \tag{3.1.13}$$

$$\partial_1 u^2(x_{j^1, j^2}) \approx (u^2_{j^1 + 0.5, j^2} - u^2_{j^1 - 0.5, j^2})/h^1. \tag{3.1.14}$$

See Figure 3.3 for a representation of the staggered grid in $2D$.

For the $3D$ case I refer to [18, 11].

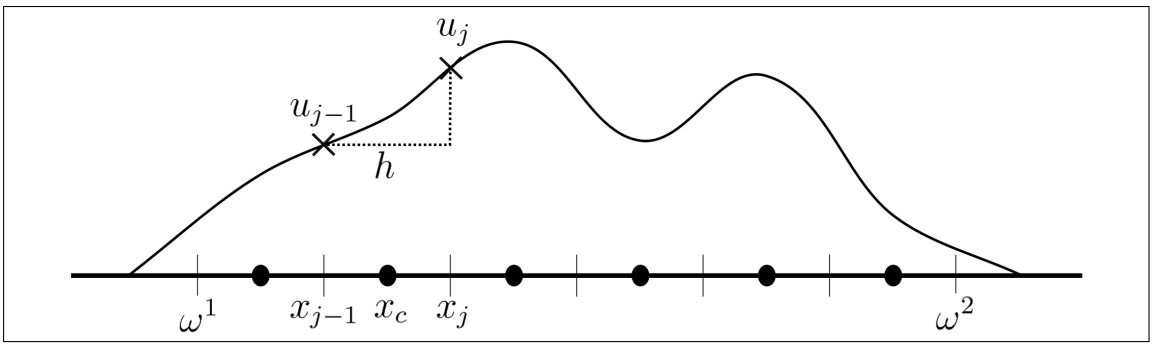


Figure 3.2: *1D Nodal grid. The figure shows the cell centers, x_j^c, and nodes, x_{j-1}^n and x_j^n, as well as the function values u at the nodes.*

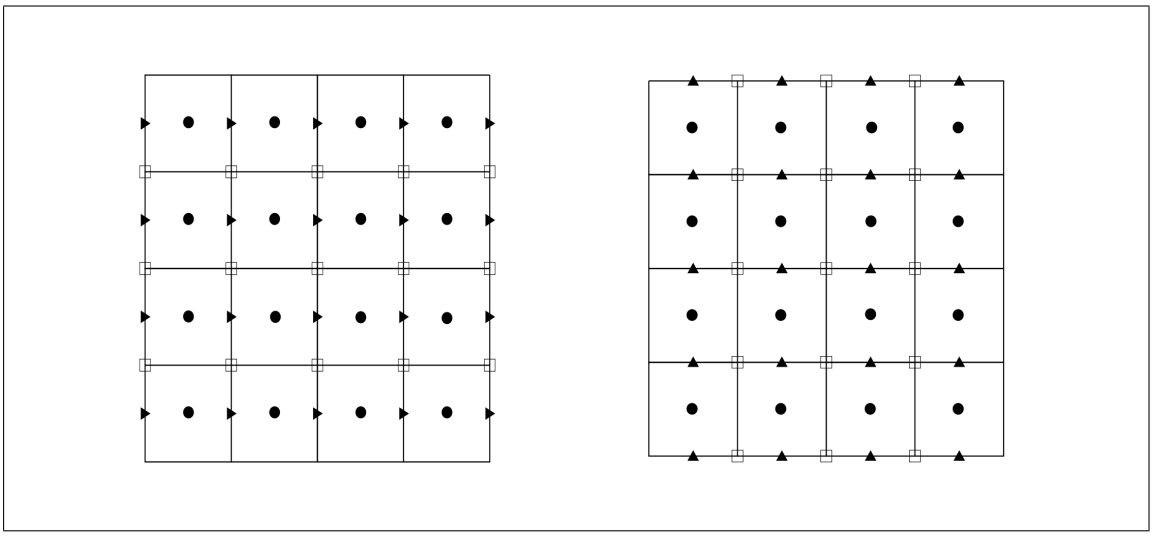


Figure 3.3: *2D grids. The figures shows cell centered (•), nodal (□) and staggered ($x_j^{s,1}$(▶) for u^1 and $x_j^{s,2}$(▲) for u^2) grids.*

3.1.2 Next Neighbour interpolation

The next neighbour interpolation is probably the simplest form of interpolation. The idea is simply to let $\mathcal{T}^{nn}(x) = \texttt{dataT}(j)$, where $x \in \text{cell}_j$. The problem of such an approach is that the resulting interpolant is discontinuous, possibly resulting in problems with the optimization.

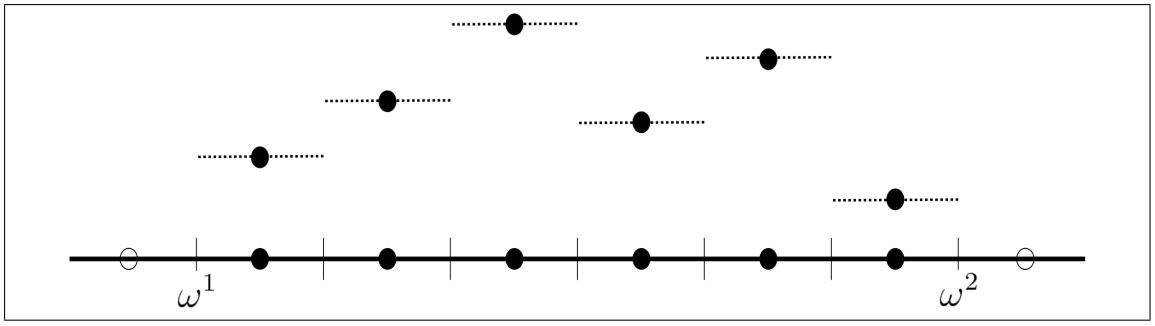


Figure 3.4: *Next neighbour interpolation in* $1D$*. The dashed line shows the resulting interpolatation.*

3.1.3 Linear interpolation

The following discussion is for one-dimensional data. For extensions to higher dimensions, see [18].

The basic idea in linear interpolation is to assign the function value at each position with a weighted sum of the function values of the neighbouring points. For this a linear mapping is used,

$$x \to x' = \left(x - \omega^1\right)/h + 0.5, \tag{3.1.15}$$

which mapps the domain $\Omega = \left(\omega^1, \omega^2\right)$ onto $\Omega' = (0.5, m + 0.5)$. This maps a point $x_j = \omega^1 + (j - 0.5)\,h$ onto j.

For an arbitrary point x, the neighbours and weights can now be found by splitting the resulting x' into an integer, p, and a remainder, ξ:

$$p = \lfloor x' \rfloor := \max\{j \in \mathbb{Z} \,|\, j \le x'\} \quad \text{and} \quad \xi = x' - p, \quad 0 \le \xi < 1. \tag{3.1.16}$$

To put this into a formula, we get

$$\mathcal{T}^{\text{linear}}(x) := \texttt{dataT}(p) \cdot (1 - \xi) + \texttt{dataT}(p + 1) \cdot \xi. \tag{3.1.17}$$

This is valid for $x_1 \le x \le x_m$ or $1 \le x' \le m$. If we add two additional data points, $(0, 0)$ and $(m + 1, 0)$, we get $0 \le p \le m$ and $0 \le \xi < 1$ for all $x \in \Omega$ or equivalently for all $x' \in [0.5, m + 0.5]$.

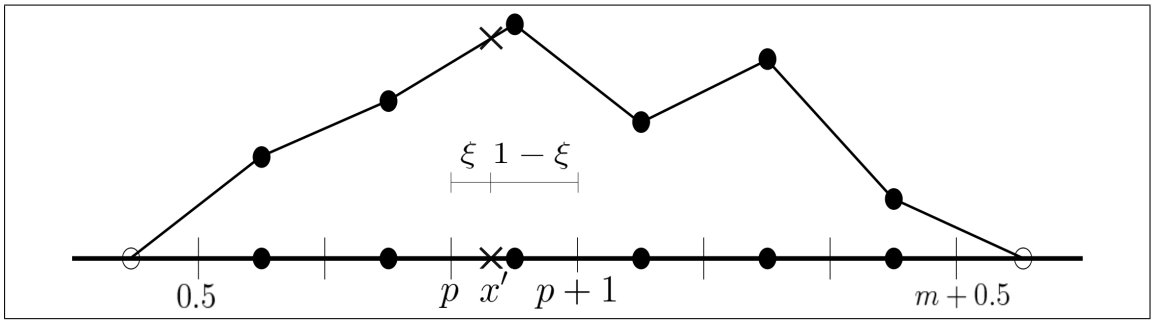


Figure 3.5: *Linear interpolation in* $1D$*.*

As seen in Figure 3.5 the linear interpolation is, unlike the next neighbour approach, continuous. However, it is not necessarily differentiable at the grid points, which is a disadvantage in comparision to e.g. a spline-based interpolation scheme. It does, however, have a much lighter computational cost, and is an ideal method if we do not need derivatives.

3.1.4 Spline interpolation

Splines are a type of functions defined piecewise by polynomials. In spline interpolation the goal is to find a function $\mathcal{T}^{\text{spline}}$ that interpolates the data and minimizes the bending energy. The bending energy is defined as

$$\mathcal{S}[\mathcal{T}] = \int_{\Omega} (\mathcal{T}''(x))^2 \, dx. \tag{3.1.18}$$

The solution to the problem

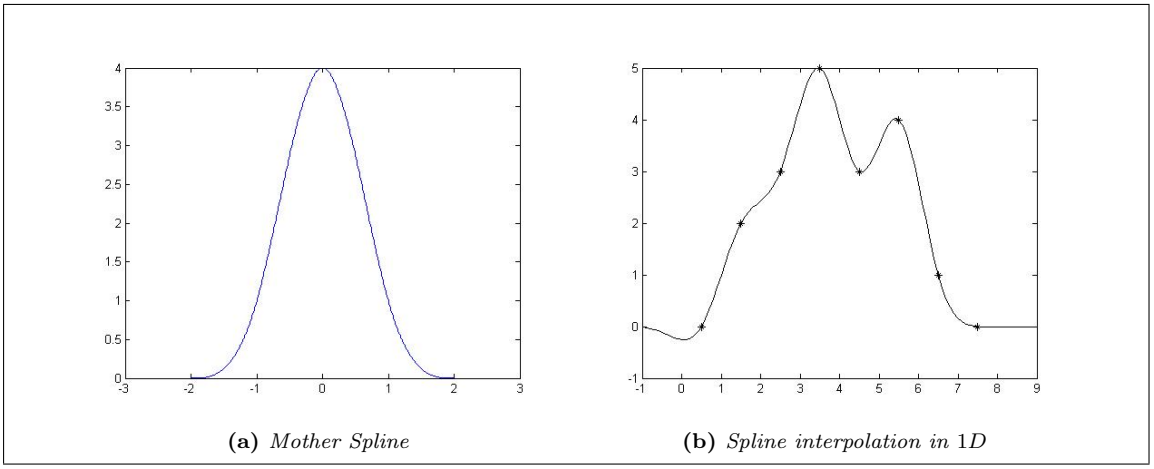
$$\min \mathcal{S}[\mathcal{T}] \quad \text{subject to} \quad \mathcal{T}(x_j) = \texttt{dataT}(j), \quad j = 1, \ldots, m, \tag{3.1.19}$$

is a cubic spline with coefficients c_j and basis functions b_j. Each basis function b_j is a translated version of a *Mother* spline b, such that $b_j(x) = b(x - j)$. We use the same mapping (Equation 3.1.15) of the cell-centered grid as in the previous section.

As basis function FAIR uses

$$b(x) = \begin{cases} (x+2)^3, & -2 \le x < -1 \\ -x^3 - 2(x+1)^3 + 6(x+1), & -1 \le x < 0 \\ x^3 + 2(x+1)^3 - 6(x+1), & 0 \le x < 1 \\ (2-x)^3, & 1 \le x < 2 \\ 0 & \text{else.} \end{cases} \tag{3.1.20}$$

This function is displayed in Figure 3.6a along with an example of data interpolated with this spline basis in Figure 3.6b.



(a) *Mother Spline* **(b)** *Spline interpolation in 1D*

Figure 3.6: *The mother spline b and an example of data interpolated with this basis.*

Using all of this information, it is possible to write out the function $\mathcal{T}$ as

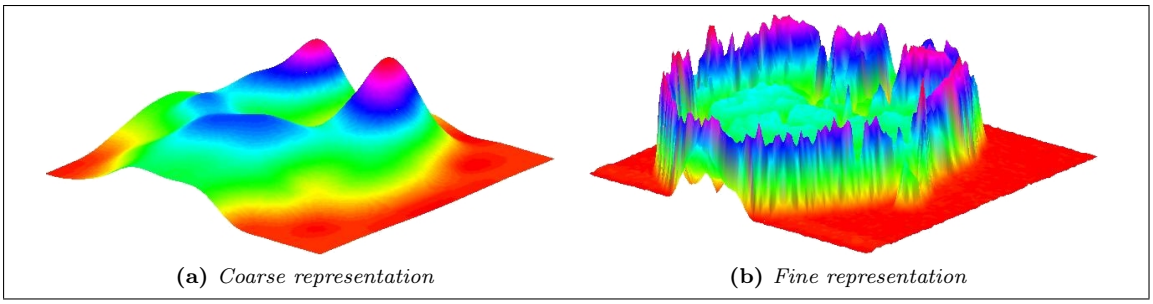
$$\mathcal{T}(x) = \mathcal{T}^{\text{spline}}(x) = \sum_{j=1}^{m} c_j b_j(x). \tag{3.1.21}$$

To achieve this a way of computing the coefficients c_j is needed. If we write $c = [c_1, \ldots, c_m]^{\mathsf{T}}$ and expand Equation 3.1.21 at $x_j = j$ we can simplify to

$$\mathcal{T}(x_j) = \texttt{dataT}(j) = \sum_{k=1}^{m} c_k b_k(j) = [b_1(j), \ldots, b_m(j)]\, c, \quad j = 1, \ldots, m. \tag{3.1.22}$$

We can now gather all of our function values $T(x_j)$ in $\mathcal{T}(\texttt{xc}) = [\mathcal{T}(x_1), \ldots, \mathcal{T}(x_m)]^{\mathsf{T}}$, and we get

$$\texttt{dataT} = T(\texttt{xc}) = [b_1(\texttt{xc}), \ldots, b_m(\texttt{xc})]\, c = B_m c, \tag{3.1.23}$$



(a) *Coarse representation* **(b)** *Fine representation*

Figure 3.7: *Multilevel surface plots. Figure 3.7a shows a coarse representation of the data whereas Figure 3.7b shows a finer representation. Note the reduced number of local minima on the coarse representation compared to the fine.*

where B_m is the matrix

$$B_m = \begin{bmatrix} 4 & 1 & & 0 \\ 1 & \ddots & \ddots & \\ & \ddots & \ddots & 1 \\ 0 & & 1 & 4 \end{bmatrix}. \tag{3.1.24}$$

This gives us a simple way of finding the coefficients c.

3.1.5 Multilevel representation

An important concept is the use of a multilevel setting in the registration process. The idea is to start out registering on a coarse version of our images before we gradually improve the resolution until the result is good enough on the finest level - Increasing the level increases the level of detail. There are a number of advantages to this. One is computational efficiency; computing on a coarse level is cheaper, and the registration will require fewer iterations on the finer levels.

Another important feature is that on the coarse grid, only global characteristics of the image will be present. This is an advantage because in our optimization, we look for global minima in our objective function. If the image is too detailed and the starting point is far from the optimal transformation, there is a significant chance that our optimization will return a false solution at local minima. Using a multilevel registration process reduces this problem. Figure 3.7 shows surface plots of two representations of an image, one finer and one coarser.

Example 3.1.1 (2D Multilevel)**.** For $\texttt{dataT} \in \mathbb{R}^{m^1,m^2}, m^1 = m^2 = 2^L$ for some $L \in \mathbb{N}$ we have a multilevel representation $\{\texttt{T}^l, l = 0,\dots,L\}$, where $\texttt{T}^L = \texttt{dataT}$ and $\texttt{T}^{l-1}$ is given as

$$\begin{aligned}\texttt{T}^{l-1} = (\texttt{T}^l(1:2:m^1-1,1:2:m^2-1) + \texttt{T}^l(2:2:m^1,1:2:m^2-1) \\ + \texttt{T}^l(1:2:m^1-1,2:2:m^2) + \texttt{T}^l(2:2:m^1,2:2:m^2))/4\end{aligned} \tag{3.1.25}$$

for l going from L to 1.

We may also use splines to interpolate on a coarser grid, $\texttt{xc}^l$ for l going from L to 1. Using splines will again give a smoother representation at the cost of computation time and memory.

In my thesis I have primarily used a coarse-to-fine representation based on linear interpolation. The reason is that the spline-based method is very memory- and time consuming for larger datasets. See Section 3.1.4 for more on splines.

3.2 Transformations

The choice of allowable transformations is an important step in the process of registering images. In theory we can transform our template image into anything we would like by choosing the right

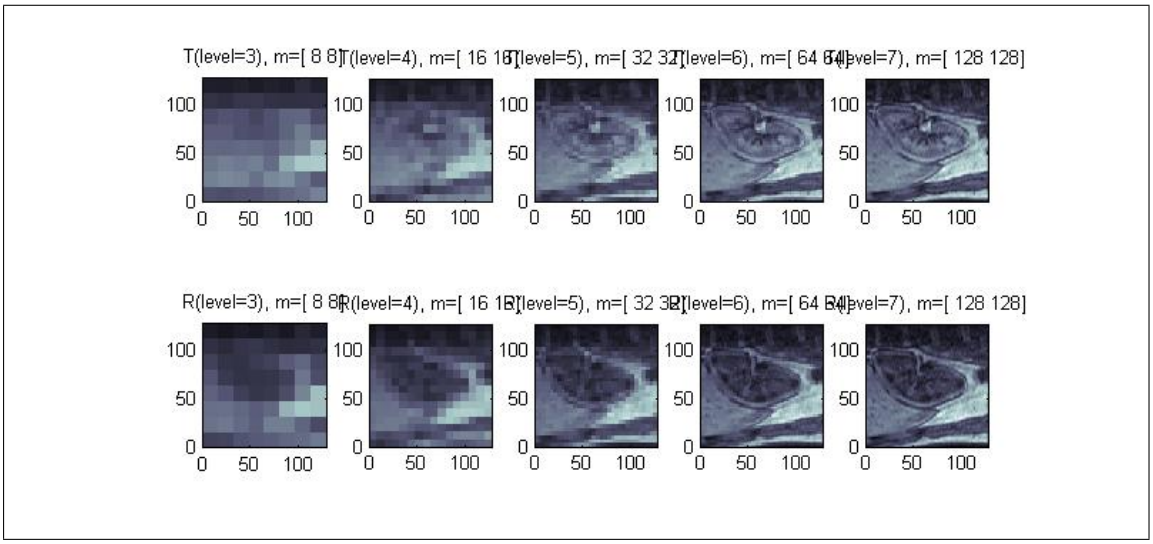


Figure 3.8: *Example of multilevel representation of data. The image shows the graphical output from FAIR's* `getMultilevel` *tool.*

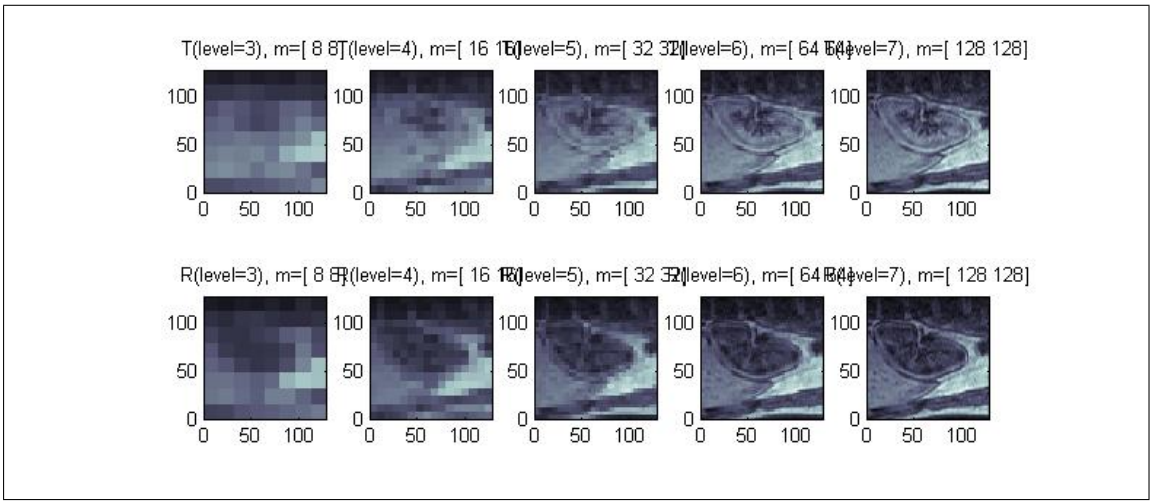


Figure 3.9: *Example of spline-based multilevel representation of data. The image shows the graphical output from FAIR's* `getMultilevel` *tool.*

transformation, from a simple translation in one direction to something not even closely resembling the original image.

The concept of grids is of importance when we start dealing with transformations: If we perform transformations of the original image we might run into problems if we wish to reconstruct the image from a transformation that is not one-to-one. See Section 3.1 for more on grids.

FIGURE (Image transformed in two ways)

Since the original image contains information we would like to keep through the transformation process, we need to restrict the allowable transformations in some way. The transformations we use can be divided into two types, parametric and nonparametric, and depending on which type we use we have different ways of restricting the transformation.

3.2.1 Parametric transformations

One way of expressing a parametric transform is as a function $y : \mathbb{R}^d \to \mathbb{R}^d$ where the components are linear combinations of basis functions q^l with coefficients basically being the parameters w_l [18]. For instance, the simple linear function $y : \mathbb{R} \to \mathbb{R}$ with $y = w_1 x + w_2$ is parameterized by $w = [w_1, w_2]^\top$ and the basis functions $q^1(x) = x$ and $q^2(x) = 1$. Setting $Q(x) = \left[q^1(x), q^2(x)\right]$ gives us the simple form $y = Q(x)w$. As an introduction to parametric registration, see for instance [24].

Affine transformations

A popular class of parametric transformations are the affine linear transformations. An affine linear transformation has the possibilities of translating, rotating, shearing and individual scaling. In our notation, an image in $\mathbb{R}^2$ has the components

$$\begin{aligned} y^1 &= w_1 x^1 + w_2 x^2 + w_3, \\ y^2 &= w_4 x^1 + w_5 x^2 + w_6 \end{aligned} \tag{3.2.1}$$

where x^1 and x^2 are the original and y^1 and y^2 are the transformed coordinates in the image. Writing the parameters in $w = [w_1, \ldots, w_6] \in \mathbb{R}^6$ and with

$$Q(x) = \begin{bmatrix} x^1 & x^2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x^1 & x^2 & 1 \end{bmatrix} \tag{3.2.2}$$

we get the form $y = Q(x)w$. For an image with $d = 3$ we get 12 parameters.

One interesting subclass of the affine linear transformations are the rigid transformations. These allow only for translation and rotation of the image, and are on the form

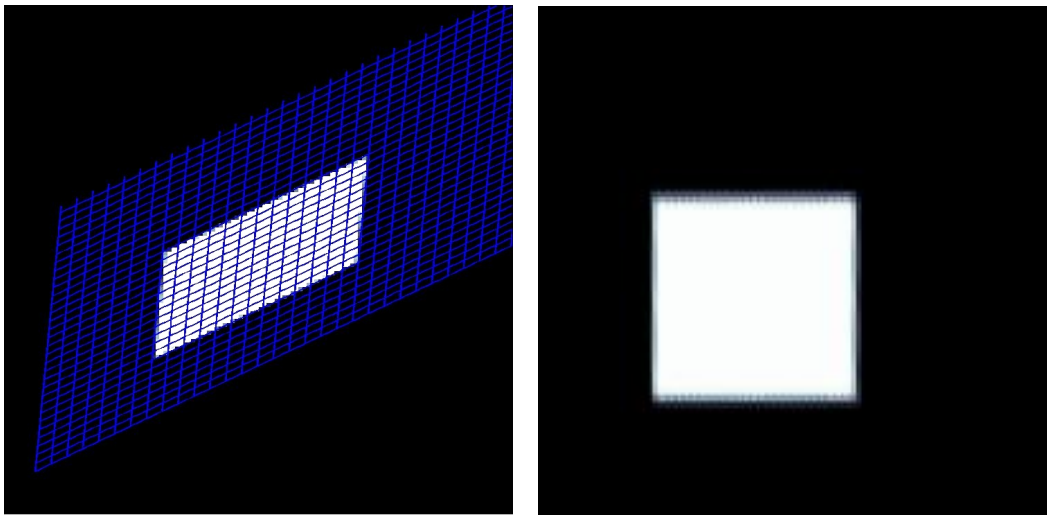
$$\begin{aligned} y^1 &= cos(w_1)x^1 - sin(w_1)x^2 + w_2, \\ y^2 &= sin(w_1)x^1 + cos(w_1)x^2 + w_3. \end{aligned} \tag{3.2.3}$$

where $w = [w_1, w_2, w_3]^\top$ parameterizes the transformation. We also see that this corresponds to Equation 3.2.1 with parameters

$$w = \begin{bmatrix} cos(w_1) & -sin(w_1) & w_2 & sin(w_1) & cos(w_1) & w_3 \end{bmatrix}^\top. \tag{3.2.4}$$

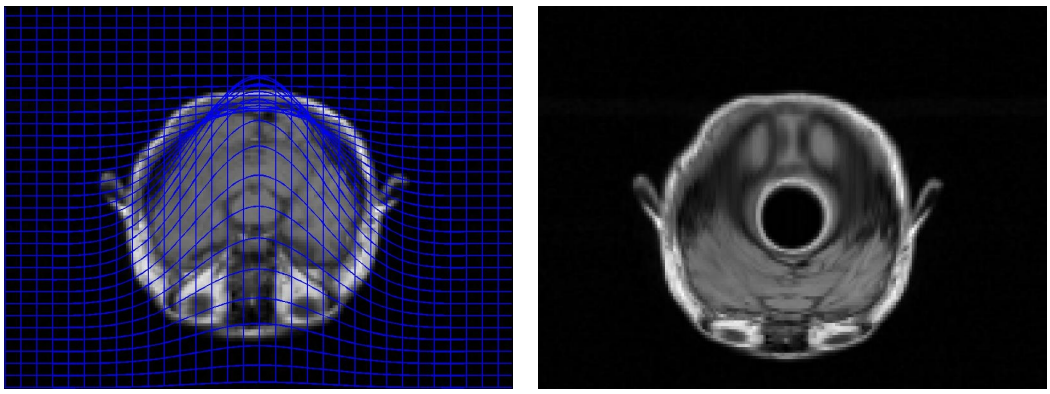
Spline-based transformations

Another interesting family of transformations is that of spline-based transformations. Where the affine transformations can be parametrized by a relative small number of parameters (for a $2D$ image we have 6 parameters for an affine transformation, or 3 if we're restricted to rigid transformations), a spline-based approach gives us the possibility of having as many parameters or degrees of freedom as we like. This gives us more flexibility in the range of transformations possible, but we also need to watch out for unreasonable transformations, as seen in Figure 3.11.



(a) *Original image with transformed grid* **(b)** *Transformed image*

Figure 3.10: *Linear affine transformed grid overlaid an image and the resulting transformed image.*



(a) *Original image with transformed grid* **(b)** *Transformed image*

Figure 3.11: *Unreasonable grid obtained using a spline-based transformation. Note how the grid folds over itself, creating a distorted transformation of the image.*

For a two-dimensional image the transformation is governed by the following equations:

$$y^1 = x^1 + \sum_{j^1=1}^{p^1} \sum_{j^2=1}^{p^2} w^1_{j^1,j^2} b^{j^1}(x^1) b^{j^2}(x^2) \tag{3.2.5}$$

$$y^2 = x^2 + \sum_{j^1=1}^{p^1} \sum_{j^2=1}^{p^2} w^2_{j^1,j^2} b^{j^1}(x^1) b^{j^2}(x^2) \tag{3.2.6}$$

where b^j are the splines (see Section 3.1.4), $p = \left[p^1, p^2\right]$ are the number of coefficients in the spline expansion and w^1 and w^2 are the coefficients for the components of y. I have not used spline-based transformation to a large extent; instead I have used non-parametric transformations to obtain more freedom in transforming the images (see Section 3.2.3 and 3.3.2 for more on non-parametric transformations).

3.2.2 Regularization

One important aspect of image registration is the ill-posedness of the problem. A problem is considered well-posed if it has the following properties:

1. A solution exists
2. The solution is unique
3. The solution depends continuously on the data. [13]

A problem that is not well-posed is considered ill-posed. Image registration is inheritably ill-posed because of the lack of unique solutions in most cases [7].

Consider a symmetric figure, for instance a square, and attempt to register a translated version of the same square. One solution would then be to simply translate the template over the reference. However, if we also rotate by 90 degrees we have a similar solution. Another example would be to attempt to register an area of (near) constant intensity. In this area every change will yield the same results.

One way of attempting to remedy this lack of uniqueness is by adding a regularizer, $\mathcal{S}$, to the registration process. The goal in adding a regularizer is to modify the registration problem such that it becomes solvable, and ideally well-posed. In practice the well-posedness might be impossible to obtain as there will still be local minima [18].

Most variants of regularization used in image registration are cost functions based on the L_2-norm of the derivative of the displacement $u = y - y^{ref}$, where y^{ref} is the preferred solution. In this way, moving away from y^{ref} will add a penalty to the cost function.

We will use regularizers of the form

$$\mathcal{S}[u] = \frac{\alpha}{2} \int_\Omega |\mathcal{B}[u]|^2 \, dx, \tag{3.2.7}$$

where $\mathcal{B}$ is a differential operator, $|\cdot|$ is a Euclidian norm and α is a regularization parameter deciding how big an impact the regularization is to have on the registration process.

Example 3.2.1 (Curvature Regularizer in two dimensions)**.** The diffusion regularizer is based on the second order derivatives. Let $d = 2$ and $y = \left[y^1, y^2\right]^\top$. The choice

$$\mathcal{B} = \begin{bmatrix} \Delta & 0 \\ 0 & \Delta \end{bmatrix}, \tag{3.2.8}$$

where $\Delta y^i = \partial_{1,1} y^i + \partial_{2,2} y^i$ results in the regularizer [6]

$$\mathcal{S}[y] = \frac{1}{2} \int_\Omega |\mathcal{B}[y]|^2 \, dx = \frac{1}{2} \int_\Omega (\Delta y^1)^2 + (\Delta y^2)^2 \, dx. \tag{3.2.9}$$

This choice of regularizer does not penalize affine linear transformations since

$$\mathcal{S}[Ax + b] = 0 \quad \text{for all} \quad A \in \mathbb{R}^{d\times d}, b \in \mathbb{R}^d. \tag{3.2.10}$$

Example 3.2.2 (Elastic operator in two dimensions)**.** The elastic regularizer is the elastic potential measuring the energy introduced by deforming an elastic material. It was introduced by Broit [2] and is one of the most commonly used regularizers in image registration.

Again, let $d = 2$ and $y = \left[y^1, y^2\right]^\top$. The divergence is an interesting differential operator that tells us about changes in the volume. The choice of

$$\mathcal{B} = \begin{bmatrix} \sqrt{\mu}\,\nabla & 0 \\ 0 & \sqrt{\mu}\,\nabla \\ \sqrt{\lambda+\mu}\,\partial_1 & \sqrt{\lambda+\mu}\,\partial_2 \end{bmatrix}, \tag{3.2.11}$$

where $\nabla = [\partial_1, \partial_2]^\top$, gives us the elastic potential

$$\mathcal{S}[y] = \frac{\alpha}{2}\int_\Omega |\mathcal{B}[y]|^2\,dx \tag{3.2.12}$$

$$= \frac{\alpha}{2}\int_\Omega \mu\langle\nabla y, \nabla y\rangle + (\lambda+\mu)(\nabla\cdot y)^2\,dx. \tag{3.2.13}$$

μ and λ are the so called Lamé constants. Note that for $\lambda = -1$ and $\mu = 1$ this turns into the diffusion regularizer,

$$\mathcal{S}[y] = \frac{\alpha}{2}\int_\Omega \langle\nabla y, \nabla y\rangle\,dx = \frac{\alpha}{2}\int_\Omega (\partial_1 y^1)^2 + (\partial_2 y^1)^2 + (\partial_1 y^2)^2 + (\partial_2 y^2)^2\,dx. \tag{3.2.14}$$

Example 3.2.3 (Thin-plate-spline regularization in two dimensions)**.** A third option is the Thin-plate-spline regularization, or TPS for short. Like the curvature regularizer this is also a second order regularizer, but here the energy is obtained from

$$\mathcal{B} = \mathcal{I}_2 \otimes \mathcal{B}^1, \quad \text{with} \quad \mathcal{B}^1 = \begin{bmatrix} \partial_1^2 \\ \sqrt{2}\partial_1\partial_2 \\ \partial_2^2 \end{bmatrix}, \tag{3.2.15}$$

and is given as

$$\mathcal{S}[y] = \int_{\mathbb{R}^2} |\mathcal{B}y|^2\,dx. \tag{3.2.16}$$

See [18] for a discussion on boundary conditions.

The choice of regularizer is not always straightforward. In the registrations I have performed I have used all of the above mentioned regularization schemes, depending on which gave the best results in each case. See for instance [6] for more on regularization.

3.2.3 Nonparametric transformations

In nonparametric transformation the registration process is entirely governed by the regularization. In theory, any transformation will be possible, so the regularization is vital in preserving the important structure in the image. See more in Section 3.3.2 for details on the optimization process.

3.2.4 Landmark-based registration

A different take on registration is the Landmark-based approach. The idea here is to find *landmarks*, specific points in the image, that we wish to focus on. The registration then becomes a way of matching landmarks in the template image with the corresponding landmarks in the reference,

or at least to minimize the distance between marks. Landmark-based registration can work well with sufficent landmarks placed in correct positions, but this is also one of the main disadvantages of this method: There are as of yet no good methods of automatically placing landmarks, so in practice placing landmarks has to be done in a manual or semi-manual way. Some work is non the less being done on this, see for example [9]. The manual work is time-consuming and often difficult without prior knowledge of the objects being studied. For instance it can be difficult for someone without medical or anatomical knowledge of the brain to place corresponding markers.

Another option is to attach physical landmarks to the patient before the images are taken. This makes registration, but has the disadvantage of being invasive.

The underlying idea is rather simple. For landmarks $t_j = \left[t_j^1, t_j^2\right]$ in the template, we have a corresponding landmark $r_j = \left[r_j^1, r_j^2\right]$ for $j = 1, \ldots, n$, where n is the number of landmarks. Our goal is to find a transformation $y : \mathbb{R}^2 \to \mathbb{R}^2$ s.t.

$$y(r_j) = t_j \quad \text{for} \quad j = 1, \ldots, n. \tag{3.2.17}$$

The distance between points is calculated as the Euclidian distance,

$$\mathcal{D}^{\mathrm{LM}}[y] = \sum_{j=1}^{n} \|y(r_j) - t_j\|^2. \tag{3.2.18}$$

In some cases, a perfect match might not be viable, and we need to relax the demand for a fit between landmarks in the reference and template images. In this case we get a minimization problem,

$$\mathcal{D}^{\mathrm{LM}}[y] = \sum_{j=1}^{n} \|y(r_j) - t_j\|^2 \stackrel{!}{=} \min. \tag{3.2.19}$$

We might also add some kind of regularizing term and attempt to minimize the combination of the regularization and the distance between corresponding landmarks. The regularization might for example be restricting the transformation to specific types (e.g. affine), or by adding some form of energy measure for the transformation (e.g. elastic regularization).

Affine linear Landmark-based registration

For an affine linear transformation (see Section 3.2.1 for details) we have

$$\begin{bmatrix} y^1 \\ y^2 \end{bmatrix} = \begin{bmatrix} w_1 & w_2 \\ w_4 & w_5 \end{bmatrix} \begin{bmatrix} x^1 \\ x^2 \end{bmatrix} + \begin{bmatrix} w_3 \\ w_6 \end{bmatrix}. \tag{3.2.20}$$

From this and Equation 3.2.18 we get

$$\mathcal{D}^{\mathrm{LM}}[y] = \sum_{j=1}^{n} \|y(r_j) - t_j\|^2 = \sum_{j=1}^{n} (y^1(r_j) - t_j^1)^2 + \sum_{j=1}^{n} (y^2(r_j) - t_j^2)^2, \tag{3.2.21}$$

which makes our optimization problem decoupled.

Example 3.2.4 (Affine linear landmark based registration)**.** As we see in Figure 3.12, affine linear transformations are not able to match the landmarks perfectly when we have more conditions (landmarks) than parameters (for affine we have $3d$ parameters).

Quadratic Landmark-based registration

To get a better match than in the affine linear case, we might try a different transformation space. One example is the quadratic family of transformations,

$$y^i = w_1^i + w_2^1 x^1 + w_3^i x^2 + w_4^i (x^1)^2 + w_5^i (x^2)^2 + w_6^i x^1 x^2, \quad i = 1, \ldots, d. \tag{3.2.22}$$

Example 3.2.5 (Quadratic landmark based registration)**.** In this case the match between corresponding landmarks is perfect. However, the transformation has not preserved the structure of the image, and the results are thus meaningless. Result shown in Figure 3.13.

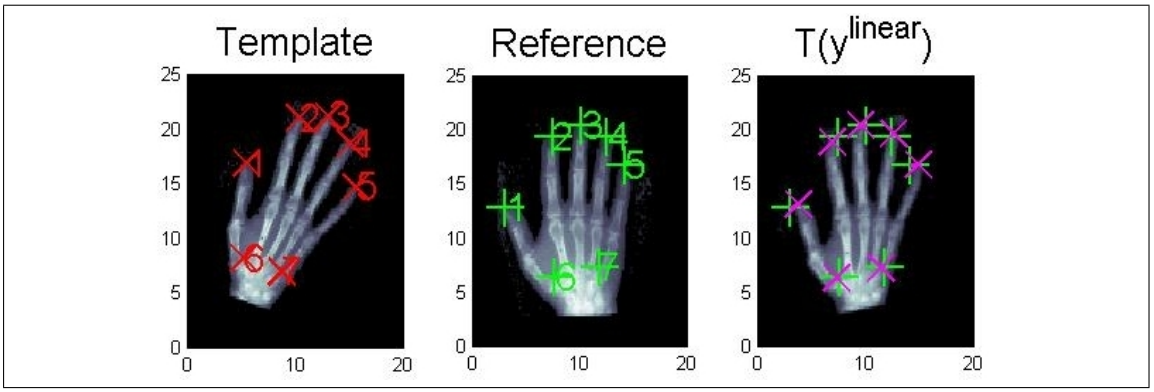


Figure 3.12: *Linear affine landmark based registration. Left: Template image with landmarks. Middle: Reference image with landmarks. Right: Transformed template image with transformed landmarks*

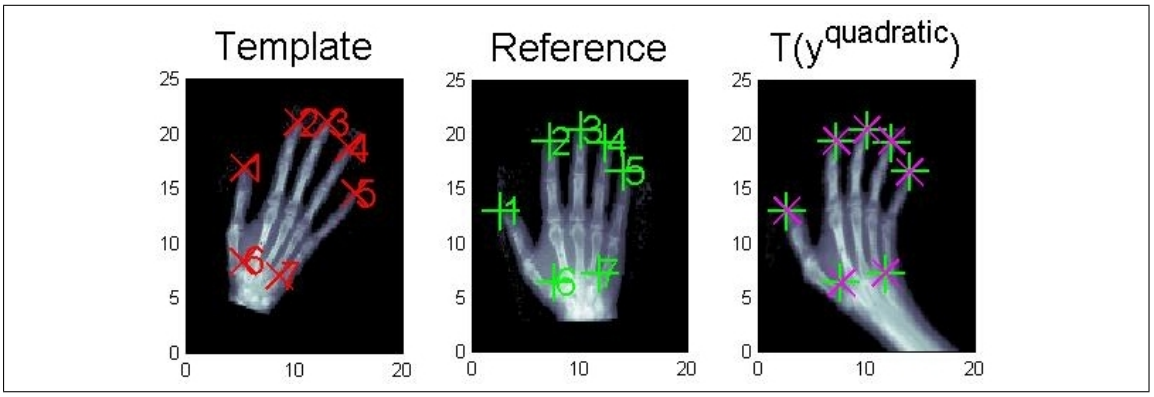


Figure 3.13: *Quadratic landmark based registration. Left: Template image with landmarks. Middle: Reference image with landmarks. Right: Transformed template image with transformed landmarks*

3.3 Optimization

As with any numerical problem, we need an optimization method to arrive at a solution. Optimization is the process of selecting the best available solution from some set of possible solutions. In our case, we want to get the best possible fit from our registration, and this means minimizing the functional

$$\mathcal{J}[y] = \mathcal{D}[\mathcal{T}[y], \mathcal{R}] + \alpha\mathcal{S}[y - y^{ref}] \xrightarrow{y} \min, \tag{3.3.1}$$

where $\mathcal{J}$ is our objective function, $\mathcal{D}$ is the distance measure, $\mathcal{S}$ is the regularization term and y is the transformed grid [18]. Depending on whether we use parametric or nonparametric transformations, the transformed grid might be seen as a function of the original grid and the parameters of the transformation.

3.3.1 Parametric optimization

Our discretization of the parametric transformation yields us the optimization problem

$$J[y] = D(T(y(\mathtt{xc}, \mathtt{wc})), R(\mathtt{xc})) + S(\mathtt{wc}) \xrightarrow{\mathtt{wc}} \min, \tag{3.3.2}$$

where $\mathtt{yc} = y(\mathtt{xc}, \mathtt{wc})$ is the transformation and $T(\mathtt{yc})$ is the transformed template image. `wc` and `xc` describe the parameters and the current grid.

The idea is twofold: To find a direction in which we search for a minimum and to find the ideal step length in that direction. Starting with y^k, we compute $y^{k+1} = y^k + \alpha_k p_k$, where the (positive) scalar α is the step length and p_k is the direction. It is usually required (the FAIR algorithm is no exception) that p_k is a descent direction, meaning $p_k^\top \nabla J_k < 0$ [19].

We use a Gauss-Newton type algorithm to find our decent direction. This is a variation of solving the classical Newton's method, which is based on finding the extrema of the Taylor expansion truncated after the second order term [8],

$$J(\mathtt{wc} + \mathtt{dw}) \approx \hat{J}(\mathtt{wc} + \mathtt{dw}) = J + dJ\mathtt{dw} + \frac{1}{2}\mathtt{dw}^\top H \mathtt{dw}. \tag{3.3.3}$$

Equation 3.3.3 attains its extreme values when `dw` solves $H\mathtt{dw} = -dJ$. Solving this straightforward using the exact Hessian matrix,

$$H(f) = \begin{bmatrix} \dfrac{\partial f}{\partial x_1^2} & \dfrac{\partial f}{\partial x_1 \partial x_2} & \cdots & \dfrac{\partial f}{\partial x_1 x_n} \\ \dfrac{\partial f}{\partial x_2 \partial x_1} & \dfrac{\partial f}{\partial x_2^2} & \cdots & \dfrac{\partial f}{\partial x_2 x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \dfrac{\partial f}{\partial x_n \partial x_1} & \dfrac{\partial f}{\partial x_n \partial x_2} & \cdots & \dfrac{\partial f}{\partial x_n^2} \end{bmatrix}, \tag{3.3.4}$$

gives us the standard Newton's method.

The Gauss-Newton approach has the advantage of using an approximation of the Hessian. This saves us the process of having to recompute the Hessian every iteration.

With the assumption that our objective function J can be written as $J(\mathtt{wc}) = \psi(r(\mathtt{wc}))$, we approximate the Hessian of J, H, with $dr^\top d^2\psi dr$, which is positive semi-definite. $\hat{J}$ is thus convex and `dw` is a descent direction since $\mathtt{dJ}^\top\mathtt{dw} < 0$. Another advantage, in particular for image registration, is that this lets us neglect higher order derivatives of the inner function dr which depends on the noisy data [18].

When we have found the descent direction we need to find out how far in this direction we wish to go. Starting with $t = 1$, $y^t = y + t \cdot dy$ is computed. If this gives us a sufficient decrease in

the objective function we can continue. If not, t is replaced by $\frac{1}{2}t$. The test we use is called the *Armijo* criterion:

$$J(y^t) < J(y) + \texttt{tol}t(dJ^\top y), \tag{3.3.5}$$

where `tol` is a tolerance value between 0 and 1 [19].

3.3.2 Nonparametric optimization

When we have a nonparametric transformation, the objective function is basically

$$J(\texttt{yc}) = D(T(\texttt{yc}), R(\texttt{xc})) + S(\texttt{yc} - \texttt{yRef}). \tag{3.3.6}$$

The process is very similar to the one we had in the parametric case 3.3.1, the main difference being that we can't express the transformation in terms of parameters. For details on the differences I refer to the FAIR book [18].

The process of non-parametric registration is much slower than the affine linear counterpart due to the large amount of information that has to be processed. To improve this situation FAIR performs a parametric preregistration of the data on the coarsest level. The preregistration itself is usually very quick, but the improved starting guess for the non-parametric registration has the potential to improve performance, both concerning time and registration results.

3.3.3 l-BFGS

When dealing with large data sets, particularly $3D$ data, memory may become an issue. This may be countered with completing the registration on a coarser level of data, but this will affect results since finer detail will not be aligned. Another option is to use a different optimization process.

As seen in Equation 3.3.3 and the discussion following, one central part of the optimization is the approximation of the Hessian. The idea behind the BFGS-method (named after Broyden, Fletcher, Goldfarb and Shanno) is to replace the updating of the Hessian with an approximation based on only the gradient information. The version of BFGS implemented in FAIR is a limited memory BFGS scheme; l-BFGS.

The approximation to the inverse Hessian becomes (see [18] and especially [19] for details)

$$\hat{H}_{k+1} = (I - \rho_k s_k z_k^\top)\hat{H}_k(I - \rho_k z_k s_k^\top) + \rho_k s_k s_k^\top), \tag{3.3.7}$$

where

$$s_k = y^{k+1} - y^k, \quad z_k = \nabla J(y^{k+1}) - \nabla J(y^k) \quad \text{and} \quad \rho_k = \frac{1}{z_k^\top s_k}. \tag{3.3.8}$$

3.4 Distance Measures

The choice of distance measure is of great importance in the registration model used in FAIR. We want to transform the template image so that it aligns with the reference image, and to do this we need a measure of similarity between the images. This is where distance measures come in. In general, distance measures can be formulated as a functional

$$\mathcal{D}[T, R] = \int_\Omega \phi(\mathcal{T}, \mathcal{R})\, dx, \tag{3.4.1}$$

for some function ϕ describing the similarity at point x. For the transformed template $\mathcal{T}[y]$ we get

$$\mathcal{D}[y] := \mathcal{D}[\mathcal{T}[y], \mathcal{R}]. \tag{3.4.2}$$

The distance measures Normalized Cross-Correlation, Sum of Squared Differences, Mutual Information and Normalized Gradient Fields are included in the FAIR package, and discussed in [18]. In addition I have implemented a method based on the second order derivatives; Normalized Hessian Field.

3.4.1 Sum of Squared Differences

The Sum of Squared Differences, or SSD for short, is one of the most intuitive distance measures:

$$\mathcal{D}^{SSD}[\mathcal{T},\mathcal{R}] = \frac{1}{2}\int_\Omega (\mathcal{T}(x)-\mathcal{R}(x))^2\,dx. \tag{3.4.3}$$

This is the L_2-norm of the difference image, and simply compares the intensity values of the two images. It is squares as to not give negative values. One of the shortcomings of this method is that it assumes the intensity values at x in the reference $\mathcal{R}(x)$ and in the transformed template $\mathcal{T}[y](x)$ to be identical, which is seldom the case in multimodal image registration.

Of course, the integral in Equation 3.4.3 can not be computed analytically. Instead, we will have to discretize. We define the function

$$\psi(x) = \frac{1}{2}(\mathcal{T}(x)-\mathcal{R}(x))^2. \tag{3.4.4}$$

Our discretization is on a cell-centered grid $\mathtt{xc}$ of width h. Denote $T^h = T(\mathtt{xc})$ and $R^h = R(\mathtt{xc})$. The discretized SSD is then defined as

$$D^{SSD,h}(T^h,R^h) = \frac{1}{2}\cdot hd\cdot\|T^h-R^h\|^2, \tag{3.4.5}$$

where $\mathtt{hd} = h_1\cdots h_d$. The derivate of the discretized functional with respect to $\mathtt{yc}$ can be computed to be

$$dJ(\mathtt{yc}) = \mathtt{hd}\cdot dT(\mathtt{yc})\cdot(T(\mathtt{yc})-R(\mathtt{xc})). \tag{3.4.6}$$

Example 3.4.1 (Registration using Sum of Squared Differences)**.** Example of registration using Sum of Squared Differences. With the regularization parameter set too high (Figure 3.14a), the registration is unable to find anything resembling the solution. Lowering the regularization (Figure 3.14b) makes the registration able to find a direction, but as is evident both from the grid and the resulting image this is not a viable solution.

3.4.2 Normalized Cross-Correlation

The Normalized Cross-Correlation is an improvement over the Sum of Squared Differences in that instead of assuming the gray-values of $T(\mathtt{yc})$ and $R(\mathtt{xc})$ to be identical, it assumes a linear dependence of $T[y]$ and R:

$$\lambda\mathcal{T}[y] = \mu\mathcal{R},\quad \lambda,\mu\in\mathbb{R}. \tag{3.4.7}$$

The cross-correlation of two images T and R is given by

$$\langle\mathcal{T},\mathcal{R}\rangle = \int_\Omega \mathcal{T}(x)\mathcal{R}(x)\,dx. \tag{3.4.8}$$

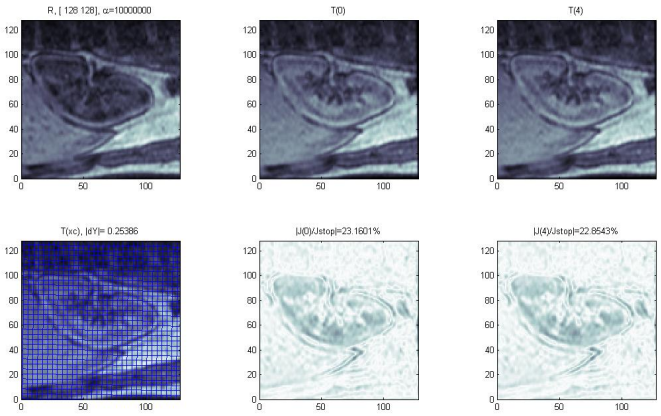
Ideally this should give a large response when the alignment is correct, but local high intensities in the images can give matches at the incorrect position [22]. To remedy this the cross-correlation can be normalized. The version of normalized cross-correlation used in FAIR also squares the terms and includes a minus sign to change the problem into a minimization problem with values in the range $[0,1]$:

$$D^{NCC}[\mathcal{T},\mathcal{R}] = 1-\mathrm{NCC}[\mathcal{T},\mathcal{R}]^2 = 1-\frac{\langle\mathcal{T},\mathcal{R}\rangle^2}{\|\mathcal{T}\|^2\,\|\mathcal{R}\|^2}, \tag{3.4.9}$$

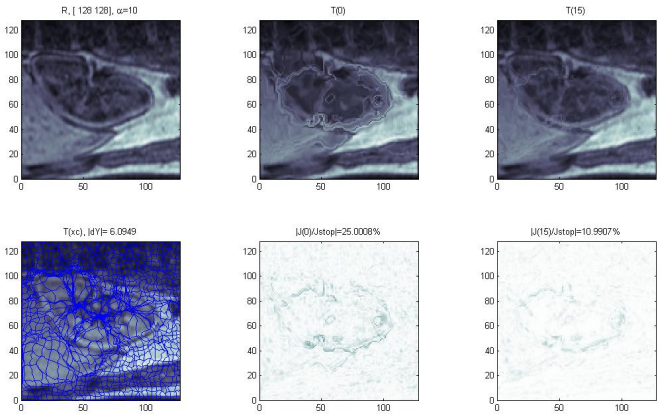
where $\|\mathcal{T}\| = \sqrt{\langle\mathcal{T},\mathcal{T}\rangle}$.

As in the SSD, we need a discretized version. Let $\mathtt{Rc} = R(\mathtt{xc})$ and $\mathtt{Tc} = T(\mathtt{yc})$. We then get

$$\mathrm{NCC}^h(\mathtt{Tc},\mathtt{Rc}) = \frac{\langle\mathtt{Tc},\mathtt{Rc}\rangle}{\|\mathtt{Tc}\|\|\mathtt{Rc}\|}, \tag{3.4.10}$$



(a) *Regularization constant* $\alpha = 1e7$



(b) *Regularization constant* $\alpha = 1e0$

Figure 3.14: *Example of images registered using Sum of Squared Differences.*

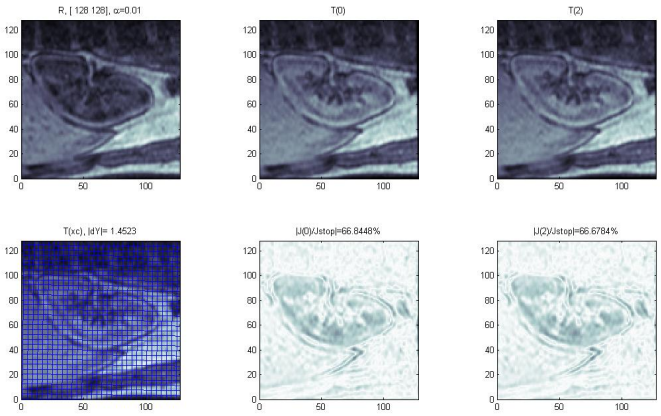
with $\|\mathtt{Tc}\| = \sqrt{\mathtt{Tc}^\top \mathtt{Tc}}$. The distance measure in Equation 3.4.9 can then be rephrased as follows:

$$D^{\mathrm{NCC}}(\mathtt{Tc}) = \psi(\mathtt{Tc}) = 1 - \frac{(\mathtt{Rc}^\top \mathtt{Tc})}{(\mathtt{Rc}^\top \mathtt{Rc})\,(\mathtt{Tc}^\top \mathtt{Tc})}. \tag{3.4.11}$$

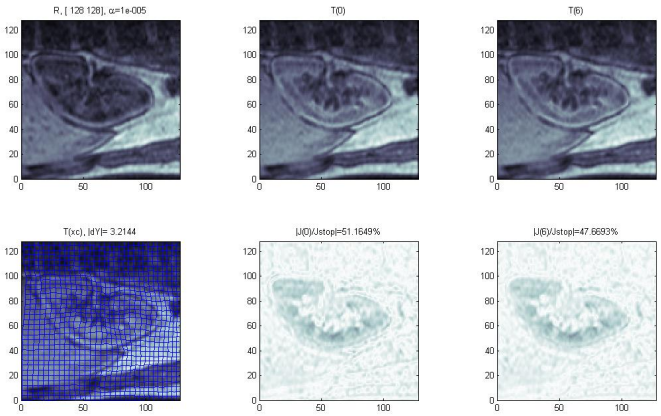
Example 3.4.2 (Example of registration using Normalized Cross Correlation)**.** Similar to the example with Sum of Squared Differences, it is difficult to get decent results. Figure 3.15a shows an example with the regularization set too high. Figure 3.15b shows some distortion due to insufficient regularization. The results do however seem better than the ones obtained using SSD.

3.4.3 Mutual information

Mutual information is one of the most used distance measures in medical image registration. It originated in information theory, and the concept is to study the mutual information of two sequences T and R, or the normalized entropy of their joint density. Unlike Sum of Squared Differences and Normalized Cross-Correlation, Mutual Information is very general since no assumptions on the nature of the dependence are made [17]. This distance measure was proposed independently by Collignon [5] and Viola [28]. For more on Mutual Information see also [21].



(a) *Regularization constant* $\alpha = 1e-2$



(b) *Regularization constant* $\alpha = 1e-5$

Figure 3.15: *Example of images registered using Normalized Cross Correlation.*

The first step in the process is to produce a histogram, summarizing the coincidences of all possible n pairs of the sequence

$$\rho^{\text{hist}}(t,r) = \#\{(T_i,R_j)|T_i = t \wedge R_j = r\}/n. \tag{3.4.12}$$

The entropy H is then a measure for quantifying the "sharpness"of the histogram ρ^{hist}:

$$H[\rho^{\text{hist}}] = -\sum_{t,r} \rho^{\text{hist}}(t,r) \log \rho^{\text{hist}}(t,r) = -\frac{1}{n}\sum_{i,j} \log \rho^{\text{hist}}(T_i,R_j). \tag{3.4.13}$$

The computation of the mutual information involves a normalization step, and is defined by

$$\hat{\text{MI}}[\rho^{\text{hist}}] = H[\rho_T^{\text{hist}}] + H[\rho_R^{\text{hist}}] - H[\rho^{\text{hist}}], \tag{3.4.14}$$

where $\rho_T^{\text{hist}} = \sum_r \rho^{\text{hist}}(t,r)$ and $\rho_R^{\text{hist}} = \sum_r \rho^{\text{hist}}(t,r)$ are the marginal densities.

The process of taking MI to the discretized framework is not straightforward. The quantity ρ needs to be estimated. The popular approaches to this involve discretizing T. One method is based on histograms, another on Parzen-window estimators. The histogram method is considered inferior [26], so we will focus on Parzen-window-based estimators.

The basic idea here is to work with a smooth kernel function which spreads out the sampled data. For registration purposes we will use a smooth, compactly supported kernel; a cubic spline. This kernel is bell shaped with integral one. The width of the kernel k is controlled by a dilatation factor σ,

$$k(t,\sigma) = \frac{k(t/\sigma)}{\sigma}. \tag{3.4.15}$$

Since the integral is being kept at one, dilatation also affects the height of the function.

The estimated density is then obtained by summing shifted copies of the kernel and dividing by the number of copies

$$\rho(t;\sigma) = \rho(t;\mathcal{T},\mathtt{xc},\sigma) = \frac{1}{m}\sum_{j=1}^{m} k(t-t_j,\sigma). \tag{3.4.16}$$

We set $\sigma = (t_n - t_0)/n$. For two images T and R with m-point discretization $\mathtt{xc}$ we get

$$\rho_{[\mathcal{T},\mathcal{R}]}(t,r) = \rho(t,r;\mathcal{T},\mathcal{R},\mathtt{xc},\sigma) = \frac{1}{m}\sum_{j=1}^{m} k(t-\mathcal{T}(\mathtt{xc}_j),\sigma)k(r-\mathcal{R}(\mathtt{xc}_j),\sigma). \tag{3.4.17}$$

The marginal densities $\rho_{[\mathcal{T}]}$ and $\rho_{[\mathcal{R}]}$ are

$$\rho_{[\mathcal{T}]} = \int_{\mathbb{R}} \rho_{[\mathcal{T},\mathcal{R}]}(t,r)\,dr \quad \text{and} \quad \rho_{[\mathcal{R}]} = \int_{\mathbb{R}} \rho_{[\mathcal{T},\mathcal{R}]}(t,r)\,dt, \tag{3.4.18}$$

and, finally, the discretized MI distance measure is given by

$$\mathrm{MI}[\mathcal{T},\mathcal{R}] = \int_{\mathbb{R}} \rho_{[\mathcal{T}]}\log\rho_{[\mathcal{T}]}\,dt + \int_{\mathbb{R}} \rho_{[\mathcal{R}]}\log\rho_{[\mathcal{R}]}\,dr - \int_{\mathbb{R}^2} \rho_{[\mathcal{T},\mathcal{R}]}\log\rho_{[\mathcal{T},\mathcal{R}]}\,d(t,r). \tag{3.4.19}$$

In our discretization, our main task is to compute the integral

$$I = \int_{\mathbb{R}^2} \rho_{[\mathcal{T},\mathcal{R}]}\log\rho_{[\mathcal{T},\mathcal{R}]}\,d(t,r). \tag{3.4.20}$$

We assume that the values of the functions $\mathcal{T}$ and $\mathcal{R}$ are within a known range $[t_0,t_n]$ and $[r_0,r_n]$. Typically, $t_0 = r_0 = 0$ and $t_n = r_n = 255$. The integrals will therefore be approximated by a midpoint quadrature rule using an n_t-by-n_r grid. Let $\eta_t = (t_n - t_0)/n_t$ and $\eta_r = (r_n - r_0)/n_r$. Furthermore, let

$$t_i = t_0 + (i-0.5)\eta_t, \qquad r_j = r_0 + (j-0.5)\eta_r, \qquad i = 1,\dots,n_t, \qquad j = 1,\dots,n_r. \tag{3.4.21}$$

To prevent consideration for $0\log 0$, we add a small tolerance,

$$I = \eta_t\eta_r\rho^{\mathsf{T}}\log(\rho+\epsilon) = \eta_t\eta_r\sum_{i=1}^{n_t}\sum_{j=1}^{n_r}\rho_{i,j}\log(\rho_{i,j}+\epsilon). \tag{3.4.22}$$

The discretized marginal densities becomes

$$\rho_{\mathcal{T}}(t_k) = \eta_r\sum_{i=1}^{n_r}\rho_{k,i} \quad \text{and} \quad \rho_{\mathcal{R}}(r_k) = \eta_t\sum_{i=1}^{n_t}\rho_{i,k}. \tag{3.4.23}$$

When using Mutual Information as a distance measure in FAIR there is an optional parameter for choosing the size of the spline grid.

Example 3.4.3. Example of registration using Mutual Information, results shown in Figure 3.16. A 32-by-32 spline grid was used. To register with MI we need to use a different optimization process than the Gauss-Newton type discussed in Section 3.3. The alternative implemented in FAIR is a BFGS method (l-BFGS), see [18] or [19] for more. This method reduces the memory consumption and time required to perform registration, especially using Mutual Information. To give an example of the difference, using the Gauss-Newton scheme on the images in this example took more than 15 *minutes*, whereas the lBFGS approach used 6.35s.

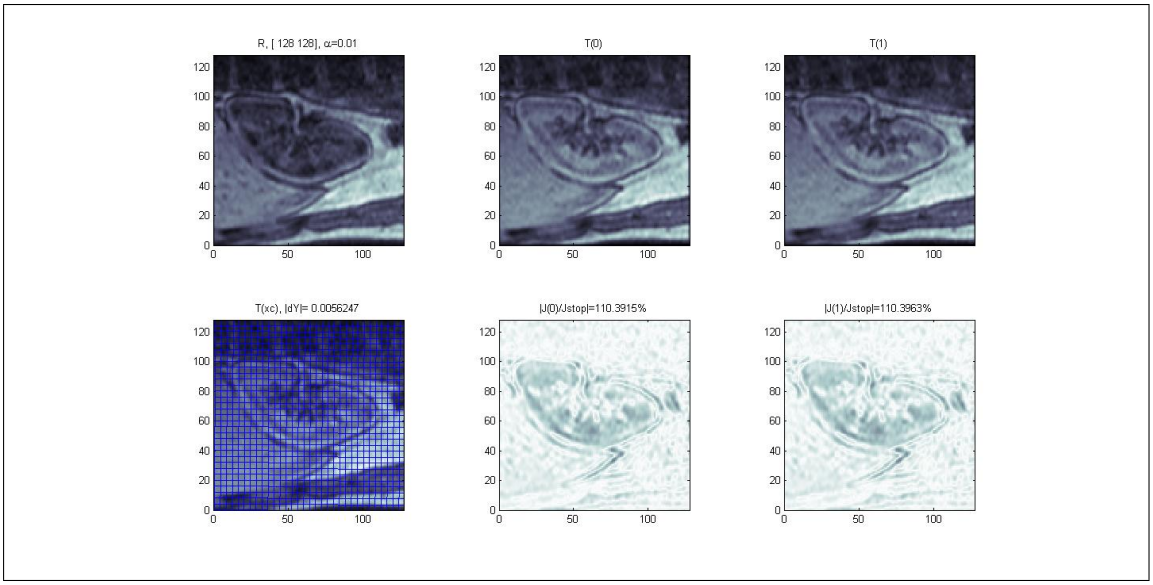


Figure 3.16: *Example of images registered using Mutual Information.*

3.4.4 Normalized Gradient Fields

The Normalized Gradient Field is a distance measure developed by Jan Modersitzki along with Eldad Haber [12]. It is based on the idea that information in the image $\mathcal{T}$ can be displayed by the intensity changes, and thus by the image gradient $\nabla\mathcal{T}$.

Looking at the steepest descent of a generic distance measure $\mathcal{D}[y] = \mathcal{D}[\mathcal{T}[y], \mathcal{R}]$, the chain rules gives us

$$d_y\mathcal{D} = d_{\mathcal{T}}\mathcal{D}\, d_y\mathcal{T}, \tag{3.4.24}$$

implying that $\nabla\mathcal{T} = (d_y\mathcal{T})^\top$ is an important ingredient.

Instead of assuming that intensity values in the transformed $\mathcal{T}[y]$ and $\mathcal{R}$ correspond, we use the premise that intensity changes appear at corresponding positions. Intensity changes are indicated by the gradient, with larger values of the gradient corresponding to steeper intensity differences. Since the magnitude of change and thus the magnitude of the gradient representing an edge may not be identical from one image to the other, we replace $\nabla\mathcal{T}$ with the normalized $\nabla\mathcal{T}/|\nabla\mathcal{T}|$, thus removing information about the strength and only focusing on the positions of the change.

We start off the discussion of Normalized Gradient Fields (NGF) with the continuous version. We define the NGF by

$$\mathrm{n}[\mathcal{T}] = \mathrm{n}[\mathcal{T}, \eta] = \frac{\nabla\mathcal{T}}{\sqrt{|\nabla\mathcal{T}| + \eta^2}},$$

where η is an edge parameter defining what is to be looked at as an edge ($|\nabla T| > \eta$) and what is to be considered within the noise levels ($|\nabla\mathcal{T}| \leq \eta$). Without this parameter, any noise in the picture will be exaggerated by the normalization.

Next, we need to make sure that there is no bias in the direction of the NGF. Let $t := \mathrm{n}[\mathcal{T}]$ and $r := \mathrm{n}[\mathcal{R}]$. Since a change from low to high intensity in one image might correspond to a change from high to low values in the other, we want $t = \pm r$ to be treated identically. We also need to make sure that we don't get a match if t or r is zero. To get this, instead of minimizing the area spanned by the two vectors, we maximize the linear dependency of the two. This leads us to

$$d(t, r) = 1 - (t^\top r)^2.$$

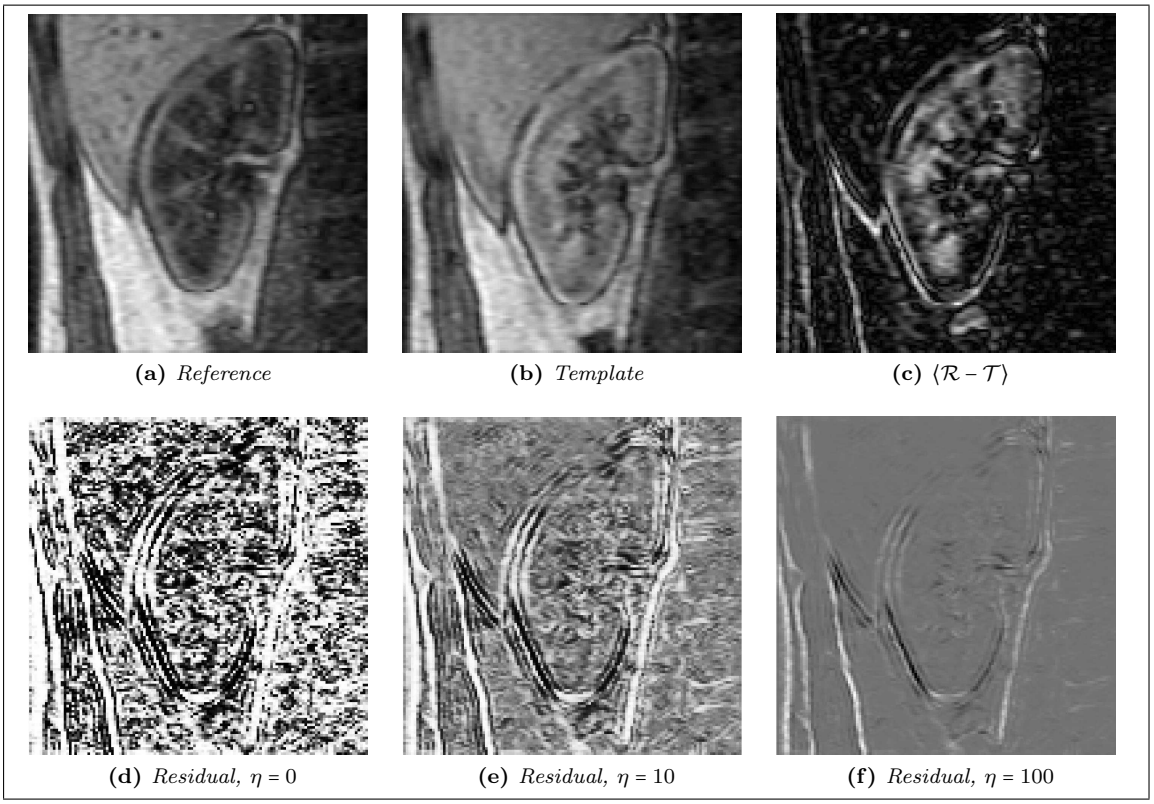
This gives us minimal value zero for $t = \pm r$ and maximum value one if t is orthogonal to r or if t or r is zero.

Our distance measure is as follows:

$$D^{\mathrm{NGF}}[\mathcal{T},\mathcal{R}] = \mathrm{NGF}[\mathcal{T},\mathcal{R}] = \int_\Omega 1 - (\mathrm{n}[\mathcal{T}(x)]^\top \mathrm{n}[\mathcal{R}(x)])^2\, dx,$$

where $\mathrm{n}[\mathcal{T}(x)]^\top \mathrm{n}[\mathcal{R}(x)]$ is referred to as the residual.

Example 3.4.4 (The importance of the edge parameter in Normalized Gradient Field)**.** To show the importance of the edge parameter, consider the following example. Two relatively similar images taken of a kidney at two different times in an acquisition process with injected contrast agent are shown in Figure 3.17. As seen in the figure the residual becomes very noisy in the absence of the parameter η. However, choosing a value that is too high will result in a loss of detail that might interfere with the registration.



(a) *Reference* **(b)** *Template* **(c)** $\langle \mathcal{R} - \mathcal{T} \rangle$

(d) *Residual,* $\eta = 0$ **(e)** *Residual,* $\eta = 10$ **(f)** *Residual,* $\eta = 100$

Figure 3.17: *Two images of a kidney with contrast agent taken some time apart. In the top row, 3.17a and 3.17b shows the reference and template images and 3.17c shows the absolute difference of the two. The bottom row shows three images of the NGF residual with three different values of* η*.*

Discretizing NGF

Our next step is to make a discretized version of the NGF. Let `xc` denote a cell-centered grid and `yc` $= y($`xc`$)$ be the transformed grid. The gradient ∇T is approximated by finite differences,

$$\partial_j \mathcal{T}[y](\texttt{xc}) \approx \partial_j^h T(\texttt{yc}). \tag{3.4.25}$$

One way of doing this is with the Kronecker product, $\otimes$,

$$\begin{aligned}
\partial_1^h &= I_{m^3} \otimes I_{m^2} \otimes D_1, \\
\partial_2^h &= I_{m^3} \otimes D_2 \otimes I_{m^3}, \\
\partial_3^h &= D_3 \otimes I_{m^2} \otimes I_{m^3},
\end{aligned}$$

where

$$D_j = \frac{1}{2h^j}\begin{bmatrix} -1 & 1 & & & \\ -1 & 0 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 0 & 1 \\ & & & -1 & 1 \end{bmatrix} \in \mathbb{R}^{m^j, m^j}. \tag{3.4.26}$$

Once the gradients have been computed, we can carry out all operations pointwise. Let $\mathtt{Tc} = T(\mathtt{yc}) \in \mathbb{R}^n$ and $\mathtt{Rc} = R(\mathtt{xc}) \in \mathbb{R}^n$,

$$\mathtt{gradT}_i = [(\partial_1^h \mathtt{Tc})_i, (\partial_2^h \mathtt{Tc})_i, (\partial_3^h \mathtt{Tc})_i]^\top, \qquad \mathtt{gradR}_i = [(\partial_1^h \mathtt{Rc})_i, (\partial_2^h \mathtt{Rc})_i, (\partial_3^h \mathtt{Rc})_i]^\top. \tag{3.4.27}$$

With

$$\mathrm{n}[T[y](x_i)]^\top \mathrm{n}[R(x_i)] \approx \mathtt{rc}_i := \frac{\mathtt{gradT}_i^\top \mathtt{gradR}_i}{\sqrt{\|\mathtt{gradT}_i\|^2 + \eta^2}\sqrt{\|\mathtt{gradR}_i\|^2 + \eta^2}}, \tag{3.4.28}$$

$\mathtt{hd} = h^1 \cdots h^d$, and $\hat{\omega} = \prod_{i=1}^{d} (\omega^{2i-1} - \omega^{2i})$, the discretized NGF becomes

$$\mathrm{NGF}(\mathtt{yc}) = \mathtt{hd} \sum_{i=1}^{m} (1 - \mathtt{rc}_i^2) = \hat{\omega} - \mathtt{hd} \cdot \mathtt{rc}^\top \mathtt{rc}. \tag{3.4.29}$$

Example 3.4.5. Example of registration using Normalized Gradient Fields. The edge parameter was set to 10 and the regularization parameter to 0.2. The time taken to register was 21.78s.

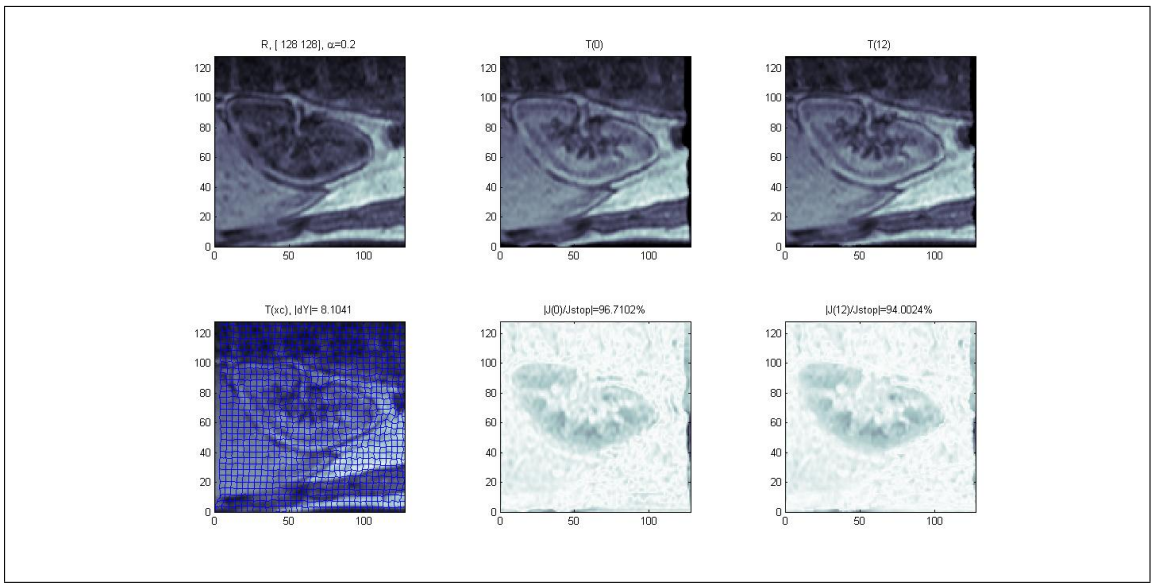


Figure 3.18: *Example of images registered using Normalized Gradient Field.*

3.4.5 Normalized Hessian Field

Unlike the previous distance measures, the Normalized Hessian Field is not a part of the FAIR package. One of the main parts of my work has been to implement a distance measure based on the second derivatives. The idea is in many ways similar to the Normalized Gradient Fields, but instead of looking at the cross correlation of gradients, we use the second order information from the Hessian matrix to compute the distance.

For each point in the image we want to find all the second order partial derivatives. Equation 3.3.4 shows a generic Hessian Matrix, which in our case yields a 2-by-2 (for a 2 dimensional image) or 3-by-3 (for 3 dimensions) symmetric matrix which we store in a vector for easier computation.

The second order derivative information is tied to the curvature or concavity of a function. A positive second derivative indicates that the function is convex or concave up; a negative second derivative means the function is concave down or simply concave. Like with Normalized Gradient Fields there is no guarantee that the sign or strength of the second derivatives will correspond when we have multimodal images, and again this can be fixed by replacing the second derivative vector with a normalized version and by computing the correlation of the two images.

We define the NHF as follows

$$\mathrm{n}[\mathcal{T}] = \frac{H[\mathcal{T}]}{\sqrt{|H[\mathcal{T}]^2 + \eta^2}}, \tag{3.4.30}$$

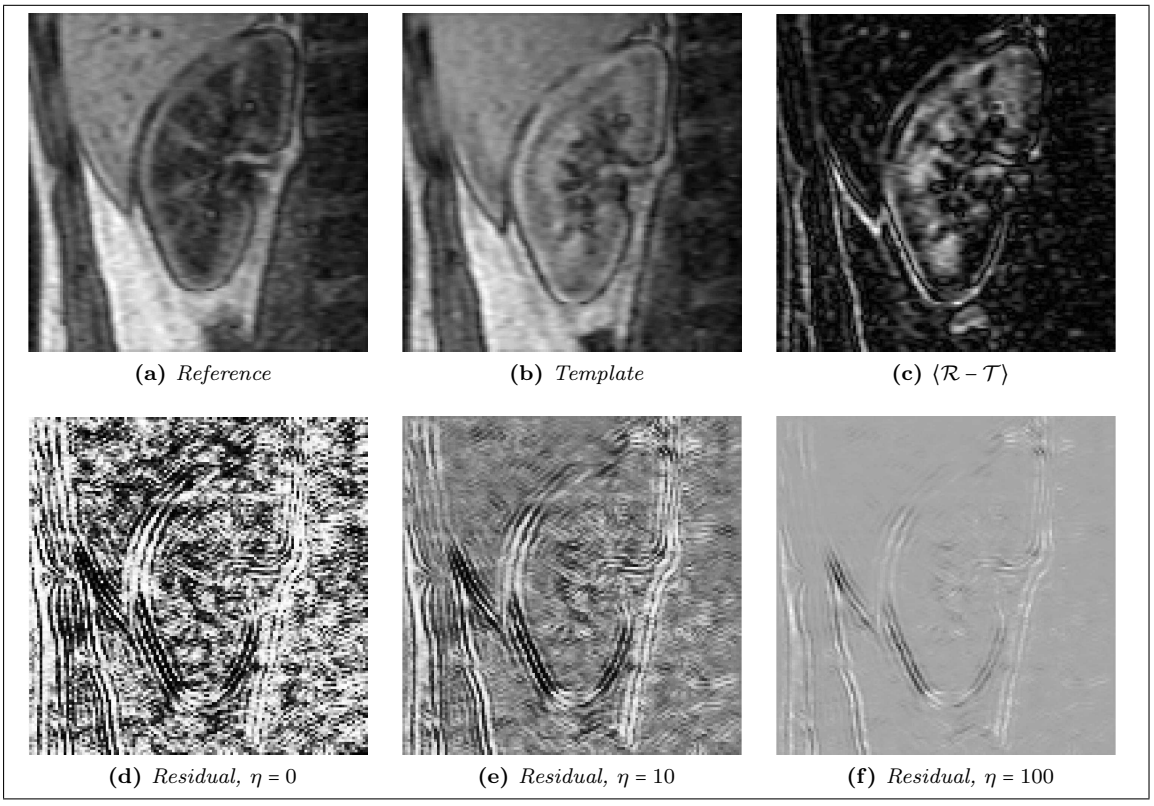
with $H[\mathcal{T}]$ being the vectorized Hessian of the image,

$$H[\mathcal{T}] = \left[\frac{\partial \mathcal{T}}{\partial x_1^2}, \frac{\partial \mathcal{T}}{\partial x_1 \partial x_2}, \ldots, \frac{\partial \mathcal{T}}{\partial x_n^2}\right]. \tag{3.4.31}$$

The distance measure is defined as

$$D^{\mathrm{NH}}[\mathcal{T}, \mathcal{R}] = \mathrm{NH}[\mathcal{T}, \mathcal{R}] = \int_\Omega 1 - (n[\mathcal{T}]^\mathsf{T} n[\mathcal{R}])^2 dx. \tag{3.4.32}$$

Example 3.4.6 (The importance of the edge parameter in Normalized Hessian Field)**.** The edge parameter is equally important in Normalized Hessian Field as in Normalized Gradient Field, as Figure 3.19 shows.



(a) *Reference* **(b)** *Template* **(c)** $\langle \mathcal{R} - \mathcal{T} \rangle$

(d) *Residual,* $\eta = 0$ **(e)** *Residual,* $\eta = 10$ **(f)** *Residual,* $\eta = 100$

Figure 3.19: *Two images of a kidney with contrast agent taken some time apart. In the top row, 3.19a and 3.19b shows the reference and template images and 3.19c shows the absolute difference of the two. The bottom row shows three images of the NGF residual with three different values of* η.

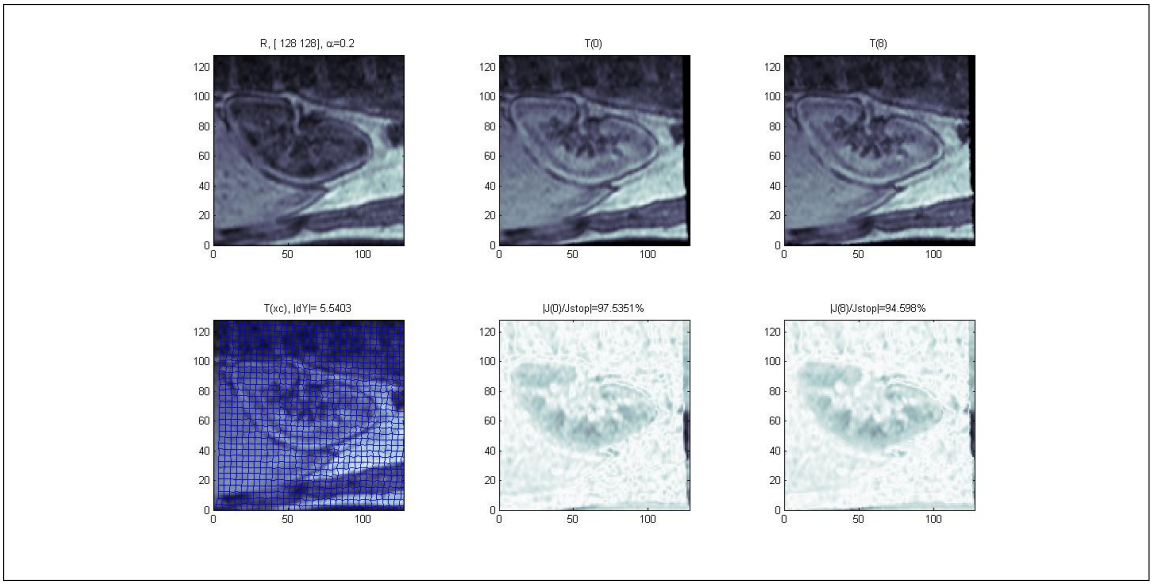


Figure 3.20: *Example of images registered using Normalized Hessian Field.*

Discretizing Normalized Hessian Field

The process of discretizing our Hessian-based distance measure is similar to that in Section 3.4.4. Let $\mathtt{Tc} = T(\mathtt{yc})$ and $\mathtt{Tc}_{i,j} = \partial_i^h \partial_j^h \mathtt{Tc}$ and similarly for $R(\mathtt{xc})$. For a two dimensional image we have

$$\mathtt{hessT}_i = \begin{bmatrix} (\mathtt{Tc}_{1,1})_i & (\mathtt{Tc}_{1,2})_i & (\mathtt{Tc}_{2,1})_i & (\mathtt{Tc}_{2,2})_i \end{bmatrix} \tag{3.4.33}$$

where the second order derivative matrices have been vectorized. This yields an $mn \times 4$ matrix for an image of dimensions m by n. The $3D$ case yields a $mn \times 9$ matrix. We approximate the Normalized Hessian Field by

$$\mathrm{n}[T[y](x_i)]^\top \mathrm{n}[R(x_i)] \approx \mathtt{rc}_i := \frac{\mathtt{hessT}_i^\top \mathtt{hessR}_i}{\sqrt{\|\mathtt{hessT}_i\|^2 + \eta^2}\sqrt{\|\mathtt{hessR}_i\|^2 + \eta^2}}. \tag{3.4.34}$$

See Section 3.4.4 for more details on the discretization.

Example 3.4.7. Example of registration using Normalized Hessian Fields as distance measure. The edge parameter was set to 10 and the regularization parameter to 0.2. The time taken to register was 18.18s.

3.4.6 Derivatives of Distance Measures

We will now discuss the derivatives of a general distance measure for the optimization. The idea is to decompose the distance measure into elementary building blocks,

$$D(\mathtt{yc}) = D(T(\mathtt{yc}), \mathtt{Rc}) = \psi(r(\mathtt{yc})). \tag{3.4.35}$$

To approximate the Hessian we use first order information and neglect the second order derivative of the residual r;

$$dD(\mathtt{yc}) = d\psi(r(\mathtt{yc}))dr(\mathtt{yc}), \qquad d^2D(\mathtt{yc}) \approx dr(\mathtt{yc})^\top d^2\psi(r(\mathtt{yc}))dr(\mathtt{yc}) \tag{3.4.36}$$

For our computation we need to find the discrete versions of the derivative. Following is an example with the Normalized Hessian Field.

Example 3.4.8 (Derivative of discrete Normalized Hessian Field)**.** We let $\mathtt{Tc}$ be the discrete transformed image and ∂_l^h be the discrete gradient operators as seen in Equation 3.4.26. Let $\partial_{ij}^h = \partial_i^h \partial_j^h$. We write

$$\mathtt{hessT} = \begin{bmatrix} \partial_{11}^h \mathtt{Tc} & \partial_{12}^h \mathtt{Tc} & \partial_{13}^h \mathtt{Tc} & \partial_{21}^h \mathtt{Tc} & \partial_{22}^h \mathtt{Tc} & \partial_{23}^h \mathtt{Tc} & \partial_{31}^h \mathtt{Tc} & \partial_{32}^h \mathtt{Tc} & \partial_{33}^h \mathtt{Tc} \end{bmatrix} \tag{3.4.37}$$

$$\mathtt{lengthHT} = \sqrt{\|\mathtt{hessT}_i\|^2 + \eta^2}. \tag{3.4.38}$$

Further we write the residual $r = r^1 r^2$ where

$$r^1 = \mathtt{hessT}_i^\top \mathtt{hessR}_i, \qquad r^2 = 1/(\mathtt{lengthHT}_i \mathtt{lengthHR}_i). \tag{3.4.39}$$

Using sparse matrices for our gradient operators, we use $r^1 \odot r^2 = \mathrm{diag}[r^1] r^2$ in our computations. We get

$$\begin{aligned} dr^1 = \mathrm{diag}\left[\mathtt{hessR}^1\right] \partial_{11}^h + \mathrm{diag}\left[\mathtt{hessR}^2\right] \partial_{12}^h + \mathrm{diag}\left[\mathtt{hessR}^3\right] \partial_{13}^h \\ + \mathrm{diag}\left[\mathtt{hessR}^4\right] \partial_{21}^h + \mathrm{diag}\left[\mathtt{hessR}^5\right] \partial_{22}^h + \mathrm{diag}\left[\mathtt{hessR}^6\right] \partial_{23}^h \\ + \mathrm{diag}\left[\mathtt{hessR}^7\right] \partial_{31}^h + \mathrm{diag}\left[\mathtt{hessR}^8\right] \partial_{32}^h + \mathrm{diag}\left[\mathtt{hessR}^9\right] \partial_{33}^h. \end{aligned} \tag{3.4.40}$$

$$\begin{aligned} dr^2 = -\mathrm{diag}\left[\frac{1}{\mathtt{lengthHR} \cdot \mathtt{lengthHT}^3}\right] \\ \cdot \left(\mathrm{diag}\left[\mathtt{hessT}^1\right] \partial_{11}^h + \mathrm{diag}\left[\mathtt{hessT}^2\right] \partial_{12}^h + \mathrm{diag}\left[\mathtt{hessT}^3\right] \partial_{13}^h \right. \\ + \mathrm{diag}\left[\mathtt{hessT}^4\right] \partial_{21}^h + \mathrm{diag}\left[\mathtt{hessT}^5\right] \partial_{22}^h + \mathrm{diag}\left[\mathtt{hessT}^6\right] \partial_{23}^h \\ \left. + \mathrm{diag}\left[\mathtt{hessT}^7\right] \partial_{31}^h + \mathrm{diag}\left[\mathtt{hessT}^8\right] \partial_{32}^h + \mathrm{diag}\left[\mathtt{hessT}^9\right] \partial_{33}^h \right). \end{aligned} \tag{3.4.41}$$

Using the chain rule, we can use the information in Equations 3.4.40 and 3.4.41 to get the derivative of r,

$$dr = \mathrm{diag}\left[r^2\right] dr^1 + \mathrm{diag}\left[r^1\right] dr^2. \tag{3.4.42}$$

Chapter 4

Applications to synthetic data

Testing the methods on synthetic data can be a good starting point for further discussion. This allows us to see the basic behavior of the registration process on a dataset where we know how the images correspond.

4.1 Single modality

To simulate a single modality registration an artificial image depicting a solid circular shape with an enclosing ring is created as a reference image. The template image is a transformed version of this reference image. Thus the images show the same information with only a spatial transformation separating them. Gaussian noise of mean 0 and variance 0.01 has been added to both images before they were scaled to the $[0, 255]$-range. Both parametric and non-parametric transformations will be shown.

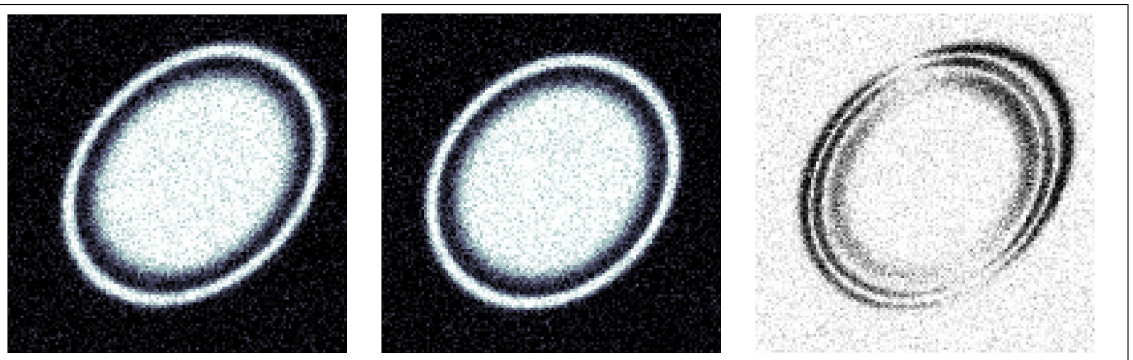
Example 4.1.1 (Comparison of distance measures on synthetic data (parametric))**.** The data is represented in Figures 4.1b and 4.1a. Since the template image in this dataset is only an affine transformed version of the reference all distance measures can be expected to do a reasonable job of registering the images. This also seems to be the case, as the difference image in each case shows mostly noise. Note that the grids found with each distance measure is different. This highlights one of problems with image registration: Even if a seemingly good solution has been found, there is often no way to be sure if this is the "true" solution. Of course, this is exaggerated in this example as the synthetic data shows more symmetry than most real data.

This example uses only parametric registration on a fixed level. The time taken and parameters are shown in Table 4.1. Sum of Squared Differences has the lowest time taken, whereas Normalized Hessian Field seems to take a long time to find a sufficient minimum.

Example 4.1.2 (Comparison of distance measures on synthetic multimodal data (non-parametric))**.** In this example the reference image is similar to the one in the previous example. The template image this time is a non-linearly transformed version of the reference image (spline-based transformation), so affine linear transforms will not give a match. The registration has

Distance measure	Optional Parameters	Time (s)
SSD	None	7.6263
NCC	None	21.3488
MI	nT = nR = 32	14.9635
NGF	$\eta = 30$	14.9633
NHF	$\eta = 10$	54.6370

Table 4.1: Parameters and time for registration on synthetic data from Example 4.1.1, "Comparison of distance measures on synthetic data (parametric)".



(a) *Template* **(b)** *Reference* **(c)** *Inversed difference* $|T(\boldsymbol{x}\boldsymbol{c}) - R(\boldsymbol{x}\boldsymbol{c})|$

Figure 4.1: *Data used in Example 4.1.1. The template is an affine transformed version of the reference image. Figure 4.1c shows the inverted difference image of the template and the reference. This highlights the differences between the images. The image has been inverted to make the differences easier to spot, as black on white stands out more than white on black. The difference image has also been adjusted so that the small intensity values stand out more.*

Distance measure	Optional Parameters	Regularization parameter, α	Time (s)
SSD	None	10^2	28.83
NCC	None	10^{-6}	9.75
MI	nT = 8 nR = 8	10^{12}	210.32
NGF	$\eta = 2$	0.5	58.48
NHF	$\eta = 0.2$	0.1	178.19

Table 4.2: Parameters and time for registration on synthetic data from Example 4.1.2, "Comparison of distance measures on synthetic multimodal data (non-parametric)".

thus been performed using non-parametric transformations. An elastic regularizer was used in all examples. Parameters and time is shown in Table 4.2 and the resulting images are shown in Figure 4.4. All registration was done over multiple levels, going from level 3 (8-by-8) to the full size on level 7 (128-by-128).

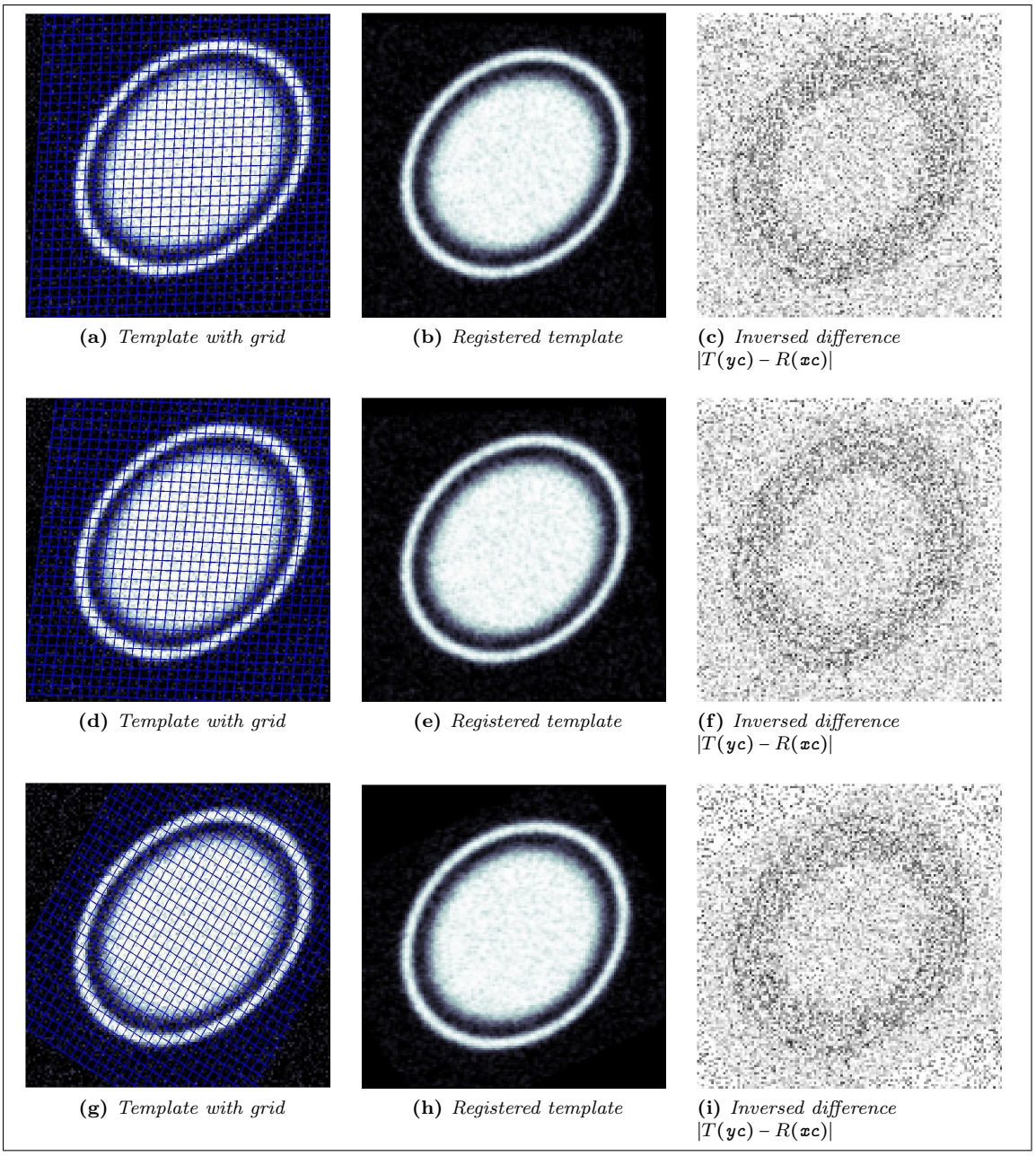
Again we see that most of the distance measures do a good job in registering the images. Perhaps surprisingly, Mutual Information is the distance measure that stands out as the one having the most problems. Of the distance measures that were able to get results, Normalized Hessian Fields again took the longest time.

4.2 Multimodal data

Multimodal data typically show different intensities and information. To simulate a multimodal dataset I have made the image shown in Figure 4.5. Example 4.2.1 shows the results of registering this data with different distance measures.

Example 4.2.1 (Comparison of distance measures on synthetic multimodal data (parametric))**.** In this example the synthetic data has been changed to a dataset containing a reference similar to the one in Example 4.1.1 and a template that is both affine transformed and inversed in intensity values. As a consequence of this we do not expect SSD to be able to register the images in any meaningful way. The results show that both the SSD and, perhaps surprisingly, NCC fail to register the images. The three other distance measures do however get good results in relatively similar times.

As in the Example 4.1.1, only parametric registration on a fixed level was performed. The time taken and optional parameters is shown in Table 4.3. Figure 4.6 shows the resulting images.



(a) *Template with grid* **(b)** *Registered template* **(c)** *Inversed difference* $|T(\boldsymbol{yc}) - R(\boldsymbol{xc})|$

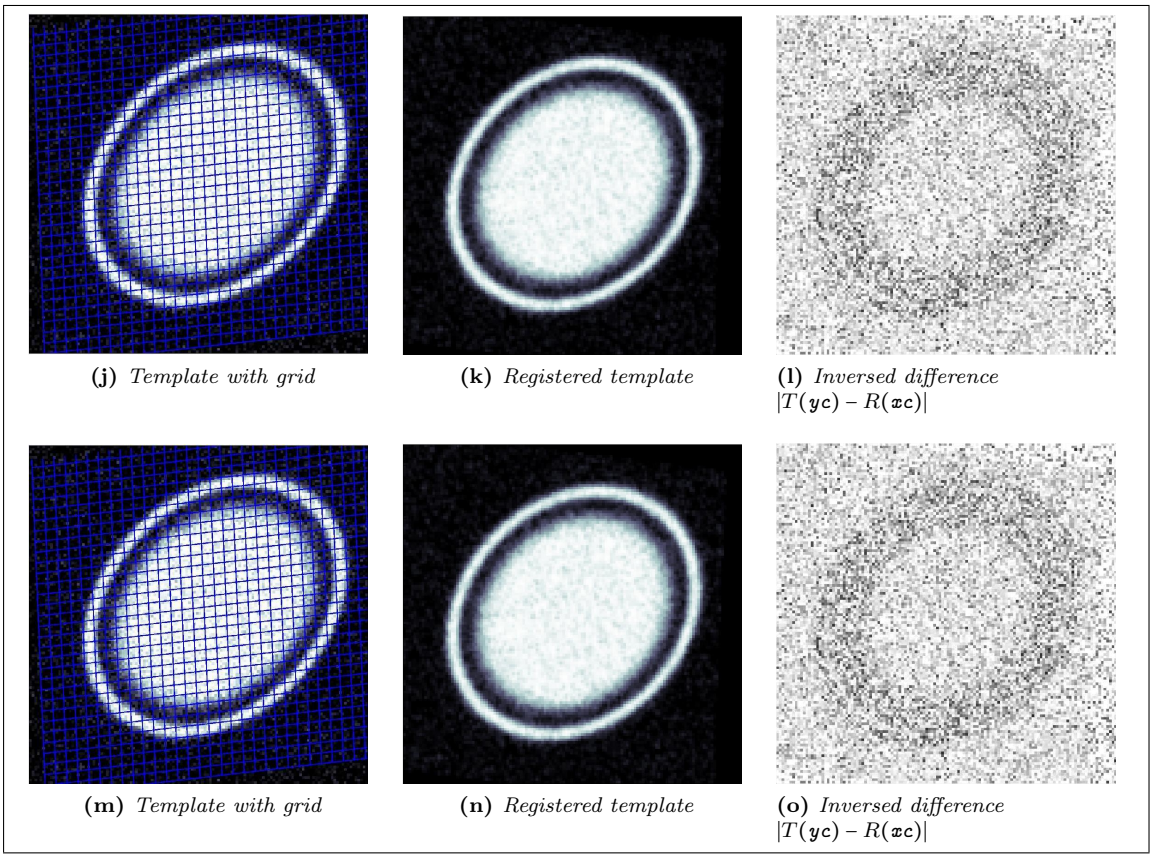
(d) *Template with grid* **(e)** *Registered template* **(f)** *Inversed difference* $|T(\boldsymbol{yc}) - R(\boldsymbol{xc})|$

(g) *Template with grid* **(h)** *Registered template* **(i)** *Inversed difference* $|T(\boldsymbol{yc}) - R(\boldsymbol{xc})|$

Figure 4.2: *Result for synthetic single modality data from Example 4.1.1. Continued on next page.*

Distance measure	Optional Parameters	Time (s)
SSD	None	29.56
NCC	None	17.01
MI	nT = nR = 8	136.53
NGF	$\eta = 30$	61.34
NHF	$\eta = 10$	69.40

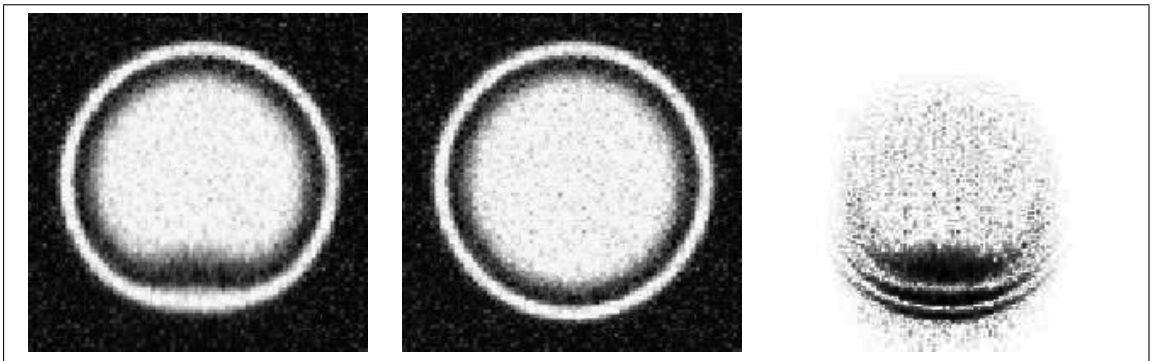
Table 4.3: Parameters and time for registration on synthetic data from Example 4.2.1, "Comparison of distance measures on synthetic multimodal data (parametric)".



(j) *Template with grid* **(k)** *Registered template* **(l)** *Inversed difference* $|T(\boldsymbol{yc}) - R(\boldsymbol{xc})|$

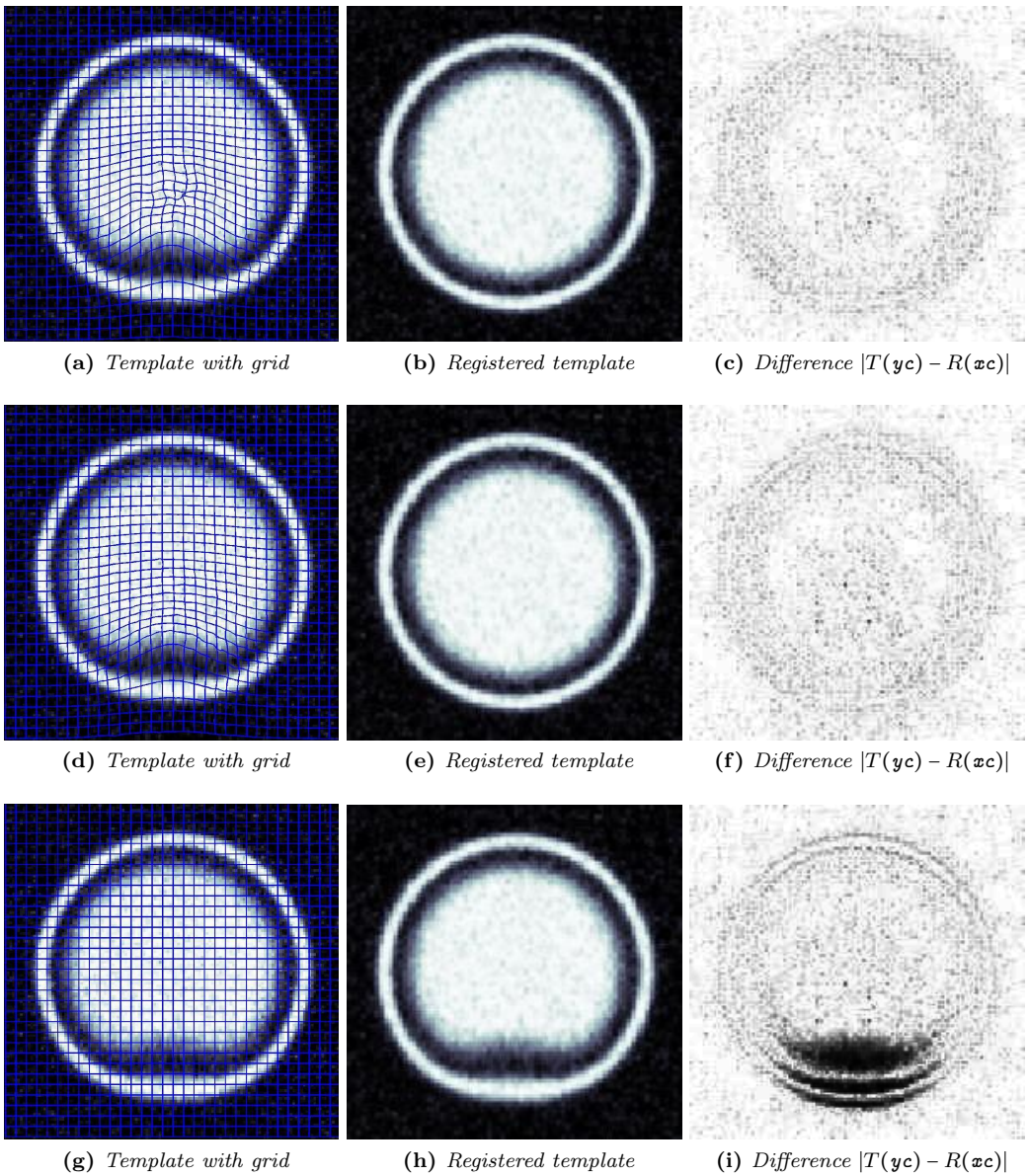
(m) *Template with grid* **(n)** *Registered template* **(o)** *Inversed difference* $|T(\boldsymbol{yc}) - R(\boldsymbol{xc})|$

Figure 4.2: *Results for synthetic data from Example 4.1.1, cont. Template and reference images shown in 4.1a and 4.1b. The rows show the results from using Sum of Squared Differences (4.2a-4.2c), Normalized Cross-Correlation (4.2d-4.2f), Mutual Information (4.2g-4.2i), Normalized Gradient Fields (4.2j-4.2l) and Normalized Hessian Field (4.2m-4.2o), respectively. Each row consists of the template overlaid with the registered grid, the registered template and the inverse of the absolute difference between registered template and reference.*



(a) *Template* **(b)** *Reference* **(c)** *Inversed difference* $|T(\boldsymbol{xc}) - R(\boldsymbol{xc})|$

Figure 4.3: *Data used in Example 4.1.2. The template is transformed using a spline-based transformation. Figure 4.3c shows the inverted difference image of the template and the reference. Like in the previous example it is inverted and adjusted to better show the differences.*

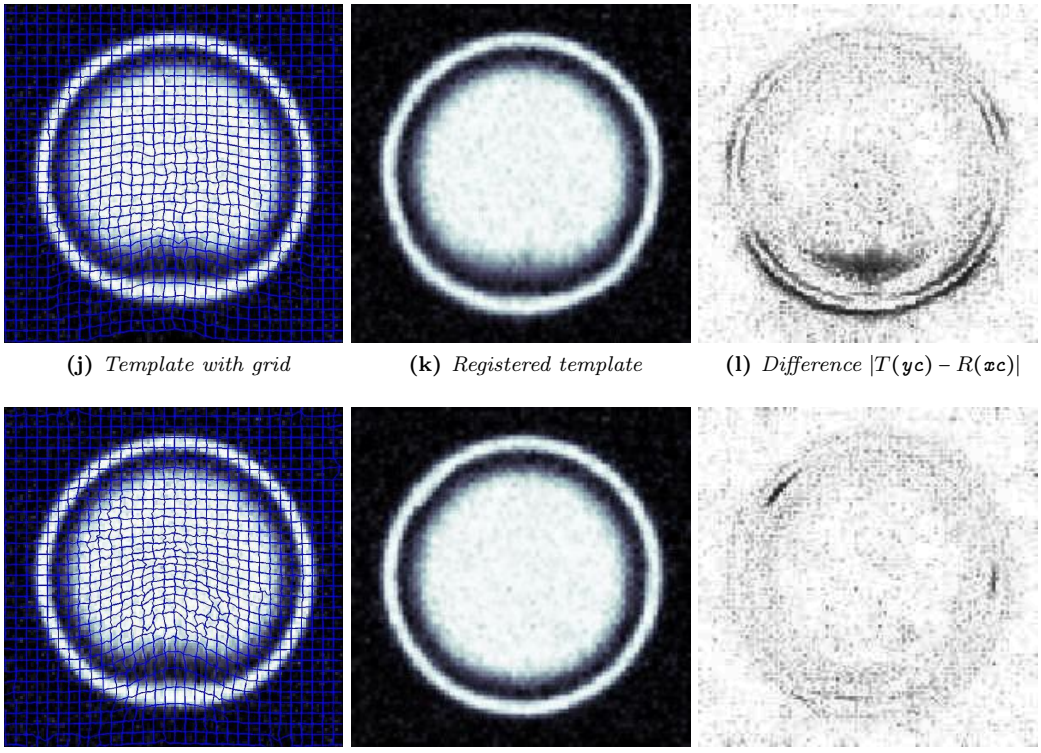


(a) *Template with grid* **(b)** *Registered template* **(c)** *Difference* $|T(\mathbf{yc}) - R(\mathbf{xc})|$

(d) *Template with grid* **(e)** *Registered template* **(f)** *Difference* $|T(\mathbf{yc}) - R(\mathbf{xc})|$

(g) *Template with grid* **(h)** *Registered template* **(i)** *Difference* $|T(\mathbf{yc}) - R(\mathbf{xc})|$

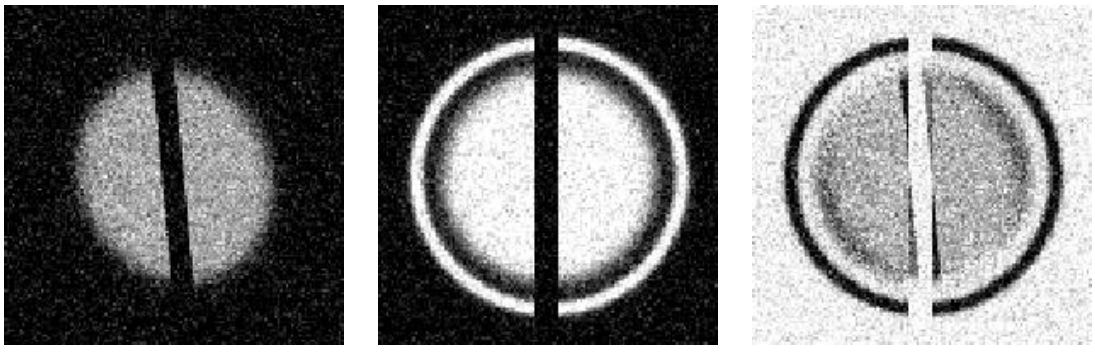
Figure 4.4: *Results for synthetic multimodal data in Example 4.1.2. Continued on next page.*



(j) *Template with grid* **(k)** *Registered template* **(l)** *Difference* $|T(\mathtt{yc}) - R(\mathtt{xc})|$

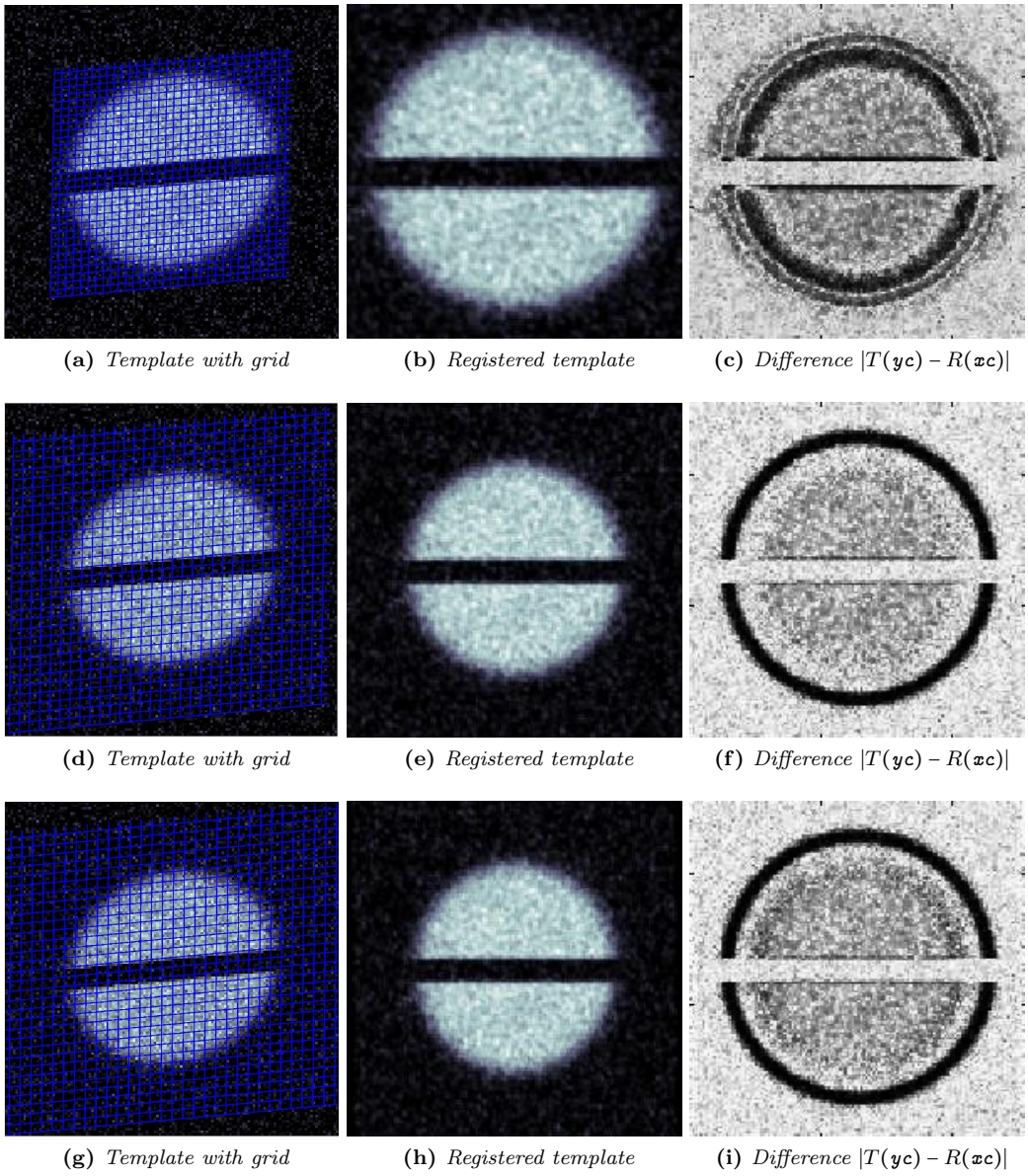
(m) *Template with grid* **(n)** *Registered template* **(o)** *Difference* $|T(\mathtt{yc}) - R(\mathtt{xc})|$

Figure 4.4: *Results for synthetic data from Example 4.1.2, continued. Template and reference images shown in 4.3a and 4.3b. The rows show the results from using Sum of Squared Differences (4.4a-4.4c), Normalized Cross-Correlation (4.4d-4.4f), Mutual Information (4.4g-4.4i), Normalized Gradient Fields (4.4j-4.4l) and Normalized Hessian Field (4.4m-4.4o), respectively. Each row consists of the template overlaid with the registered grid, the registered template and the inverse of the absolute difference between registered template and reference.*



(a) *Template* **(b)** *Reference* **(c)** *Difference* $|T(\mathtt{xc}) - R(\mathtt{xc})|$

Figure 4.5: *Data used in example 4.2.1. The difference image in Figure 4.5c has been inverted like the ones in Examples 4.1.1 and 4.1.2.*

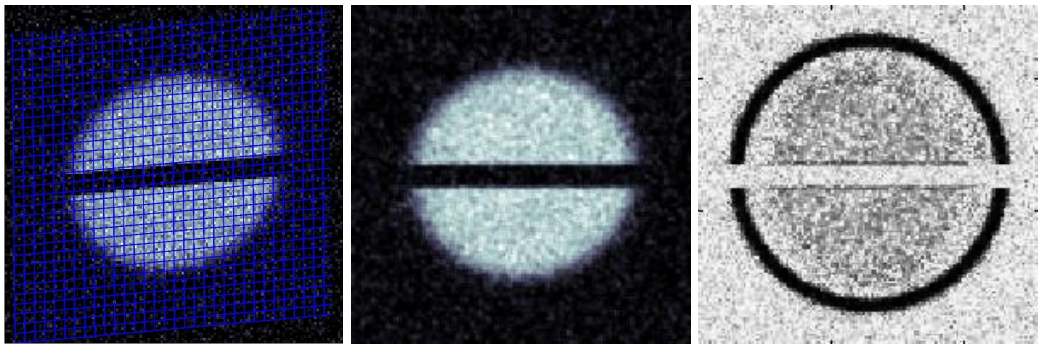


(a) *Template with grid* **(b)** *Registered template* **(c)** *Difference* $|T(\mathbf{yc}) - R(\mathbf{xc})|$

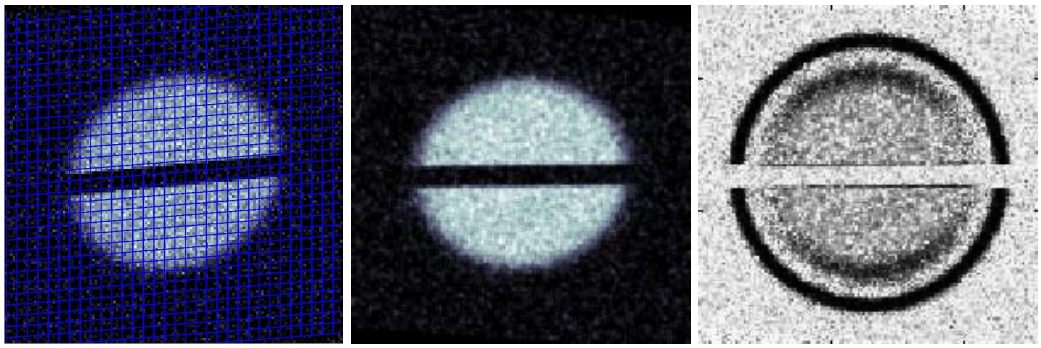
(d) *Template with grid* **(e)** *Registered template* **(f)** *Difference* $|T(\mathbf{yc}) - R(\mathbf{xc})|$

(g) *Template with grid* **(h)** *Registered template* **(i)** *Difference* $|T(\mathbf{yc}) - R(\mathbf{xc})|$

Figure 4.6: *Results for synthetic data in Example 4.2.1 using non-parametric registration.* *Continued on next page.*



(j) *Template with grid* **(k)** *Registered template* **(l)** *Difference* $|T(\mathbf{yc}) - R(\mathbf{xc})|$



(m) *Template with grid* **(n)** *Registered template* **(o)** *Difference* $|T(\mathbf{yc}) - R(\mathbf{xc})|$

Figure 4.6: *Results for synthetic data from Example 4.2.1 using non-parametric registration, cont. Template and reference images shown in 4.5a and 4.5b. The rows show the results from using Sum of Squared Differences (4.6a-4.6c), Normalized Cross-Correlation (4.6d-4.6f), Mutual Information (4.6g-4.6i), Normalized Gradient Fields (4.6j-4.6l) and Normalized Hessian Field (4.6m-4.6o), respectively. Each row consists of the template overlaid with the registered grid, the registered template and the absolute difference between registered template and reference.*

As Example 4.2.1 shows, Sum of Squared Differences do not work well even with simple examples of multimodal images. It will therefore not be used in the discussion in the following chapter.

Chapter 5

Applications to medical images

In this chapter the results on registering medical images are shown. Sum of Squared Differences and Normalized Cross-Correlation are briefly discussed, but most of the focus of this chapter will be on the application of the distance measures Mutual Information, Normalized Gradient Field and Normalized Hessian Field to multimodal medical images.

One problematic aspect of image registration, particularly when dealing with multimodal images, is the lack of good *evaluation* or *validation* methods. After registering the images it may sometimes be difficult to determine whether the results are adequate, and when dealing with multiple results of the same data it can be difficult to determine which is the better result.

I have decided to only include single slices of the registration results in this chapter and move the figures showing the entire datasets to the appendix. The image montages showing each slice of the dataset do include more information than the single slices shown in these examples, but each slice is smaller and thus it is more difficult to make out the details. In the discussion of the results in the examples I will reference these image montages.

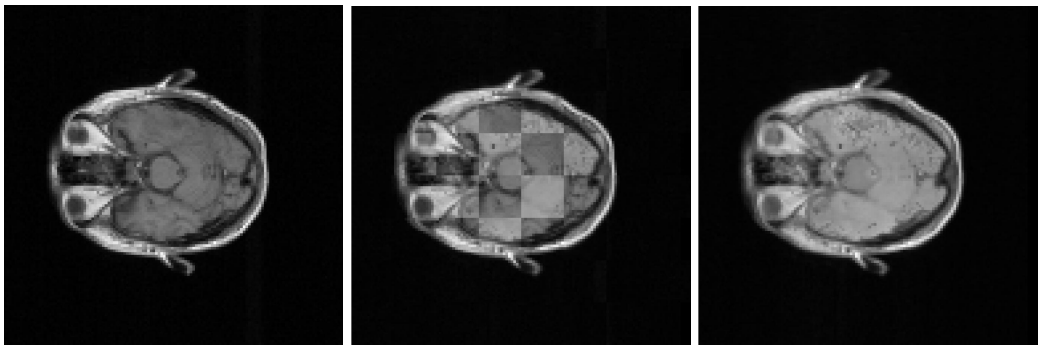
In this chapter I will use *checkerboard* images in the discussion. These are a composition of two images, the reference and registered template in our case, into a image of smaller squares or rectangles where adjacent parts are taken from different images. This lets us get some idea about the overlap of the figures, in particular inconsistencies along the edges of the rectangles in the checkerboard. An example of such a checkerboard image is shown in Figure 5.1.

5.1 Registration of brain images

I will show results from brain images from one patient in four different modalities (T1, T2 and PD-weighted MRIs, and PET) as well as MR kidney data. For the brain data I produced multilevel datasets going from 32-by-32-by-4 to 128-by-128-by-16, referred to as levels 1 (coarsest), 2 and 3 (finest) in the discussion. The finest level of the data is shown in Figure 5.2. All registration has been done using the FAIR package with a l-BFGS optimization scheme. An affine preregistration is performed at the coarsest level before the non-parametric registration governed by an elastic regularization takes place. The figures in the examples will show one slice of the data in three forms: The registered template, an inverted difference image of the registered template and the reference, and a checkerboard image of the two. Images showing montages of the entire registered data are found in Section 7.1.1. Table 5.1 shows the parameters chosen for the registrations and the time taken.

5.1.1 SSD and NCC with multimodal brain images

As discussed in Sections 3.4.1 and 3.4.2, both Sum of Squared Differences and Normalized Cross-Correlation are expected to do poorly on multimodal images. In Example 4.2.1 we saw this in practice. For reference, the time taken for the registrations were 178.76s for SSD and 137.50s for

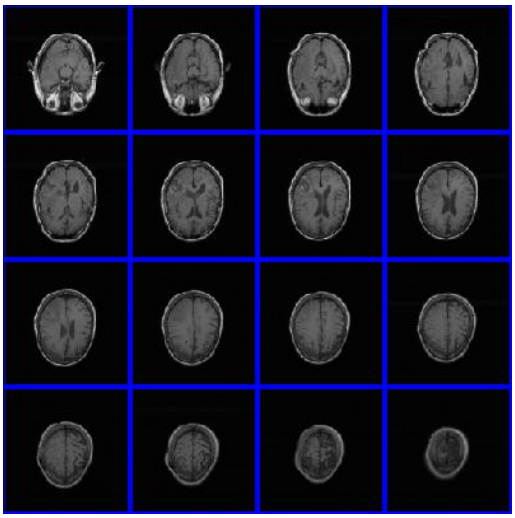


(a) *Reference* **(b)** *Checkerboard* **(c)** *Template*

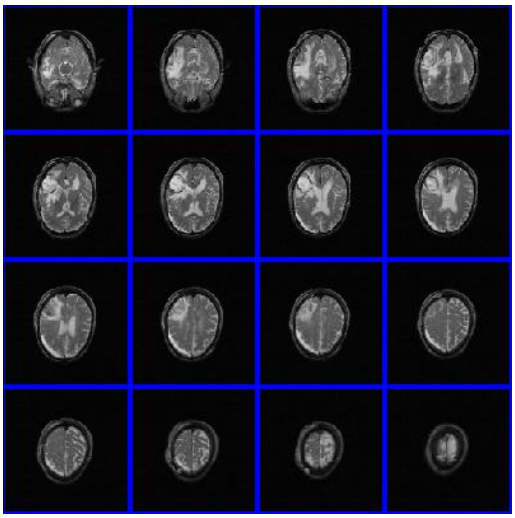
Figure 5.1: *Example of checkerboard image. 5.1a shows a T1-weighted MRI, in this example used as the reference image. 5.1c shows the registered template, here a PD-weighted MRI. The middle image, 5.1b, shows the checkerboard image of the two. The checkerboard image shows minor inconsistencies.*

Distance measure	Modality	Regularizer	Optional Parameters	Time (s)
SSD	T2	Elastic, $\alpha = 10000$	None	178.76
NCC	T2	Elastic, $\alpha = 1e-9$	None	137.50
MI	T2	Curvature, $\alpha = 1e-6$	nT = nR = 64	154.78
	PD	Curvature, $\alpha = 1e-7$	nT = nR = 64	89.22
	PET	Curvature, $\alpha = 1e-7$	nT = nR = 64	86.40
NGF	T2	Curvature, $\alpha = 100$	$\eta = 0.2$	244.30
	PD	Curvature, $\alpha = 100$	$\eta = 0.2$	200.26
	PET	Curvature, $\alpha = 100$	$\eta = 3$	130.60
NHF	T2	Curvature, $\alpha = 100$	$\eta = 0.2$	488.45
	PD	Curvature, $\alpha = 100$	$\eta = 0.2$	935.80
	PET	Curvature, $\alpha = 10$	$\eta = 0.1$	660.56
NGHF	T2	Curvature, $\alpha = 100$	$\eta = 0.2$	1222.1
	PD	Curvature, $\alpha = 100$	$\eta = 0.2$	1839.2
	PET	Curvature, $\alpha = 10$	$\eta = 0.1$	857.74

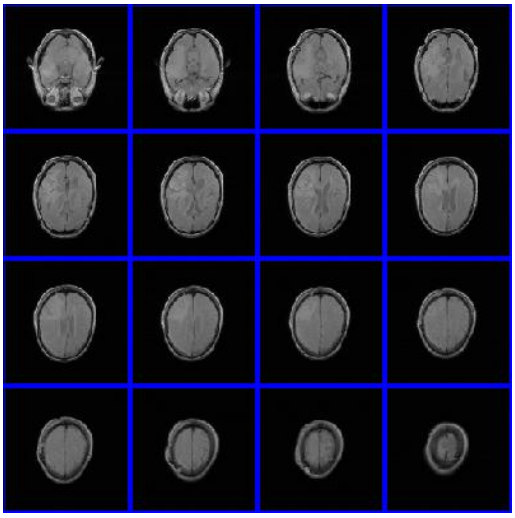
Table 5.1: Parameters and time for registration of brain data.



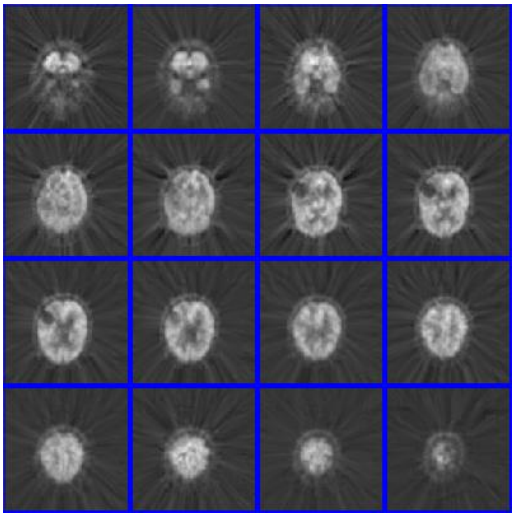
(a) *T1 Reference*



(b) *T2*

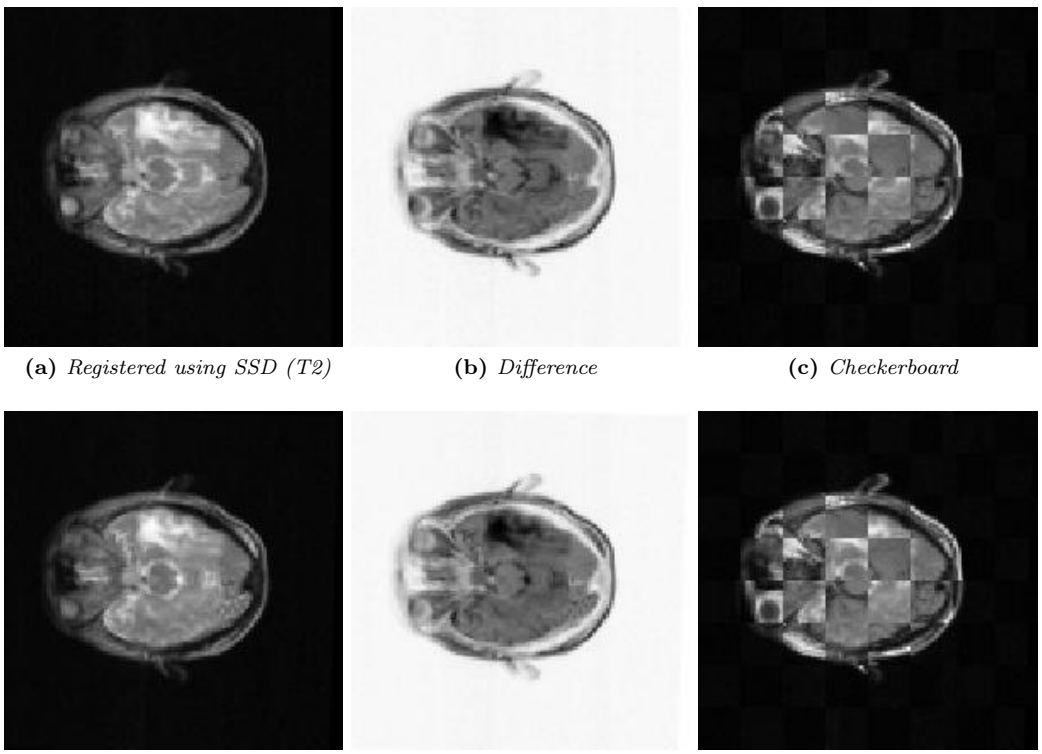


(c) *PD*



(d) *PET*

Figure 5.2: *Image montages of the brain data used in the following examples.*



(a) *Registered using SSD (T2)* **(b)** *Difference* **(c)** *Checkerboard*

(d) *Registered using NCC (T2)* **(e)** *Difference* **(f)** *Checkerboard*

Figure 5.3: *Example of multimodal registration on T2 and T1-weighted MRIs using SSD and NCC.*

NCC. The best results for these distance measures were obtained by starting at a higher level than the minimum. For SSD, 64-by-64-by-8 seemed ideal and for NCC the registration starting at the highest level gave the best results.

Example 5.1.1 (Registration of T1 and T2-weighted brain MRI using SSD and NCC)**.** Figures 5.3a-5.3c shows the results of attempting to register two brain MRIs (T1 and T2-weighted) from the same patient using SSD. Figures 5.3d-5.3f shows results from the same dataset using NCC.

Both registrations seem to have given decent results, but this is almost solely due to the affine linear preregistration and the small initial distance. The results did not improve using non-parametric registration with either distance measure.

That being said, there are tasks at which these distance measures can be used to some degree of success. Their simplicity can also be an advantage in that they use less memory and time when they do succeed.

5.1.2 Mutual Information with multimodal brain images

Mutual Information has been considered as one of the best distance measures for registering multimodal images [30]. My testing does seem to confirm that this distance measure is viable for registering between most modalities. I will show a couple of examples of the results I have gotten, both from brain and kidney images.

Example 5.1.2 (Registration of multimodal brain images using MI)**.** As we can see in Figures 5.4a-5.4c, the results on the T2 image seem very good with Mutual Information. Most of the general features are correctly matched, but there there is some difference in the angle of the two

images. Figures 5.4d-5.4f shows results from registering a PD-weighted MRI to the same reference image. Here the results are ever better.

For both of these registrations I used the 64-by-64-by-8 images as the lowest level. The time taken was 154.78s for T2 and 89.22s for the PD image. A 32-by-32 grid was used for the splines in the estimation of the densities.

The biggest difficulties arose when trying to register the PET scan images. By nature PET scans show a lot less structure than MRIs, and they are thus more difficult to register. Examining the image in Figure 5.4i shows that the edges in the PET scan do not match the ones for the reference image. The time taken was 86.40s. The second level was used as the lowest level on this registration as well.

5.1.3 Normalized Gradient and Hessian Fields with multimodal brain images

Normalized Gradient Field and Normalized Hessian Field have in common that they are differential based distance measures. They will therefore try to match the edges and intensity changes in the two images whilst ignoring the areas with constant intensities. They do this by focusing on different aspects of the change, however; the first derivative and thus the gradient looks at all intensity change whereas the second derivative from the Hessian only notices changes in the gradient. Areas of constant change (like a ramp) will therefore be noticed by the gradient but not by the Hessian.

Example 5.1.3 (Registration of multimodal brain images using NGF)**.** As Figure 5.5 shows, Normalized Gradient Field does very well registering both the T2 and PD-weighted MRIs. For the PET scan the results are much better than the ones achieved from Mutual Information. Here we see that the edges in the registered PET scan have mostly been aligned. The time taken for the three registrations were 244.30s for the T2, 200.26s for the PD and 130.60s for the PET scan. All registrations were performed starting at the second level. For the T2 and PD registrations an edge parameter of 0.2 was sufficient, whereas the PET registration was performed with the parameter set to 3.

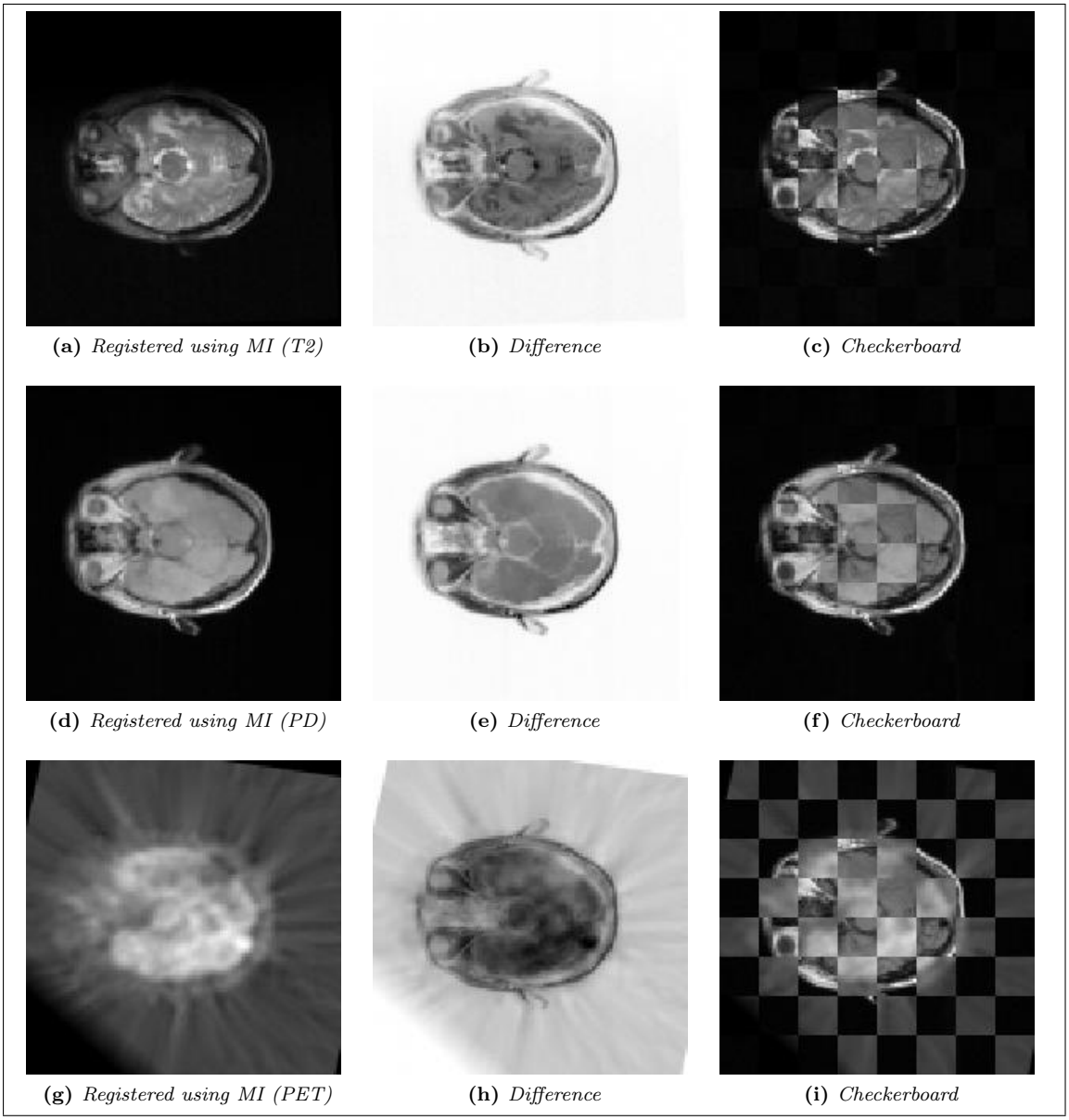
Example 5.1.4 (Registration of multimodal brain images using NHF)**.** We now turn to the Normalized Hessian Field. The results on the registrations are very similar to the ones from NGF, as may be expected for two differential based distance measures, though the time for NHF is longer: 488.45s for T2, 935.80s for PD and 660.56 for the PET scan. Like with the NGF a low edge parameter (here 0.01) was used on the two first images and a higher (here 1) was used on the PET scan.

Example 5.1.5 (Combined Normalized Gradient and Hessian Fields applied on brain data)**.** Another idea for distance measures is to make a composite distance measure from two or more distance measures. In this example I have combined Normalized Gradient and Hessian Fields into one distance measure, calculating each in turn and adding them together as

$$\mathcal{D}^{\mathrm{NGHF}} = 0.8\mathcal{D}^{\mathrm{NGF}} + 0.2\mathcal{D}^{\mathrm{NHF}}. \tag{5.1.1}$$

This would ideally have the advantage of being able to align both the first and second derivatives, which might raise the accuracy of the registration.

The registration combining Normalized Gradient and Hessian Fields gives very similar results to the ones using only NGF or NHF, closest resembling the results from NHF alone. The time taken was 1222.1s for T2, 1839.2s for PD, and 857.74s for PET.

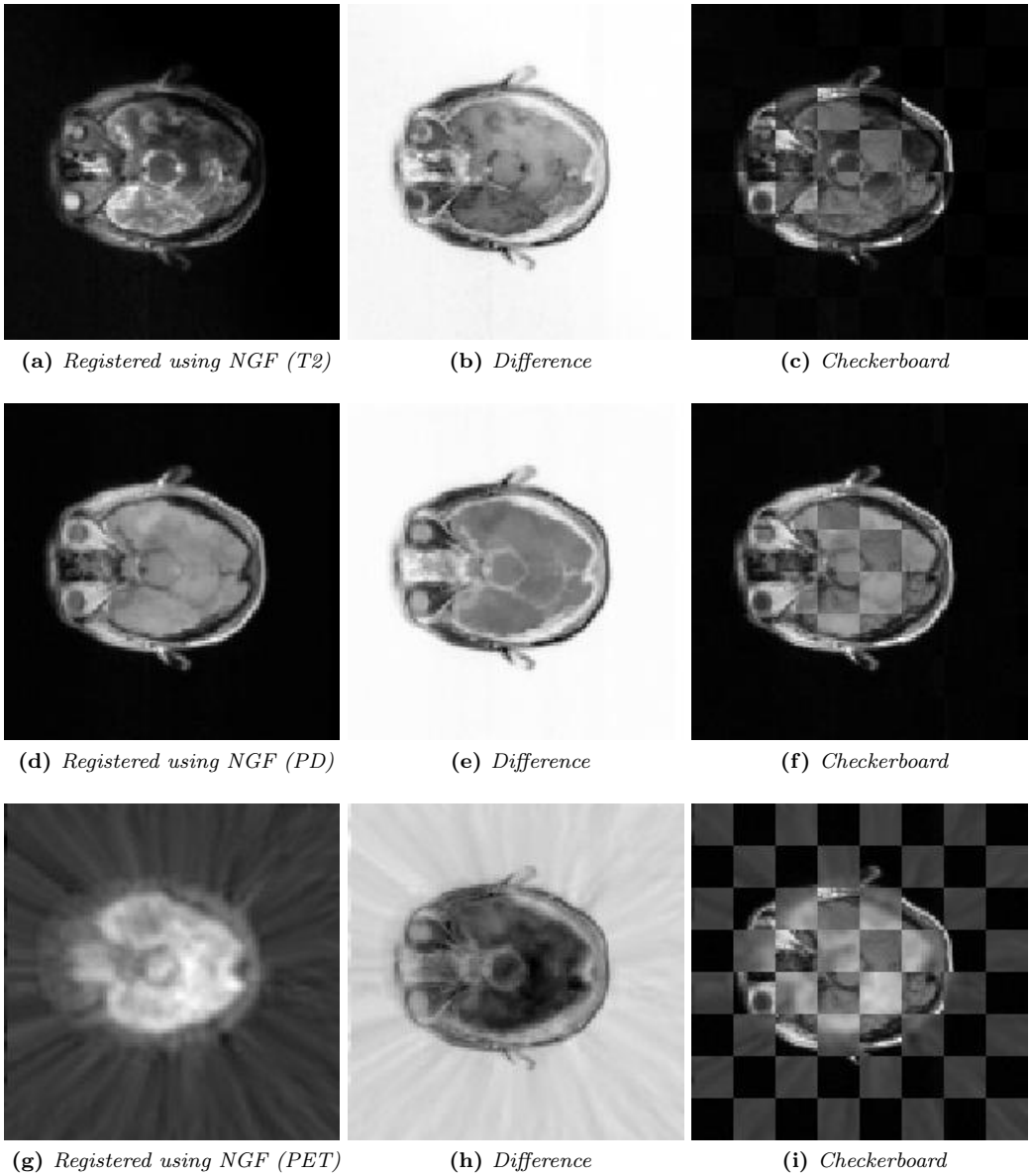


(a) *Registered using MI (T2)* **(b)** *Difference* **(c)** *Checkerboard*

(d) *Registered using MI (PD)* **(e)** *Difference* **(f)** *Checkerboard*

(g) *Registered using MI (PET)* **(h)** *Difference* **(i)** *Checkerboard*

Figure 5.4: *Example of multimodal registration of brain images using MI. Images show one slice*

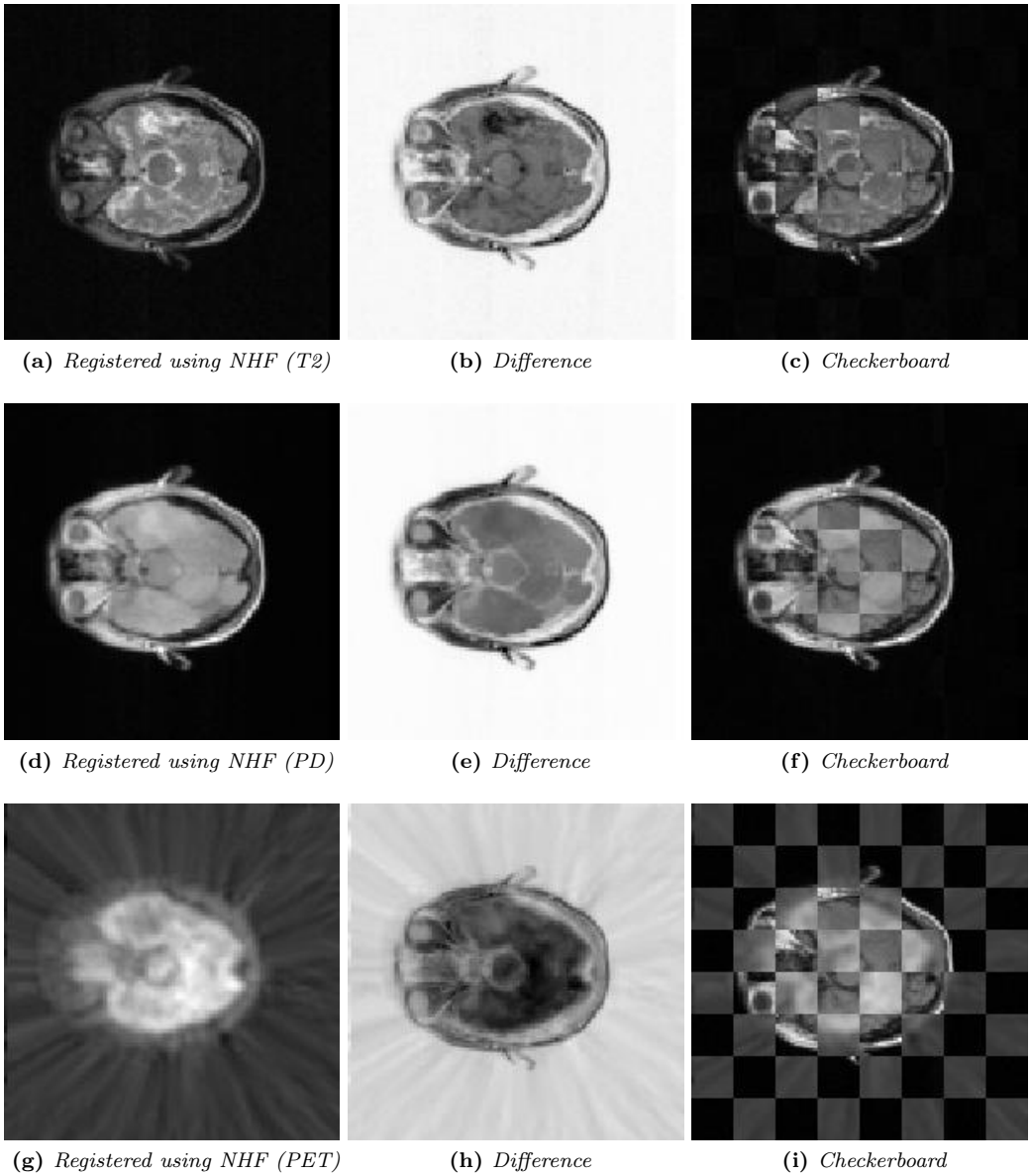


(a) *Registered using NGF (T2)* **(b)** *Difference* **(c)** *Checkerboard*

(d) *Registered using NGF (PD)* **(e)** *Difference* **(f)** *Checkerboard*

(g) *Registered using NGF (PET)* **(h)** *Difference* **(i)** *Checkerboard*

Figure 5.5: *Example of multimodal registration of brain images using NGF.*

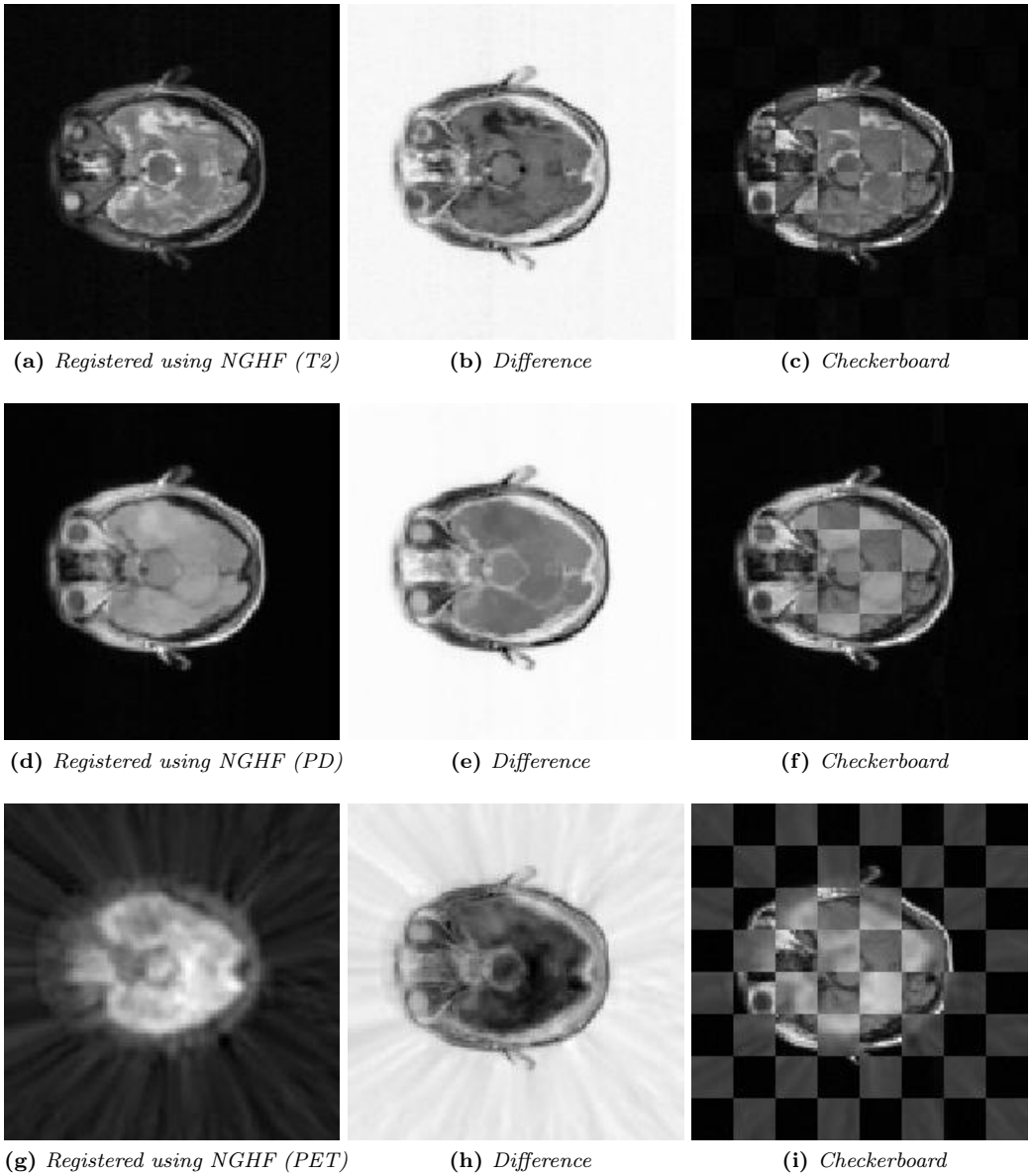


(a) *Registered using NHF (T2)* **(b)** *Difference* **(c)** *Checkerboard*

(d) *Registered using NHF (PD)* **(e)** *Difference* **(f)** *Checkerboard*

(g) *Registered using NHF (PET)* **(h)** *Difference* **(i)** *Checkerboard*

Figure 5.6: *Example of multimodal registration of brain images using NHF.*



(a) *Registered using NGHF (T2)* **(b)** *Difference* **(c)** *Checkerboard*

(d) *Registered using NGHF (PD)* **(e)** *Difference* **(f)** *Checkerboard*

(g) *Registered using NGHF (PET)* **(h)** *Difference* **(i)** *Checkerboard*

Figure 5.7: *Example of multimodal registration of brain images using NGHF.*

Distance measure	Regularizer	Optional Parameters	Time (s)
NCC	TPS, $\alpha = 5 \cdot 10^{-6}$	None	289.70
MI	TPS, $\alpha = 10^{-7}$	nT = nR = 32	953.36
NGF	TPS, $\alpha = 10$	$\eta = 0.01$	3231.25
NHF	TPS, $\alpha = 10$	$\eta = 0.01$	6617.86
NGHF	TPS, $\alpha = 10$	$\eta = 0.01$	8266.24

Table 5.2: Parameters and time for registration of kidney data.

5.2 Registration of kidney MRI

The second type of data I have attempted to register is a time sequence of 20 images taken from a kidney injected with a contrast agent. Again we have a set of images that display change in contrast for the same type of tissue, and as such SSD can not be used. NCC actually manges to achieve some result on this dataset. One added complication when dealing with these images is that the relevant part of the image, the kidneys, are surrounded by other types of tissue. This means that even matching the general shape of the kidney might prove difficult. For the sequence I have used, the first 10 images display relatively minor movement. The last images are not as aligned, and the registration is thus more difficult.

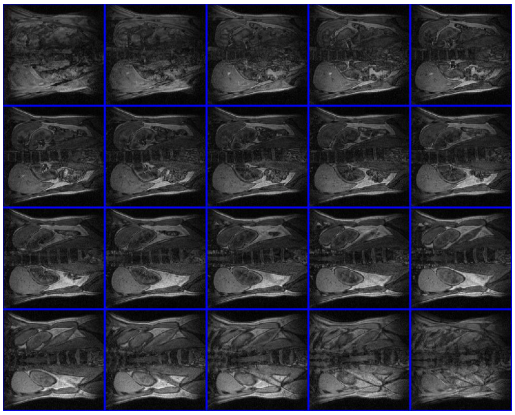
To get a better registration of the kidney, which is what we are interested in, it helps to *crop* the image by reducing it to an area of interest or use some method to segment out the kidney. In the following results I have used a box of size 96-by-54-by-20 containing the left kidneys shown in Figure 5.8. The data was then interpolated on a grid of size 128-by-64-by-16 for the following registrations.

The results are discussed in the following examples. For each example one figure is supplied, showing a spatial slice from registrations on the $3D$-volumes of the kidney at four different times. The registered template, an image showing the difference of the registered template and the reference, and a checkerboard image of the two are shown. The first image in the sequence ($t = 0$) is chosen as the reference image for the registration, and is denoted $\mathcal{R}$. The three registered images shown are at times $t = 5$, 11 and 19 and denoted $\mathcal{T}(y)_5$, $\mathcal{T}(y)_{11}$ and $\mathcal{T}(y)_{19}$, respectively. Image montages of the registered volumes are shown in Section 7.1.2. The unregistered images are shown in Figure 5.9. The parameters and time taken for the registration is shown in Table 5.2. The parameters and timespan of each registration is shown in table 5.2.

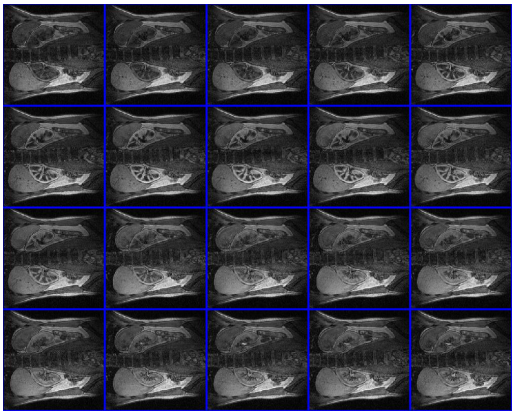
Example 5.2.1 (Kidney registration using NCC)**.** The registration using Normalized Cross-Correlation does not work very well. Some of the images seem to display only minor differences, but this is most likely only a result of the small initial distance. Figure 5.10 shows the results. As the images show, the registration has not been able to match the edges very well. This is evident both in the checkerboard image, where following the edges around the kidney shows gaps between the checkerboard tiles, and from the difference image, where it is clear that there are double borders around the kidney. This indicates that the edges of the registered image and the reference do not overlap. Time taken with NCC was 4 minutes, 49.70 seconds. The reason for the short time is that the registration has done very few iterations in total.

Example 5.2.2 (Kidney registration using MI)**.** The registration using Mutual Information performs much better than the one using NCC, but there still seems to be some problems aligning some of the edges. Figure 5.11 shows the results. In particular, the checkerboard image seen in Figure 5.11i shows some broken edges, particularly in the lower right quadrant. This is also seen in the difference image, where it is clear that there is a double edge in this area. Time taken with MI was 15 minutes, 53.36 seconds.

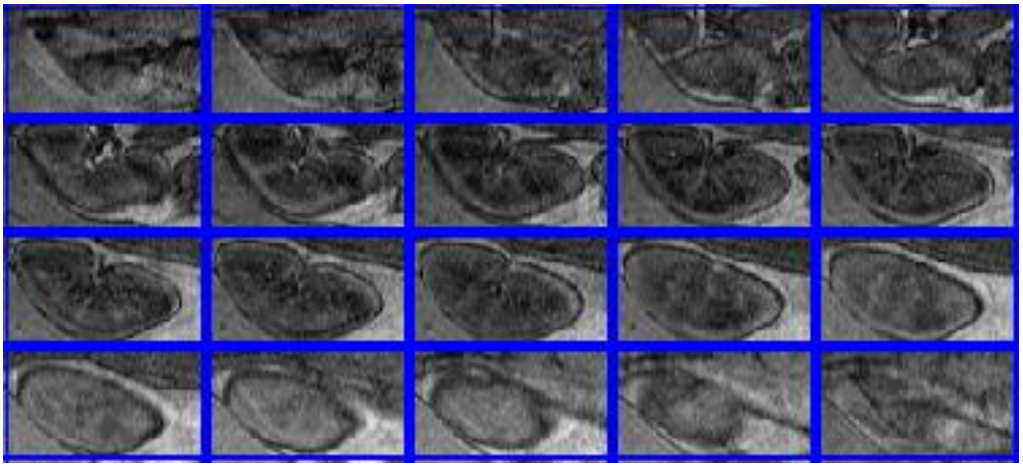
Example 5.2.3 (Kidney registration using NGF)**.** The registration using Normalized Gradient Fields seem to be give very similar results as the MI registration: There is still some problems registering some of the edges. Time taken with NGF was 53 minutes, 51.25 seconds.



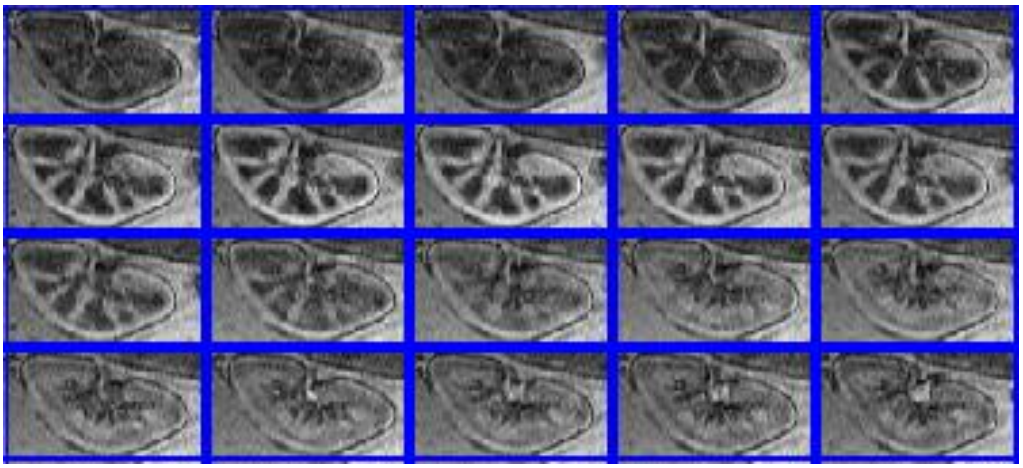
(a) *Uncropped kidney data*



(b) *Time series of abdomen slice*

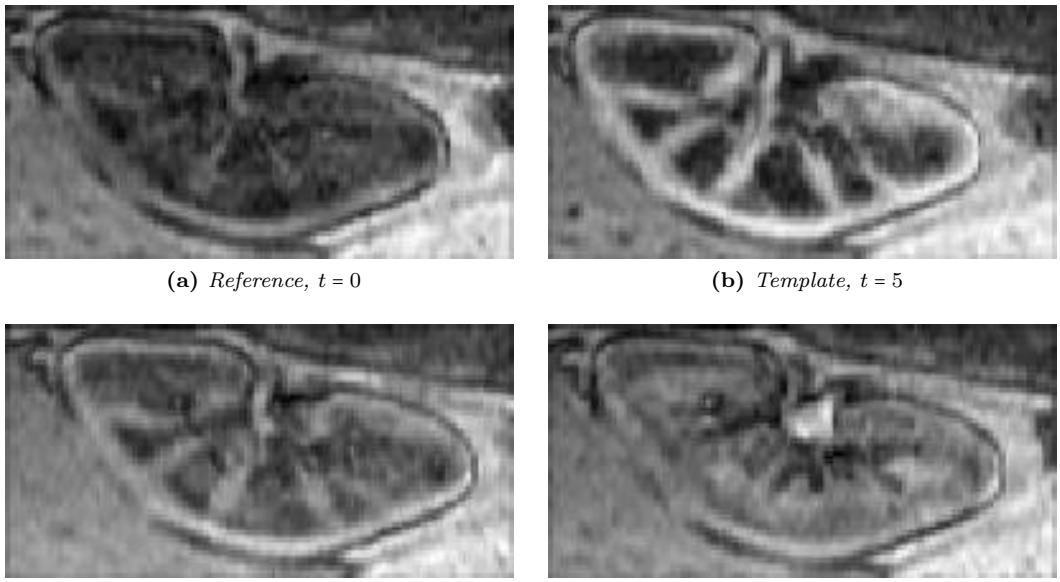


(c) *Cropped kidney data*



(d) *Time series of kidney slice*

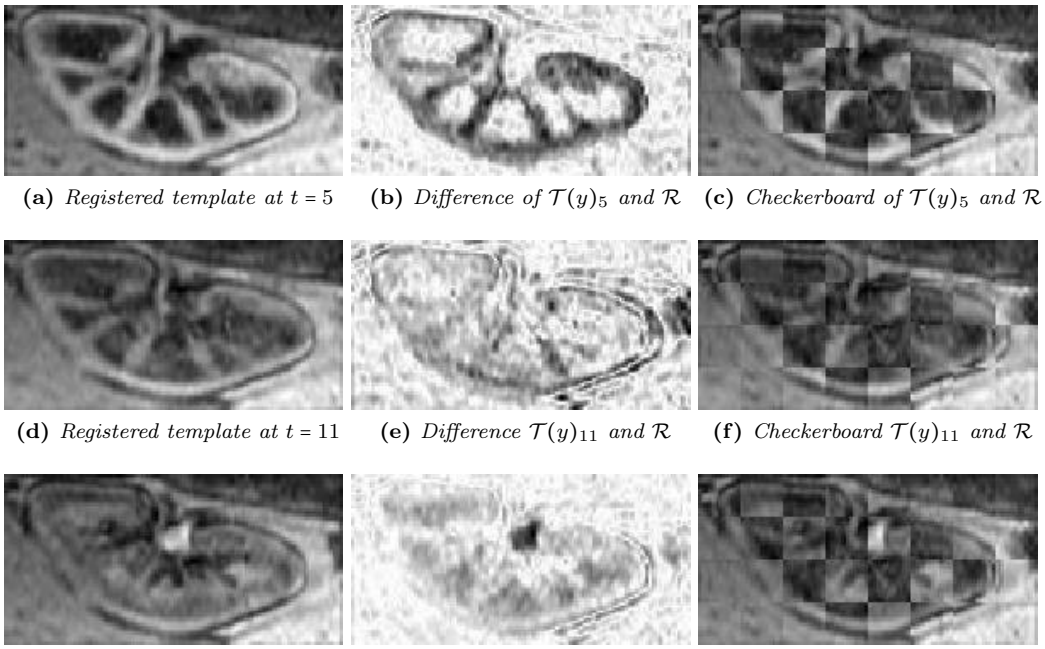
Figure 5.8: *Kidney data used in following registrations. Figure 5.8a shows the original data from the first image, and the cropped version used for registration is shown in Figure 5.8c. Figures 5.8b and 5.8d shows one fixed slice (full size and cropped, respectively) of the data through the time series.*



(a) *Reference,* $t = 0$ **(b)** *Template,* $t = 5$

(c) *Template,* $t = 11$ **(d)** *Template,* $t = 19$

Figure 5.9: *Unregistered version of the slices shown in the following examples.*



(a) *Registered template at* $t = 5$ **(b)** *Difference of* $\mathcal{T}(y)_5$ *and* $\mathcal{R}$ **(c)** *Checkerboard of* $\mathcal{T}(y)_5$ *and* $\mathcal{R}$

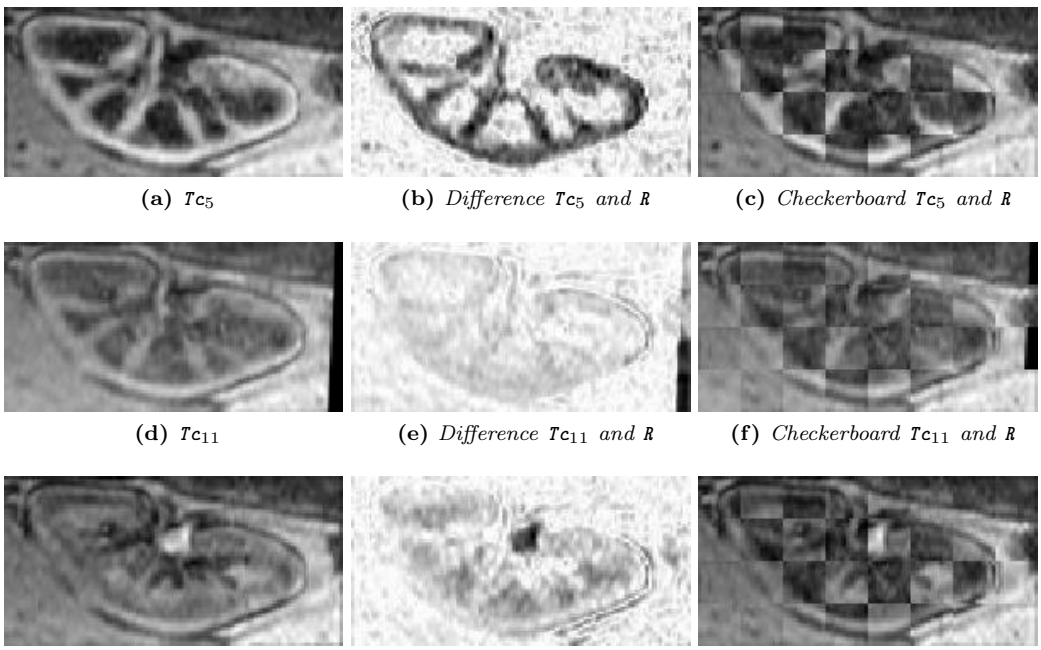
(d) *Registered template at* $t = 11$ **(e)** *Difference* $\mathcal{T}(y)_{11}$ *and* $\mathcal{R}$ **(f)** *Checkerboard* $\mathcal{T}(y)_{11}$ *and* $\mathcal{R}$

(g) *Registered template at* $t = 19$ **(h)** *Difference,* $\mathcal{T}(y)_{19}$ *and* $\mathcal{R}$ **(i)** *Checkerboard,* $\mathcal{T}(y)_{19}$ *and* $\mathcal{R}$

Figure 5.10: *Results of kidney registration from Example 5.2.1 using NCC.*

Example 5.2.4 (Kidney registration using NHF)**.** The registration using Normalized Hessian Fields gives far better results. The edges are better aligned than in the previous examples, as can be seen in Figure 5.13. The time taken with NHF was 110 minutes, 17.86 seconds.

Example 5.2.5 (Kidney registration using NGHF)**.** The registration using combined Normalized Gradient and Hessian Fields yields the best results on this dataset. The time taken with NGHF was 137 minutes, 46.24 seconds, the highest of the distance measures I have used, but as can be

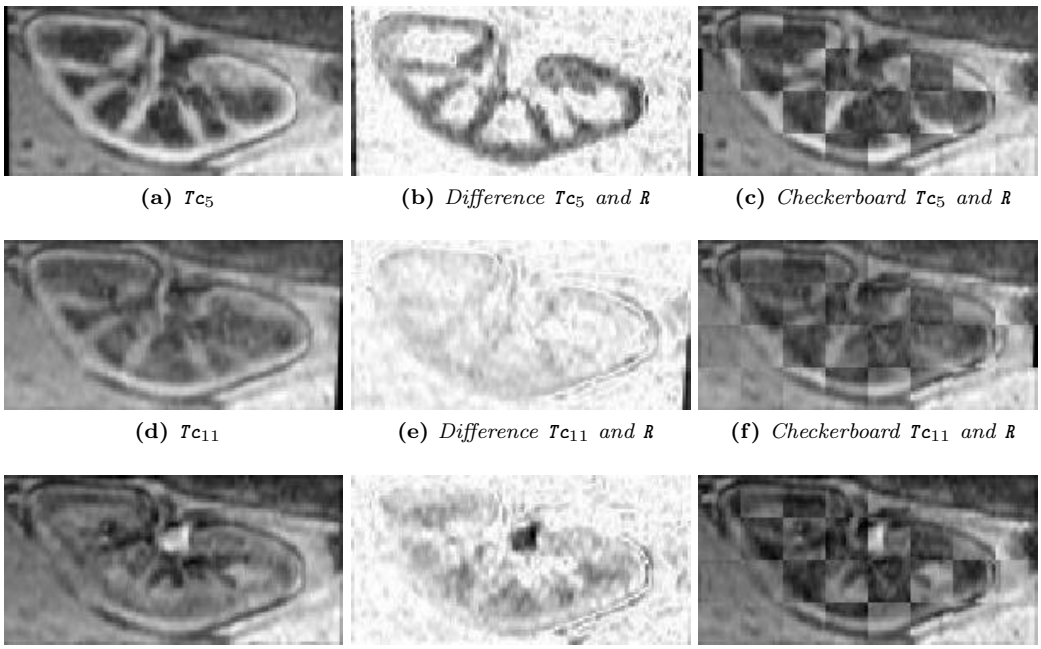


(a) Tc_5 **(b)** *Difference* Tc_5 *and* R **(c)** *Checkerboard* Tc_5 *and* R

(d) Tc_{11} **(e)** *Difference* Tc_{11} *and* R **(f)** *Checkerboard* Tc_{11} *and* R

(g) *Registered* Tc_{19} **(h)** *Difference,* Tc_{19} *and* R **(i)** *Checkerboard,* Tc_{19} *and* R

Figure 5.11: *Results of kidney registration from Example 5.2.2 using MI.*



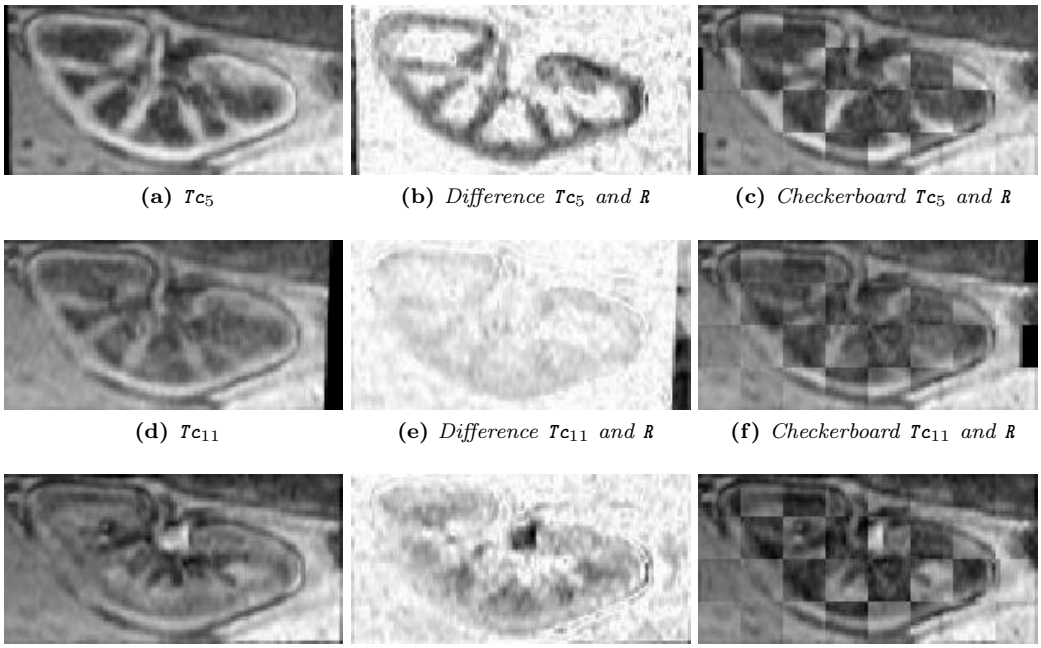
(a) Tc_5 **(b)** *Difference* Tc_5 *and* R **(c)** *Checkerboard* Tc_5 *and* R

(d) Tc_{11} **(e)** *Difference* Tc_{11} *and* R **(f)** *Checkerboard* Tc_{11} *and* R

(g) *Registered* Tc_{19} **(h)** *Difference,* Tc_{19} *and* R **(i)** *Checkerboard,* Tc_{19} *and* R

Figure 5.12: *Results of kidney registration from Example 5.2.3 using NGF.*

seen in Figure 5.14, this gives the best registration. Following the edges in the checkerboard plots show almost no discontinuities, and the difference images shown very few double edges.

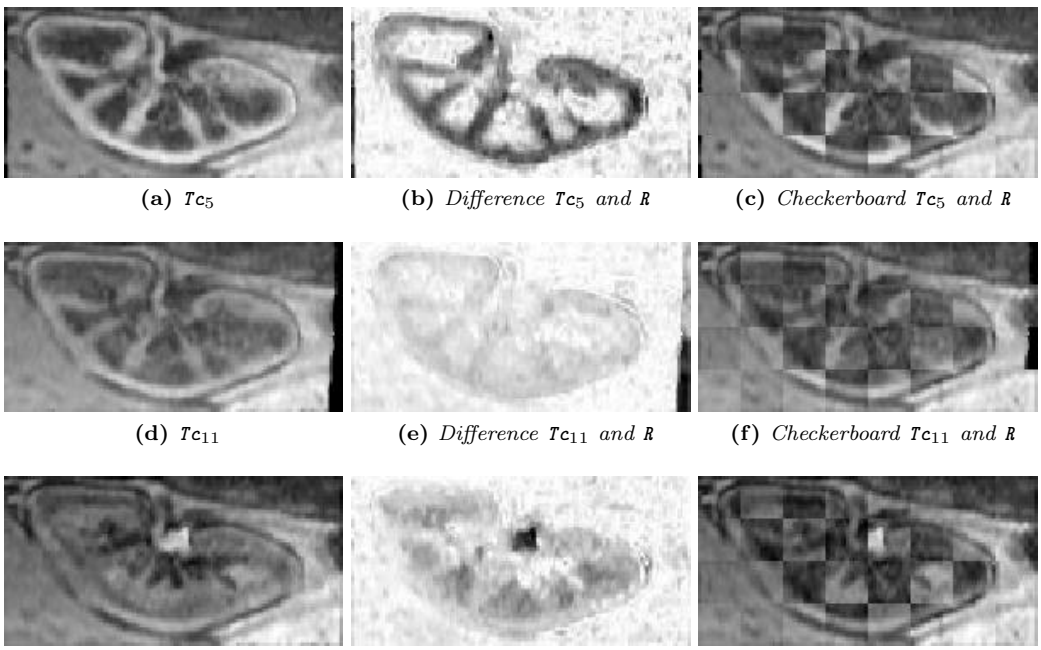


(a) Tc_5 **(b)** *Difference* Tc_5 *and* R **(c)** *Checkerboard* Tc_5 *and* R

(d) Tc_{11} **(e)** *Difference* Tc_{11} *and* R **(f)** *Checkerboard* Tc_{11} *and* R

(g) *Registered* Tc_{19} **(h)** *Difference,* Tc_{19} *and* R **(i)** *Checkerboard,* Tc_{19} *and* R

Figure 5.13: *Results of kidney registration from Example 5.2.4 using NHF.*



(a) Tc_5 **(b)** *Difference* Tc_5 *and* R **(c)** *Checkerboard* Tc_5 *and* R

(d) Tc_{11} **(e)** *Difference* Tc_{11} *and* R **(f)** *Checkerboard* Tc_{11} *and* R

(g) *Registered* Tc_{19} **(h)** *Difference,* Tc_{19} *and* R **(i)** *Checkerboard,* Tc_{19} *and* R

Figure 5.14: *Results of kidney registration from Example 5.2.5 using NGHF.*

Chapter 6

Conclusions

We present some conclusions that arose from the work and the experiences gained during the time spent on this thesis.

FAIR

The FAIR software and book is a great way to learn the basics of registration, or to start registering without complete knowledge of the registration process using the tutorials.

The book [18] contains theory behind most of the building bricks needed for registration, and could be used as a tutorial on how to make your own software. One aspect I missed from the book was a wider discussion on the choice of parameters in regularization and distance measures.

One drawback to the software is, as stated in the book, that the software is designed with clarity and ease of reading in mind, not with efficiency and speed. This meant that most of the data used needed to be scaled down in order for Matlab to have enough memory to perform the registration.

With that said, the software performed well registering images. The way it is built up also makes it rather easy to incorporate ones own methods into the package. For instance, setting up a new distance measure proved to be a simple process.

Distance measures

One of the main focus points of my work on the thesis has been the choice of distance measure in medical image registration. Of the distance measures I have tested I have focused mainly on testing the performance Normalized Gradient Field and implementing and testing Normalized Hessian Field. I have used Mutual Information as a reference for the performance of these distance measures.

My results indicate that both differential based distance measures outperform Mutual Information on alignment if not on speed, however the FAIR software uses a C script to calculate the densities of Mutual Information. This makes it difficult to directly compare the temporal performance. Normalized Gradient Field was noticeable quicker than Normalized Hessian Field in most registrations. The combined Normalized Gradient and Hessian Fields was, unsurprisingly, the slowest of the measures, but I used an inefficient implementation and split the calculations in order to save memory. Because of these concerns, I have focused on alignment.

On the brain data, Normalized Gradient Field gave the most consistently good results, whereas the best results on the kidney data were from registering with a combination of gradient and hessian based information.

One problem with both differential based distance measures is that a large initial distance may prove problematic even for parametric registration since a decent direction might be difficult to find. In the cases where I encountered this problem a solution was to perform a quick parametric

registration using either Mutual Information or Normalized Cross-Correlation to roughly align the images, and then use the parameters as a starting guess for Normalized Gradient or Hessian Field.

Outlook

If there is one sure conclusion to be drawn from this work it is that I wish there was more time. The work I have done in testing and implementing registration methods has given interesting results, but have not given me a firm conclusion in a definitive distance measure for multimodal registration of medical images. Normalized Gradient Field and Hessian Field are both promising distance measures, and they each seem to be good at registering medical images. The combination of them is therefore a tempting idea, and the results seem to indicate that it provides very good results.

One aspect I would like to work more on is the choice of parameters. Much of the work felt like a trial and error approach in choosing the right parameters, and I would like to see a better way of doing this.

Another aspect of image registration where there is much work to be done is in the aspect of validation or evaluation of image registration. As of yet there are no good methods of comparing results from different registration procedures, and often we are left with a very subjective evaluation. There are a number of projects working on developing evaluation tools, such as [4, 29].

Chapter 7

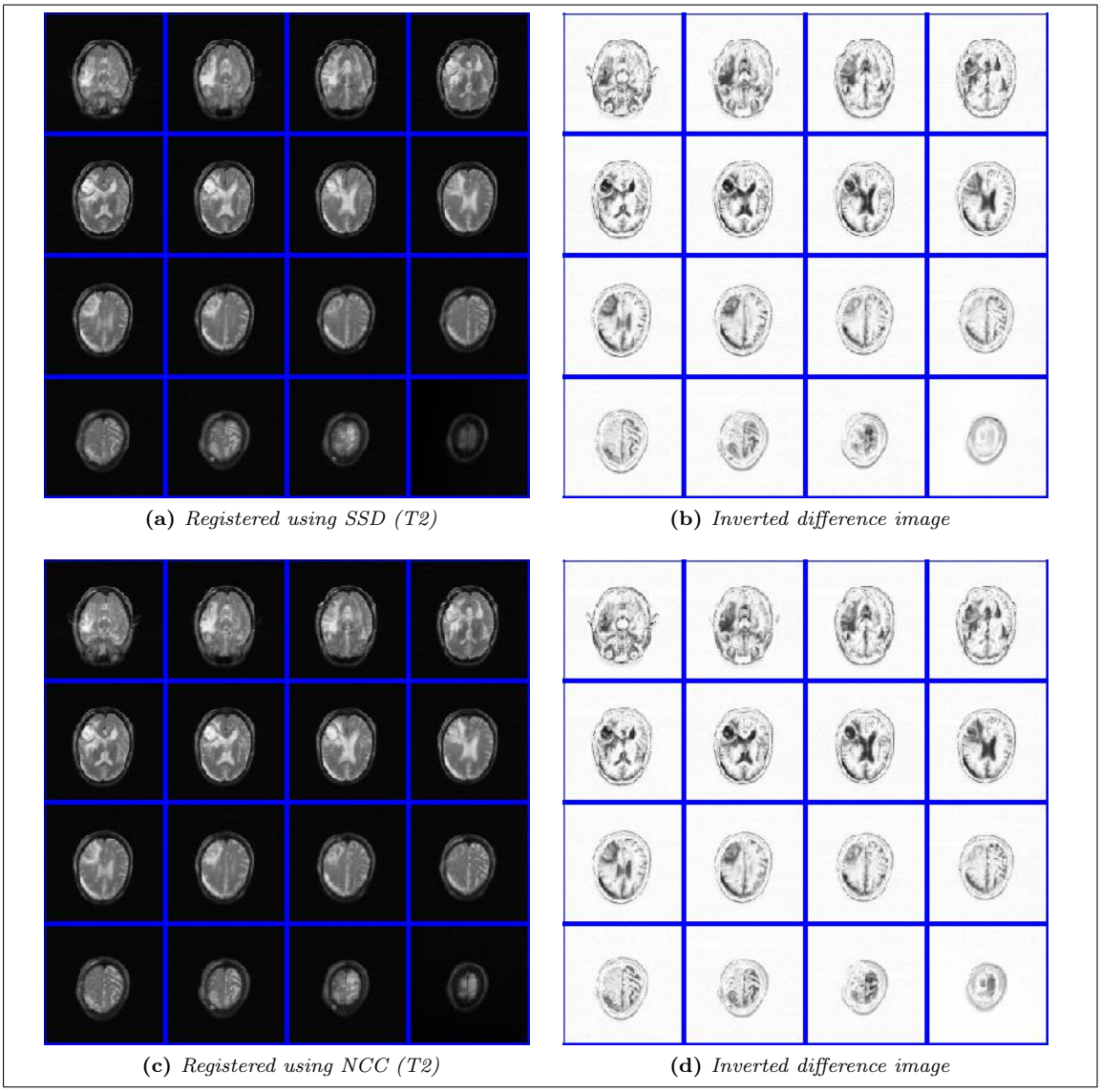
Appendix

7.1 Additional figures

As not to fill up the main part of this thesis with additional figures, some of them have been placed here in the Appendix. These include image montages of the registration results on the brain and kidney data that I did not find room for in Chapter 5.

7.1.1 Brain registration

On the following pages image montages of the registered brain volumes are shown.



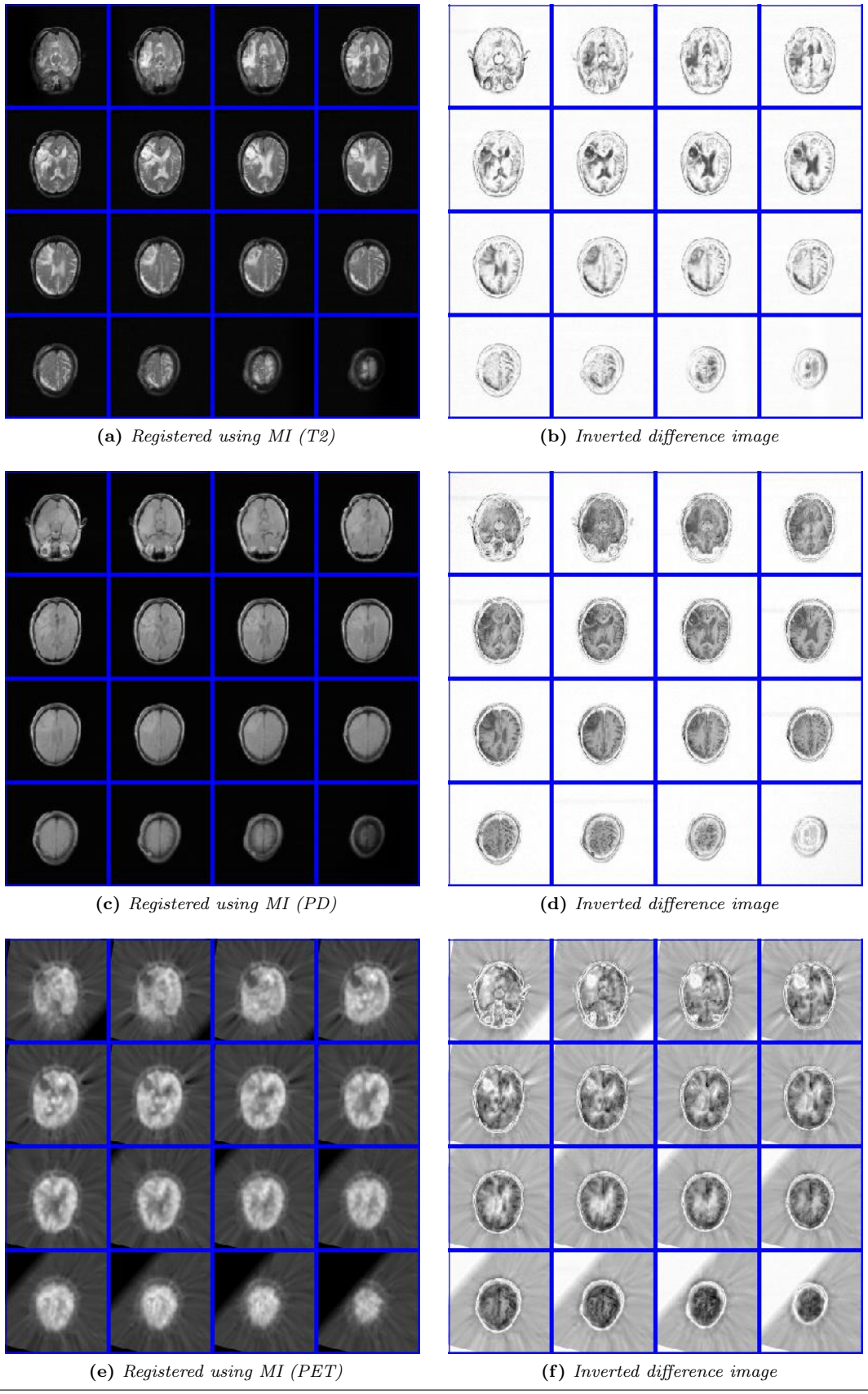
(a) *Registered using SSD (T2)*

(b) *Inverted difference image*

(c) *Registered using NCC (T2)*

(d) *Inverted difference image*

Figure 7.1: *Image montages of the results obtained from registering brain images from Chapter 5 using Sum of Squared Differences and Normalized Cross-Correlation as distance measure.*



(a) *Registered using MI (T2)*

(b) *Inverted difference image*

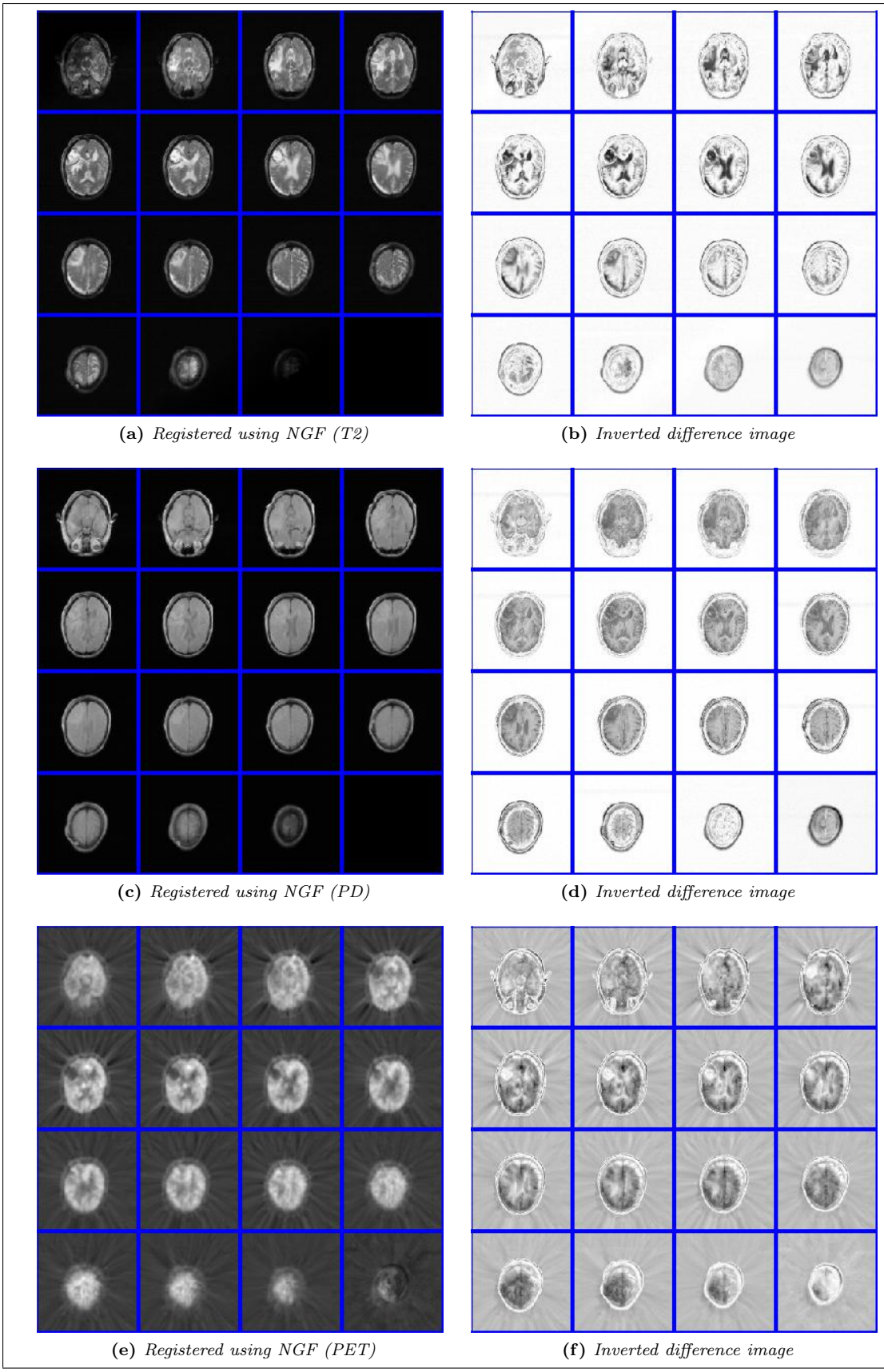
(c) *Registered using MI (PD)*

(d) *Inverted difference image*

(e) *Registered using MI (PET)*

(f) *Inverted difference image*

Figure 7.2: *Image montages of the results obtained from registering brain images from Chapter 5 using Mutual Information as distance measure.*



(a) *Registered using NGF (T2)*

(b) *Inverted difference image*

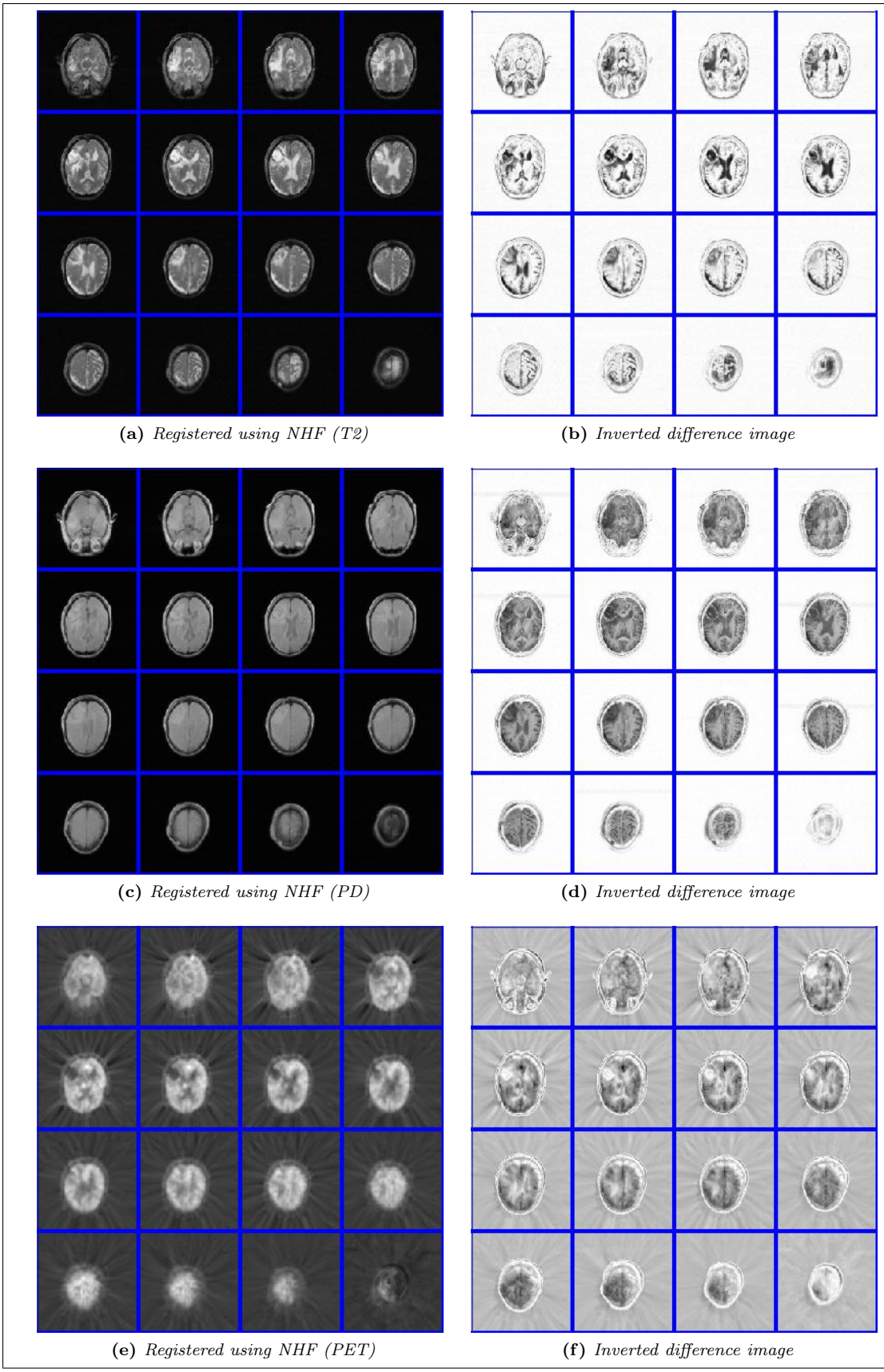
(c) *Registered using NGF (PD)*

(d) *Inverted difference image*

(e) *Registered using NGF (PET)*

(f) *Inverted difference image*

Figure 7.3: *Image montages of the results obtained from registering brain images from Chapter 5 using Normalized Gradient Fields as distance measure.*



(a) *Registered using NHF (T2)*

(b) *Inverted difference image*

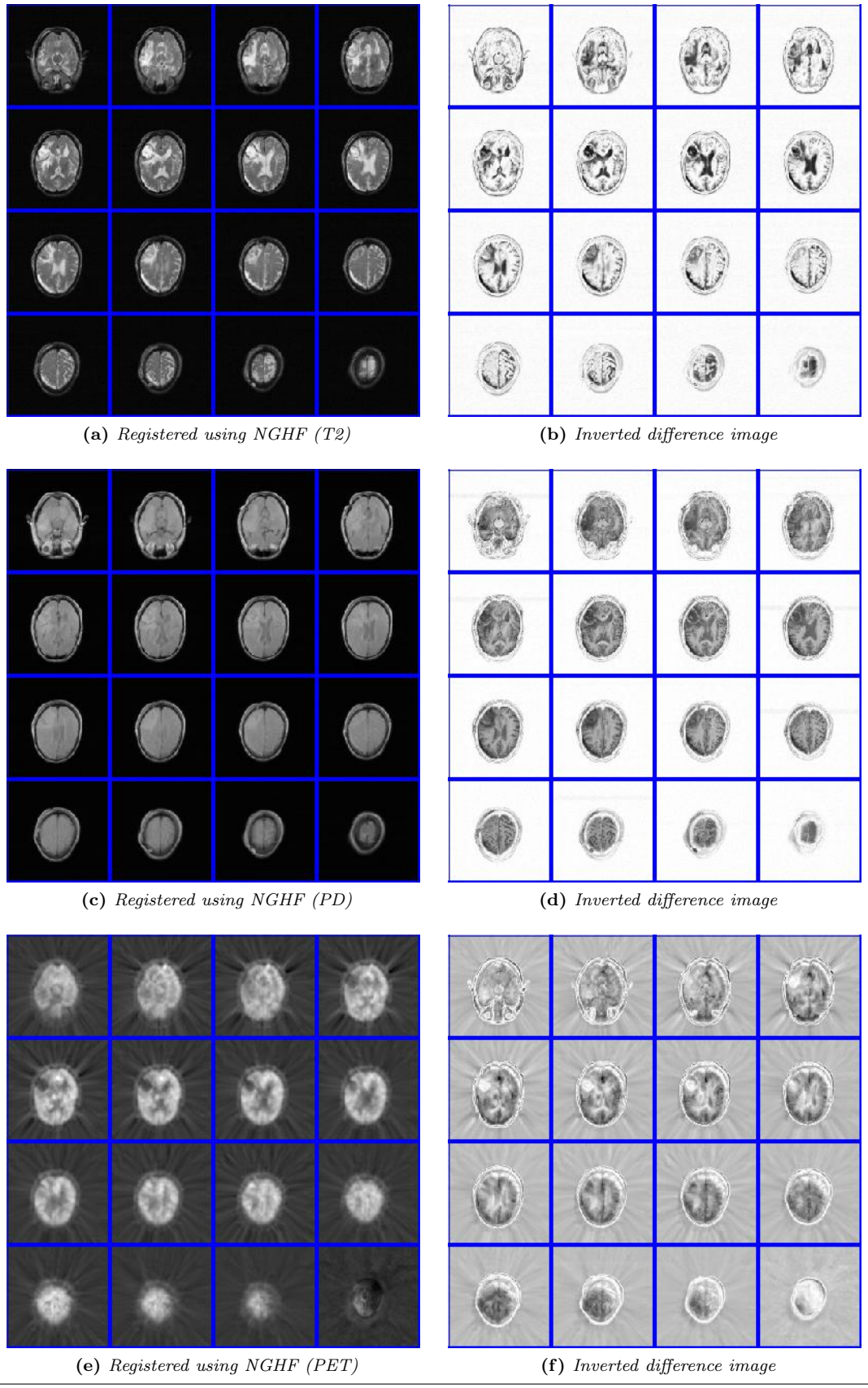
(c) *Registered using NHF (PD)*

(d) *Inverted difference image*

(e) *Registered using NHF (PET)*

(f) *Inverted difference image*

Figure 7.4: *Image montages of the results obtained from registering brain images from Chapter 5 using Normalized Hessian Fields as distance measure.*



(a) *Registered using NGHF (T2)*

(b) *Inverted difference image*

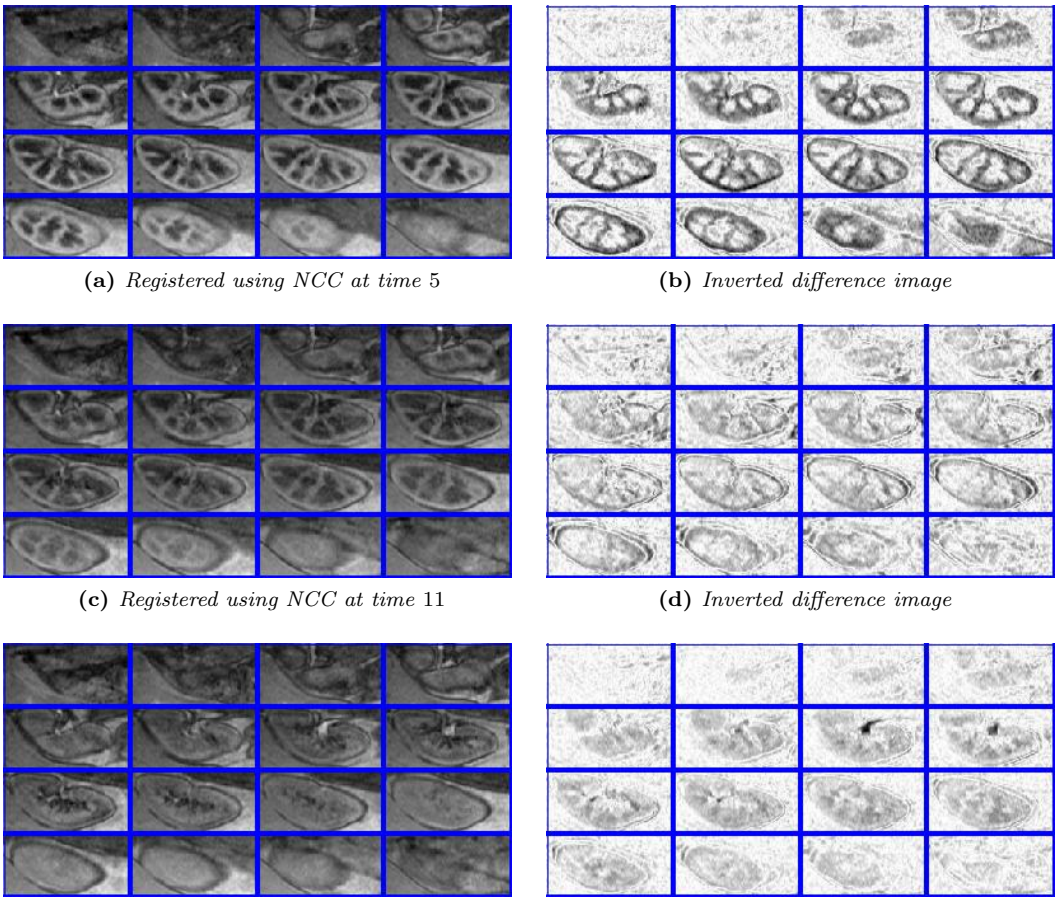
(c) *Registered using NGHF (PD)*

(d) *Inverted difference image*

(e) *Registered using NGHF (PET)*

(f) *Inverted difference image*

Figure 7.5: *Image montages of the results obtained from registering brain images from Chapter 5 using combined Normalized Gradient and Hessian Fields as distance measure.*



(a) *Registered using NCC at time* 5 **(b)** *Inverted difference image*

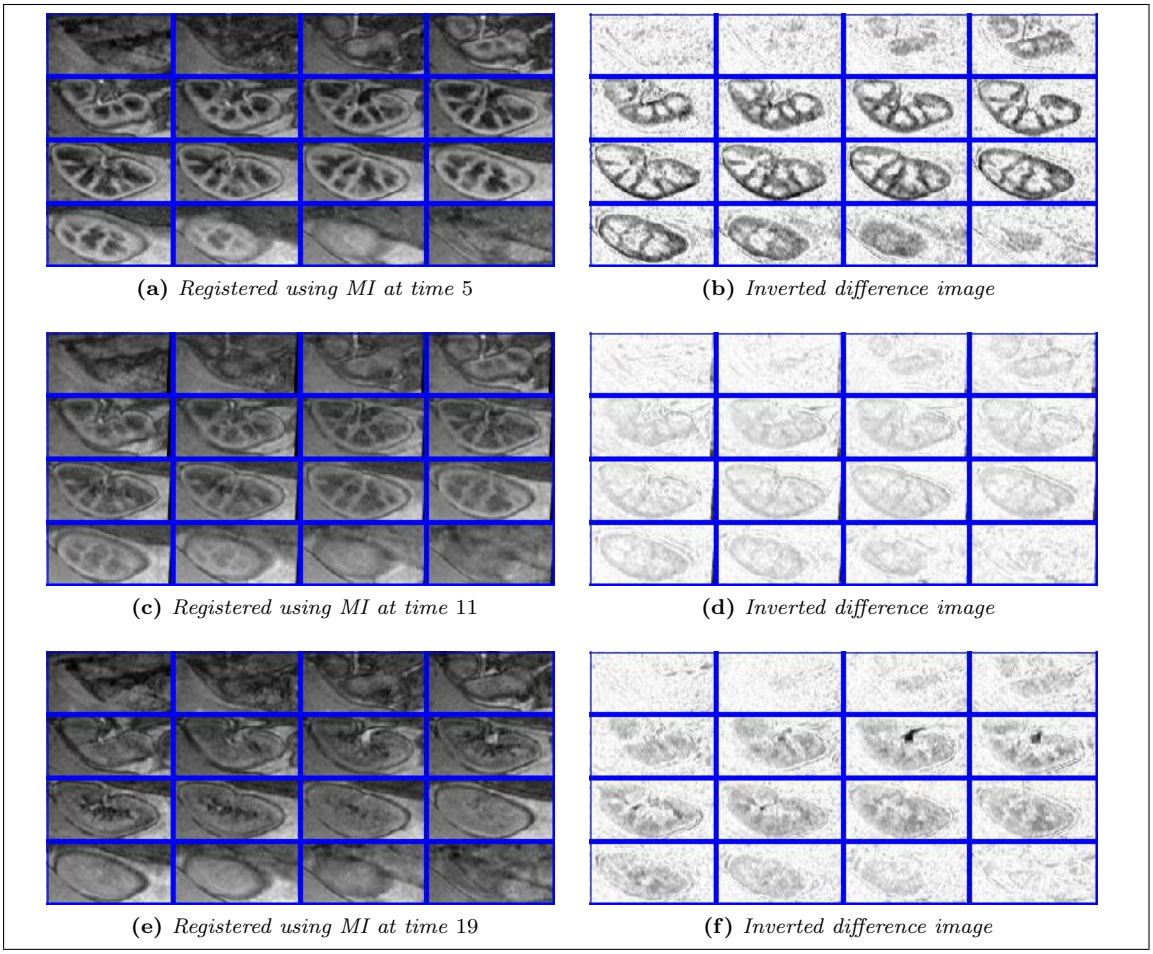
(c) *Registered using NCC at time* 11 **(d)** *Inverted difference image*

(e) *Registered using NCC at time* 19 **(f)** *Inverted difference image*

Figure 7.6: *Image montages of some of the results obtained from registering kidney images from Chapter 5 using combined Normalized Gradient and Hessian Fields as distance measure.*

7.1.2 Kidney registration

In the following pages figures depicting the registered kidney volumes are shown.

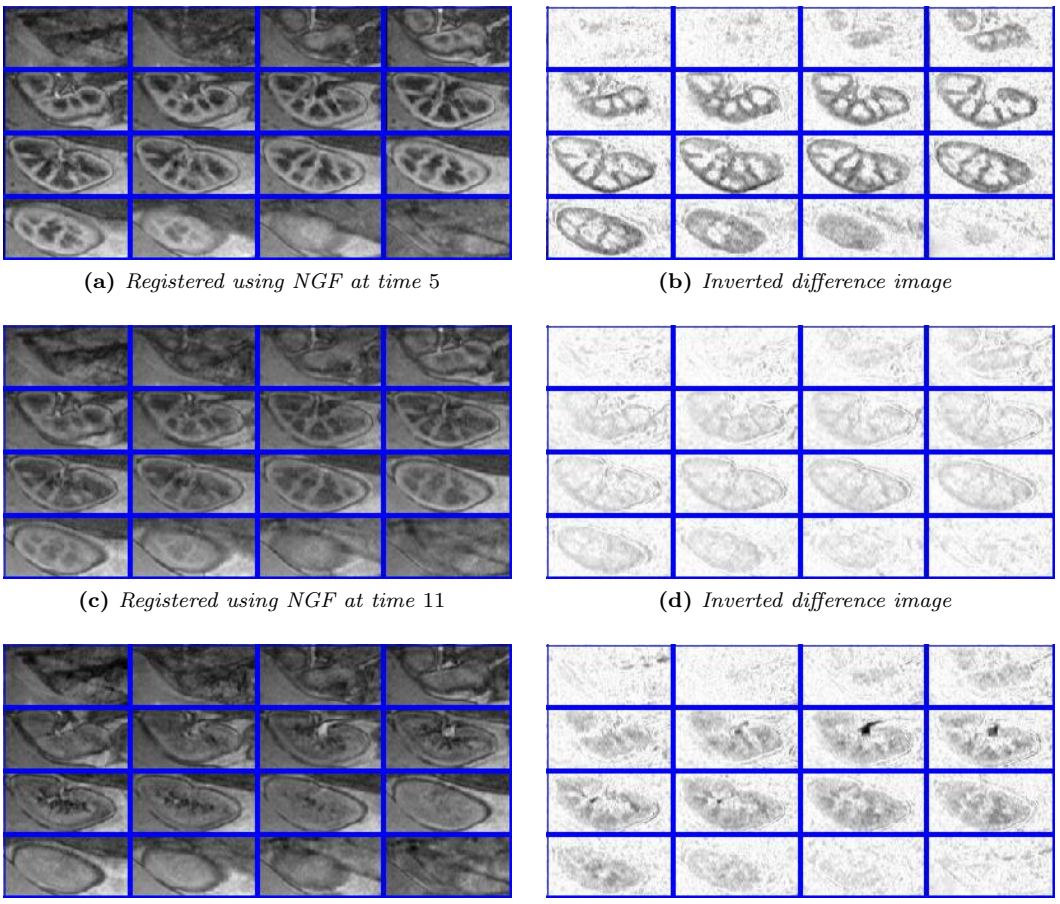


(a) *Registered using MI at time* 5 **(b)** *Inverted difference image*

(c) *Registered using MI at time* 11 **(d)** *Inverted difference image*

(e) *Registered using MI at time* 19 **(f)** *Inverted difference image*

Figure 7.7: *Image montages of some of the results obtained from registering kidney images from Chapter 5 using combined Normalized Gradient and Hessian Fields as distance measure.*

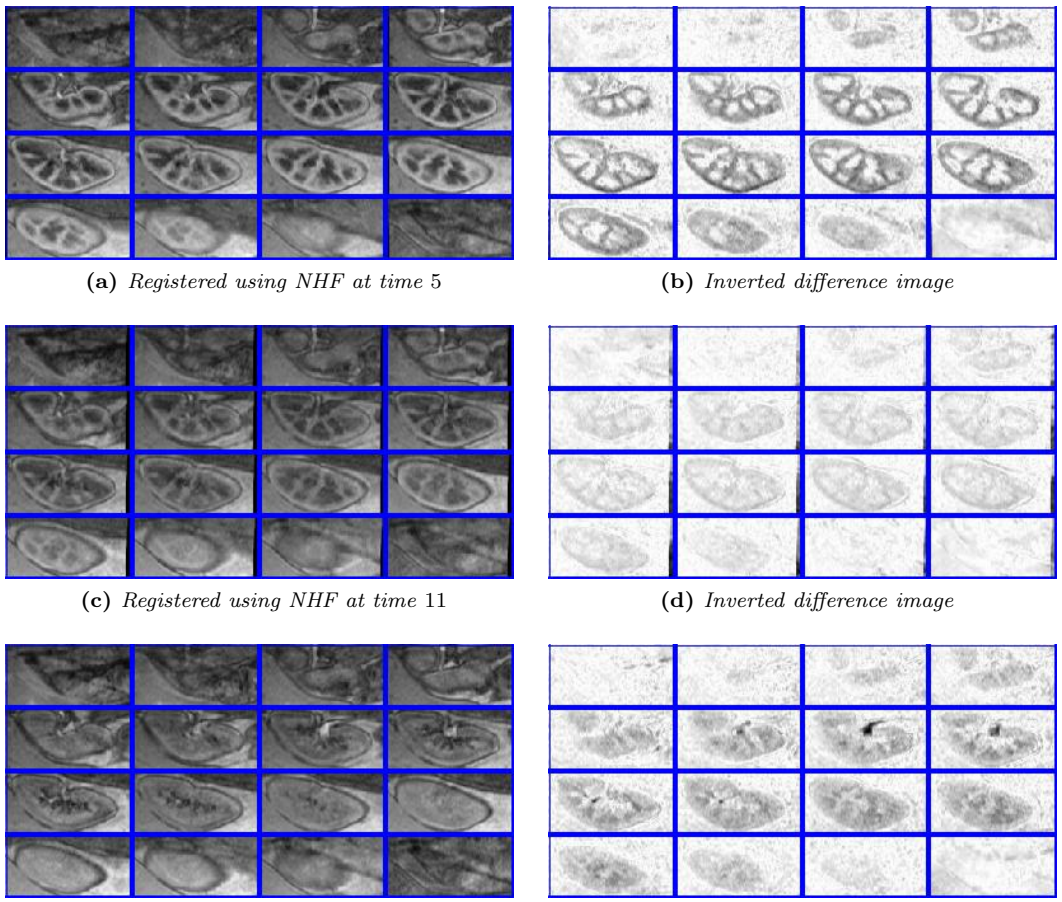


(a) *Registered using NGF at time* 5 **(b)** *Inverted difference image*

(c) *Registered using NGF at time* 11 **(d)** *Inverted difference image*

(e) *Registered using NGF at time* 19 **(f)** *Inverted difference image*

Figure 7.8: *Image montages of some of the results obtained from registering kidney images from Chapter 5 using combined Normalized Gradient and Hessian Fields as distance measure.*

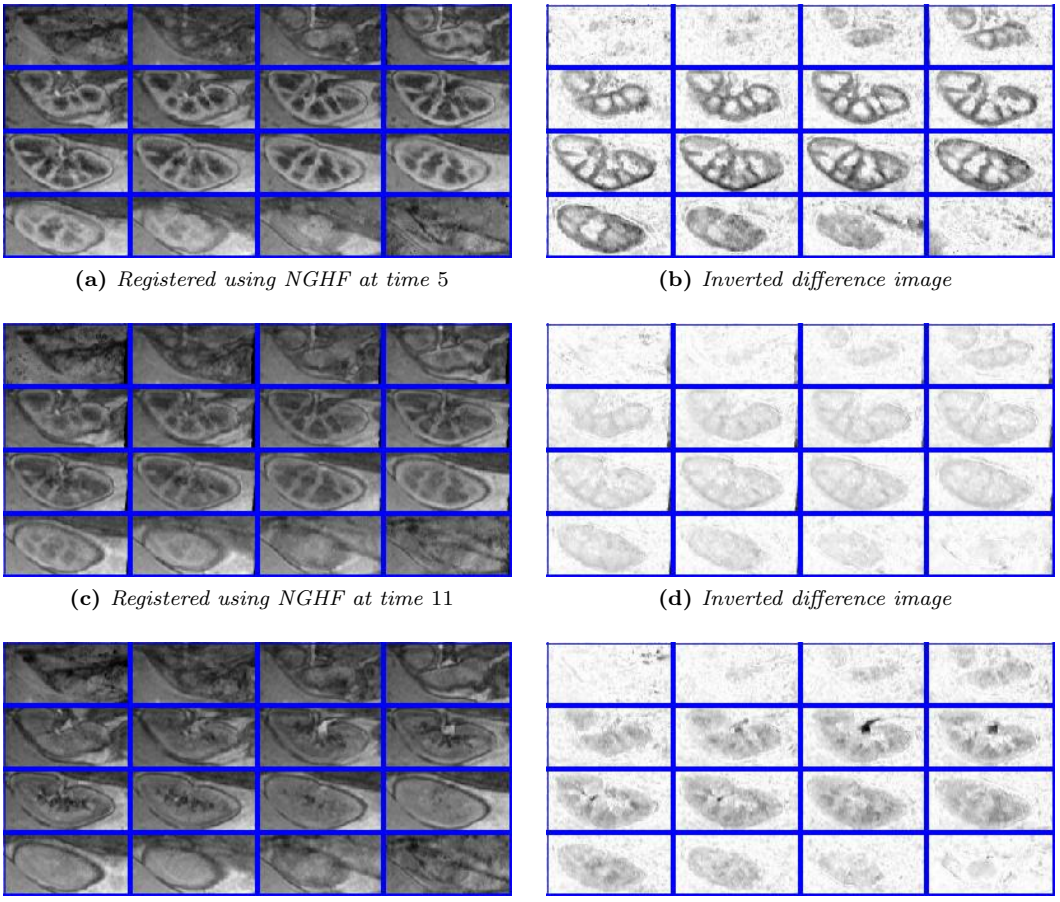


(a) *Registered using NHF at time* 5 **(b)** *Inverted difference image*

(c) *Registered using NHF at time* 11 **(d)** *Inverted difference image*

(e) *Registered using NHF at time* 19 **(f)** *Inverted difference image*

Figure 7.9: *Image montages of some of the results obtained from registering kidney images from Chapter 5 using combined Normalized Gradient and Hessian Fields as distance measure.*



(a) *Registered using NGHF at time* 5 **(b)** *Inverted difference image*

(c) *Registered using NGHF at time* 11 **(d)** *Inverted difference image*

(e) *Registered using NGHF at time* 19 **(f)** *Inverted difference image*

Figure 7.10: *Image montages of some of the results obtained from registering kidney images from Chapter 5 using combined Normalized Gradient and Hessian Fields as distance measure.*

7.2 Program code

Some of the more relevant pieces of MatLab code will be included in this section.

The fire script is for setting up the data used in the registration. To open, create and save nifti datasets I have used a script found at Matlab Central, [25]. The functions `getMultilevel`, `viewImage`,`inter`,`trafo`,`distance`,`regularizer`,`FAIRplots`,`viewIP`,`MLPIR` and `MLIR` are all part of the FAIR package.

This script is for creating multilevel datasets for FAIR from $3D$ NIfTI data.

```
% % Create multilevel datasets
% Set up dicretized grid size, m, and domain, omega
m = [128 128 32];    omega = [0 m(1) 0 m(2) 0 m(3)];

% Load datasets from nifti data
R = load_nii('T1_nii');    R = R.img;
T = load_nii('T2_nii');    T = T.img;

% Set up FAIR to use the correct viewer and interpolation
viewOptn = {'viewImage','imgmontage'};    viewImage('reset',viewOptn{:});
interOptn = {'inter','linearInter3D'};    inter('reset',interOptn{:});

% Get multilevel data based on images, domain and discretization
MLdata = getMultilevel({T,R},omega,m,'minLevel',2);
```

The main script used in most registrations. Sets up optional variables and decides which optimization scheme to use, as well as number of iterations to allow in the parametric preregistration and in the non-parametric part. Calls `register3d.m` to do the actual registration.

```
% % 3D registration of brain images
% Initialize some variables
edge = 0;    % Will be owerwritten if needed
alpha = 0;    % Will be owerwritten if needed

% Choose minimum level of registration, type of preregistration, optimization and regularization
type = 'affine';    % Preregistration
minLevel = 3;    % minimum level for registration
save_flag = 1;    % Flag for saving variables and images
lBFGS_flag = 1;    % Flag for using lBFGS instead of Gauss-Newton
regul = 'mfElastic';    % Regularization

% Open file
load MLdata

% Set up discretization and domain
m = MLdata{end}.m;    omega = MLdata{end}.omega;

% Choose distance measures to use
dist = {'SSD','NCC','MI','NGF','NHF','NGHF'};

for i=1:length(dist) % Run through all distance measures
```

```
% Determine regularization and other parameters based on dist
% measure
    switch dist{i}
        case 'SSD'
            alpha = 1e4;
        case 'NCC'
            alpha = 1e-9;
            minLevel = 4;
        case 'MI'
            alpha = 1e-7;
        case 'NGF'
            edge = 3;
            alpha = 1e2;
        case 'NHF'
            edge = 0.1;
            alpha = 1e1;
        case 'NGHF'
            edge = 0.1;
            alpha = 1e1;
    end

% Choose optimization scheme and collect variables
    if lBFGS_flag == 1
        varargin = {'dist',dist{i},...
            'maxIterPIR',100,'maxIterNPIR',100,...
            'edge',edge,'alpha',alpha,'minLevel',minLevel,...
            'nT',64,'nR',64,...
            'mfRegularizer',regul,...
            'NPIR',@lBFGS,'NPIRobj',@NPIRBFGSobjFctn}';
    else
        varargin = {'dist',dist{i},...
            'maxIterPIR',10,'maxIterNPIR',10,...
            'edge',edge,'alpha',alpha,'minLevel',minLevel,...
            'nT',64,'nR',64,...
            'mfRegularizer',regul}';
    end

% Register here
    if strcmp('affine',type)
        [yc,wc,his,xc,t] = register3d(MLdata,...
            varargin{1:end},'PIRtransf','affine3D');
    elseif strcmp('rigid',type)
        [yc,wc,his,xc,t] = register3d(MLdata,...
            varargin{1:end},'PIRtransf','rigid3D');
    end

% Save if needed
    if save_flag ==1
% Grid type depends on regularizer
        if strcmp(regul,'mfElastic')
            ycc = stg2center(yc,m);
        elseif strcmp(regul,'mfCurvature')
            ycc = yc;
        end
```

```
% Interpolate data on grid, reshape and save variables and image
        regT = inter(MLdata{end}.T,omega,ycc);
        regT = reshape(regT,m);
        regT = regT.*max(MLdata{end}.R(:))/max(regT(:));
        nifty = make_nii(regT);
        save_nii(nifty);
        save variables      end
end
```

Here is where the call to FAIRs registration is done. Contains preset variables that can be overwritten. Also sets up the choices for interpolation, parametric transformations, distance measure, regularization and viewer options for FAIR.

```
function [yc,wc,his,xc,t] = register3d(MLdata,varargin)
% Set up and do registration on 3D data using FAIR. Syntax:
% [yc,wc,his] = register3d(MLdata,omega,m,varargin)
% Input:
% MLdata - Multilevel data from getMultilevel.m
% varargin:
% dist - distance measure, 'SSD','NCC','MI' or 'NGF'
% nT,nR - parameters for MI
% edge - edge parameter for NGF
% alpha - parameter for regularization
% PIRtransf - type of PIR, 'rigid3d' or 'affine3d'
% maxIterPIR - max iterations PIR
% maxIterNPIR - max iterations NPIR
% minLevel,maxLevel - min and max levels for registration
% wc - starting guess, parameters
%
% Output
% yc - transformed grid
% wc - parameters for parametric transform
% his - iteration history

% Initialze default variables
dist = 'NGF';    para = 'NPIR';
nT = 32;    nR = 32;
edge = 10;    alpha = 1;
PIRtransf = 'affine3D';    mfRegularizer = 'mfElastic';
maxIterPIR = 50;    maxIterNPIR = 25;
minLevel = 3;    maxLevel = 4;
PIR = @GaussNewtonArmijo;    % optimizer to be used for PIR
PIRobj = @PIRobjFctn;    % objective function for PIR
NPIR = @GaussNewtonArmijo;    % optimizer to be used for NPIR
NPIRobj = @NPIRobjFctn;    % objective function for NPIR
yc = []; xc = [];    % Initialze grids

    % Overwrite default parameters if needed
for k=1:2:length(varargin),
    eval([varargin{k},'=varargin{',int2str(k+1),'};']);
end;

% Set up registration scheme:
```

```
% Viewer
    [viewer,viewOptn] = viewImage('reset','viewImage','imgmontage',...
        'colormap','gray(256)','direction','xyz');

% set-up interpolation scheme
    [int,intOptn] = inter('reset','inter','linearInter3D',...
        'regularizer','moments','theta',0);

% set-up transformation used in the parametric part
    [tra,traOptn] = trafo('reset','trafo',PIRtransf);

% set-up distance measure
    [dis,disOptn] = distance('reset','distance',dist,...
        'edge',edge,'nT',nT,'nR',nR);

    [reg,regOptn] = regularizer('reset','regularizer',...
        mfRegularizer,'alpha',alpha,'mu',1,'lambda',0 );

% prepare the plots
    FAIRplots('clear')
    showDifference = @(T,R,omega,m) viewIP(abs(R-T),omega,m,'colormap',gray(256));
    FAIRplots('set','showDifference',showDifference);

% Do the registration
% Check if the registration is parametric or non-parametric.
    if strcmp(para,'NPIR')
        [yc,wc,his,xc] = MLIR(MLdata,...
            'maxIterPIR',maxIterPIR,'maxIterNPIR',maxIterNPIR,...
            'minLevel',minLevel,'maxLevel',maxLevel,...
            'PIR',PIR,'PIRobj',PIRobj,'NPIR',NPIR,'NPIRobj',NPIRobj);
    elseif strcmp(para,'PIR')
        [wc,his] = MLPIR(MLdata,...
            'maxIter',maxIterPIR,...
            'minLevel',minLevel,'maxLevel',maxLevel,...
            'PIR',PIR);
    end
```

This is the implementation of Normalized Hessian Field

```
% % Normalized Hessian distance measure (Kronecker-based)
function [Dc,rc,dD,drc,d2psi] = NHkron(Tc,Rc,omega,m,varargin)
% Based on NGFdot.m from FAIR

persistent G1 G2 G3 % the gradient operators
% initialize output variables
Dc = []; dD = []; rc = []; drc = []; d2psi = [];

% Initialize local variables
h = (omega(2:2:end)-omega(1:2:end))./m; % voxel size for integration
hd = prod(h); n = prod(m);

dim = length(omega)/2;
```

```
edge = 100;    % the edge parameter
doDerivative = (nargout > 3);    % compute derivatives only on demand

% overwrites default parameters:
for k=1:2:length(varargin)      eval([varargin{k},'=varargin{',int2str(k+1),'};']);
end;

if size(G1,2)  = n,    % set up gradient if necessary
    for j=1:dim,    % compute discrete 1D derivatives
        d{j} = spdiags(ones(m(j),1)*[-1,1],[-1,1],m(j),m(j))/(2*h(j));
        d{j}([1,end]) = d{j}([2,end-1]);
    end;
    if dim == 2,
        G1 = sparse(kron(speye(m(2)),d{1}));
        G2 = sparse(kron(d{2},speye(m(1))));
        G3 = 0;
    else
        G1 = sparse(kron(kron(speye(m(3)),speye(m(2))),d{1}));
        G2 = sparse(kron(kron(speye(m(3)),d{2}),speye(m(1))));
        G3 = sparse(kron(kron(d{3},speye(m(2))),speye(m(1))));
    end;
end;

% Compute second derivatives
hessT = [G1*G1*Tc,G1*G2*Tc,G1*G3*Tc,G2*G1*Tc,G2*G2*Tc,...
    G2*G3*Tc,G3*G1*Tc,G3*G2*Tc,G3*G3*Tc];
hessR = [G1*G1*Rc,G1*G2*Rc,G1*G3*Rc,G2*G1*Rc,G2*G2*Rc,...
    G2*G3*Rc,G3*G1*Rc,G3*G2*Rc,G3*G3*Rc];

% Compute length of second order derivatives
lengthHT = sqrt(sum(hessT.^2,2) + edge^2);
lengthHR = sqrt(sum(hessR.^2,2) + edge^2);

% Compute correlation and distance
r1 = sum(hessR.*hessT,2);
r2 = 1./(lengthHT.*lengthHR);
rc = r1 .* r2;
Dc = prod([omega(2:2:end)-omega(1:2:end)]) - hd * (rc'*rc);

% Find derivative on demand
if  doDerivative, return; end;

sdiag = @(a) spdiags(reshape(a,[],1),0,length(a),length(a));

dr1 = sdiag(hessR(:,1))*G1*G1 + sdiag(hessR(:,2))*G2*G1 + sdiag(hessR(:,3))*G3*G1...
    + sdiag(hessR(:,4))*G1*G2 + sdiag(hessR(:,5))*G2*G2 + sdiag(hessR(:,6))*G3*G2...
        + sdiag(hessR(:,7))*G1*G3 + sdiag(hessR(:,8))*G2*G3 + sdiag(hessR(:,9))*G3*G3;
dr2 = -sdiag(1./(lengthHR.*lengthHT.^3)) * ...
    ( sdiag(hessT(:,1))*G1*G1 + sdiag(hessT(:,2))*G2*G1 + sdiag(hessT(:,3))*G3*G1...
        + sdiag(hessT(:,4))*G1*G2 + sdiag(hessT(:,5))*G2*G2 + sdiag(hessT(:,6))*G3*G2...
            + sdiag(hessT(:,7))*G1*G3 + sdiag(hessT(:,8))*G2*G3 + sdiag(hessT(:,9))*G3*G3);
drc = (sdiag(r2)*dr1 + sdiag(r1)*dr2);
dD = -2*hd*rc'*drc;
d2psi = 2*hd;
```

Bibliography

[1] M.A. Bernstein, K.F. King, and X.J. Zhou. *Handbook of MRI Pulse Sequences.* Elsevier Academic Press, 2004.

[2] C. Broit. *Optimal registration of deformed images.* PhD thesis, 1981.

[3] L.G. Brown. A survey of image registration techniques. 1992.

[4] G.E. Christensen. *Introduction to the Non-Rigid Image Registration Evaluation Project (NIREP).* 2006.

[5] A. Collignon. Automated multimodality medical image registration using information theory. 1995.

[6] B. Fischer and J. Modersitzky. A unified approach to fast image registration and a new curvature based registration technique. 2003.

[7] B. Fischer and J. Modersitzky. Ill-posed medicinean introduction to image registration. 2008.

[8] R. Fletcher. *Practical Methods of Optimization.* Wiley, 2nd edition, 1987.

[9] S. Frantz, K. Rohr, and H.S. Stiehl. Validating point-based mr/ct registration based on semi-automatic landmark extraction. 1999.

[10] R. Gonzalez and R. Woods. *Digital Image Processing.* Pearson Education, 3rd edition, 2008.

[11] E. Haber and J. Modersitzky. Numerical methods for volume preserving image registration. 2004.

[12] E. Haber and J. Modersitzky. Intensity gradient based registration and fusion of multi-modal images. 2006.

[13] J. Hadamard. *Sur les problèmes aux dérivées partielles et leur signification physique.* Princeton University Bulletin, 1902.

[14] E.L. Hahn. Spin echoes. 1950.

[15] G.T. Herman. *Fundamentals of Computerized Tomography.* Springer, 2nd edition, 2009.

[16] P.J. Hore. *Nuclear Magnetic Resonance.* Oxford Science Pubplications, 1st edition, 1995.

[17] F. Maes. Multimodality image registration by maximization of mutual information. 1997.

[18] J. Modersitzky. *FAIR, Flexible Algorithms for Image Registration.* SIAM, 2009.

[19] J. Nocedal and S.J. Wright. *Numerical Optimization.* Springer, 2nd edition, 2006.

[20] M.E. Phelps. *PET - Physics, Instruments, and Scanners.* Springer, 1st edition, 2006.

[21] J.P.W. Pluim. Mutual-information-based registration of medical images: A survey. 2003.

[22] W.K. Pratt. *Digitial Image Processing.* Wiley, 1st edition, 1978.

[23] I.I. Rabi. A new method of measuring nuclear magnetic moment. 1938.

[24] L. Romero and F. Calderon. A tutorial on parametric image registration. 2007.

[25] Jimmy Shen. Tools for nifti and analyze image. `http://www.mathworks.com/matlabcentral/fileexchange/8797-tools-for-nifti-and-analyze-image`, May 2011.

[26] B.W. Silverman. *Density estimation for statistics and data analysis.* Chapman and Hall, 1985.

[27] P.A. Tipler and R.A. Llewellyn. *Modern physics.* W.H. Freeman, 5th edition, 2008.

[28] P.A. Viola. *Alignment by maximization of mutual information.* PhD thesis, 1995.

[29] J. West. Comparison and evaluation of retrospective intermodality image registration techniques. 1997.

[30] B. Zitova and J. Flusser. Image registration methods: a survey. 2003.