Implementation of geometric distortion correction of EPI images in clinical workflow

Ingrid Framås Syversen

Master of Science in Physics and Mathematics
Submission date: June 2018
Supervisor: Pål Erik Goa, IFY
Co-supervisor: Mattijs Elschot, ISB-MH-NTNU

Norwegian University of Science and Technology
Department of Physics

Abstract

Background: Geometric distortion in echo-planar (EPI) magnetic resonance (MR) images, caused by low bandwidth combined with variations in magnetic susceptibility, is a problem that can make it challenging to connect functional MR data to anatomical position. A distortion correction method based on the acquisition of images with opposing phase encoding directions has been successfully used at NTNU for several years. However, this method can currently only be performed offline on anonymised images and is therefore not used in clinical routine. The aim of this master's project was to implement and test this method in the clinical software environment of syngo.via Frontier.

Materials and methods: A prototype for distortion correction was made in MeVisLab and installed in syngo.via Frontier. It was tested on diffusion-weighted EPI images from 13 breast and 16 prostate patients, and the images were examined visually to determine the robustness in terms of percentage non-failed corrections, and the quality of corrections. Various image similarity metrics were also calculated in order to try to determine the quality of corrections quantitatively. To assess the user-friendliness of the prototype, it was tested by a radiographer and a medical physicist.

Results: None of the corrections failed, giving a robustness of 100%. By visual assessment, the quality of correction was determined to be successful for 12 of 13 breast patients and all 16 prostate patients. The calculated metrics generally showed good results for the breast images, but they gave some inconclusive results for the prostate images. The feedback from the user-friendliness testing showed that after minimal training it would be relatively easy to use the prototype.

Conclusion: The prototype was able to successfully correct geometrically distorted images. Nevertheless, the calculated metrics yielded varying results, and more investigations should therefore be performed to find suitable metrics for determining the quality of corrections quantitatively. Although there is still room for improvements, the prototype has potential to be an easy-to-use tool for geometric distortion correction of EPI images in clinical workflow.

Sammendrag

Bakgrunn: Geometrisk distorsjon i ekko-planare (EPI) magnetresonansbilder (MR-bilder), forårsaket av lav båndbredde i kombinasjon med varierende magnetisk susceptibilitet, er et problem som kan gjøre det utfordrende å koble funksjonelle MR-data til anatomisk posisjon. En distorsjonskorreksjonsmetode basert på opptak av bilder med motsatt fasekodingsretning har vært vellykket brukt på NTNU i flere år. Imidlertid kan denne metoden kun bli utført offline på anonymiserte bilder, og er derfor ikke brukt i klinisk rutine. Målet med dette masterprosjektet var å implementere og teste denne metoden i den kliniske programvaren syngo.via Frontier.
Materialer og metode: En prototype for distorsjonskorreksjon ble laget i MeVisLab og installert i syngo.via Frontier. Den ble testet på diffusjonsvektede EPI-bilder fra 13 bryst- og 16 prostatapasienter, og bildene ble undersøkt visuelt for å bestemme robustheten, definert som prosent ikke-feilede korreksjoner, samt korreksjonskvaliteten. Ulike metrikker for bildelikhet ble også beregnet for å prøve å bestemme korreksjonskvaliteten kvantitativt. For å vurdere prototypens brukervennlighet, ble den testet av en radiograf og en medisinsk fysiker.
Resultater: Ingen av korreksjonene feilet, noe som gir en robusthet på 100%. Ved visuell undersøkelse ble korreksjonskvaliteten vurdert til å være vellykket for 12 av 13 brystpasienter og alle 16 prostatapasienter. De beregnede metrikkene viste generelt gode resultater for brystbildene, men ga noen inkonklusive resultater for prostatabildene. Tilbakemeldingene fra brukervennlighetstestingen viste at etter minimalt med opplæring vil prototypen være relativt enkel å bruke.
Konklusjon: Prototypen klarte vellykket å korrigere geometrisk deformerte bilder. Likevel ga de beregnede metrikkene varierende resultater, og videre undersøkelser bør derfor utføres for å finne passende metrikker for å bestemme korreksjonskvaliteten kvantitativt. Selv om det fortsatt er rom for forbedringer, har prototypen potensial til å være et lettbrukelig verktøy for geometrisk distorsjonskorreksjon av EPI-bilder i klinisk arbeidsflyt.

Preface

This thesis is the final part of my master's degree in Applied Physics and Mathematics at the Norwegian University of Science and Technology (NTNU), where I have chosen to specialise in Biophysics and medical technology. The work was performed throughout the spring semester of 2018.

I would like to express my gratitude to my supervisors Pål Erik Goa and Mattijs Elschot, for giving me the opportunity to work with this interesting project, and for their invaluable help, guidance and motivation throughout the whole process. I also want to thank Anders Rodell, Wolfgang Aichinger and Stefan Huwer from Siemens for their help with questions related to MeVisLab and syngo.via Frontier. In addition, I would like to thank those who took their time to test my application.

At last, I want to thank my family for their support, and my friends for making these five years in Trondheim so memorable.

Trondheim, June 2018

Ingrid Framås Syversen

Contents

Abbreviations

ADC	Apparent diffusion coefficient
CMTK	Computational Morphometry Toolkit
DWI	Diffusion-weighted imaging
EEW	ExternalExecutableWrapper
EPI	Echo-planar imaging
ESP	Echo spacing
ETL	Echo train length
FID	Free induction decay
fMRI	Functional MRI
FOV	Field of view
FT	Fourier transform
LUT	Look-up table
MRI	Magnetic resonance imaging
MSD	Mean squared difference
NCC	Normalised cross-correlation
NMI	Normalised mutual information
PLACE	Phase labelling for additional coordinate encoding
PSF	Point spread function

RF	Radiofrequency
SE	Spin echo
SNR	Signal-to-noise ratio
TE	Echo time
TR	Repetition time
TRAIL	Two reduced acquisitions interleaved
UID	Unique identifier

Chapter 1

Introduction

In the later years, there have been huge technological developments in the field of magnetic resonance imaging (MRI), leading to new functional MRI (fMRI) techniques [1]. The difference from anatomical MRI is that information about the tissue microenvironment can be assessed, for example the diffusion or perfusion of the tissue. fMRI is usually dependent on a good temporal resolution, and the acquisition method must therefore be fast. Echo-planar imaging (EPI) is well-suited for this.

In EPI, images can be acquired in a very short time because of the rapid switching of magnetic gradients [2]. However, susceptibility-induced geometric distortion of the images is a major problem. It appears as local deformations, and heavily distorted images may consequently hinder correct diagnosis and analysis. Even though a trained radiologist is able to see beyond the distortion when looking at the images qualitatively, quantitative analysis becomes more difficult. The functional images must be co-registered—that is, aligned—to anatomically correct images in order to connect functional data to anatomical position. Geometric distortion makes this challenging, and in some cases impossible. The distortion must therefore be corrected. This correction will also become even more important in the future, when machine learning will be increasingly used for quantification of various parameters from functional images.

There exist several methods for distortion correction, where the method developed by Holland et al. has been successfully used at NTNU for several years [3, 4, 5]. Unfortunately, this method can currently only be performed offline on anonymised images. The distortion correction is therefore not yet implemented in clinical routine.

Recently, St. Olavs Hospital and NTNU have acquired a new image processing software from Siemens named syngo.via Frontier Development Kit. In addition to the available prototypes in the Frontier Store, this opens up for the development of own image processing prototypes. Since syngo.via Frontier is connected to the hospital image storage and viewing system (PACS), the method for EPI distortion correction can be implemented as a prototype that can be used in the clinical workflow.

The aim of this master's thesis is to get started on facilitating the use of distortion corrected EPI images in clinical routine. The main objectives are:

1. To implement the EPI distortion correction method in syngo.via Frontier Development Kit.
2. To evaluate the results from using the processing tool, by means of:
    - User-friendliness of application.
    - Robustness in terms of percentage non-failed corrections.
    - Quality of corrections, tested on breast and prostate diffusion-weighted EPI images.

The thesis will begin with a description of relevant theory, followed by the materials and methods used. The results will then be presented, and they will further be discussed and related to other research in the field. Suggestions for future work will also be presented. At last, a conclusion of the project is given. Relevant scripts are included in the appendix, in addition to the feedback form used for the evaluation of prototype user-friendliness.

Chapter 2

Theory

2.1 MRI in general

Sections 2.1.1 and 2.1.2 are adapted from a previously written project thesis by the author [6].

2.1.1 Basic principles of MRI

Magnetic resonance imaging (MRI) is a non-invasive medical imaging modality that takes advantage of the magnetic properties of protons [7]. The theory behind MRI is quite comprehensive, and a detailed description is beyond the scope of this thesis. Only an introduction to the principles of MRI relevant for this master's thesis will be presented in the following. A more thorough description can be found in various textbooks [8, 9].

The proton is a positively charged particle that exists in large amounts in biological material, and it has a quantum mechanical property called spin [10]. It also has a magnetic dipole moment $\vec{\mu}$. If the proton spins are put into an external magnetic field $\vec{B}_0 = B_0\hat{z}$, there will be an energy splitting. The spins with a positive z-component are called "spin up" and are in the low-energy state, and those with a negative z-component are called "spin down" and are in the high-energy state. In addition, $\vec{B}_0$ will exert a torque on $\vec{\mu}$, which leads to a movement of $\vec{\mu}$ given by

$$\frac{d\vec{\mu}}{dt} = \gamma\vec{\mu} \times \vec{B}. \tag{2.1}$$

This is called the Bloch equation, and the solution to this is precession of $\vec{\mu}$ [8]. Thus, it will

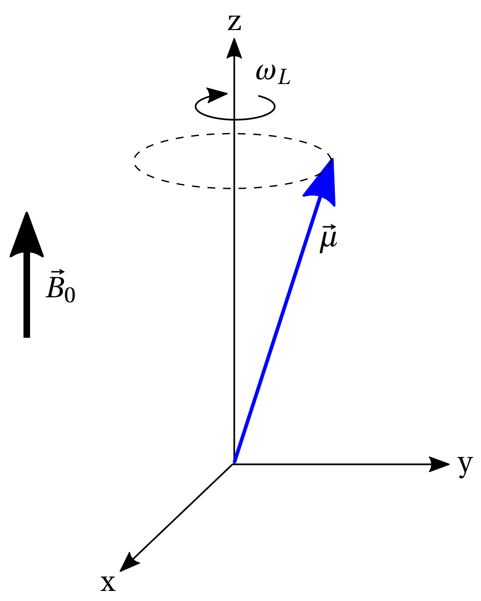


Figure 2.1: In an external magnetic field $\vec{B}_0$ along the z-direction, the magnetic moment $\vec{\mu}$ will precess around the z-axis with the Larmour frequency ω_L.

precess around the direction of the magnetic field with the Larmour frequency

$$\omega_L = \gamma B_0, \tag{2.2}$$

where γ is the gyromagnetic ratio (Figure 2.1). In equilibrium, there will be more spins with spin up than spin down, and there will therefore be a net magnetisation $\vec{M}$ in the direction of the external field. The motion of the net magnetisation can be also described by the Bloch equation,

$$\frac{d\vec{M}}{dt} = \gamma \vec{M} \times \vec{B}. \tag{2.3}$$

To be able to detect a signal of the magnetisation, the spins must be excited [7]. In this context, excitation means that the net magnetisation is tipped away from the z-direction and down to the xy-plane. If a second external magnetic field $\vec{B}_1$ that rotates with the Larmour frequency of the spins is applied, resonance occurs, and the net magnetisation will also rotate around the direction of $\vec{B}_1$. This can easier be visualised using a frame of reference rotating at the Larmour frequency, see Figure 2.2. In this frame of reference, it appears as if the spins are only affected by the $\vec{B}_1$ field. $\vec{B}_1$ has a rotation frequency in the radiofrequency (RF) range, and its application is therefore called an RF pulse. By applying $\vec{B}_1$ for a certain

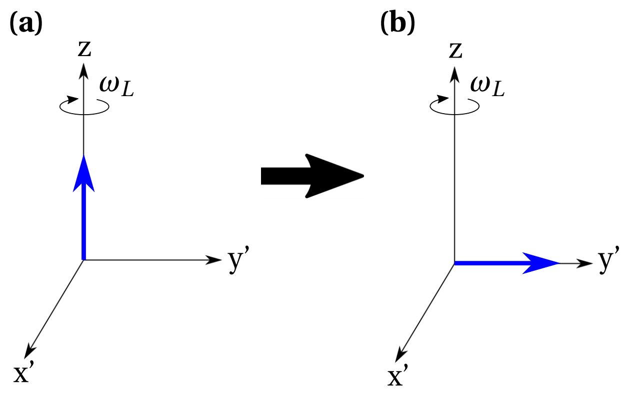


Figure 2.2: Excitation of the spins, shown in a frame of reference rotating at the Larmour frequency ω_L. **(a)** The net magnetisation vector before excitation. **(b)** The net magnetisation vector immediately after excitation. $\vec{B}_1$ has rotated it 90° around the x'-axis.

amount of time t_{rf}, $\vec{M}$ will rotate an angle α, given by

$$\alpha = \gamma B_1 t_{rf}. \tag{2.4}$$

B_1 and t_{rf} can be adapted to rotate $\vec{M}$ an angle of for example 90° or 180°, and the application of $\vec{B}_1$ is then called a 90° or 180° pulse, respectively.

After the spins have been excited, the net magnetisation will start to precess in the xy-plane. However, interactions with the external field $\vec{B}_0$ will restore the magnetisation back to its equilibrium state in a process called relaxation [8]. There are two kinds of relaxation processes; longitudinal and transverse. Longitudinal relaxation is the regrowth of net magnetisation in the z-direction, M_z. This is also called spin-lattice relaxation, because when the individual spins are relaxed they release energy to their environment ("lattice") in the form of heat [1]. The regrowth of M_z is exponential and is given by

$$M_z(t) = M_z(t = 0^+)e^{-R_1 t} + M_0\left(1 - e^{-R_1 t}\right), \tag{2.5}$$

where $t = 0^+$ is immediately after excitation, M_0 is the initial longitudinal magnetisation before excitation, and R_1 is the longitudinal relaxation rate [8]. Its inverse is called the longitudinal relaxation time, T_1.

Transverse relaxation is a dephasing process, also known as spin-spin relaxation [8]. Shortly after the excitation, all spins will precess with the same phase. However, due to the interaction of magnetic fields from adjacent spins, not all spins will rotate with the exact same

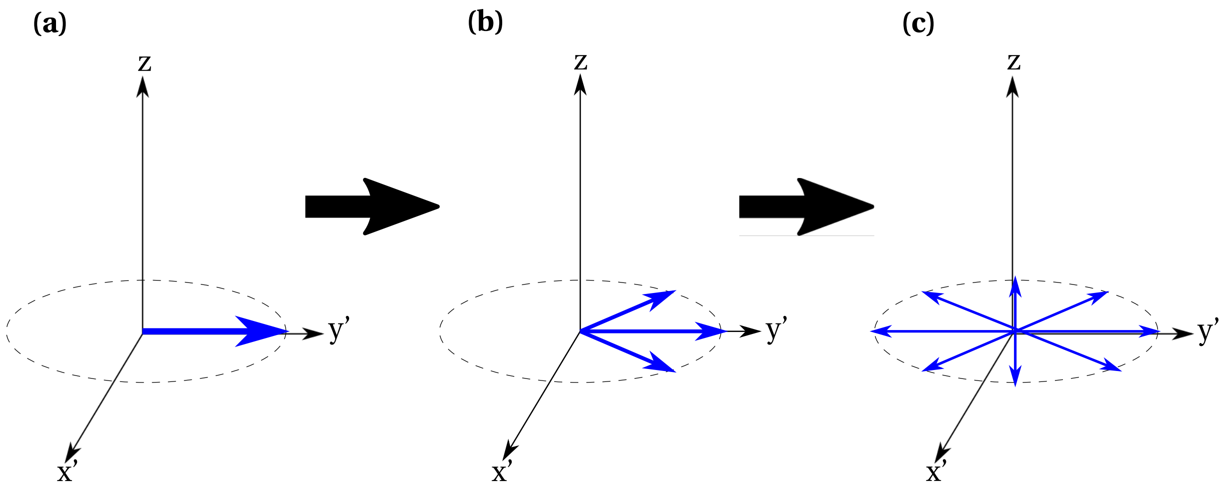


Figure 2.3: Illustration of transverse dephasing of spins, shown in a frame of reference rotating with the Larmour frequency. **(a)** Immediately after excitation, all spins are in the same phase. **(b)** Because the spins are rotating at slightly different frequencies due to spin-spin interactions and external field inhomogeneities, the spins will start to dephase. **(c)** After a while, the spins will be completely out of phase.

frequency. This leads to a dephasing of the spins, and after a while they will completely cancel each other out (Figure 2.3). This is an exponential decay with transverse relaxation rate R_2, and its inverse is the transverse relaxation time T_2. However, in addition to this dynamic spin-spin dephasing, there is an additional static dephasing effect from inhomogeneities in the external magnetic field $\vec{B}_0$ [8]. This is because different materials get magnetised to a different extent by an external magnetic field—that is, they have different magnetic susceptibility [11]. The rate of the resulting total dephasing is called R_2*, with its inverse denoted as T_2*. Its relation with R_2 is given by

$$R_2{}^* = R_2 + R_2', \tag{2.6}$$

where R_2' describes the extra dephasing added by the external field inhomogeneities. The resulting exponential decay of transverse magnetisation is given by

$$M_{xy}(t) = M_{xy}(t = 0^+)e^{-R_2{}^*t}. \tag{2.7}$$

The MR signal is recorded using receiver coils. When the spins have been excited, there will be a rotating net magnetisation in the transverse plane. According to the laws of electromagnetism, this varying magnetic field will induce a current in the receiver coils of the MRI scanner [8]. This current is the MR signal. The signal that is recorded after an excitation is called the free induction decay (FID) signal (Figure 2.4) [1].

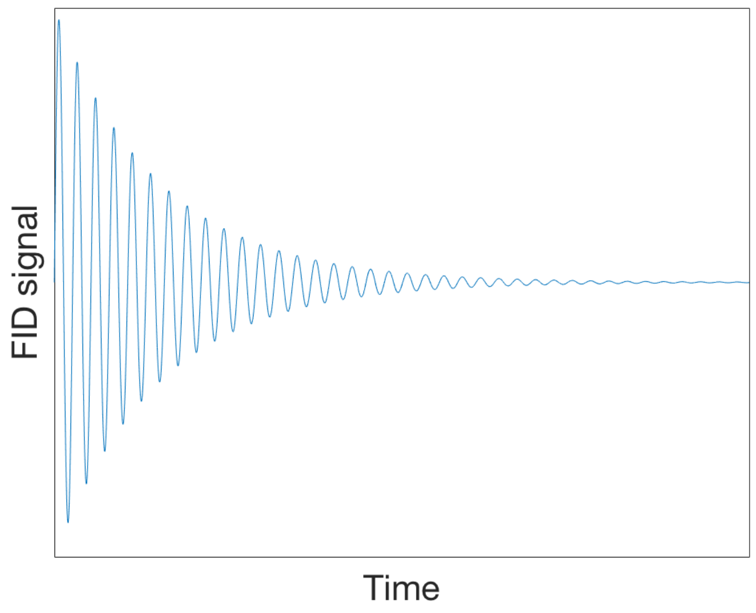


Figure 2.4: The FID signal oscillates with the Larmour frequency because the rotating $\vec{M}_{xy}$ induces a varying current in the receiver coils, while being damped exponentially by T_2* dephasing.

The fact that the Larmour frequency varies with field strength can be used to locate the spins [8]. By applying an external magnetic field that varies linearly with z-position, the Larmour frequency will also vary linearly. A spatially varying magnetic field is called a magnetic field gradient. The gradients are what enables spatial encoding in MRI. By applying gradients in different directions, the position of the spins along those directions can be determined. However, what is actually recorded in MRI, is the Fourier transform (FT) of the object. Hence, the k-space of the object is sampled. The application of gradients determine which positions in k-space are being sampled. An illustrated example of this is shown in Section 2.2.1. By inverse Fourier transforming the acquired k-space, an image of the object can be obtained, as shown in Figure 2.5.

To acquire an image, an MR sequence has to be performed [8]. This is a combination of RF pulses and gradients which makes it possible to record a signal that contains the desired information. Different combinations create different sequences, and this can be used to extract a great variety of information from MR images [1]. In the following, MR sequences

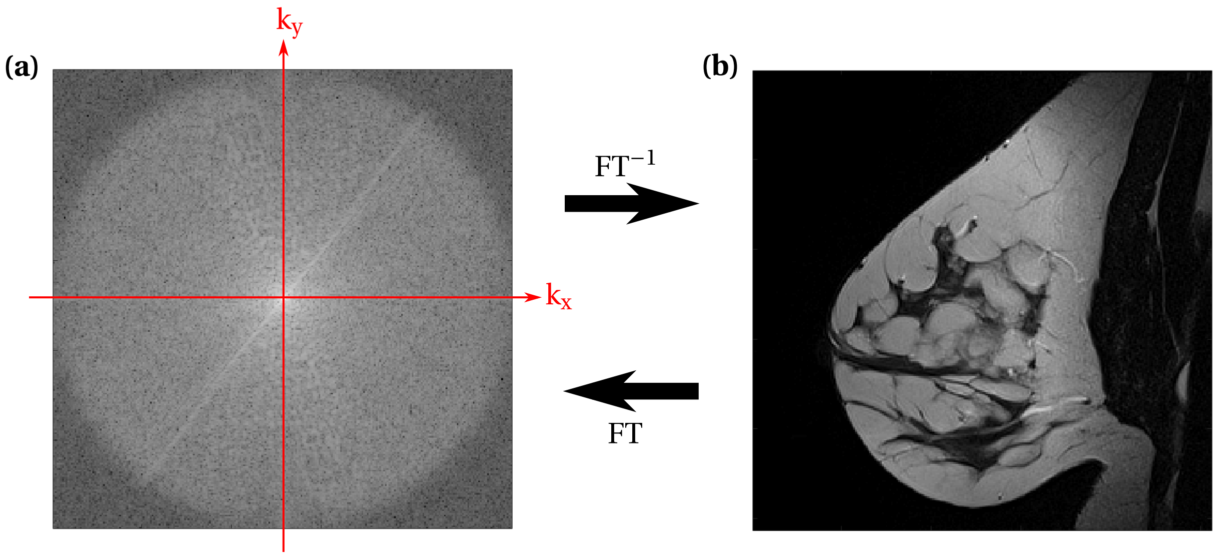


Figure 2.5: The relationship between **(a)** k-space and **(b)** a T_2-weighted breast MR image. The MR image can be found by taking the inverse Fourier transform (FT^{-1}) of k-space, while the k-space can accordingly be derived by taking the Fourier transform of the image.

relevant for this thesis will be introduced.

2.1.2 Spin echo sequence

After the spins have been excited, they will start to dephase, in a combination of static and dynamic dephasing. However, it is possible to reverse the static dephasing by applying a 180° pulse (Figure 2.6). The rephasing following the 180° pulse is called a spin echo, and the time at which this rephasing occurs is called the echo time (TE) [9]. This principle is used to create a spin echo sequence. It is a simple, but fundamental sequence in MRI, which other more complex sequences can be based on [2].

A spin echo (SE) sequence starts with a 90° pulse to excite the spins (Figure 2.7) [8, 9]. However, it is desirable to only excite one slice at a time. Therefore, a slice selection gradient (G_z) is applied so that the Larmour frequency varies with z-position. Only spins with Larmour frequency close to the frequency of the RF pulse will be excited. Then, a short phase encoding gradient blip (G_y) is applied to move to the desired y-position in k-space, and a frequency encoding gradient (G_x) to move to the desired x-position in k-space. The spins will completely dephase, and a 180° pulse is applied to rephase them again. G_x, also called the read-out gradient, is applied again and a spin echo occurs at t = TE. It is called the read-out gradient because signal acquisition is performed while it is on. One line in k-space is sampled during

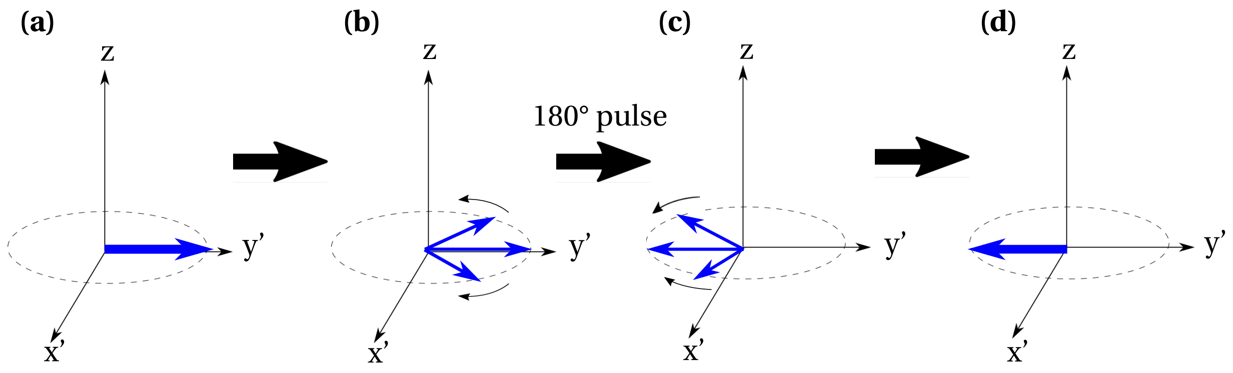


Figure 2.6: Illustration of spin echo, shown in a frame of reference rotating with the Larmour frequency. **(a)** Immediately after excitation ($t = 0^+$), all spins are in the same phase. **(b)** The spins will start to dephase because they precess at slightly different frequencies. **(c)** By applying a 180° pulse, the spins are rotated through 180°, and will start to rephase again. **(d)** At t = TE, a spin echo occurs and all the spins are in the same phase again.

this application of G_x. To sample more lines, the whole sequence is repeated several times. It is also possible to apply more than one 180° pulse after the same 90° pulse to achieve several spin echoes. The time between successive 90° pulses is called the repetition time (TR). Note that this approach where the slice selection gradient is used is called two-dimensional (2D) imaging. Three-dimensional (3D) imaging is obtained by omitting the slice select gradient and blipping G_z in the same way as G_y, so that phase encoding is achieved in both y- and z-direction [8].

2.1.3 Diffusion-weighted MRI

Diffusion is the random movement of molecules in a medium [2]. In the body, diffusion of water molecules can be restricted in one or more directions, depending of the structure of the tissue [1]. Diffusion-weighted imaging (DWI) can therefore give information about tissue structure and function. It has a number of uses, and can for example be used to characterise breast and prostate tumours [12, 13].

One way to obtain diffusion-weighting is by adding a pair of diffusion-sensitising gradients ($G_{\text{diffusion}}$) to a spin echo sequence (or to the SE-EPI sequence described in Section 2.2.1), see Figure 2.8 [2]. The first diffusion gradient, which is placed before the 180° pulse, will produce a position-dependent phase shift of the spins. The second diffusion gradient, which is placed after the 180° pulse, will produce a corresponding position-dependent negative phase shift, which reverses the phase shift produced by the first gradient. This means that for stationary spins, there will be no net phase shift after the two diffusion gradients. However, during

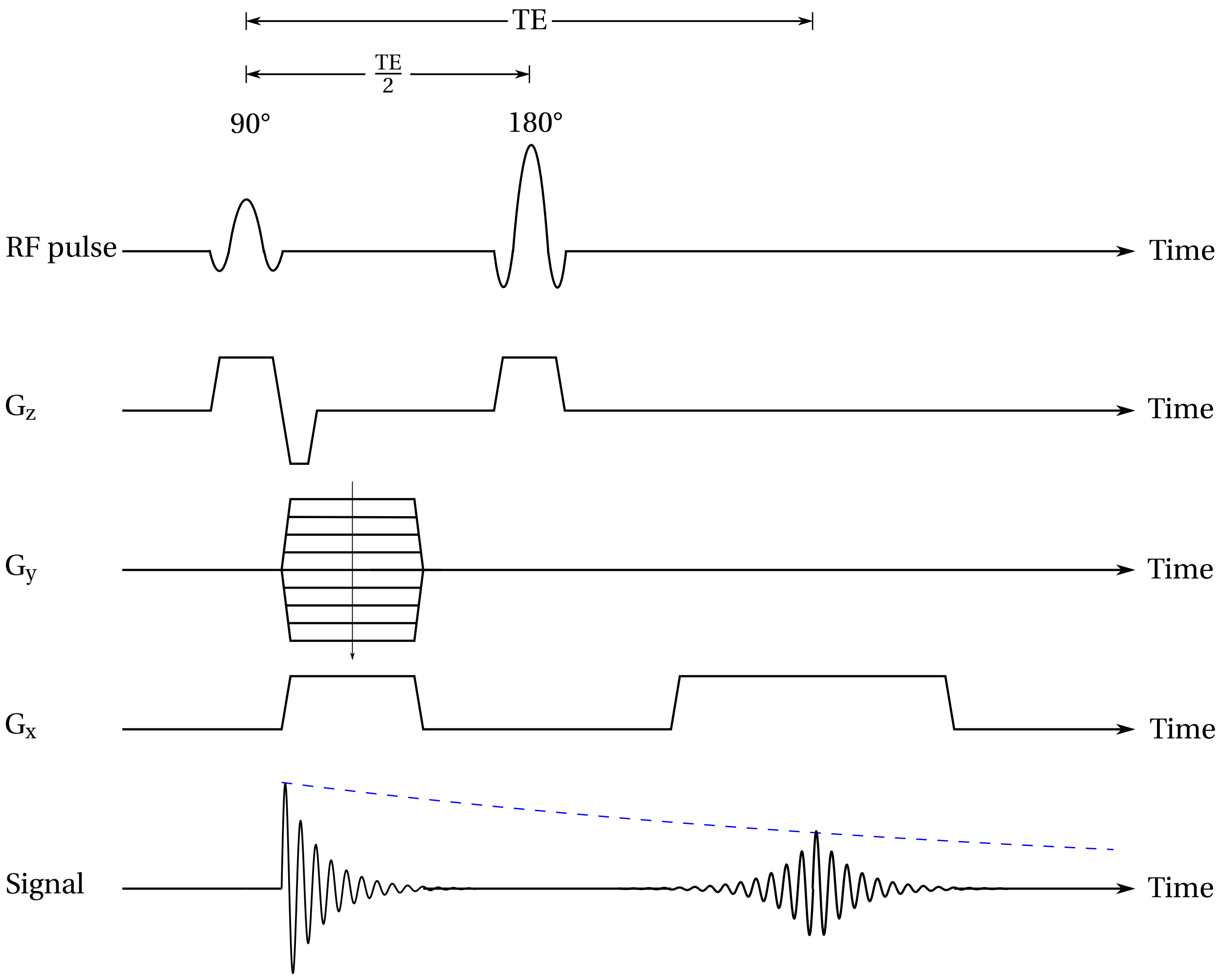


Figure 2.7: Schematic illustration of a spin echo pulse sequence. A 90° pulse is applied simultaneously as the slice select gradient (G_z), followed by a phase encoding blip (G_y) and a preparatory frequency encoding gradient (G_x) to move to the desired position in k-space. The spins are dephased, and then a 180° pulse is applied at $t = \frac{TE}{2}$. G_x is applied again, and a spin echo occurs at $t = TE$. The blue dashed curve indicates the dynamic (T_2) dephasing.

the time between the two gradients, some of the spins will be displaced, depending on the diffusion of the tissue. Thus, some of the spins will not experience equal gradient strength from both diffusion gradients, and they will therefore have residual phase after the application of the second gradient. The refocusing of the signal will consequently not be perfect. This means that areas with higher diffusion will give lower signal in the resulting diffusion-weighted image, because spins that are out of phase will cancel each other out.

The resulting signal $S(b)$ is given by

$$S(b) = S_0 e^{-bD}, \tag{2.8}$$

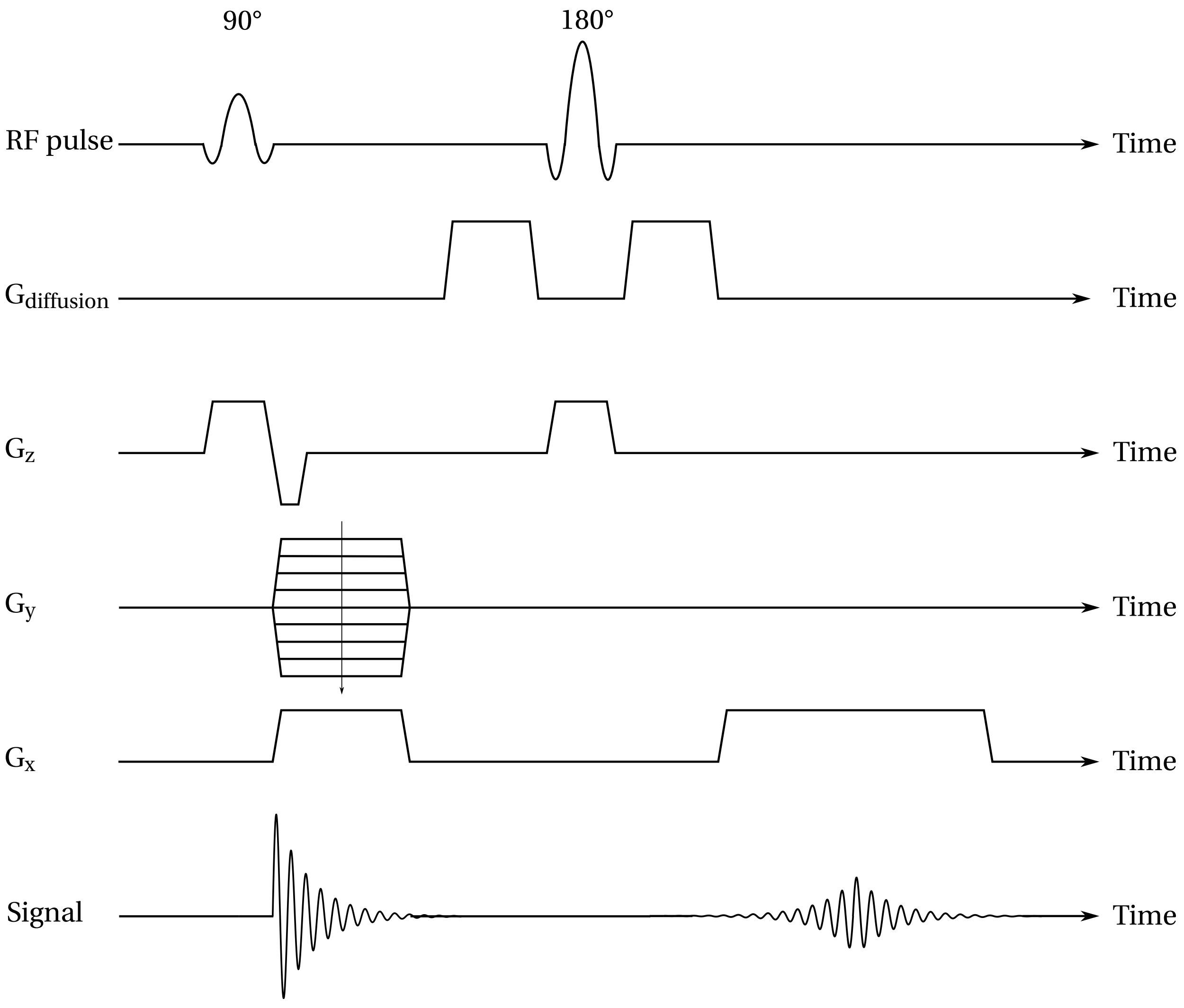


Figure 2.8: Schematic illustration of a diffusion-weighted pulse sequence. A 90° pulse is applied simultaneously as the slice select gradient (G_z), followed by a phase encoding blip (G_y) and a preparatory frequency encoding gradient (G_x) to move to the desired position in k-space. The spins are dephased, and the first diffusion gradient ($G_{diffusion}$) is applied. A 180° pulse is applied, before the second diffusion gradient. G_x is then applied again and a spin echo occurs, where only the spins that have not diffused between the diffusion gradients will be completely rephased.

where S_0 is the signal in the absence of diffusion, b is the b-value and D is the diffusion coefficient of the spin [2]. The b-value controls the degree of diffusion weighting, and depends on the diffusion gradient strength, for how long the gradients are turned on, and the time between the diffusion gradients. Often, both images with $b = 0$ (denoted b_0) and images with $b > 0$ are acquired, which allows for calculation of the diffusion coefficient by performing a linear regression of Equation (2.8). Note that D is the diffusion coefficient along the direction the diffusion gradients are applied. Since D can vary between directions, diffusion-weighting is usually performed for several directions. However, when measuring the diffusion using DWI, there are many spins in each pixel. Therefore, it is instead the apparent diffusion coefficient (ADC) that is observed, which is the mean diffusion in a pixel.

2.2 Echo-planar imaging

In echo-planar imaging (EPI), images can be acquired in a very short time. This makes it well-suited for acquiring functional MR images, where a high temporal resolution is needed to follow the temporal changes of the spins [1]. It is able to create images at such a speed because a whole 2D image can be acquired from a single excitation. However, this imposes strict requirements to the hardware of the MRI scanner. The technique was conceived in 1977 by Peter Mansfield, but because of practical difficulties, images of acceptable quality were not produced until the second half of the 1980s [14, 15]. Since then, there have been great improvements in hardware, and EPI has become a widely used imaging sequence and revolutionised the field of fMRI.

2.2.1 EPI sequence

EPI can be performed in several ways, but here a common spin echo (SE)-EPI sequence will be described (Figure 2.9). This sequence begins in the same way as the basic SE sequence, with a 90° excitation pulse and a slice select gradient (G_z) [2, 15]. The preparatory phase encoding (G_y) and frequency encoding (G_x) gradients are then applied, followed by a 180° pulse to rephase the spins. The read-out gradient G_x is applied again, and this produces a gradient-driven rephasing and dephasing of the spins, also called a gradient echo. Then, instead of waiting for the next excitation pulse, the phase encoding gradient is blipped, and the read-out gradient is applied again in the opposite direction as the previous application. A new gradient echo is produced, and this procedure is repeated many times. A new line in k-space is acquired for each gradient echo. The reversals of the read-out gradient and the blipping of the phase encoding gradient leads to a raster-like sampling of k-space, as shown in Figure 2.10.

The gradient echoes are formed under the envelope of the spin echo, where the peak amplitude of the signal occurs at the echo time of the spin echo, TE_{se} [2]. The time at which the central line in k-space is acquired is called the effective echo time, TE_{eff}. TE_{se} and TE_{eff} may or may not occur at the same time, but if they do, the sensitivity to off-resonance effects

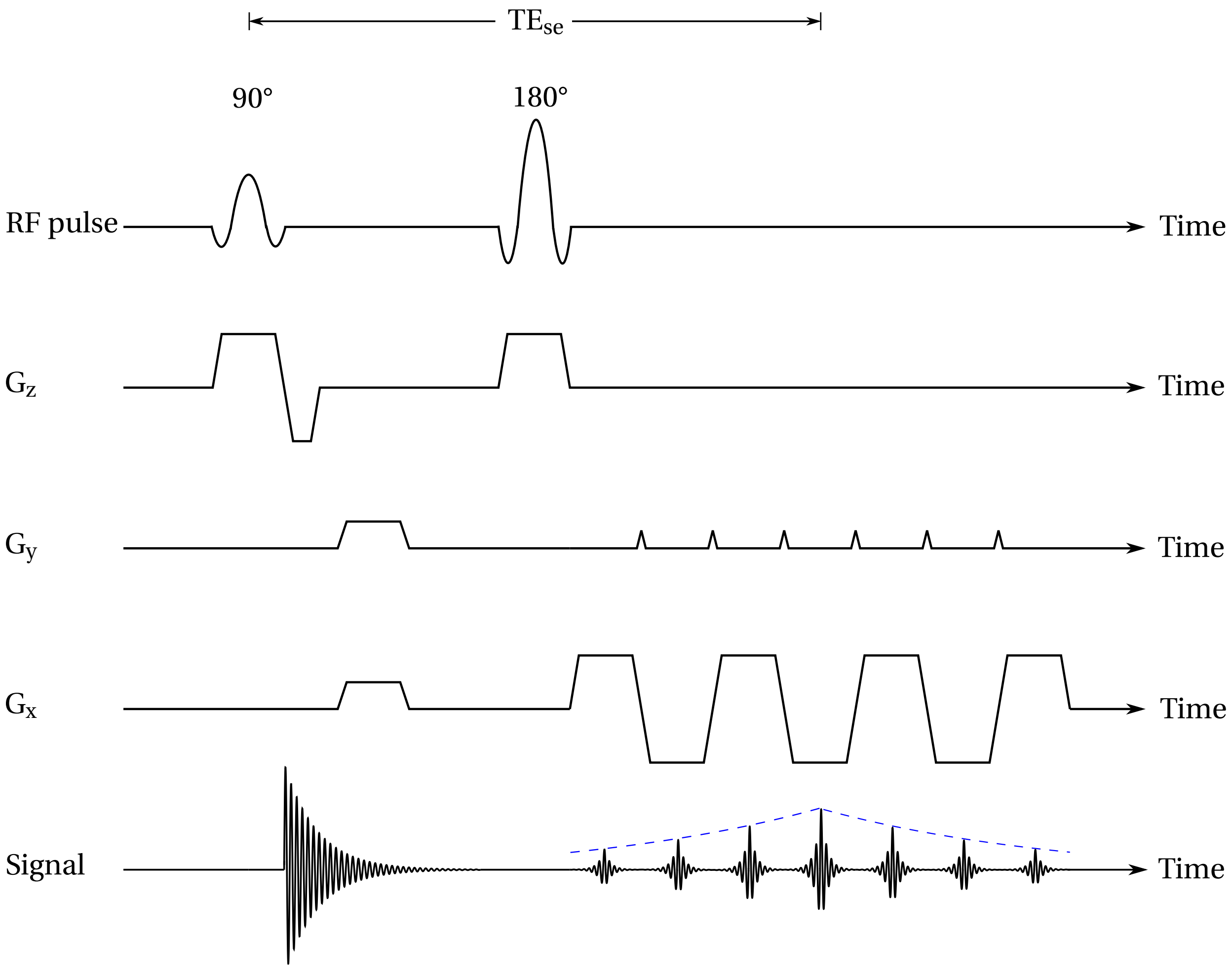


Figure 2.9: Schematic illustration of an SE-EPI sequence. A 90° pulse is applied simultaneously as the slice select gradient (G_z), followed by preparatory phase encoding (G_y) and frequency encoding gradients (G_x) to move to the desired position in k-space. The spins are dephased, and a 180° pulse is applied. Then, the read-out gradient is oscillated to produce several gradient echoes, with a phase encoding blip at each gradient reversal. The gradient echoes are formed under the envelope of the spin echo, which is indicated by the blue dashed curve. The peak amplitude of the signal occurs at $t = \mathrm{TE}_{se}$.

caused by magnetic field inhomogeneities is substantially reduced.

The number of k-space lines sampled in one excitation is known as the echo train length (ETL) of the sequence [2]. The time between two adjacent echoes is the echo spacing (ESP). The ETL is limited by decay of the transverse magnetisation, and also by the ESP. It is possible to sample all the k-space lines in a single excitation, so that a whole 2D slice is acquired at once, and this is called single-shot imaging. This is, of course, the fastest way to acquire images, but the image quality might be compromised. Multi-shot imaging, where k-space is sampled over several excitations, may improve image quality, but is more sensitive to motion artefacts because of the longer imaging time.

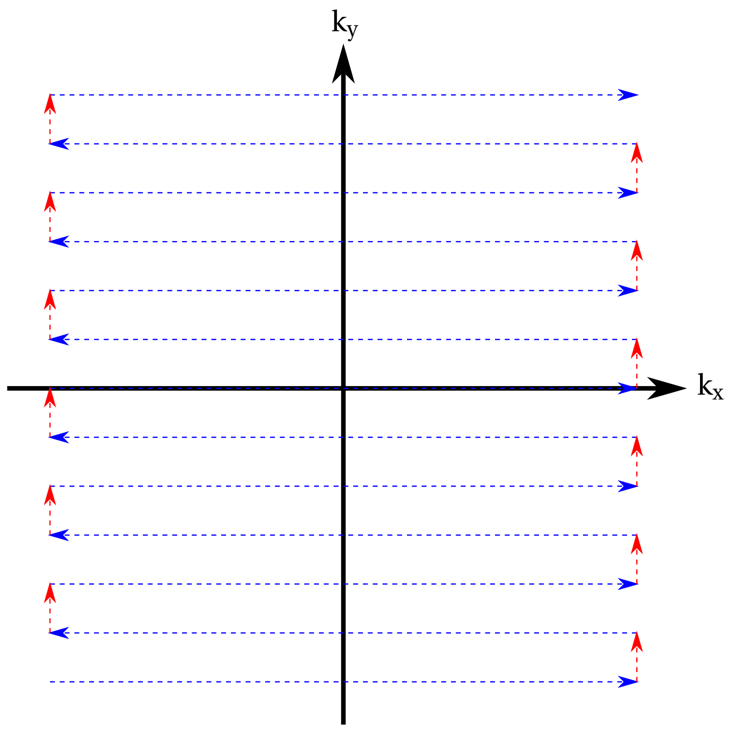


Figure 2.10: K-space trajectory of the SE-EPI sequence described. It is sampled in a raster-like pattern, where the blue dashed lines represent the read-out gradients, and the red dashed lines represent the phase encoding blips.

The main limitation of EPI is that it is prone to image artefacts. There are several reasons for this, including rapid gradient switching, the reversal of every second echo, and the long read-out period causing phase shift accumulation [15]. Common artefacts are ghosting, eddy-current artefacts, chemical shift artefacts, image distortion, blurring, and signal loss due to intravoxel dephasing [2]. In this thesis, the focus will be on susceptibility-induced geometric distortion.

2.2.2 Susceptibility-induced geometric distortion

The pixel bandwidth Δv of an MR image can be expressed as the frequency difference from one pixel to the next [15]. It is given by

$$\Delta v_x = \frac{dk_x}{dt}\Delta x \tag{2.9}$$

in the x-direction, and accordingly in the y-direction. Δx is the pixel spacing in image space. The bandwidth therefore depends on how fast k-space is traversed. In EPI, the time between sampling points in the read-out direction is very short, which gives a high bandwidth in this direction because k-space is traversed very fast. However, in the phase encoding direction, k-space traversal is slower and the time between sampling points is relatively long, because a whole k-space line is sampled in between them. This gives a quite low bandwidth in the phase encoding direction, which can give rise to artefacts along this direction.

Different materials have different magnetic susceptibility, meaning that they will become magnetised to a different extent in the presence of an external magnetic field [11]. This gives rise to magnetic field inhomogeneities, where spins will precess at slightly different resonance frequencies. Since it is the difference in frequency that enables spatial encoding, spins with a different frequency than expected will be misplaced in the resulting image [2, 15]. Due to the low bandwidth in the phase encoding direction in EPI, this effect will be enhanced. The lower the bandwidth, the larger number of pixels the spins will be misplaced. The effect will also increase with increasing field strength. This artefact is called susceptibility-induced geometric distortion, and can be seen in Figure 2.11. The distortion is especially prominent near boundaries between low and high susceptibility, such as air-tissue or bone-tissue interfaces. It appears as stretching or compression of the imaged object. Note that stretching will cause the signal intensity to be lower in that region, and compression will cause it to be higher because a larger number of spins are located in the same pixel.

This artefact is one of the major problems in EPI today. EPI is often used in fMRI, where information about function is sought. However, it is desirable to register the functional images to anatomical images, that is, to align them so that function can be connected to an anatomical position [17]. Geometric distortion can make this very difficult, and in some cases impossible. The functional information may also be compromised by the distortion.

Measures can be taken in order to reduce the extent of the geometric distortion. As mentioned earlier, a lower magnetic field strength will reduce the distortion, but this will also reduce the signal-to-noise ratio (SNR). Another way is to increase the bandwidth by decreasing the ESP and/or the ETL. However, this can compromise the spatial resolution, even though other techniques may be used to overcome this [2]. In addition, it also decreases the SNR because a higher bandwidth allows for more noise [15]. A different approach is to try to reduce the field inhomogeneities by for example choosing the imaging plane wisely, or by filling up air cavities (e.g. bowels) with another medium.

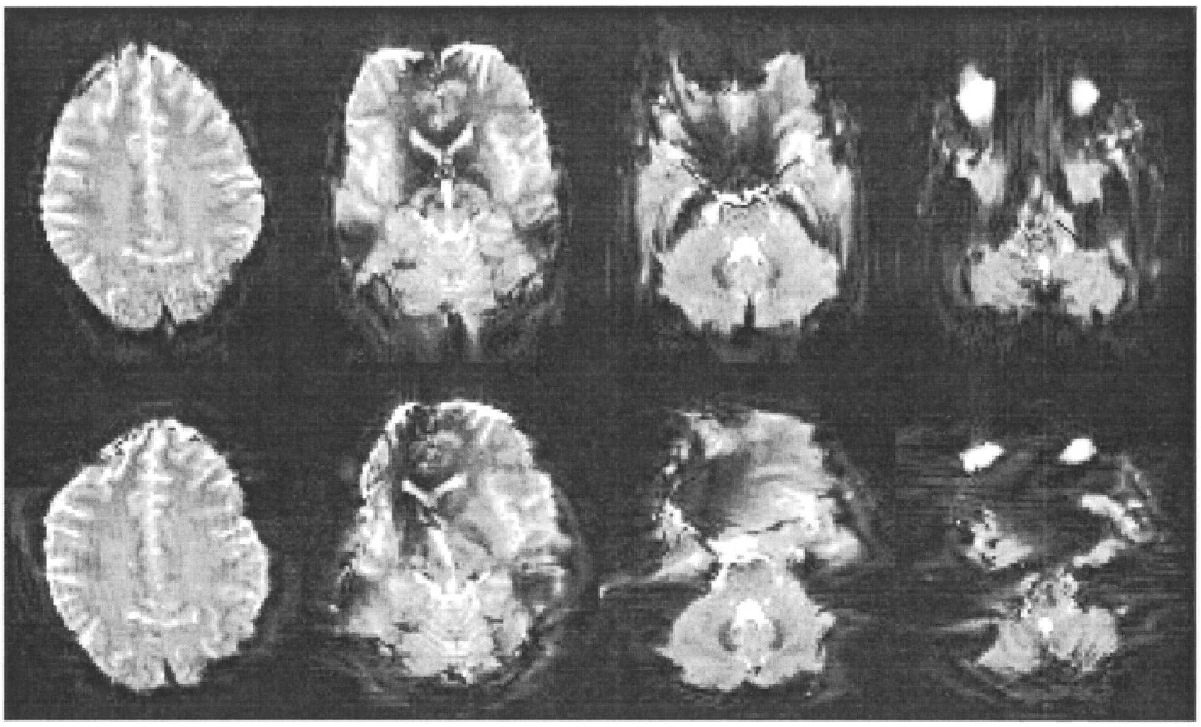


Figure 2.11: Geometric distortion in selected EPI brain slices. In the upper row, the phase encoding direction is anterior-posterior, and the distortion is therefore seen in this direction. In the bottom row, the phase encoding direction is left-right, with corresponding distortion in this direction. Reproduced with permission from [16] (©2000 IEEE).

2.2.3 Distortion correction methods

There exist several post-processing methods to correct geometrically distorted images. In this section, a variety of methods will be presented briefly, before the correction method used in this thesis will be presented in more detail.

The most commonly used method, and one of the earliest to be developed, is field map-based unwarping. A field map shows the magnetic field inhomogeneities, and based on this the pixels can be relocated [18]. By acquiring a pair of phase maps at two different echo times, the phase difference between them can be determined. The field inhomogeneity is found from this phase difference. Then, the pixels can be relocated to their correct place because the pixel shift depends on the field offset and the bandwidth per pixel. However, the acquisitions required for producing the field map take considerably longer time than acquiring the EPI images themselves, allowing for subject motion during the scans, which can lead to large errors in the resulting field map [3]. The field map method also has problems near tissue boundaries, and in regions with very high field inhomogeneity. In addition, the field map can be very noisy in areas with low signal.

Another way to correct geometric distortion is based on the point spread function (PSF). Since a distorted image is made from a convolution between the true, undistorted image and a PSF, it is possible to find the undistorted image if the PSF is known [19, 20]. By applying additional phase encoding gradients in different directions in an EPI sequence, it is possible to image the contents within each voxel. Therefore, the PSF of all the points in the image can be found, and the displacement field of the image can be determined. This method does also require relatively long scan times, but other techniques may be used to overcome this [3]. Furthermore, it is less sensitive to noise and large field inhomogeneities than the field map method.

A kind of simplified PSF method using an EPI field map also exists, known as "phase labelling for additional coordinate encoding" (PLACE) [21]. Here, two EPI acquisitions are performed, where the preparatory phase encoding gradient of the second scan has a slightly larger area than in the first scan, so that there is a slight phase shift between the two scans in the phase encoding direction. This leads to a known phase difference in the y-direction of the two sampled k-spaces. The phase difference between the distorted images is then directly encoded into the y-coordinate of each pixel in undistorted space, and the correct location of the pixel is found. As it only requires two EPI scans, and the post-processing is not too time-consuming, this method is quite fast [3].

A method with a different approach is "two reduced acquisitions interleaved" (TRAIL) [22, 23, 24]. Here, two read-outs are performed per excitation, and the data is then interleaved in image space. This reduces blurring and distortion in the phase encoding direction by a factor of 2. Additional distortion correction can be performed from measurements of local phase changes. However, the SNR is reduced by a factor of $\sqrt{2}$.

It is also possible to correct the images by directly co-registering them with anatomical images [16, 25]. If the geometric distortion is not too comprehensive, nonlinear registration might be sufficient correction. This has been successfully performed, but it can fail in areas with heavier, nonlinear distortion, and it does not correct for intensity distortions [26]. It also depends on the registration algorithm used. However, registration has been suggested to be a finer step to improve the distortion correction after another method has been performed [27].

The most promising methods today are based on reversed gradients [26]. Here, two EPI scans are performed using the phase encoding gradients in opposite directions [28]. The thought is that the geometric distortion will also be opposite, and the undistorted image will then be the midway between these two images [3]. In one of the reversed gradient methods, the unwarping is done in 1D only, along the phase encoding direction [29]. This method performs an acceptable correction along this direction, but may show streaking or discontinuities in the other directions [27].

Andersson et al. further developed this idea into an alternative reversed gradient method, where the distortion correction is performed in 3D [30]. This is accomplished by computing a continuous and smooth 3D deformation field using discrete cosine basis functions. Relocation of the pixels is then done using this deformation field. This method is able to perform relatively good distortion correction, but the computations can be time-consuming. In addition, the resolution of the distortion field is limited by the highest frequency component of the discrete cosine transform [3].

2.2.3.1 Method by Holland et al.

The method developed by Holland et al. is similar to the one by Andersson et al. in the use of a 3D deformation field, but it is more computationally efficient [3]. This method will be the focus of this thesis. Here, the deformation field is found by setting up a cost function and minimising it.

The pixel shift u caused by the distortion depends on the field offset $\Delta\nu_{\mathrm{B}}$ and the bandwidth per pixel $\Delta\nu_{\mathrm{pp}}$ in the phase encoding direction,

$$u = \frac{\Delta\nu_{\mathrm{B}}}{\Delta\nu_{\mathrm{pp}}}. \tag{2.10}$$

Let I denote the undistorted image, I_1 the forward phase encoded image, and I_2 the reverse phase encoded image. Pixel i placed at (x_i, y_i, z_i) in I will then be placed at $(x_i, y_i + u_i, z_i)$ in

I_1, and at $(x_i, y_i - u_i, z_i)$ in I_2 [3]. The Jacobian J of this transformation is given by

$$J = 1 + \frac{\partial u(y)}{\partial y}, \tag{2.11}$$

so that

$$I(x_i, y_i, z_i) = J_{1i} I_1(x_i, y_i + u_i, z_i) = J_{2i} I_2(x_i, y_i - u_i, z_i). \tag{2.12}$$

Here, J_{1i} and J_{2i} are the Jacobian of the ith pixel ($i = 1, 2, ..., N$) in the forward and reverse phase encoded image, respectively. They can be calculated discretely as

$$J_{1i} = 1 + \frac{u_{\text{ip}} - u_{\text{im}}}{2} \tag{2.13}$$

and

$$J_{2i} = 1 - \frac{u_{\text{ip}} - u_{\text{im}}}{2}, \tag{2.14}$$

where *ip* is the neighbouring pixel on the $+\hat{y}$ side of i, and *im* on the $-\hat{y}$ side. From this, a suitable cost function is

$$f(u_1, ..., u_N) = \frac{1}{N} \sum_{i=1}^{N} \left[J_{1i} I_1(x_i, y_i + u_i, z_i) - J_{2i} I_2(x_i, y_i - u_i, z_i)\right]^2 + \lambda_1 \sum_{i=1}^{N} u_i^2 + \lambda_2 \sum_{i=1}^{N} \left[\vec{\nabla}_i u_i\right]^2. \tag{2.15}$$

The first term of this function vanishes by design when the two distorted images are correctly unwarped into identical undistorted images. λ_1 and λ_2 are regularisation parameters that control the quality of the unwarping. The two last penalty terms constrain the amplitude of the displacements, and control the smoothness of the deformation field, respectively.

The cost function must then be minimised in an efficient way. It is important to make sure that it is indeed a global minimum, and to not get trapped in any local minima. This is obtained by smoothing the forward and reverse phase encoded images, for example by convolution with an isotropic Gaussian kernel, so that they become more similar [3]. It is then easier to find the global minimum, and this is done iteratively using decreasing levels of smoothing, all the way to the original unsmoothened images. The distortion field $\vec{u}$ is first initialised to zeros. The minimisation is performed using the equation

$$\mathbf{H}(\vec{u}) \cdot \vec{v} = -\vec{g}(\vec{u}), \tag{2.16}$$

where $\mathbf{H}(\vec{u})$ is the Hessian of f at $\vec{u}$, and $\vec{g}(\vec{u})$ is the gradient of f at $\vec{u}$. $\vec{v} = (v_1, ..., v_N)$ is the displacement of $\vec{u}$ that moves f to the new global minimum $\vec{u} + \vec{v}$ at the next level of smoothing. Several methods can be used to find $\vec{v}$, such as conjugate gradients squared, generalised minimal residuals, or biconjugate gradients stabilised method.

To sum up the method, $\vec{u}$ is first initialised to zeros, and the smoothing kernel is set to 3.5-4 mm [3]. Then:

1. The forward and reverse original images are smoothed.
2. $\mathbf{H}(\vec{u})$ and $\vec{g}(\vec{u})$ is set up for the current distortion field.
3. $\vec{v}$ is found by solving Equation (2.16).
4. The new distortion field $\vec{u}$ is updated to be $\vec{u} + \vec{v}$.
5. The smoothing kernel is decreased by 0.25-0.5 mm.

Step 1-5 is repeated until there is no smoothing. This method is shown to provide superior accuracy relative to the field map method (see Figure 2.12), and it is also very fast, since it only requires an additional b_0 acquisition in the opposite phase encoding direction. Diffusion-weighted images can be corrected by applying the distortion field calculated from the b_0 images.

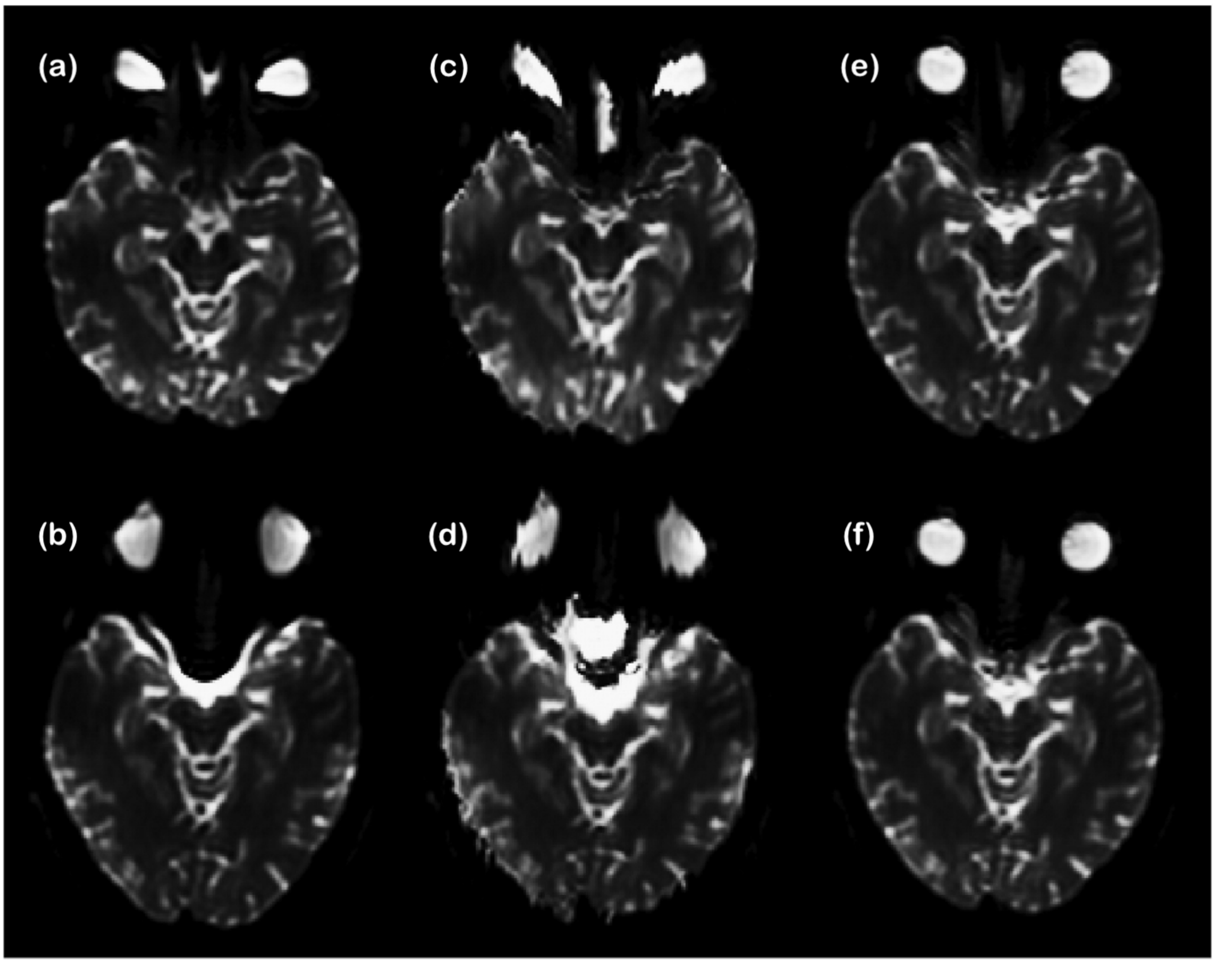


Figure 2.12: Comparison of distortion correction using the field map method and the method described by Holland et al. **(a)** Forward phase encoded image, uncorrected. **(b)** Reverse phase encoded image, uncorrected. **(c)**, **(d)** Forward and reverse images, respectively, corrected using the field map method. **(e)**, **(f)** Forward and reverse images, respectively, corrected using the method by Holland et al. The forward and reverse images look virtually identical, and the result is superior to the field map-based correction. Reproduced with permission from [3].

Chapter 3

Materials and methods

3.1 Making the prototype

In order to integrate distortion correction into the clinical workflow, a tool must be made that is easily available for the clinicians. In the Siemens software syngo.via Frontier (Siemens Healthcare GmbH, Erlangen, Germany), there is a library of various applications—so-called prototypes—that can be used for post-processing of medical images. In addition, users can make their own prototypes using MeVisLab (MeVis Medical Solutions AG, Bremen, Germany; Fraunhofer MEVIS, Bremen, Germany). This software provides a framework for image processing, using modules based on C++ and Python programming. Here, the process of making a prototype for distortion correction of EPI images will be described.

3.1.1 Requirements

There are certain requirements this prototype needs to fulfil:

- To let the user select forward and reverse phase encoded b_0 images, and diffusion-weighted images.
- To let the user choose certain parameters for the distortion correction.
- To correct geometric distortion in all the images mentioned above.
- To let the user view both the uncorrected and the corrected images.
- The corrected images can be sent back to the syngo.via server.
- It must be easy for the user to navigate through the prototype.

3.1.2 Starting point

To make a prototype for syngo.via Frontier, the MeVisLab add-on packages Frontier Development Kit and MR StarterKit (Siemens Healthcare GmbH, Erlangen, Germany) must also be installed. These contain modules that are necessary for compatibility with the Frontier environment. MR StarterKit also provides a demo prototype which can be used as a basis to develop own prototypes suitable for MR images, and this was used as a starting point for the distortion correction prototype. This demo prototype had some simple functionality where the user could select an image to be rendered, in addition to a basic image processing example. There was also a menu with some example buttons. A demo prototype can be made using the "Create Modules" wizard that comes with the MR StarterKit. Note that a Frontier license is needed to exploit all the functionalities of MeVisLab and the Frontier add-on packages, and to build the prototype installer.

When building the distortion correction prototype in MeVisLab, it is represented by the module EPIdistortionCorrection (Figure 3.1), with all the implementation hidden as internal networks (Figure 3.2) and scripts. The modules represent a task or some functionality, for example a step in the image processing or how the layout in the prototype should be. They can either be built from internal networks containing other modules, or only from scripts, or from a combination of both. Normally, the module is an object that is defined by a C++ script, while its functions are implemented using Python scripts. The scripts belonging to a module can be found by right-clicking on a module and choosing "Related Files". A set of scripts is generated automatically when creating the demo prototype using the wizard, and most of these can be tailored, but own scripts can also be created later.

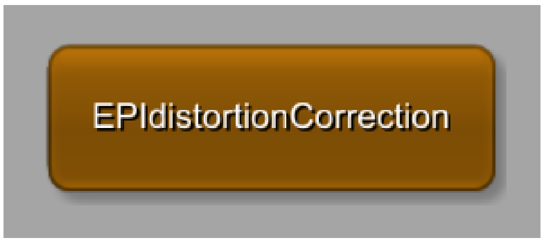


Figure 3.1: The module EPIdistortionCorrection.

The modules have different colours depending on their type. Blue modules perform some kind of image processing, green modules modify the layout and control how the images

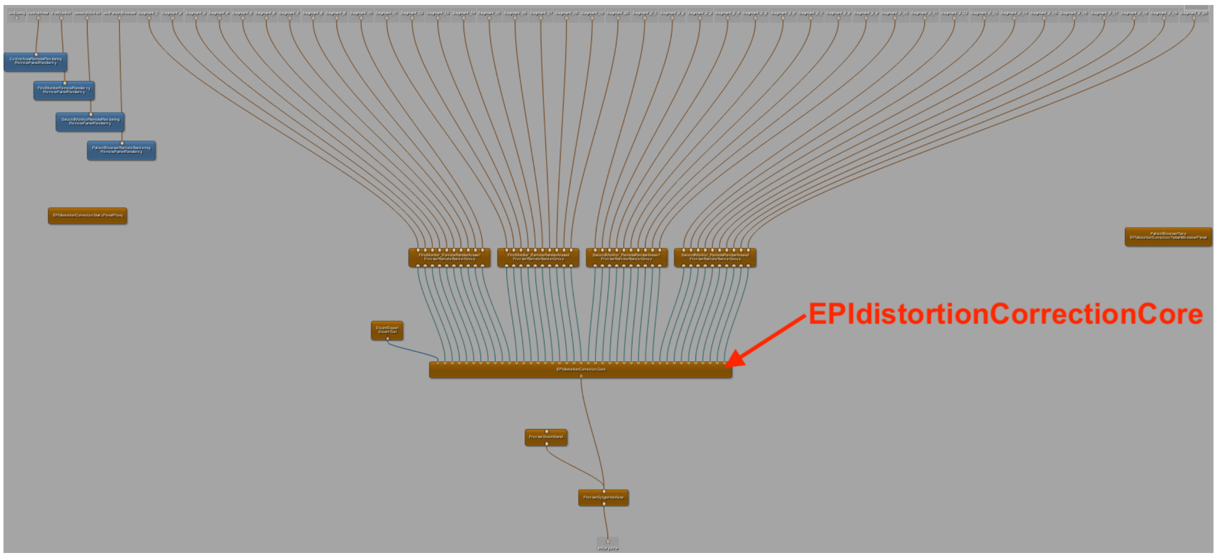


Figure 3.2: Overview of the internal network of the EPIdistortionCorrection module. This particular network is not intended to be tailored and its details are therefore not described here. The core module of the prototype, EPIdistortionCorrectionCore, is indicated with a red arrow. This module will be modified to achieve the desired functionalities.

are rendered in the prototype, and the brown modules are combinations of the other types. MeVisLab and the add-on packages come with a library of modules, but own modules can also be made. Modules can be named with instance names to keep track of what they are used for, and then the instance name is shown in the upper line on the module, and the module name is shown in the bottom line. The modules also have panels with choices and parameters that are shown when the modules are double-clicked.

The modules can have zero, one or several input and output fields. In/out fields that look like triangles represent images, squares represent pointers to data structures, and half-circles represent some kind of layout design or image rendering. Coloured lines connect the in/out fields of different modules together. In addition, there are parameter connections represented by thin dark grey lines with an arrow between modules, and these pass on internal parameter values from one module to another.

In the following, the process of modifying the demo prototype—in order to meet the requirements from Section 3.1.1—will be described. Relevant scripts are included in Appendix A.

3.1.3 Loading several images into the prototype

The demo prototype only comes with the possibility to load one image series. Since it might be necessary to load images from different series to perform the distortion correction, this feature must be modified in the internal network of EPIdistortionCorrectionCore (Figure 3.3). In the demo prototype there was only one MRPatientBrowserRoles module, so two more were added to the network (Figure 3.4). The scripts EPIdistortionCorrection.py (A.1) and EPIdistortionCorrectionPatientBrowserPanel.py (A.2) were also modified in order to let the user first choose the forward phase encoded b_0 image (MRPatientBrowser), then the reverse phase encoded b_0 image (MRPatientBrowser2), and then all the diffusion-weighted images (MRPatientBrowser3). Since some of the image volumes might be represented as a 2D mosaic image, DeMosaic modules were added to convert them into 3D images. Mosaics are 2D image grids that are made up from all the slices of a 3D volume. Switch modules were added to allow the user to select whether the images initially were mosaics.

3.1.4 Image processing

After the images have been loaded into the prototype, they are sent into the EPIdistortion-CorrectionProcessing_Custom module (Figure 3.5). If the user has selected more than one forward or reverse phase encoded b_0 image, they are averaged by a custom-made module (MeanOfPhases) before the main processing is performed. This is because the calculation of the distortion field can only take in one forward and one reverse phase encoded image, but the user might want to take the average of several images to increase the SNR.

3.1.4.1 Distortion correction

The distortion correction is performed using the command-line tool CMTK (Computational Morphometry Toolkit; Neuroimaging Informatics Tools and Resources Clearinghouse, Menlo Park, CA, United States, `www.nitrc.org`) [31, 32]. Its distortion correction function, called `epiunwarp`, is based on the method by Holland et al. described in Section 2.2.3.1. This function corrects the forward and reverse b_0 images and calculates the distortion field, which in turn can be used to correct diffusion-weighted images. The functions `reformatx` and `imagemath` are used to correct these. Note that for these functions to work properly in Windows, the Cygwin version of CMTK should be installed. Cygwin, which provides a Unix-like command line environment, must also be installed. To make this run smoothly when inte-

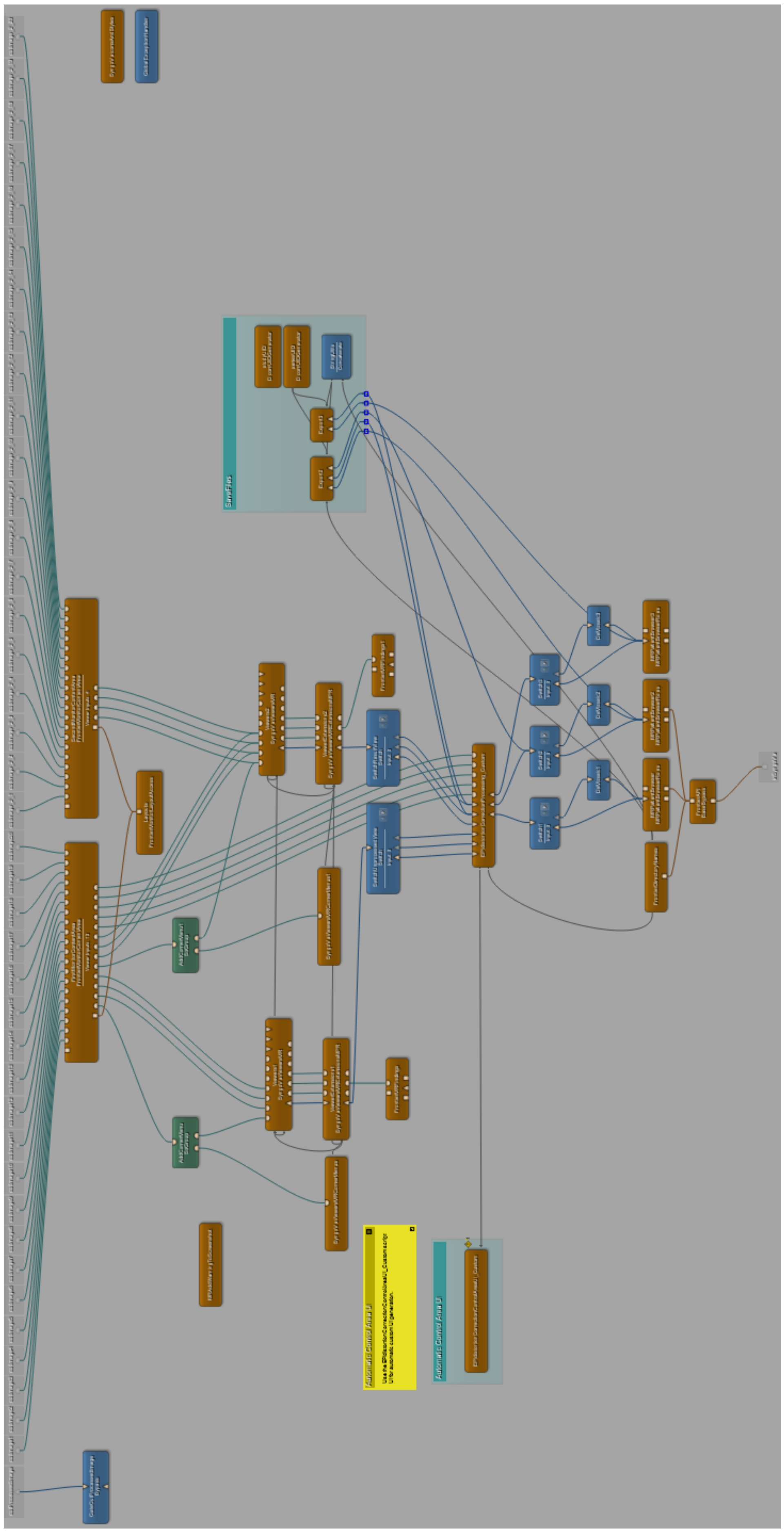


Figure 3.3: Overview of the internal network of the EPIdistortionCorrectionCore module. This is the network after it has been modified. Relevant parts will be shown in more detail later.

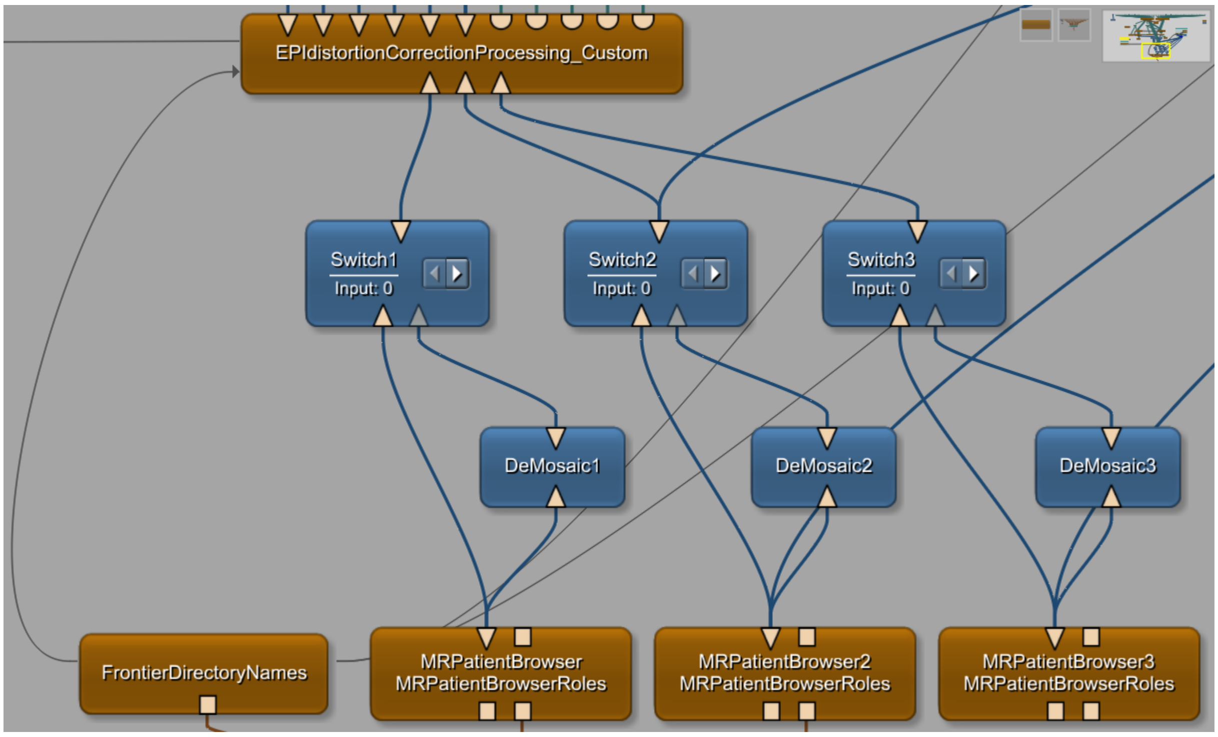


Figure 3.4: There are three MRPatientBrowserRoles modules so that the user can select first the forward and then the reverse phase encoded b_0 images, and then all the diffusion-weighted images. DeMosaic modules resolve 2D mosaic images into 3D volumes, and the Switch modules allow the user to choose whether the images initially are mosaics. After the images have been loaded into the prototype, they are sent into the main processing module, EPIdistortionCorrectionProcessing_Custom.

grating CMTK into MeVisLab, the Cygwin binaries should be placed in the same folder as the CMTK binaries.

The CMTK tools are the ones already in use for distortion correction at NTNU, but here they will be implemented more user-friendly for users who are not accustomed with this type of command-line tools. The method will also be easier integrated into clinical workflow, since the images do not need to be anonymised, downloaded, and transported to a dedicated research computer before processing in syngo.via Frontier.

The external executables are integrated into the MeVisLab network by means of the ExternalExecutableWrapper module, which comes with the MR StarterKit package (Figure 3.6). First, the distortion field is calculated from the forward and reverse phase encoded b_0 images, and these are corrected, using the `epiunwarp` function. The default optimisation parameters of the `epiunwarp` function were used. Then, `reformatx` and `imagemath` use this

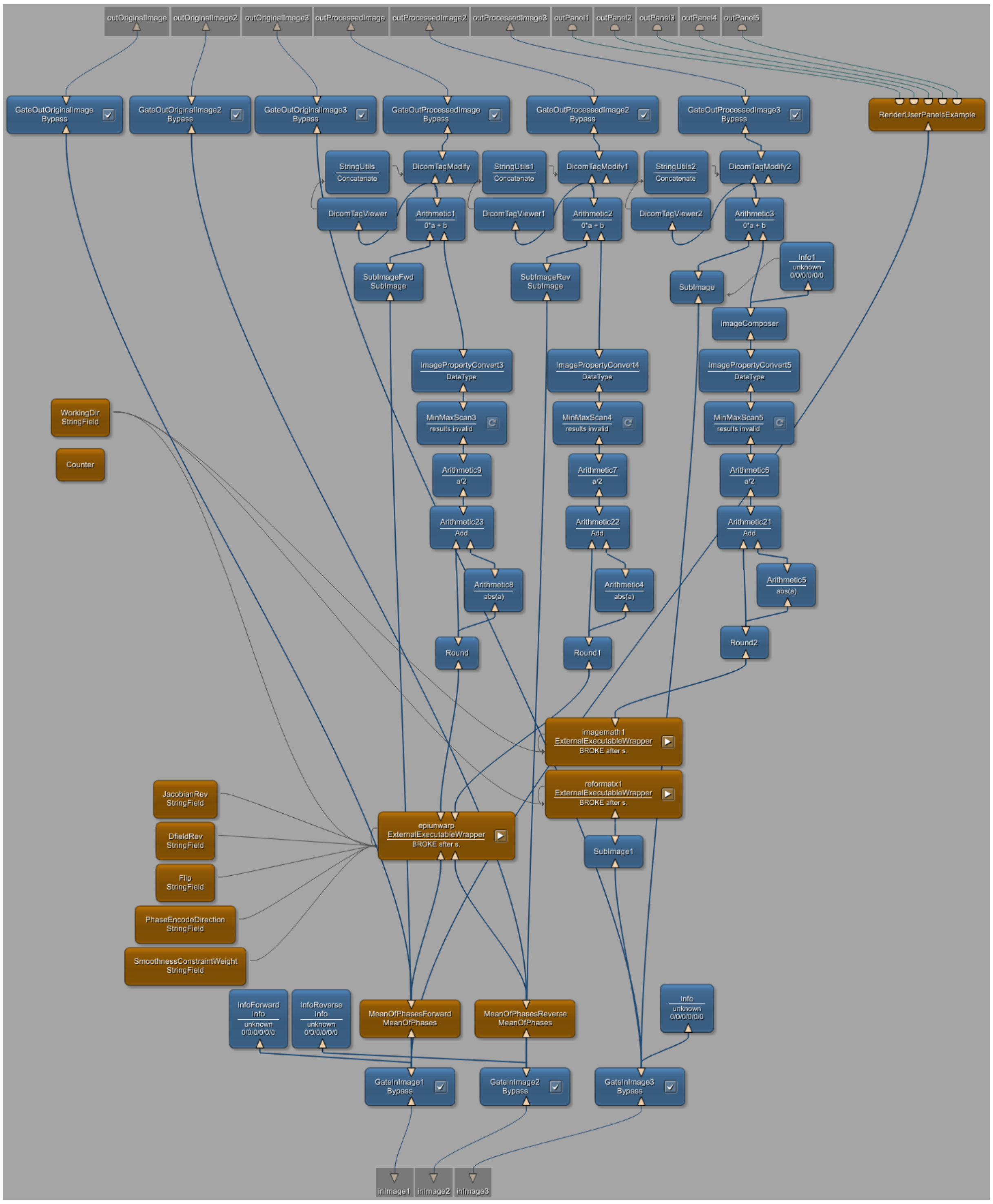


Figure 3.5: Overview of the internal network of the EPIdistortionCorrectionProcessing_Custom module. Important parts of the network will be shown in more detail later.

distortion field to correct the diffusion-weighted images. However, these functions can only handle 3D images, whereas the DW images are stored in a 4D volume. This is solved by using the SubImage module to select 3D volumes one by one, which are passed on to `reformatx` and `imagemath`. This process is iterated until all the diffusion-weighted images are corrected. To obtain this, a FieldListener is created that starts `imagemath` when `reformatx` is finished, and another FieldListener is created that starts `reformatx` when `imagemath` is finished if there are still more images to be corrected (A.3, A.4). Also, the index of the 3D image selected by SubImage is incremented when `imagemath` has finished. The corrected 3D images are composed into a 4D volume again later.

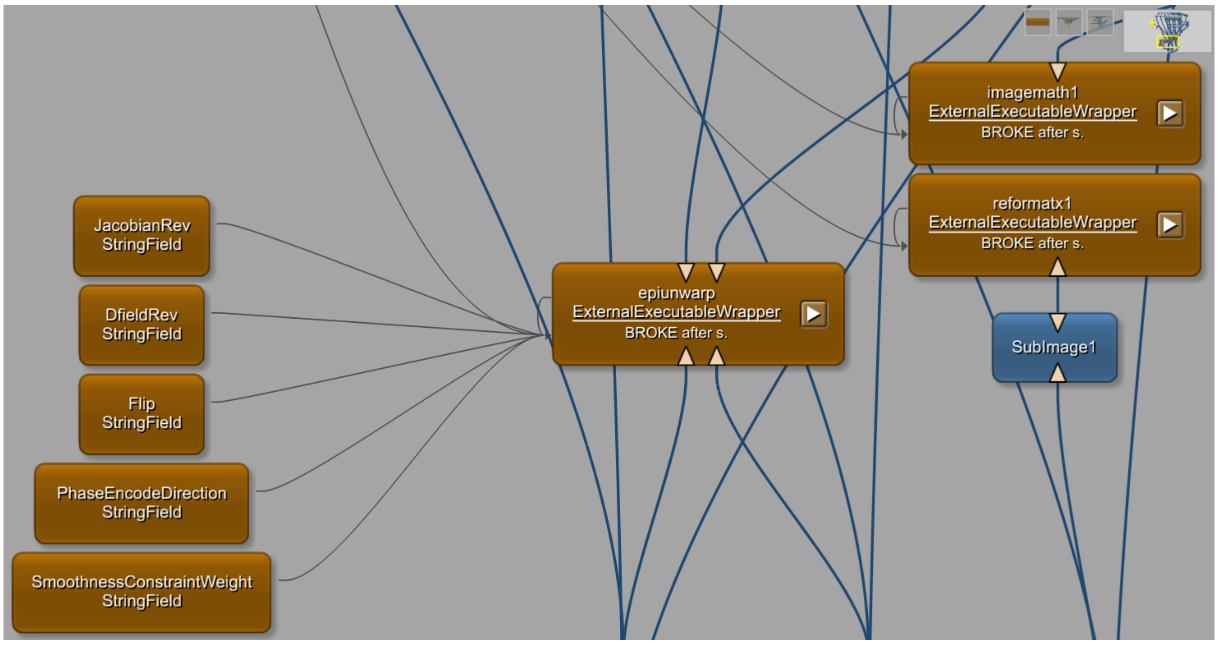


Figure 3.6: Distortion correction using the ExternalExecutableWrapper module. The StringField modules on the left are used to pass on arguments to the `epiunwarp` function. `epiunwarp` is executed first, and then the calculated distortion field is used to correct the diffusion-weighted images using `reformatx` and `imagemath`. Since these functions only apply to 3D volumes, 3D images are first selected from the 4D diffusion-weighted series using the SubImage module. `reformatx` and `imagemath` are then iterated for each 3D volume in the DW image series. The corrected 3D images are composed into a 4D volume again later.

After distortion correction, the images are in the NIfTI format. Since they are going to be stored in the DICOM format, the data type must be converted from float to 16-bit unsigned integer. However, the distortion correction may have given some of the pixels a negative value, and these will then be converted into large, positive integers. This problem was solved by setting all negative pixels to zero before changing the data type. To obtain this the pixel values were first rounded, and then the image and the absolute value of the image were added together, before the resulting image was divided by two. It was discovered that the module Arithmetic had some problems with adding the images correctly when negative pixels were involved, so an Arithmetic2 module was used to add the images together. The float image was then converted to 16-bit unsigned integer using the ImagePropertyConvert module. At last, all of the corrected 3D diffusion-weighted images were composed into a single 4D volume again using the ImageComposer module.

3.1.5 Image rendering layout

Back in the internal network of EPIdistortionCorrectionCore (Figure 3.3), the image rendering layout of the prototype can be tailored. The SyngoViaViewersMRExtensionsMPR and the SyngoViaViewersMR modules control this (Figure 3.7). In the demo prototype, the user can select whether the colour bar should be visible only for the unprocessed image, but not for the processed image. The same goes for choosing the type of colour map. This is because only the aforementioned modules for the unprocessed image are connected to SyngoViaViewersMRCornerMenus and AddCornerMenu modules. To get this functionality also for the processed image, these modules have to be replicated and connected in the same way to the image rendering modules for the processed image.

It is also desirable to have the same look-up tables (LUTs) for both the unprocessed and the processed images, so all fields in ViewerExtensions1 controlling LUT values are passed on with connectors to ViewerExtensions2. The LUT determines the screen intensity of a given pixel value, so that using different LUTs can give different contrast in the image. In addition, Switch modules are added before these so that the user can choose whether the forward or reverse phase encoded b_0 images should be rendered, or the diffusion-weighed images. The different layout types of the prototype can be changed in EPIdistortionCorrectionCoreLayouts_Custom.py (A.5). The default layout is set to "Single Screen 2x3", with 2x3 images where the top three images show transverse, sagittal and coronal view of the unprocessed image, and the bottom three images show the same for the processed image.

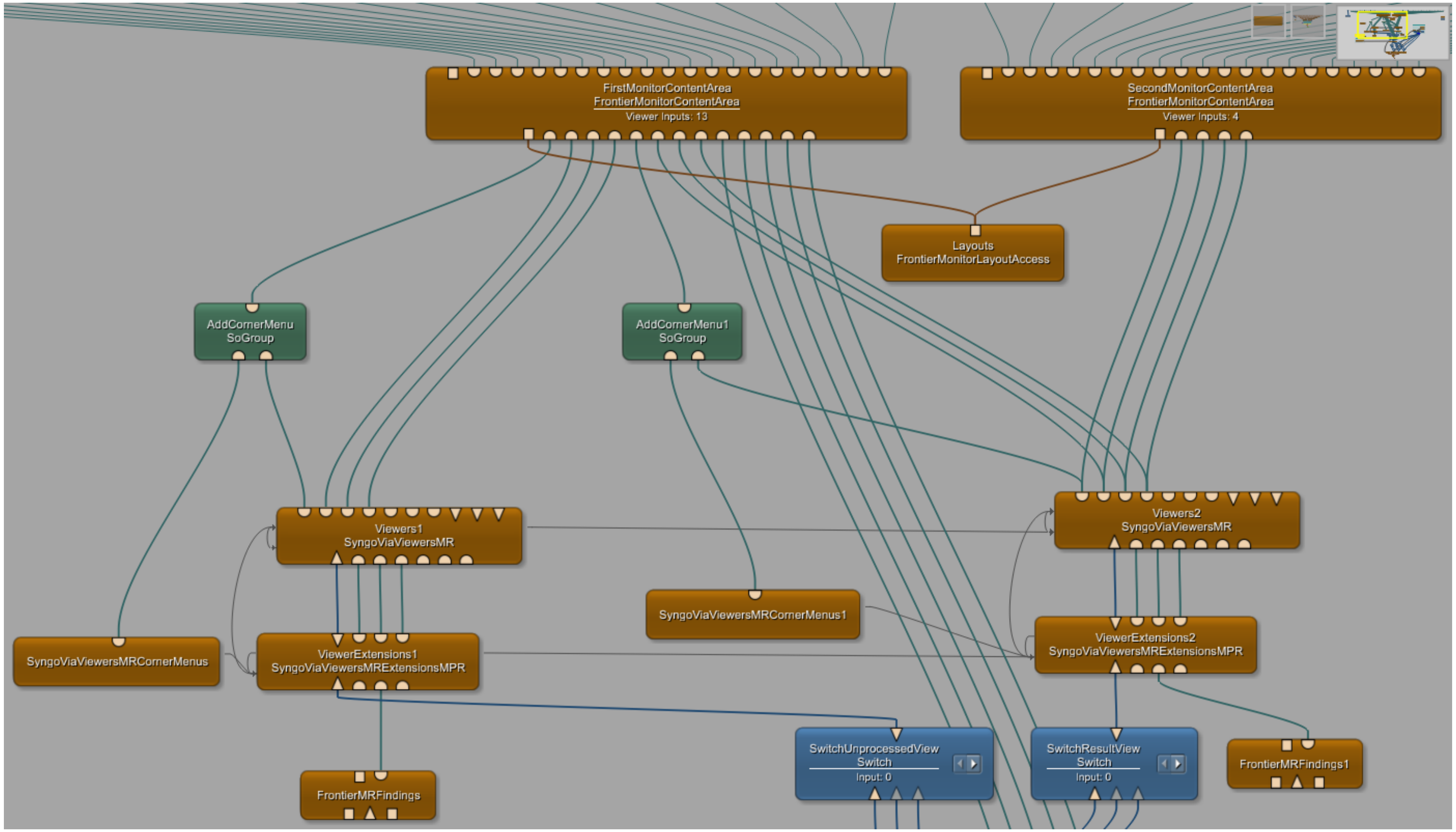


Figure 3.7: The part of the internal network of EPIdistortionCorrectionCore that controls the image rendering layout of the prototype. Switch modules allow the user to choose which images to be rendered (forward or reverse b_0, or diffusion-weighted). SyngoViaViewersMRCornerMenus and AddCornerMenu are replicated and connected to ViewerExtensions2 and Viewers2 in the same manner as to ViewerExtensions1 and Viewers1.

3.1.6 Saving images

The SaveFiles group in the internal network of EPIdistortionCorrectionCore is used to save the corrected images (Figure 3.8). It contains DicomUIDGenerator modules to create new unique identifiers (UIDs) for the headers of the new DICOM files, a StringUtils module to pass on the directory name where the images should be stored, and two custom-built Export modules to save the images. Export2 is used to save the corrected reverse phase encoded b_0 image, whereas Export3 is used to save all the corrected forward phase encoded images (both b_0 and DW).

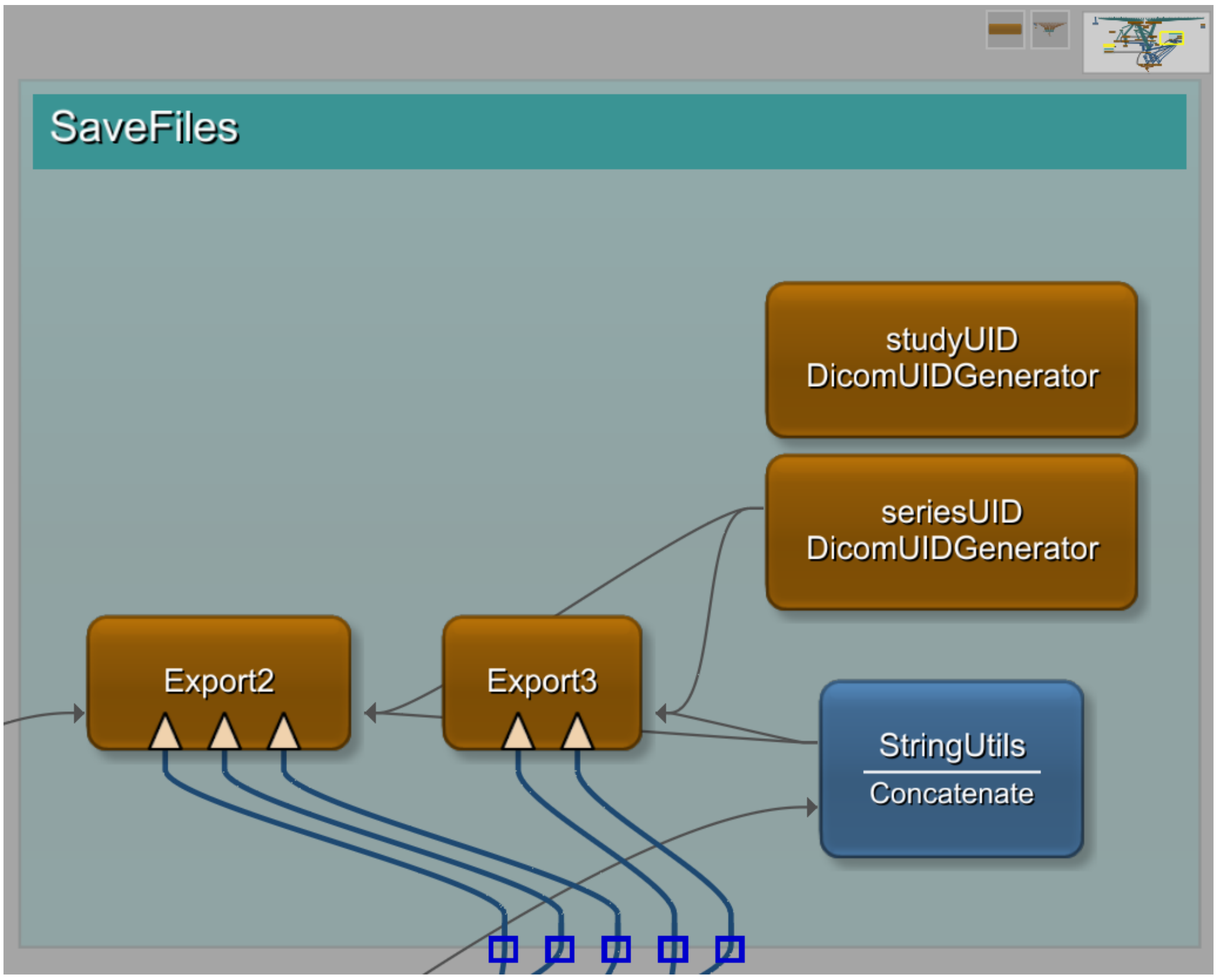


Figure 3.8: The group SaveFiles is used to save the corrected images. Export2 is used to save the reverse phase encoded b_0 image, while Export3 is used to save all the corrected forward phase encoded images (both b_0 and diffusion-weighted).

The internal network of the Export modules is shown in Figure 3.9. However, in the Export module that saves the reverse phase encoded image, there is in addition a distortion correction step that is applied if there are more than one reverse phase encoded image, because these were not corrected inside EPIdistortionCorrectionProcessing_Custom. These images are then corrected using SubImage, `reformatx` and `imagemath` in the exact same manner as before. This is necessary for the DICOM header tags from the uncorrected image to be preserved, because in MeVisLab the frame-specific tags cannot be restored if the image matrix size is not the same as in the original image. Therefore, the corrected reverse and forward (including diffusion-weighted) phase encoded images series must be saved with the same dimensions as the uncorrected image series. Because of this, it is important that the user selects all the images in the series when choosing the images to be distortion corrected in the prototype.

If the images initially were mosaics, they are composed back into a mosaic image using the custom-made MakeMosaic module (Figure 3.10). The itkFlipImageFilter is used to flip the z-axis of the image, because the DeMosaic module initially flipped it. SubImage is then used to select one 2D frame from the image volume, and ImageComposer then places the frame on a certain position on the image grid. The whole process is controlled by the functions makeMosaicFwd and makeMosaicRev in EPIdistortionCorrectionCoreProcessing_Custom.py (A.6).

When the image dimensions of the corrected image match those of the original image, the DICOM header of the original image is passed on to the corrected image. This is done using an Arithmetic module with the expression $0 * a + b$, where a is the original image and b is the corrected image (Figure 3.9). This works because it is always the DICOM header from image a that is passed on from the Arithmetic module. Then, a MinMaxScan module is used to set the correct minimum and maximum values of the image, before some of the DICOM tags are modified using a DicomTagModify module. Here, the postfix "`/DICO`" is added to the ImageType tag of the image, a new seriesInstanceUID is given, and the postfix "_DistortionCorrected" is added to the SeriesDescription.

When the image dimensions and the DICOM header of the corrected image are correct, the image can be exported back to the syngo.via server using the AnnotatedDicomExport module, which comes with the MR StarterKit. Also, for testing of the prototype, it is desirable to be able to save the images locally. This is performed using the DicomTool module. Note that for the frame-specific DICOM tags to be preserved, the box "Restore frame specific" in the module's panel must be checked.

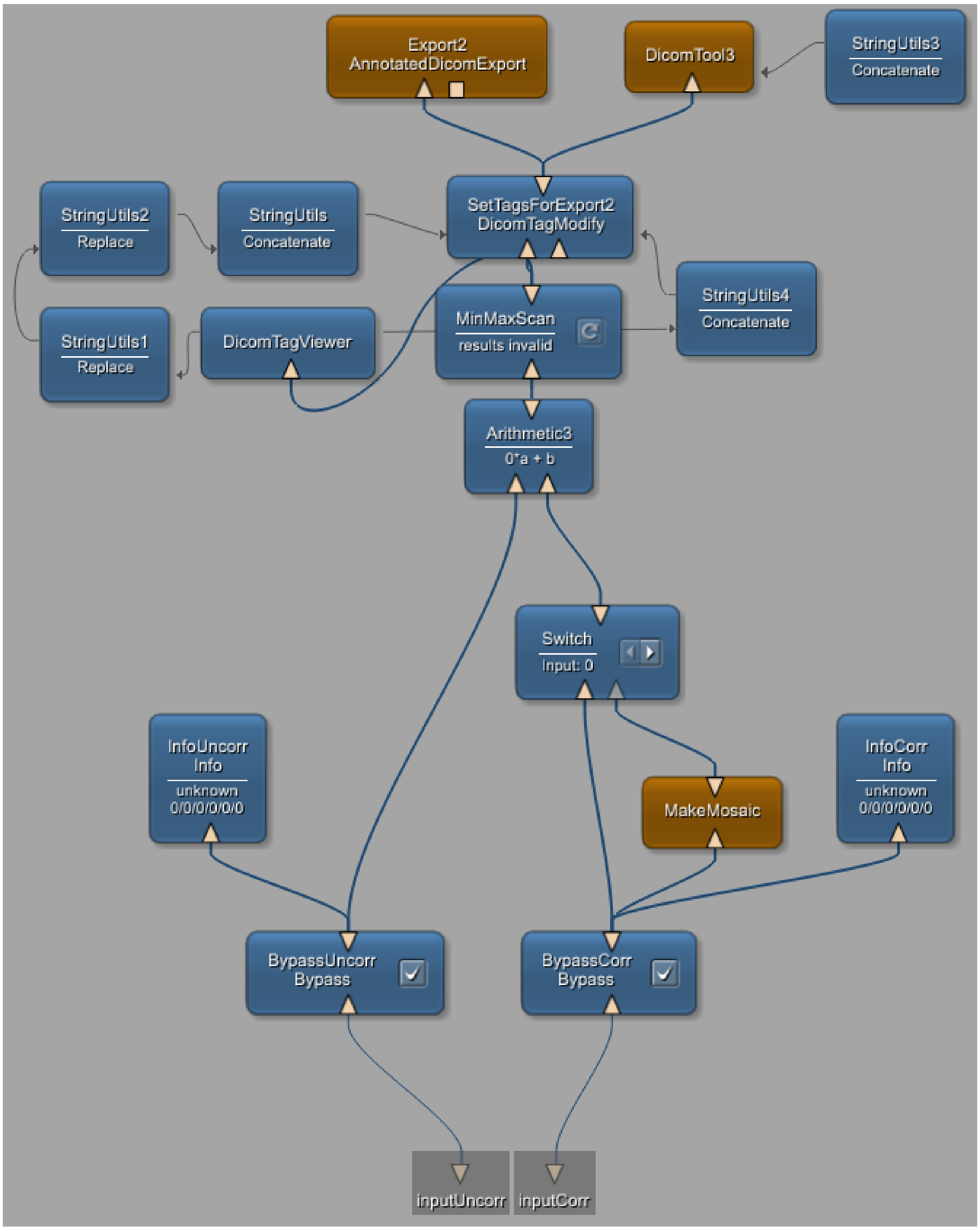


Figure 3.9: The internal network of an Export module. Note that in the Export module that saves the reverse phase encoded image, there is in addition a distortion correction step that is applied if there are more than one reverse phase encoded image. If the image initially was a mosaic, it is composed back into a mosaic image with the MakeMosaic module. DICOM header tags are then modified, before the image is saved.

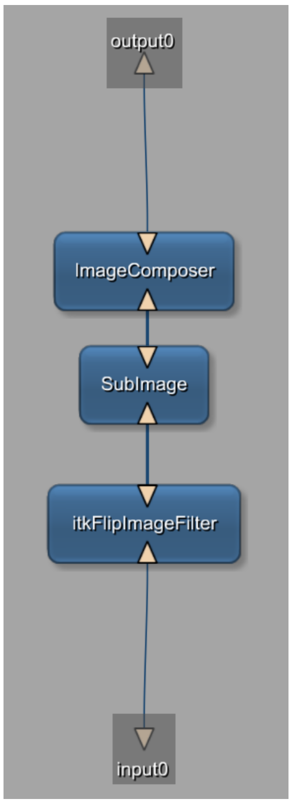


Figure 3.10: The internal network of the MakeMosaic module. The z-axis of the images is flipped by the itkFlipImageFilter module, a frame is then selected by SubImage, which is placed on the image grid by ImageComposer.

3.1.7 User interface

Designing the user interface is an important step in order to make the prototype easy to use. First, the user has to choose the forward and reverse phase encoded b_0 images, and the diffusion-weighted images. To guide the user, pop-up windows were created to tell the user which images to choose and when. This was implemented in EPIdistortionCorrection.py (A.1). Note that screenshots of the user interfaces described here will be shown in Section 4.1.

After choosing the images, the user enters the main window of the prototype. Here, the uncorrected and the corrected images will be shown, and there is a sidebar menu on the left side of the screen. This feature originated from the demo prototype, but the contents of the menu were modified using EPIdistortionCorrectionCoreControlArea_Custom.py (A.7). Functions that are called when pressing various buttons are defined in EPIdistortionCorrectionCoreProcessing_Custom.py (A.6).

The upper tab of the sidebar menu contains information about the prototype, and how to perform the distortion correction. In the second and third tabs, the user can select which unprocessed and processed images should be rendered (forward or reverse b_0, or diffusion-weighted) in the top and bottom part of the screen, respectively. Then, the tabs containing the distortion correction steps follow, and these are:

1. Parameter selection: The user selects whether the images are mosaics, and whether the reverse phase encoded image is flipped relative to the forward phase encoded image. The phase encoding direction of the images must also be chosen.

2. Calculate distortion correction: Clicking the button in this tab initiates the `epiunwarp` function in EPIdistortionCorrectionProcessing_Custom, which corrects the forward and reverse phase encoded b_0 images. When `epiunwarp` is running, a "knight rider"—a bar of alternating brightness—appears above the information tab to show the user that the prototype is working on the distortion correction.

3. Correct images: Clicking the button in this tab initiates the loop with SubImage, `reformatx` and `imagemath` in EPIdistortionCorrectionProcessing_Custom, that corrects all the forward phase encoded (including diffusion-weighted) images. The number of images currently corrected is shown to the user to show the progression of the distortion correction. If there are more than one reverse phase encoded b_0 image, the correction of these (in the Export2 module) automatically starts after the correction of all the forward phase encoded images has finished.

4. Save results: Here, the user can export the corrected images back to the syngo.via server.

In addition, a tab was made with the option to save the images locally. However, this step was only used in the testing of the prototype, and does not appear in the finished prototype.

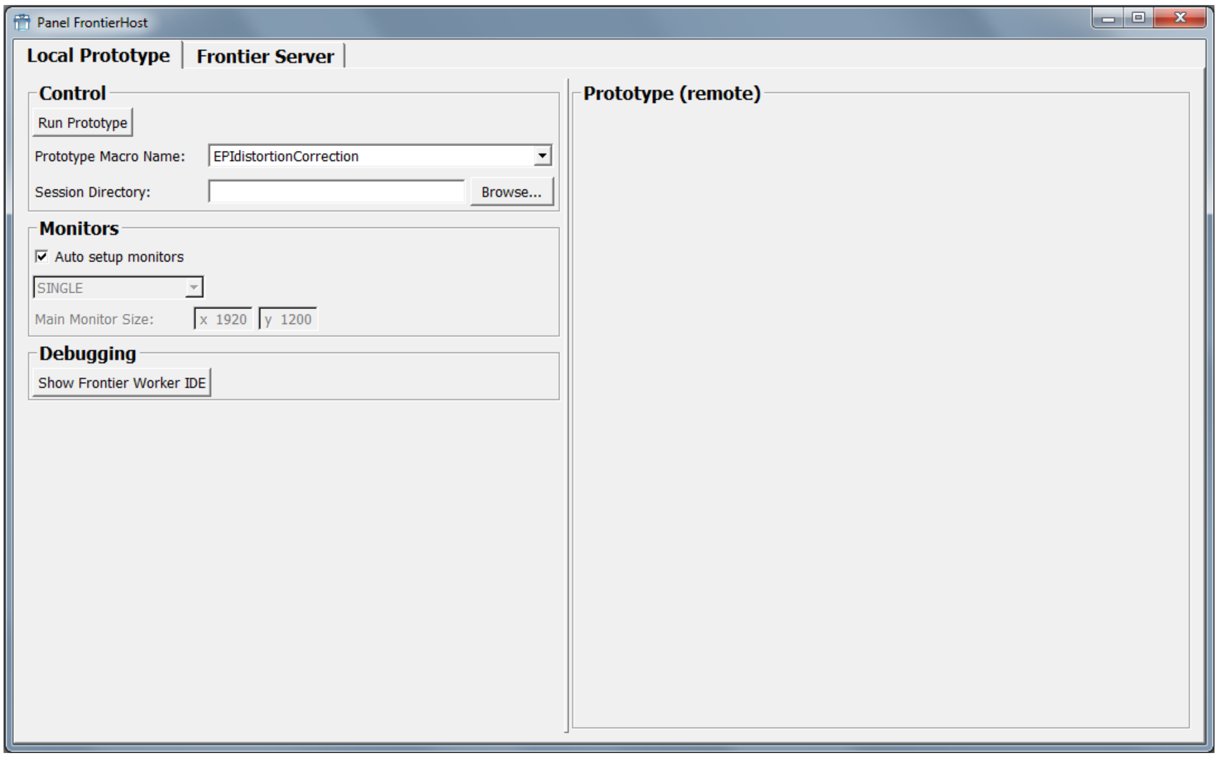
There are also some buttons in the bottom of the sidebar menu, but these originate from the demo prototype. The only modification here was that a "Send Results" button was removed, since this option was put into step 4.

3.1.8 Making an installer

For the prototype to be used in syngo.via Frontier, an installer must be made. This can be done using the "Create Installer" wizard that comes with the MR StarterKit. Note that some additional external tools might need to be installed to make the prototype installer. This can be checked using the wizard. The prototype can then be installed on any computer with syngo.via Frontier.

3.2 Testing the prototype

The Frontier Development Kit package comes with a module called FrontierHost, which can be used to test the prototype in MeVisLab (Figure 3.11). Running the prototype locally in FrontierHost gives basically the same functionality as running it in syngo.via Frontier, except from saving the images to the syngo.via server. FrontierHost was therefore used for testing and debugging during development of the prototype.



Prototype Macro Name: EPIdistortionCorrection
Session Directory:
☑ Auto setup monitors
SINGLE
Main Monitor Size: x 1920 y 1200

Figure 3.11: The window of the FrontierHost module. Here, the prototype can be tested locally in MeVisLab without installing it in syngo.via Frontier. The name of the prototype and the directory of the input images can be chosen on the left side of the window, while the prototype interface will be shown on the right side of the window.

3.2.1 Correcting images

The performance of the distortion correction was tested on breast and prostate diffusion-weighted images. Sagittal EPI breast images from 13 patients, and coronal and transverse EPI prostate images from 16 patients were distortion corrected. All images were acquired at 3T (details are shown in Table 3.1). The breast patients had three forward phase encoded b_0 images, one reverse phase encoded b_0 image, and 60 diffusion-weighted images (30 images with b = 200, 30 images with b = 700). Phase encoding direction was anterior-posterior. These had varying degrees of motion between image volumes, and were therefore motion corrected before distortion correction using the CMTK functions `registrationx` and `reformatx`. The prostate patients had one forward phase encoded b_0 image, one reverse phase encoded b_0 image, and three trace-weighted DW images (b = 50, b = 400 and b = 800), for both coronal and transverse images. Phase encoding direction was right-left. There was no discernible motion between these images.

All images were distortion corrected running the prototype in FrontierHost, and the corrected images were stored locally. The number of failed corrections was determined by visual examination of the images. A correction was defined as failed if the corrected image was unchanged from, or even worse than the uncorrected image. The robustness was then determined, defined as the percentage non-failed corrections. The quality of the corrected images was also assessed visually. The quality of the correction was defined as successful if the resulting image was the midway between the forward and reverse phase encoded images, and the shapes of the anatomy looked apparently natural. To compare the similarity between the forward and reverse b_0 images before and after correction, the mean squared difference (MSD) between them was calculated. CMTK's `similarity` function was used for this. The MSD looks directly at the intensity difference between pixels of the two images, and is therefore suitable to compare images with the same type of contrast. A paired, two-sided Wilcoxon signed-rank test was performed on the MSD values before and after correction, and results with p-value < 0.05 were defined as statistically significant [33]. This test was chosen because with the relatively small cohort of this project, the values cannot be expected to be normally distributed. The MATLAB function `signrank` was used to perform the signed-rank test (R2017b, The MathWorks, Inc., Natick, Massachusetts, United States). In addition, the time it took to go through all the distortion correction steps of the prototype was noted.

Table 3.1: Overview of the MR sequences used. The breast images were acquired on a 3T Skyra scanner (Siemens Medical Systems, Erlangen, Germany). The prostate images were acquired on a 3T Biograph mMR scanner (Siemens Medical Systems, Erlangen, Germany).

Image type	**Sequence type**	**TR (ms)**	**TE (ms)**	**Matrix size**	**FOV (mm)**	**In-plane resolution (mm)**	**Slice thickness (mm)**	**# slices**	**B-values (s/mm^2)**	**Phase encoding direction**
Breast DWI	Single-shot SE-EPI	9300	85	90×90	180×180	2.0×2.0	2.5	60	0, 200, 700	Anterior-posterior
Breast T_2	Turbo SE	5530	118	256×256	180×180	0.7×0.7	2.5	60		
Breast T_2, fat-suppressed	Turbo SE	7900	73	256×256	180×180	0.7×0.7	1.5	60		
Prostate DWI, coronal	Single-shot SE-EPI	10400	67	98×100	250×256	2.56×2.56	4.0	40	0, 50, 400, 800	Right-left
Prostate DWI, transverse	Single-shot SE-EPI	6100	67	96×102	240×255	2.5×2.5	3.0	23	0, 50, 400, 800	Right-left
Prostate T_2, coronal	Turbo SE	5470	101	320×320	192×192	0.6×0.6	3.0	19		
Prostate T_2, transverse	Turbo SE	6840	104	384×384	192×192	0.5×0.5	3.0	23		

3.2.1.1 Co-registration

Co-registration was also used in order to determine the quality of corrections. This was performed by building a network containing the CMTK functions `registrationx` and `reformatx` (Figure 3.12). First, an uncorrected forward phase encoded b_0 image is co-registered with a T_2-weighted image, and then a corrected b_0 image is co-registered with the same T_2-weighted image. T_2-weighted images were used for the co-registration because they are anatomical images, and they are therefore the gold standard when investigating whether the corrected EPI images are anatomically correct.

Both the breast and prostate images were co-registered to T_2-weighted images using normalised mutual information (NMI), which quantifies the amount of information obtained about an image when looking at the other image, as the registration metric [34]. Using normalised cross-correlation (NCC) as the metric, where the pixel-wise cross-correlation of the images is maximised, was also initially tried but did not give a successful co-registration [35]. While NCC assumes a linear relationship between the pixel intensities of the two images, NMI also looks at the pixel intensities but allows them to have different ratios between the images (e.g., bright areas in one image can be dark in the other image). Therefore, NMI is more suitable when co-registering images with different types of contrast. For the prostate patients, co-registration using NMI was also performed for cropped images with only the prostate area in the field of view (FOV), to see whether the edges of the images had any effect on the results. In addition, for 11 of the breast patients, fat-suppressed T_2-weighted images had been acquired. The contrast of these is more similar to the b_0 images, so they were also co-registered with these. Here, NCC was used as the registration metric because it gave a better co-registration than using NMI. For all co-registered images, NMI, NCC and MSD were calculated using CMTK's `similarity` function in order to quantify the similarity between the b_0 and the T_2-weighted images before and after correction. The signed-rank test was also performed for these values. These metrics were used for comparison between uncorrected and corrected images, to try to determine the quality of corrections.

3.2.2 User-friendliness

The prototype was installed on a computer with syngo.via Frontier, and tested to make sure that it worked as intended there.

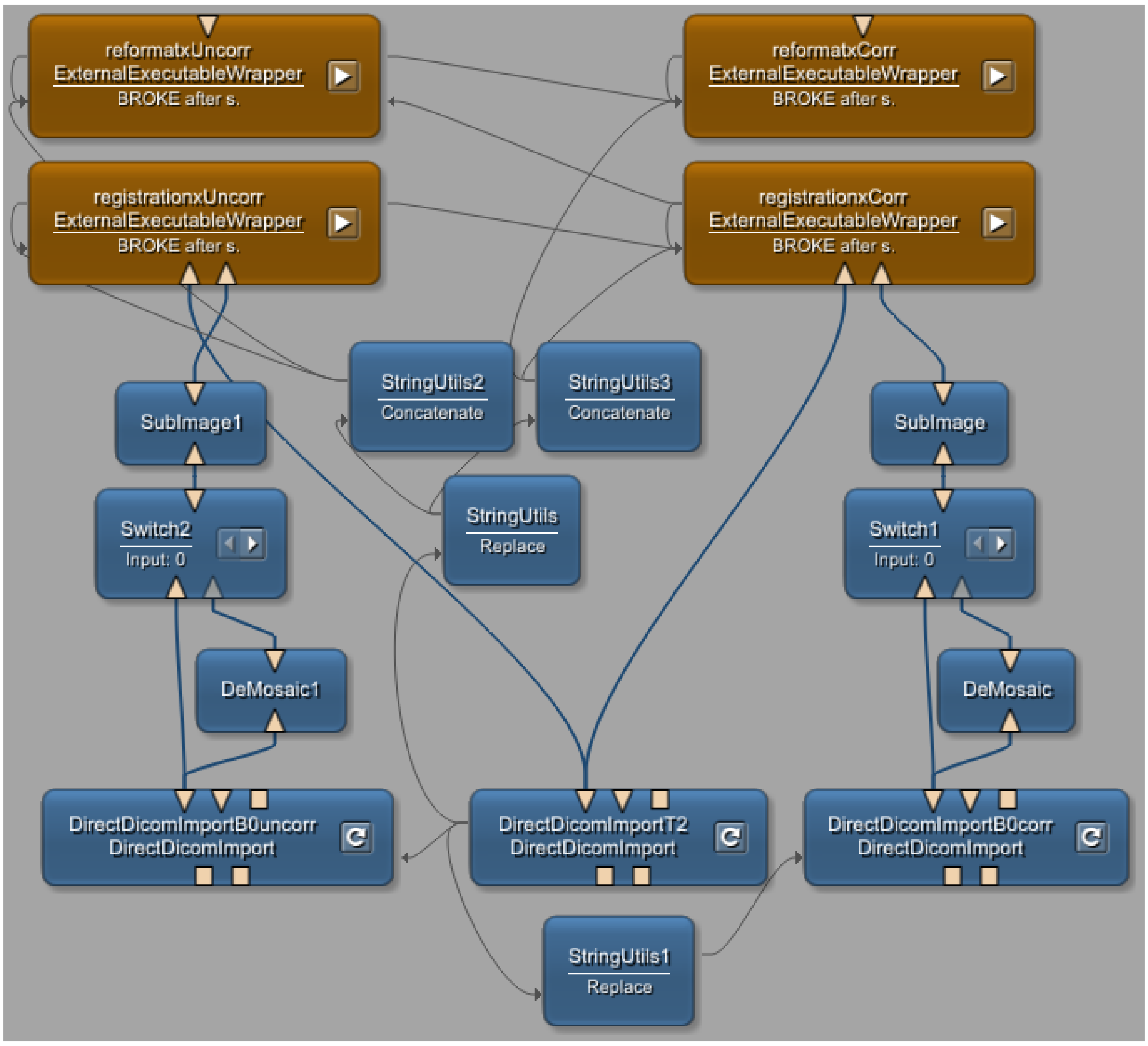


Figure 3.12: The network built for performing co-registration. Uncorrected and corrected forward phase encoded b_0 images, in addition to a T_2-weighted image, are loaded by the DirectDicomImport modules. Co-registration is performed by the CMTK functions `registrationx` and `reformatx`, integrated into the network by means of ExternalExecutableWrapper modules.

To evaluate the user-friendliness of the prototype, it was tested by a radiographer and a medical physicist not involved in the project. However, problems occurred when trying to install an updated version of the prototype in syngo.via Frontier, so the testing had to be performed in FrontierHost in MeVisLab. The test subjects tried to correct transverse diffusion-weighted images from a prostate patient, and evaluated the prototype by filling out the feedback form attached in Appendix B.

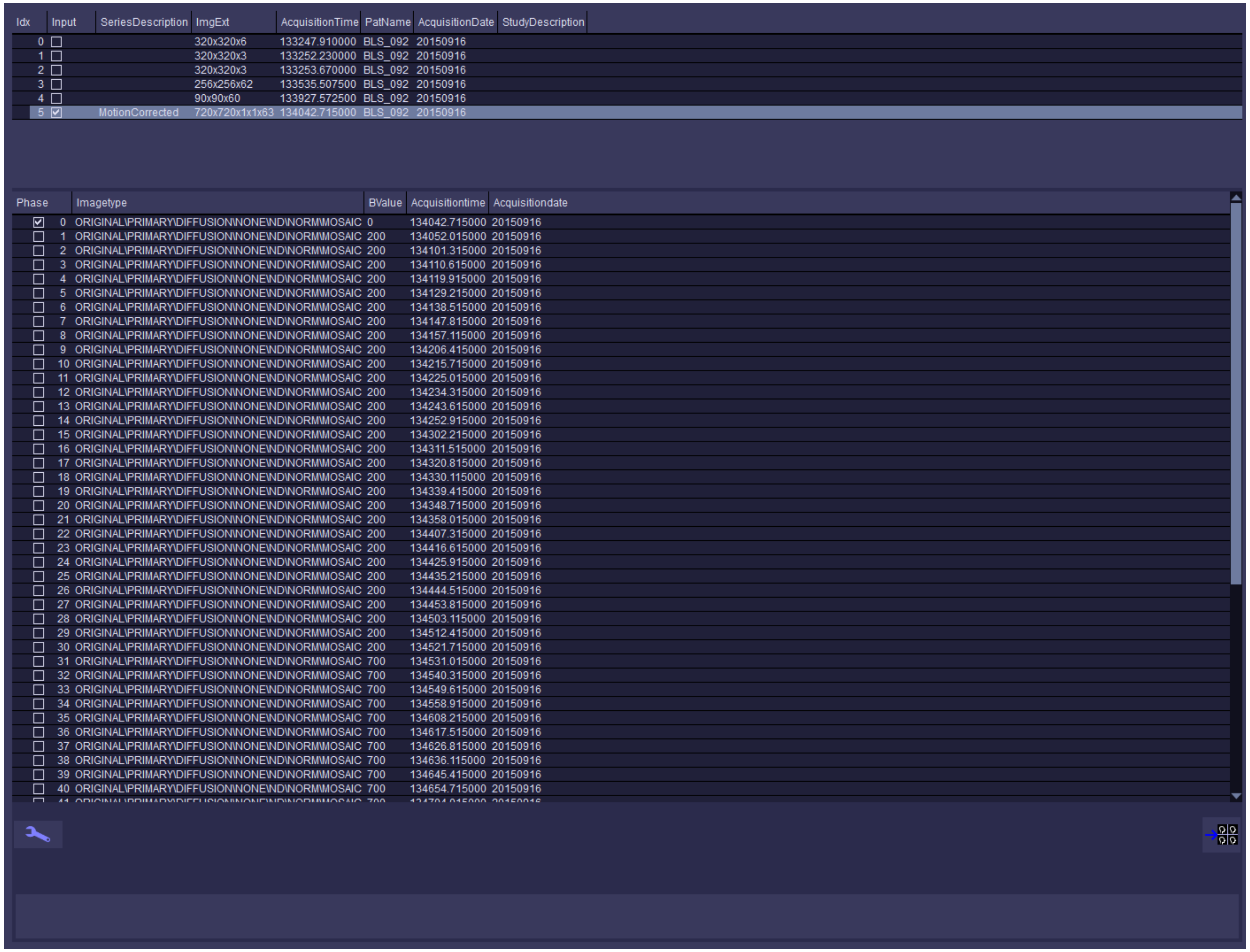
Chapter 4

Results

4.1 The prototype

The browser where the user can select the images to be corrected is shown in Figure 4.1. The image series can be selected in the top of the screen, and images in the series can be selected in the bottom of the screen. Information to the user about which images to select when are given in pop-up windows (Figure 4.2). When all images are selected, the user enters the main window of the prototype, shown in Figure 4.3. Here, the uncorrected image is shown in the top row, and the corrected image will be shown in the bottom row. On the left of the screen there is a menu for performing the distortion correction. The tabs of the menu, described in Section 3.1.7, are shown in Figure 4.4.

To correct the images, the user should first read the information tab, before moving on to the parameter selection and make sure that the correct parameters are specified. Then, the distortion correction calculation is performed by pressing the button in the "Calculate distortion correction" tab. When this has finished and the corrected forward/reverse images appear in the bottom row of the screen, the user can move on to correct the diffusion-weighted images by clicking the button in the next tab. Which images to be rendered on the screen (forward or reverse b_0, or diffusion-weighted) can be chosen in the "Choose image in top/bottom panel" tabs. At last, when all images are corrected, the user can save the resulting images in the "Save results" tab.

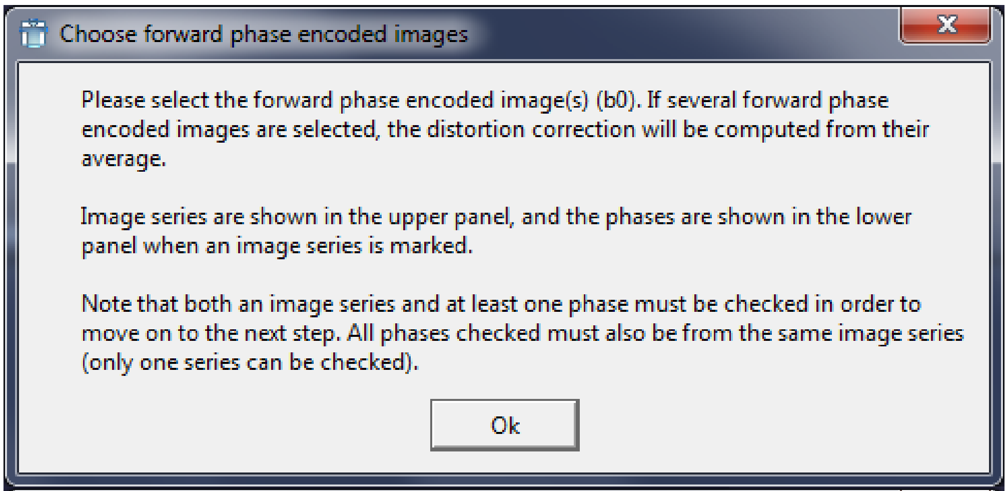


Idx	Input	SeriesDescription	ImgExt	AcquisitionTime	PatName	AcquisitionDate	StudyDescription
0	☐		320x320x6	133247.910000	BLS_092	20150916	
1	☐		320x320x3	133252.230000	BLS_092	20150916	
2	☐		320x320x3	133253.670000	BLS_092	20150916	
3	☐		256x256x62	133535.507500	BLS_092	20150916	
4	☐		90x90x60	133927.572500	BLS_092	20150916	
5	✓	MotionCorrected	720x720x1x1x63	134042.715000	BLS_092	20150916	

Phase		Imagetype	BValue	Acquisitiontime	Acquisitiondate
✓	0	ORIGINAL\PRIMARY\DIFFUSION\NONE\ND\NORM\MOSAIC	0	134042.715000	20150916
☐	1	ORIGINAL\PRIMARY\DIFFUSION\NONE\ND\NORM\MOSAIC	200	134052.015000	20150916
☐	2	ORIGINAL\PRIMARY\DIFFUSION\NONE\ND\NORM\MOSAIC	200	134101.315000	20150916
☐	3	ORIGINAL\PRIMARY\DIFFUSION\NONE\ND\NORM\MOSAIC	200	134110.615000	20150916
☐	4	ORIGINAL\PRIMARY\DIFFUSION\NONE\ND\NORM\MOSAIC	200	134119.915000	20150916
☐	5	ORIGINAL\PRIMARY\DIFFUSION\NONE\ND\NORM\MOSAIC	200	134129.215000	20150916
☐	6	ORIGINAL\PRIMARY\DIFFUSION\NONE\ND\NORM\MOSAIC	200	134138.515000	20150916
☐	7	ORIGINAL\PRIMARY\DIFFUSION\NONE\ND\NORM\MOSAIC	200	134147.815000	20150916
☐	8	ORIGINAL\PRIMARY\DIFFUSION\NONE\ND\NORM\MOSAIC	200	134157.115000	20150916
☐	9	ORIGINAL\PRIMARY\DIFFUSION\NONE\ND\NORM\MOSAIC	200	134206.415000	20150916
☐	10	ORIGINAL\PRIMARY\DIFFUSION\NONE\ND\NORM\MOSAIC	200	134215.715000	20150916
☐	11	ORIGINAL\PRIMARY\DIFFUSION\NONE\ND\NORM\MOSAIC	200	134225.015000	20150916
☐	12	ORIGINAL\PRIMARY\DIFFUSION\NONE\ND\NORM\MOSAIC	200	134234.315000	20150916
☐	13	ORIGINAL\PRIMARY\DIFFUSION\NONE\ND\NORM\MOSAIC	200	134243.615000	20150916
☐	14	ORIGINAL\PRIMARY\DIFFUSION\NONE\ND\NORM\MOSAIC	200	134252.915000	20150916
☐	15	ORIGINAL\PRIMARY\DIFFUSION\NONE\ND\NORM\MOSAIC	200	134302.215000	20150916
☐	16	ORIGINAL\PRIMARY\DIFFUSION\NONE\ND\NORM\MOSAIC	200	134311.515000	20150916
☐	17	ORIGINAL\PRIMARY\DIFFUSION\NONE\ND\NORM\MOSAIC	200	134320.815000	20150916
☐	18	ORIGINAL\PRIMARY\DIFFUSION\NONE\ND\NORM\MOSAIC	200	134330.115000	20150916
☐	19	ORIGINAL\PRIMARY\DIFFUSION\NONE\ND\NORM\MOSAIC	200	134339.415000	20150916
☐	20	ORIGINAL\PRIMARY\DIFFUSION\NONE\ND\NORM\MOSAIC	200	134348.715000	20150916
☐	21	ORIGINAL\PRIMARY\DIFFUSION\NONE\ND\NORM\MOSAIC	200	134358.015000	20150916
☐	22	ORIGINAL\PRIMARY\DIFFUSION\NONE\ND\NORM\MOSAIC	200	134407.315000	20150916
☐	23	ORIGINAL\PRIMARY\DIFFUSION\NONE\ND\NORM\MOSAIC	200	134416.615000	20150916
☐	24	ORIGINAL\PRIMARY\DIFFUSION\NONE\ND\NORM\MOSAIC	200	134425.915000	20150916
☐	25	ORIGINAL\PRIMARY\DIFFUSION\NONE\ND\NORM\MOSAIC	200	134435.215000	20150916
☐	26	ORIGINAL\PRIMARY\DIFFUSION\NONE\ND\NORM\MOSAIC	200	134444.515000	20150916
☐	27	ORIGINAL\PRIMARY\DIFFUSION\NONE\ND\NORM\MOSAIC	200	134453.815000	20150916
☐	28	ORIGINAL\PRIMARY\DIFFUSION\NONE\ND\NORM\MOSAIC	200	134503.115000	20150916
☐	29	ORIGINAL\PRIMARY\DIFFUSION\NONE\ND\NORM\MOSAIC	200	134512.415000	20150916
☐	30	ORIGINAL\PRIMARY\DIFFUSION\NONE\ND\NORM\MOSAIC	200	134521.715000	20150916
☐	31	ORIGINAL\PRIMARY\DIFFUSION\NONE\ND\NORM\MOSAIC	700	134531.015000	20150916
☐	32	ORIGINAL\PRIMARY\DIFFUSION\NONE\ND\NORM\MOSAIC	700	134540.315000	20150916
☐	33	ORIGINAL\PRIMARY\DIFFUSION\NONE\ND\NORM\MOSAIC	700	134549.615000	20150916
☐	34	ORIGINAL\PRIMARY\DIFFUSION\NONE\ND\NORM\MOSAIC	700	134558.915000	20150916
☐	35	ORIGINAL\PRIMARY\DIFFUSION\NONE\ND\NORM\MOSAIC	700	134608.215000	20150916
☐	36	ORIGINAL\PRIMARY\DIFFUSION\NONE\ND\NORM\MOSAIC	700	134617.515000	20150916
☐	37	ORIGINAL\PRIMARY\DIFFUSION\NONE\ND\NORM\MOSAIC	700	134626.815000	20150916
☐	38	ORIGINAL\PRIMARY\DIFFUSION\NONE\ND\NORM\MOSAIC	700	134636.115000	20150916
☐	39	ORIGINAL\PRIMARY\DIFFUSION\NONE\ND\NORM\MOSAIC	700	134645.415000	20150916
☐	40	ORIGINAL\PRIMARY\DIFFUSION\NONE\ND\NORM\MOSAIC	700	134654.715000	20150916

Figure 4.1: The browser for selecting images. Image series can be selected in the top panel, and images in the series can be selected in the bottom panel. To move on to the next step, the button on the bottom right of the screen must be clicked.

(a)



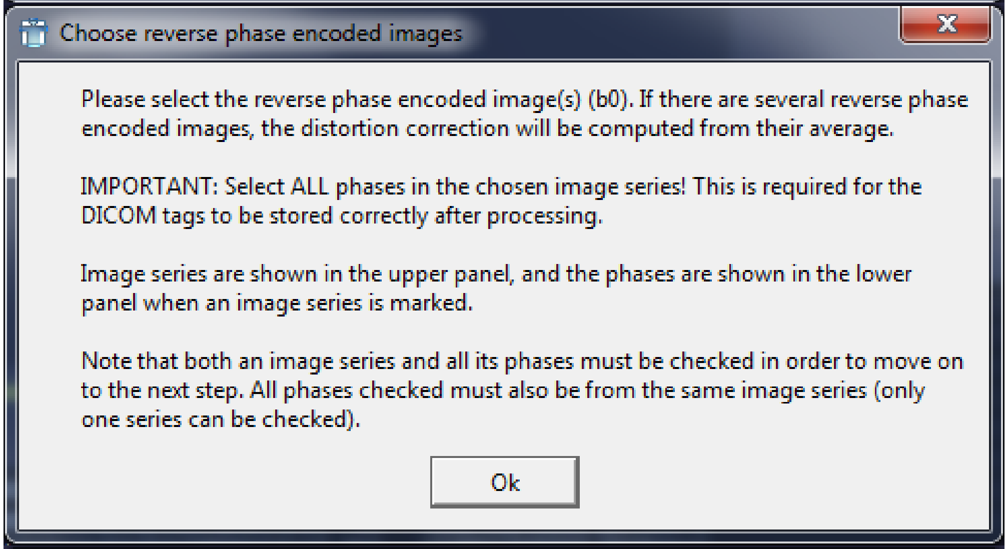
Choose forward phase encoded images

Please select the forward phase encoded image(s) (b0). If several forward phase encoded images are selected, the distortion correction will be computed from their average.

Image series are shown in the upper panel, and the phases are shown in the lower panel when an image series is marked.

Note that both an image series and at least one phase must be checked in order to move on to the next step. All phases checked must also be from the same image series (only one series can be checked).

(b)



Choose reverse phase encoded images

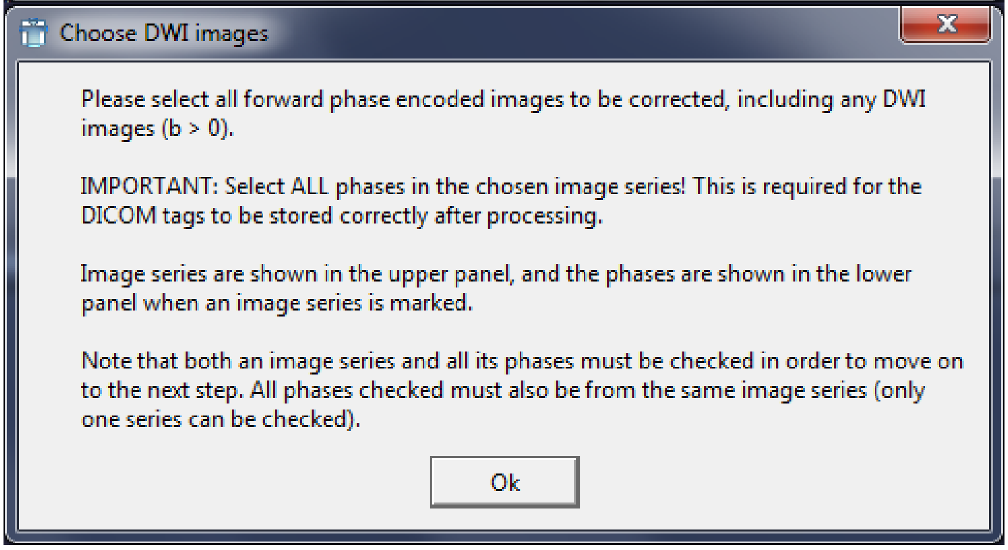
Please select the reverse phase encoded image(s) (b0). If there are several reverse phase encoded images, the distortion correction will be computed from their average.

IMPORTANT: Select ALL phases in the chosen image series! This is required for the DICOM tags to be stored correctly after processing.

Image series are shown in the upper panel, and the phases are shown in the lower panel when an image series is marked.

Note that both an image series and all its phases must be checked in order to move on to the next step. All phases checked must also be from the same image series (only one series can be checked).

(c)



Choose DWI images

Please select all forward phase encoded images to be corrected, including any DWI images (b > 0).

IMPORTANT: Select ALL phases in the chosen image series! This is required for the DICOM tags to be stored correctly after processing.

Image series are shown in the upper panel, and the phases are shown in the lower panel when an image series is marked.

Note that both an image series and all its phases must be checked in order to move on to the next step. All phases checked must also be from the same image series (only one series can be checked).

Figure 4.2: Pop-up windows shown to the user before selecting images for the prototype. Information is given about how and when to select **(a)** forward phase encoded b_0 images, **(b)** reverse phase encoded b_0 images, and **(c)** diffusion-weighted images.

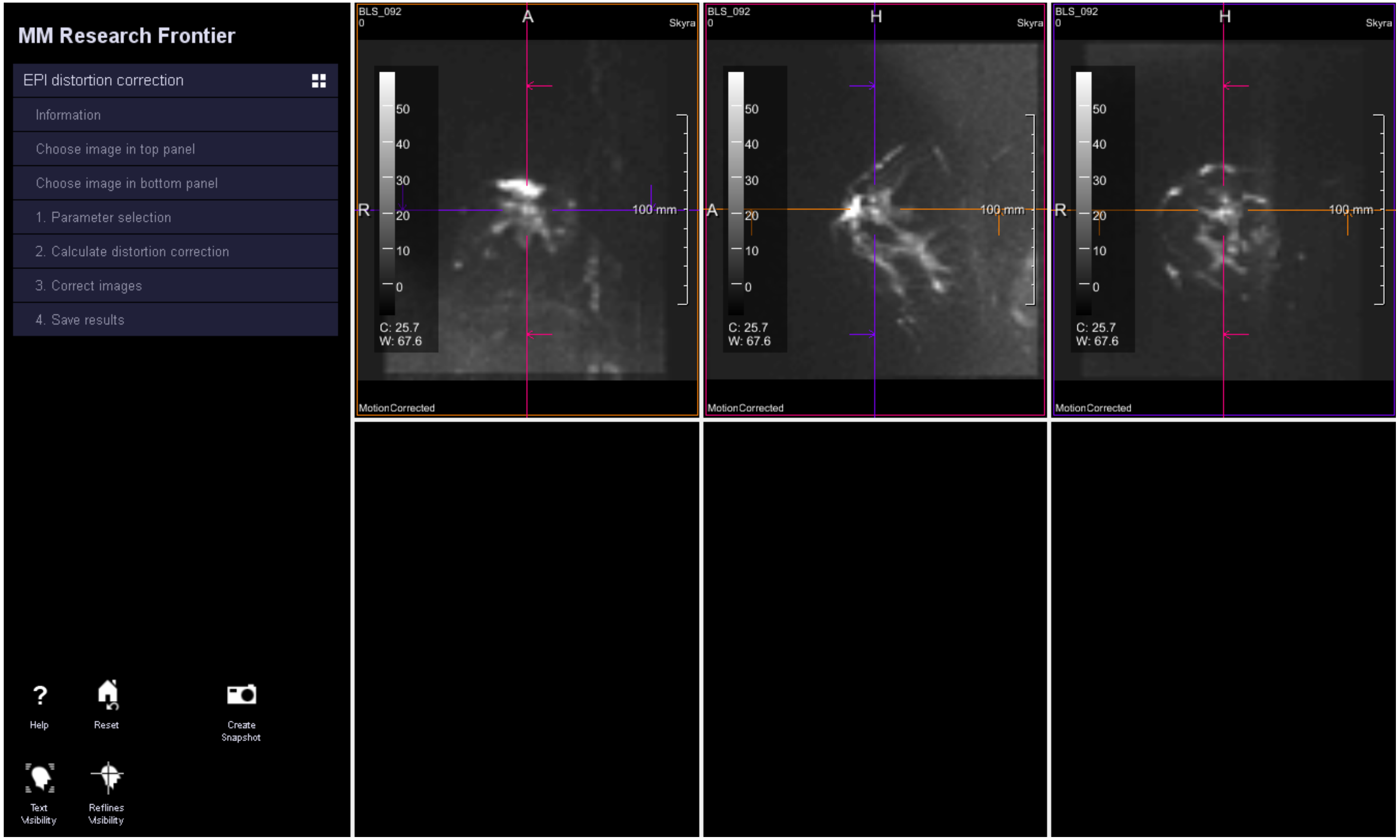
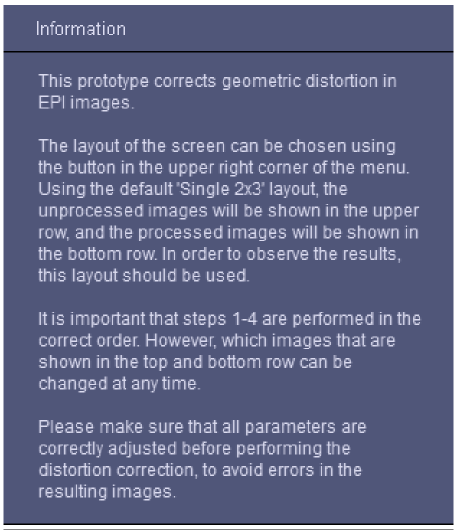


Figure 4.3: The interface of the main window of the prototype. The uncorrected image is shown in the top row, and the corrected image will be shown in the bottom row (transverse, sagittal and coronal view). The menu for performing the distortion correction is on the left of the screen.

(a)



Information

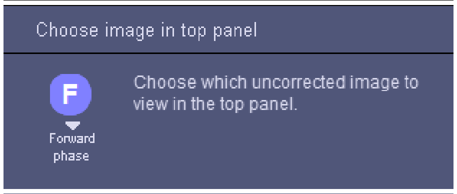
This prototype corrects geometric distortion in EPI images.

The layout of the screen can be chosen using the button in the upper right corner of the menu. Using the default 'Single 2x3' layout, the unprocessed images will be shown in the upper row, and the processed images will be shown in the bottom row. In order to observe the results, this layout should be used.

It is important that steps 1-4 are performed in the correct order. However, which images that are shown in the top and bottom row can be changed at any time.

Please make sure that all parameters are correctly adjusted before performing the distortion correction, to avoid errors in the resulting images.

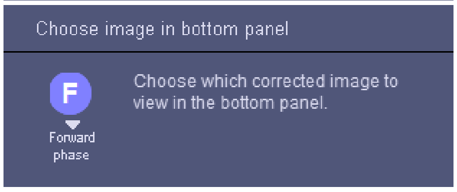
(b)



Choose image in top panel

Choose which uncorrected image to view in the top panel.

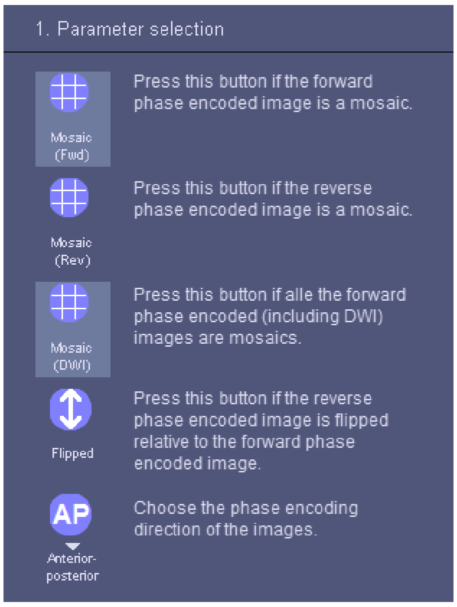
(c)



Choose image in bottom panel

Choose which corrected image to view in the bottom panel.

(d)



1. Parameter selection

Press this button if the forward phase encoded image is a mosaic.

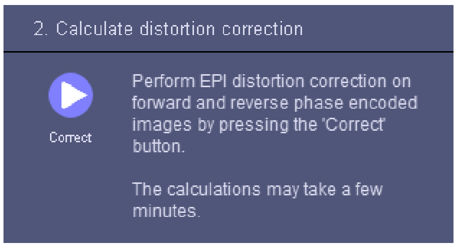
Press this button if the reverse phase encoded image is a mosaic.

Press this button if alle the forward phase encoded (including DWI) images are mosaics.

Press this button if the reverse phase encoded image is flipped relative to the forward phase encoded image.

Choose the phase encoding direction of the images.

(e)

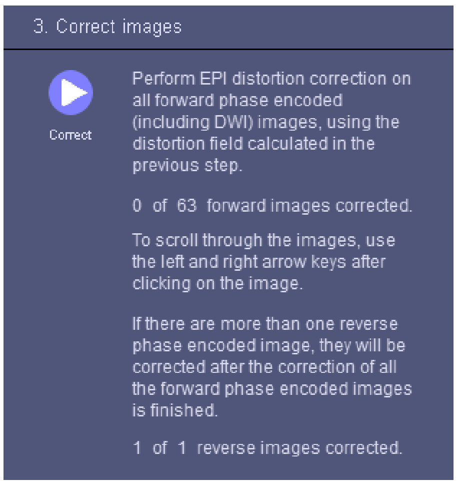


2. Calculate distortion correction

Perform EPI distortion correction on forward and reverse phase encoded images by pressing the 'Correct' button.

The calculations may take a few minutes.

(f)



3. Correct images

Perform EPI distortion correction on all forward phase encoded (including DWI) images, using the distortion field calculated in the previous step.

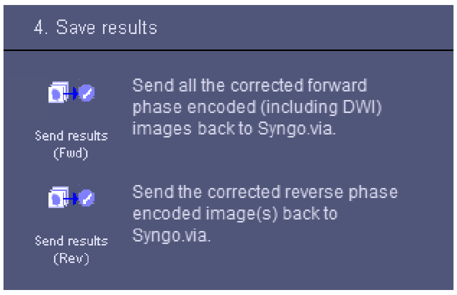
0 of 63 forward images corrected.

To scroll through the images, use the left and right arrow keys after clicking on the image.

If there are more than one reverse phase encoded image, they will be corrected after the correction of all the forward phase encoded images is finished.

1 of 1 reverse images corrected.

(g)

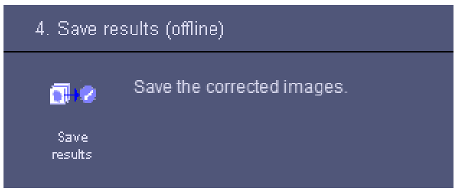


4. Save results

Send all the corrected forward phase encoded (including DWI) images back to Syngo.via.

Send the corrected reverse phase encoded image(s) back to Syngo.via.

(h)



4. Save results (offline)

Save the corrected images.

Figure 4.4: Tabs of the sidebar menu. **(a)** Information to the user about how to perform the distortion correction. **(b)**, **(c)** Let the user select which uncorrected and corrected image to be rendered in the top and bottom row of the screen, respectively. **(d)** Parameters that the user can select for the distortion correction. **(e)** The user can calculate the distortion correction from the forward and reverse phase encoded b_0 images. **(f)** The user can correct the other images. **(g)** The user can send the results back to the syngo.via server. **(h)** Additional step for saving the corrected images locally when testing the prototype (this step is not included in the final, installed version of the prototype).

4.2 Testing the prototype

When testing the finished prototype in FrontierHost, it worked as intended and it was able to perform the distortion correction and save the corrected images locally.

4.2.1 Correcting images

For all three image types, none of the distortion corrections failed. This gives a robustness of 100%. By visual assessment of the images, the quality of correction was found to be successful for 12 of 13 breast patients. For the last breast patient, the corrected images appeared to be the midway between the forward and reverse phase encoded images, but there was too much noise in the images to determine whether the result was anatomically correct. For both the coronal and transverse prostate images, the quality of correction was found to be successful for all patients. Examples of uncorrected and corrected forward and reverse phase encoded b_0 images from breast, prostate (coronal) and prostate (transverse) can be seen in Figures 4.5, 4.6 and 4.7, respectively. All images are from a central slice showing the breast or prostate. The mean relative change in MSD between forward and reverse phase encoded b_0 images from before to after distortion correction for the three image types, with p-values, is shown in Table 4.1, and a box plot showing the corresponding distribution of relative change in MSD for all patients can be seen in Figure 4.8.

For the breast images, the calculation of the distortion correction took approximately 4 to 7 minutes. The correction of all the diffusion-weighted images took approximately 5 to 10 minutes, and the time from opening the prototype to have finished saving all the images took approximately 15 to 20 minutes. For the coronal prostate images, these times were 3-4 minutes, 30-40 seconds and 6-7 minutes. For the transverse prostate images, they were 2-3 minutes, 20-30 seconds and 4-5 minutes. Note that these numbers are very approximate and are only meant to give the order of magnitude.

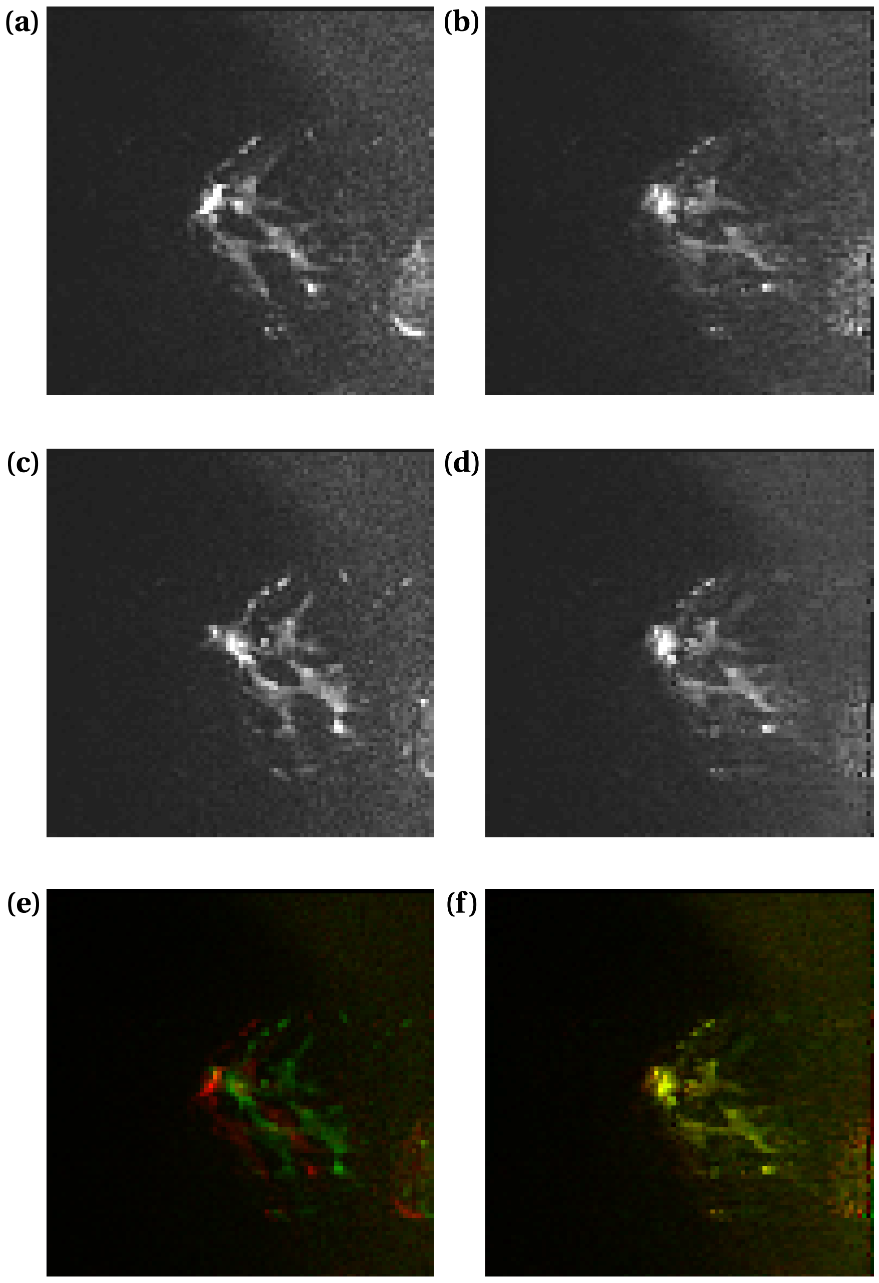


Figure 4.5: Uncorrected and corrected forward and reverse phase encoded b_0 images from a breast patient, sagittal view. **(a)**, **(b)** Uncorrected and corrected forward phase encoded b_0 image, respectively. Phase encoding direction is anterior-posterior (which is left-right when looking at the images), and distortions can therefore be seen along this direction. **(c)**, **(d)** Uncorrected and corrected reverse phase encoded b_0 image, respectively. The corrected images are the midway between the two uncorrected images, and the natural shape of the breast is restored. **(e)** Overlay of the forward and reverse uncorrected images. The forward image is coloured red, while the reverse image is coloured green. For closely matched pixel intensities, the colour will become yellow. **(f)** Overlay of the forward and reverse corrected images. The yellow colour shows that the images match to a large extent. The relative change in MSD in this case was −86.48%.

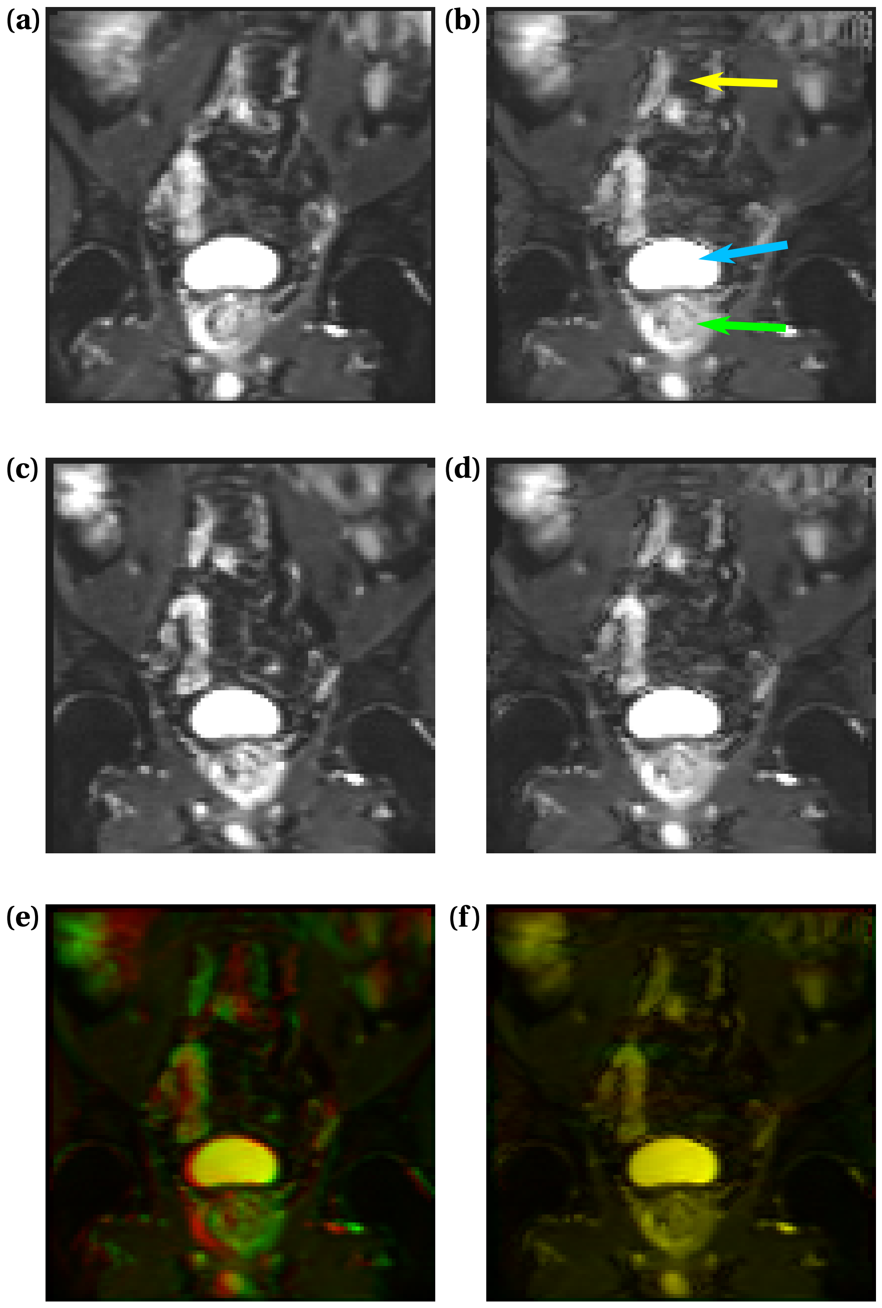


Figure 4.6: Uncorrected and corrected forward and reverse phase encoded b_0 images from a prostate patient, coronal view. **(a)**, **(b)** Uncorrected and corrected forward phase encoded b_0 image, respectively. Phase encoding direction is right-left, and distortions can therefore be seen along this direction. The yellow arrow indicates the spine, the blue arrow indicates the bladder, and the green arrow indicates the prostate. **(c)**, **(d)** Uncorrected and corrected reverse phase encoded b_0 image, respectively. The corrected images are the midway between the two uncorrected images, and distortions in the prostate area as well as skewness of the spine have been corrected. **(e)** Overlay of the forward and reverse uncorrected images. The forward image is coloured red, while the reverse image is coloured green. For closely matched pixel intensities, the colour will become yellow. **(f)** Overlay of the forward and reverse corrected images. The yellow colour shows that the images match to a large extent. The relative change in MSD in this case was −92.11%.

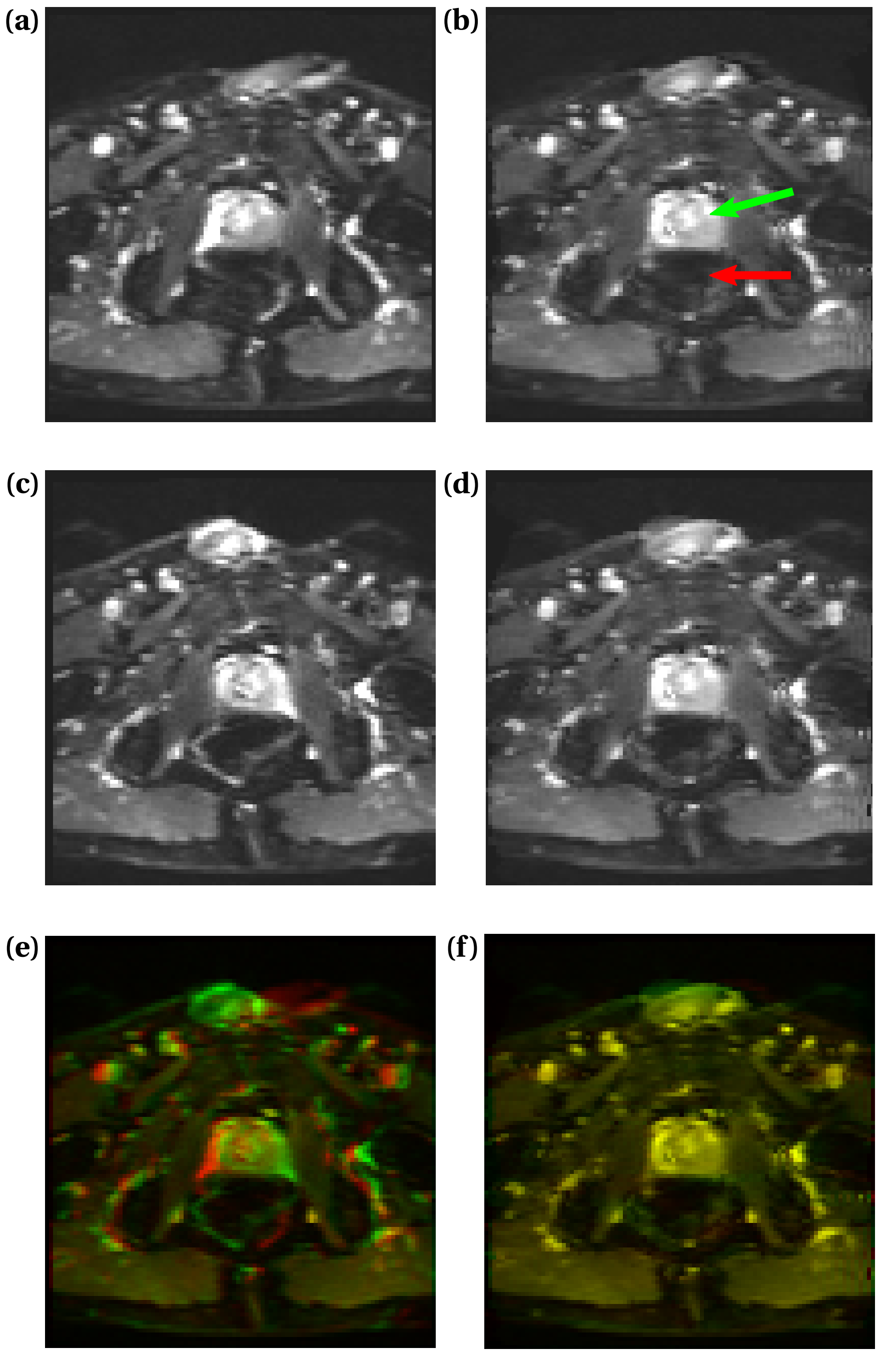


Figure 4.7: Uncorrected and corrected forward and reverse phase encoded b_0 images from a prostate patient, transverse view. **(a)**, **(b)** Uncorrected and corrected forward phase encoded b_0 image, respectively. Phase encoding direction is right-left, and distortions can therefore be seen along this direction. The green arrow indicates the prostate, and the red arrow indicates the rectum. **(c)**, **(d)** Uncorrected and corrected reverse phase encoded b_0 image, respectively. Note that these show some ghosting artefacts in the upper part of the images. The corrected images are the midway between the two uncorrected images, and distortions in the prostate and rectal area have been corrected. **(e)** Overlay of the forward and reverse uncorrected images. The forward image is coloured red, while the reverse image is coloured green. For closely matched pixel intensities, the colour will become yellow. **(f)** Overlay of the forward and reverse corrected images. The yellow colour shows that the images match to a large extent. The relative change in MSD in this case was −93.29%.

Table 4.1: Mean relative change in MSD between forward and reverse phase encoded b_0 images from before to after distortion correction for breast and prostate (coronal and transverse) images. All three groups show a significant negative change. A relative change of −100% would indicate that the resulting forward and reverse images are identical.

Type of image	Mean relative change in MSD (%)	95% CI (%)	P-value
Breast	−86.96	[-89.86, -84.05]	0.0002
Prostate (coronal)	−87.69	[-91.05, -84.33]	0.0004
Prostate (transverse)	−90.94	[-92.77, -89.12]	0.0004

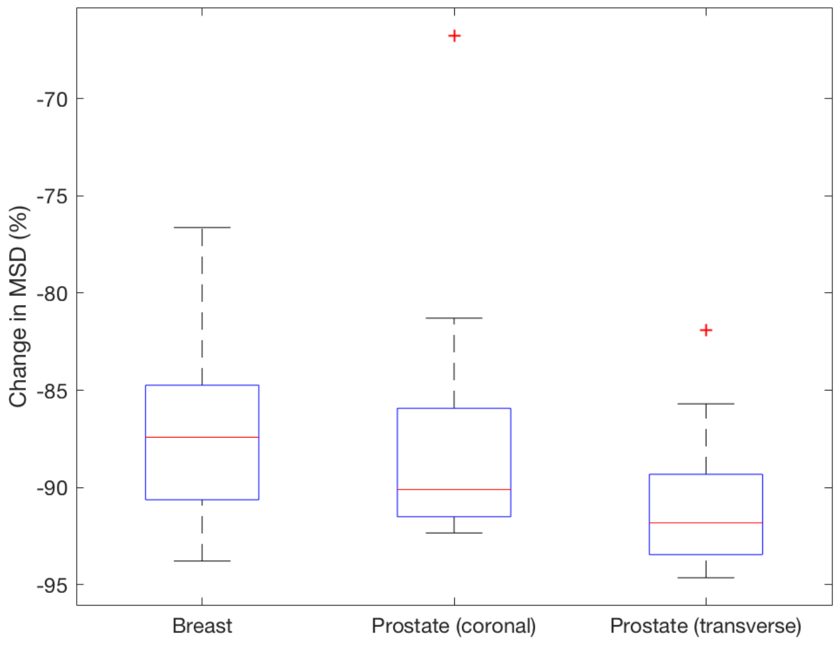


Figure 4.8: Box plot showing the distribution of relative change in MSD between forward and reverse phase encoded b_0 images from before to after distortion correction, for breast, prostate (coronal) and prostate (transverse) images. The red line inside the boxes indicates the median values, while the top and bottom sides of the box indicate the 75th and 25th percentile of the distribution, respectively [36]. The whiskers extend from the top and bottom of the box and reach all values outside the interquartile range. Outliers that are further away from the box than 1.5 times the interquartile range are not included by the whiskers, but are shown by a red + sign.

4.2.1.1 Co-registration

All forward b_0 images were successfully co-registered to the T_2-weighted images. Examples of breast b_0 images co-registered to a conventional T_2-weighted image are shown in Figure 4.9, and examples of breast b_0 images co-registered to a fat-suppressed T_2-weighted image are shown in Figure 4.10. Examples of coronal and transverse b_0 images co-registered to T_2-weighted images, full-size and cropped, respectively, are shown in Figures 4.11, 4.12, 4.13 and 4.14.

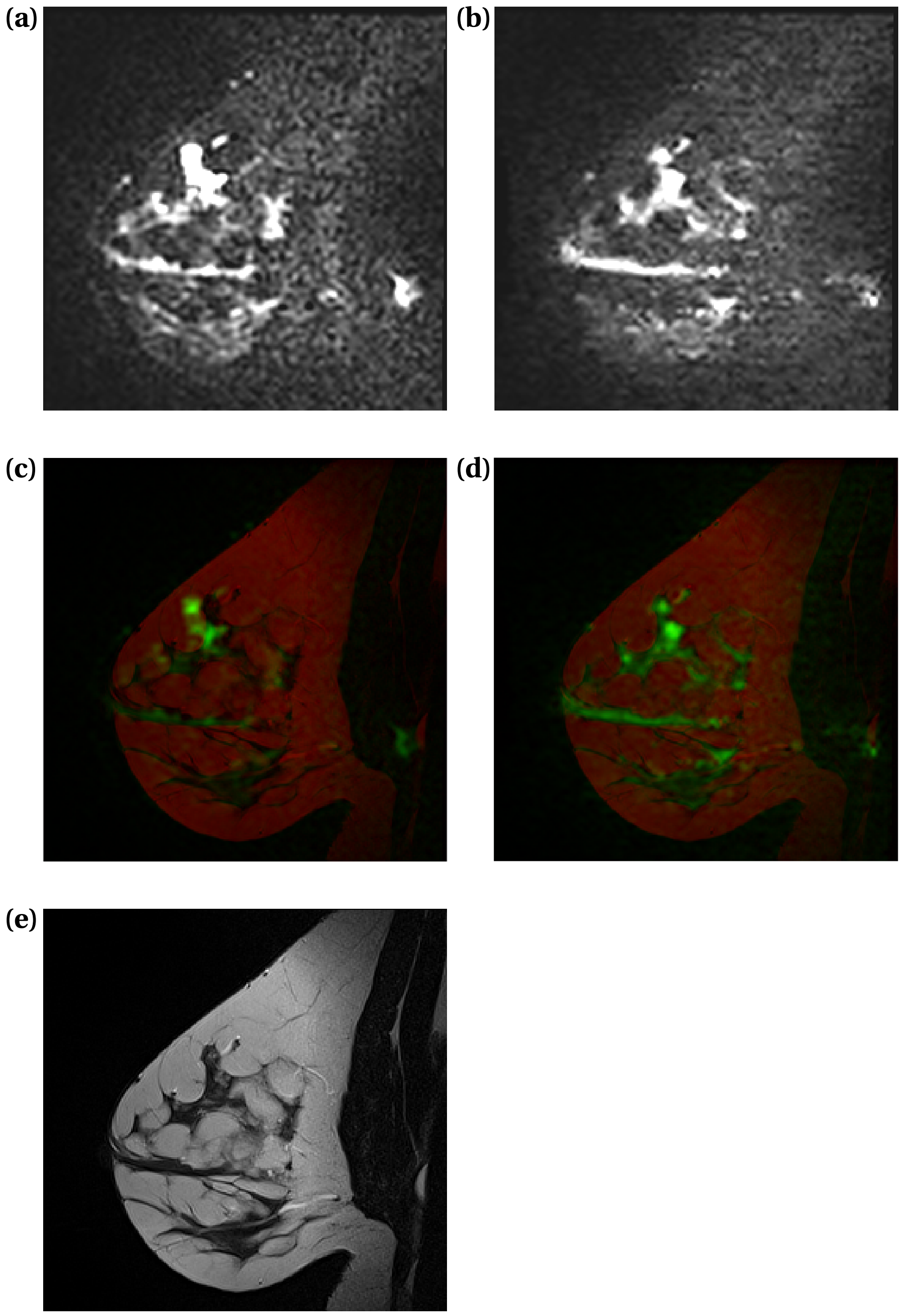


Figure 4.9: Breast (sagittal) b_0 images co-registered to a T_2-weighted image. **(a)**, **(b)** Uncorrected and corrected co-registered forward b_0 image, respectively. Phase encoding direction is anterior-posterior (which is left-right when looking at the images). **(c)**, **(d)** Overlay of the uncorrected and corrected b_0 image on the T_2-weighted image, respectively. The b_0 images are coloured green, and the T_2-weighted image is coloured red. The colour will become yellow for closely matched pixel intensities, but this is not achieved here because the contrasts of the b_0 and T_2-weighted images are different. However, the overlay of the corrected b_0 image on the T_2-weighted image is clearly a good match. In this case, the relative changes in NMI, NCC and MSD were 1.15%, 2.22% and −0.29%, respectively. **(e)** T_2-weighted image.

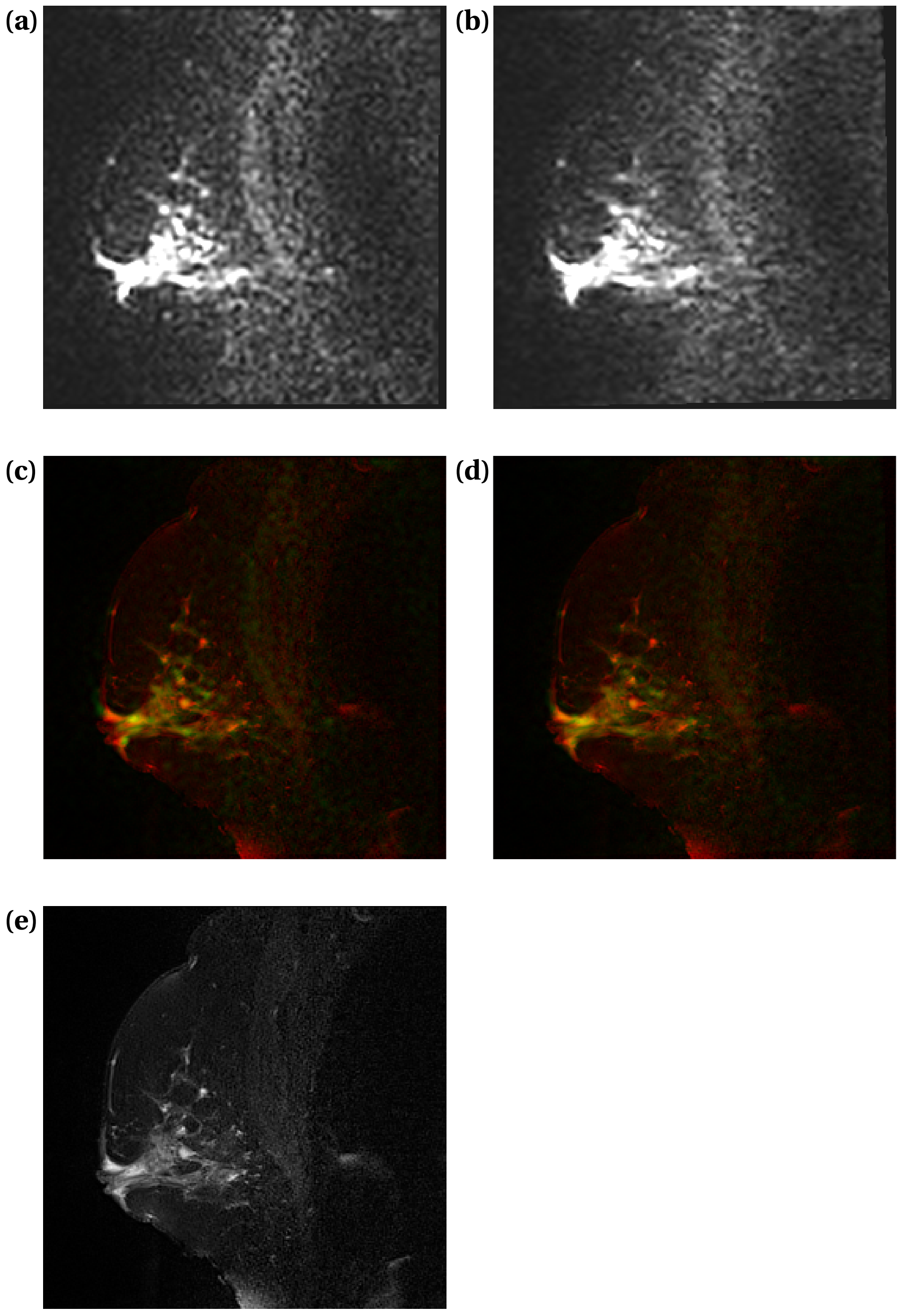


Figure 4.10: Breast (sagittal) b_0 images co-registered to a fat-suppressed T_2-weighted image. **(a)**, **(b)** Uncorrected and corrected co-registered forward b_0 image, respectively. Phase encoding direction is anterior-posterior (which is left-right when looking at the images). **(c)**, **(d)** Overlay of the uncorrected and corrected b_0 image on the T_2-weighted image, respectively. The b_0 images are coloured green, and the T_2-weighted image is coloured red. The colour will become yellow for closely matched pixel intensities. There is more yellow in the corrected overlay than the uncorrected one, and more green colour is visible in the uncorrected overlay, implying that the corrected image is a better match with the T_2-weighted image. In this case, the relative changes in NMI, NCC and MSD were 0.48%, 7.41% and −14.22%, respectively. **(e)** Fat-suppressed T_2-weighted image, which has a much more similar contrast to the b_0 images than conventional T_2-weighted images.

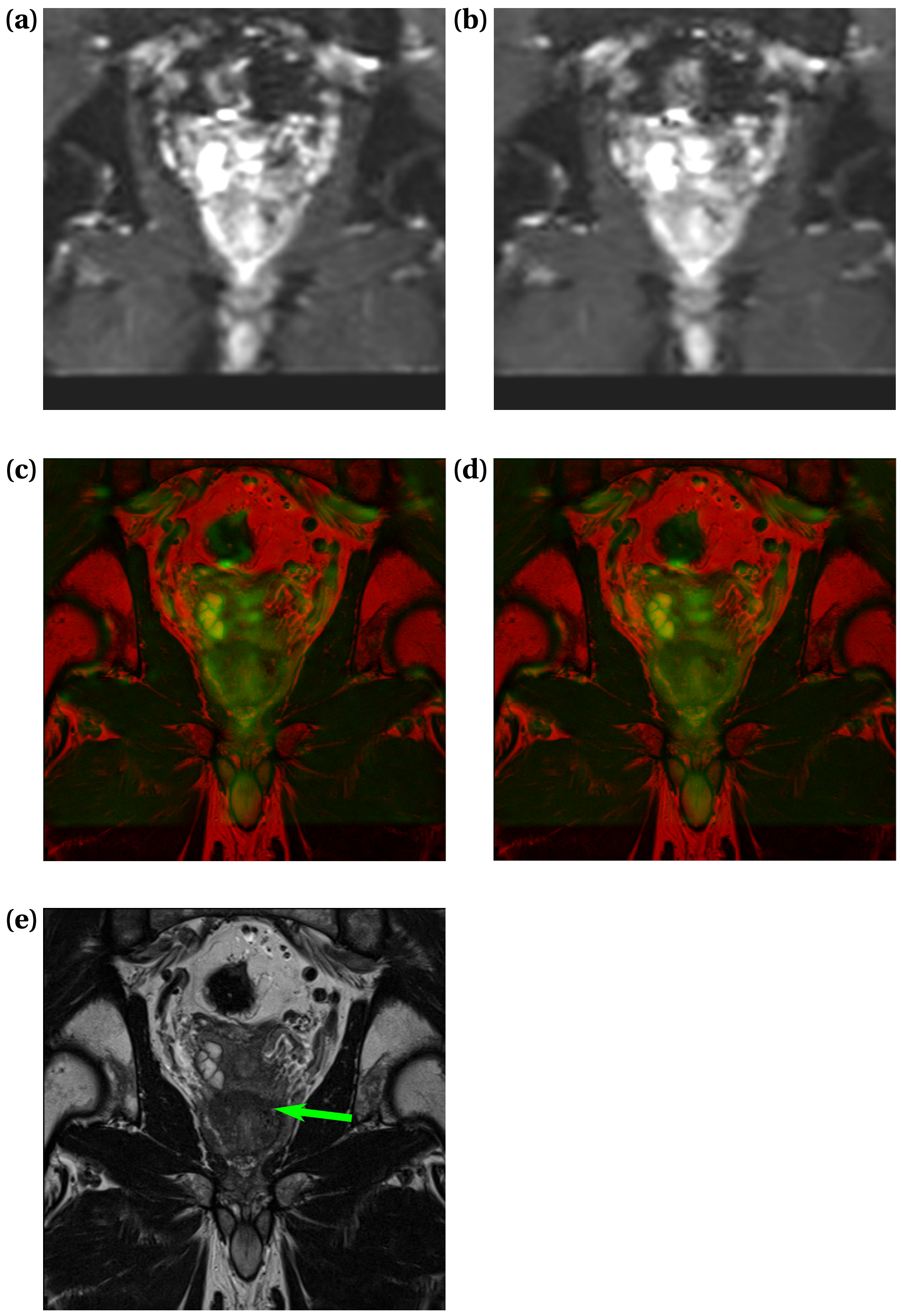


Figure 4.11: Prostate (coronal) b_0 images co-registered to a T_2-weighted image. **(a)**, **(b)** Uncorrected and corrected co-registered forward b_0 image, respectively. Phase encoding direction is right-left. **(c)**, **(d)** Overlay of the uncorrected and corrected b_0 image on the T_2-weighted image, respectively. The b_0 images are coloured green, and the T_2-weighted image is coloured red. The colour will become yellow for closely matched pixel intensities. There is not much yellow in the images because the contrasts of the b_0 and T_2-weighted images are so different, but there is a visible increase in the match of the prostate area and below it. In this case, the relative changes in NMI, NCC and MSD were 0.09%, 13.58% and −0.90%, respectively. **(e)** T_2-weighted image. The green arrow indicates the prostate.

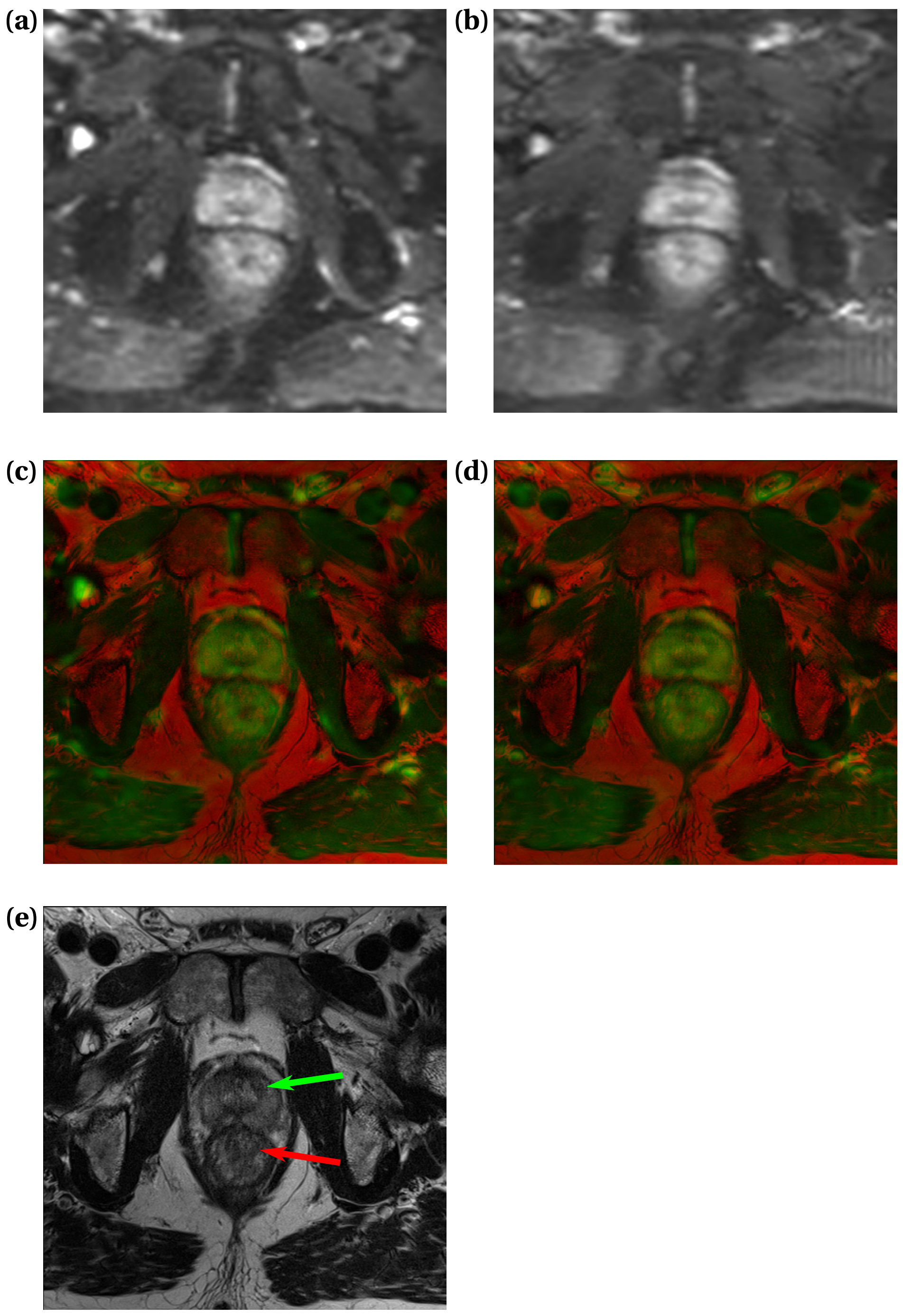


Figure 4.12: Prostate (transverse) b_0 images co-registered to a T_2-weighted image. **(a)**, **(b)** Uncorrected and corrected co-registered forward b_0 image, respectively. Phase encoding direction is right-left. **(c)**, **(d)** Overlay of the uncorrected and corrected b_0 image on the T_2-weighted image, respectively. The b_0 images are coloured green, and the T_2-weighted image is coloured red. The colour will become yellow for closely matched pixel intensities. There is not much yellow in the images because the contrasts of the b_0 and T_2-weighted images are so different, but there is a little more yellow in the corrected overlay than the uncorrected one. The skewness in the prostate and rectal area has improved, and it is a better match to the T_2-weighted image in the corrected case. In this case, the relative changes in NMI, NCC and MSD were 0.26%, 12.00% and −0.32%, respectively. **(e)** T_2-weighted image. The green arrow indicates the prostate, and the red arrow indicates the rectum.

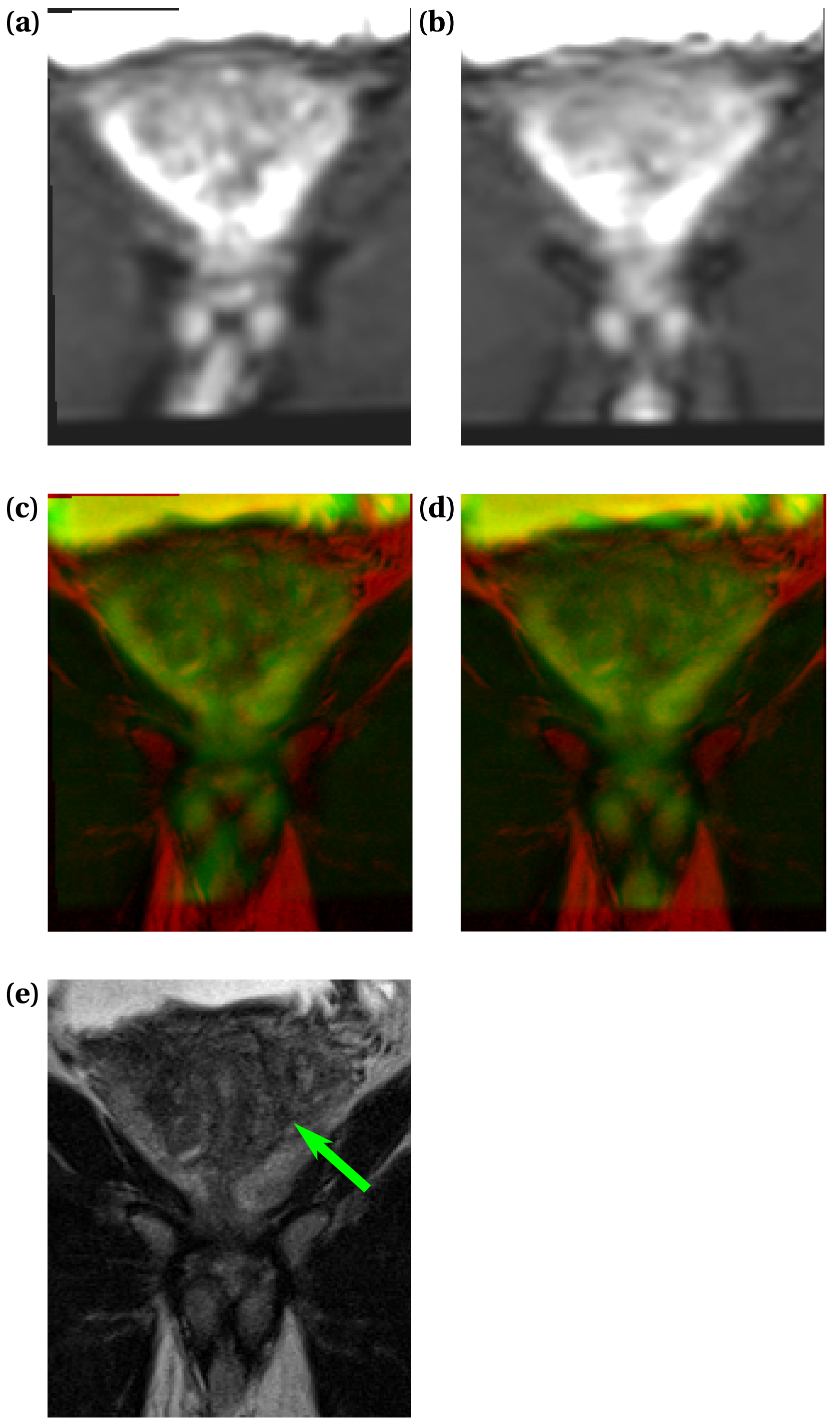


Figure 4.13: Cropped prostate (coronal) b_0 images co-registered to a T_2-weighted image. **(a)**, **(b)** Uncorrected and corrected co-registered forward b_0 image, respectively. Phase encoding direction is right-left. **(c)**, **(d)** Overlay of the uncorrected and corrected b_0 image on the T_2-weighted image, respectively. The b_0 images are coloured green, and the T_2-weighted image is coloured red. The colour will become yellow for closely matched pixel intensities. There is not much yellow in the images because the contrasts of the b_0 and T_2-weighted images are so different, but the bladder (above the prostate) shows a good match in both cases. The skewness in the prostate area and below it has improved, and it is a better match to the T_2-weighted image in the corrected case. In this case, the relative changes in NMI, NCC and MSD were 0.27%, 4.45% and −6.44%, respectively. **(e)** T_2-weighted image. The green arrow indicates the prostate.

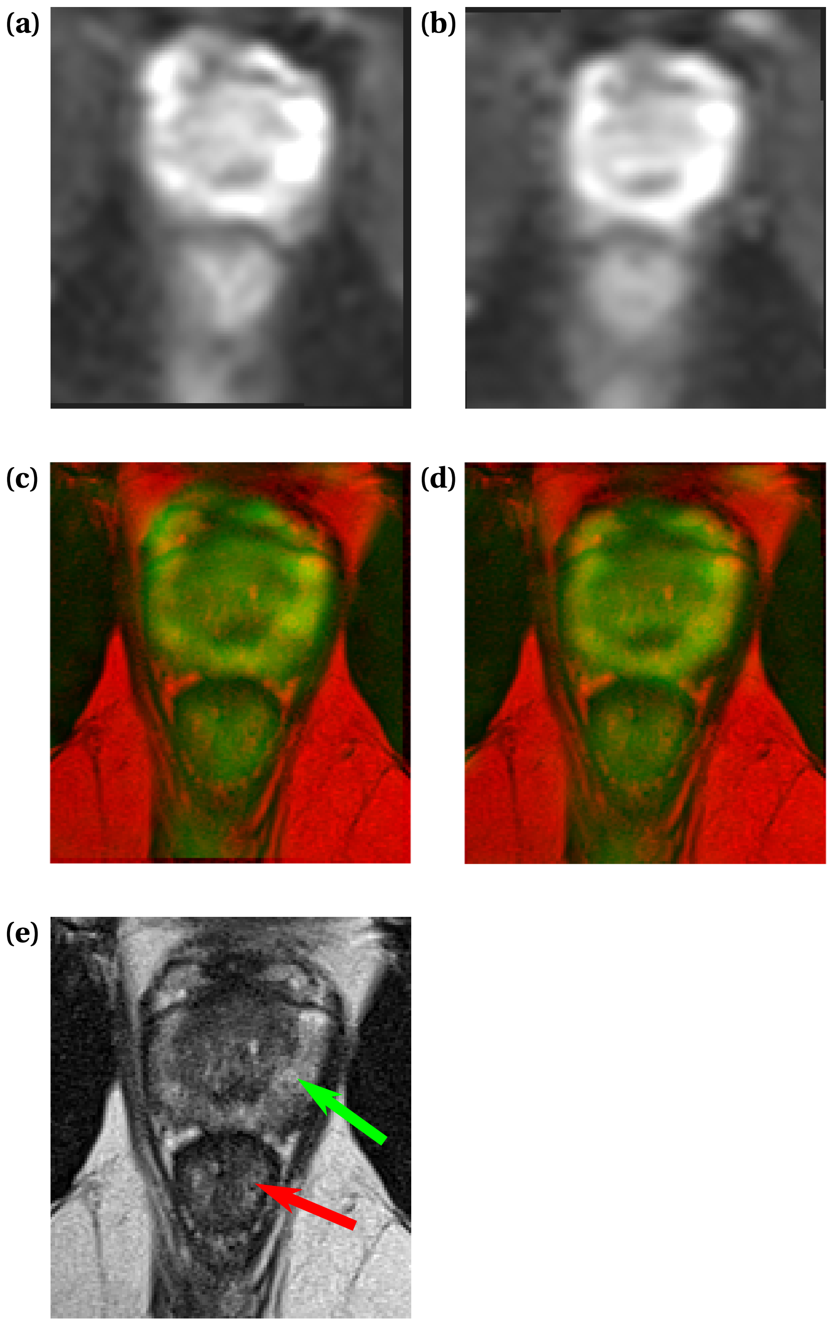


Figure 4.14: Cropped prostate (transverse) b_0 images co-registered to a T_2-weighted image. **(a)**, **(b)** Uncorrected and corrected co-registered forward b_0 image, respectively. Phase encoding direction is right-left. **(c)**, **(d)** Overlay of the uncorrected and corrected b_0 image on the T_2-weighted image, respectively. The b_0 images are coloured green, and the T_2-weighted image is coloured red. The colour will become yellow for closely matched pixel intensities. There is not much yellow in the images because the contrasts of the b_0 and T_2-weighted images are so different. However, the skewness in the prostate and rectal area, in addition to their shapes, have improved and they are a better match to the T_2-weighted image in the corrected case. In this case, the relative changes in NMI, NCC and MSD were 0.03%, 7.57% and −0.58%, respectively. **(e)** T_2-weighted image. The green arrow indicates the prostate, and the red arrow indicates the rectum.

The mean relative changes in NMI, NCC and MSD between forward phase encoded b_0 and T_2-weighted images from before to after distortion correction, with p-values, are shown in Tables 4.2, 4.3 and 4.4, respectively. Box plots showing the corresponding distributions of relative change in NMI, NCC and MSD for all patients can be seen in Figures 4.15, 4.16 and 4.17, respectively.

Table 4.2: Mean relative change in NMI between forward phase encoded b_0 and T_2-weighted images from before to after distortion correction for the different types of images. In theory, a successful correction should give a positive change in NMI. Only the breast group has a 95% CI enclosing only positive numbers, and a significant difference between NMI for uncorrected and corrected images. Co-reg. = co-registered, fat-supp. = fat-suppressed, rel. = relative.

Type of image	**Mean rel. change in NMI (%)**	**95% CI (%)**	**P-value**
Breast	0.42	[0.06, 0.78]	0.048
Breast (co-reg. to fat-supp. T_2)	0.36	[-0.07, 0.65]	0.118
Prostate (coronal)	0.01	[-0.08, 0.09]	0.918
Prostate (transverse)	0.06	[-0.02, 0.13]	0.121
Prostate, cropped (coronal)	0.08	[-0.05, 0.22]	0.196
Prostate, cropped (transverse)	−0.03	[-0.20, 0.15]	0.836

Table 4.3: Mean relative change in NCC between forward phase encoded b_0 and T_2-weighted images from before to after distortion correction for the different types of images. In theory, a successful correction should give a positive change in NCC. Breast (co-registered to fat-suppressed T_2), prostate (coronal) and cropped prostate (coronal) have a 95% CI only enclosing positive numbers. All groups have a significant difference between NCC for uncorrected and corrected images. However, for the transverse prostate groups, the relative change is negative, which is the opposite of what was expected. Co-reg. = co-registered, fat-supp. = fat-suppressed, rel. = relative.

Type of image	**Mean rel. change in NCC (%)**	**95% CI (%)**	**P-value**
Breast	4.06	[-0.19, 8.31]	0.033
Breast (co-reg. to fat-supp. T_2)	5.31	[3.45, 7.16]	0.001
Prostate (coronal)	8.62	[0.96, 16.28]	0.0004
Prostate (transverse)	−8.35	[-33.50, 16.79]	0.044
Prostate, cropped (coronal)	4.66	[0.51, 8.82]	0.011
Prostate, cropped (transverse)	−36.24	[-74.75, 2.27]	0.001

Table 4.4: Mean relative change in MSD between forward phase encoded b_0 and T_2-weighted images from before to after distortion correction for the different types of images. In theory, a successful correction should give a negative change in MSD. Breast (co-registered to fat-suppressed T_2), prostate (coronal) and cropped prostate (coronal) have a 95% CI only enclosing negative numbers. All groups have a significant difference between MSD for uncorrected and corrected images. However, for the transverse prostate groups, the relative change is positive, which is the opposite of what was expected. Co-reg. = co-registered, fat-supp. = fat-suppressed, rel. = relative.

Type of image	**Mean rel. change in MSD (%)**	**95% CI (%)**	**P-value**
Breast	−2.60	[-6.02, 0.82]	0.021
Breast (co-reg. to fat-supp. T_2)	−9.89	[-15.22, -4.57]	0.010
Prostate (coronal)	−1.05	[-1.42, -0.68]	0.0004
Prostate (transverse)	0.98	[0.17, 1.78]	0.026
Prostate, cropped (coronal)	−1.90	[-2.82, -0.99]	0.001
Prostate, cropped (transverse)	6.33	[3.00, 9.65]	0.002

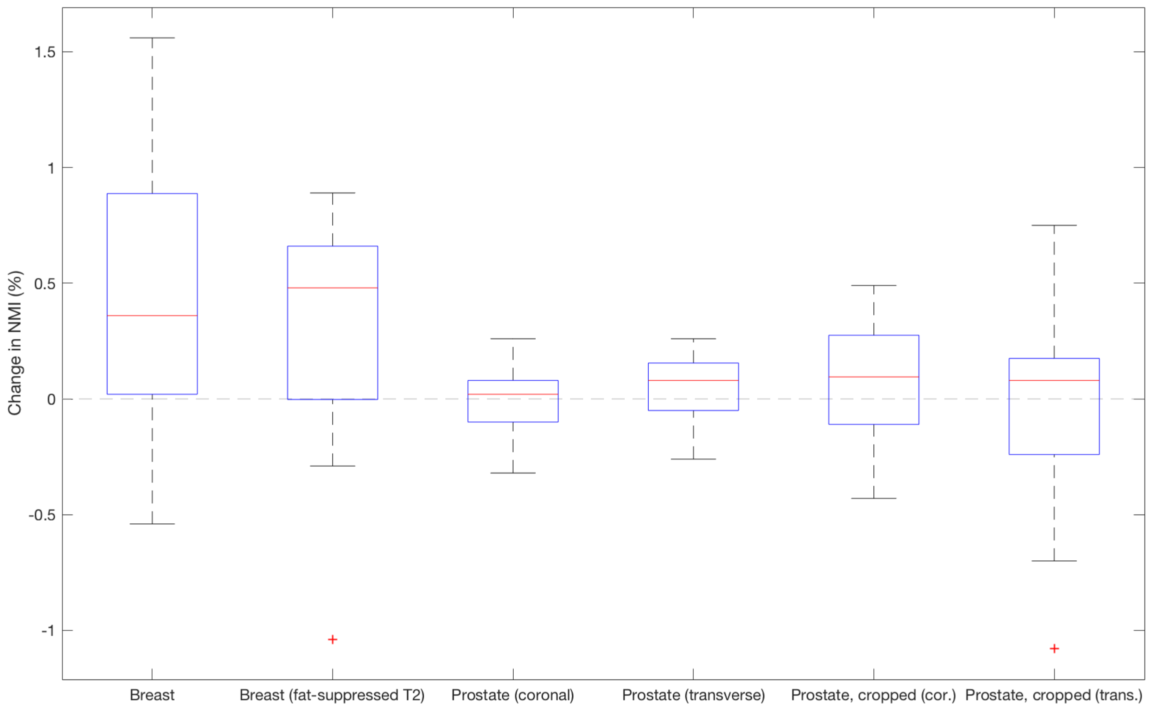


Figure 4.15: Box plot showing the distribution of relative change in NMI between forward phase encoded b_0 and T_2-weighted images from before to after distortion correction. In theory, a successful correction should give a positive change in NMI. For all groups there are both positive and negative values, although there for the two breast groups is a trend towards positive numbers. Only the breast group has a significant difference between NMI for uncorrected and corrected images. Cor. = coronal, trans. = transverse.

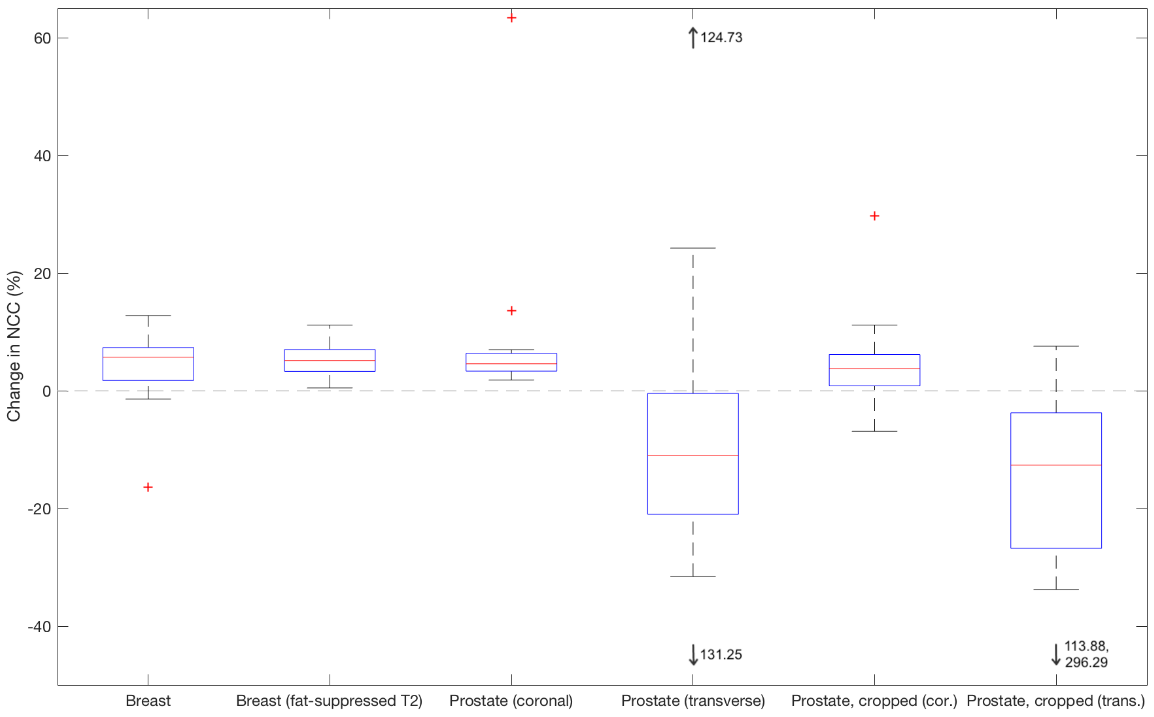


Figure 4.16: Box plot showing the distribution of relative change in NCC between forward phase encoded b_0 and T_2-weighted images from before to after distortion correction. The y-axis has been zoomed in to show the boxes more clearly, and extreme outliers are indicated by arrows and their values. In theory, a successful correction should give a positive change in NCC. Most groups show a trend towards positive numbers, but for the transverse prostate groups there is a trend towards negative numbers. All groups have a significant difference between NCC for uncorrected and corrected images. Cor. = coronal, trans. = transverse.

4.2.2 User-friendliness

The prototype was successfully installed in syngo.via Frontier on the first attempt, and it essentially worked as intended there. However, there were two things that deviated—firstly, that Frontier seemed to automatically split up the mosaic images before they were sent into the prototype. This led to the loss of some frame-specific DICOM tag values, including the b-values. Secondly, when saving the images after correction, they cannot be saved under the original patient name. They will instead be saved under the new patient name "!ResearchOnly_(Original- PatientName)", and this behaviour is set deliberately by Frontier for security reasons. The distortion correction calculation and the correction of diffusion-weighted images took approximately half the time compared to that in FrontierHost. However, as mentioned in Section 3.2.2, there were problems with installing an updated version of the prototype later.

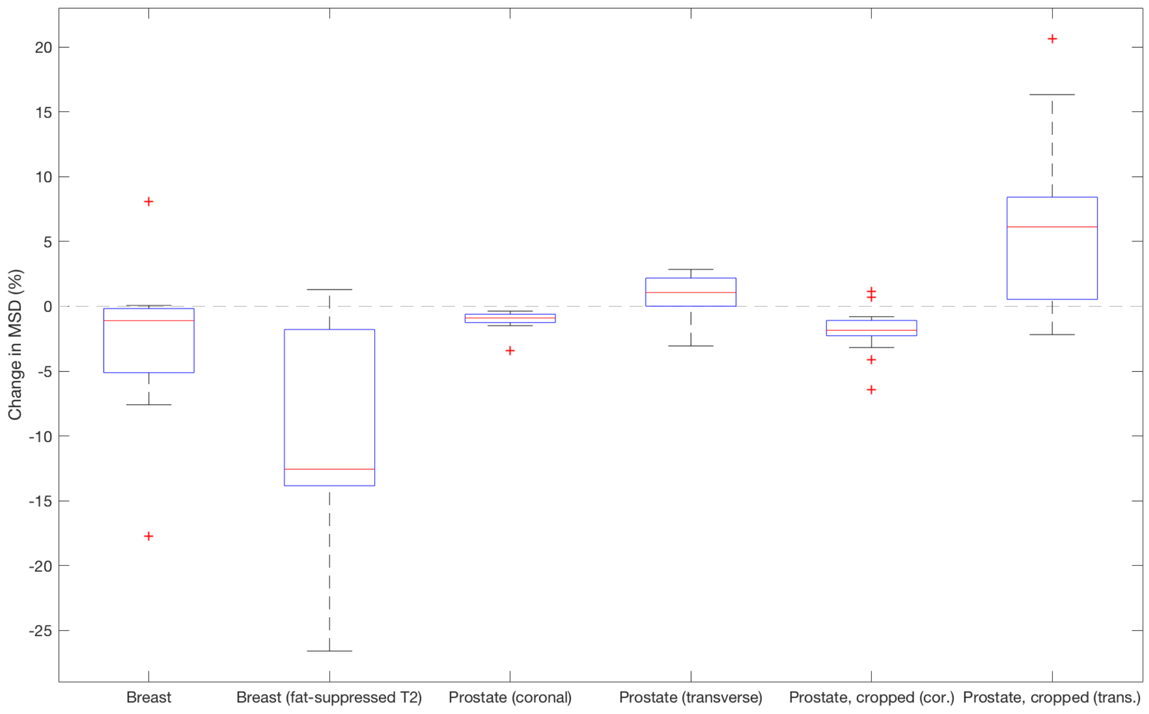


Figure 4.17: Box plot showing the distribution of relative change in MSD between forward phase encoded b_0 and T_2-weighted images from before to after distortion correction. In theory, a successful correction should give a negative change in MSD. Most groups show a trend towards negative numbers, but for the transverse prostate groups there is a trend towards positive numbers. All groups have a significant difference between MSD for uncorrected and corrected images. Cor. = coronal, trans. = transverse.

The test subjects' answers on the short answer questions from the feedback form are shown in Table 4.5. They did not think it took too long time to use the prototype, and they both pointed out that the use would be faster after the first time they tried it. It also became clear that to use the prototype as it is now, the user should have some knowledge about the distortion correction method used.

The two things that turned out to be the most challenging were selection of the images—especially which images to choose when—in the browser, because the user interface of that was not very intuitive, and understanding how to view the different uncorrected and corrected images (forward and reverse b_0 and DW images) in the prototype. It was also suggested to have step 2 and 3, which are the calculation of the distortion correction and the correction of the DW images, in one single step. Another thing was to have more information about the images (e.g. b-values) in the main window of the prototype. In addition, it could be somewhat unclear what buttons to press in the parameter selection. However, both test subjects answered that they could have used the prototype in their workflow, because

Table 4.5: The short answer questions from the feedback form with answers from the test subjects.

	Radiographer	**Medical physicist**
How long time did it take to use the application, from start to end?	18 minutes	13 minutes
What do you think about the time it took to correct the images?	Okay	Okay
Were you able to correct the images?	Yes	Yes
How easy was it to understand how to correct them? (1 = very difficult, 5 = very easy)	3	4
How intuitive was it to understand the user interface and to navigate in the prototype? (1 = very difficult, 5 = very easy)	3	4
Is this a prototype that you could have used in your workflow?	Yes	Yes

after trying the prototype for the first time it would be relatively easy to use.

Chapter 5

Discussion

This chapter will begin with a discussion of the methods used and their strengths and challenges. The results will further be discussed and interpreted, and related to other research in the field. At last, implementation of the prototype in clinical workflow will be discussed, followed by suggestions for future work.

5.1 Methods

5.1.1 Making the prototype

MeVisLab can be challenging for those who have not used the program before. The combination of modules and scripts may be somewhat confusing, and it can be hard to understand the relationship between them. When making an MR-related prototype for syngo.via Frontier, it is highly recommended to base it on a demo prototype from the MR StarterKit. This provides a functioning framework which can further be developed and tailored. An advice for new developers, to understand how everything is connected, is to try to change small things in the networks—e.g. the instance name of a module—and see in which scripts error messages occur. Also, one can change small things in the networks and scripts to see how the behaviour of the prototype changes in FrontierHost. Once these relations have been understood, further prototype development will be much easier. Also, it is possible to discuss problems with fellow MeVisLab developers in the Fraunhofer MEVIS Forum (`https://forum.mevis.fraunhofer.de`).

5.1.1.1 Image processing

The vital module in this project, in order to implement the distortion correction method by Holland et al., was the ExternalExecutableWrapper (EEW). This module can, however, be slightly challenging in the beginning due to the lack of a detailed help documentation. A crucial point of getting the module to run the external executable, is to ensure that all path names are correct. The EEW creates the folders algo_input and algo_output, where the input and output files of the executable are placed, respectively. When specifying paths as arguments for an executable, these should be written on the form "algo_input/FileName". For these relative paths to work, the "Executable Working Dir" of the "Binary File & Arguments" tab of the EEW panel should be identical to the "Working Dir" of the "Files & Directories" tab. It can be wise to create a connector between these two fields to pass on the correct name from the "Working Dir" to the "Executable Working Dir".

There was also a problem when trying to correct all the diffusion-weighted images consecutively by iterating `reformatx` and `imagemath`, because the EEW will only run the executable after a function has finished, and not inside it. Thus, if trying to start the executable inside an iterative loop in the function and incrementing the 3D image selected by SubImage in every loop, all these incrementations would first be performed inside the function, before all iterations of the executables would be run after this. The result of this is that the same 3D image will be corrected every time the executable is run. Trying to start the next run of the executable from the post processing script of the previous one did not solve the problem either, as this solution was very unstable because the post processing script of the EEW was not run every time. The solution was then to use FieldListeners to initiate the next executable, as these are triggered when the previous executable has finished. They will then call a function where the execution of the next executable is initiated. Before this solution was found, the user had to click the button one time for each image to be corrected, which is not very user-friendly if there are many images.

Some pixels in the corrected images ended up with a negative value, and it was decided to set all of these to zero. This had to be done because DICOM images are stored in the unsigned 16-bit integer format where there can be no negative numbers. The reason why some pixels become negative is probably that when applying the calculated distortion field, intensity is being removed from a place where there is little or no intensity from before. There could be imperfections in the distortion field, for example due to noise or movement between images. It was actually noted that before the breast images were motion corrected, the extent of negative pixels in the diffusion-weighted images was higher than after motion correction. It was considered reasonable to set the negative pixels to zero because a negative pixel intensity can be interpreted as no intensity. If instead trying to adjust this by shifting the scale so that the lowest pixel value would be set to zero, the result would be that the image intensities would no longer be comparable to those in non-corrected images. It would also cause problems when trying to calculate parameters based on the pixel values.

5.1.1.2 DICOM tags

As mentioned in Section 3.1.6, to preserve all frame-specific DICOM tags when saving the corrected images, the solution was that the user has to select all the images in a series. However, a workaround solution for this problem was found later, suggested by someone from Siemens. The MRPatientBrowserRoles module has a field called "phasesChecked", which contains a string telling which phases in the image series that have been selected. By connecting the square output of MRPatientBrowserRoles (see Figure 3.4) to a module called MultiFileVolumeListImageOutput, it is possible to extract the DICOM headers from the selected phases. This way, the user could get away with selecting only some of the images in the series. However, this was discovered very late in the process, so this could not be implemented in this version of the prototype due to time limitations.

5.1.1.3 User interface and user-friendliness

In the menu of the prototype, it is currently possible to change a few parameters of the distortion correction. This includes choosing whether the images are mosaics, whether the reverse image is flipped relative to the forward images, and the phase encoding direction. For all other parameters, the default values were used. These were the maximum and minimum smoothing kernel, number of iterations, folding constraint weight and smoothness constraint weight (the two last terms of Equation (2.15)), among others. If these other parameters should be changed, there are two options; to let the users select the values of all parameters themselves, or to predefine a set of parameters for different image types (e.g. breast, prostate) and then let the user select which image type is being corrected. The first option requires the user to have more knowledge about the distortion correction method and the parameters than the second option. To select the parameters, the user could either select from a list of values in a dropdown menu, or there could be fields where the user can type in the desired values—however, this requires that the user types in a valid value. For the second option, analyses need to be performed to determine the best set of parameters for each image type. Also, the possibility of automatic detection of mosaics, flipped reverse images and phase encoding direction needs to be investigated more closely.

The EPI distortion correction prototype developed here is still an early version, and needs to be further modified to improve robustness, flexibility and efficiency. To obtain this, the prototype should be as little "hard-coded" as possible. This has been tried to achieve in this first version of the prototype, for example in the correction of the diffusion-weighted images, which adapts to the number of images present. However, there is still room for improvements. In addition to the points already discussed, it should also be possible to use the prototype for distortion correction of other types of EPI images than DWI, and the user interface should therefore be adapted to fit all types of cases. To make the distortion correction process more efficient, it should be investigated whether it is possible to run parallel computations in MeVisLab and Frontier. The existing code must also be examined to see if it can be made more efficient.

5.1.2 Testing the prototype

5.1.2.1 Correcting images

The default values of the distortion correction parameters were used for all patients. It is possible that the corrected images would have been better if the parameters had been optimised individually for each patient, but then the results would not have been comparable between patients. In addition, adjusting the parameters for each patient might not feasible in clinical practice because it would not be very effective, perhaps unless the distortion correction obviously fails. Another option could then have been to perform a more extensive analysis where the best set of parameters for all patients was found, but this would have been too time-consuming for this project. It was therefore chosen to use the default parameter values for all patients, and they seemed to work satisfactory, at least for the images investigated in this project. It should also be noted that changing the value of certain parameters may increase the time it takes to calculate the distortion correction.

The MSD was chosen for comparison of forward and reverse phase encoded b_0 images before and after correction. It should be well-suited for this since the forward and reverse b_0 images have the same type of contrast, giving an MSD of 0 if the images are identical. The L_2-square norm can also be used for this purpose, and it is qualitatively very similar to the MSD [4]. For comparison between corrected b_0 and T_2-weighted images, NMI, NCC and MSD were calculated. These metrics were chosen because they represent slightly different ways to measure the similarity between two images, and it was desirable to see which of them performed best. Other similar metrics that have been previously used are Mattes mutual information, which is qualitatively similar to the NMI but not normalised, and other unnormalised forms of the NMI and NCC [37, 4, 26]. Linear correlation coefficient, Jaccard index, and the translation and rotation parameters from the co-registration have also been reported used [38, 26, 5]. In addition, it has been suggested to smooth the b_0 and T_2-weighted images before calculating the metrics, in order to reduce noise effects [38].

5.1.2.2 User-friendliness

Due to problems when trying to install an updated version of the prototype in syngo.via Frontier, the user-friendliness testing had to be performed in FrontierHost in MeVisLab. However, the functionality and layout of the prototype should be the same in FrontierHost as in syngo.via Frontier, the only difference might be the speed of the calculations, and that the corrected images cannot be saved to the server when using FrontierHost. The user experience would therefore be quite similar, so it was not considered to be a large problem to test the prototype in FrontierHost. The prototype was tested by a radiographer and a medical physicist, but in the future it should probably also be tested by radiologists.

The reason for the installation problems in syngo.via Frontier was not known, even though an expert from Siemens also was involved in the troubleshooting. The only difference in the prototype from the first version, which was installed successfully, was that the values of some DICOM tags were changed. This problem has to be further investigated and solved at a later point.

5.2 Results

5.2.1 The prototype

The user interface and the functionalities of the prototype ended up as planned. The layout styles are to a large extent controlled by code and modules from the demo prototype, to fit in well with the syngo.via Frontier environment. The interface of the prototype should be relatively intuitive for the user to understand, but pop-up windows and tabs with information were in addition made to make things clearer. This was considered to be especially important in the browser where the user selects the images, because this is possibly the least intuitive part of the prototype, and it is important for the distortion correction that the correct images are selected in the correct order.

5.2.2 Testing the prototype

5.2.2.1 Correcting images

None of the performed distortion corrections failed. By visual assessment, the quality of correction for 12 of 13 breast patients and all prostate patients was determined to be successful. The natural anatomy appears to have been restored by the correction. However, in the edges of the images, "banding effects" can sometimes be seen, such as in the edges on the right of the images in Figure 4.7b and d. These appear as streaks or "bands" of alternating bright and dark pixel intensity. These are the result of discontinuities in the distortion field, and can be reduced by increasing the value of the distortion correction parameter "smoothness constraint weight", which is the last term in the cost function, Equation (2.15). However, increasing the smoothness of the distortion field can also lead to a less successful correction because the absolute values of the displacement could be restrained. This will lead to a smaller displacement than necessary to correct for the distortion, and the resulting image will look more and more similar to the uncorrected image for higher values of this parameter. Furthermore, since the banding effects observed here are not too severe and are only found in the edges, they do not interfere with the regions of interest—breast and prostate—that are located near the centre of the images. They are therefore considered as acceptable, and do not affect the correction quality at this extent.

The corrected forward and reverse phase encoded b_0 images look very similar. This is supported by the calculated MSD, which decreased significantly by approximately 87-91% after correction. An MSD decrease of 100% would indicate that the forward and reverse images are identical. Teruel et al. performed distortion correction on breast images from the same study as the breast images in this project, and found a mean L_2-square norm change of 94% [4]. As both MSD and L_2-square norm measure the intensity difference between pixels in the images, their result is in almost the same range as here, only slightly better. However, other distortion correction parameters could have been used, as the parameter values used are not stated in the paper, only that they were kept fixed for all patients.

Regarding the efficiency of the distortion correction using FrontierHost, approximate times were noted but the exact time can vary and depends on several factors, such as computer strength, type of image and degree of geometric distortion. The computer used in this case was almost 10 years old and had a computer processing unit (CPU) at 3.10 GHz with 4 cores and 4 threads, and a random access memory (RAM) of 8 GB. Newer computers with better CPU and RAM would most likely be more efficient. Also, regarding the time it took to run through the whole prototype, it will probably take longer time to run through it for those who use it for the first time.

5.2.2.1.1 Co-registration

Co-registration was successfully performed for all patients using NMI as the metric to be optimised. However, for co-registration of breast b_0 images to fat-suppressed T_2-weighted images NCC was the best metric, probably because the contrast of these two image types is more similar. When comparing co-registered b_0 images to the T_2-weighted images, the change in NMI was significantly positive for the breast group. Breast co-registered to fat-suppressed T_2-weighted images also showed a trend towards positive changes, although not significant. For the prostate images, there were no significant changes in the NMI. All groups had a significant change in NCC and MSD. However, while most groups showed a positive change in NCC, the two transverse prostate groups had a negative change. Likewise, most groups had a negative change in MSD, while the two transverse prostate groups had a positive change. Nevertheless, both the distortion correction and co-registration look successful by visual examination, even for the transverse prostate images.

For the breast patients, all three metrics support the visual assessment of the quality of corrections, while for the prostate patients the results are more inconclusive. This is possibly because the prostate area is surrounded by much more structures than the breast. The values of the metrics are based on the whole image volume, while the distortion effects can be quite local. Therefore, the prostate images were also cropped to only include (approximately) the prostate area, but this did not considerably affect the change in NMI. However, for the NCC and MSD, the changes were enhanced for the cropped images relative to the full-size images. That is, the metrics that already had a positive change became more positive, while the metrics that already had a negative change became more negative. For the coronal prostate images, this supports the hypothesis about local distortion effects versus whole-volume metrics.

For the transverse prostate images, the results are more complicated. There is a significant change in both NCC and MSD, but it is negative and positive, respectively. Since both NCC and MSD impose stricter requirements to pixel intensity similarities between the images than NMI, an intensity mismatch between the b_0 and T_2-weighted images might cause this behaviour. A hypothesis could be that for the transverse b_0 images, the prostate and rectum appear bright and the area around them is dark, while on the transverse T_2-weighted images the prostate and rectum are quite dark and the area around them is bright (see Figure 5.1a, b). However, for the coronal images this relationship is opposite, for both b_0 and T_2-weighted images the prostate area is brighter than the area around it (see Figure 5.1c, d). This way, in the uncorrected transverse images, the bright pixels of a distorted prostate in the b_0 image would be a better match to the bright area around the prostate in the T_2-weighted image. Accordingly, the match would be poorer after distortion correction. This would also explain why the negative change in NCC and the positive change in MSD are enhanced for the cropped images. Most other structures in the image that could affect the metrics are eliminated in the cropped images, and the effects of the prostate area are enhanced. However, even if this hypothesis was correct, this would only be an explanation of the behaviour, and the results cannot be used to draw any conclusions. Another reason for the inconclusive results could possibly be the banding effects in the corrected images, but these should not be severe enough to affect the metrics to that extent. In addition, these effects should be removed in the cropped images. However, it should also be noted that the co-registration could actually partially compensate for linear distortion effects, thus decreasing the difference between the uncorrected and corrected image [5].

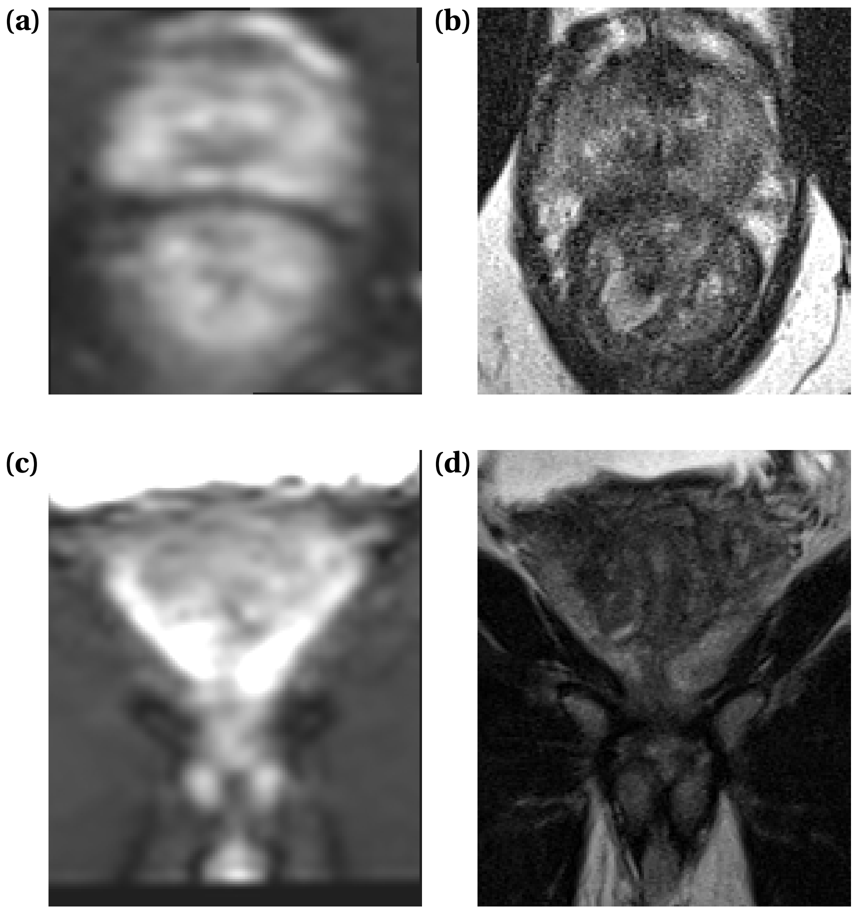


Figure 5.1: **(a)**, **(b)** Examples of corrected b_0 and T_2-weighted cropped, transverse prostate images, respectively. In the b_0 image the prostate area is brighter than the area around it, while in the T_2-weighted image the prostate area is darker than the area around it. In this case, the relative changes in NMI, NCC and MSD were −0.70%, −113.88% and 14.65%, respectively. Note that these two images are from the same patient as in Figure 4.12. **(c)**, **(d)** Examples of corrected b_0 and T_2-weighted cropped, coronal prostate images, respectively. In both images, the prostate area is brighter than the area around it. In this case, the relative changes in NMI, NCC and MSD were 0.27%, 4.45% and −6.44%, respectively.

Another thing that might have affected the metrics is the fact that after distortion correction and co-registration, the first and last slices of the image volumes can end up being partially or completely zero, because pixels have been shifted. This phenomenon was present to a much greater extent in the co-registered transverse prostate images than the coronal ones. For the uncorrected co-registered transverse prostate images it was also present, however, the outer slices were often only partially zero, while for the corrected images the outer slices were more often completely zero (see Figure 5.2a, b). This might have affected the similarity metrics, possibly to a large enough extent to make the relative change of NCC negative (and positive for MSD) for the transverse images. Although the prostate images were in addition cropped to try to eliminate edge effects, they were only cropped in the in-plane direction, and not in the slice direction. When comparing the full-sized transverse prostate images with the cropped ones, it was found that the outer slices of the cropped images often had a larger percentage of zero-valued pixels than the full-sized images (see Figure 5.2c, d). This might explain why the relative change in NCC became even more negative (and more positive for MSD) for the cropped transverse images, because their larger percentage of zero-valued pixels in the image volumes would give a poorer similarity to the T_2-weighted images. For the coronal prostate images, the full-sized images had a larger percentage of zero-valued pixels after co-registration (however, this was in-plane and not in the outer slices) than the cropped images, because the FOV of the coronal b_0 images was very different from the T_2-weighted images (see Figure 5.2e, f). Accordingly, this might explain why the relative change in NCC became even more positive (and more negative for MSD) for the cropped coronal images, because their lower percentage of zero-valued pixels in the image volumes would give a better similarity to the T_2-weighted images. In the future, the outer slices of the images should probably also be removed in order to exclude this phenomenon. This could not be tested in this project due to time limitations.

Teruel et al. calculated Mattes mutual information between the b_0 and fat-suppressed T_2-weighted breast images before and after distortion correction [4]. They found that the metric was higher for all patients after correction, and the difference was significant when pairwise comparing the metric before and after correction. This is similar to the results in this project, where breast b_0 images co-registered both to conventional T_2-weighted images and to fat-suppressed T_2-weighted images showed a trend towards an increase in NMI, although only significant for the conventional T_2-weighted images. Hancu et al. also performed distortion correction on breast images using the same distortion correction method, and used NCC to compare the b_0 images with fat-suppressed T_2-weighted images [39]. They found a significantly higher NCC for the corrected images than the uncorrected ones, which is the same as found in this project. No comparable results have been found for the prostate, although others have used the distortion correction method in their papers to show how prostate and tumour alignment relative to T_2-weighted images qualitatively improves after distortion correction [5, 40].

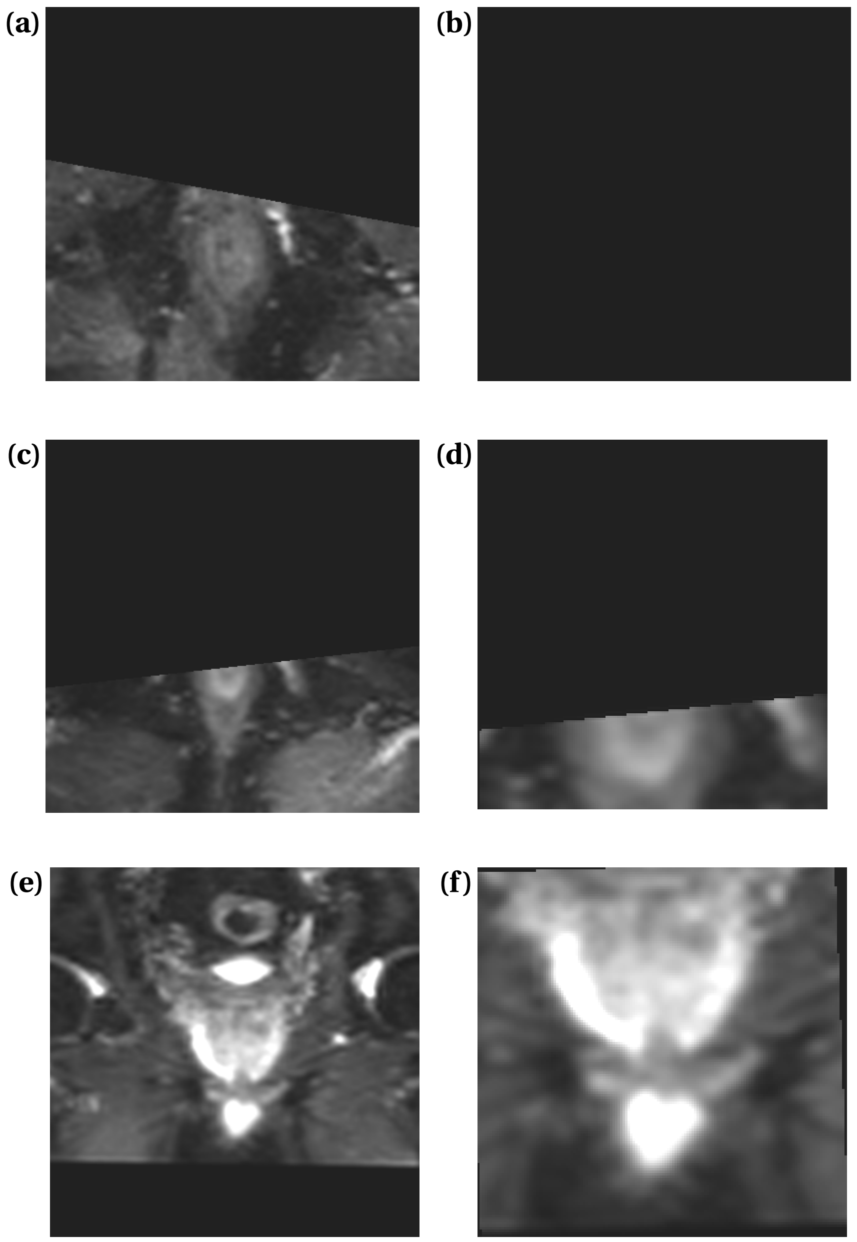


Figure 5.2: **(a)**, **(b)** Examples of the outer slice of uncorrected and corrected b_0 transverse prostate images, respectively. The uncorrected slice is only partially zero, while the corrected slice is completely zero. In this case, the relative change in NCC between the uncorrected and the corrected image was negative, and the relative change in MSD was positive. **(c)**, **(d)** Examples of the outer slice of corrected full-sized and cropped b_0 transverse prostate images, respectively. The cropped slice has a larger percentage of zero-valued pixels than the full-sized slice. Here, the relative change in NCC was more negative in the cropped case than in the full-size case, and the relative change in MSD was more positive in the cropped case than in the full-size case. **(e)**, **(f)** Examples of a central slice of corrected full-sized and cropped b_0 coronal prostate images, respectively. The full-sized slice has a larger percentage of zero-valued pixels than the cropped slice. Here, the relative change in NCC was more positive in the cropped case than in the full-size case, and the relative change in MSD was more negative in the cropped case than in the full-size case.

To sum up, for the breast images all three metrics supported the visual assessment of the corrected images, where images from 12 of 13 patients were successfully corrected. For the coronal prostate images, NCC and MSD supported the visual assessment, where all images were successfully corrected. However, for the transverse prostate images, all metrics gave inconclusive results even though all images were visually assessed to be successfully corrected. Therefore, even though the metrics examined gave good results for some of the groups, one should be careful of using them to assess the quality of corrections on a general basis, at least without considering various factors that could affect them. If they are to be used, these factors should be ruled out, or new metrics have to be tried. It should also be considered to always use fat-suppressed T_2-weighted images—instead of conventional T_2-weighted images—for co-registration, since these often have a more similar contrast and may therefore give better values of the similarity metrics. For the breast images investigated in this project, using fat-suppressed T_2-weighted images yielded a larger relative change in NCC and MSD than using the conventional ones.

5.2.2.2 User-friendliness

After successfully installing the prototype in syngo.via Frontier, it was discovered that Frontier automatically split up mosaic images before they were sent into the prototype, and since the image volumes then do not have the original dimensions anymore, frame-specific DICOM tags will be lost (e.g. b-values). Since it can be useful to know the b-values when choosing images for the correction, this problem should be solved, but it is not yet known how deep this behaviour is implemented. It could either be a feature that is deeply rooted in syngo.via Frontier, or it might be controlled by the network or some code in MeVisLab. This has to be further investigated.

Also, the corrected images could not be exported to the server under the original patient name, and this behaviour is set deliberately by Frontier for security reasons. However, a (somewhat complicated) workaround for this has been suggested in the Fraunhofer MEVIS Forum. Nevertheless, whether this should be implemented to save the images under the correct patient name has to be considered carefully, so that it is safe to use in terms of privacy policies. There must be no risk of overwriting or deleting original image series or other personal data from the server.

Since the distortion correction calculations took approximately half the time in syngo.via Frontier compared to in FrontierHost, the time to use the prototype there would probably be in the order of 2 to 10 minutes. Depending on what images to be corrected and who is going to use the prototype, this might be a little too long for use in clinical practice. The time it takes to perform the calculations depends on the syngo.via server. Therefore, the efficiency might be improved in the future if the server is upgraded.

The feedback from the test subjects showed that the prototype has potential to be an easy-to-use tool for distortion correction, but as it is now some training might be needed when using it for the first time. Also, they did not think it took too long time to use it. However, the browser for selecting the images to be corrected should be improved. As already mentioned in Section 5.1.1.2, the prototype will in the future be changed so that it is possible to select only some of the images in a series. In addition, after consulting with someone from Siemens, it became clear that it is possible to select all the images (forward and reverse b_0 and DW images) and assign them to the correct roles in only one step instead of three, which might be easier for the user. This will also be implemented in a future version of the prototype. Furthermore, explanations of what images to choose when must be improved.

It must also be made clearer for the user how to select which images should be rendered in the top and bottom row in the prototype interface. Both test subjects did not immediately understand how to view the diffusion-weighted images after correcting them. This should either be explained better in this step (step 3), or the whole layout of how to choose which images to be rendered should be changed. In addition, the other feedback from the test subjects must also be taken into consideration, and the prototype should be further adapted—e.g. in terms of terminology used—depending on which professions might use the prototype in the future.

5.3 Implementation in clinical workflow

For the prototype to be used in clinical workflow, it needs to be efficient enough for it to pay off to use distortion corrected images, rather than using the uncorrected images. As mentioned before, the efficiency of the calculations depends both on the prototype itself and the syngo.via server, and there is room for improvements. One must also consider which professions are going to use the prototype—is it radiologists, radiographers or medical physicists? The prototype must then accordingly be adapted to fit into their workflow. The most probable application of the prototype is distortion correction of functional images before quantification of various parameters (e.g. ADC), but if it is efficient enough, it might also be used before visual assessment of heavily distorted images.

Another important question is whether syngo.via Frontier is the optimal platform for this kind of prototypes. Although it has some limitations, its strengths are that syngo.via is already installed and in use at St. Olavs hospital, and that images can be taken directly from the server for in-house processing without having to anonymise the data. Furthermore, the MeVisLab add-on packages Frontier Development Kit and MR StarterKit provide a framework that facilitates development of own prototypes. NTNU and St. Olavs hospital have in addition established a dialogue with Siemens, which can make it easier to solve any problems that may occur. At this time, syngo.via Frontier is therefore considered as suitable for this use and the distortion correction prototype will be further developed on this platform.

5.4 Future work

Further, the prototype must be improved to make it more robust, flexible and efficient, based on the suggestions already given in this chapter. The possibility of allowing the user to change more parameters in the prototype should be assessed. Furthermore, the user-friendliness should also be improved based on the feedback from the test subjects. Motion and eddy-current correction could in addition be integrated into the prototype at a later point, so that this does not have to be performed in separate steps. Eventually, the prototype should be integrated into clinical workflow for a test period to see how it performs. However, first and foremost, the installation problems in syngo.via Frontier must be solved.

Even though the images in this project were found to be successfully corrected using the default parameters, a more thorough investigation should be performed to optimise the distortion correction parameters for a larger number of patients, to see if the corrected images can be improved even further. More research needs also to be performed to find metrics that reliably can quantitatively determine the quality of corrections. A number of different metrics must therefore be tested for this purpose, and evaluated on different types of images.

The metrics can be used in the prototype to warn the user if the correction has failed, or in more automated distortion correction methods to detect poorly corrected images. They can also be used to evaluate the outcome of different distortion correction parameters.

Chapter 6

Conclusion

In this master's project, a prototype for geometric distortion correction of EPI images has been successfully developed in MeVisLab and installed in syngo.via Frontier. It has been tested on diffusion-weighted EPI images from breast and prostate patients, where none of the corrections failed, giving a robustness of 100%. By visual assessment of the images, the quality of correction was determined to be successful for 12 of 13 breast patients and all 16 prostate patients.

The MSD between forward and reverse images before and after distortion correction was calculated, and it was found to decrease significantly for all types of images, which supports the results from the visual examination. Uncorrected and corrected forward b_0 images were co-registered to anatomically correct T_2-weighted images, and NMI, NCC and MSD between the b_0 and T_2-weighted images before and after correction were calculated. These metrics generally showed good results for the breast images, but gave some inconclusive results for the prostate images, and one might need to be careful of using them to assess the quality of corrections on a general basis. More investigations should be performed to find suitable metrics for this use.

To assess the user-friendliness of the prototype, a radiographer and a medical physicist tested it and gave feedback about both its efficiency and the user interface. From this, the conclusion is that the prototype has potential to be an easy-to-use tool for geometric distortion correction of EPI images in clinical workflow, although there is still room for improvements. In the future, the prototype will be further developed to improve its functionality, efficiency and user-friendliness.

Bibliography

[1] C. Westbrook, C. K. Roth, and J. Talbot, *MRI in Practice*, 4th ed. Wiley-Blackwell, 2011.

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

[3] D. Holland, J. M. Kuperman, and A. M. Dale, "Efficient correction of inhomogeneous static magnetic field-induced distortion in Echo Planar Imaging," *NeuroImage*, vol. 50, no. 1, pp. 175–183, 2010.

[4] J. R. Teruel, H. E. Fjøsne, A. Østlie, D. Holland, A. M. Dale, T. F. Bathen, and P. E. Goa, "Inhomogeneous static magnetic field-induced distortion correction applied to diffusion weighted MRI of the breast at 3T," *Magnetic Resonance in Medicine*, vol. 74, no. 4, pp. 1138–1144, 2015.

[5] G. Nketiah, K. M. Selnaes, E. Sandsmark, J. R. Teruel, B. Krüger-Stokke, H. Bertilsson, T. F. Bathen, and M. Elschot, "Geometric distortion correction in prostate diffusion-weighted MRI and its effect on quantitative apparent diffusion coefficient analysis," *Magnetic Resonance in Medicine*, vol. 79, no. 5, pp. 2524–2532, 2018.

[6] I. F. Syversen, "Prediction of chemoradiotherapy response in rectal cancer using static and dynamic R2* measurements," Project thesis (unpublished), Norwegian University of Science and Technology, 2017.

[7] E. Grøvik, "Multimodal Dynamic MRI for Structural and Functional Assessment of Cancer," Doctoral thesis, University of Oslo, 2017.

[8] R. W. Brown, Y.-C. N. Cheng, E. M. Haacke, M. R. Thompson, and R. Venkatesan, *Magnetic Resonance Imaging: Physical Principles and Sequence Design*, 2nd ed. Wiley-Blackwell, 2014.

[9] M. T. Vlaardingerbroek and J. A. den Boer, *Magnetic Resonance Imaging: Theory and Practice*, 3rd ed. Springer, 2003.

[10] P. C. Hemmer, *Kvantemekanikk*, 5th ed. Tapir Akademisk Forlag, 2005.

[11] J. F. Schenck, "The role of magnetic susceptibility in magnetic resonance imaging: MRI magnetic compatibility of the first and second kinds," *Medical Physics*, vol. 23, no. 6, pp. 815–850, 1996.

[12] Y. Guo, Y.-Q. Cai, Z.-L. Cai, Y.-G. Gao, N.-Y. An, L. Ma, S. Mahankali, and J.-H. Gao, "Differentiation of clinically benign and malignant breast lesions using diffusion-weighted imaging," *Journal of Magnetic Resonance Imaging*, vol. 16, no. 2, pp. 172–178, 2002.

[13] N. M. DeSouza, S. A. Reinsberg, E. D. Scurr, J. M. Brewster, and G. S. Payne, "Magnetic resonance imaging in prostate cancer: The value of apparent diffusion coefficients for identifying malignant nodules," *British Journal of Radiology*, vol. 80, no. 950, pp. 90–95, 2007.

[14] P. Mansfield, "Multi-planar image formation using NMR spin echoes," *Journal of Physics C: Solid State Physics*, vol. 10, no. 3, pp. L55–L58, 1977.

[15] F. Schmitt, M. K. Stehling, and R. Turner, *Echo-Planar Imaging: Theory, Technique and Application.* Springer-Verlag Berlin Heidelberg, 1998.

[16] J. Kybic, P. Thévenaz, A. Nirkko, and M. Unser, "Unwarping of unidirectionally distorted EPI images," *IEEE Transactions on Medical Imaging*, vol. 19, no. 2, pp. 80–93, 2000.

[17] B. Zitová and J. Flusser, "Image registration methods: A survey," *Image and Vision Computing*, vol. 21, no. 11, pp. 977–1000, 2003.

[18] P. Jezzard and R. S. Balaban, "Correction for Geometric Distortion in Echo-Planar Images from B0 Field Variations," *Magnetic Resonance in Medicine*, vol. 34, no. 1, pp. 65–73, 1995.

[19] M. D. Robson, J. C. Gore, and R. T. Constable, "Measurement of the point spread function in MRI using constant time imaging." *Magnetic Resonance in Medicine*, vol. 38, no. 12, pp. 733–740, 1997.

[20] H. Zeng and R. T. Constable, "Image distortion correction in EPI: Comparison of

field mapping with point spread function mapping," *Magnetic Resonance in Medicine*, vol. 48, no. 1, pp. 137–146, 2002.

[21] Q. S. Xiang and F. Q. Ye, "Correction for geometric distortion and N/2 ghosting in EPI by phase labeling for additional coordinate encoding (PLACE)," *Magnetic Resonance in Medicine*, vol. 57, no. 4, pp. 731–741, 2007.

[22] A. N. Priest, D. W. Carmichael, E. De Vita, and R. J. Ordidge, "Method for spatially interleaving two images to halve EPI readout times: Two reduced acquisitions interleaved (TRAIL)," *Magnetic Resonance in Medicine*, vol. 51, no. 6, pp. 1212–1222, 2004.

[23] A. N. Priest, E. De Vita, D. L. Thomas, and R. J. Ordidge, "EPI distortion correction from a simultaneously acquired distortion map using TRAIL," *Journal of Magnetic Resonance Imaging*, vol. 23, no. 4, pp. 597–603, 2006.

[24] A. N. Priest, E. De Vita, and R. J. Ordidge, "Doubling the resolution of echo-planar brain imaging by acquisition of two k-space lines per gradient reversal using TRAIL," *NMR in Biomedicine*, vol. 21, no. 2, pp. 79–88, 2008.

[25] C. Studholme, R. Todd Constable, and J. S. Duncan, "Accurate alignment of functional EPI data to anatomical MRI using a physics-based distortion model," *IEEE Transactions on Medical Imaging*, vol. 19, no. 11, pp. 1115–1127, 2000.

[26] X. Hong, X. V. To, I. Teh, J. R. Soh, and K. H. Chuang, "Evaluation of EPI distortion correction methods for quantitative MRI of the brain at high magnetic field," *Magnetic Resonance Imaging*, vol. 33, no. 9, pp. 1098–1105, 2015.

[27] M. Wu, L.-C. Chang, L. Walker, H. Lemaitre, A. S. Barnett, S. Marenco, and C. Pierpaoli, "Comparison of EPI Distortion Correction Methods in Diffuision Tensor MRI Using a Novel Framework," in *Med Image Comput Comput Assist Interv.*, vol. 11, no. 2, 2008, pp. 321–329.

[28] H. Chang and J. M. Fitzpatrick, "A Technique for Accurate Magnetic Resonance Imaging in the Presence of Field Inhomogeneities," *IEEE Transactions on Medical Imaging*, vol. 11, no. 3, pp. 319–329, 1992.

[29] P. S. Morgan, R. W. Bowtell, D. J. McIntyre, and B. S. Worthington, "Correction of Spatial Distortion in EPI Due to Inhomogeneous Static Magnetic Fields Using the Reversed

Gradient Method," *Journal of Magnetic Resonance Imaging*, vol. 19, no. 4, pp. 499–507, 2004.

[30] J. L. Andersson, S. Skare, and J. Ashburner, "How to correct susceptibility distortions in spin-echo echo-planar images: Application to diffusion tensor imaging," *NeuroImage*, vol. 20, no. 2, pp. 870–888, 2003.

[31] T. Rohlfing, "User Guide to The Computational Morphometry Toolkit," Menlo Park, CA, 2011.

[32] T. Rohlfing, "Unwarping Echo Planar Images Using CMTK," Menlo Park, CA, 2012.

[33] F. Wilcoxon, "Individual Comparisons by Ranking Methods," *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, 1945.

[34] J. P. Pluim, J. B. Maintz, and M. A. Viergever, "Mutual-information-based registration of medical images: A survey," *IEEE Transactions on Medical Imaging*, vol. 22, no. 8, pp. 986–1004, 2003.

[35] V. Roshni and K. Revathy, "Using mutual information and cross correlation as metrics for registration of images," *Journal of Theoretical and Applied Information Technology*, vol. 4, pp. 474–481, 2008.

[36] D. F. Williamson, R. A. Parker, and J. S. Kendrick, "The Box Plot: A Simple Visual Method to Interpret Data," *Annals of Internal Medicine*, vol. 110, no. 11, pp. 916–921, 1989.

[37] J. M. Treiber, N. S. White, T. C. Steed, H. Bartsch, D. Holland, N. Farid, C. R. McDonald, B. S. Carter, A. M. Dale, and C. C. Chen, "Characterization and correction of geometric distortions in 814 Diffusion Weighted Images," *PLoS ONE*, vol. 11, no. 3, 2016.

[38] J. Vardal, R. A. Salo, C. Larsson, A. M. Dale, D. Holland, I. R. Groote, and A. Bjørnerud, "Correction of B0-distortions in echo-planar-imaging-based perfusion-weighted MRI," *Journal of Magnetic Resonance Imaging*, vol. 39, no. 3, pp. 722–728, 2014.

[39] I. Hancu, S. K. Lee, K. Hulsey, R. Lenkinski, D. Holland, J. I. Sperl, and E. T. Tan, "Distortion correction in diffusion-weighted imaging of the breast: Performance assessment of prospective, retrospective, and combined (prospective + retrospective) approaches," *Magnetic Resonance in Medicine*, vol. 78, no. 1, pp. 247–253, 2017.

[40] R. A. Rakow-Penner, N. S. White, D. J. Margolis, J. K. Parsons, N. Schenker-Ahmed, J. M. Kuperman, H. Bartsch, H. W. Choi, W. G. Bradley, A. Shabaik, J. Huang, M. A. Liss, L. Marks, C. J. Kane, R. E. Reiter, S. S. Raman, D. S. Karow, and A. M. Dale, "Prostate diffusion imaging with distortion correction," *Magnetic Resonance Imaging*, vol. 33, no. 9, pp. 1178–1181, 2015.

Appendix A

Scripts

Note that only relevant parts of the scripts are included.

A.1 EPIdistortionCorrection.py

```
from PythonQt import QtGui
from mevis import *
import os.path
import ast

_frontier = None
_monitorConfiguration = None
_landscapeLayout = True
_monitorGrid = None
_gridElement = None
_monitorLayoutObject = None
_segmentImagesAndPositionList = list()
_computedHeight = 0
_computedWidth = 0

_patientBrowserOpened = 1

# ———————————— Patient/Volume data ————————————

```

```python
def startImport():

  MLAB.log("starting MRPatientBrowser")
  ctx.field("EPIdistortionCorrectionCore.MRPatientBrowser2.
     patientBrowserHasBeenOpened").value = 1

  # Must stop the bypassing to avoid dividing with 0 in MeanOfPhases
  ctx.field("EPIdistortionCorrectionCore.
     EPIdistortionCorrectionProcessing_Custom.MeanOfPhasesForward.Bypass1.
     noBypass").value = True

  # index DICOM data
  dataDirectory = _frontier.getIncomingDicomDirectory()
  ctx.field("EPIdistortionCorrectionCore.MRPatientBrowser.
     frontierSourceDir").setValue(dataDirectory)
  ctx.field("EPIdistortionCorrectionCore.MRPatientBrowser.import").touch()

  return

def importFinished():

  global _frontier
  global _landscapeLayout

  MLAB.log("import finished")

  # Show patient browser to enable the user to select a volume.
  ctx.module("PatientBrowserPanel").call("showPatientBrowser")
  ctx.field("EPIdistortionCorrectionCore.MRPatientBrowser.
     resetCheckBoxStatus").touch()
  # show patient browser
  if _landscapeLayout:
    _frontier.selectLayout("SINGLE_CA-LEFT_H(1)", ["controlArea", "outPatientBrowser"])
  else:
    _frontier.selectLayout("SINGLE_CA-BOTTOM_H(1)", ["controlArea", "outPatientBrowser"])

  # Popup window with information
  title = "Choose forward phase encoded images"
  text = 'Please select the forward phase encoded image(s) (b0). If several forward phase encoded images are selected, the distortion correction will be computed from their average.\n\nImage series are shown in the upper panel, and the phases are shown in the lower panel when an image series is marked.\n\nNote that both an image series and at least one phase must be checked in order to move on to the next step. All phases checked must also be from the same image series (only one series can be checked).'
  buttonNames = ["Ok"]

  if (ctx.field("EPIdistortionCorrectionCore.frontierEnvironment").value):

    frontier = None
    frontierAPI = ctx.field("EPIdistortionCorrectionCore.inSyngoVia").object()

    if (frontierAPI != None):
      frontierAPI.showMessageBox(title, text, buttonNames)

  else :
    MLAB.showInformation(text, title, buttonNames, 0)

  return

def loadingCompleted():
  global _monitorConfiguration, _landscapeLayout

  MLAB.log("loading completed")

  #_frontier.setIsProcessing(False)

  ctx.module("EPIdistortionCorrectionCore").call("switchToInitialLayout")

  ctx.module("EPIdistortionCorrectionCore").call("loadingDatasetCompleted"
     )

  #Allowing for computation in MeanOfPhases
  ctx.field("EPIdistortionCorrectionCore.
     EPIdistortionCorrectionProcessing_Custom.MeanOfPhasesForward.
     Arithmetic5.i1").value = ctx.field("EPIdistortionCorrectionCore.
     EPIdistortionCorrectionProcessing_Custom.InfoForward.sizeT").value

ctx.field("EPIdistortionCorrectionCore.
   EPIdistortionCorrectionProcessing_Custom.MeanOfPhasesForward.Bypass1.
   noBypass").value = False

  #Starting the next patient browser
  startImport2()

  return


# ------------ Patient/Volume data 2

def startImport2():

  MLAB.log("starting MRPatientBrowser2")
  ctx.field("EPIdistortionCorrectionCore.MRPatientBrowser2.
     patientBrowserHasBeenOpened").value = 2

  # Must stop the bypassing to avoid dividing with 0 in MeanOfPhases
  ctx.field("EPIdistortionCorrectionCore.
     EPIdistortionCorrectionProcessing_Custom.MeanOfPhasesReverse.Bypass1.
     noBypass").value = True

  # index DICOM data
  dataDirectory = _frontier.getIncomingDicomDirectory()
  ctx.field("EPIdistortionCorrectionCore.MRPatientBrowser2.
     frontierSourceDir").setValue(dataDirectory)
  ctx.field("EPIdistortionCorrectionCore.MRPatientBrowser2.import").touch
     ()

  return

def importFinished2():

  global _frontier
  global _landscapeLayout

  MLAB.log("import finished 2")

  # Show patient browser to enable the user to select a volume.
  ctx.module("PatientBrowserPanel").call("showPatientBrowser")
  ctx.field("EPIdistortionCorrectionCore.MRPatientBrowser2.
     resetCheckBoxStatus").touch()
  # show patient browser
  if _landscapeLayout:
    _frontier.selectLayout("SINGLE_CA-LEFT_H(1)", ["controlArea", "
       outPatientBrowser"])
  else:
    _frontier.selectLayout("SINGLE_CA-BOTTOM_H(1)", ["controlArea", "outPatientBrowser"])

    # Popup window with information
  title = "Choose reverse phase encoded images"
  text = 'Please select the reverse phase encoded image(s) (b0). If there are several reverse phase encoded images, the distortion correction will be computed from their average.\n\nIMPORTANT: Select ALL phases in the chosen image series! This is required for the DICOM tags to be stored correctly after processing.\n\nImage series are shown in the upper panel, and the phases are shown in the lower panel when an image series is marked.\n\nNote that both an image series and all its phases must be checked in order to move on to the next step. All phases checked must also be from the same image series (only one series can be checked).'
  buttonNames = ["Ok"]

  if (ctx.field("EPIdistortionCorrectionCore.frontierEnvironment").value):

    frontier = None
    frontierAPI = ctx.field("EPIdistortionCorrectionCore.inSyngoVia").object()

    if (frontierAPI != None):
      frontierAPI.showMessageBox(title, text, buttonNames)

  else :
    MLAB.showInformation(text, title, buttonNames, 0)

  return

def loadingCompleted2():
  global _monitorConfiguration, _landscapeLayout

  MLAB.log("loading completed 2")

  #_frontier.setIsProcessing(False)

  ctx.module("EPIdistortionCorrectionCore").call("switchToInitialLayout")

  ctx.module("EPIdistortionCorrectionCore").call("loadingDatasetCompleted"
     )


  #Allowing for computation in MeanOfPhases
  ctx.field("EPIdistortionCorrectionCore.
     EPIdistortionCorrectionProcessing_Custom.MeanOfPhasesReverse.
     Arithmetic5.i1").value = ctx.field("EPIdistortionCorrectionCore.
     EPIdistortionCorrectionProcessing_Custom.InfoReverse.sizeT").value
  ctx.field("EPIdistortionCorrectionCore.
     EPIdistortionCorrectionProcessing_Custom.MeanOfPhasesReverse.Bypass1.
     noBypass").value = False

  #Starting the next patient browser
  startImport3()

  return

# ——————————— Patient/Volume data 3

def startImport3():

  MLAB.log("starting MRPatientBrowser3")
  ctx.field("EPIdistortionCorrectionCore.MRPatientBrowser2.
     patientBrowserHasBeenOpened").value = 3

  # index DICOM data
  dataDirectory = _frontier.getIncomingDicomDirectory()
  ctx.field("EPIdistortionCorrectionCore.MRPatientBrowser3.frontierSourceDir").setValue(dataDirectory)
  ctx.field("EPIdistortionCorrectionCore.MRPatientBrowser3.import").touch()

  return

def importFinished3():

  global _frontier
  global _landscapeLayout

  MLAB.log("import finished 3")

  # Show patient browser to enable the user to select a volume.
  ctx.module("PatientBrowserPanel").call("showPatientBrowser")
  ctx.field("EPIdistortionCorrectionCore.MRPatientBrowser3.resetCheckBoxStatus").touch()
  # show patient browser
  if _landscapeLayout:
    _frontier.selectLayout("SINGLE_CA-LEFT_H(1)", ["controlArea", "outPatientBrowser"])
  else:
    _frontier.selectLayout("SINGLE_CA-BOTTOM_H(1)", ["controlArea", "outPatientBrowser"])

  # Popup window with information
  title = "Choose DWI images"
  text = 'Please select all forward phase encoded images to be corrected, including any DWI images (b > 0).\n\nIMPORTANT: Select ALL phases in the chosen image series! This is required for the DICOM tags to be
```

```python
    stored correctly after processing.\n\nImage series are shown in the upper panel, and the phases are shown in the lower panel when an image series is marked.\n\nNote that both an image series and all its phases must be checked in order to move on to the next step. All phases checked must also be from the same image series (only one series can be checked).'
  buttonNames = ["Ok"]

  if (ctx.field("EPIdistortionCorrectionCore.frontierEnvironment").value):
    frontier = None
    frontierAPI = ctx.field("EPIdistortionCorrectionCore.inSyngoVia").object()
    if (frontierAPI != None):
      frontierAPI.showMessageBox(title, text, buttonNames)
  else :
    MLAB.showInformation(text, title, buttonNames, 0)

  return

def loadingCompleted3():
  global _monitorConfiguration, _landscapeLayout

  MLAB.log("loading completed 3")

  #_frontier.setIsProcessing(False)

  ctx.module("EPIdistortionCorrectionCore").call("switchToInitialLayout")

  ctx.module("EPIdistortionCorrectionCore").call("loadingDatasetCompleted")
  ctx.field("EPIdistortionCorrectionCore.MRPatientBrowser2.patientBrowserHasBeenOpened").value = 0

```

```
# Initializing various parameters
ctx.field("EPIdistortionCorrectionCore.EPIdistortionCorrectionProcessing_Custom.Flip.value").value = "--no-flip"
ctx.field("EPIdistortionCorrectionCore.EPIdistortionCorrectionProcessing_Custom.Flip.FlipPressed").value = False
ctx.field("EPIdistortionCorrectionCore.EPIdistortionCorrectionProcessing_Custom.PhaseEncodeDirection.value").value = "--phase-encode-ap"
ctx.field("EPIdistortionCorrectionCore.EPIdistortionCorrectionProcessing_Custom.imagemath1.NumTimesDWICorrectionPerformed").value = 0
ctx.field("EPIdistortionCorrectionCore.EPIdistortionCorrectionProcessing_Custom.SubImage1.t").value = 0
ctx.field("EPIdistortionCorrectionCore.EPIdistortionCorrectionProcessing_Custom.ImageComposer.clear").touch()
 ctx.field("EPIdistortionCorrectionCore.Export2.imagemath1.NumTimesDWICorrectionPerformed").value = 0
ctx.field("EPIdistortionCorrectionCore.Export2.SubImage1.t").value = 0
ctx.field("EPIdistortionCorrectionCore.Export2.ImageComposer.clear").touch()

ctx.field("EPIdistortionCorrectionCore.Switch1.currentInput").value = 0
ctx.field("EPIdistortionCorrectionCore.Switch2.currentInput").value = 0
ctx.field("EPIdistortionCorrectionCore.Switch3.currentInput").value = 0
ctx.field("EPIdistortionCorrectionCore.Export2.Switch.currentInput").value = 0
ctx.field("EPIdistortionCorrectionCore.Export3.Switch.currentInput").value = 0

```

```
  ctx.field("EPIdistortionCorrectionCore.Switch1.mosaicPressed").value =
     False
  ctx.field("EPIdistortionCorrectionCore.Switch2.mosaicPressed").value =
     False
  ctx.field("EPIdistortionCorrectionCore.Switch3.mosaicPressed").value =
     False
  ctx.field("EPIdistortionCorrectionCore.Export2.Switch.mosaicPressed").
     value = False
  ctx.field("EPIdistortionCorrectionCore.Export3.Switch.mosaicPressed").
     value = False

  ctx.field("EPIdistortionCorrectionCore.Export2.Switch1.currentInput").
     value = 0

  if ctx.field("EPIdistortionCorrectionCore.Export2.InfoUncorr.sizeT").
     value == 1:    ctx.field("EPIdistortionCorrectionCore.Export2.
     imagemath1.NumTimesDWICorrectionPerformed").value = 1
  else:    ctx.field("EPIdistortionCorrectionCore.Export2.imagemath1.
     NumTimesDWICorrectionPerformed").value = 0

  return
```

A.2 EPIdistortionCorrectionPatientBrowserPanel.py

```
from mevis import *

g_layoutPatientBrowser = None
g_scenePatientBrowser = None
g_PatientBrowserGraphicsView = None

g_layoutPatientBrowser2 = None
g_scenePatientBrowser2 = None
```

```
g_PatientBrowserGraphicsView2 = None

def showPatientBrowser():
  global g_PatientBrowserGraphicsView
  global g_layoutPatientBrowser
  global g_scenePatientBrowser
  global g_PatientBrowserGraphicsView

  g_scenePatientBrowser = g_PatientBrowserGraphicsView.scene()
  contentArea = g_scenePatientBrowser.addLayer();

  g_layoutPatientBrowser = g_scenePatientBrowser.createGridLayout(
      contentArea)

  # always show patient browser
  # if—elif—else to distinguish between the three patient browsers
  if ctx.field("parent:EPIdistortionCorrectionCore.MRPatientBrowser2.
      patientBrowserHasBeenOpened").value == 1:
    mdl = g_scenePatientBrowser.addMDL ("Panel { module=parent:
        EPIdistortionCorrectionCore.MRPatientBrowser window=
        MRPatientBrowserRoles }")
    g_layoutPatientBrowser.addItem(mdl,0,0)
  elif ctx.field("parent:EPIdistortionCorrectionCore.MRPatientBrowser2.
      patientBrowserHasBeenOpened").value == 2:
    mdl = g_scenePatientBrowser.addMDL ("Panel { module=parent:
        EPIdistortionCorrectionCore.MRPatientBrowser2 window=
        MRPatientBrowserRoles }")
    g_layoutPatientBrowser.addItem(mdl,0,0)
  else:
    mdl = g_scenePatientBrowser.addMDL ("Panel { module=parent:
        EPIdistortionCorrectionCore.MRPatientBrowser3 window=
        MRPatientBrowserRoles }")
    g_layoutPatientBrowser.addItem(mdl,0,0)
```

A.3 EPIdistortionCorrectionProcessing_Custom.script

```
Interface  {
  Inputs {
    Field inImage1 { internalName = GateInImage1.input0 }
    Field inImage2 { internalName = GateInImage2.input0 }
    Field inImage3 { internalName = GateInImage3.input0 }
  }
  Outputs {
    Field outOriginalImage   { internalName = GateOutOriginalImage.output0 }
    Field outOriginalImage2   { internalName = GateOutOriginalImage2.output0 }
    Field outOriginalImage3   { internalName = GateOutOriginalImage3.output0 }
    Field outProcessedImage  { internalName = GateOutProcessedImage.output0 }
    Field outProcessedImage2 { internalName = GateOutProcessedImage2.output0 }
    Field outProcessedImage3 { internalName = GateOutProcessedImage3.output0 }
    Field outPanel1          { internalName = RenderUserPanelsExample.outPanel1 }
    Field outPanel2          { internalName = RenderUserPanelsExample.outPanel2 }
    Field outPanel3          { internalName = RenderUserPanelsExample.outPanel3 }
    Field outPanel4          { internalName = RenderUserPanelsExample.outPanel4 }
    Field outPanel5          { internalName = RenderUserPanelsExample.outPanel5 }
  }
  Parameters {
    Field workingDir                        { internalName = WorkingDir.value }

    // Trigger fields
    Field externalExecutableEpiunwarpStartedTrigger  { internalName = epiunwarp.executionStartedTrigger }
    Field externalExecutableEpiunwarpFinishedTrigger { internalName = epiunwarp.executionFinishedTrigger }
    Field externalExecutableReformatx1StartedTrigger  { internalName = reformatx1.executionStartedTrigger }
    Field externalExecutableReformatx1FinishedTrigger { internalName = reformatx1.executionFinishedTrigger }
    Field externalExecutableImagemath1StartedTrigger  { internalName = imagemath1.executionStartedTrigger }
    Field externalExecutableImagemath1FinishedTrigger { internalName = imagemath1.executionFinishedTrigger }
  }
}

Commands {
  source = $(LOCAL)/EPIdistortionCorrectionProcessing_Custom.py

  // FieldListeners to start the next iteration of distortion correction when the previous one has finished
  FieldListener externalExecutableReformatx1FinishedTrigger { command = StartImagemath }
  FieldListener externalExecutableImagemath1FinishedTrigger { command = NextIteration }
}

Window {
  fixedWidth  = 500
  fixedHeight = 200
  Category { expandY = Yes
    Empty { expandY = Yes }
    Label { title = "This is the main processing module." expandX = Yes
        textAlignment = Center }
    Label { title = "It is recommended to add all your custom modules and
        code here." expandX = Yes textAlignment = Center }
    Empty { expandY = Yes }
  }
}
```

A.4 EPIdistortionCorrectionProcessing_Custom.py

```
from mevis import *

# Starts the next iteration of the distortion correction. The 3D image
    selected by SubImage has already been incremented in the post−
    processing script of imagemath
def NextIteration():

  ctx.field("ImageComposer.add").touch()

  if ctx.field("imagemath1.NumTimesDWICorrectionPerformed").value < ctx.
      field("Info.sizeT").value:
    MLAB.log("starting to correct DWI image")
    ctx.field("reformatx1.startExecution").touch()
  else:
    MLAB.log("all fwd images are corrected")
    if ctx.field("InfoReverse.sizeT").value > 1:
      ctx.field("parent:Export2.Switch1.currentInput").value = 1
      CorrectRev()

  return

# Starts imagemath when reformatx has finished
def StartImagemath():

  ctx.field("imagemath1.startExecution").touch()

  return

# Corrects the reverse phase encoded b = 0 images if there are more than
    one
def CorrectRev():

  if ctx.field("parent:Export2.imagemath1.NumTimesDWICorrectionPerformed")
      .value < ctx.field("parent:Export2.InfoUncorr.sizeT").value:
    MLAB.log("starting to correct rev images")
    ctx.field("parent:Export2.reformatx1.startExecution").touch()
  else:
    MLAB.log("all rev images are corrected")

  return
```

A.5 EPIdistortionCorrectionCoreLayouts_Custom.py

```
from MRViewers import Layout, MonitorLayout

def initLayouts():

  #
  # Modify this function if you want to customize the available layouts.
  #
```

```

global g_controlArea

# Define the layouts

layout = Layout("Single_Screen_2x2", title = "Single 2x2")
firstMonitorLayout = MonitorLayout()
firstMonitorLayout.addViewer(1, 0, 0, 1, 1)
firstMonitorLayout.addViewer(2, 0, 1, 1, 1)
firstMonitorLayout.addViewer(3, 1, 0, 1, 1)
firstMonitorLayout.addViewer(4, 1, 1, 1, 1)
layout.setFirstMonitorLayout(firstMonitorLayout)
addLayout(layout)

layout = Layout("Single_Screen_2x3", title = "Single 2x3")
firstMonitorLayout = MonitorLayout()
firstMonitorLayout.addViewer(1, 0, 0, 1, 1)
firstMonitorLayout.addViewer(2, 0, 1, 1, 1)
firstMonitorLayout.addViewer(3, 0, 2, 1, 1)
firstMonitorLayout.addViewer(5, 1, 0, 1, 1)
firstMonitorLayout.addViewer(6, 1, 1, 1, 1)
firstMonitorLayout.addViewer(7, 1, 2, 1, 1)
layout.setFirstMonitorLayout(firstMonitorLayout)
addLayout(layout)

layout = Layout("Single_Screen_3+1_Static", title = "Single 3+1 (static)
    ")
firstMonitorLayout = MonitorLayout()
firstMonitorLayout.addViewer(1, 0, 0, 1, 1)
firstMonitorLayout.addViewer(2, 0, 2, 1, 1)
firstMonitorLayout.addViewer(3, 0, 1, 1, 1)
firstMonitorLayout.addViewer(4, 1, 0, 1, 3)
firstMonitorLayout.forceStaticLayout(True)
```

```
layout.setFirstMonitorLayout(firstMonitorLayout)
addLayout(layout)

layout = Layout("Single_Screen_3+2", title = "Single 3+2")
firstMonitorLayout = MonitorLayout()
firstMonitorLayout.addViewer(1, 0, 0, 1, 2)
firstMonitorLayout.addViewer(2, 0, 2, 1, 2)
firstMonitorLayout.addViewer(3, 0, 4, 1, 2)
firstMonitorLayout.addViewer(4, 1, 0, 1, 3)
firstMonitorLayout.addViewer(5, 1, 3, 1, 3)
layout.setFirstMonitorLayout(firstMonitorLayout)
addLayout(layout)

layout = Layout("Single_Screen_3+2_Static", title = "Single 3+2 (static)
    ")
firstMonitorLayout = MonitorLayout()
firstMonitorLayout.addViewer(1, 0, 0, 1, 2)
firstMonitorLayout.addViewer(2, 0, 2, 1, 2)
firstMonitorLayout.addViewer(3, 0, 4, 1, 2)
firstMonitorLayout.addViewer(4, 1, 0, 1, 3)
firstMonitorLayout.addViewer(5, 1, 3, 1, 3)
firstMonitorLayout.forceStaticLayout(True)
layout.setFirstMonitorLayout(firstMonitorLayout)
addLayout(layout)

layout = Layout("Complex", title = "Complex")
firstMonitorLayout = MonitorLayout()
firstMonitorLayout.addViewer(1, 0, 0, 3, 3)
firstMonitorLayout.addViewer(2, 0, 3, 3, 3)
firstMonitorLayout.addViewer(3, 3, 0, 3, 3)
firstMonitorLayout.addViewer(4, 3, 3, 3, 3)
firstMonitorLayout.addViewer(5, 6, 0, 2, 2) # Segment 5 is displayed in
    all of the small viewers...
```

```
firstMonitorLayout.addViewer(5, 6, 2, 2, 2)
firstMonitorLayout.addViewer(5, 6, 4, 2, 2)
firstMonitorLayout.addViewer(5, 6, 6, 2, 2)
firstMonitorLayout.addViewer(5, 0, 6, 2, 2)
firstMonitorLayout.addViewer(5, 2, 6, 2, 2)
firstMonitorLayout.addViewer(5, 4, 6, 2, 2)
firstMonitorLayout.forceStaticLayout(True)
layout.setFirstMonitorLayout(firstMonitorLayout)
addLayout(layout)

layout = Layout("Dual_Screen_1x1", title = "Dual 1x1")
firstMonitorLayout = MonitorLayout()
firstMonitorLayout.addViewer(1, 0, 0, 1, 1)
layout.setFirstMonitorLayout(firstMonitorLayout)
secondMonitorLayout = MonitorLayout()
secondMonitorLayout.addViewer(1, 0, 0, 1, 1)
layout.setSecondMonitorLayout(secondMonitorLayout)
addLayout(layout)

layout = Layout("Dual_Screen_1x3_1x1", title = "Dual 1x3 + 1x1")
firstMonitorLayout = MonitorLayout()
firstMonitorLayout.addViewer(1, 0, 0, 1, 1)
firstMonitorLayout.addViewer(2, 0, 1, 1, 1)
firstMonitorLayout.addViewer(3, 0, 2, 1, 1)
layout.setFirstMonitorLayout(firstMonitorLayout)
secondMonitorLayout = MonitorLayout()
secondMonitorLayout.addViewer(4, 0, 0, 1, 1)
layout.setSecondMonitorLayout(secondMonitorLayout)
addLayout(layout)

layout = Layout("Dual_Screen_2x2", title = "Dual 2x2")
firstMonitorLayout = MonitorLayout()
firstMonitorLayout.addViewer(1, 0, 0, 1, 1)
firstMonitorLayout.addViewer(2, 0, 1, 1, 1)
firstMonitorLayout.addViewer(3, 1, 0, 1 ,1)
firstMonitorLayout.addViewer(4, 1, 1, 1, 1)
layout.setFirstMonitorLayout(firstMonitorLayout)
secondMonitorLayout = MonitorLayout()
secondMonitorLayout.addViewer(1, 0, 0, 1, 1)
secondMonitorLayout.addViewer(2, 0, 1, 1, 1)
secondMonitorLayout.addViewer(3, 1, 0, 1 ,1)
secondMonitorLayout.addViewer(4, 1, 1, 1, 1)
layout.setSecondMonitorLayout(secondMonitorLayout)
addLayout(layout)

layout = Layout("Dual_Screen_1x2", title = "Dual 1x2")
firstMonitorLayout = MonitorLayout()
firstMonitorLayout.addViewer(1, 0, 0, 1, 1)
firstMonitorLayout.addViewer(2, 0, 1, 1, 1)
layout.setFirstMonitorLayout(firstMonitorLayout)
secondMonitorLayout = MonitorLayout()
secondMonitorLayout.addViewer(3, 0, 0, 1, 1)
secondMonitorLayout.addViewer(4, 0, 1, 1, 1)
secondMonitorLayout.forceStaticLayout(True)
layout.setSecondMonitorLayout(secondMonitorLayout)
addLayout(layout)

layout = Layout("Dual_Screen_2x1", title = "Dual 2x1")
firstMonitorLayout = MonitorLayout()
firstMonitorLayout.addViewer(1, 0, 0, 1, 1)
firstMonitorLayout.addViewer(2, 1, 0, 1, 1)
firstMonitorLayout.forceStaticLayout(True)
layout.setFirstMonitorLayout(firstMonitorLayout)
secondMonitorLayout = MonitorLayout()
secondMonitorLayout.addViewer(3, 0, 0, 1, 1)
secondMonitorLayout.addViewer(4, 1, 0, 1, 1)
```

```
  layout.setSecondMonitorLayout(secondMonitorLayout)
  addLayout(layout)

  # Set initial layout which is shown after an image is loaded depending
     on the number of monitors.
  # If you just provide single screen layouts you have pass it also to
     setInitialLayoutForDualMonitorConfiguration()
  setInitialLayoutForSingleMonitorConfiguration("Single_Screen_2x3")
  setInitialLayoutForDualMonitorConfiguration("Single_Screen_2x3")

  return
```

A.6 EPIdistortionCorrectionCoreProcessing_Custom.py

```
from mevis import *
from time import sleep

# Some global variables. Be careful when using them!
#
#global g_vortal                    # Reference to the vortal
#global g_controlArea               # Reference to the Control Area
#global g_secondaryWindow           # Reference to the secondary window
#global g_layouts                   # Reference to the layouts
#global g_lastSnapshotFolder        # Reference to the snapshot folder
global g_prototypeName             # Reference to the prototype name
global g_task_EPIdistortionCorrection   # Reference to the task
   specification

# ----------- User code ---------------
#
# Add own functionality here if needed.
#
```

```

# Calculate distortion correction
def ExecuteExternalAlgoButtonClicked1():

  ctx.field("EPIdistortionCorrectionProcessing_Custom.epiunwarp.
     startExecution").touch()

  return

# Correct images
def ExecuteExternalAlgoButtonClicked2():

  if ctx.field("EPIdistortionCorrectionProcessing_Custom.imagemath1.
     NumTimesDWICorrectionPerformed").value < ctx.field("
     EPIdistortionCorrectionProcessing_Custom.Info.sizeT").value:
    MLAB.log("starting to correct DWI images")        ctx.field("
       EPIdistortionCorrectionProcessing_Custom.reformatx1.startExecution"
       ).touch()
  else:
    MLAB.log("all fwd images are corrected")

  return

# Resolve mosaic images
def MosaicButtonClicked1():

  if ctx.field("Switch1.mosaicPressed").value == False:
    ctx.field("Switch1.currentInput").value = 1
    ctx.field("Switch1.mosaicPressed").value = True
  else:
    ctx.field("Switch1.currentInput").value = 0
    ctx.field("Switch1.mosaicPressed").value = False

  return

def MosaicButtonClicked2():

  if ctx.field("Switch2.mosaicPressed").value == False:
    ctx.field("Switch2.currentInput").value = 1
    ctx.field("Switch2.mosaicPressed").value = True
    ctx.field("Export2.Switch.currentInput").value = 1
    ctx.field("Export2.Switch.mosaicPressed").value = True
  else:
    ctx.field("Switch2.currentInput").value = 0
    ctx.field("Switch2.mosaicPressed").value = False
    ctx.field("Export2.Switch.currentInput").value = 0
    ctx.field("Export2.Switch.mosaicPressed").value = False

  return

def MosaicButtonClicked3():

  if ctx.field("Switch3.mosaicPressed").value == False:
    ctx.field("Switch3.currentInput").value = 1
    ctx.field("Switch3.mosaicPressed").value = True
    ctx.field("Export3.Switch.currentInput").value = 1
    ctx.field("Export3.Switch.mosaicPressed").value = True
  else:
    ctx.field("Switch3.currentInput").value = 0
    ctx.field("Switch3.mosaicPressed").value = False
    ctx.field("Export3.Switch.currentInput").value = 0
    ctx.field("Export3.Switch.mosaicPressed").value = False

  return

# Distortion correction parameter (is the reverse image flipped)
```

```
def FlipButtonClicked():

  if ctx.field("EPIdistortionCorrectionProcessing_Custom.Flip.FlipPressed").value == False:
    ctx.field("EPIdistortionCorrectionProcessing_Custom.Flip.value").value = ""
    ctx.field("EPIdistortionCorrectionProcessing_Custom.Flip.FlipPressed").value = True
  else:
    ctx.field("EPIdistortionCorrectionProcessing_Custom.Flip.value").value = "--no-flip"
    ctx.field("EPIdistortionCorrectionProcessing_Custom.Flip.FlipPressed").value = False

  return

# Set the phase encoding direction parameter
def PhaseEncodeAP():

  ctx.field("EPIdistortionCorrectionProcessing_Custom.PhaseEncodeDirection.value").value = "--phase-encode-ap"

  return

def PhaseEncodeIS():

  ctx.field("EPIdistortionCorrectionProcessing_Custom.PhaseEncodeDirection.value").value = "--phase-encode-is"

  return

def PhaseEncodeLR():

```

```python
  ctx.field("EPIdistortionCorrectionProcessing_Custom.PhaseEncodeDirection.value").value = "--phase-encode-lr"

  return

# Save images locally
def saveResultsOffline():

  ctx.field("seriesUID.create").touch()


  if ctx.field("Export3.Switch.mosaicPressed").value == True:
    makeMosaicFwd()

  ctx.field("Export3.DicomTool3.saveSlices").touch()


  if ctx.field("Export2.Switch.mosaicPressed").value == True:
    makeMosaicRev()

  ctx.field("Export2.DicomTool2.saveSlices").touch()

  return

# Make a mosaic image
def makeMosaicRev():

  MLAB.log("Running makeMosaicRev()")

  ctx.field("Export2.MakeMosaic.ImageComposer.clear").touch()

  numTilesX = (ctx.field("Export2.InfoUncorr.sizeX").value)/(ctx.field("Export2.InfoCorr.sizeX").value)
```

```
numTilesY = (ctx.field("Export2.InfoUncorr.sizeY").value)/(ctx.field("
   Export2.InfoCorr.sizeY").value)
numSlices = ctx.field("Export2.InfoCorr.sizeZ").value
numPhases = ctx.field("Export2.InfoCorr.sizeT").value

xJump = ctx.field("Export2.InfoCorr.sizeX").value
yJump = ctx.field("Export2.InfoCorr.sizeY").value

tPos = 0
xPos = 0
yPos = 0
zPos = 0
cPos = 0
uPos = 0

zSlice = 0

for t in range(0,numPhases):
  ctx.field("Export2.MakeMosaic.SubImage.t").value = t
  for y in range(0,numTilesY):
    for x in range(0,numTilesX):
      ctx.field("Export2.MakeMosaic.ImageComposer.userPos").value = [
         xPos,yPos,zPos,cPos,tPos,uPos]
      ctx.field("Export2.MakeMosaic.SubImage.z").value = zSlice
      ctx.field("Export2.MakeMosaic.ImageComposer.add").touch()
      zSlice += 1
      xPos += xJump
    yPos += yJump
    xPos = 0
  tPos += 1
  yPos = 0
  xPos = 0
  zSlice = 0
  MLAB.log("makeMosaicRev() finished")

  return

def makeMosaicFwd():

  MLAB.log("Running makeMosaicFwd()")

  ctx.field("Export3.MakeMosaic.ImageComposer.clear").touch()

  numTilesX = (ctx.field("Export3.InfoUncorr.sizeX").value)/(ctx.field("
     Export3.InfoCorr.sizeX").value)
  numTilesY = (ctx.field("Export3.InfoUncorr.sizeY").value)/(ctx.field("
     Export3.InfoCorr.sizeY").value)
  numSlices = ctx.field("Export3.InfoCorr.sizeZ").value
  numPhases = ctx.field("Export3.InfoCorr.sizeT").value

  xJump = ctx.field("Export3.InfoCorr.sizeX").value
  yJump = ctx.field("Export3.InfoCorr.sizeY").value

  tPos = 0
  xPos = 0
  yPos = 0
  zPos = 0
  cPos = 0
  uPos = 0

  zSlice = 0

  for t in range(0,numPhases):
    ctx.field("Export3.MakeMosaic.SubImage.t").value = t
    for y in range(0,numTilesY):
      for x in range(0,numTilesX):
        ctx.field("Export3.MakeMosaic.ImageComposer.userPos").value = [xPos,yPos,zPos,cPos,tPos,uPos]
        ctx.field("Export3.MakeMosaic.SubImage.z").value = zSlice
        ctx.field("Export3.MakeMosaic.ImageComposer.add").touch()
        zSlice += 1
        xPos += xJump
      yPos += yJump
      xPos = 0
    tPos += 1
    yPos = 0
    xPos = 0
    zSlice = 0
  MLAB.log("makeMosaicFwd() finished")

  return
```

A.7 EPIdistortionCorrectionCoreControlArea_Custom.py

```
global g_task_EPIdistortionCorrection

def initControlArea(controlArea):

  global g_task_EPIdistortionCorrection

  # Set the prototype name.
  # In standalone, the passed name is used as task flow label and as window title.
  # In frontier environment, the passed name is not used, instead as task flow label 'MM Research Frontier' is used by default.
  setPrototypeName("EPI distortion correction")

  # Add task and set its label.
  #g_task_EPIdistortionCorrection = addTaskFromUIModule(controlArea)  # Use this line to automatically generate the Control Area from the EPIdistortionCorrectionControlAreaUI_Custom module of the internal network.
  g_task_EPIdistortionCorrection = addTaskManually(controlArea)      # Use this line to generate the Control Area manually. You need to edit the AddTaskManually() method to actually add controls.

  # Add the common controls which are shown on the bottom of the control area.
  # If a control is not connected, it will not be shown
  # The connected functions are defined in EPIdistortionCorrectionCoreCommonControlsCommands_Custom.py
  commonControls = controlArea.addCommonControls()
  commonControls.help.connect("clicked",           showHelp)
  commonControls.reset.connect("clicked",          resetViewers)
  commonControls.snapshot.connect("clicked",       createSnapshots)
#  commonControls.sendResults.connect("clicked",    sendResults)
  commonControls.toggleRefLines.connect("clicked", toggleRefLines)
  commonControls.toggleText.connect("clicked",     toggleImageText)

  return g_task_EPIdistortionCorrection

def addTaskManually(controlArea):

  # do not change this line. It adds the main task
  task = addTaskAndInitKnightRider(controlArea, "EPI distortion correction", canCollapse = False)

  # Add information step:
  newStep = task.addStep("Information", idNr="myStep0")
  newStep.setCollapsed(True) # Specifies whether the task is initially collapsed
  newStep.connect("expanded", ExampleTaskExpanded)
  newStep.addMDL("Label { title = @@This prototype corrects geometric distortion in EPI images.\n\nThe layout of the screen can be chosen using the button in the upper right corner of the menu. Using the default 'Single 2x3' layout, the unprocessed images will be shown in the upper row, and the processed images will be shown in the bottom row. In order to observe the results, this layout should be used.\n\nIt is important that steps 1–4 are performed in the correct order. However, which images that are shown in the top and bottom row can be changed at any time.\n\nPlease make sure that all parameters are correctly adjusted before performing the distortion correction, to avoid errors in the resulting images.@@ fixedWidth = 266 textWrap = WordBreak }", 24, 12)

  # Add "choose image" step:
  newStep = task.addStep("Choose image in top panel", idNr="myStep1")
  newStep.setCollapsed(True) # Specifies whether the task is initially collapsed
  newStep.connect("expanded", ExampleTaskExpanded)

  # Choose which image to render in the top panel:
  paletteButton = newStep.addPaletteButton((0, 0))
  iconPath1 = ctx.expandFilename("$(LOCAL)/Graphics/ButtonF.png")
  subButton1 = paletteButton.addButton(iconPath1, title = "Forward phase encoded image")
  subButton1.connect("clicked", lambda: ctx.field("SwitchUnprocessedView.currentInput").setIntValue(0))
  iconPath2 = ctx.expandFilename("$(LOCAL)/Graphics/ButtonR.png")
  subButton2 = paletteButton.addButton(iconPath2, title = "Reverse phase encoded image")
  subButton2.connect("clicked", lambda: ctx.field("SwitchUnprocessedView.currentInput").setIntValue(1))
  iconPath3 = ctx.expandFilename("$(LOCAL)/Graphics/ButtonD.png")
```

```
subButton3 = paletteButton.addButton(iconPath3, title = "All forward
    phase encoded (including DWI) images")
subButton3.connect("clicked", lambda: ctx.field("SwitchUnprocessedView.
    currentInput").setIntValue(2))
newStep.addMDL("Label { title = @@Choose which uncorrected image to view
     in the top panel.@@ fixedWidth = 200 textWrap = WordBreak }", 90,
    12)

# Add "choose image" step:
newStep = task.addStep("Choose image in bottom panel", idNr="myStep2")
newStep.setCollapsed(True) # Specifies whether the task is initially
    collapsed
newStep.connect("expanded", ExampleTaskExpanded)

# Choose which image to render in the top panel:
paletteButton = newStep.addPaletteButton((0, 0))
iconPath1 = ctx.expandFilename("$(LOCAL)/Graphics/ButtonF.png")
subButton1 = paletteButton.addButton(iconPath1, title = "Forward phase
    encoded image")
subButton1.connect("clicked", lambda: ctx.field("SwitchResultView.
    currentInput").setIntValue(0))
iconPath2 = ctx.expandFilename("$(LOCAL)/Graphics/ButtonR.png")
subButton2 = paletteButton.addButton(iconPath2, title = "Reverse phase
    encoded image")
subButton2.connect("clicked", lambda: ctx.field("SwitchResultView.
    currentInput").setIntValue(1))
iconPath3 = ctx.expandFilename("$(LOCAL)/Graphics/ButtonD.png")
subButton3 = paletteButton.addButton(iconPath3, title = "All forward
    phase encoded (including DWI) images")
subButton3.connect("clicked", lambda: ctx.field("SwitchResultView.
    currentInput").setIntValue(2))
newStep.addMDL("Label { title = @@Choose which corrected image to view
    in the bottom panel.@@ fixedWidth = 200 textWrap = WordBreak }", 90,
    12)

# Add parameter selection step:
newStep = task.addStep("1. Parameter selection", idNr="myStepParameters")
newStep.setCollapsed(True) # Specifies whether the task is initially collapsed
newStep.connect("expanded", ExampleTaskExpanded)

button = newStep.addToggleButton(ctx.expandFilename("$(LOCAL)/Graphics/ButtonMosaic.png"), "Mosaic (Fwd)", (0, 0))
button.connect("clicked", lambda : MosaicButtonClicked1())
newStep.addMDL("Label { title = @@Press this button if the forward phase encoded image is a mosaic.@@ fixedWidth = 200 textWrap = WordBreak }", 90, 12)

button = newStep.addToggleButton(ctx.expandFilename("$(LOCAL)/Graphics/ButtonMosaic.png"), "Mosaic (Rev)", (0, 1))
button.connect("clicked", lambda : MosaicButtonClicked2())
newStep.addMDL("Label { title = @@Press this button if the reverse phase encoded image is a mosaic.@@ fixedWidth = 200 textWrap = WordBreak }", 90, 86)

button = newStep.addToggleButton(ctx.expandFilename("$(LOCAL)/Graphics/ButtonMosaic.png"), "Mosaic (DWI)", (0, 2))
button.connect("clicked", lambda : MosaicButtonClicked3())
newStep.addMDL("Label { title = @@Press this button if alle the forward phase encoded (including DWI) images are mosaics.@@ fixedWidth = 200 textWrap = WordBreak }", 90, 160)

button = newStep.addToggleButton(ctx.expandFilename("$(LOCAL)/Graphics/ButtonFlip.png"), "Flipped", (0, 3))
button.connect("clicked", lambda : FlipButtonClicked())
```

```
  newStep.addMDL("Label { title = @@Press this button if the reverse phase
      encoded image is flipped relative to the forward phase encoded image
     .@@ fixedWidth = 200 textWrap = WordBreak }", 90, 232)

  paletteButton = newStep.addPaletteButton((0,4))
  iconPath1 = ctx.expandFilename("$(LOCAL)/Graphics/ButtonAP.png")
  subButton1 = paletteButton.addButton(iconPath1, title = "Anterior−
     posterior")
  subButton1.connect("clicked", lambda : PhaseEncodeAP())
  iconPath2 = ctx.expandFilename("$(LOCAL)/Graphics/ButtonIS.png")
  subButton2 = paletteButton.addButton(iconPath2, title = "Inferior−
     superior")
  subButton2.connect("clicked", lambda : PhaseEncodeIS())
  iconPath3 = ctx.expandFilename("$(LOCAL)/Graphics/ButtonLR.png")
  subButton3 = paletteButton.addButton(iconPath3, title = "Left−right")
  subButton3.connect("clicked", lambda : PhaseEncodeLR())
  newStep.addMDL("Label { title = @@Choose the phase encoding direction of
      the images.@@ fixedWidth = 200 textWrap = WordBreak }", 90, 308)

  #Choose the image type, to determine certain parameters.
#  paletteButton = newStep.addPaletteButton((0, 5))
#  iconPath1 = ctx.expandFilename("$(LOCAL)/Graphics/ButtonB.png")
#  subButton1 = paletteButton.addButton(iconPath1, title = "Breast images
   ")
#  subButton1.connect("clicked", lambda : ImageTypeBreast())
#  iconPath2 = ctx.expandFilename("$(LOCAL)/Graphics/ButtonP.png")
#  subButton2 = paletteButton.addButton(iconPath2, title = "Prostate
   images")
#  subButton2.connect("clicked", lambda : ImageTypeProstate())
#  iconPath3 = ctx.expandFilename("$(LOCAL)/Graphics/ButtonO.png")
#  subButton3 = paletteButton.addButton(iconPath3, title = "Other")
#  subButton3.connect("clicked", lambda : ImageTypeOther())
#  newStep.addMDL("Label { title = @@Choose the image type.@@ fixedWidth =
     200 textWrap = WordBreak }", 90, 382)

  # Add "calculate distortion correction" step
  step = task.addStep("2. Calculate distortion correction")
  step.setCollapsed(True)
  iconPath = ctx.expandFilename("$(LOCAL)/Graphics/ExecuteButton.png")
  button = step.addButton(iconPath, "Correct", (0, 0), name="
     buttonExecuteExternal")
  button.setToolTip("Click here to calculate distortion correction.")
  button.connect("clicked", lambda : ExecuteExternalAlgoButtonClicked1())
  step.addMDL("Label { title = @@Perform EPI distortion correction on
     forward and reverse phase encoded images by pressing the 'Correct'
     button.\n\nThe calculations may take a few minutes.@@ fixedWidth =
     200 textWrap = WordBreak }", 90, 12)

  # Add "correct images" step
  step = task.addStep("3. Correct images")
  step.setCollapsed(True)
  iconPath = ctx.expandFilename("$(LOCAL)/Graphics/ExecuteButton.png")
  button = step.addButton(iconPath, "Correct", (0, 0), name="
     buttonExecuteExternal2")
  button.setToolTip("Click here to correct forward phase encoded images.")
  button.connect("clicked", lambda : ExecuteExternalAlgoButtonClicked2())
  step.addMDL("Label { title = @@Perform EPI distortion correction on all
     forward phase encoded (including DWI) images, using the distortion
     field calculated in the previous step.@@ fixedWidth = 200 textWrap =
     WordBreak }", 90, 12)
  panelDefinition  = 'Horizontal { expandX = Minimum alignX = Left '
  panelDefinition += '  Label { titleField =
     EPIdistortionCorrectionProcessing_Custom.imagemath1.
     NumTimesDWICorrectionPerformed alignX = Left } '
  panelDefinition += '  Label { title = " of " } '
  panelDefinition += '  Label { titleField =
      EPIdistortionCorrectionProcessing_Custom.Info.sizeT alignX = Left } '
  panelDefinition += '  Label { title = " forward images corrected." } '
  panelDefinition += '}'
  step.addMDL(panelDefinition, 90, 100)
  step.addMDL("Label { title = @@To scroll through the images, use the
     left and right arrow keys after clicking on the image.@@ fixedWidth =
      200 textWrap = WordBreak }", 90, 124)
  step.addMDL("Label { title = @@If there are more than one reverse phase
     encoded image, they will be corrected after the correction of all the
      forward phase encoded images is finished.@@ fixedWidth = 200
     textWrap = WordBreak }", 90, 182)
  panelDefinition  = 'Horizontal { expandX = Minimum alignX = Left '
  panelDefinition += '  Label { titleField = Export2.imagemath1.
     NumTimesDWICorrectionPerformed alignX = Left } '
  panelDefinition += '  Label { title = " of " } '
  panelDefinition += '  Label { titleField = Export2.InfoUncorr.sizeT
     alignX = Left } '
  panelDefinition += '  Label { title = " reverse images corrected." } '
  panelDefinition += '}'
  step.addMDL(panelDefinition, 90, 268)

  # Add "save results" step:
  newStep = task.addStep("4. Save results", idNr="myStepSave")
  newStep.setCollapsed(True) # Specifies whether the task is initially
     collapsed
  newStep.connect("expanded", ExampleTaskExpanded)
  iconPath = ctx.expandFilename("$(LOCAL)/Graphics/saveImage.png")
  button = newStep.addButton(iconPath, "Send results (Fwd)", (0, 0), name=
     "buttonSendResults3")
  button.connect("clicked", lambda : sendResultsFwd())
  newStep.addMDL("Label { title = @@Send all the corrected forward phase
     encoded (including DWI) images back to Syngo.via.@@ fixedWidth = 200
    textWrap = WordBreak }", 90, 16)
  button = newStep.addButton(iconPath, "Send results (Rev)", (0, 1), name=
    "buttonSendResults2")
  button.connect("clicked", lambda : sendResultsRev())
  newStep.addMDL("Label { title = @@Send the corrected reverse phase
    encoded image(s) back to Syngo.via.@@ fixedWidth = 200 textWrap =
    WordBreak }", 90, 90)

  # Add "save results locally" step:
#  newStep = task.addStep("4. Save results (offline)", idNr="
    myStepSaveOffline")
#  newStep.setCollapsed(True) # Specifies whether the task is initially
    collapsed
#  newStep.connect("expanded", ExampleTaskExpanded)
#  button = newStep.addButton(iconPath, "Save results", (0, 0), name="
    buttonSendResults1")
#  button.connect("clicked", lambda : saveResultsOffline())
#  newStep.addMDL("Label { title = @@Save the corrected images.@@
    fixedWidth = 200 textWrap = WordBreak }", 90, 16)
```

Appendix B

Feedback form

Attached is the feedback form used for user-friendliness testing of the prototype.

Testing of EPI distortion correction prototype

Please read the instructions of the prototype carefully.

What is your profession? ____

How long time did it take to use the application, from start to end? _____________________

What do you think about the amount of time it took to correct the images?

☐ Okay ☐ Too long

Were you able to correct the images? ☐ Yes ☐ No

How easy was it to understand how to correct them? (1 = very difficult, 5 = very easy)

☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

If anything about the correction process was difficult to understand, what was it? ________

____

____

____

Additional feedback about the correction process: ___________________________________

____

____

____

How intuitive was it to understand the user-interface and to navigate in the prototype? (1 = very difficult, 5 = very easy)

☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

If anything in the user-interface was difficult to understand, what was it? _______________

____

____

____

Additional feedback about the user-interface: ______________________________________

____

____

____

Is this a prototype that you could have used in your workflow? ☐ Yes ☐ No

Why/why not? ____

____

____

____

Other comments: ____

____

____

____