UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR MEDIZINISCHE INFORMATIK

Aus dem Institut für Medizinische Informatik der Universität zu Lübeck
Direktor: Prof. Dr. rer. nat. habil. Heinz Handels

Fast MRI registration leveraging k-space properties

Schnelle MRT-Registrierung mittels k-Raum Eigenschaften

Masterarbeit
im Rahmen des Studienganges Medizinische Informatik
der Universität zu Lübeck

vorgelegt von
Jan Meyer

ausgegeben und betreut von
Prof. Dr. Mattias Heinrich

mit Unterstützung von
Ziad Al-Haj Hemidi, Eytan Kats

Lübeck, den 1. Januar 2025

Ich versichere an Eides statt, die vorliegende Arbeit selbstständig und nur unter Benutzung der angegebenen Hilfsmittel angefertigt zu haben.

Lübeck, den 1. Januar 2025

Abstract

Something about MRI and registration

Kurzfassung

Irgendwas über MRT und Registration

Contents

Chapter 1

Introduction

Magnetic Resonance Imaging (MRI) is a commonly used medical imaging technique based on measuring an magnetic field. It has a lot of benefits because it is non-invasive, radiation-free and has great contrast for soft tissue. However, speed is its main weakness as full-body scan can take up to 30 minutes. This can be a burden on patients due to needing to remain still for a long amount of time in the scanner as well as hindering efficiency in terms of patients per day. Thus, MRI acquisition is usually accelerated by subsampling the k-space in which the raw MR data is recorded. This can, however, lead to problematic image artifacts that hinder many processing algorithms. Another problem of the slow data acquisition are motion artifacts which are common for organs like lung and heart. While breathing can be controlled for a short amount of time, cardiac motion is involuntarily and should not be stopped.
These challenges have been traditionally addressed with computationally intensive algorithms that often iterativly solve an optimization problem. This however, also requires a lot of time and computational resources in post-processing. New approaches based on deep neural networks promise to rectify these challenges. Neural networks have seen a rise in popularity in the recent years in many fields such as image processing. While segmentation networks were established quite early, registration networks and even networks for aliasing-free MR reconstruction have recently come into focus.
In this thesis, the possibility of using an unsupervised deep learning approach to align subsampled MRI data as well its potential usage in a motion reconstruction pipeline will be investigated.

1.1 Challenges of MRI Acquisition

To motivate the research in this thesis, the challenges of MR imaging need to further be analyzed.

1.2 Related Work

The network used for the main parts of this thesis are are *Fourier-Net* [**Fourier-Net**] and *Fourier-Net+* [**Fourier-Net+**]. These are based on works such as the unsupervised *VoxelMorph* [**Voxelmorph**], which is also used as an comparison, as well as the *IC-Net* [**IC-Net**] and *SYM-Net* [**SYM-Net**]. Additionally, the traditional registration

algorithm *NiftyReg* [**NiftiReg**] was also used in the evaluation.
There are many works on image registration, though the field of medical image registration with deep learning is more limited. A brief overview of registration methods, fundamentals of deep learning with already existing networks for image registration as well as covering potential applications and challenges can be found in [**Chen2020**, **Haskins2020**, **Fu2020**, **Zou2022**, **Chen2023**]. Information on MRI in general, including MRI sampling and acceleration, can be found in [**Serai2021**, **SamplingStrategies**, **PulseSequences**, **AdvancesPI**, **CS-MRI**].
For MRI reconstruction, a lot of work is based on traditional iterative algorithms [**AdvancesPI**, **CS-MRI**, **ParallelMRI**, **GRAPPA**], however, there exist also deep learning approaches [**DeepMRIReconstructionRadialSubsampling**, **DeepMRIReconstructionSubsampli**
addressing the problem. While motion in time-series data if often corrected after the reconstruction, an interesting use-case it the motion-compensated reconstruction with can be achieved by combining motion estimation with reconstruction. Both of these parts can be either a traditional algorithm [**GRICS**] or neural networks [**Pan2024**, **Zou2024**].

1.3 Contributions and Structure of the Thesis

Expanding already established work [**Fourier-Net**, **Fourier-Net+**], unsupervised neural network training was used to efficiently align subsampled MR images. While the networks architecture was already tested on e.g. inter-patient brain scan alignment, a new use-case with undersampled cardiac MR scan was explored. The registration performance was also compared to other methods. Furthermore, a potential application in a motion-compensated image reconstruction pipeline was tested.
In chapter 2, the foundations of the thesis are introduced. This includes general information about MRI such as basic magnetic principles in section 2.1.1 image acquisition (section 2.1.2), acceleration (section 2.1.3), and reconstruction (section 2.1.4). Building of the latter, image transformations (section 2.2.1) and registration (section 2.2.2) are introduced followed by deep learning basics such as common architectures (section 2.3.1), their use in image registration (section 2.3.2) and general principles of network training and testing (section 2.3.3).
In chapter 3, the materials and methods of the thesis are explained. This includes the specific network architectures in section 3.1, namely *Fourier-Net* (section 3.1.1) and *Fourier-Net+* (section 3.1.2), followed by the datasets in section3.2. Lastly, the conducted experiments are explained in section 3.3 with results and discussion following in chapters 4 and 5 as well as a brief summary in chapter 6.

Chapter 2

Fundamentals

In this chapter, the foundations of the thesis are explained. Starting with the principles behind MRI acquisition, acceleration and reconstruction, it also covers image registration and deep learning basics.

2.1 Magnetic Resonance Imaging

MRI is a non-invasive, radiation-free, tomographic imaging technology based on measurements of a magnetic field. An MRI machine comprises four main components, as seen in Figure 2.1. The first component is a strong magnet powerful enough to generate a static magnetic field B_0 that is required to induce nuclear proton polarization. The second is a radio frequency (RF) system which generates an alternating magnetic field B_1 at the resonant frequency f and detects the MR signal that is returned from the patient. The third component is the set of gradient systems (oriented orthogonally in X, Y and Z directions) that generates linear magnetic field variations, which are then superimposed upon B_0 and are used to spatially encode the MR signal. In clinical MR scanners, the three gradient sets and whole-body RF coils are typically concentrically positioned inside the bore of the magnet. The fourth component is a computer providing the user interface, generating images to be displayed and interpreted on the console [**Serai2021**].

2.1.1 Magnetic Excitation and Relaxation

In the following section, the basic principles underlying MR imaging are presented. These include the relation between RF excitation and relaxation.

Radio Frequency Excitation

Individual nuclei precess around the field B_0 at a resonance frequency known as the Larmor frequency ω_0 of the net magnetization vector, which generally occurs in the RF range of the electromagnetic spectrum and is related to the external magnetic field as:

$$\omega_0 = \gamma \cdot B_0, \tag{2.1}$$

with γ being the gyromagnetic ratio, which is a fixed value depending on the nuclei [**SamplingStrategies**]. When nuclei are placed in the presence of a strong static

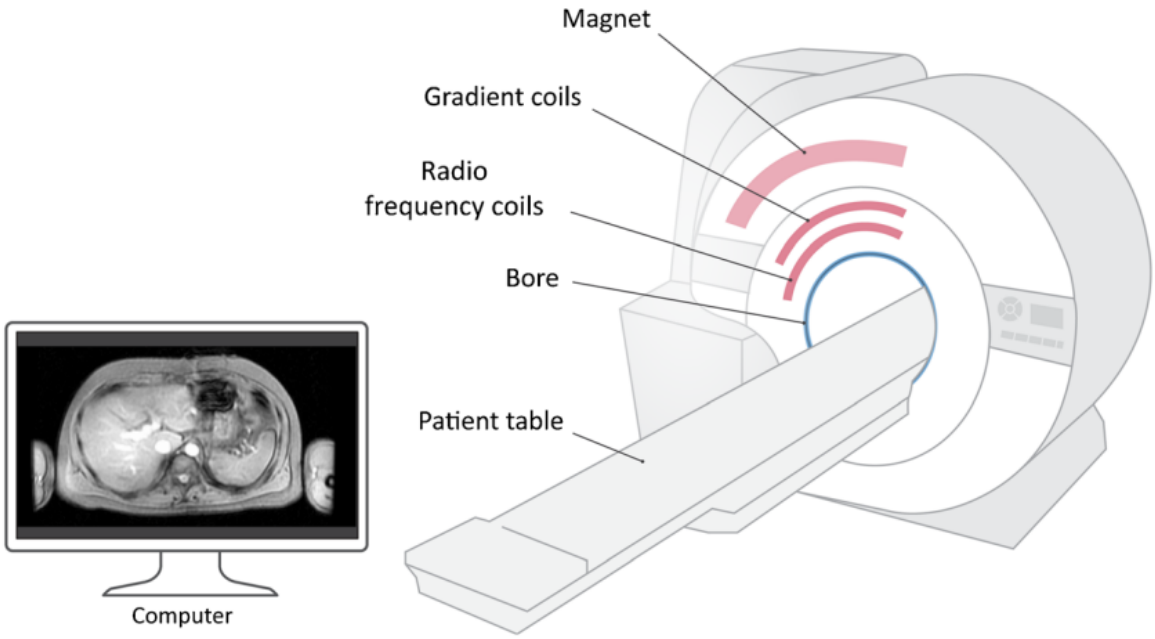


Figure 2.1: Schematic of an MRI scanner taken from [**Serai2021**].

magnetic field such as B_0 the nuclei split into two energy states, either aligned parallel to the magnetic field B_0 (called a spin-up state) or aligned anti-parallel to B_0 (called a spin-down state). The spin-up state has a slightly lower energy level as compared to the spin-down state and is therefore preferred. This slight difference in the spin states (0.001%) results in an overall net magnetization M aligned in the same direction as B_0. To create an MR signal, the spins are excited out of their resting equilibrium, i.e. tipping M away from B_0. To detect the signal from the hydrogen nuclei in the tissues an additional external field B_1 is introduced at the resonant Larmor frequency ω_0 that can affect magnetization vector, causing it to rotate into a plane orthogonal to its original orientation. The rotated vector continues to precess around the B_0. The precession of the magnetization vector in the transverse plane can be detected by an RF coil tuned to the resonant frequency ω_0. RF coils can be operated in a receive-only mode, in which case the inherent body coil is used as a transmitter; or the RF coils can be both transmit and receive. The purpose of the RF transmit coil is to create a time-varying B_1 field at right angles to B_0 that could be linearly or circularly polarized. The closer the receiving coil is to the source of the MR signal, the better the signal-to-noise ratio (SNR). Receiver coils generally comprise arrays of smaller individual coils or elements; however, each individual coil element has a limited depth penetration. Multiple arrays of coils, termed phased-array coils, can be used together to achieve a higher coverage. The multiple coils are electronically decoupled from one another so that they do not appear as just a single large coil. The images from individual coils are independently reconstructed and then grouped together to create the final image. An RF pulse of amplitude B_1, called excitation pulse, is applied for a certain time duration to tip the magnetization at an angle away from the B_0 field. The precessing transverse magnetization induces a voltage in the receiver RF coil; this induced voltage is known as the

free induction decay. After the pulse, the magnetization returns to thermal equilibrium by processes known as MR relaxation. To fully encode the spatial information within the field of view (FOV), pulse sequences must be iterated numerous times. The time between successive iterations of a pulse sequence is known as the repetition time (TR). The time between the application of the initial RF pulse and the middle of the detected echo is known as the echo time (TE). Due to this the overall time to acquire an MR image is quite long [**Serai2021**].

Magnetic Relaxation

Once the RF pulse is turned off, M continues to precess as it returns to its thermal equilibrium state. During this time, two types of relaxation occur: T_1 (longitudinal or spin lattice) and T_2 (transverse or spin/spin). One attribute that makes T_1 and T_2 so valuable for determining the signal in MRI is their sensitivity to the presence and type of tissue. It is this tissue dependence property that gives MR its excellent soft-tissue contrast. T_1 relaxation describes the recovery of the longitudinal magnetization back to thermal equilibrium following a perturbation by an RF pulse. The longitudinal component regrows along the Z direction with a time constant T_1. In other words, after the RF pulse is turned off, the protons that were disturbed give their energy to the surrounding environment and return back to their original equilibrium state, realigning with B_0. Hence T_1 relaxation is also called longitudinal relaxation. Furthermore, because T_1 relaxation involves the loss of energy that was put into the spin system by the RF pulse, it is also referred to as spin-lattice relaxation, the lattice consisting of surrounding macromolecules. This loss of energy is stimulated by the fluctuating magnetic fields associated with the dipole–dipole interactions of neighboring magnetic moments. T_1 relaxation can only occur when these magnetic field fluctuations occur at the resonant frequency ω_0. The rate at which the spin magnetization M_z recovers to M_0 at time t is called T_1 relaxation time. It can be expressed as follows:

$$M_z = M_0 \cdot \left(1 - e^{-\frac{t}{T_1}}\right). \tag{2.2}$$

A preferred method of measuring T_1 relaxation is the Look-Locker method, which is based on the principle that one does not need to wait for the net magnetization vector to equilibrate in order to measure T_1. Instead, an RF pulse is used with a small flip, which can be repeatedly applied. The acquisition pulse sequence is designed to generate a train of signals that gradually approaches a steady-state recovery. The recovery curve measured by this technique can be fitted to an exponential curve to provide an effective T_1 measurement. This has become one of the most popular T_1-mapping method for abdomen and cardiac imaging. The general principle of T_1 mapping is to acquire multiple images with different T_1 weightings and to fit the signal intensities of the images to the equation of T_1 relaxation. T_1 relaxation values for tissues can be estimated by fitting the data to the following equation:

$$S = S_0 \cdot \left(1 - A \cdot e^{\frac{\mathrm{TI}}{T_1}}\right), \tag{2.3}$$

where S is the signal intensity measured at each inversion time value TI, S_0 is the initial signal intensity at time $t = 0$ and A is a constant.
T_2 relaxation results in the loss of transverse magnetization caused by interactions between the magnetic fields of neighboring hydrogen nuclei. It is not an energy loss process like T_1 but is a loss of phase coherence within the spin system. This process, also known as spin-spin relaxation, leads to the destruction of transverse magnetization and causes the magnetic moments of the tissue to dephase. T_2 mapping is a method of measuring the T_2 value of the tissue. T_2 relaxation time can be calculated using a T_2 sequence with different echo times (TEs). The most fundamental sequence for T_2 mapping is signal measured with spinecho techniques (multiple sequences with different TE values) [19]. Other 2-D sequences have been used, such as multi-echo spin echo (MSME) and fast spin echo (FSE) [19]. Synthetic MRI is another quantitative method in which a single saturation recovery turbo spin-echo sequence is used to estimate T_2 transverse relaxation [20]. T_2 relaxation values for tissues can be estimated by fitting the data to the following equation using a mono-exponential decay curve:

$$S(TE) = S_0 \cdot e^{\frac{\mathrm{TI}}{\mathrm{T_2}}}, \tag{2.4}$$

where $S(TE)$ is the signal intensity measured at each TE and S_0 is the initial signal intensity at time $t = 0$.

2.1.2 Image Acquisition and the Concept of K-Space

After discussing the physical mechanisms behind MRI, it is time to look at the actual process of acquiring the image as well as the concept of the k-space. The image contrast in MR imaging arises from tissues generating MR signals with different intensities due to their physical properties. Contrast weighting of the MR signal is obtained by the design of pulse sequences, which consist of repetitive trains of RF pulses. Two of the most common pulse sequences are gradient echo and spin echo sequences.

Gradient Echo Sequence

In a gradient echo (GE) sequence, the free induction decay (FID) signal is manipulated by a bi-polar gradient. The excitation pulse tilts the magnetization by α degrees. For $\alpha = 90$ the longitudinal magnetization is rotated in the transverse plane. The data is sampled during a gradient echo at time TE after the excitation pulse. This gradient echo is achieved by dephasing the spins with a negative gradient before they are rephased by an opposite gradient with opposite polarity to generate an echo. A schematic overview of the process can be seen in Figure 2.2.
The pulse sequence is repeated a number of times to acquire the entire image. Changing the TR, TE, and flip angles of a GE sequence influences the contrast weighting. T_2^*-weighted contrast can be achieved by using small flip angles and a long TE and moderate TR. By using large flip angles, a short TR and a short TE, a T_1-weighted signal can be acquired. Using small flip angles in combination with a long TR and a short TE generates proton density contrast [**PulseSequences**].

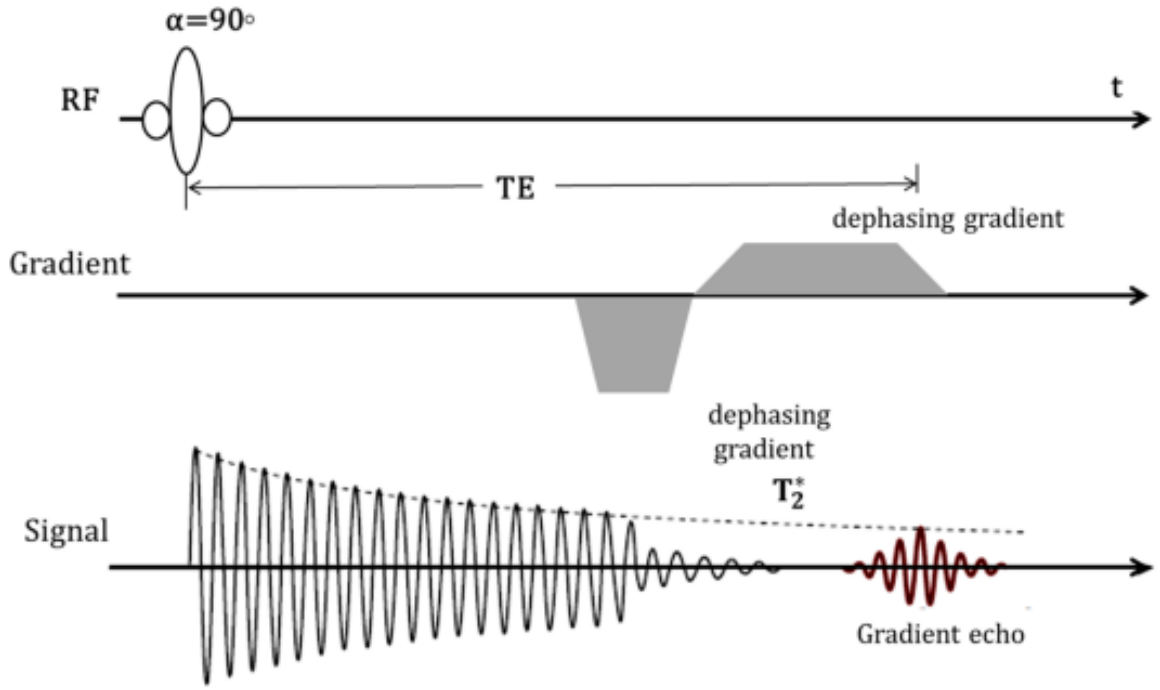


Figure 2.2: Schematic of an gradient echo sequence taken from [**PulseSequences**].

Spin Echo Sequence

With a spin echo (SE) sequence, pure T_2-weighted contrast can be generated. When a 90 pulse rotates the magnetization into the transverse plane, the resulting FID signal quickly decays due to the strong T_2^* dephasing. If after a time $\frac{TE}{2}$ a 180 pulse is applied, the spins will be flipped and start to rephase. After another time, $\frac{TE}{2}$, a measurable echo signal is created. The spin dephasing due to static magnetic field inhomogeneities is compensated by inverting the spins with the 180 refocusing pulse. This process is visualized in Figure 2.3.
Consequently, the decay of the signal at time TE will solely originate from the T_2 relaxation. A SE sequence can also be used to generate proton density or T_1-weighted signals by using a short TE and a long or short TR, respectively [**PulseSequences**].

Introduction to K-Space

The concept of the k-space is a generalization of the simple relation of a time-variant signal to a spectrum of two or more dimensions. An 2D image is related to a 2D k-space data set by a 2D FT, as seen in Figure 2.4 As the FT is an information-preserving operation, the k-space data contains exactly the same information as the image data. Thus, in order to get the full image information, the full k-space data needs to be measured. The task of forming an image by this approach is then converted to the task of finding a way to measure the necessary corresponding k-space data. Using the Larmor frequency again from Equation 2.1 the coordinates in 2D k-space will thus be given as $k_x = \gamma \cdot G_x \cdot t$ and $k_y = \gamma \cdot G_y \cdot t$.

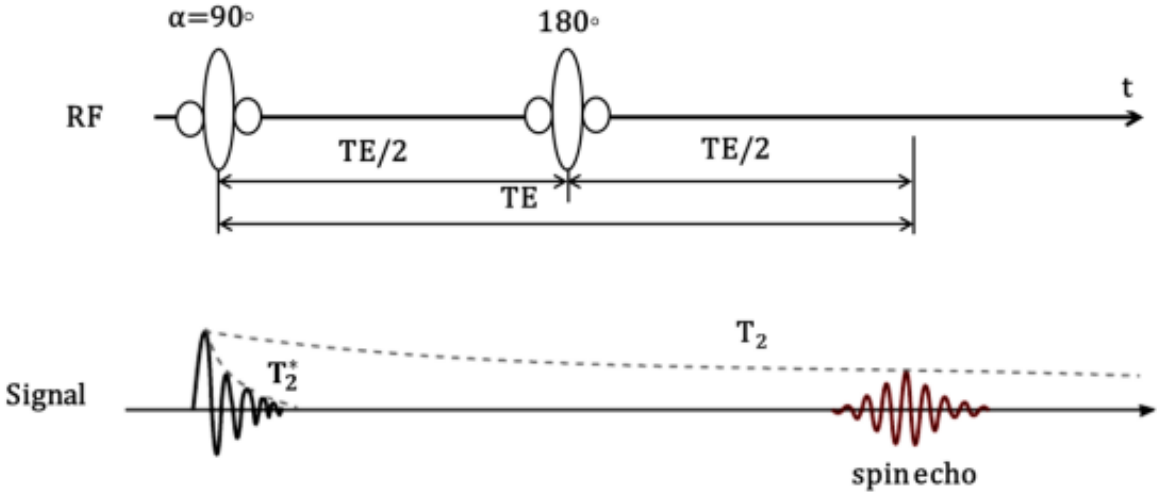


Figure 2.3: Schematic of an spin echo sequence taken from [**PulseSequences**].

In order to acquire an MR image, methods need to be developed to traverse the k-space. There are only two possibilities as seen in Figure 2.5. The first one is to apply a gradient, in which case the k-space trajectory will be a line defined by the orientation of the gradient (see Figure 2.5a). As long as the gradient is kept constant, the k-space trajectory will be a straight line. The second method for traversing the k-space is the application of a refocusing pulse, which will lead to a jump of the k-space trajectory around the origin (see Figure 2.5b). It is clear that a spin-echo formation alone will not allow coverage of all of the k-space, since the trajectory would only jump between the two mirror symmetric points. A combination of spin echoes and gradients can, however, produce very efficient k-space trajectories. All of the existing k-space-based imaging techniques are based on some combination of these two basic means of traversing the k-space.
Is should be noted that the signals are measured at discrete time intervals called the dwell-time of data acquisition. This discrete sampling leads to an ambiguous assignment of frequencies above a given threshold which is called the Nyquist frequency. The Nyquist frequency therefore determines the acquisition bandwidth inside which the signal should occur. The definition of the k-space coordinates implies that these are invariant with respect to the actual strength of the gradient used, as long as G and t are constant. Mathematically, data acquisition under a strong gradient and in a shorter acquisition time will yield the same k-space data and thus the same image as acquisition under a weaker gradient and a longer acquisition time. A shorter acquisition time and a closer spacing of sampling points are equivalent to a higher-acquisition bandwidth. Since the received noise grows with the square root of the bandwidth, a faster imaging technique will therefore by principle always carry the penalty of a lower signal-to-noise ratio (SNR) For a conventional spin-echo imaging sequence a typical acquisition time of 5 to 10 ms for each phase-encoding step is used. Acquisition of 192 to 256 phase-encoding steps therefore requires a total net acquisition time of 1 to 2 s. However, this is much shorter than the actual acquisition time of such a sequence,

which is determined by the repetition time defined by the T_1-contrast. It follows that the SNR of a fast imaging sequence leading to acquisition times of approximately 50 ms, will be lowered by at least a factor of 5 to 10, regardless of the actual sequence used. Fast imaging thus relies on radio-frequency coil designs improving the SNR.
The discrete data sampling has some additional consequences regarding the sampling density and coverage of data in k-space. Different parts of the k-space encode different features of the image: The center of the k-space hold the lower frequencies which represent the image contrast, whereas the outer parts encode the higher frequencies for sharp structures. Due to the symmetry of the 2D FT, this statement can also be reversed: The image center will be encoded by low-resolution k-space data. Therefore, sampling of sparsely distributed k-space data will reduce the effective field of view of the final image.
Mathematically, the final image will look exactly the same, irrespective of the way the data is sampled in the k-space. In practice, however, different approaches can have a major impact on image quality as the data has to sampled sequentially. The observed spins evolve not only as a function of the gradients defining the intended k-space trajectory, but are also influenced by other mechanisms unrelated to image encoding. It is noteworthy that the coordinates in k-space are expressed in units of a phase angle across space. Any mechanism affecting the phase of the signal along this trajectory will therefore alter the k-space trajectory. Some such commonly occurring mechanisms are flow and motion effects, magnetic field inhomogeneities, as well as susceptibility and chemical shift effects. The consequence of such phase effects in terms of imaging properties is dependent on the particular sequence and data sampling speed. In addition to phase effects, the signal decay with T_2 needs to be taken into account also, especially if the data acquisition time is similar or even longer than the T_2 of the observed tissues under construction [**SamplingStrategies**].

Rectilinear K-Space Sampling

Almost all MR imaging sequences are based on rectilinear k-space sampling, i. e., the sampling points are placed on a rectangular grid. This ia also referred to as Cartesian sampling and allows the usage of the fast Fourier transform (FFT) which enables image reconstruction in a very short time (under 1 s). For comparison, using the 2D FT to transform a non-rectilinear grid can take about 5 to 50 min per image [**SamplingStrategies**]. In a Cartesian acquisition, one or more lines of k-space parallel to a Cartesian axis are collected following application of an RF pulse and generation of an RF echo. Adjacent points in a single line are collected in rapid succession in a single echo. The time between echoes and adjacent lines is much slower. The direction of fast acquisition is known as the frequency direction and the other one or two directions are known as the phase-encode direction or directions. Following line-by-line filling of k-space in a Cartesian sequence, each layer of a filled grid is subsequently used to reconstruct a single image slice [**Bardo2021**].

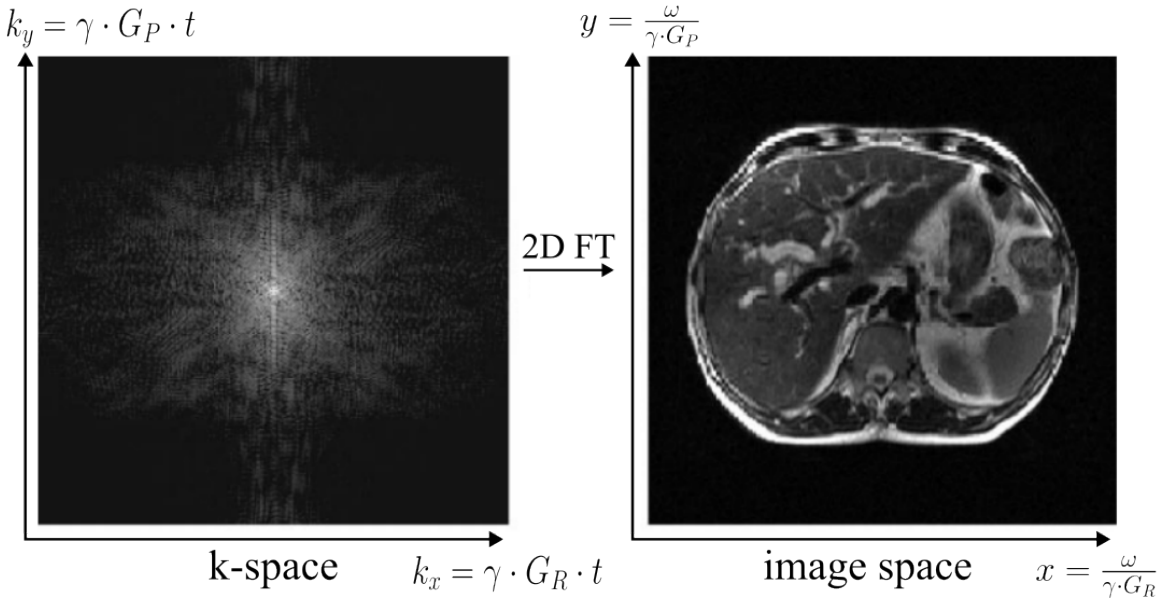
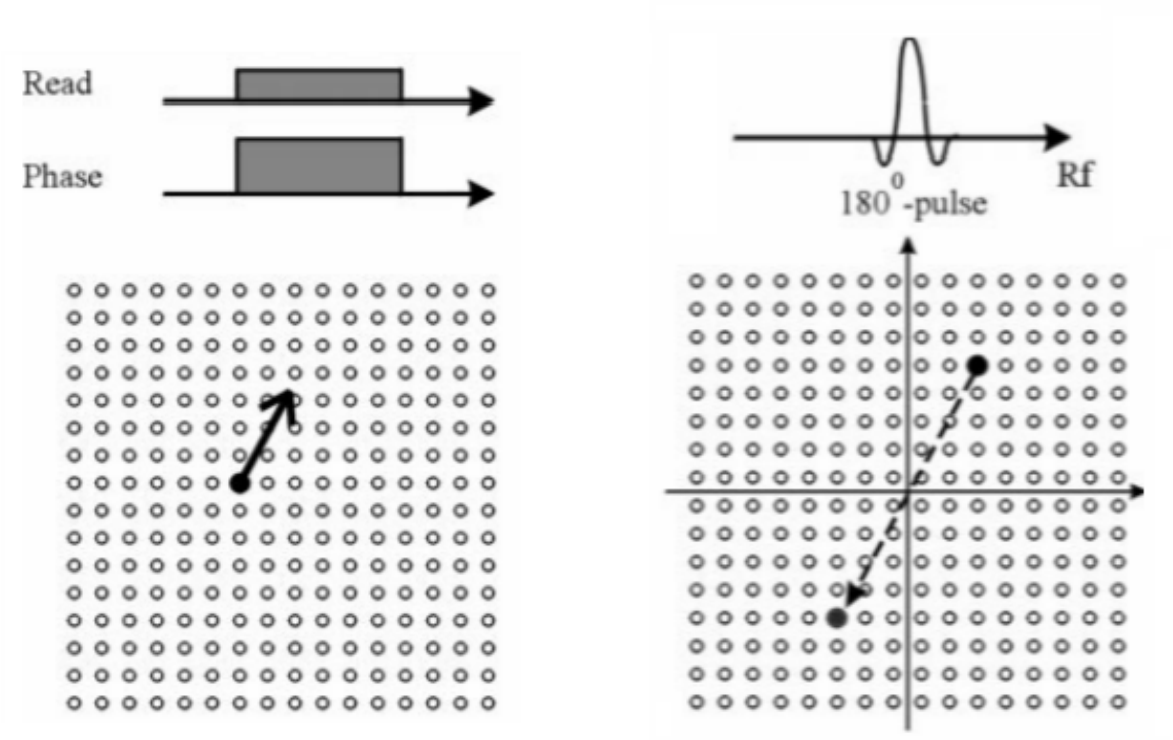


Figure 2.4: Correspondence of the image domain and the k-space via a 2D FT, adapted from [**SamplingStrategies**]. The coordinates of each domain are defined by the Larmor frequency according to Equation 2.1.

Non-Rectilinear K-Space Sampling

A common problem of rectilinear sampling techniques is the very heterogeneous nature of the k-space trajectories used: Data are acquired along straight k-space lines under a constant gradient. Going around the corner of the trajectory requires very fast switching for a brief period. This is very demanding on the gradient power amplifiers, which have to be able to alternate between the two modes. A more common load on the gradients is offered by the use of curved k-space trajectories. Especially spiral trajectories [**SpiralMRI**, **SpiralMRI2**] have won considerable interest due to their very efficient use of the gradient system. A practical problem relates to image reconstruction which requires algorithms not (yet) commonly available and which can take several seconds per image on a typical scanner [**SamplingStrategies**]. The ultimate success of spiral imaging will then be determined by its inherent imaging properties, which are significantly different from rectilinear scans. Constant off-resonance effects, such as chemical shift, field inhomogeneity, and susceptibility, will not lead to a displacement in the phase-encoding direction, but to blurring of the corresponding structures [**SamplingStrategies**]. A problem of spiral imaging is the fact that severe artifacts can arise when the k-space trajectory produced by the gradient system is not exactly identical to the trajectory used for image reconstruction. It is thus necessary to accurately calibrate the gradient system or to even measure the actual trajectory used. A favorable property of spirals is the motion correction inherent to the trajectory [**SpiralMRI3**]. Apart from spirals, other non-rectilinear k-space trajectories have been suggested. Back projection uses a star-like trajectory and allows the realization of



(a) Schematic of a constant gradient creating a straight k-space trajectory.

(b) Schematic of an refocusing pulse inverting the phase of the k-space.

Figure 2.5: Methods of moving in the k-space using a) constant gradients to move along straight lines or b) mirror the k-space at the origin using a refocusing pulse. Adapted from [**SamplingStrategies**].

extremely short echo times Its inhomogeneous k-space coverage with a high sampling density for points at the center of k-space and its somewhat awkward artifact behavior have limited its success for conventional imaging [**SamplingStrategies**].
Other trajectories, such as rosettes or even random k-space trajectories, have also been explored. A common feature of all non-rectilinear trajectories relates to their non-periodic nature, which produces severe artifacts when the field of view of the data acquisition is less than the size of the object. For rectilinear scans this leads to severe fold-over artifact, which can be tolerated as long as it does not affect relevant structures or even is totally avoided when it occurs in the readout direction, where oversampling can be applied without penalty in the data acquisition time. For non-rectilinear scans oversampling is not an option, since the field of view is determined by the distance between adjacent parts of the trajectory, rather than by the sampling rate along the trajectory. For an insufficient field of view of the k-space data, misregistration artifacts can be very severe, depending on the trajectory used. Spirals produce a circular rim artifact around the image. It is therefore mandatory to ensure that the k-space trajectory is sufficiently dense to always cover the whole object of interest [**SamplingStrategies**].

2.1.3 Imaging Acceleration

To alleviate the slow image acquisition times of the MR imaging, different MRI acceleration techniques are used. Most of these included scanning less k-space lines during signal acquisition, omitting certain frequencies The amount of k-space lines that are omitted are usually given by a reduction factor R. While there are different subsampling techniques, most involve fully sampling a center region containing the lower frequencies and dropping higher frequencies, which are deemed less important. However, all of these methods create artifacts like image blur during image reconstruction, as seen in Figure 2.6, leading to different technologies being developed to minimize these effects such as parallel imaging and compressed sensing.

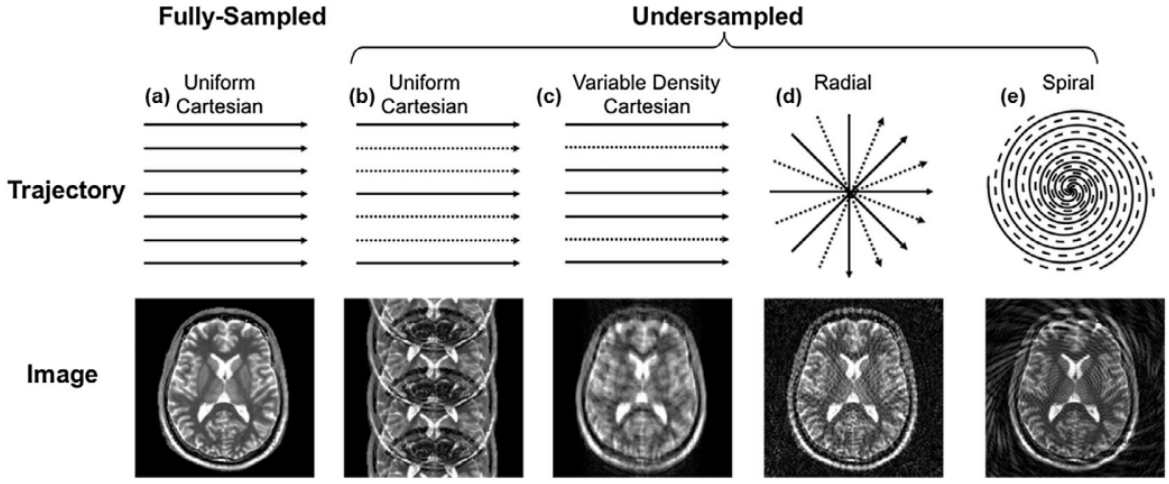


Figure 2.6: Examples of different k-space trajectories and their associated aliasing artifacts (missing data represented by dashed lines) taken from [**AdvancesPI**].

Parallel Imaging

Parallel imaging (PI) is a robust method for accelerating the acquisition of MRI data by acquiring a reduced amount of k-space data with an array of receiver coils. These undersampled data can be acquired more quickly, but the undersampling leads to aliased images. One of several PI algorithms can then be used to reconstruct artifact-free images from either the aliased images (SENSE-type reconstruction) or from the undersampled data (GRAPPA-type reconstruction). PI requires special hardware known as phased array coils that contain multiple independent receiver channels. Each coil element is most sensitive to the magnetization closest to it and less sensitive to magnetization further away. Individually, each coil has high local signal-to-noise ratio (SNR) but inhomogeneous coverage. To expand the FOV while maintaining high SNR, several coils are organized in an array. The images from each channel are usually combined into a single image with relatively homogeneous intensity by taking e.g. a root sum-of-squares combination. Theoretically, the maximum acceleration factor is limited by the

number of coils ,thus, for PI to be successful, each coil must have a unique sensitivity variation along the direction that is accelerated. For example, an array with four coils arranged in a line may be able to attain a maximum R = 4 along one direction, but no acceleration would be possible in the perpendicular direction [**AdvancesPI**].
As mentioned before, PI techniques fall into one of two classes, depending on whether aliased pixels are separated in the image domain (SENSE) or missing phase encoding lines are reconstructed in k-space (GRAPPA). Sensitivity encoding (SENSE) [**SENSE1**] is a PI technique which unfolds superimposed pixels in the image domain. For Cartesian k-space sampling with a uniform acceleration factor of R = 3 and a receiver array with four coils the undersampling reduces the FOV threefold, such that three pixels from the fully-sampled image fold onto the same pixel in the aliased image. SENSE uses prior knowledge of the coil sensitivity profiles to separate folded pixels and recover the full FOV image. If the acceleration factor exceeds the number of coils, then the SENSE algortihm will not be able to recover an un-aliased image. In practice, the largest achievable acceleration factor is usually smaller than the theoretical limit because coil sensitivities overlap and are not orthogonal [**AdvancesPI**].
One common feature of PI is the amplification of noise in the reconstructed image, though with varying degrees:

$$SNR_{SENSE}(x,y) = \frac{SNR_{full}(x,y)}{\sqrt{R} \cdot g(x,y)}, \tag{2.5}$$

where the geometry factor $g(x,y)$ describes the spatial pattern of noise enhancement, $SNR_{SENSE}(x,y)$ the SNR in the reconstructed SENSE image and $SNR_{full}(x,y)$ the SNR of the fully sampled image. In addition to the g-factor losses, which depend on variables like number of coils, array configuration, etc., the SNR decreases with the square root of the acceleration factor R, which is known as Fourier averaging.
While SENSE unfolds aliased signals in the image domain, generalized partially parallel acquisitions (GRAPPA) [**GRAPPA**] synthesizes missing data points directly in k-space. There, the use of inhomogeneous receiver coils effectively spreads information from one k-space point to nearby k-space points. GRAPPA exploits these k-space redundancies across coils to reconstruct missing k-space data using neighboring acquired points. A single missing k-space data point (target point) is synthesized as a linear combination of acquired neighboring k-space points (source points) with the spatial arrangement of source and target points being called GRAPPA kernel. Each acquired source point is multiplied by a coefficient (GRAPPA weight) and the results are added to estimate the target point. A single target point for one coil is reconstructed using source points from all other coils. For Cartesian sampling, the weights are shift invariant to a first approximation, so the same GRAPPA weights can be applied throughout k-space. The reconstruction can be described as convolving or sliding the GRAPPA kernel throughout k-space. While SENSE uses additional information in the form of coil sensitivity profiles to unfold aliased pixels, GRAPPA requires extra data to estimate the weights. GRAPPA is considered to be autocalibrating because several additional phase encoding lines, called the auto-calibration signal (ACS), are collected near the k-space origin for calculating the weights. The SNR of images recon-

structed using GRAPPA is also reduced according to Equation 2.5, however, because GRAPPA does not require an explicit estimate of the coil sensitivities, it tends to be more robust than SENSE to inconsistencies between the calibration and undersampled data [**AdvancesPI**].
SENSE and GRAPPA can be combined in what is called iterative self-consistent parallel imaging (SPIRiT) [**SPIRiT**]. Like GRAPPA k-space kernels are used to recover missing information by exploiting correlations between neighboring k-space points, though the reconstruction is framed as an inverse problem similar to SENSE. Regardless of the original sampling trajectory, SPIRiT outputs a Cartesian k-space. The reconstruction is typically initialized with the undersampled zero-filled k-space and is solved iteratively. The algorithm minimizes and balances the errors of two terms: calibration consistency and data consistency. The first is calculated using a so-called SPIRiT kernel in the k-space similar to GRAPPA, while the second term enforces consistency with the undersampled data as the difference between the reconstructed data and acquired data should be zero at the originally sampled positions. Thus reconstruction should only recover missing k-space points without changing the known data points [**AdvancesPI**].ESPIRiT [**ESPIRiT**] is an extension of SPIRiT that uses k-space kernel operations to derive a set of eigenvector maps that behave like coil sensitivities, which can be incorporated in a generalized SENSE reconstruction. This requires calibration data from the fully sampled region at the center of k-space. Unlike SENSE and GRAPPA, SPIRiT reconstructs images iteratively and can require long computation times which can be addressed by using e.g. parallelized GPUs [**AdvancesPI**].

Compressed Sensing

The idea behind compressed sensing (CS) is that sparse or compressible signals can be acquired in an efficient way by applying compression already in the data acquisition process. According to CS theory, sparse or compressible signals can be recovered from fewer samples than required by the Shannon-Nyquist sampling theorem [**CS-MRI**]. This is achieved by applying an appropriate sampling scheme and reconstruction that employs signal sparsity to recover the signal. MRI fulfills two important requirements for the application of CS: medical imaging is naturally compressible by sparse coding and MRI scanners acquire samples of the encoded image in spatial frequency, rather than direct pixel values. CS emerged as an abstract mathematical idea that if one measures a relatively small number of random linear combinations of the signal values the signal can be reconstructed with good accuracy from these few measurements by a non-linear procedure due to the underlying signal being compressible. In MRI, the sampled linear combinations are the individual Fourier coefficients (k-space samples) and CS is able to make accurate reconstructions from a small subset of k-space, rather than an entire k-space grid [**CS-MRI**]. It should be noted that unlike PI approaches, the k-space is incoherently sampled; thus, noise-like artefacts appear in the image when a direct inverse Fourier transform is performed. To remove the artefacts caused by the undersampling, iterative reconstruction is used. There are a number of sparsifying transforms that can be used for CS such as wavelets, which are very common, and

total variation, which enforces the sparsity of the image gradients. The advantages of total variation include its simplicity, rotation invariance, and capability of preserving edges and providing good image quality [**PulseSequences**]. There are also several methods combining CS and PI trying to achieve higher acceleration factors [**PI+CS**, **PI+CS2**].

Deep Learning Based Subsampling

There are also approaches for learning new subsampling strategies in a data-driven manner (pruning unimportant k-space frequencies) [**MRISubsamplingPruning**] as well as deep learning based radial [**DeepMRIReconstructionRadialSubsampling**] and non-Cartesian [**DeepMRIReconstructionSubsampling**] subsampling for MRI acceleration.

2.1.4 Motion-Compensated Image Reconstruction

Patient motion during acquisition is one of the major impediments of high-quality MRI scans. This is especially true for thoracic and abdominal imaging, as organs move during breathing. Steps can be taken during or after the reconstruction to mitigate motion artifacts.

Intraview and Interview Motion

Due to the long acquisition times, motion is one of the major extrinsic factors influencing MR image quality. Patient and physiological motion induces aliasing along the phase-encoding direction and/or blurring of the image content [**Kuestner2022**]. This motion can induce several consequences on MR signal formation. Intraview and interview motion have to be distinguished between: motion is intraview when occurring during individual MR experiments (between RF excitation and echo formation), whereas motion is interview when occurring between individual MR experiments. Whenever the period of motion is slow compared to the period of MR acquisition TR, the assumption can be made that motion is interview. This is often a reasonable assumption when considering pseudo-periodic motion induced by respiration, and also possibly by cardiac contraction, which are the two common sources of motion in cardiac and abdominal imaging (typically, the adult respiratory period is about 4 to 5 s, and $TR \approx 10ms$ for fast imaging). Interview motion results in spatial encoding inconsistencies, and thus in image deterioration which can take complex forms (blurring/ghosting artifacts) as acquisition is performed in a Fourier space. Several strategies can be employed in order to handle patient motion better. Patient cooperation is the most commonly used method. However, breath-holds cannot last much longer than 20 s and physiological drifts cannot be completely avoided. This leads to a limitation on the time-period of signal recording and thus SNR. Moreover, the position of organs in successive breath-holds may not be reproducible. Synchronization techniques are well-established and systematically used in clinical protocols, but they require a high-level of motion reproducibility. This is often a limiting factor considering heart rate variability (whether in

free breathing or during a breathhold), and respiratory variability in terms of amplitude and frequency [**GRICS**].

Reconstruction Pipelines

Motion-resolved data acquisition is usually accelerated by PI or CS techniques yielding subsampled k-space data. In order to reconstruct aliasing-free images, these methods rely on reconstruction schemes that incorporate sparsity or low-rank constraints to solve the ill-posed problem [**CS-MRI**, **ParallelMRI**, **LowRank+SparseMRI**]. Fixed sparsity assumptions in CS are often too restrictive and incapable of fully modeling spatio-temporal dynamics [**Kuestner2022**]. After reconstruction, non-rigid motion fields can be estimated in image space from reconstructed images by solving a registration problem. A particular interest and challenge lies in the derivation of reliable motion fields which capture the spatio-temporal non-rigid deformations, such as respiratory or cardiac movement. Instead of performing these two steps sequentially, motion-compensated image reconstruction schemes like *GRICS* [**GRICS**] integrate both motion field estimation and motion correction into the reconstruction process. These methods require reliable motion-resolved images from which the motion fields can be estimated. Motion field estimation can be controlled or supported by external motion surrogate signals, initial motion field estimates, from motion-aliased images or low-frequency image contents. Moreover, spatio-temporal redundancies can be exploited to achieve an aliasing-free image. While these methods have been proven to be more robust against registration errors, they can require a significantly increased computational demand and/or limit imaging acceleration [**Kuestner2022**].
There are two major parts of a motion-compensated reconstruction pipeline: a non-rigid registration model to reliably estimate the motion fields between frames and an unrolled reconstruction model that reconstructs the motion-corrected frames. Both of these can use either conventional iterative algorithms like in GRICS [**GRICS**] or neural networks that learn the given task [**Kuestner2022**].

2.2 Image Transformations and Registration

In this chapter, a brief overview over the different kinds of image transformations is given, followed by an introduction into the challenges of image registration.
The goal of image registration is to compute a transformation that aligns two images - moving and fixed. This transformation is then applied to the moving image to create an warped image that more closely resembles the fixed image. But first, the different kinds of image transformations must be discussed.

2.2.1 Image Transformations

In medical image registration, there are three basic transformation types that are typically applied: rigid, affine and non-linear(deformable) [**Strittmatter2023**].

Rigid Transformations

Rigid transformations are linear and global transformations that affect the whole image. As these are global operations, one can express them as matrix and vector operations. A rigid transformation includes translation and rotation and can be represented by:

$$\mathbf{T}_{rigid}(x) = \mathbf{R} \cdot \overrightarrow{p} + \overrightarrow{t}, \tag{2.6}$$

with $\mathbf{R}$ being the rotation matrix, $\overrightarrow{p}$ a point in the image and $\overrightarrow{t}$ the translation vector. For 3D images, the rotation matrix and the translation vector require three parameters each. Thus, six parameters have to be calculated for a rigid transformation for 3D images [**Strittmatter2023**].

Affine Transformations

Affine transformations are also linear and global transformations, however they include translation, rotation, scaling and shearing. Matrix multiplication can be used to merge all of these individual transformations into a single matrix:

$$\overrightarrow{p}' = \mathbf{R} \cdot \mathbf{S} \cdot \mathbf{t} \cdot \overrightarrow{p} \tag{2.7}$$

$$= \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2.8}$$

$$= \begin{bmatrix} s \cdot \cos(\theta) & -s \cdot \sin(\theta) & t_x \\ s \cdot \sin(\theta) & s \cdot \cos(\theta) & t_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}, \tag{2.9}$$

with θ being the angle of the rotation for the rotation matric $\mathbf{R}$, s the scaling factor for the scaling matrix $\mathbf{S}$ as well as t_x and t_y being the translations in x- and y-direction. This can be further generalized as a single matrix multiplication:

$$\overrightarrow{p}' = \mathbf{T}_{affine} \cdot \overrightarrow{p} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}, \tag{2.10}$$

with $\mathbf{T}_{affine}$ being the general affine transformation matrix [**Strittmatter2023**].

Non-Linear Transformations

Non-linear or deformable transformations are used to model local deformations in images that rigid and affine transformations cannot capture as these transformations are capable of locally warping the image. Non-linear transformations include radial basis functions, physical continuum models, and large deformation models like diffeomorphisms. These transformations are complex and do not preserve straightness or parallelism:

$$\mathbf{T}_{deformable}(x) = x + \phi(x), \tag{2.11}$$

with ϕ being the deformation or displacement field. Deformable image registration is an ill-posed problem, often requiring regularization to ensure a smooth and plausible displacement field [**Strittmatter2023**].

2.2.2 Image Registration

Image registration is a challenging, yet important task for image processing. In the field of medical image analysis, image registration remains one of the main research topics and challenges. This task consists of transforming a moving image to match a fixed image [**NiftiReg**]. In the medical field this can be used for clinical applications such as disease diagnosis and monitoring, image-guided treatment delivery, and post-operative assessment. Medical image registration is typically used to pre-process data for tasks like object detection (for e.g. tumor growth monitoring) and segmentation (for e.g. organ atlas creation). Thus the performance of these methods is dependent on the quality of image registration [**Chen2020**].
Medical image registration was historically done manually by clinicians, however, registration tasks are often challenging and the quality of manual alignments is dependent on the expertise of the user. These manual registrations are thus not only time consuming, but also hardly reproducible leading to high interobserver-variability. While the need for automatic registration is very much apparent, the computational cost of traditional registration algorithms prohibited their usage for a long time. With the rise of deep learning, neural network gained popularity and now provide an alternative to conventional algorithms and manual registration that is fast and computationally efficient during execution [**Haskins2020**]. However, it should be noted that these networks still need a lot of data and computational power to be trained.
In pair-wise image registration two images, called moving (M) and fixed (F), are to be aligned with a spatial transformation T. As discussed before there are three types of transformations: rigid, affine, and non-linear (deformable). While the latter is the most difficult to compute these are also the transformations most likely encountered in clinical practice [**Zou2022**]. Additionally, deformable image registration can also be utilized for various computer-assisted interventions like biopsy [**Tam2016**] and (MRI-guided) radiotherapy [**Chen2017**, **Rigaud2019**]. This can be described as an optimization problem:

$$T' = \arg\max S(F, T(M)), \tag{2.12}$$

where T' is the best transformation maximizing the similarity S between the two images. This process can be done iteratively, continuously improving the estimates for the desired T, such that the defined similarity in the cost function is maximized [**Chen2020**].
Intuitively, deformable image registration is an ill-posed problem making it fundamentally different from other computer vision tasks such as object localization, segmentation or classification. Given two images, deformable image registration aims to find a spatial transformation that warps the moving image to match the fixed image as closely as possible. However, there is no ground-truth available for the desired deformation field and without enforcing any constraints on the properties of the spatial transformation, the resulting cost function is ill-conditioned and highly non-convex. In order to

address the latter and ensure tractability, all image registration algorithms regularize the estimated deformation field, based on some prior assumptions on the properties of the underlying unknown deformation [**Chen2020**].
Many methods have been proposed for medical image registration to deal with the complex challenges of this task. Popular conventional registration methods include optical flow [**Yang2008**], demons [**Vercauteren2009**] and many more. However, most of these still lack accuracy and computation speed, which makes newer deep learning approaches all the more interesting [**Fu2020**].

2.3 Deep Learning

Deep learning, and neural networks in general, have seen a significant rise in usage over the last couple of years due to many breakthroughs in areas of image recognition, segmentation and also registration. After discussing different network architectures, specific use-cases for image registration are discussed and a brief overview of network training and testing (including evaluation metrics) is given.

2.3.1 Deep Learning Architectures

Neural networks, despite the theoretical foundations being around for decades, have seen a drastic rise in popularity over the last few years as constraints on computational power and data size have been alleviated. Especially deep neural networks, which are often summarized under the term deep learning (DL). Recent years have witnessed an almost exponential growth in the development and use of DL algorithms, sustained thus far by rapid improvements in computational hardware (e.g. GPUs). Consequently, clinical applications requiring image classification, segmentation, registration, or object detection/localization, have witnessed significant improvements in algorithmic performance, in terms of accuracy and/or efficiency [**Chen2020**]. The following network architecture are widely used for different tasks including medical image registration.

Convolutional Neural Networks

Convolutional neural networks (CNNs) are a type of deep neural networks with regularized multilayer perceptron, which are mainly used for image processing. CNNs use convolution operations instead of general matrix multiplications in typical neural networks. These convolutional filters make CNNs very suitable for visual signal processing. Because of their excellent feature extraction ability, CNNs are some of the most successful models for image analysis. Different variants of CNN have been proposed and have achieved the-state-of-art performances in various image processing tasks. A typical CNN usually consists of multiple convolutional layers, max pooling layers, batch normalization layers, sometimes dropout layers, a sigmoid or softmax layer. In each convolutional layer, multiple channels of feature maps are extracted by sliding trainable convolutional kernels across the input feature maps. Hierarchical features with high-level abstraction are extracted using multiple convolutional layers. These feature

maps usually go through multiple fully connected layer before reaching the final decision layer. Max pooling layers are often used to reduce the image sizes and to promote spatial invariance of the network. Batch normalization is used to reduce internal covariate shift among the training samples. Weight regularization and dropout layers are used to alleviate data overfitting [**Fu2020**]. The loss function is often defined as the difference between the predicted and the target output. CNNs are usually trained by minimizing the loss via gradient back propagation using optimization methods like Adam [**Adam**].

U-Net

The U-Net [**U-Net**] architecture is an extension of the usual CNN structure often used for image segmentation, however, it can also be applied to image registration tasks. It adopts symmetrical contractive and expansive paths with skip connections between them as seen in Figure 2.7. The encoding blocks on the left extract important features from the image using convolution layers and max pooling, which are then stored in the latent space in the middle. From there it is reconstructed using upsampling and (de-)convolutions in the decoding blocks on the right. Additionally, skip connections are used to improve the spatial resolution of the segmentation. This architecture allows effective feature learning from a small number of training datasets [**Fu2020**].

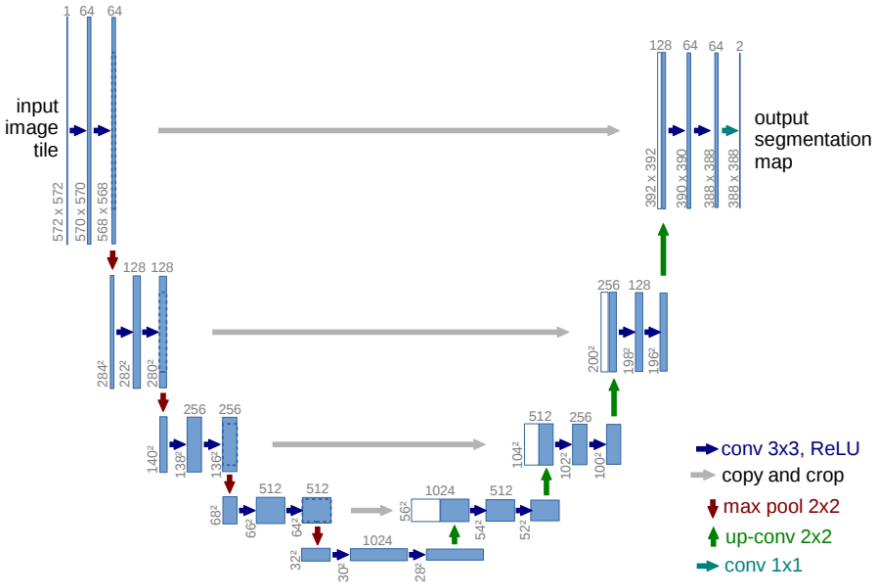


Figure 2.7: U-Net architecture taken from [**U-Net**] with an contractive path on the left and an expansive path on the right.

Autoencoders

An autoencoder (AE) is a type of CNNs that learns to reconstruct an image from its input without supervision. AEs usually consists of an encoder which extracts the

input features, which are stored a low-dimensional latent state space, similar to a U-Net, and a decoder which restore the original input from the latent space. To prevent an AE from learning an identity function, regularized autoencoders were invented, which can be used for e.g. denoising AEs. Variational AEs (VAEs) are generative models that learn latent representation using a variational approach, which constrains the variability of the outputs. VAEs can been used for anomaly detection and image generation [**Fu2020**].

Generative Adversarial Networks

Generative adversarial networks (GANs) consist of two competing networks, a generator and a discriminator. The generator is trained to generate artificial data that approximate a target data distribution from a low-dimensional latent space similar to an AE. The discriminator is trained to distinguish the artificial data from actual data. The discriminator encourages the generator to predict realistic data by penalizing unrealistic predictions via learning. Therefore, the discriminative loss could be considered as a dynamic network-based loss term. The generator and discriminator both are getting better during training to reach Nash equilibrium. In medical imaging, GANs have been used to perform image synthesis for inter- or intra-modality, such as MRI to synthetic CT and vise versa. In medical image registration, GANs are usually used to either provide additional regularization or translate multi-modal registration to uni-modal registration [**Fu2020**].

2.3.2 Deep Learning for Image Registration

Recently, there has been a surge in the use of deep learning based approaches for medical image registration. Their success is largely due to their ability to perform fast inference, and the flexibility to leverage auxiliary information such as anatomical masks as part of the training process. The most effective methods, such as *VoxelMorph* [**Voxelmorph**], typically employ a U-Net style architecture to estimate dense spatial deformation fields. These methods require only one forward pass during inference, making them orders of magnitude faster than traditional iterative methods. Following the success of *VoxelMorph*, numerous deep neural networks have been proposed for various registration tasks [**Fourier-Net+**]. Other approaches also utilize CNNs, AEs and GANs. Typical strategies are discussed in more detail in the following sections.

Supervised Registration

Supervised registration describes training a network with a ground truth displacement field that is either real (created by hand) or synthetic (generated via traditional iterative registration algorithms). Thus the loss can easily be calculated as the difference in the displacement fields of the network prediction and the ground truth (see Figure 2.8 for a visual overview). These methods have achieved notable results with real displacement fields as supervision. However, this approach is very limited by the size

and the diversity of the dataset. As the displacement fields are often calculated by conventional algorithms their effectiveness might be limited for difficult problems with which the traditional algorithms struggle. Fully supervised methods are widely studied and have notable results, but the generation of real or synthetic displacement fields is hard, and these displacements fields might be different from the real ground truth, which can impact the accuracy and efficiency of these kinds of methods [**Zou2022**]. Notable approaches include *BIRNet* [**BIRNet**] and *LAPNet* [**LAPNet**], however the latter works in the k-space domain.

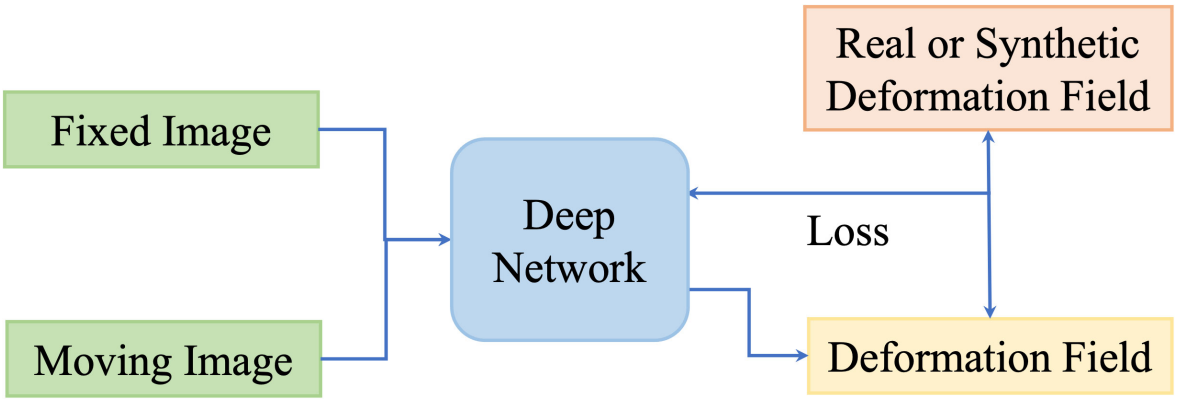


Figure 2.8: Example graph illustrating the training process of a supervised network, taken from [**Zou2022**].

Unsupervised Registration

As the preparation of the ground truth displacement field for supervised methods is inconvenient, limitations in generalizing results in different domains and various registration tasks are inevitable. Thus, unsupervised registration has a more convenient training process with paired images as inputs, but without a ground truth. Generally, unsupervised learning consists of similarity-based (see Figure 2.9) and GAN-based methods (see Figure 2.10), where the loss function computes the similarity between the aligned images and the smoothness of the displacement field, rather than the difference to a ground truth [**Zou2022**]. Well known example are *IC-Net* [**IC-Net**], *VoxelMorph* [**Voxelmorph**], *TransMorph* [**TransMorph**] and *SYMNet* [**SYM-Net**].

2.3.3 Network Training and Testing

In order to train any neural network, a large amount of data is needed. This data can take many forms such as videos, images, audio, text and many more. However, the total data set is usually devided into three subsets: the training, validation and testing subset.

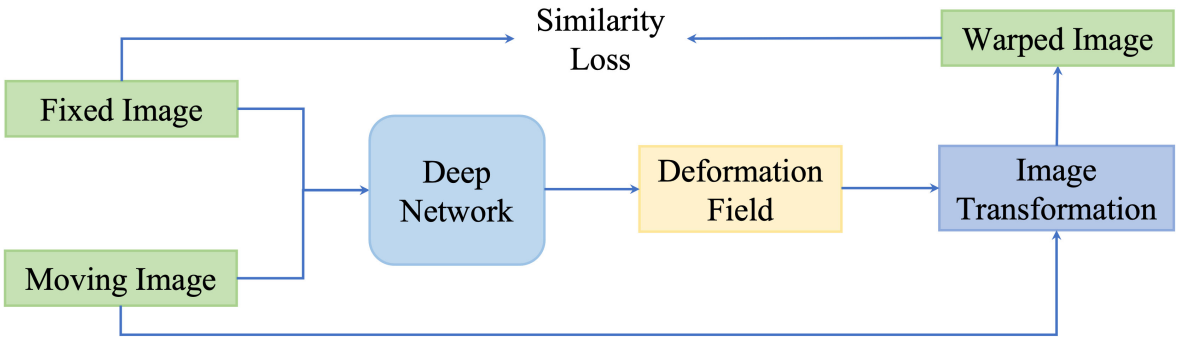


Figure 2.9: Example graph illustrating the training process of an unsupervised network with only a image similarity loss, taken from [**Zou2022**].

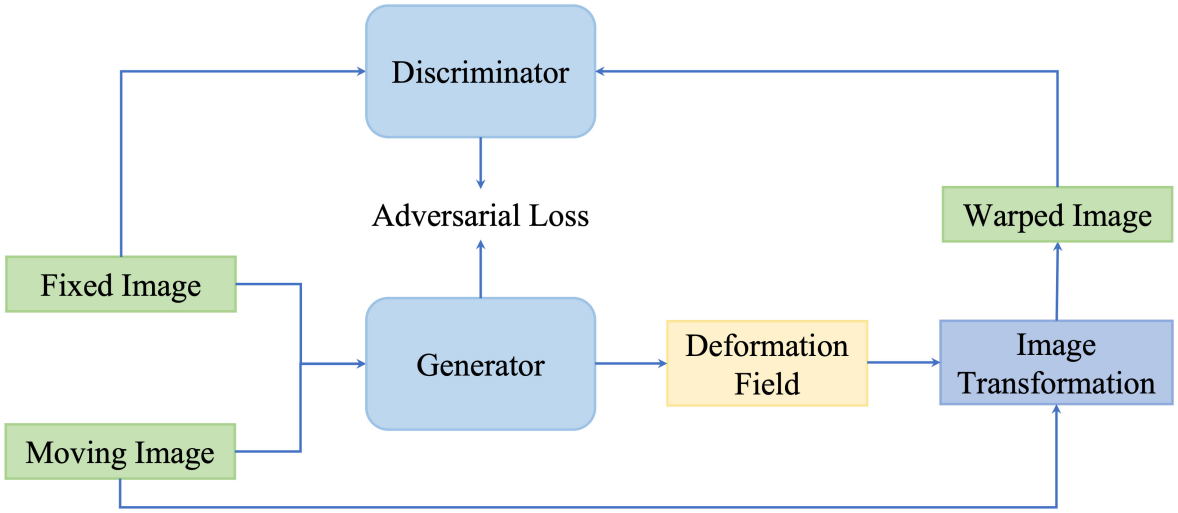


Figure 2.10: Example graph illustrating the training process of a GAN, taken from [**Zou2022**].

Training and Back-Propagation

The first subset is used for, as the name suggests, the training of the network. This if often the largest of the subsets and should include at lot of variance for the network to learn well. Neural networks in general learn by adapting their weights to perform on increasingly better at a specific task and on a specific dataset/domain. This is done in a process called back-propagation, where the loss calculated between the current output of the model, which is acquired by a simple forward pass through the network, and the desired output is propagated backwards through the model and the weights are optimized using e.g. gradiant decent or similar methods.
The second subsets is also used during the training, but is not learned, i.e. no back-propagation is performed. This is done to validate the networks training process and to prevent potential overfitting. Overfitting is a common problem, especially for larger networks, where the network learns every example from the training set, but no general properties as intended. To prevent this, the unseen validation data can be repeatedly

used, as it is not learned by the network, to see whether the network improves on unseen data. Once it has been trained for a long time and the validation performance stagnates, or in the worst case even drops, the training process can be stopped. This is known as early stopping and is a very common strategy to prevent overfitting while optimizing training time.

Testing and Evaluation Metrics

The last subset is the testing set, which is used for the final evaluation of the models performance. These usually include tests for the networks performance, as well as memory consumption and inference/execution time. It is important that the test set has not been used for training and is completely unseen by the network. This may sound trivial, but for datasets with a lot of patients providing data all of the data of the patient should be withheld for the testing, which can easily be overlooked.
Different metrics can be used for evaluation of a networks registration performance including similarity metrics computed on the images as well as properties of the generated displacement fields.
The mean squared error (MSE) is calculated between a fixed (ground truth) image and the warped image giving a pixel-wise comparison:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N} |F(x,y) - W(x,y)|^2, \tag{2.13}$$

where F is the fixed image and W is the warped image with N being the number of pixels in the images. The lower the MSE the higher the similarity with 0 being a perfect match (the images are exactly the same).
The MSE can also be used to calculate another useful metric - the preak SNR (PSNR), which describes the ratio between the maximum possible power of a signal and the power of corrupting noise. It is defined as:

$$\text{PSNR} = 20 \cdot \log_{10}\left(\frac{2^B - 1}{\sqrt{\text{MSE}}}\right), \tag{2.14}$$

with B being the bit depth of the image and the metric being defined in decibel (dB) due to the logarithmic scale.
Another common image metric is the structural similarity index measure (SSIM), which operates between 1 (complete similarity) and 0 (no similarity) and tries to estimate the general similarity instead of a pixel-wise comparison making it more robust against contrast changes compared to e.g. MSE.

$$\text{SSIM} = \frac{(2\mu_F\mu_W + c_1)\cdot(2\sigma_{FW} + c_2)}{(\mu_F^2 + \mu_W^2 + c_1)\cdot(\sigma_F^2 + \sigma_W^2 + c_2)}, \tag{2.15}$$

where μ_F, μ_W are the mean values of the images F and W; σ_F^2, σ_W^2 the variances of F and W as well as σ_{FW} the covariance for F and W, with c_1, c_2 being constants derived

from the dynamic range of the images.
All of these metrics can also be used as loss functions for network training.
For comparison of segmentations the Dice score is a commonly used metric to estimate the similarity of two segmentations. A score of 1 indicates a complete overlap/match and a score of 0 indicates no overlapping of the segmentations. The Dice score is calculated as follows:

$$\text{Dice} = \frac{2|M_F \cap M_W|}{|M_F| + |M_W|}, \tag{2.16}$$

with M_F and M_W being segmentations corresponding to F and W with M_F being a manual segmentation which is warped to obtain M_W [**NiftiReg**].
Aside from image similarity measures and the evaluation of image segmentations, the displacement field itself can also be evaluated. This is usually based on the assumption that the displacement should be smooth as, for example, folding the image could result in physically unrealistic anatomic structures, indicating errors. Jacobian matrices are the derivatives of the displacement field ϕ that form a second order tensor field:

$$J_\phi(p) = \nabla\phi(p) = \begin{pmatrix} \frac{\partial\phi_1(p)}{\partial x_1} & \frac{\partial\phi_1(p)}{\partial x_2} & \frac{\partial\phi_1(p)}{\partial x_3} \\ \frac{\partial\phi_2(p)}{\partial x_1} & \frac{\partial\phi_2(p)}{\partial x_2} & \frac{\partial\phi_2(p)}{\partial x_3} \\ \frac{\partial\phi_3(p)}{\partial x_1} & \frac{\partial\phi_3(p)}{\partial x_2} & \frac{\partial\phi_3(p)}{\partial x_3} \end{pmatrix}, \tag{2.17}$$

for a point p and encode the local stretching, shearing and rotating of the displacement field. The determinants of such matrices, also called the Jacobian determinants of the deformation $|J_\phi|$, must be positive everywhere to avoid folding as a region of negative determinants would indicate that the one-to-one mapping has been lost [**DARTEL**]. Thus the percentage of non-positive Jacobian determinants of the deformation $\% \, |J_\phi| \leq 0$ can be used to evaluate the quality of the generated displacement field [**Chen2023**].

Chapter 3

Materials and Methods

In this chapter, the networks, data and experiments are discussed.

3.1 Network Architectures

As a starting point *Fourier-Net* [**Fourier-Net**] and its successor *Fourier-Net+* [**Fourier-Net+**] were used. These networks, which are explained in the following pages, enable fast and accurate registration while needing less resources compared to similar approaches. These attributes are very beneficial for a potential application like motion correction because the current networks, e.g. *LAPNet* [**LAPNet**], are usually supervised and require large computational resources.

3.1.1 Fourier-Net

Fourier-Net is a new unsupervised approach that aims to learn a low-dimensional representation of the displacement field in a band-limited Fourier domain instead of the full field in the spatial domain. This band-limited representation is then decoded by a model-driven decoder to the dense, full-resolution displacement field in the spatial domain. This allows for fewer parameters and computational operations, resulting in faster inference speeds [**Fourier-Net**]. The architecture is based on the U-Net [**U-Net**], like most deep registration approaches, but replaces the expanding path with a parameter-free model-driven decoder as mentioned before. The encoder of *Fourier-Net* consists of a CNN, which takes two images (fixed and moving) as inputs. The output is a displacement field that is then converted from the spatial domain into the Fourier domain via an discrete Fourier transformation (DFT). From there, this band-limiting representation is padded with zeros to the full resolution of the original displacement field. The field is then recovered by using the inverse DFT (iDFT) to convert it back into the spatial domain. This displacement field is then used to warp the moving image into the fixed image. Additionally, squaring and scaling layers [**Dalca2018**] can be added before warping the image in order to encourage a diffeomorphism in final deformation.

Encoder

The encoder of *Fourier-Net* consists of a CNN that generates the displacement field between the two inputs followed by a DFT layer that produces a band-limited repre-

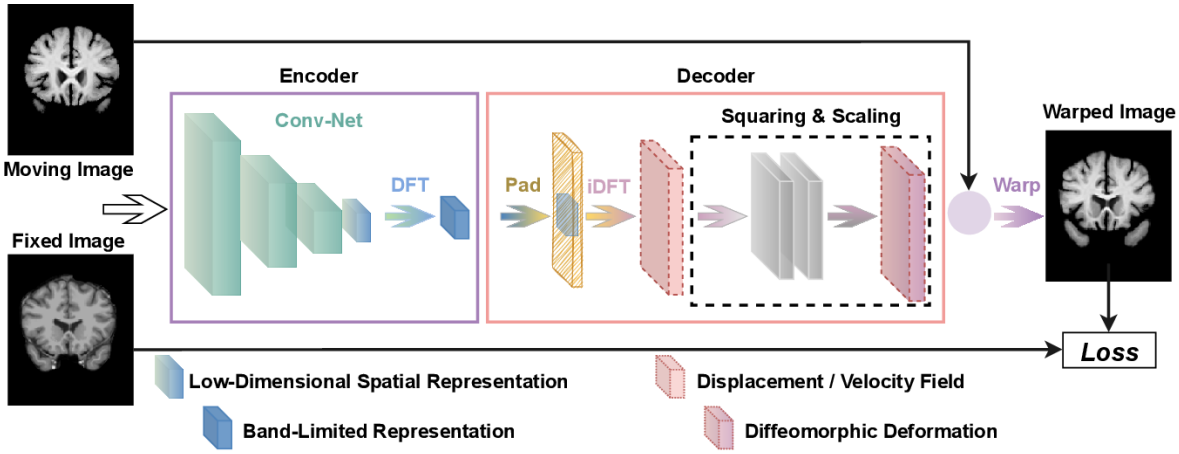


Figure 3.1: Architecture of *Fourier-Net* taken from [**Fourier-Net**].

sentation of the full displacement field. The fully convolutional neural network (FCN) from *SYMNet* [**SYM-Net**] was modified to function as the CNN of the encoder. Its (original) architecture (see Figure 3.2) is again based on the *U-Net* with a contracting and expanding path. The FCN concatenates the inputs images X and Y as a single 2-channel input and estimates two dense, non-linear displacement fields ϕ_{XY} and ϕ_{YX}, however we only need the displacement field for the moving image, denoted as $\mathbb{S}_\phi$, since we are not interested in transforming the fixed image. This is actually a low dimensional representation because *Fourier-Net* does not utilize the last two levels of the FCNs expanding path that would be needed to reconstruct the actual full-resolution displacement field ϕ.

For each level in the contracting path of the FCN, two successive convolution layers are applied, which contain one $3 \times 3 \times 3$ convolution layer with a stride of 1, followed by a $3 \times 3 \times 3$ convolution layer with a stride of 2 to further compute the high-level features between the input images as well as downsample the features by half until the lowest level of the network is reached. For each level in the expanding path of the FCN, the feature maps from the contracting path are concatenated through skip connections and apply $3 \times 3 \times 3$ convolution with a stride of 1 and $2 \times 2 \times 2$ deconvolution layer for upsampling the feature maps to twice of its size. At the end of the expanding path, two $5 \times 5 \times 5$ convolution layers with a stride of 1 are appended to the last convolution layer and generate the displacement fields θ_{XY} and θ_{YX} [**SYM-Net**]. Each convolution layer in the FCN is followed by a rectified linear unit (ReLU) activation, except for the output convolution layer that does not have an activation function because *Fourier-Net* only uses the first two levels of the expanding path, thus leading the FCN to generate a low dimensional representation $\mathbb{S}_\phi$ of the full-resolution displacement field ϕ.

As discussed previously, the encoder aims to learn a displacement (or velocity) field in the band-limited Fourier domain. Intuitively, this may require convolutions to be able to handle complex-valued numbers, which can be done by using complex-valued CNNs [**Trabelsi2017**], which are suitable when both input and output are complex values, however these complex-valued operations sacrifice computational efficiency. Other

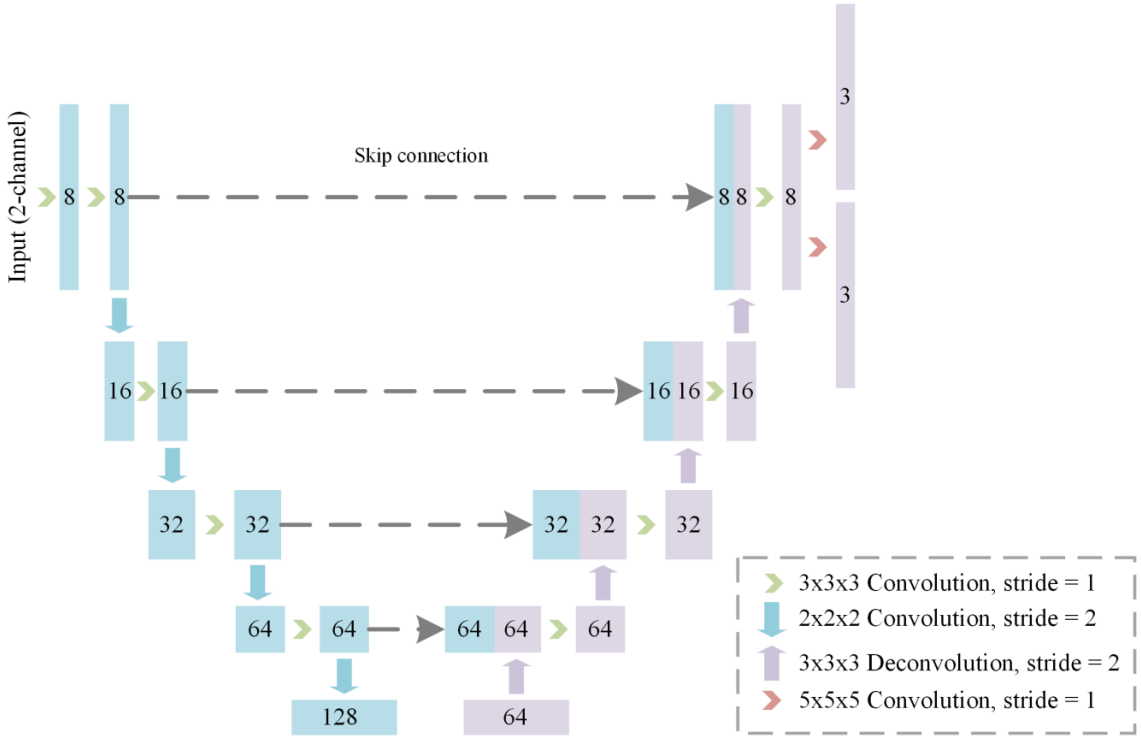


Figure 3.2: Architecture of the FCN from *SYMNet* [**SYM-Net**].

approaches like *DeepFlash* [**DeepFlash**] tackle this problem by converting the input images to the Fourier domain and using two individual real-valued CNNs to learn the real and imaginary parts separately. This, however, increases training and inference cost. To bridge the domain gap between real-valued spatial images and complex-valued band-limited displacement fields without increasing complexity, *Fourier-Net* uses a DFT layer after the FCN. This is a simple and effective way to produce complex-valued band-limited displacement fields without the network needing to be able to handle complex values itself. The DFT applied to the displacement field ϕ can be defined as follows:

$$[\mathcal{F}(\phi)]_{k,l} = \sum_{n=0}^{H-1} \sum_{m=0}^{W-1} \phi_{n,m} \cdot \exp\left(i \cdot \left(\frac{2\pi k}{H} \cdot n + \frac{2\pi l}{W} \cdot m\right)\right), \tag{3.1}$$

where ϕ has size $H \times W$, $n \in [0, H-1]$ and $m \in [0, W-1]$ are the discrete indices in the spatial domain, and $k \in [0, H-1]$ and $l \in [0, W-1]$ are the discrete indices in the frequency domain with i being the imaginary unit. However, ϕ in this equation is actually the low dimensional representation of the displacement field output by the modified FCN, which can be formulated as follows:

$$\mathbb{S}_\phi = \mathrm{FCN}(M, F; \Theta), \tag{3.2}$$

with M being the moving and F the fixed image, as well as Θ representing the parameters of the FCN. Thus, the whole encoder can be defined as:

$$\mathbb{B}_\phi = \mathcal{F}(\mathbb{S}_\phi) = \mathcal{F}(\mathrm{FCN}(M, F; \Theta)), \tag{3.3}$$

with the DFT layer $\mathcal{F}$, full-resolution spatial displacement field ϕ and the complex band-limited displacement field $\mathbb{B}_\phi$. The low dimensional representation $\mathbb{S}_\phi$ actually contains all the information of the band-limited Fourier coefficients in $\mathbb{B}_\phi$. As such, *Fourier-Net* does not need to learn the coefficients of $\mathbb{B}_\phi$, but instead only the real-valued coefficients in $\mathbb{S}_\phi$, which is the low dimensional spatial representation of the full-resolution spatial displacement field ϕ, which is then reconstructed by the decoder.

Decoder

The decoder contains no learnable parameters, instead the usual expansive path is replaced with a zero-padding layer, an iDFT layer, and an optional squaring and scaling layer.
The output from the encoder is a band-limited representation $\mathbb{B}_\phi$ in the frequency domain of the low dimensional displacement field $\mathbb{S}_\phi$ in the spatial domain. To recover the full-resolution displacement field ϕ in the spatial domain, we first pad the patch $\mathbb{B}_\phi$, containing mostly low frequency signals, to the original image resolution with zeros. We then feed the zero-padded complex-valued coefficients, denoted as $\mathcal{F}(\phi)$, to an iDFT layer consisting of two steps: shifting the Fourier coefficients from centers to corners and then applying the iDFT to convert them into the spatial domain:

$$\phi_{n,m} = \frac{1}{HW} \sum_{k=0}^{H-1} \sum_{l=0}^{W-1} \mathcal{D}_{k,l} [\mathcal{F}(\phi)]_{k,l} \cdot \exp\left(i \cdot \left(\frac{2\pi n}{H} \cdot k + \frac{2\pi m}{W} \cdot l \right) \right). \tag{3.4}$$

The $H \times W$ sized sampling mask $\mathcal{D}$ is a low-pass filter that has zeros as entries if they are on the positions of high-frequency signals in ϕ and ones if they are on the low-frequency positions. Thus we can reconstruct the full spatial displacement field ϕ from $\mathbb{B}_\phi$ despite the latter being band-limited. Approaching the problem from the other side we can also think about working backwards from the final displacement. For this, after applying equation (3.1), the low-frequency signals are shifted to a center patch with size $\frac{H}{a} \times \frac{W}{b}$ with $a = 2 \cdot Z_a, b = 2 \cdot Z_b, Z_a, Z_b \in \mathbb{Z}^+$, which is then center-cropped to get $\mathbb{B}_\phi$. This crop of $\mathcal{F}(\phi)$ can be reconstructed using the iDFT from equation (3.4) with the cropping functioning as a kind of low-pass filtering:

$$[\mathbb{S}_\phi]_{n,m} = \frac{ab}{HW} \sum_{k=1}^{\frac{H}{a}-1} \sum_{l=1}^{\frac{W}{b}-1} [\mathbb{B}_\phi]_{k,l} \cdot \exp\left(i \cdot \left(\frac{2\pi an}{H} \cdot k + \frac{2\pi bm}{W} \cdot l \right) \right), \tag{3.5}$$

with $n \in [0, \frac{H}{a} - 1]$ and $m \in [0, \frac{W}{b} - 1]$ being the indices of the spatial domain, while $k \in [0, \frac{H}{a} - 1]$ and $l \in [0, \frac{W}{b} - 1]$ are the indices of the frequency domain with i being the imaginary unit. Thus $\mathbb{S}_\phi$ actually contains all the necessary information from ϕ, as long as they have the same low-frequency coefficients $\mathbb{B}_\phi$. This can be formulated as:

$$[\mathbb{S}_\phi]_{n,m} = ab \cdot \phi_{an,bm}, \tag{3.6}$$

because most entries of $\mathcal{F}(\phi)$ are zeros, and the remaining values are exactly the same as in $\mathbb{B}_\phi$, which means that $\mathbb{S}_\phi$ contains all the information ϕ can provide. For this, however, $\mathbb{B}_\phi$ needs to be padded to the original image size first in order to get the full-resolution displacement and not a low dimensional representation. This ultimately shows that there is a unique mapping between $\mathbb{S}_\phi$ and ϕ, which means that it is reasonable to use a network to learn $\mathbb{S}_\phi$ directly from image pairs and then reconstruct the displacement field in a very efficient manner [**Fourier-Net**]. The complete reconstruction process is visualized in Figure 3.3.
As both padding and iDFT layers are differentiable, *Fourier-Net* can be optimized via back-propagation. For *Diff-Fourier-Net* extra squaring and squaring layers [**Dalca2018**] are needed in the decoder turning the displacement field into a stationary velocity field. Typically seven scaling and squaring layers are used to impose such a diffeomorphism [**Fourier-Net**, **Dalca2018**].

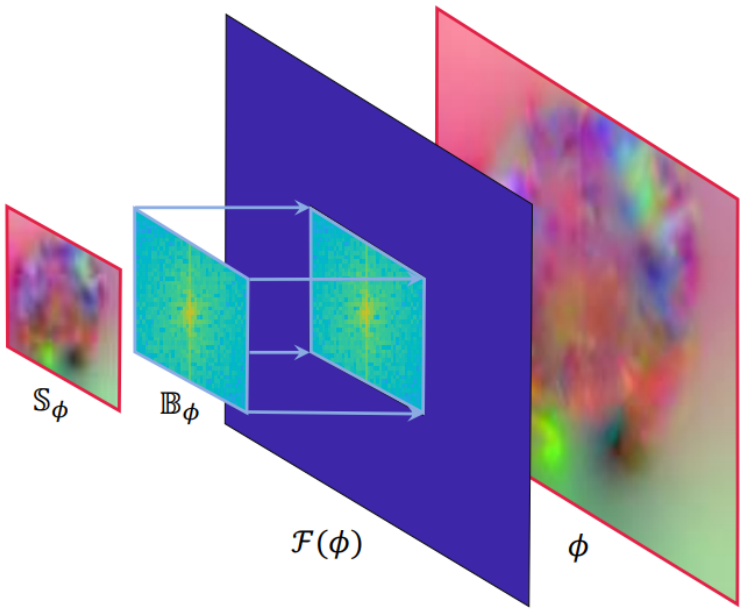


Figure 3.3: Reconstruction of the displacement field via the decoder taken from [**Fourier-Net+**].

Diffeomorphic Transforms

Diffeomorphic deformations are differentiable and invertible, thus preserving topology, which is a desirable property for transformations. $\phi : \mathbb{R}^N \rightarrow \mathbb{R}^N$ represents the deformation that maps the coordinates from one image to coordinates in another image, as long as both images have the dimension N. When using a stationary velocity field representation like e.g. *DARTEL* [**DARTEL**], the deformation field is defined through the following ordinary differential equation (ODE) [**Dalca2018**]:

$$\frac{\partial \phi^{(t)}}{\partial t} = v(\phi^{(t)}), \tag{3.7}$$

where $\phi^{(0)} = \mathrm{Id}$ is the identity transformation and t is time. The final registration field $\phi^{(1)}$ can be obtained by integrating the stationary velocity field v over $t = [0, 1]$. This is typically done by integration numerically using scaling and squaring [**ScaleAndSquare**]. The integration of a stationary ODE represents a one-parameter subgroup of diffeomorphisms. In group theory, v is a member of the Lie algebra and is exponentiated to produce $\exp(v) = \phi^{(1)}$, which is also a member of the Lie group. From the properties of one-parameter subgroups, for any scalars t and t', $\exp((t + t') \cdot v) = \exp(t \cdot v) \circ \exp(t' \cdot v)$, where $\circ$ is a composition map associated with the Lie group. Starting from $\phi^{(1/2^T)} = p + v(p)$ where p is a map of spatial locations, we use the recurrence $\phi^{(1/2)^{t1}} = \phi^{(1/2^t)} \circ \phi^{(1/2^t)}$ to obtain $\phi^1 = \phi^{(1/2)} \circ \phi^{(1/2)}$. T is chosen so that $v \approx 0$ [**Dalca2018**].
As diffeomorphic deformations are defined as smooth and invertible deformations, the output of the iDFT layer in *Fourier-Net* can be regarded as a stationary velocity field v instead of a displacement field ϕ. In Figure 3.1 scaling and squaring layers are visualized. These apply the diffeomorphic transformation in three steps [**ScaleAndSquare**]:

1. Scaling: Divide the velocity field v by a factor 2^N , so that $\frac{v}{2^N}$ is close to zero (depending on the desired accuracy).

2. Exponentiation: Compute $\exp\left(\frac{v}{2^N}\right) = \phi^{(1)}\left(\frac{v}{2^N}\right)$ with a first-order explicit numerical scheme.

3. Squaring: N recursive squarings of $\phi^{(1)}\left(\frac{1}{2^N}\right) = \exp\left(\frac{v}{2^N}\right)$ to yield an accurate estimation of $\phi^{(1)}(1) = \phi^{(1)}\left(\frac{1}{2^N}\right)^{2^N} = \exp\left(\frac{v}{2^N}\right)^{2^N} = \exp(v)$.

Thus, the diffeomorphic deformation can be efficiently calculated. When used, the specific version is then called *Fourier-Net Diff* to differentiate it from the baseline version.

Spatial Transformer

The warping layer of *Fourier-Net* utilizes the *Spatial Transformer* [**SpatialTransformer**], which allows for spatial image manipulation within the network. This is a differentiable and learnable module for neural networks which applies a spatial transformation to a feature map during a single forward pass. The spatial transformer mechanism is split into three parts as seen in Figure 3.4. First is the localization network, which takes the input and outputs the parameters for the transformation. These are then used to create a sample grid using the grid generator. Lastly, the sampler produces the output feature map based on the input at the grid points.
From the input feature map $U \in \mathbb{R}^{H \times W \times C}$ with width W, height H and channels C the localization network f_{loc} computes the parameters $\theta = f_{\mathrm{loc}}(U)$ of the transformation $\mathcal{T}_\theta$ which is later applied to the feature map. Thus the size of θ varies depending on the transformation. The localization network function can both be implemented as a fully-connected network or as a CNN, but should include a final regression layer to produce the transformation parameters.
In order to warp the input feature map, each output pixel is computed by applying a

sampling kernel centered at a particular location in the input feature map. The output pixels are defined to lie on a regular grid $G = G_i$ of pixels, forming an output feature map $V \in \mathbb{R}^{H' \times W' \times C}$, where H' and W' are the height and width of the grid with C again being the number of channels, which is the same for input and output.
In order to perform a spatial transformation of the input feature map U, the sampler must take the set of sampling points $\mathcal{T}_\theta(G)$, along and produce the sampled output feature map V. Each coordinate (x_i^s, y_i^s) in $\mathcal{T}_\theta(G)$ defines the spatial location in the input where a sampling kernel is applied to get the value at a particular pixel in the output:

$$V_i^c = \sum_n^H \sum_m^W U_{nm}^c k(x_i^s - m; \Phi_x) k(y_i^s - n; \Phi_y), \tag{3.8}$$

with Φ_x and Φ_y being the parameters for a generic sampling kernel k that defines the image interpolation, U_{nm}^c is the value of the input feature maps at location (n, m) in the channel $c \in [1, ..., C]$ and V_i^c is the value for every pixel $i \in [1, ..., H'W']$ for the output feature map. Any sampling kernel can be used, as long as (sub-) gradients can be defined with respect to (x_i^s, y_i^s) to allow the loss gradients to flow back not only to the input feature map, but also to the sampling grid coordinates and therefore back to the transformation parameters θ and localization network, thus enabling back-propagation [**SpatialTransformer**].

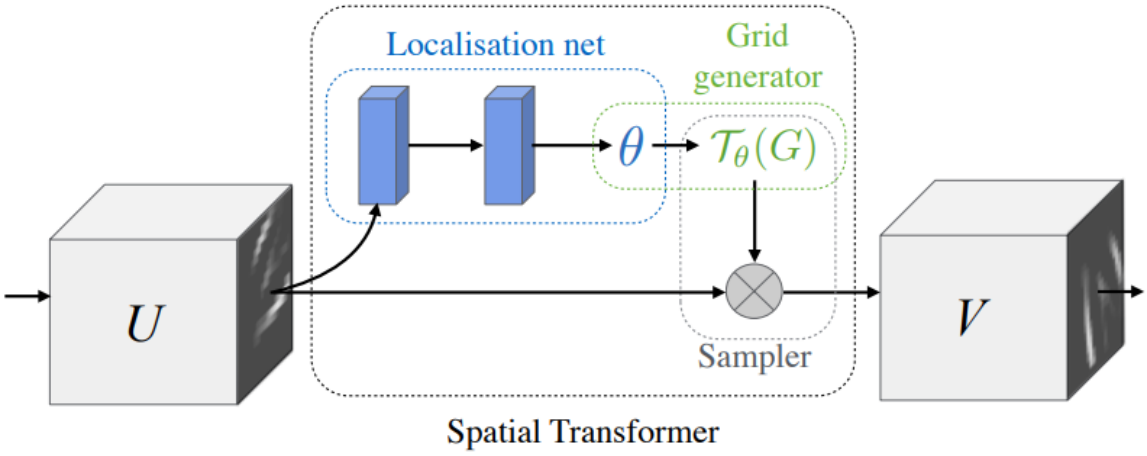


Figure 3.4: Architecture of the *Spatial Transformer* taken from [**SpatialTransformer**].

Loss Function

The loss function consists of two parts to enable unsupervised learning, which are balanced using the scalar parameter λ. The first, $\mathcal{L}_1$, measures the similarity between

the fixed image and the moving image after warping, while the second, $\mathcal{L}_2$, ensures a smooth displacement field. Thus, the unsupervised loss $\mathcal{L}$ can be calculated as follows:

$$\begin{aligned}\mathcal{L}(\Theta) &= \min\Big(\mathcal{L}_1(\phi(\Theta)) + \lambda \cdot \mathcal{L}_2(\phi(\Theta))\Big) \\ &= \min\Big(\mathcal{L}_1(v(\Theta)) + \lambda \cdot \mathcal{L}_2(v(\Theta))\Big),\end{aligned} \tag{3.9}$$

for both displacement fields ϕ and velocity fields v. The first part of the loss function consists of:

$$\mathcal{L}_1(\phi(\Theta)) = \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}_{Sim}(M_i \circ (\phi_i(\Theta) + \mathrm{Id}) - F_i), \tag{3.10}$$

where $\circ$ denotes the warping operation, N the number of training pairs with moving images M_i and fixed images F_i, Θ the network parameters, ϕ_i the displacement field, Id the identity grid. $\mathcal{L}_{Sim}$ determines the similarity between warped moving images and fixed images via MSE or NCC, and the second term of the unsupervised loss, $\mathcal{L}_2$, defines the smoothness regularization function that controls smoothness of the displacement fields:

$$\mathcal{L}_2(\phi(\Theta)) = \frac{1}{N}\sum_{i=1}^{N}||\nabla\phi_i(\Theta)||_2^2, \tag{3.11}$$

with ∇ denoting the first order gradient and $||\cdot||_2^2$ denoting the squared L_2-Norm.
When using the squaring and scaling layers, thus making the deformation of the moving image diffeomorphic, the loss needs to be modified by replacing the displacement field θ with the velocity field v. Thus, both parts of the of the loss function need to be changed:

$$\mathcal{L}_1(v(\Theta)) = \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}_{Sim}(M_i \circ Exp(v_i(\Theta) - F_i), \tag{3.12}$$

$$\mathcal{L}_2(v(\Theta)) = \frac{1}{N}\sum_{i=1}^{N}||\nabla v_i(\Theta)||_2^2 \tag{3.13}$$

3.1.2 Fourier Net+

Fourier-Net+, as the name suggests, is an extension of Fourier-Net which takes the band-limited spatial representation of the images as input, instead of their original full-resolution counterparts. This leads to further reduction in the number of convolutional layers in the contracting path of the network, resulting in a decrease of parameters, memory usage, and computational operations. This makes *Fourier-Net+* even more efficient than its predecessor [**Fourier-Net+**].
As seen in Figure 3.5, the network architecture is almost the same as for *Fourier-Net* (see Figure 3.1 for comparison). However, while the decoder, and thus the loss function, remain the same, the encoder is slightly altered to make the network even more efficient. For this, similarly to the decoder, a DFT is used, however this time the idea is

applied to the input images. These are first transformed into the Fourier domain, then low-pass filtered by center-cropping and finally reconstructed from their band-limited representation back into the spatial domain via an iDFT. The two images, now compressed, are the input for the encoder of *Fourier-Net*, meaning the CNN and following DFT. However, due to the band-limiting before the CNN, the latter can be made much more light-weight, thus reducing computational cost. This is visualized in Figure 3.7. Thus, *Fourier-Net+* too is overall lighter than the baseline *Fourier-Net* in terms of the number of parameters and computations. However, such a light network may face limitations in accurately capturing complex deformations. To counter this potential weakness, the authors propose a cascaded version of *Fourier-Net+*, which uses multiple versions of *Fourier-Net+* cascaded one after the other to achieve a better overall displacement field [**Fourier-Net+**]. A schematic for this can be seen in Figure 3.8.

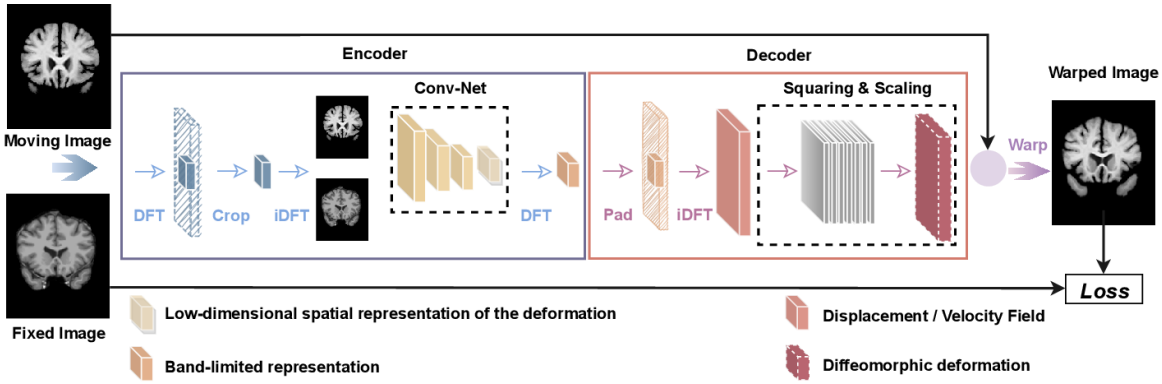


Figure 3.5: Architecture of *Fourier-Net+* taken from [**Fourier-Net+**].

Changes to the Encoder

In order to further reduce the amount of computational operations, *Fourier-Net+* discards the early layers of the encoder. Instead, a DFT $\mathcal{F}(I_M)$ followed by a center-crop to produce the band-limited representation $\mathbb{B}_{I_M}$ in the frequency domain and iDFT are used to get the spatial patch $\mathbb{S}_{I_M}$, while the rest of the encoder from *Fourier-Net* stays the same. The input I_M is thus compressed to a lower resolution (i.e. band-limited) using the frequency space, which reduces the computational cost. This process is visualized in Figure 3.6. The encoder of *Fourier-Net+* has several convolutional layers less in the contracting path compared to *Fourier-Net*, which leads to a further accelerated registration process while reducing the memory footprint. These advantages are visualized in Figure 3.7 where the amount of different layers between a conventional *U-Net*, *Fourier-Net* (with the smaller decoder) and *Fourier-Net+* (with a smaller encoder and decoder) are shown.

Effects of Cascading

As seen in the previous section, *Fourier-Net+* is lighter than *Fourier-Net* due to the band-limited representation of both images and deformations lowering the number

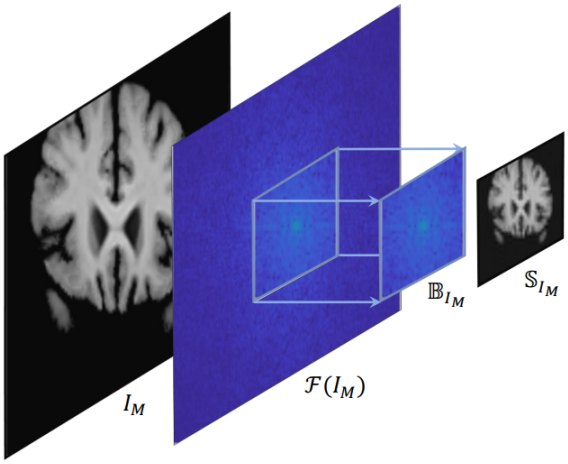


Figure 3.6: Compression in the frequency domain of the encoder used in *Fourier-Net+* taken from [**Fourier-Net+**].

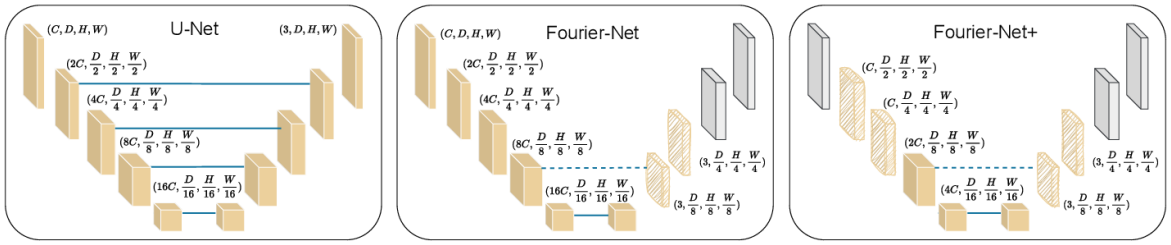


Figure 3.7: Architecture of the CNN for a typical U-Net, *Fourier-Net* and *Fourier-Net+* taken from [**Fourier-Net+**].

of parameters and computations. This, however, can lead to limitations when trying to accurately capture complex deformations. To this end, a cascaded version of *Fourier-Net+* called *K*×*Fourier-Net+* (see Figure 3.8), with K denoting the amount of cascades, can be used where the the warped image of one cascade is used as the moving image of the next. It is important to note that the weights are not shared between cascades. The squaring and scaling layers, in case a diffeomorphic deformation is wanted, are applied after the last cascade. The same is true for the calculation of the loss function.
In order to more accurately describe this version of *Fourier-Net+* one can look at the in- and outputs of the different cascades. The first cascade of *K*×*Fourier-Net+* has the moving image I_M and fixed image I_F as inputs. While the latter always stays the same for all cascades the warped image $I_M^{w(1)}$ from the first cascade is used as input for the second cascade instead of the original moving image. Thus, in general $I_M^{w(k-1)}$ and I_F are the inputs for a cascade $k \in [1, K]$ with output $\delta\phi^{(k)}$. Furthermore, $I_M^{w(k)}$ can be defined as:

$$I_M^{w(k)} = ((((I_M \circ \delta\phi^{(1)}) \circ \delta\phi^{(2)}) \circ \dots) \circ \delta\phi^{(k-1)}) \circ \delta\phi^{(k)} = I_M \circ \phi^{(k)}, \tag{3.14}$$

with $\phi^{(k)}$ being the displacement field computed by composing the outputs of the first cascade up to the k-th cascade:

$$\phi^{(k)} = \delta\phi^{(1)} \circ \delta\phi^{(2)} \circ \dots \circ \delta\phi^{(k-1)} \circ \delta\phi^{(k)}. \tag{3.15}$$

Thus, the output displacement field of the K-th cascade $\phi^{(K)}$ is the final displacement which is then used to warp I_M, the original moving image, in order to compute the loss [**Fourier-Net+**].

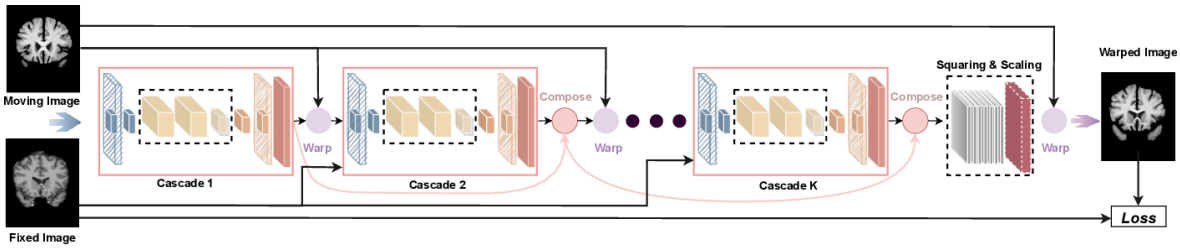


Figure 3.8: Cascaded version of *Fourier-Net+* taken from [**Fourier-Net+**].

3.2 Datasets

In the following chapter, the datasets used in this thesis are presented and potential pre-processing steps, as well as uses, discussed. The *ACDC* cardiac dataset was mainly used for ablation studies and parameter tests. While it contains no k-space data, segmentations are given for the end-systolic and end-diastolic frames of the test set, which can be used to evaluate the registration performance.

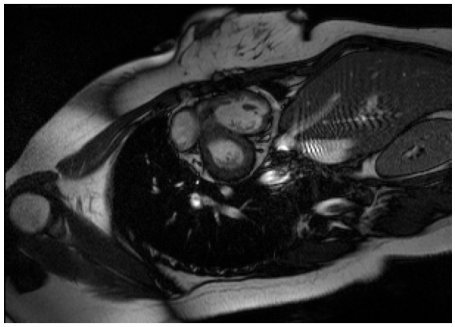
The *CMRxRecon* dataset was used for the downstream-tests regarding MR image reconstruction. The cardiac MRI k-space data was used for extending the networks to frame-to-frame registration in order for them to be integrated into a motion reconstruction pipeline. It contains subsampled data, but does not provide segmentations for multi-coil data.

3.2.1 ACDC Dataset

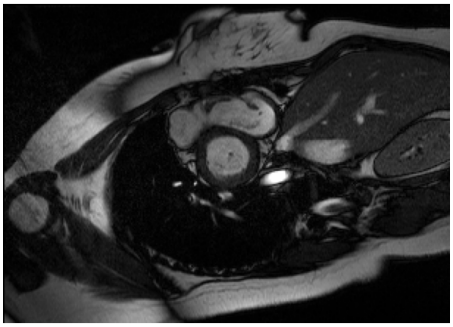
The *ACDC* dataset [**ACDC**] from the *Automated Cardiac Diagnosis Challenge (ACDC)* during the *MICCAI 2017* conference. It contains cardiac cine-MRI short-axis data from 150 subjects that were divided into 5 subgroups (4 pathological, 1 healthy). For each subject systolic (see Figure 3.9a) and diastolic frames (see Figure 3.9b) are provided with corresponding segmentations (see Figure 3.9c and 3.9d), which enables direct comparison via e.g. the Dice score. The segmentations contain only four values with 0, 1, 2 and 3 representing pixels located in the background, in the RV cavity, in the myocardium, and in the LV cavity. The frames themselves are 3D volumes with size $216 \times 256 \times 10$, thus ten image slices can be extracted which each have size 216×256. The same holds true for the segmentation, which means that both the image data and segmentations can generated in the same manner.
For the training data, the original 4D data was used as we do not require the segmentations for the end-systolic and end-diastolic frames. As the 4D data has size $216 \times 256 \times 10 \times 30$ we can extract 30 frames for each of the ten slices. These can be sorted into 251376 image pairs for training. For the validation and test data the end-systolic and end-diastolic frames with their segmentations were used given us 641 image pairs each as we can only have these two frames to align.

3.2.2 CMRxRecon Dataset

The *CMRxRecon* dataset [**CMRxRecon**] from the *CMRxRecon2023* challenge specializes in Cardiac magnetic resonance imaging (CMR). The dataset includes fully sampled and subsampled multi-coil k-space data, as well as auto-calibration lines. This includes imaging of different anatomical views like long-axis (2-chamber, 3-chamber, and 4-chamber) and short-axis (SAX). There is a total of 120 training data, 60 validation data, and 120 test data from healthy volunteers. One of the goals of the challenge is the reconstruction from subsampling and motion correction for the heart movement. As the data was recorded in the k-space (see Figure 3.11) and stored as *.mat* files we first need to reconstruct the images to use them for training and testing of *Fourier-Net/Fourier-Net+*. For this, multi-coil reconstruction algorithms like *SENSE* [**SENSE1**] can be used. The sensitivity of the different coils can be seen in Figure 3.10, where each coil focuses on a specific area of the image. These images are then stitched together in reconstruction to produce a combined image with great overall contrast. The dataset, as mentioned before, contains fully sampled as well as subsampled k-space data (see Figure 3.11a and Figure 3.11b), with the latter being typically used to accelerate the MRI acquisition process. This is done by not using all of the available k-space data, but



(a) Image of systolic phase.



(b) Image of diastolic phase.

(c) Segmentation of systolic phase.

(d) Segmentation of diastolic phase.

Figure 3.9: Example images for systolic (a) and diastolic frames (b) with corresponding segmentations (c),(d) taken from the *ACDC* dataset [**ACDC**].

rather masking the signal to achieve the subsampling. The center region of the k-space is always fully sampled, but the outer regions are subsampled depending on the sampling strategy [**SamplingStrategies**]. The dataset contains subsampled k-space data for 4x, 8x and 10x acceleration with the latter of course leading to more distortions in the reconstructed image as the subsampling can induce image reconstruction artifacts. This can be seen in Figure 3.11d, where the image reconstructed from 4x accelerated ($R = 4$) k-space data seems blurred when compared to the fully sampled ($R = 0$) one in Figure 3.11c. For all experiments the short-axis view data was used. As image sizes between patients can vary slightly, interpolation was used to standardize the images to a size of 246×512 to avoid further problems with e.g. loss calculation. Additionally, min-max-normalization is used to standardize the data range for all images to $[0, 1]$ as this was also varying. The reconstructed images were stored for every image slice for every patient. Thus, all frames for a specific image slice were in a single folder to enable easy access for later data load-in for frame-to-frame registration.

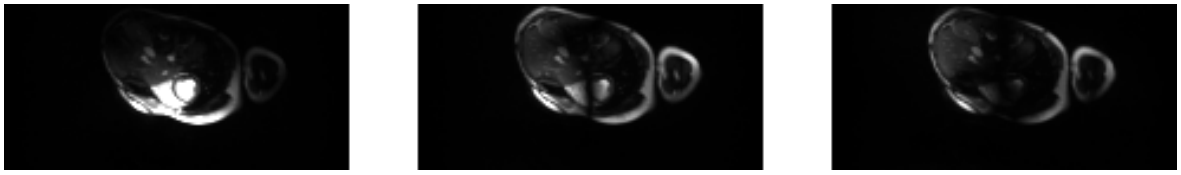
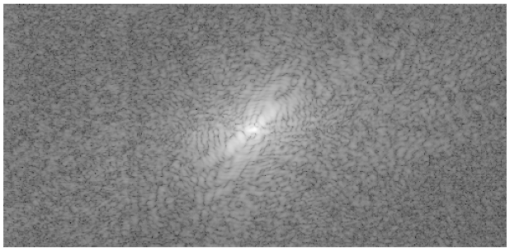
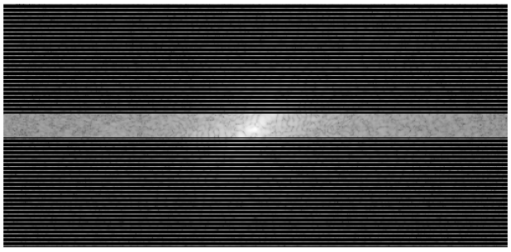


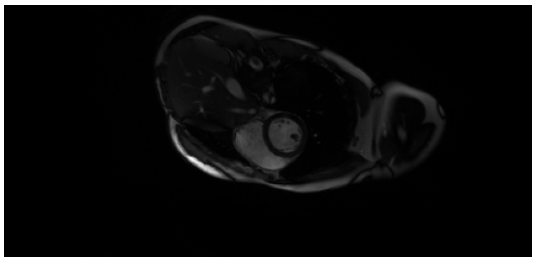
Figure 3.10: MRI images for three different coils from the *CMRxRecon* dataset [**CMRxRecon**].



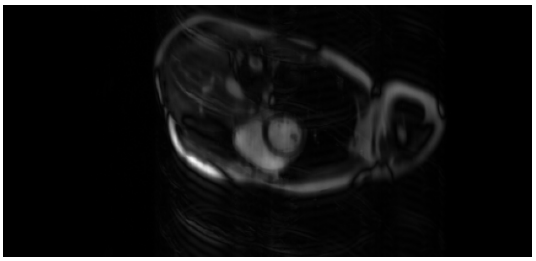
(a) Example for a fully sampled k-space with the low frequencies in the middle.



(b) Example of a subsampled k-space for 4x acceleration.



(c) Corresponding image reconstructed from the fully sampled k-space.



(d) Corresponding image reconstructed from the subsampled k-space.

Figure 3.11: Fully sampled and subsampled k-space data from the from the *CMRxRecon* dataset [**CMRxRecon**] with corresponding images.

3.3 Experiments

In the following chapter the different experiments that were conducted are described and explained. First, experiments were conducted on the *ACDC* dataset to find the optimal parameters for our model using ground truth segmentations. Then a motion-compensated reconstruction pipeline was used as an down-stream task on the *CMRxRecon* dataset.

3.3.1 Parameter Tests on the ACDC Dataset

As the evaluation on the *CMRxRecon* dataset is limited to image similarity measures, further testing was done on the *ACDC* dataset which contains cardiac data with segmentations. These can be used to better evaluate the registration performance by using e.g. the Dice score.

Fourier-Net versus Fourier-Net+

First, *Fourier-Net*, *Fourier-Net+* and *4xFourier-Net* are compared for both normal and diffeomorphic versions. Additionally, network versions creating dense instead of band-limited displacements were used a baseline for the best achievable performance. The registration performance was evaluated on the test set using the percentage of Dice scores calculated with and without the background label, percentage of SSIM, MSE multiplied by $10^{-}3$ (denoted by m for milli) and the percentage of negative Jacobian determinant of deformation. The mean inference time on the GPU together with memory consumption for each model are also added to the comparison with the latter containing number of trainable model parameters, mult-add operations (in millions or billion as denoted by M or G) and the total memory in Megabyte (MB). The latter is only given for the normal versions as the diffeomorphic version require the same amount of memory as well as the baseline dense versions. The first network was trained for a maximum of 15 epochs with early stopping activating after 3 epochs without improvement of the validation Dice score. The other variants were than trained with the same number of epochs, which worked out to be only 6 epochs due to the large training set in the experiments without any data augmentation. All networks were trained on the fully sampled *ACDC* data with MSE as the unsupervised similarity loss, a channel size of 8, learning rate of 0.0001 and $\lambda = 0.01$. *Fourier-Net+* and *4xFourier-Net* used a FT crop size of 48×48 for these experiments. All of these parameters were constant for the diffeomorphic and dense versions of the networks. The results and subsequent discussion can be found in section 4.1.1.

Starting Channel Size

Next, the impact of the starting channels on the *ACDC* data was examined as these dictate the number of features that the network uses. For this, *Fourier-Net+* and *4xFourier-Net+* was trained with MSE-loss, a learning rate of 0.0001, $\lambda = 0.01$, FT crop of 24×24 and no diffeomorphic transform. The experiments were again only conducted on the fully sampled data. Four different channel sizes were used for training: 8, 16, 32 and 64. The first network variant was trained with early stopping, which ended at 6 epochs. For better comparability, the other network variants were also trained with the same number of epochs. The registration performance was evaluated on the test set using the percentage of the Dice score calculated once on all labels, once without the background using only the cardiac labels, as well as the percentage of SSIM, MSE multiplied by $10^{-}3$ and the percentage of negative Jacobian determinant of deformation. Additionally the number of network parameters, number of mult-add operations (in millions or billion as denoted by M or G) and memory consumption in Megabyte (MB) are listed for all network variants to better compare the relationship between registration performance and memory consumption to potentially find a optimal configuration for both performance and efficiency. The mean inference time was measured on GPU in seconds. The results and subsequent discussion can be found in section 4.1.2.

Fourier-Transform Crop Size

The FT crop size, used for compressing the input images, is a parameter specific to *Fourier-Net+* and *4xFourier-Net+*. Four different sizes of the FT crop were analyzed: 80×168, 40×84, 48×48 and 24×24. A larger crop would obviously compress the images less, thus leading to less efficiency, however, a very small crop, while very efficient, might lose some of the image details leading to a decrease in performance. The experiments were again only conducted on the fully sampled data as one would expect similar results on subsampled data. The network variants were trained with MSE as the unsupervised similarity loss, a channel size of 8, learning rate of 0.0001 and $\lambda = 0.01$. Again, the networks were trained for only 5 epochs, as this is were the first model ended training due to early stopping. The registration performance was evaluated on the test set using the Dice score calculated once on all labels, once without the background, as well as the SSIM, MSE and the percentage of negative Jacobian determinant of deformation. Additionally, the number of network parameters, number of mult-add operations in millions (M) and memory consumption in Megabyte (MB) are listed for all network variants to better compare the relationship between registration performance and memory consumption to potentially find a optimal configuration for both performance and efficiency. The mean inference time was measured on GPU in seconds. The results and subsequent discussion can be found in section 4.1.3.

Comparison with VoxelMorph

After finding parameters which optimize registration performance while trying to minimize memory consumption, a comparison to another commonly used registration network should be made. For this, a diffeomorphic version [**VoxelMorphDiff**] of the well known *VoxelMorph* [**Voxelmorph**] was used with integration steps $T = 7$. It was trained with the default number of layers, MSE as the similarity loss, a learning rate of 0.0001, $\lambda = 0.01$ and provides a dense displacement field to align the image pair. For comparison *Fourier-Net*, *Fourier-Net+* and *4xFourier-Net+* were trained for 6 epochs with MSE as the unsupervised similarity loss, a channel size of 16, learning rate of 0.0001 and $\lambda = 0.01$ as well as a FT crop size of 48×48, which provide a good balance of performance and memory imprint for the networks. To evaluate registration performance, the Dice score was again computed for all labels and without the background as well as the similarity metrics SSIM, MSE and the percentage of negative Jacobian determinant of deformation. All networks were trained on the fully sampled *ACDC* data. The inference times of all networks were computed on the GPU and the number of network parameters, number of mult-add operations in billions (G) and memory consumption in Megabyte (MB) are also given to further compare the efficiency of the networks. The results can be seen in section 4.1.4.

Dense Displacement on Accelerated Data

The difference between a dense displacement field and a band-limited one was already explored in section 3.3.1.1. This, however, only includes fully sampled data, not accel-

erated data. This section explores potential performance changes on the latter with the hypothesis that the network variants with the dense displacement will perform worse than the variants with the band-limited displacement field for strongly subsampled data. This expectation stems from the fact that frequencies outside the center region of the k-space are dropped for the accelerated data thus reducing the advantage of the dense displacement in comparison to the band-limited (i.e. center-cropped) displacement.
The tests again included *Fourier-Net*, *Fourier-Net+* and *4xFourier-Net+*, which were all trained for 6 epochs with MSE as the unsupervised similarity loss, a channel size of 16, learning rate of 0.0001 and $\lambda = 0.01$. The registration performance was evaluated on the test set using the Dice score calculated once on all labels, once without the background, as well as the SSIM, MSE and the percentage of negative Jacobian determinant of deformation. Additionally, the number of network parameters, number of mult-add operations in millions (M) and memory consumption in Megabyte (MB) are given.

Comparison on Subsampled Data

After comparing *Fourier-Net*, *Fourier-Net+* and *4xFourier-Net+* with both dense and band-limited displacement fields as well as another comparison with *VoxelMorph* on fully sampled data an extension to subsampled data needs to be made. For this *Fourier-Net*, *Fourier-Net+* and *4xFourier-Net+* were compared to a baseline consisting of the unaligned test image pairs as well as *NiftyReg*, *VoxelMorph* and *LAPNet*. Thus, observations of the performance on accelerated data can be made as well as comparisons to the aligned baseline (which all methods should aim to beat), a traditional registration algorithm, a dense registration network working in the image domain and a another dense network which works in the k-space domain.
Fourier-Net, *Fourier-Net+* and *4xFourier-Net+* were trained for 6 epochs with MSE as the unsupervised similarity loss, a channel size of 16, learning rate of 0.0001 and $\lambda = 0.01$ as well as a FT crop size of 48×48 for all subsampling data. VoxelMorph was trained with default layers and MSE as the similarity loss, a learning rate of 0.0001 and $\lambda = 0.01$. Three different acceleration factors were tested: $R = 4$, $R = 8$ and $R = 10$. For reference, the data for the fully sampled data ($R = 0$) is also provided. The registration performance of all methods was evaluated on the test set using the Dice score calculated once on all labels, once without the background, as well as the SSIM and MSE as image metrics. All times are computed on CPU for better comparability as *NiftyReg* currently only works on CPU as the GPU capable versions were deprecated and the other networks would gain an unfair time advantage when ran on GPU. The baseline obviously has no time associated as it only denotes the metrics on the test data before registration. The results and an comprehensive discussion are provided in section 4.1.6.

3.3.2 Integration into a Motion-Compensated Reconstruction Pipeline

Describe qualitative test to see whether the network can be used to improve a reconstruction pipeline for cardiac data.

Domain Translation

Before implementing the reconstruction pipeline the domain translation capabilities of *Fourier-Net*, *Fourier-Net+* and *4xFourier-Net+* were tested. For this the three networks, previously trained on *ACDC*, were trained again from scratch on *CMRxRecon*. They were then compared on the *CMRxRecon* test data to see how similar the performance would be as the cardiac scans of the two datasets are fairly similar.

Reconstruction Pipeline

After ensuring that the previous results were translatable to the new data, the reconstruction pipeline had to be evaluated. As discussed in section **??**, neural networks can be used in motion-compensated MRI reconstruction pipelines for either the reconstruction or the corrected of motion between the frames. As *Fourier-Net*, *Fourier-Net+* and *4xFourier-Net+* were already trained for frame-to-frame registration, they could be easily used for the latter. ESPIRiT [**ESPIRiT**] was used to simulate the coil sensitivity maps needed for a SENSE-type reconstruction from the provided k-space data using an eigenvalue decomposition. For the SENSE reconstruction iterative conjugate gradient was used as it is very memory effectiv due to *Fourier-Net+* being a very small network and the fact that no big network is needed for the actual reconstruction as a lightweight algorithm does the work.
This experiment was conducted on the *CMRxRecon* dataset as it provides the needed k-space data (both fully sampled and accelerated) as well as the subsampling masks required for reconstruction. However, the cardiac scans only show a small amount of movement between frames due to cardiac movement no larger movement of e.g. the lung. Thus, the input images were further motion-corrupted as the cardiac motion present was deemed not severe enough. To simulate this motion or mis-triggering, a similar strategy to [**Oksuz2020**] was used, swapping $z = \{16, 32\}$ k-space lines between the frames in k-space. These new fully sampled frames were then subsampled using the subsampling masks before an inverse FT to generate the corrupted images for the pipeline. An overview of the whole reconstruction pipeline can be seen in Figure 3.12.
Fourier-Net, *Fourier-Net+* and *4xFourier-Net+* have been pre-trained on image reconstructed from this data via an inverse FT for frame-to-frame registration using the optimal settings obtained in section 3.3.1 (MSE-loss, 16 channels, $\lambda = 0.01$). As the *CMRxRecon* dataset does not include segmentations for Dice calculation, image similarity metrics need to be utilized. For this, SSIM and MSE were used again, however, the PSNR (see section **??**) and HaarPSI [**HaarPSI**] were also added. For comparison *VoxelMorph* was used. The results can be seen in section **??**.

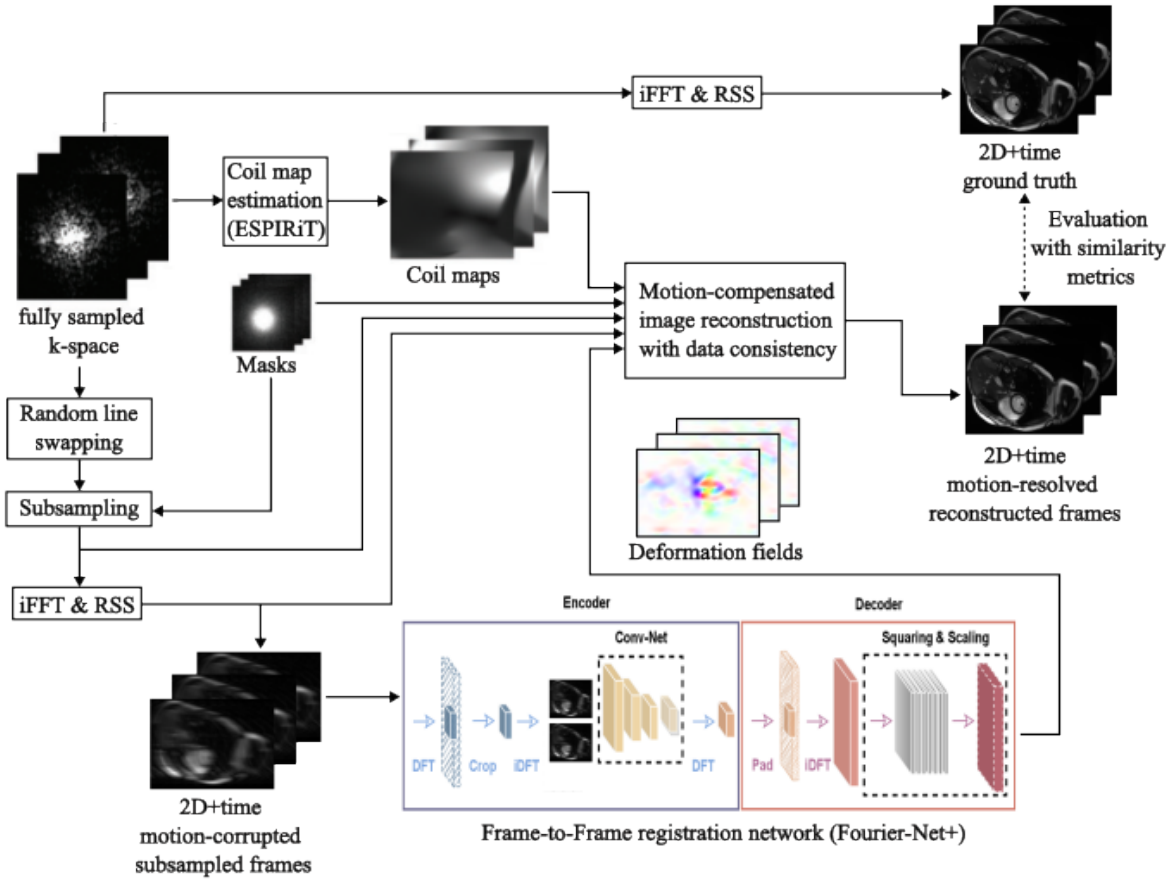


Figure 3.12: Overview of the general reconstruction pipeline using *Fourier-Net+* as a registration network (inspired by [**Kuestner2022**]).

Chapter 4

Results

In this chapter, the results for the experiment described in the previous chapter will be examined.

4.1 Parameter Tests on the ACDC Dataset

Due to the limitations of the *CMRxRecon* dataset in terms of segmentations, the registration performance is further evaluated on the *ACDC* dataset. Some of the parameter studies done on *CMRxRecon* were repeated, others were added to further test the networks ability.

4.1.1 Fourier-Net versus Fourier-Net+

Again, *Fourier-Net*, *Fourier-Net+* and *4xFourier-Net* are compared in terms of registration performance and inference time (on GPU) together with their diffeomorphic versions, however, versions with dense instead of band-limited displacements are also added as a baseline for peak performance as these utilize all of the k-space information. The memory consumption is also added to the comparison with the number of parameters, mult-add operations and the total memory. The latter is only given for the dense and normal versions as the diffeomorphic version required roughly the same amount of memory as the normal ones, while the dense versions have a slightly larger encoder. Note that all network variants were trained for 6 epochs. The results can be seen in Table 4.1.
For the dense displacement variants the diffeomorphic transform increases the performance in terms of Dice score (with and without the background label) slightly for *Fourier-Net* and *4xFourier-Net*, but decreases for *Fourier-Net+*. The SSIM decreases for all models slightly, while the MSE stays about the same. Unsurprisingly, the percentage of non-positive Jabobian determinants goes down for the diffeomorphic variants. The times between the baseline and diffeomorphic versions do not vary a lot with the *Fourier-Net* being faster without the diffeomorphis, while *Fourier-Net+* and *4xFourier-Net* are faster.
The results for the band-limited versions are very similar with *Fourier-Net* and *Fourier-Net+* performing slightly worse in terms of Dice (with background) with the diffeomorphis, while *4xFourier-Net* performs better by almost 2%. When excluding the background label *Fourier-Net* again performs slightly worse, but *Fourier-Net+* and

4xFourier-Net both improve, the latter with almost 3% more than the baseline version. These changes are not consistent with the dense displacement versions, however, the band-limited *4xFourier-Net* version might be an outlier, as all of the other models are not affected as much by the diffeomorphic transform. The SSIM values slightly decrease for *Fourier-Net* and *Fourier-Net+* with the diffeomorphism, but again improve slightly for *4xFourier-Net*. The MSE is slightly better for *Fourier-Net* and *4xFourier-Net*, but a bit worse for *Fourier-Net+*. The percentage of non-positive Jabobian determinants again decreases for the diffeomorphic variants, which is consistent with the previous observations on the dense displacement versions. The times again vary slightly with *Fourier-Net* and *Fourier-Net+* being a bit slower, while *4xFourier-Net* is quite a bit faster.
Now to the difference between the dense and band-limited displacement. Overall, the dense displacement versions perform better in terms of Dice score for all models both with and without the background label. This is to be expected as the dense displacement versions can use all of the available k-space data for the registration task, while the band-limited versions have only a limited amount of k-space data, directly impacting and limiting their performance. The dense displacement is also better in the SSIM metric, however the difference is not quite as large as with the Dice scores. As the differences between the frames is not quite as large, the weaker registration performance does not impact this metric as much as the Dice score, which is focused on the moving cardiac region. The same is true for the MSE, but the dense displacement is again far superior. Only in terms of the percentage of non-positive Jabobian determinants does the band-limited displacement actually perform better as the band-limiting probably allows for only a smaller deformation. However, the difference is still very small, especially for the diffeomorhpic versions, and it could be argued that a better registration performance at the cost of some image folding is an acceptable trade-off. In terms of inference time the dense displacement variants are surprisingly slightly faster despite having a larger encoder, however all of the model are in the low milliseconds ($< 40\,ms$). Last but not least, the memory consumption needs to be addressed. The dense displacment variants have a larger encoder as discussed before and thus have a lot more parameters, especially for the more optimized *Fourier-Net+* and *4xFourier-Net* (about 5 times more). The number of Mult-Adds and the amount of total memory are more than doubled (almost tripled for *4xFourier-Net*) for the dense displacement versions compared to those with a band-limited displacement across all models.

Table 4.1: Results for *Fourier-Net*, *Fourier-Net+* and *4xFourier-Net+* with both dense and band-limited displacement fields as well as diffeomorphic transforms on the fully sampled *ACDC* test data.

Metrics	Dense Displacement		
	Fourier-Net	Fourier-Net+	4xFourier-Net+
% Dice	78.22 ± 13.84	78.43 ± 13.96	79.34 ± 14.03
% Dice*	71.03 ± 18.87	70.50 ± 19.17	72.02 ± 18.94
% SSIM	91.92 ± 3.32	89.09 ± 4.10	89.65 ± 3.85
MSE (m)	0.09 ± 0.06	0.17 ± 0.13	0.15 ± 0.12
$\% \lvert J_\phi \rvert \leq 0$	0.53 ± 0.52	0.32 ± 0.55	0.04 ± 0.09
Time [s]	0.0070	0.0107	0.0219
	Diff-Fourier-Net	Diff-Fourier-Net+	Diff-4xFourier-Net+
% Dice	78.56 ± 14.13	77.88 ± 13.81	79.49 ± 14.26
% Dice*	71.31 ± 19.13	70.19 ± 18.82	72.12 ± 19.33
% SSIM	91.79 ± 3.38	89.08 ± 4.10	89.62 ± 3.98
MSE (m)	0.09 ± 0.06	0.16 ± 0.13	0.15 ± 0.12
$\% \lvert J_\phi \rvert \leq 0$	0.03 ± 0.06	0.00 ± 0.00	0.00 ± 0.00
Time [s]	0.0082	0.0075	0.0203
Parameters	645,216	380,470	1,521,880
Mult-Adds (G)	1.12	0.04	0.18
Memory [MB]	97.63	5.81	21.90
Metrics	Band-limited Displacement		
	Fourier-Net	Fourier-Net+	4xFourier-Net+
% Dice	77.95 ± 14.71	75.35 ± 14.28	76.59 ± 14.84
% Dice*	69.96 ± 21.55	64.74 ± 21.12	66.82 ± 21.55
% SSIM	91.35 ± 3.51	88.42 ± 3.94	88.59 ± 4.23
MSE (m)	1.00 ± 0.71	2.06 ± 1.54	1.92 ± 1.46
$\% \lvert J_\phi \rvert \leq 0$	0.36 ± 0.38	0.13 ± 0.35	0.03 ± 0.12
Time [s]	0.0145	0.0145	0.0335
	Diff-Fourier-Net	Diff-Fourier-Net+	Diff-4xFourier-Net+
% Dice	77.93 ± 14.93	75.17 ± 13.43	78.20 ± 14.01
% Dice*	69.91 ± 21.99	65.67 ± 18.61	69.61 ± 19.11
% SSIM	91.24 ± 3.56	87.81 ± 3.94	88.81 ± 4.13
MSE (m)	0.98 ± 0.67	2.29 ± 1.61	1.85 ± 1.41
$\% \lvert J_\phi \rvert \leq 0$	0.01 ± 0.03	0.00 ± 0.00	0.00 ± 0.00
Time [s]	0.0150	0.0207	0.0188
Parameters	434,519	75,429	301,716
Mult-Adds (M)	595.04	10.98	43.91
Memory [MB]	44.69	2.25	7.66

4.1.2 Starting Channel Size

Next, the impact of the starting channels was examined. For this, *Fourier-Net+* and *4xFourier-Net+* were trained with four different channel sizes: 8, 16, 32 and 64. The memory consumption as well as inference time on the GPU were calculated to extend the evaluation given by the other metrics. All network variants were trained for 6 epochs. The results can be seen in Table 4.2.
For both *Fourier-Net+* and *4xFourier-Net+* there is a clear increase in Dice (with and without the background label) with larger channel sizes as more features increase the registration performance. The SSIM and MSE metrics only increase marginally with larger channel size. The percentage of non-positive Jabobian determinants actually decreases as the registration performance increases, likely due to better/more features enabling a smoother and better displacement field to be generated by the models.
The inference time is not heavily impacted by channel size as the times remain very similar for all sizes. The memory however increases exponentially when doubling the channel sizes as all layers are effected, not just the starting one as the name suggests. Thus the number of parameters drastically increases with starting size leading to an increase in Mult-Add and thus overall memory. As *4xFourier-Net+* is just a cascaded version of *Fourier-Net+* a channel size of 16 for the latter is about the same as channel size 8 for the cascaded version and so on. Thus it seems that there is no point at which both the performance and memory consumption are maximized at the same time. This can also be seen in the direct comparison between *Fourier-Net+* and *4xFourier-Net+*, where the latter has a better performance, but also needs more memory. While an increase of channel size also increases performance in terms of Dice (with the background label) by 0.36%, 0.43% and 0.41% for the latter, which is quite consistent, it can be observed that an increase of channel size for *Fourier-Net+* has an bigger impact at the beginning (1.38%, 0.78% and 0.15%) and fades towards larger channels sizes. This effect can also be seen when excluding the background label (0.69%, 0.78% and 0.42% for *4xFourier-Net+* compared to 2.16%, 1.09% and 0.10% for *Fourier-Net+*), but is less pronounced for the SSIM and MSE as these do not capture the difference in the cardiac regions as well. Note that all versions of *Fourier-Net+* and *4xFourier-Net+* are smaller in terms of total memory than *Fourier-Net* with channel size 8 (44.69 MB), except for *4xFourier-Net+* with channel size 64.

4.1.3 Fourier-Transform Crop Size

As a parameter specific to *Fourier-Net+* and *4xFourier-Net+*, the impact of the FT crop size used for compressing the images was analyzed. Four different sizes of the FT crop were used for training: 80×168, 40×84, 48×48 and 24×24. The experiments were again only conducted on the fully sampled data and each network trained for 5 epochs. The results can be seen in Table 4.3.
The registration performance, as measured by the Dice score both with and without the background label, decreases drastically with a smaller crop size. This is as expected as the smaller FT drop compresses the images more thus making an accurate registration harder. The SSIM and MSE also get worse (SSIM decreases, MSE increases), however

Table 4.2: Results for different starting channel sizes of *Fourier-Net+* and *4xFourier-Net+* on the fully sampled *ACDC* test data.

Metrics	Starting Channels - Fourier-Net+			
	8	16	32	64
% Dice	75.50 ± 13.79	76.88 ± 13.86	77.66 ± 13.60	77.81 ± 13.76
% Dice*	65.70 ± 18.96	67.86 ± 19.06	68.95 ± 18.65	69.05 ± 18.80
% SSIM	88.05 ± 3.89	88.67 ± 3.88	88.83 ± 3.83	89.02 ± 3.67
MSE (m)	0.22 ± 0.16	0.20 ± 0.15	0.19 ± 0.15	0.19 ± 0.15
$\% \lvert J_\phi \rvert \leq 0$	0.07 ± 0.25	0.04 ± 0.14	0.01 ± 0.04	0.00 ± 0.03
Time [s]	0.0072	0.0072	0.0080	0.0079
Parameters	75,429	300,477	1,199,469	4,793,037
Mult-Adds (M)	10.98	42.89	169.54	674.10
Memory [MB]	2.25	4.64	11.22	31.57
Metrics	Starting Channels - 4xFourier-Net+			
	8	16	32	64
% Dice	77.54 ± 13.73	77.90 ± 13.92	78.33 ± 14.14	78.74 ± 13.91
% Dice*	68.52 ± 18.62	69.21 ± 19.02	69.99 ± 19.21	70.41 ± 19.02
% SSIM	88.70 ± 4.17	88.89 ± 4.05	89.08 ± 4.01	89.29 ± 3.74
MSE (m)	0.19 ± 0.14	0.19 ± 0.14	0.18 ± 0.14	0.18 ± 0.14
$\% \lvert J_\phi \rvert \leq 0$	0.02 ± 0.07	0.01 ± 0.04	0.01 ± 0.05	0.00 ± 0.00
Time [s]	0.0301	0.0244	0.0297	0.0277
Parameters	301,716	1,201,908	4,797,876	19,172,148
Mult-Adds (G)	0.04	0.17	0.68	2.70
Memory [MB]	7.66	17.23	43.56	124.94

the amount of performance lost is hard to gauge, especially for the MSE metric. Interestingly, the percentage of non-positive Jabobian determinants actually decreases for smaller crop sizes, perhaps because the displacements on the compressed images are not as extreme and thus more smooth. Note however, that the crop size of 48×48 is a bit of an outlier in this regard breaking the trend with a higher percentage as 40×84. The time also seems to decrease slightly with a smaller crop, although this is quite noisy as the time from 80×168 to 40×84 is halved for *Fourier-Net+*, but the time needed for 48×48 and 24×24 increases again very slightly. For *4xFourier-Net+* the time decrease for all smaller crop sizes, except for 24×24.
The number of parameters does not change for different crop sizes as the networks themselves do not change, however the number of Mult-Adds and the total memory still change with the image size. Both decrease with a larger crop size as the image gets smaller. This effect is again not linear as a reduction in image size yields a larger reduction in memory going from 80×168 to 40×84 than from 48×48 to 24×24.

Again, there is no sweet-spot to be found that maximizes both memory efficiency and registration performance similar to the experiments in the previous chapter.

Table 4.3: Results for four different FT crop sizes for *Fourier-Net+* and *4xFourier-Net+* examined on the fully sampled *ACDC* test data.

Metrics	FT crop size - Fourier-Net+			
	80×168	40×84	48×48	24×24
% Dice	78.24 ± 14.43	76.61 ± 13.90	75.27 ± 13.49	73.77 ± 14.42
% Dice*	70.66 ± 19.70	67.60 ± 19.24	65.48 ± 18.76	63.37 ± 20.07
% SSIM	89.81 ± 4.09	88.58 ± 3.87	87.98 ± 3.91	87.00 ± 3.99
MSE (m)	0.15 ± 0.12	0.20 ± 0.15	0.22 ± 0.16	0.27 ± 0.19
$\% \lvert J_\phi \rvert \leq 0$	0.22 ± 0.46	0.05 ± 0.15	0.09 ± 0.25	0.00 ± 0.02
Time [s]	0.0158	0.0077	0.0079	0.0081
Parameters	75,429	75,429	75,429	75,429
Mult-Adds (M)	64.03	16.37	10.98	2.74
Memory [MB]	9.51	2.97	2.25	1.12
Metrics	FT crop size - 4xFourier-Net+			
	80×168	40×84	48×48	24×24
% Dice	78.12 ± 15.01	77.49 ± 14.67	74.95 ± 14.15	72.58 ± 14.67
% Dice*	70.08 ± 21.92	68.03 ± 21.57	64.54 ± 20.75	61.19 ± 21.63
% SSIM	89.91 ± 4.01	89.03 ± 3.98	87.98 ± 3.94	87.02 ± 3.95
MSE (m)	1.45 ± 1.11	1.81 ± 1.37	2.22 ± 1.61	2.64 ± 1.84
$\% \lvert J_\phi \rvert \leq 0$	0.12 ± 0.23	0.03 ± 0.15	0.06 ± 0.23	0.02 ± 0.15
Time [s]	0.0390	0.0301	0.0277	0.0289
Parameters	301,716	301,716	301,716	301,716
Mult-Adds (M)	256.14	65.50	43.91	10.98
Memory [MB]	36.70	10.54	7.66	3.15

4.1.4 Comparison with VoxelMorph

In this test, *Fourier-Net*, *Fourier-Net+* and *4xFourier-Net+* are compared to the famous *VoxelMorph* [**Voxelmorph**]. This unsupervised network brought deep learning based registration approaches into the mainstream and computes a dense displacement field in contrast to the band-limited displacement of the other networks. From the previous experiments a FT crop size of 48×48 and channel size of 16 was chosen to strike a good balance between performance and memory efficiency. The experiment was only done on the fully sampled test data and all networks were trained for 6 epochs. The results can be seen in Table 4.4.
VoxelMorph performs worst in terms of Dice score with the background label, however it does outperform *Fourier-Net+* in Dice without the background label. The best registration performance in terms of Dice (both with and without the background label)

has *Fourier-Net* followed by *4xFourier-Net+* as expected. *VoxelMorph*, however, does perform best in terms of SSIM and MSE followed by *Fourier-Net*, so it seems that the dense network aligns the overall image well, but struggles to compensate the strong changes in the cardiac region between frames. It is conspicuous, however, that the percentage of non-positive Jabobian determinants for *VoxelMorph* is quite high with almost 90% while the other models are all under one percent. This is perhaps caused by the dense displacement which causes a better overall alignment (as indicated by the good SSIM and MSE values) at the cost of more folding occurring. In terms of time *4xFourier-Net+* is slowest followed by *VoxelMorph*, *Fourier-Net* and *Fourier-Net+*, however all pretty fast (less than 30 ms). The model with the least amount of parameters and Mult-Adds is *VoxelMorph*, however perhaps due to the dense displacment it needs quite a lot of memory with almost 40 MB. *Fourier-Net* is the largest model in terms of model paramters, Mult-Adds and total memory (90 MB). *Fourier-Net+* and *4xFourier-Net+* both have a higher number of paramters and Mult-Adds, but a lower total amount of memory needed. Overall, *4xFourier-Net+* is the only model that is truly more efficient and has better performance in terms of Dice than *VoxelMorph*, as *Fourier-Net+* is more efficient, but not necessarily better in terms of Dice score, while *Fourier-Net* is far superior in registration performance (as measured by the Dice score), but also less efficient in terms of memory consumption.

Table 4.4: Comparison of *Fourier-Net*, *Fourier-Net+*, *4xFourier-Net+* and *VoxelMorph* with similarity metrics and memory consumption on the fully sampled *ACDC* test data.

	Fourier-Net	Fourier-Net+	4xFourier-Net+	VoxelMorph
% Dice	78.31 ± 13.96	76.88 ± 13.86	77.90 ± 13.92	75.84 ± 13.46
% Dice*	71.17 ± 18.99	67.86 ± 19.06	69.21 ± 19.02	68.25 ± 18.36
% SSIM	91.53 ± 3.49	88.67 ± 3.88	88.89 ± 4.05	93.53 ± 3.30
MSE (m)	0.09 ± 0.07	0.20 ± 0.15	0.19 ± 0.14	0.06 ± 0.04
$\% \lvert J_\phi \rvert \leq 0$	0.44 ± 0.45	0.04 ± 0.14	0.01 ± 0.04	49.98 ± 0.73
Time [s]	0.0099	0.0071	0.0244	0.0145
Parameters	1,735,447	300,477	1,201,908	84,322
Mult-Adds (G)	2.35	0.04289	0.17157	0.00157
Memory [MB]	90.08	4.64	17.23	39.04

4.1.5 Dense Displacement on Accelerated Data

The difference between a dense displacement field and a band-limited one was already explored in section 4.1.1. However, the results in Table 4.1 only include fully sampled data, not accelerated data. These new tests again included *Fourier-Net*, *Fourier-Net+* and *4xFourier-Net+*, but are extended to subsampled data. Results for $R = 4$ can be seen in Table 4.5, for $R = 8$ in Table 4.6 and for $R = 10$ in Table 4.7.

For $R = 4$, the dense version of *Fourier-Net*, *Fourier-Net+* and *4xFourier-Net+* performed better than the band-limited version in terms of Dice though the differences are less than a percent with and less than two percent without the background label. The same is true for the SSIM and MSE values with the dense versions being better. The percentage of non-positive Jabobian determinants is higher for the band-limited *Fourier-Net*, while band-limited *Fourier-Net+* and *4xFourier-Net+* show almost no folding. Band-limited *Fourier-Net* and *Fourier-Net+* are faster, likely due to the smaller network size as the encoder lack some layers compared to the dense versions, however, band-limited *4xFourier-Net+* is slightly slower than its dense counterpart.
For $R = 8$, all band-limited networks perform worse in terms of Dice, SSIM and MSE. The percentage of non-positive Jabobian determinants is again higher for the band-limited *Fourier-Net* compared to the dense version, while the band-limited *Fourier-Net+* and *4xFourier-Net+* show almost no folding. The band-limited networks are again faster with the exception of *4xFourier-Net+* which is a bit slower compared to the dense version.
For $R = 10$, the dense networks again perform better in terms of Dice and SSIM compared to the band-limited versions. For the MSE band-limited *Fourier-Net* is just as good as its dense counterpart, while band-limited *Fourier-Net+* and *4xFourier-Net+* are again slightly worse compared to the dense versions. The percentage of non-positive Jabobian determinants for the band-limited *Fourier-Net* is equal to the percentage of the dense version, while band-limited *Fourier-Net+* and *4xFourier-Net+* again show almost no folding. Band-limited *Fourier-Net* and *4xFourier-Net+* are faster in terms of inference time, while the band-limited version of *Fourier-Net+* is about as fast as the dense version.

Table 4.5: Results for *Fourier-Net*, *Fourier-Net+* and *4xFourier-Net+* with both dense and band-limited displacement fields on the $R = 4$ *ACDC* test data.

Metrics	Dense Displacement		
	Fourier-Net	Fourier-Net+	4xFourier-Net+
% Dice	77.37 ± 13.92	76.90 ± 14.12	77.35 ± 14.04
% Dice*	68.98 ± 19.10	68.39 ± 19.30	68.76 ± 19.32
% SSIM	84.75 ± 7.01	79.79 ± 9.76	80.32 ± 9.37
MSE (m)	0.08 ± 0.05	0.15 ± 0.12	0.13 ± 0.10
$\% \lvert J_\phi \rvert \leq 0$	0.16 ± 0.19	0.11 ± 0.21	0.03 ± 0.07
Time [s]	0.0085	0.0083	0.0263
Metrics	Band-limited Displacement		
	Fourier-Net	Fourier-Net+	4xFourier-Net+
% Dice	76.55 ± 13.89	76.20 ± 13.57	77.23 ± 13.73
% Dice*	67.90 ± 19.44	66.30 ± 19.01	67.67 ± 18.97
% SSIM	82.93 ± 8.12	77.74 ± 10.69	77.96 ± 10.57
MSE (m)	0.10 ± 0.06	0.19 ± 0.14	0.18 ± 0.13
$\% \lvert J_\phi \rvert \leq 0$	0.35 ± 0.41	0.01 ± 0.05	0.00 ± 0.02
Time [s]	0.0069	0.0065	0.0312

Table 4.6: Results for *Fourier-Net*, *Fourier-Net+* and *4xFourier-Net+* with both dense and band-limited displacement fields on the $R = 8$ *ACDC* test data.

Metrics	Dense Displacement		
	Fourier-Net	Fourier-Net+	4xFourier-Net+
% Dice	77.48 ± 14.00	77.30 ± 13.72	77.80 ± 14.02
% Dice*	68.86 ± 19.34	68.39 ± 19.15	69.18 ± 19.37
% SSIM	90.08 ± 3.05	87.60 ± 3.78	87.91 ± 3.64
MSE (m)	0.05 ± 0.04	0.10 ± 0.10	0.09 ± 0.09
$\% \lvert J_\phi \rvert \leq 0$	0.11 ± 0.15	0.06 ± 0.13	0.02 ± 0.06
Time [s]	0.0115	0.0108	0.0385
Metrics	Band-limited Displacement		
	Fourier-Net	Fourier-Net+	4xFourier-Net+
% Dice	75.52 ± 13.69	75.76 ± 13.43	76.56 ± 13.47
% Dice*	66.86 ± 18.97	65.45 ± 18.63	66.81 ± 18.77
% SSIM	89.51 ± 3.34	86.30 ± 4.18	86.29 ± 4.23
MSE (m)	0.06 ± 0.04	0.13 ± 0.11	0.13 ± 0.11
$\% \lvert J_\phi \rvert \leq 0$	0.27 ± 0.29	0.03 ± 0.11	0.00 ± 0.02
Time [s]	0.0090	0.0077	0.0466

Table 4.7: Results for *Fourier-Net*, *Fourier-Net+* and *4xFourier-Net+* with both dense and band-limited displacement fields on the $R = 10$ *ACDC* test data.

Metrics	Dense Displacement		
	Fourier-Net	Fourier-Net+	4xFourier-Net+
% Dice	77.39 ± 13.89	77.31 ± 13.82	77.43 ± 13.99
% Dice*	68.86 ± 19.27	68.69 ± 19.07	68.90 ± 19.43
% SSIM	92.92 ± 2.68	91.09 ± 3.42	91.33 ± 3.29
MSE (m)	0.05 ± 0.04	0.09 ± 0.09	0.08 ± 0.09
$\% \lvert J_\phi \rvert \leq 0$	0.11 ± 0.15	0.06 ± 0.11	0.04 ± 0.11
Time [s]	0.0280	0.0095	0.0183
Metrics	Band-limited Displacement		
	Fourier-Net	Fourier-Net+	4xFourier-Net+
% Dice	76.87 ± 13.97	76.10 ± 13.83	76.98 ± 13.57
% Dice*	68.24 ± 19.27	66.27 ± 19.20	67.40 ± 18.90
% SSIM	92.77 ± 2.74	90.32 ± 3.29	90.42 ± 3.33
MSE (m)	0.05 ± 0.04	0.12 ± 0.12	0.11 ± 0.11
$\% \lvert J_\phi \rvert \leq 0$	0.11 ± 0.15	0.01 ± 0.05	0.00 ± 0.03
Time [s]	0.0081	0.0096	0.0279

4.1.6 Comparison on Subsampled Data

In section 4.1.4, *Fourier-Net*, *Fourier-Net+* and *4xFourier-Net+* were already compared to *VoxelMorph*, however, these comparisons were only done on fully sampled, not accelerated data. To further add to the comparison, *NiftyReg* was again used as a traditional registration algorithm. The unaligned test image pairs were treated as a baseline for the lower bound of registration performance. The results can be seen in Table 4.8 for fully sampled ($R = 0$) and subsampled ($R = 4$, $R = 8$, $R = 10$) *ACDC* test data. Values which are worse than the baseline are marked in red, while the best results for each acceleration level and metric is highlighted with blue. All times are computed on CPU due to *NiftyReg* only working on the CPU as the GPU capable versions were deprecated. Additionally, Figure 4.1 shows the performance of the methods for each segmentation label (excluding the background) while Figure 4.2 shows example images and segmentations warped by the different methods for a visual comparison.
For $R = 0$, *Fourier-Net* performs best in terms of Dice (both with and without the background label), closely followed by *4xFourier-Net+* while *NiftyReg* performs worse than the baseline. *VoxelMorph* performs best in terms of SSIM, followed by *Fourier-Net* and *NiftyReg*, as well as MSE where *NiftyReg* again performs worse than the baseline on the fully sampled data. *NiftyReg* and *4xFourier-Net+* had the lowest percentage of non-positive Jabobian determinants while *VoxelMorph* had the highest value by far (almost 50% while the other method were all under 1%) indicating folding. *Fourier-Net* is the fastest method with under $0.1s$ per image pair, while *NiftyReg* takes over $100s$ making it the slowest by a large margin (all other methods were under $1s$).
For $R = 4$, the registration performance decreases for all methods with *NiftyReg* again performing worse than the baseline, while *4xFourier-Net+* performed best in terms of Dice with the background label, while *Fourier-Net* performed best in Dice without the background. *VoxelMorph* performs best in terms of SSIM and MSE, followed by *Fourier-Net* and *NiftyReg*, with none of methods performing worse than the baseline. *4xFourier-Net+* again has the lowest percentage of non-positive Jabobian determinants closely followed by *Fourier-Net+*, while *VoxelMorph*, similar to the fully sampled data, has the highest value. In terms of time *NiftyReg* is again the worst method with about $80s$, while all other methods need around $0.1s$ with *Fourier-Net+* being the fastest.
For $R = 8$, *4xFourier-Net+* again performs best for Dice with the background label, while *Fourier-Net* performs best in term of Dice without the background label. *NiftyReg* performs worse than the baseline in terms of Dice (both with and without the background label). *NiftyReg* again performs best for SSIM and MSE followed by *Fourier-Net* and *NiftyReg*. *4xFourier-Net+* has the lowest percentage of non-positive Jabobian determinants followed by *Fourier-Net+*, while *VoxelMorph* again has the worst value. *NiftyReg* is the worst method in terms of time with over $80s$, similar to $R = 4$, while *Fourier-Net+* again is the fastest, however, being slightly slower than before.
For $R = 10$, *4xFourier-Net+* performs best in terms of Dice with the background label, while *Fourier-Net* performs best in terms of Dice without the background. *NiftyReg* again performs worse than the baseline for Dice (both with and without background

label). *VoxelMorph* is again the best method in terms of SSIM and MSE, followed by *Fourier-Net* and *NiftyReg* with no method being worse than the baseline. *4xFourier-Net+* has again the lowest percentage of non-positive Jabobian determinants closely followed by *Fourier-Net+*, while *VoxelMorph* has the worst value like before. *Fourier-Net+* is again the fastest method with under $0.01s$, while *NiftyReg* is again the slowest despite its best time yet with about $47s$.

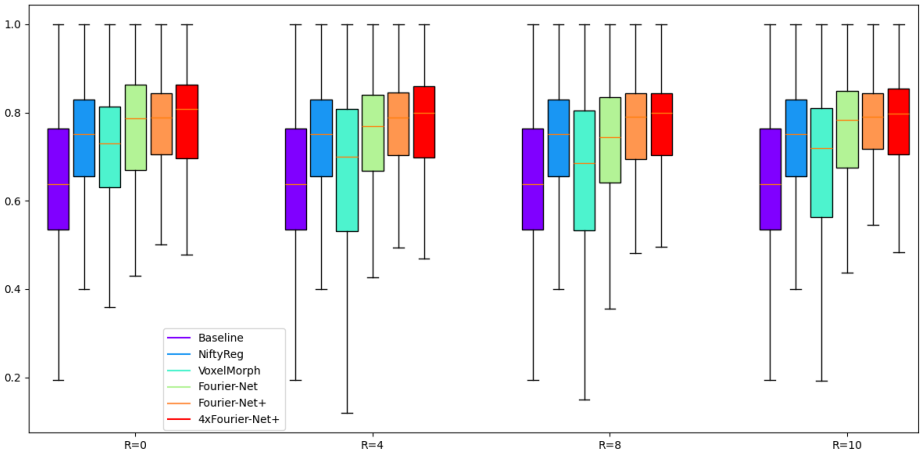
Table 4.8: Test results for *NiftiReg* (NR), *VoxelMorph* (VM), *Fourier-Net* (F-Net), *Fourier-Net+* (F-Net+) and *4xFourier-Net+* (4xF-Net+) on the *ACDC* test data from fully sampled ($R = 0$) to $R = 10$ with an unaligned baseline for comparison. The best results for each metric and subsampling are highlighted in blue, while values worse than the unaligned baseline are marked with red.

	Method	% DICE	% DICE*	% SSIM	MSE (m)	$\%\,\lvert J_\phi\rvert \leq 0$	Time [s]
$R = 0$	Baseline	70.85 ± 18.27	60.35 ± 25.24	86.39 ± 4.08	0.33 ± 0.23	-	-
	NR	70.74 ± 13.77	57.56 ± 18.86	91.26 ± 3.18	1.94 ± 1.61	0.00 ± 0.02	122.52
	VM	75.84 ± 13.46	68.25 ± 18.36	93.53 ± 3.30	0.06 ± 0.04	49.98 ± 0.73	0.1845
	F-Net	78.31 ± 13.96	71.17 ± 18.99	91.53 ± 3.49	0.09 ± 0.07	0.24 ± 0.25	0.1918
	F-Net+	76.88 ± 13.86	67.86 ± 19.06	88.67 ± 3.88	0.20 ± 0.15	0.02 ± 0.08	0.0893
	4xF-Net+	77.90 ± 13.92	69.21 ± 19.02	88.89 ± 4.05	0.19 ± 0.14	0.00 ± 0.02	0.3262
$R = 4$	Baseline	70.85 ± 18.27	60.35 ± 25.24	76.80 ± 11.02	0.28 ± 0.19	-	-
	NR	69.89 ± 13.73	56.47 ± 17.86	86.03 ± 6.41	0.15 ± 0.14	0.12 ± 0.15	80.08
	VM	71.78 ± 14.57	63.15 ± 18.98	90.73 ± 4.72	0.04 ± 0.03	49.36 ± 1.20	0.1264
	F-Net	76.55 ± 13.89	67.90 ± 19.44	82.93 ± 8.12	0.10 ± 0.06	0.19 ± 0.23	0.1006
	F-Net+	76.20 ± 13.57	66.30 ± 19.01	77.74 ± 10.69	0.19 ± 0.14	0.01 ± 0.03	0.0294
	4xF-Net+	77.23 ± 13.73	67.67 ± 18.97	77.96 ± 10.57	0.18 ± 0.13	0.00 ± 0.02	0.1131
$R = 8$	Baseline	70.85 ± 18.27	60.35 ± 25.24	85.35 ± 4.43	0.22 ± 0.17	-	-
	NR	70.04 ± 13.42	56.37 ± 17.63	91.07 ± 2.80	0.12 ± 0.13	0.08 ± 0.10	88.36
	VM	71.51 ± 14.22	62.17 ± 18.80	94.17 ± 2.80	0.03 ± 0.02	49.18 ± 1.36	0.1973
	F-Net	75.52 ± 13.69	66.86 ± 18.97	89.51 ± 3.34	0.06 ± 0.04	0.27 ± 0.29	0.2404
	F-Net+	75.76 ± 13.43	65.45 ± 18.63	86.30 ± 4.18	0.13 ± 0.11	0.03 ± 0.11	0.1482
	4xF-Net+	76.56 ± 13.47	66.81 ± 18.77	86.29 ± 4.23	0.13 ± 0.11	0.00 ± 0.02	0.5283
$R = 10$	Baseline	70.85 ± 18.27	60.35 ± 25.24	89.17 ± 3.58	0.20 ± 0.17	-	-
	NR	70.40 ± 13.34	56.61 ± 17.71	93.47 ± 2.37	0.11 ± 0.13	0.06 ± 0.08	47.44
	VM	71.89 ± 13.94	62.22 ± 18.58	95.85 ± 2.58	0.02 ± 0.02	48.56 ± 1.75	0.0577
	F-Net	76.87 ± 13.97	68.24 ± 19.27	92.77 ± 2.74	0.05 ± 0.04	0.11 ± 0.15	0.0296
	F-Net+	76.10 ± 13.83	66.27 ± 19.20	90.32 ± 3.29	0.12 ± 0.12	0.01 ± 0.05	0.0059
	4xF-Net+	76.98 ± 13.57	67.40 ± 18.90	90.42 ± 3.33	0.11 ± 0.11	0.00 ± 0.03	0.0275

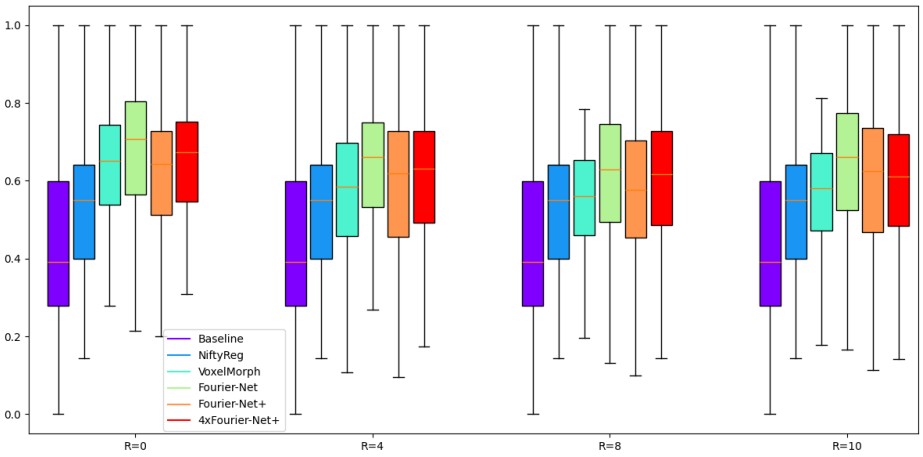
Cardiac Labels

To further evaluate the difference between the methods performance, one can look at the Dice scores of the three individual cardiac labels. The results can be seen as boxplots in Figure 4.1a for the left ventricle, Figure 4.1b for the myocardium and Figure 4.1c for the right ventricle. The Dice scores for each label vary widely but the performance of all methods is best for the right ventricle and worst for the myocardium. This seems to be an underlying principle of the data as the same behavior can be seen in the unaligned baseline.
NiftyReg performs well for the left ventricle, even surpassing *VoxelMorph*, while it is the worst method for the myocardium (being only slightly better than the baseline)

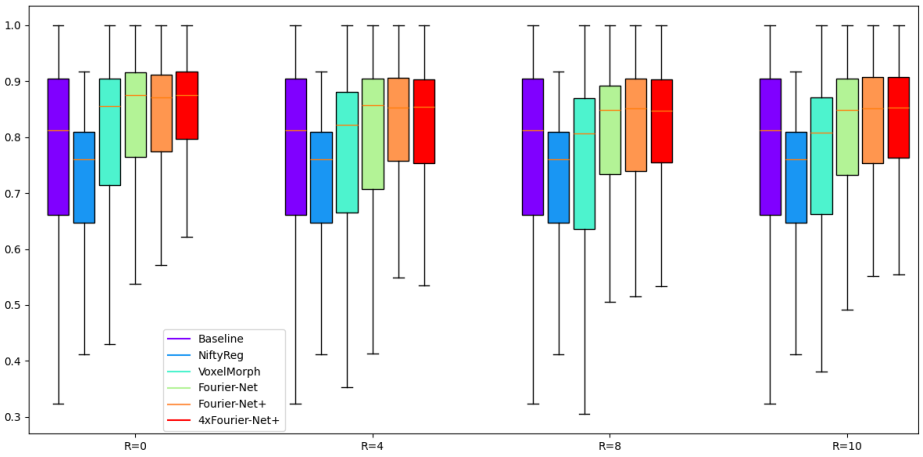
and the right ventricle (far worse than the baseline). Overall, *NiftyReg* is not heavily affected by the artifacts present in the subsampled data, but is also not able to perform better than the other methods consistently.
VoxelMorph performs best on the right ventricle compared to the other labels being close the *Fourier-Net* variant for $R = 0$, but falling off for the subsampled data, even performing worse than the baseline. It is better than the baseline, *NiftyReg* and even *Fourier-Net+* on $R = 0$ for the myocardium, but again loses performance for the subsampled data. This trend also continues for the left ventricle were *VoxelMorph* performs about as well as the baseline, but better than *NiftyReg*, however ends up falling behind the baseline. Overall, *VoxelMorph* is able to outperform the baseline, *NiftyReg* and even *Fourier-Net+* on one occasion, but is severely hampered by the artifacts caused by the subsampling of the k-space thus limiting its usability for accelerated MRI applications.
Fourier-Net is consistently the best method for the challenging myocardium, however it is only about as good as *Fourier-Net+* and *4xFourier-Net+* for the left and right ventricle (even performing slightly worse than the baseline for $R = 8$ on the latter). While *Fourier-Net*, overall, is effected by the subsampling the effect is far less severe compared to *VoxelMorph* and *Fourier-Net* reaches a far better performance, especially for the challenging myocardium.
Fourier-Net+ performs well for the left ventricle being barely affected by the subsampling, although it is surpassed by *Fourier-Net* on $R = 0$ and by *4xFourier-Net+* for $R = 4$ and $R = 10$. For the challenging myocardium *Fourier-Net+* again performs well, but is surpassed by *Fourier-Net* for all acceleration factors and only manages to beat *4xFourier-Net+* for $R = 10$. For the right ventricle *Fourier-Net+* is among the best methods, even performing better than *Fourier-Net* and *4xFourier-Net+* for $R = 8$.
4xFourier-Net+ consistently has the best median value for the left ventricle and is close to being the best model for the right ventricle. Only on the myocardium it is consistently outperformed by *Fourier-Net*.



(a) Boxplot of the Dice scores for the left ventricle cavity.



(b) Boxplot of the Dice scores for the myocardium.



(c) Boxplot of the Dice scores for right ventricle cavity.

Figure 4.1: Boxplots of Dice scores (split by label excluding the background) for all models on fully sampled ($R = 0$) and subsampled ($R = 4$, $R = 8$, $R = 10$) *ACDC* test data.

Visual Examination

In order to explain the differences observed previously a visual examination can be useful. There one can look at the warped images and segmentations as well as the generated displacement fields visualized in Figure 4.2. For all acceleration factors, an example moving and fixed image can be compared to the warped images by the different methods. For further analysis, the corresponding segmentations (with Dice scores) and displacement fields for the methods are given.
For $R = 0$, it is apparent that *Fourier-Net+* and *4xFourier-Net+* have very localized displacements centered on the cardiac region leading to a very smooth warped segmentation compared to *Fourier-Net* and *VoxelMorph* were the cardiac labels mix in some regions. Thus those networks have learned a less localized transformation, which can also be seen in the displacements. However, despite this *Fourier-Net* still achieves the best Dice score with about 16% increase compared to the baseline. While *NiftyReg* does produce a very smooth warped segmentation the actual displacement is very small and not localized on the cardiac region leading to the worst Dice score of all methods.
For $R = 4$, the image artifacts due to the subsampling are very apparent. The segmentations for the moving and fixed image obviously remain the same. *Fourier-Net+* and *4xFourier-Net+* again have very smooth segmentations, however the displacements reflect the difficulties of adapting to subsampled data in becoming slightly less local. Surprisingly, *NiftyReg* actually has a more localized displacement compared to the fully sampled data, however the segmentation looks less smooth and the Dice score is slightly lower. Somewhat similar, *VoxelMorph* also has a more localized displacement leading to a smoother segmentation and a higher Dice score. The results of *Fourier-Net* look overall very similar to the fully sampled case, perhaps with a bit more localized displacement leading to a smoother segmentation, although this does not lead to a better Dice Score.
For $R = 8$, the displacements for all methods become more global due to the strong presence of image artifacts, however the Dice scores are better than before for all methods. *NiftyReg* even outperforms *VoxelMorph* for the first time. This trend does not continue for $R = 10$, were *NiftyReg* is again far behind all other methods in terms of Dice with a very much global displacement. *VoxelMorph* seems to compensate more for the rippling artifacts present in the fixed image than for the change in the cardiac region as seen in the displacement. *Fourier-Net*, *Fourier-Net+* and *4xFourier-Net+* do not share this behavior although their displacements also become more global and less focused on the cardiac region. The latter network performs best for the first time managing an increase in Dice of about 16% compared to the baseline for this specific case despite the heavy subsampling showing the robustness of the network.

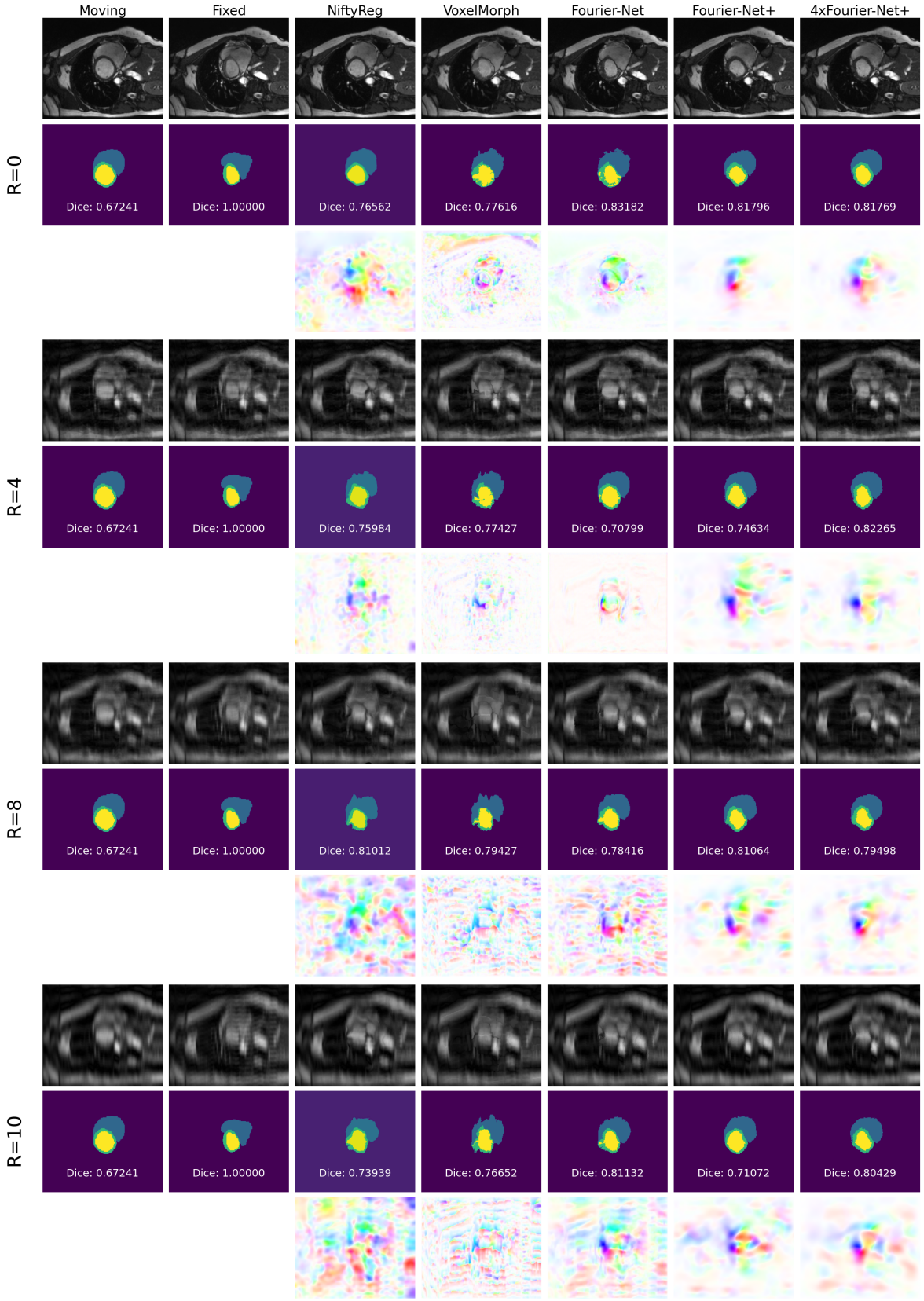


Figure 4.2: Examples of warped images, segmentations and flow fields for *NiftiReg*, *VoxelMorph*, *Fourier-Net*, *Fourier-Net+* and *4xFourier-Net+* together with the original image pair from the fully sampled ($R = 0$) and sub-sampled ($R = 4$, $R = 8$, $R = 10$) *ACDC* test data.

4.2 Integration into a Motion-Compensated Reconstruction Pipeline

Stuff.

4.2.1 Domain Translation

Trained *Fourier-Net*, *Fourier-Net+* and *4xFourier-Net+* again on the accelerated *CMRxRecon* data (without crops).
Compared *Fourier-Net*, *Fourier-Net+* and *4xFourier-Net+* trained for 6 epochs on *ACDC* to *Fourier-Net+* trained for 14 epochs on *CMRxRecon* to evaluate the domain translation capabilities of *Fourier-Net+*. All other network parameters are the same.

Table 4.9: Results for *Fourier-Net*, *Fourier-Net+* and *4xFourier-Net+* trained on the $R = 4$ *ACDC* and *CMRxRecon* data and tested on the $R = 4$ *CMRxRecon* test data.

Metrics	Trained on *ACDC*		
	Fourier-Net	Fourier-Net+	4xFourier-Net+
% SSIM	98.21 ± 1.07	97.88 ± 1.04	97.88 ± 1.04
MSE (m)	0.01 ± 0.01	0.01 ± 0.01	0.01 ± 0.01
$\% \lvert J_\phi \rvert \leq 0$	0.03 ± 0.09	0.00 ± 0.00	0.00 ± 0.00
Time [s]	0.0038	0.0037	0.0185
Metrics	Trained on *CMRxRecon*		
	Fourier-Net	Fourier-Net+	4xFourier-Net+
% SSIM	97.99 ± 1.04	97.88 ± 1.06	97.88 ± 1.07
MSE (m)	0.01 ± 0.01	0.01 ± 0.01	0.01 ± 0.01
$\% \lvert J_\phi \rvert \leq 0$	0.00 ± 0.01	0.00 ± 0.00	0.00 ± 0.00
Time [s]	0.0029	0.0043	0.0120

4.2.2 Reconstruction Pipeline

After ensuring that the previous results were translatable to the new data, the reconstruction pipeline had to be evaluated. As the *CMRxRecon* dataset does not include segmentations for Dice calculation, image similarity metrics need to be utilized. For this, SSIM and MSE were used again, however, the PSNR (see section 2.3.3.2) and HaarPSI [**HaarPSI**] were also added. After ensuring that the reconstruction pipeline worked, the input images were further corrupted as the cardiac motion present was not deemed severe enough. To simulate motion or mis-triggering a similar strategy to [**Oksuz2020**] was used, swapping $z = \{16, 32\}$ k-space lines between the frames. Results for the motion-compensated reconstruction using *VoxelMorph*, *Fourier-Net*, *Fourier-Net+* and *4xFourier-Net+* is shown in Table 4.10.

Table 4.10: Reconstruction results *VoxelMorph*, *Fourier-Net*, *Fourier-Net+* and *4xFourier-Net+* on the *CMRxRecon* test data for $R = 4$, $R = 8$ and $R = 10$ as well as an baseline without motino-correction. The best results for each metric and subsampling are highlighted in blue, while values worse than the unaligned baseline are marked with red.

	Methods	Motion-correction for motion $z = 16$			
		% HaarPSI	PSNR [dB]	% SSIM	MSE (m)
$R = 4$	Baseline	56.884 ± 8.129	28.703 ± 2.388	78.079 ± 5.887	0.160 ± 0.125
	VoxelMorph	56.909 ± 8.118	28.709 ± 2.404	78.090 ± 5.903	0.160 ± 0.126
	Fourier-Net	56.918 ± 8.160	28.706 ± 2.401	78.123 ± 5.893	0.160 ± 0.126
	Fourier-Net+	56.854 ± 8.158	28.699 ± 2.399	78.071 ± 5.951	0.160 ± 0.126
	4xFourier-Net+	56.903 ± 8.109	28.702 ± 2.380	78.085 ± 5.877	0.160 ± 0.125
$R = 8$	Baseline	53.318 ± 7.641	28.104 ± 2.383	77.311 ± 5.934	0.183 ± 0.135
	VoxelMorph	53.342 ± 7.687	28.110 ± 2.405	77.341 ± 5.900	0.183 ± 0.139
	Fourier-Net	53.394 ± 7.681	28.128 ± 2.398	77.416 ± 5.927	0.183 ± 0.138
	Fourier-Net+	53.388 ± 7.664	28.118 ± 2.404	77.398 ± 5.906	0.183 ± 0.138
	4xFourier-Net+	53.376 ± 7.688	28.133 ± 2.398	77.402 ± 5.936	0.182 ± 0.138
$R = 10$	Baseline	52.211 ± 7.388	27.906 ± 2.364	77.192 ± 5.843	0.191 ± 0.140
	VoxelMorph	52.211 ± 7.393	27.907 ± 2.368	77.194 ± 5.863	0.191 ± 0.140
	Fourier-Net	52.192 ± 7.361	27.895 ± 2.362	77.160 ± 5.815	0.191 ± 0.139
	Fourier-Net+	52.240 ± 7.379	27.924 ± 2.357	77.244 ± 5.866	0.189 ± 0.139
	4xFourier-Net+	52.272 ± 7.330	27.931 ± 2.349	77.262 ± 5.816	0.189 ± 0.137
		Motion-correction for motion $z = 32$			
$R = 4$	Baseline	52.754 ± 7.725	27.534 ± 2.175	74.379 ± 5.869	0.202 ± 0.129
	VoxelMorph	52.747 ± 7.658	27.512 ± 2.188	74.349 ± 5.900	0.203 ± 0.129
	Fourier-Net	52.802 ± 7.708	27.536 ± 2.197	74.421 ± 5.937	0.202 ± 0.128
	Fourier-Net+	52.725 ± 7.724	27.522 ± 2.198	74.366 ± 5.892	0.203 ± 0.129
	4xFourier-Net+	52.711 ± 7.718	27.500 ± 2.205	74.353 ± 5.986	0.204 ± 0.132
$R = 8$	Baseline	50.071 ± 7.259	27.153 ± 2.199	74.033 ± 5.844	0.221 ± 0.138
	VoxelMorph	50.091 ± 7.301	27.164 ± 2.198	74.060 ± 5.890	0.220 ± 0.138
	Fourier-Net	50.152 ± 7.325	27.178 ± 2.202	74.094 ± 5.826	0.220 ± 0.140
	Fourier-Net+	50.159 ± 7.362	27.196 ± 2.221	74.119 ± 5.853	0.220 ± 0.141
	4xFourier-Net+	50.128 ± 7.276	27.165 ± 2.201	74.054 ± 5.842	0.220 ± 0.137
$R = 10$	Baseline	49.311 ± 7.049	27.037 ± 2.182	74.029 ± 5.782	0.227 ± 0.143
	VoxelMorph	49.241 ± 7.047	27.026 ± 2.179	73.976 ± 5.795	0.227 ± 0.142
	Fourier-Net	49.255 ± 7.011	27.024 ± 2.168	73.952 ± 5.793	0.227 ± 0.141
	Fourier-Net+	49.296 ± 7.001	27.045 ± 2.172	74.028 ± 5.784	0.226 ± 0.140
	4xFourier-Net+	49.287 ± 7.066	27.032 ± 2.176	74.022 ± 5.820	0.227 ± 0.143

Chapter 5

Discussion

Possible implications will be discussed to determine the effectiveness of this new method.

5.1 Parameter Tests on the ACDC Dataset

First, the results of the parameter tests on the *ACDC* dataset need to be analyzed.

5.1.1 Fourier-Net versus Fourier-Net+

5.1.2 Starting Channel Size

5.1.3 Fourier-Transform Crop Size

5.1.4 Comparison with VoxelMorph

5.1.5 Dense Displacements

The results for the band-limited networks are consistently worse (or in the best case just as good) compared to their dense versions across all acceleration factors for Dice, SSIM and MSE. The band-limiting of the displacement helps with avoiding folding and in most cases decrease inference time, however at the expense of registration performance. Our hypothesis that the advantage of the dense displacement compared to the band-limited displacement disappears for accelerated data does not seem to hold as the dense networks outperform the band-limited ones, though the difference is small in some cases. In the end, the larger dense networks perform better, but also require more memory. Thus, similar to the previous experiments, a balance between performance and memory efficiency has to be found.

5.1.6 Comparison on Subsampled Data

To summarize the results, *Fourier-Net+* is the fastest method for all acceleration levels while *NiftyReg* is far behind all other methods in execution time as it is not machine learning based. *VoxelMorph* performs best in terms of SSIM and MSE for all acceleration factors, however it also has the highest percentage of non-positive Jabobian determinants by a large margin indicating potential folding artifacts. *4xFourier-Net+*

performs best in terms of Dice with the background label except for $R = 0$ while consistently having the lowest percentage of non-positive Jabobian determinants with a mean value under 0.004% for all acceleration factors. *Fourier-Net+* performs well in terms of Dice (with and without the background label), however it does not manage to surpass the other *Fourier-Net* variants despite having the second lowest percentage of non-positive Jabobian determinants across the acceleration factors. *Fourier-Net* performs best in terms of Dice without the background label for all acceleration level as well as in Dice with the background for $R = 0$. *NiftyReg* consistently performs worse than the baseline in terms of Dice (with and without the background) across all acceleration factors while having decent SSIM and MSE values. The performance, in general, decreased for all methods for higher acceleration factors as seen in the Dice metric, however, perhaps counter-intuitively, the SSIM is highest and the MSE the lowest for the $R = 10$ baseline with only $R = 4$ breaking the trend. This shows once again that image similarity metrics are not completely objective when it comes to evaluating registration performance across different acceleration factors.

5.2 Integration into a Motion-Compensated Reconstruction Pipeline

5.2.1 Domain Translation

5.2.2 Reconstruction Pipeline

5.3 Limitations and Outlook

Chapter 6

Conclusion

After finding optimal model parameters for *Fourier-Net*, *Fourier-Net+* and *4xFourier-Net+*, their registration performance was compared to *NiftyReg* and *VoxelMorph* on the *ACDC* dataset. As expected, the three networks performed very well for the different reduction factor present in the dataset with *Fourier-Net* generally performing best, though *4xFourier-Net+* had similar results while being more efficient.
Trying to further optimize the performance, different loss functions e.g. utilizing k-space information or the contrastive InfoNCE loss were tested, though non of these were able to provide a better performance.
Lastly, the networks were used in an motion-compensated reconstruction pipeline on *CMRxRecon* data that was further motion-corrupted.