Magnetic Resonance Imaging motion correction in k-space

Detecting, estimating and correcting the bulk motion artifacts in k-space data

by

Rajesh Rajwade

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Monday November 25, 2019 at 01:00 PM.

Student number:	4737989	
Project duration:	February 1, 2019 – November 25, 2019	
Thesis committee:	Dr. A. Sbrizzi,	UMC, Utrecht
	Dr. ir. R. F. Remis,	TU Delft, Supervisor
	Dr. D. M. J. Tax ,	TU Delft

TU Delft

UMC Utrecht

Acknowledgments

I would like to thank my thesis advisor Dr. Alessandro Sbrizzi of the Imaging Division, University Medical Center, Utrecht. Because of his continuous guidance and support, I could complete my research and report. I am thankful for his honest feedback which will help me improve myself not only as a student but also as a person. I would also like to thank my committee members Dr. Rob Remis and Dr. David Tax for their regular feedback and suggestions during bi-weekly meetings. I am also thankful to Dr. Ir. C.A.T. van den Berg for providing me the opportunity to work at University Medical Center, Utrecht.

I would like to thank my friends in Delft and Utrecht because of whom I could survive during the last two years, thousands of miles away from my home.

Most importantly, I would like to thank my parents and my brother for supporting me financially as well as emotionally throughout this journey.

Rajesh Rajwade
Delft, November 2019

Abstract

Magnetic Resonance Imaging is a widely used technique to obtain images of the interior of the human body for diagnosis and treatment. MRI machines capture the raw signal in spatial frequency domain i.e. k-space and the image is obtained via Fourier transform. The Cartesian acquisition is one of the most commonly used acquisition patterns in MRI and is most susceptible to the patient's motion. Due to long scanning times, the possibility of the patient's movement is higher which introduces bulk motion artifacts reducing the quality of the image. Motion artifacts can affect the diagnosis and the necessity of re-scanning can cause significant financial costs as well as delays in diagnostics. Current methods for correcting motion artifacts work in image domain which need completely sampled k-space for reconstruction and hence are not useful for real-time artifacts correction. In this thesis, machine learning methods that can detect, estimate and correct motion artifacts in k-space were investigated making it possible to correct artifacts in real-time without the necessity of reconstruction. For each of these methods, we analyze the performance and discuss the merits and demerits.

Contents

List of Figures

1

Introduction

Magnetic Resonance Imaging (MRI) is a non-invasive imaging method that gives clear, detailed images of soft-tissue structures. During an MRI scan, the patient must remain still in an enclosed machine, which may be a problem for pediatrics, stroke patients, and in the elderly. Due to long scanning times, artifacts can occur due to motion which causes significant financial costs due to repeated scans. The motion artifacts can affect the diagnosis if an image of acceptable diagnostic quality is not obtained. Although many solutions exist for the detection and correction of motion artifacts, those solutions are limited in their applicability. Techniques such as anatomical restraints are uncomfortable and anesthesia has a risk of ill effects on patient's health. In this project, the aim is to implement a technique that can detect and correct the motion artifacts in a brain MRI scan with the Cartesian acquisition by comparing two consecutively scanned spatial frequency domain lines (also known as k-space). This method is useful to detect the occurrence of motion in an MRI scan "on the go" and further measures (either correction or re-scan) can be taken depending on determined motion parameters.

1.1. MRI Acquisition

During an MRI scan, a patient is placed in a static magnetic field produced by an MR scanner magnet. Due to the application of this static magnetic field, all protons inside tissues, align parallel to the magnetic field. During image acquisition, Radio Frequency (RF) pulse is emitted from scanner tuned to Larmor frequency resonating with the rotation frequency of the intrinsic magnetization vector of hydrogen atom proton. The duration of this pulse is such that, it tilts the spin magnetization such that a transverse component can be detected. Current generated in the coil due to transverse magnetization by Faraday induction is called Nuclear Magnetic Resonance (NMR). The loss of coherence of the spin system attenuates the NMR signal with a time constant called transverse relaxation time (T2). Magnetization vector relaxes towards its equilibrium orientation parallel to the magnetic field with time constant called spin-lattice relaxation time (T1). Different tissues have different T1 and T2 relaxation times which give good contrast in MR images.

To identify the location of the signals coming from different parts of body, small magnetic field gradients are applied with different frequencies and/or phases. On one of the axis, this frequency is changed and on the other, the phase is changed using these gradients. The signal is encoded using different frequencies and phases to know the location of it's origin. This encoding is referred as frequency and phase encoding.

1.2. Raw data from MRI and k-space

During data acquisition of MR sequence, transversal components of magnetization in an imaging object after excitation are sampled from the receiver coil signal. As gradients are applied on phase and frequency encoding, this raw signal is already in a Fourier-like format suitable for filling the k-space matrix [9].
The relationship between signal acquired by MR scanner (raw signal) and object image is given by imaging equation,

$$\sigma(t) = \int_{\mathbb{R}^2} \rho(\vec{r}) \mathrm{e}^{-\gamma i \int_0^t \vec{G}(\tau)\cdot\vec{r}\,\mathrm{d}\tau} \mathrm{d}\vec{r}. \tag{1.1}$$

where, σ is acquired signal, ρ is image value at point $\vec{r} \in \mathbb{R}^2$ given by transverse magnetization state, γ is a physical constant and $\vec{G}.\vec{r}$ with $\vec{G} \equiv (G_x, G_y)$ is gradient field. By setting $\vec{k}(t) = \frac{\gamma}{2\pi}\int_0^t \vec{G}(\tau)\mathrm{d}\tau$, in equation (1.1), we get,

$$\sigma(k) = \int_{\mathbb{R}^2} \rho(\vec{r}) \mathrm{e}^{-i2\pi\vec{k}\cdot\vec{r}} \mathrm{d}\vec{r}. \tag{1.2}$$

To get the image (ρ) from the acquired signal σ, inverse Fourier transform is applied,

$$\rho(\vec{r}) = \int_{\mathbb{R}^2} \sigma(k) \mathrm{e}^{i2\pi\vec{k}\cdot\vec{r}} \mathrm{d}\vec{k}. \tag{1.3}$$

This relationship is similar to frequency-time relationship and hence, $\vec{k}$ can be considered as spatial frequency. In the field of MRI imaging, the frequency domain is called k-space [28]. In 2D imaging, each point in k-space indicates spatial frequency over the x-axis and y-axis and amplitude at each point indicates the contribution of that spatial frequency in an image. Data near the center of k-space corresponds to low spatial frequencies (i.e. general shapes and image contrast), whereas data from the periphery relates to high-spatial frequencies (i.e. edges, details) [8]. If the k-space is filled by data near the center, the image will have low resolution, but if k-space is filled by data farther away from the center, the image will have high resolution but poor contrast. Cartesian sampling, radial sampling, and spiral sampling are the three most commonly used sampling patterns to fill k-space. In clinical MRI, Cartesian sampling is most commonly used because of simple reconstruction by using Inverse Fast Fourier Transform (IFFT) and instrumental imperfections of early MRI systems [33]. In the Cartesian acquisition, data is filled in k-space line by line. Due to this, the Cartesian acquisition is susceptible to object motion. The data acquired during the MR imaging sequence is then taken to the image domain by using techniques like fast Fourier transform with computational complexity $O(N \log N)$.

1.3. Bulk head motion and its effects

The inconsistency between the various portions of the k-space data used for image reconstruction is the main cause of motion artifacts. The simple reconstruction technique like IFFT assumes that the object has remained stationary during the scanning time. If the object is not stationary during the scanning period, it introduces artifacts.

Though physiological processes (e.g. breathing, cardiac motion, blood pulsation, and tremors) introduce some motion artifacts, in case of brain MR imaging, sudden position changes, for example, due to swallowing, often lead to artefacts[21].This is also known as bulk patient motion and is one of the most common sources of artifacts in MRI brain imaging [2]. In bulk-head motion, there are two types namely translation and rotation also known as rigid motion. The effect of these motions in 2-D image space can be shown by the equation mentioned below.

1. Translation:
In (1.4), the effect of translation motion in k-space can be observed. The occurrence of translation motion in image domain causes change in phase in k-space (frequency) domain. Here, it can be observed that there is no change in the amplitude of the signal.

$$f(x - x_0, y - y_0) \xleftrightarrow{\text{DTFT}} F(u, v)e^{-j2\pi(ux_0 + vy_0)}. \tag{1.4}$$

Here, $f(x, y)$ is image data and $F(u, v)$ is its k-space (Fourier) equivalent. x_0, y_0 are translation motion parameters along x and y axis respectively and j is unit imaginary number.
2. Rotation:
In equations (1.5) and (1.6), rotation occurred in image space is equivalent to rotation in k-space. Equation (1.5) indicates the effect of rotation in polar co-ordinates and equation (1.6) indicates the effect of rotation in Cartesian co-ordinates.

$$f(r, \theta - \theta_0) \xleftrightarrow{\text{DTFT}} F(\omega, \phi - \theta_0), \tag{1.5}$$

$$f(x\cos(\theta_0) + y\sin(\theta_0), -x\sin(\theta_0) + y\cos(\theta_0)), \tag{1.6}$$

$$\text{DTFT} \updownarrow$$

$$F(u\cos(\theta_0) + v\sin(\theta_0), -u\sin(\theta_0) + v\cos(\theta_0).$$

Here, f(x,y) is image data and F(u,v) is its k-space (Fourier) equivalent. On the left side of the equations (1.5) and (1.6), is image data with rotation parameter θ_0 and on right side its k-space equivalent.

In figure 1.1, a 2D slice of brain MRI image can be seen. Ghosting effects can be observed in the image which is the effect of motion during MRI scan.

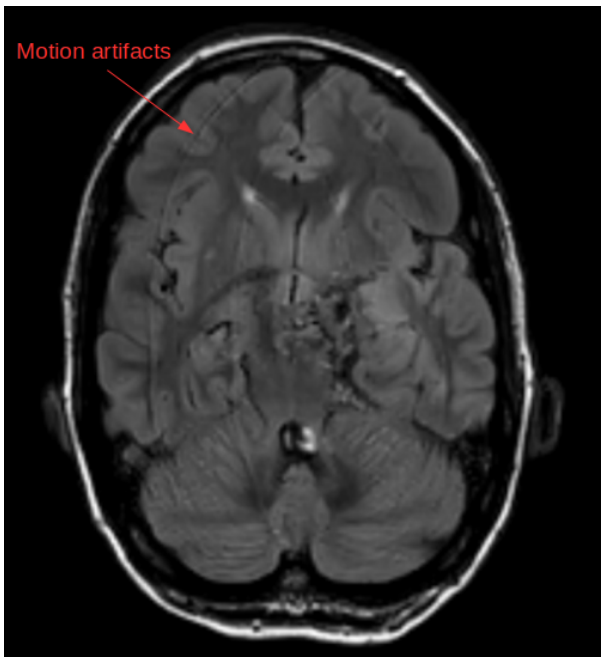


Figure 1.1: Motion affected brain MRI image
Source: University Medical Center, Utrecht

1.4. Objective and Research Question

This project aims to detect and correct motion artifacts in k-space. By estimating the motion parameters we aim to compensate for object motion in k-space which will save the necessity of performing reconstruction before correction. In this project, the aim is to estimate motion in between consecutive scanned rows of k-space with Cartesian acquisition ”on the go”. The detected motion parameters are then used to correct the motion artifacts and reconstruct the image.

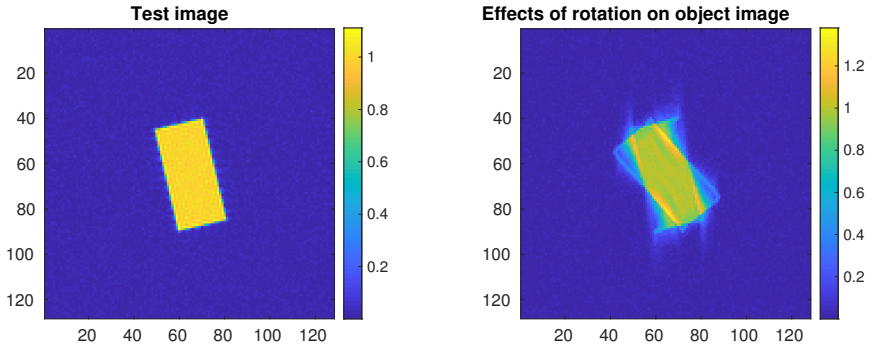
The main research question we are trying to answer is *”can we predict and correct bulk motion artifacts in k-space?”* The second question we are trying to address is *”which regression method works best for this prediction of motion parameters ?”*.

We implemented and compared four different methods for predicting motion parameters. Least squares regression was selected as a starting point as it is one of the most commonly used and simple methods. The least-squares method is a model driven method which assumes the model shape and then minimizes the unknown parameters to fit that shape. In real-life scenarios, the model is not always completely known. Hence, the next thee methods implemented were data-driven. Then we implemented Gaussian Process regression. Gaussian process regression provides a Bayesian non-parametric approach and has a unique feature of predicting the targets with the confidence interval for each target. Prior information about the shape of the model can be incorporated by using different kernel functions and some of the kernel functions have the ability of automatic feature selection. Then we moved on to more complex and more expressive models. In search of more complex models, we selected fully connected neural networks as FNN models are complex but are easy to build. After analyzing the FNN models, we moved on to implementing Convolutional Neural Networks as they can extract the features from the spatial structure of data and have advantages like sparse interactions, parameter sharing, and equivariant representations.

These predicted motion parameters are then used to reconstruct the actual locations of k-space from where the data is collected and then reconstruct the original image by using the iterative non-uniform Fourier transform.

1.5. Datasets

The brain MRI images are huge and hence need more memory for storage and processing. The images are more complex and difficult to obtain. Because of no specific geometry, verifying the results with the analytical solution is not possible. Hence, It was decided to use some simpler models for initial experimentation instead of brain MRI data. As it was easy to generate images of the rectangle and it was possible to create a complete analytical model assuming rectangle as an image space object, it was decided to use rectangles with different dimensions as the experimental dataset. Multiple datasets were created using different rotation or translation or both motions in-between consecutive rows of k-space data. As we know that, the k-space equivalent of the rectangle is a 2D sinc function, creating k-space data with rotation was possible without introducing structural errors. As Signal to Noise Ratio (SNR) of typical MRI scanner is around 50, a dataset with SNR = 50 was also created from the noise-free dataset. The continuous motion was also simulated with time-dependent motion parameters. In figure 1.2a, a sample image created can be observed and in figure 1.2b, the effects of simulated motion artifacts can be observed.



(a) Synthetic Data test image (b) simulated rotation motion affected image

Figure 1.2: Synthetic data image with simulated motion artifacts

1.6. Organization of the report

This thesis report is divided into six chapters. In chapter 1, the background and objective of this thesis is discussed. The datasets created and used for the experiments were also discussed in the first chapter. In chapter 2, the previous work in the MRI motion correction field is discussed. In chapters 3, 4, 5 and 6 the experiments performed with Least squares regression, Gaussian process regression, Fully Connected Neural Networks and Convolutional Neural Networks for predicting motion parameters were discussed respectively. In chapter7, the results on brain image dataset were discussed. Finally, in chapter 8 the results, conclusion, and outlook of this project were discussed.

2

Related work

MRI motion can be handled using three methods [18]:

1. Preventing motion
2. Avoiding the effects of motion
3. Correcting the effects of motion

Preventing the motion using restraints is the most commonly used method but the level of immobilization achieved is low. To achieve higher levels of immobilization during a scan, custom made molds are used which have higher manufacturing time and are uncomfortable [3][22][7]. Another way to prevent motion is sedation. This may have an ill effect on patients like hypoxemia and is not useful on normal volunteers for research studies due to safety concerns[19]. Another commonly used method to prevent the motion is training the patients before their scan. This method is only useful on healthy subjects and cannot be used for unconscious or acutely ill patients[30].

To avoid the effects of motion, fast imaging techniques like Fast spin echo[12] and echo-planar imaging[20] comes at a cost of low image quality and/or resolution. To avoid the effects of motion, Radial or spiral imaging schemes can be used which are more robust to the motion artefacts[14][27][13][29]. Though the use of these acquisition methods is increasing in research, most commonly used acquisition method in clinical MRI is a Cartesian acquisition.

The third approach to handle motion artifacts is allowing the motion to occur, then detect, quantify and discard or correct the acquired data. In [17] a supervised machine learning approach using random forests to classify the motion affected image was implemented. Two types of motion are considered, bulk motion and respiratory motion respectively. The synthetic motion affected dataset was created by altering the k-space data according to a motion trajectory. Here, the problem addressed was only classification problem, detecting the presence or absence of motion. In [24] a deep CNN was developed to correct the motion affecting MRI images. In this paper, the focus was to correct the motion artifacts by using deep CNN as a motion correction filter in image space. In [15], the motion artifacts in brain MRI images were corrected using variants of Generative Adversarial Networks(GAN). In [25] a new method ”Periodically Rotated Overlapping ParallEL Lines with Enhanced Reconstruction” for data collection and reconstruction is developed. In this method, the central region of k-space is over-sampled helping the correction and reconstruction of motion affected data. This method needs more acquisition time as it collects more data by oversampling at the center. In [32], the radial acquisition was used with the registration technique for measuring and correcting rigid motion. Though this method implements motion correction in k-space but needs image

registration and radial acquisition techniques. In [16] using deep network architectures of Variational Auto-encoder and GAN, motion affected image to motion-free image translation problem was tried to solve. In this method, a motion-corrected image is generated from a motion-corrupted image. To do this, complete k-space data has to be collected first and hence real-time motion estimation is not possible.

In Figure 2.1, in the proposed method, estimation and correction of motion artifacts are done in k-space, saving time to reconstruct the image before artifacts correction. This is an advantage over the current methods as this method can be implemented to estimate the motion parameters in real-time. This is not only useful for prospective motion correction but also for interventional techniques, such as Radiotherapy which aims at tracking the tumor in real-time. The proposed method is implemented for Cartesian acquisition which is the most commonly used method in clinical MRI. In the proposed method, the aim is to estimating motion parameters using consecutive k-space rows as input. After estimation, the correction of motion artifacts is done by using a non-uniform Fourier transform for reconstruction. The advantage here is we do not need to reconstruct the complete image for artifacts correction and the method can be used as an online correction(”on the go”) method parallelly while capturing the next k-space row in a Cartesian acquisition. In the next chapters, we

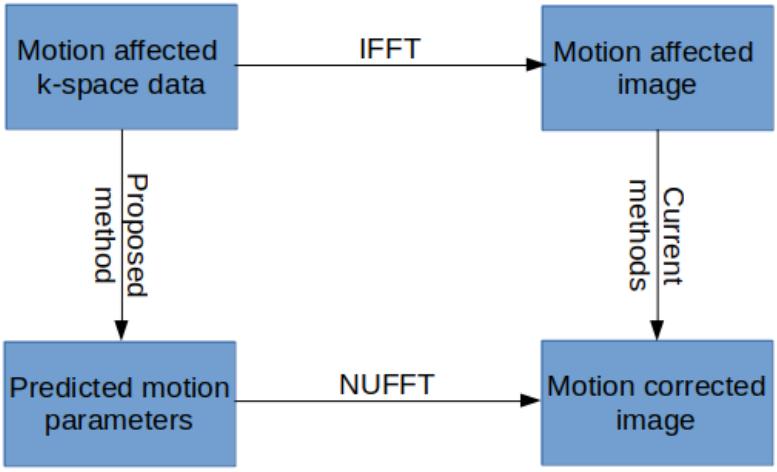
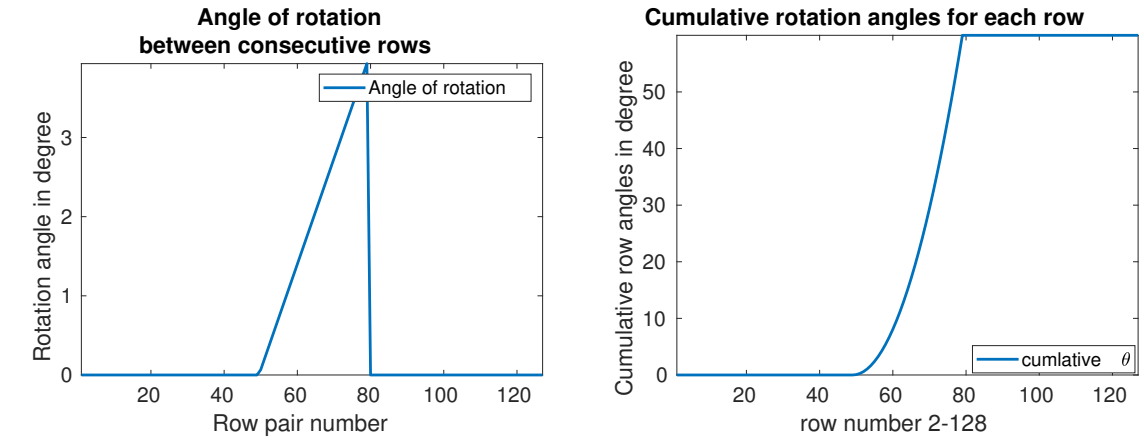


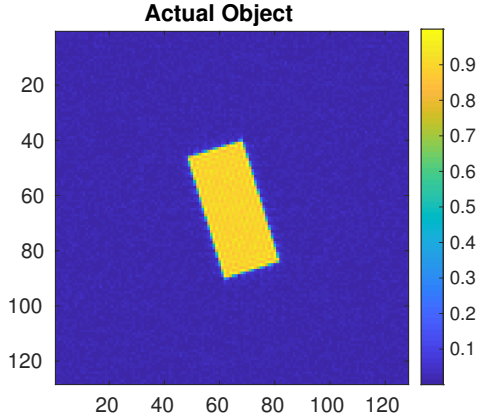
Figure 2.1: Proposed method

try to experiment with multiple regression methods to predict motion parameters. We start with a simple method like least squares and then experiment with methods like the Gaussian process, Fully connected Neural Network and Convolutional Neural Network. After achieving certain prediction accuracy, predicted values were used to improve the quality of the reconstructed image. The reconstruction is done by iterative Non- Uniform Fourier Transform(NUFFT) with the help of ”MIRT toolbox” by Fessler[10]. In figures 2.2c and 2.2d, you can see original and corrupted object due to rotation motion. The rotation angle in consecutive rows and its cumulative value is as shown in figure 2.2a and 2.2b. Using these cumulative angles, we create a non-uniform co-ordinates system from where the k-space raw data is collected(figure 2.2b). Using these values of raw k-space data and iterative NUFFT, we reconstruct the image as shown in figure 2.2f. In iterative NUFFT, we solve minimize $||F^{NUFFT}X - D||$ problem in an iterative way to find X, using linear least squares, where F^{NUFFT} is non-uniform Fourier transform operator, D is raw data from k-space and X is an object image.

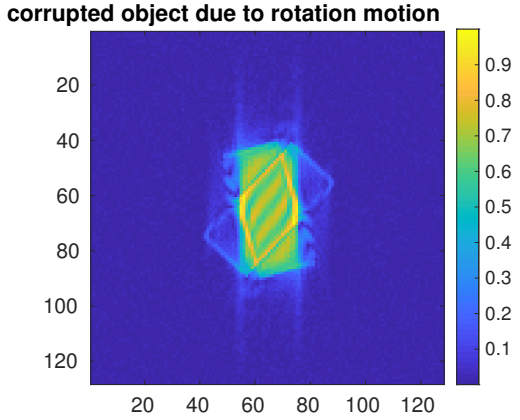


(a)

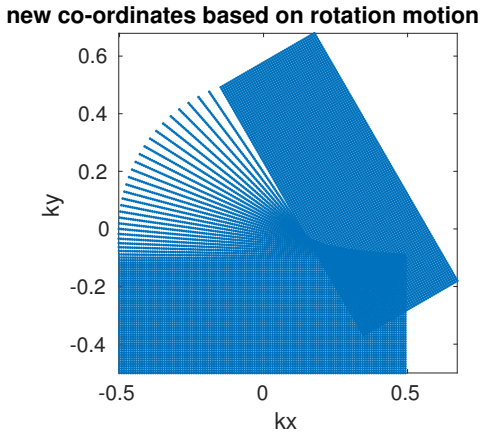
(b)



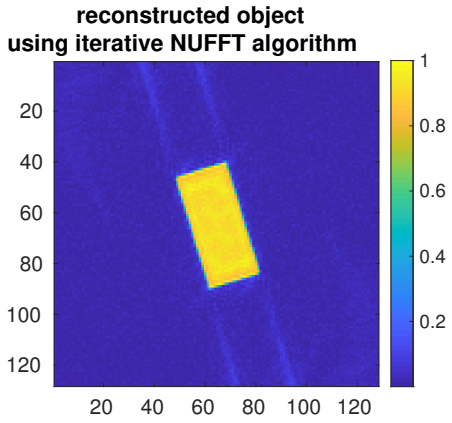
(c)



(d)



(e)



(f)

Figure 2.2: (a) rotation angle values for each row (b) cumulated values of rotation angles (c) original object under scan (d) reconstructed object using IFFT of raw k-space data (e) New co-ordinates of scanned data based on cumulated theta values (f) reconstructed object using new co-ordinates of scanned data, raw scanned data and iterative NUFFT

In the next chapter, we discuss the least squares method implementation for predicting motion parameters on synthetic dataset.

3

Least Squares Regression

To start estimating the motion parameters, we first assume the geometry of the object (rectangle object with dimensions A_x and A_y) is known and we try to estimate the motion parameters by minimizing the difference between collected k-space data and function of motion affected k-space(model). This estimation of motion parameters is done for each k-space row. Assuming we have an image of 128×128, we have 128 equations for each row and 3 unknowns $\theta, X_{trans}, Y_{trans}$. Though the simplest form is a linear model, after multiple experiments with the linear model it was observed that objective function is too complicated to fit with the linear model. Hence we start our discussion with Non-Linear Least Squares.

Non-Linear least squares method is used to solve a set of m non-linear equations in n unknowns($m \geq n$). With least-squares we try to estimate motion parameters $\theta, X_{trans}, Y_{trans}$ by fitting the expected model equation to scanned data. The objective function here will look like the equation below.

$$\underset{\theta}{\text{minimize}} \sum_{i=1}^{N} ||(f_i(\theta, x_{trans}, y_{trans}) - A(i,:))||^2. \tag{3.1}$$

where, $f_i(\theta, X_{trans}, Y_{trans})$ is k-space function of known object for i^{th} row(model equation) and A(i,:) is the k-space data acquired at i^{th} row. After solving this for each row of k-space data scanned, we get row-wise vectors for $\theta, X_{trans}, Y_{trans}$.
To solve non-linear least square, the trust-region-reflective algorithm was used by means of `MATLAB` "`lsqnonlin`" function[34].

3.1. Experiments with Non-Linear Least Squares

3.1.1. Experiment 1

The project aims to predict the motion parameters from the data and correct the motion affected part of that data. To do this, we started with a simple experiment. First, we tried to predict the motion parameters(in this case angle of rotation θ) between two images one with rotation and one without rotation. Here, we use $N \times N$ k-space data of the image of a rectangle of known dimensions rotated with angle θ and try to predict θ which is the angle of rotation. So, in this case, we have $N \times N$ equations and one unknown θ is to be predicted. In Figure 3.1, images on the left are of non-rotated and rotated rectangle and images on right are k-space data for the same. The k-space data values are real in this case as both the objects are centered and there is no translation motion in this case.

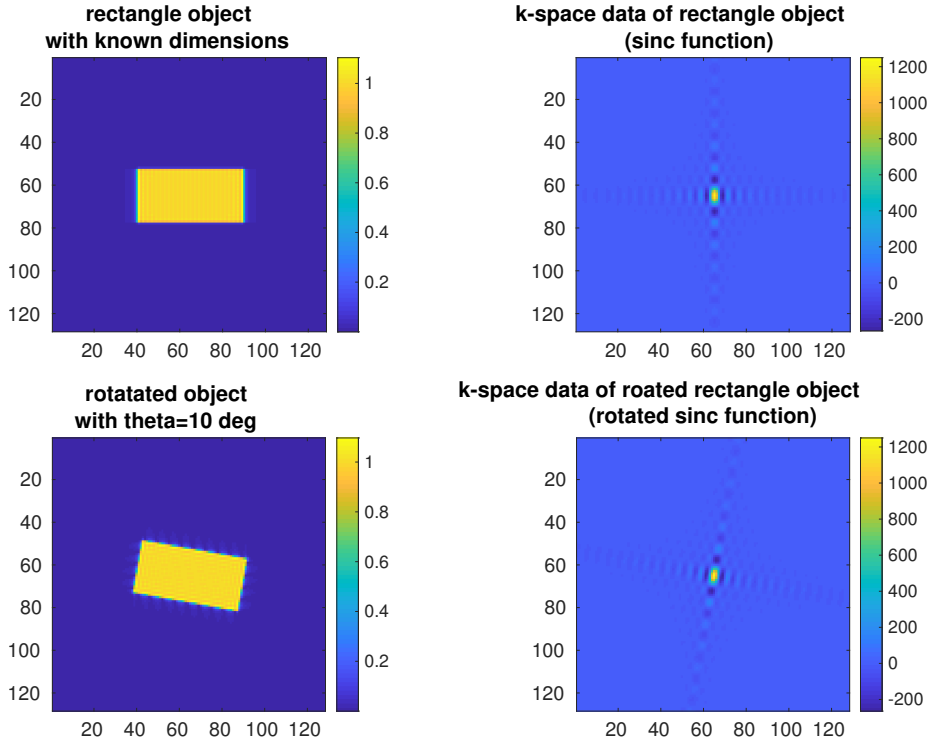


Figure 3.1: Simple model of rectangle with known dimensions and its corresponding k-space data

The objective function will look like,

$$\text{minimize}\Bigg(\sum_{i=1}^{N} ||(A_x)(A_y)(\text{sinc}((K_x\cos(\theta)+K_y(i)\sin(\theta))(A_x/N))\times \text{sinc}((-K_x\sin(\theta)+K_y(i)\cos(\theta))(A_y/N))) - I_{k_xk_y}||^2\Bigg). \tag{3.2}$$

In the above equation, K_x and K_y are k-space co-ordinates, A_x and A_y are rectangle dimensions, $N \times N$ is image size, and $I_{k_xk_y}$ is k-space read data matrix of size $N \times N$. We minimize the objective function to find the value of θ using least squares method. Value of objective function for different values of θ (domain plot) will look as shown in Figure 3.2. Here, the angle of rotation between two rectangles is $\theta = 10^o$. In figure 3.2 you can observe that the domain has a clear global minimum at 10 and -170 as the objective function is π periodic. So by using least squares, we can find the values of theta.

3.1.2. Experiment 2

From the previous experiment it was concluded that if the k-space signal function is known, we can use non-linear least squares to find the angle of rotation between two images. In the next experiment, we tried to find the angle of rotation between consecutive rows. As we are working with the Cartesian acquisition in k-space, the rows in k-space will be scanned one by one. This experiment aims to find the angle of rotation if the object rotates by angle θ

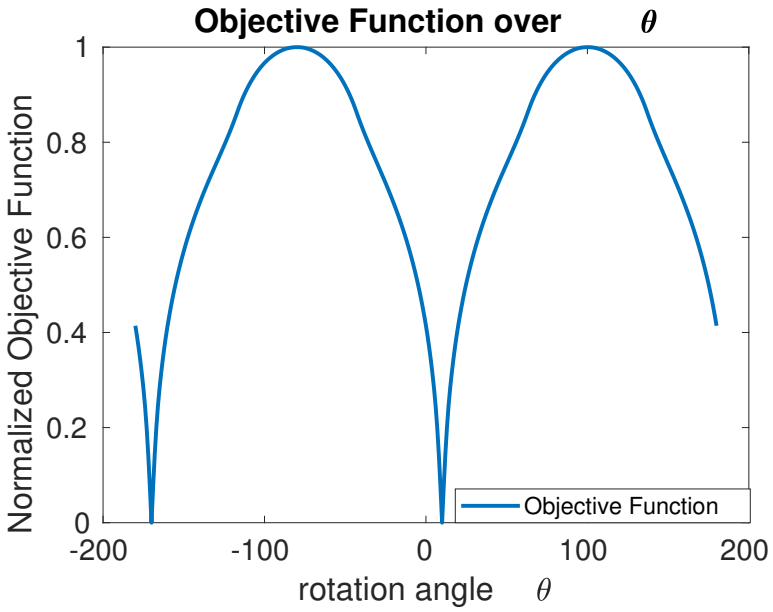


Figure 3.2: Plot of objective function over θ to find single θ value for complete image. The plot indicates that in noiseless case, clear global minima can be observed and using least squares, it is possible to predict angle of rotation θ when k-space function of object is known

between the scanning of two rows. In this experiment, we have a single row of N points as input data and we try to minimize the objective function for each row to predict the value of θ_i. So, we have N data points with N equations and a single unknown θ_i to fit using least squares for each row. For a k-space data of $N \times N$, N rotation parameters are estimated.

$$\text{minimize}\Bigg(\sum_{i=1}^{N} ||(A_x)(A_y)(\text{sinc}((K_x\cos(\theta_i) + K_y(i)\sin(\theta_i))(A_x/N))\times \text{sinc}((-K_x\sin(\theta_i) + K_y(i)\cos(\theta_i))(A_y/N))) - I_{k_xk_y}(i,:)||^2\Bigg). \tag{3.3}$$

In equation (3.2) a single θ is to be predicted for $N \times N$ input data whereas in equation (3.3), we try to predict N θ values for $N \times N$ input data.

In the Figure 3.3, we show the similar domain plot as in Figure 3.2, but for every k-space row. On the x-axis we have row number, on the y-axis we have values of θ and on z-axis objective function which is to be minimized is plotted. The objective function is the difference between the assumed k-space function and the collected k-space data. In Figure 3.3, it can be observed that the objective function becomes difficult to minimize as we move away from the center due to multiple local minima values. From figure 3.1, it is clear that low frequencies contribute more to the k-space data (describing the global shape, contrast)than high frequencies(describing edges, corners). At the center of the k-space, we have lower frequencies and hence it is easier to minimize the objective function at the center. Hence, it was decided to use the minimization at center first for predicting the motion parameter values and then using the predicted values at the center for initializing the minimization for next rows and so on in a recursive fashion; The solution of the objective function of i^{th} pair of lines is used to initialize the minimization of objective function for $i + 1^{th}$ pair of lines.

The object under consideration is a rectangle with known dimensions. The object starts rotating from the original position slowly as shown in Figure 3.5 when capturing of k-space data is started. The object which is reconstructed from raw k-space data acquired is shown in figure 3.4.

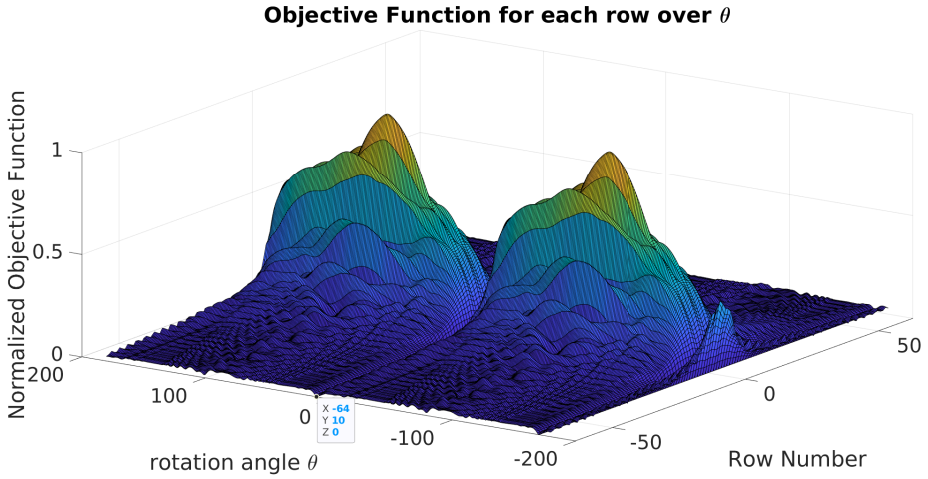


Figure 3.3: Normalized objective function plot for each k-space row over θ. As we move away from central row 0 to row ± 64 the objective function has more and more local minima values.

In 3.5 the dotted line indicates actual motion occurred and the blue solid line indicates predicted values using non-linear least squares. The rotation parameter θ is time-dependent and is sinusoidal.

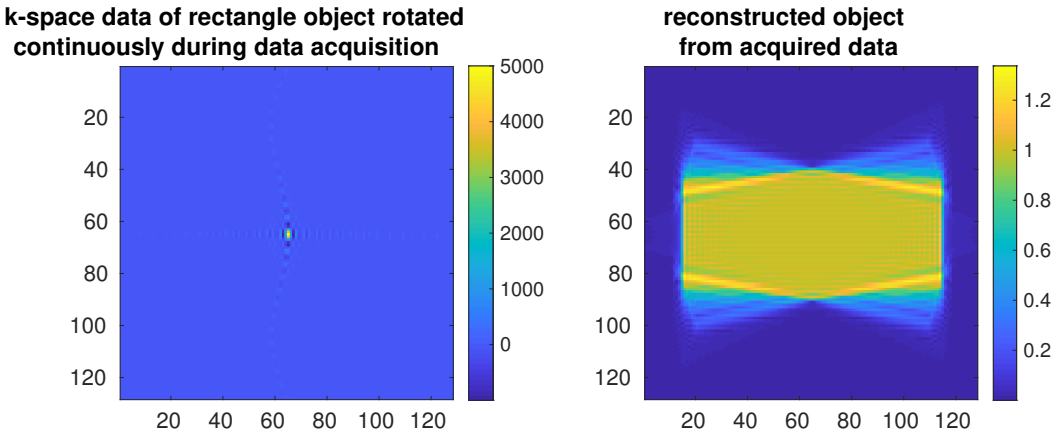


Figure 3.4: k-space data scanned with continuous object rotation in sinusoidal motion pattern and reconstructed object with inverse Fourier transform.

From this experiment, it was concluded that in noiseless case, if k-space data function is known, rotation parameters can be estimated using non-linear least squares considering

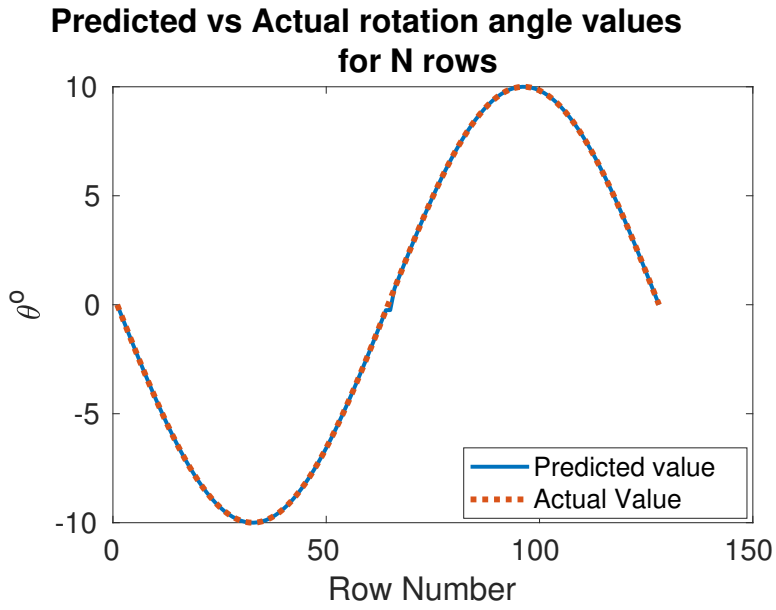


Figure 3.5: Prediction of rotation motion for each row for continuous sinusoidal rotating motion. Here, central minimization is done first and using the results to initialize next minimization, good prediction of motion values is obtained in noiseless data. Object geometry and k-space model of object is assumed to be completely known.

motion is smooth. Now, in the next experiment, it was tried to find out if non-linear least squares can be used to estimate multiple motion types together(rotation and translation) in the presence of noise.

3.1.3. Experiment 3

In this experiment, we try to find if for a known object, with multiple types of motions (rotation and translation) and when acquired k-space data is affected by noise, if motion parameters can be estimated. From equation (1.4) and equation (1.6), k-space equation (3.4) can be derived assuming object in image space is a rectangle of dimensions A_x and A_y, image size is $N \times N$ and $\theta_i, x_{trans_i}, y_{trans_i}$ are rotation and translation parameters along x and y-axis for i^{th} row. $I^{\mathcal{N}}_{k_x k_y}$ is k-space noisy data scanned. For different noise levels, we try to find the motion parameters using least squares.

$$\text{minimize}\left(\sum_{N}^{i=1} ||\big((A_x)(A_y)(\text{sinc}((K_x\cos(\theta_i)+K_y(i)\sin(\theta_i))(A_x/N))\times \text{sinc}((-K_x\sin(\theta_i)+K_y(i)\cos(\theta_i))(A_y/N)))\times e^{-i2\pi(((K_x\cos(\theta_i)+K_y(i)\sin(\theta_i))(x_{trans_i}/N))+((-K_x\sin(\theta_i)+K_y(i)\cos(\theta_i))(y_{trans_i}/N)))}\big) - I^{\mathcal{N}}_{k_x k_y(i)}(i,:)||^2\right). \quad (3.4)$$

In figure 3.6a, 3.6b and 3.6c, the effect on noise on the predictions for motion parameters can be observed. Here, you can observe the effect of noise is more at the rows away from central k-space. As we discussed before, the amplitudes of high frequencies are less, the effect of noise on high-frequency data is more than the effect of noise on low-frequency data. So, for the same amount of noise, higher frequencies affect more than the lower frequencies.

When SNR value goes higher(figure 3.6), the motion parameters estimated are close to actual values not only at the center but also away from the center. Here, the motion parameters are changing slowly and the object geometry is completely known. Now, in the next experiment, we tried to estimate fast-changing motion value parameters using least squares.

3.1.4. Experiment 4

In this experiment, the objective is to find if the least-squares minimization can estimate the motion parameters of the fast-moving rectangle of known dimensions by using noisy scanned data in k-space. Here, we consider the rectangle object is rotating with an angle of 2^o per row and translation motion parameters are the same as the previous experiment(slowly moving). The only change from the previous experiment is the fast-changing rotation motion parameter. Similar to previous experiment, equation (3.4) is used for minimization. As seen in Figure 3.7, we get good predictions at the center but as we move away from the center, predictions start deviating from actual motion parameter values. This is analogous to the observation made using figure 3.3 where it was observed that the domain has global minima at the center but as we move away from the center, the objective function has more local minima. As the value of one of the motion parameters is changing fast, as we move away from the center, the least square algorithm tends to get stuck in local minima. As we are using prediction from the previous row for initializing the algorithm to predict the motion parameters for next rows, once the algorithm gets stuck in local minima, the predictions of motion parameters from that row go wrong and the error in the prediction increases.

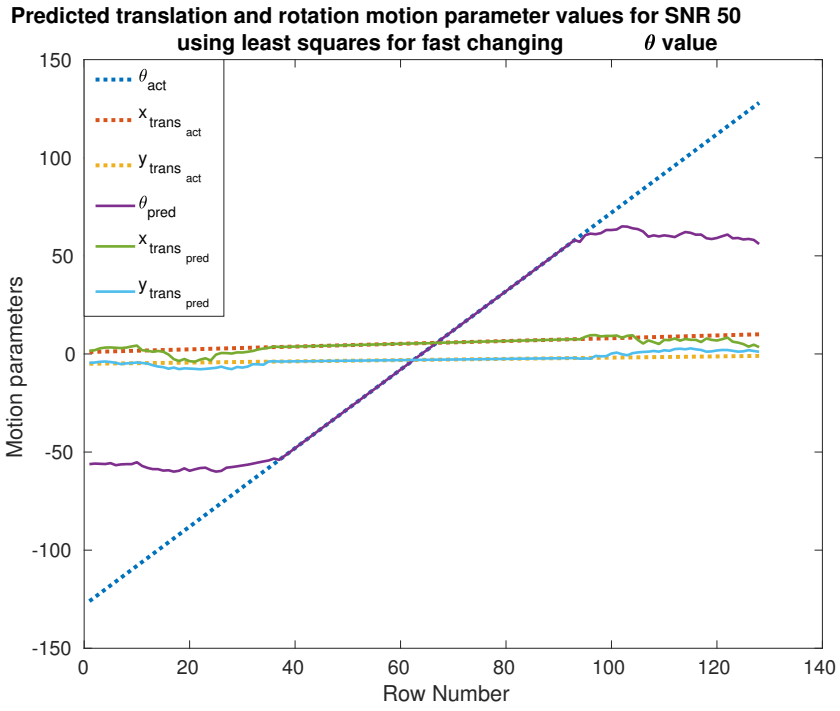
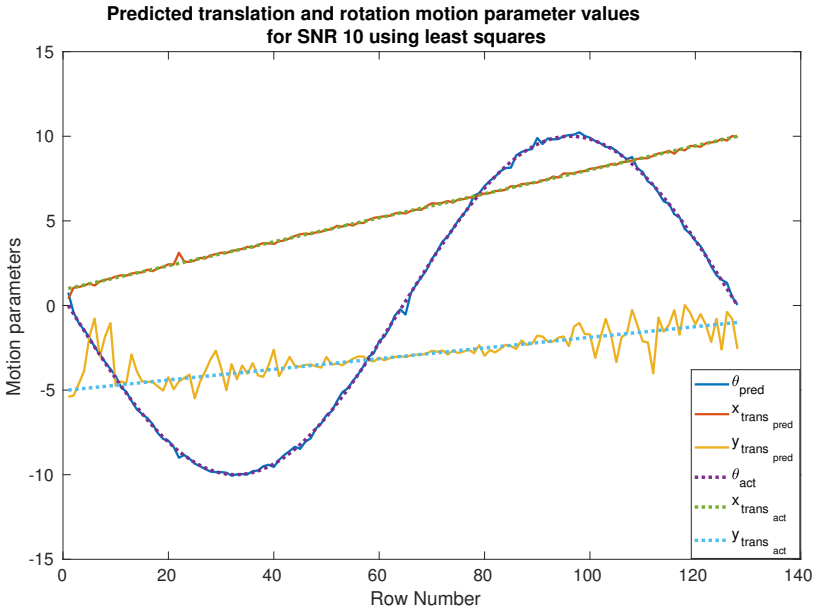
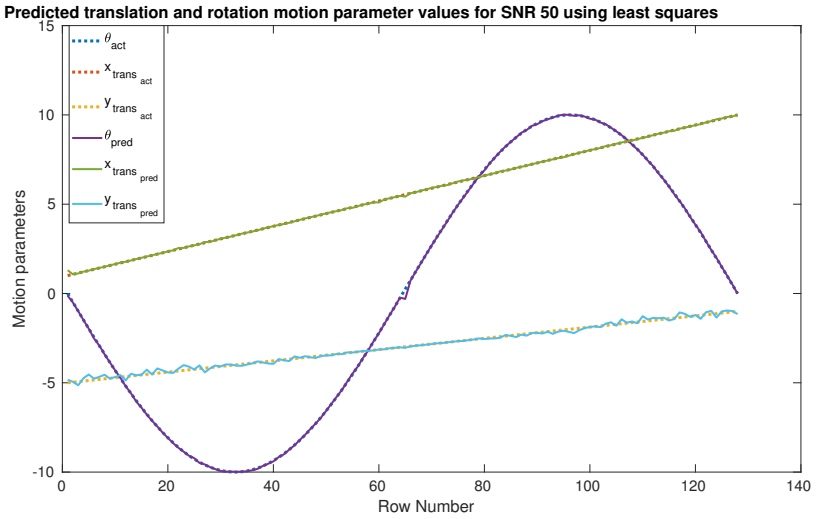


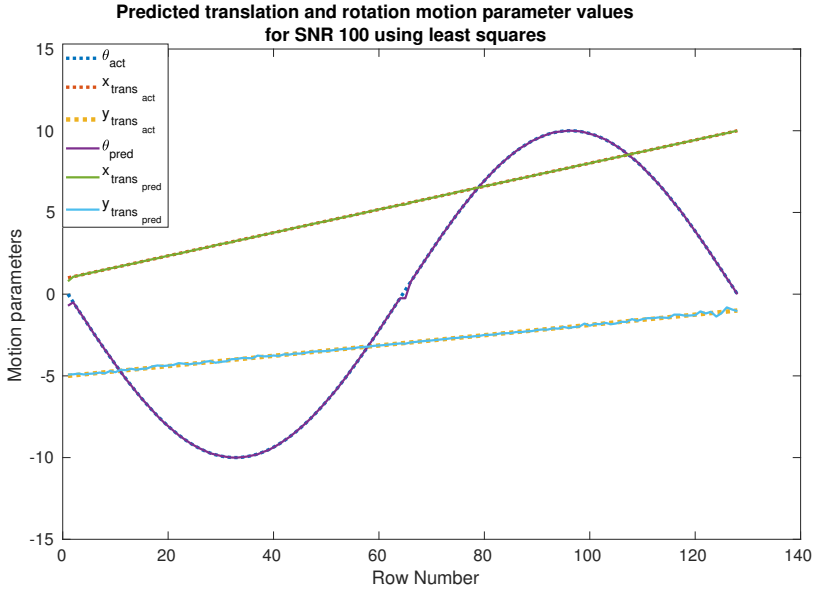
Figure 3.7: Prediction of rotation and translation motion for each row. SNR of scanned data is 50. The value of θ is changing by 2^o per row. Here, the motion is smooth but fast. The values predicted at the center and near central region are close to actual values but as we move away from the center, predictions are wrong due to local minima issue. Once the predictions go wrong for single row, as we use the same prediction for initialization of next row motion parameter predictions, all next predictions go wrong.



(a) SNR=10



(b) SNR=50



(c) SNR=100

Figure 3.6: Prediction of rotation and translation motion for each row. SNR of scanned data is 10, 50, 100 respectively. Object is a rectangle with known dimensions.

From experiment 4, it was concluded that if the motion value parameters are changing fast or the motion occurs when k-space lines away from the center are being scanned, it is difficult to predict the motion parameters. Also, with the least-squares minimization, the object geometry is supposed to be known which is never the case in real-life scenarios. In experiment 3, it was observed that predictions using least-squares are affected by noise more for the rows away from the central k-space data. The least-squares minimization is used as the first step to understand the nature of the problem. In the next step, we will discuss Gaussian Process regression method and experiments performed with the same.

4

Gaussian Process Regression

In the previous section, we experimented with nonlinear least squares to find if we can predict the motion parameters when object geometry is known. In a practical scenario, the object geometry will be unknown and the motion parameters will need to be predicted. For this purpose, we need a regression model that will read the k-space data (pair of rows in this case) and will find the motion parameters associated with it (data-driven). We selected the Gaussian Process (GP) regression as a starting point of regression models for the following reasons. The Gaussian process provides a Bayesian non-parametric approach for regression and classification. It directly captures the model uncertainty. When using GP, prior knowledge and specifications about the shape of the model can be used by selecting different kernel functions. Automatic Relevance Determination (ARD) kernels in Gaussian Process can be used for automatic feature selection.

In this chapter, we discuss theory, experiments, and results of Gaussian Process models for predicting motion parameters in k-space.

A Gaussian Process is defined as a probability distribution over functions $y(x)$ such that the set of values of $y(x)$ evaluated at an arbitrary set of points $x_1, \ldots, x_N$ jointly have a Gaussian distribution[4].
Now consider a function,

$$y(x) = w^T \phi(x). \tag{4.1}$$

where x is the input vector and w is M- dimensional weight vector. Now, consider prior over w is given by a Gaussian,

$$p(w) = \mathcal{N}(w|0, \alpha^{-1} I). \tag{4.2}$$

where α is a hyperparameter that represents the precision of distribution. In practical scenarios, we want to model the values of y at specific x values i.e training data points $x_1, \ldots x_N$ Hence we are interested in joint distribution of the function values $y(x_1), \ldots y(x_N)$, which we denote by the vector $\mathbf{y}$ with elements $y_n = y(x_n)$ for $n = 1, \ldots, N$.

$$\mathbf{y} = \Phi \mathbf{w}. \tag{4.3}$$

where Φ is the design matrix with elements $\Phi_{nk} = \phi_k(x_n)$ From equation (4.2), we can conclude that,

$$\mathbb{E}[\mathbf{y}] = \Phi \mathbb{E}[w] = 0. \tag{4.4}$$

$$\text{cov}[\mathbf{y}] = \mathbb{E}[\mathbf{y}\mathbf{y}^T] = \Phi \mathbb{E}[w w^T] \Phi^T = \frac{1}{\alpha} \Phi \Phi^T = K. \tag{4.5}$$

$$K_{nm} = k(x_n, x_m) = \frac{1}{\alpha} \phi(x_n)^T \phi(x_m). \tag{4.6}$$

where, $k(x, x^{'})$ is kernel function. To apply Gaussian Process models to the problem of regression, we need to take account of the noise on the observed target values, which are given by

$$t_n = y_n + \epsilon_n. \tag{4.7}$$

Here, we consider Gaussian noise processes,

$$p(t_n|y_n) = \mathcal{N}(t_n|y_n, \beta^{-1}). \tag{4.8}$$

where β is a hyperparameter representing noise precision. As the noise is independent for each data point, the joint distribution of the target values $t = (t_1, ..., t_N)^T$ conditioned on the values of $y = (y_1, ..., y_N)^T$ is given by an isotropic Gaussian of the form,

$$p(\mathbf{t}|\mathbf{y}) = \mathcal{N}(\mathbf{t}|\mathbf{y}, \beta^{-1}\mathbf{I_N}). \tag{4.9}$$

where I_N denotes the $N \times N$ unit matrix.

$$p(\mathbf{y}) = N(\mathbf{y}|0, \mathbf{K}). \tag{4.10}$$

where $p(y)$ is marginal distribution is a Gaussian with mean 0 and the covariance matrix is K. In order to find the marginal distribution $p(t)$, conditioned on the input values $x_1, ..., x_N$, we need to integrate over $\mathbf{y}$.

$$p(\mathbf{t}) = \int p(\mathbf{t}|\mathbf{y})p(\mathbf{y})d\mathbf{y} = N(\mathbf{t}|0, \mathbf{C}). \tag{4.11}$$

Where $\mathbf{C}$ is covariance matrix with elements,

$$\mathbf{C}(x_n, x_m) = k(x_n, x_m) + \beta^{-1}\delta_{nm}. \tag{4.12}$$

Now predictive conditional distribution is given by,

$$p(\mathbf{t_{N+1}}) = N(\mathbf{t_{N+1}}|\mathbf{0}, \mathbf{C_{N+1}}). \tag{4.13}$$

Here, $\mathbf{C_{N+1}}$ is given by, $\mathbf{C_{N+1}} = \begin{bmatrix} \mathbf{C_N} & \mathbf{k} \\ \mathbf{k^T} & c \end{bmatrix}$ where $\mathbf{C_N}$ is the $N \times N$ covariance matrix with elements given by equation (4.12) for $n, m = 1, ..., N$, the vector k has elements $k(\mathbf{x_n}, \mathbf{x_{N+1}})$ for $n = 1, ..., N$, and the scalar $c = k(\mathbf{x_{N+1}}, \mathbf{x_{N+1}}) + \beta^{-1}$.
Conditional distribution is given by $p(t_{N+1}|\mathbf{t})$ which is a Gaussian distribution with mean and covariance given by,

$$m(x_{N+1}) = \mathbf{k^T C_N^{-1} t} \qquad \sigma^2(x_{N+1}) = c - \mathbf{k^T C_N^{-1} k}. \tag{4.14}$$

Equation (4.14) is used in predicting the future/missing values of time series.

4.0.1. Squared Exponential Kernel

Depending on the prior information about the data, different kernels (**k**) are used to suit the structure of data. The squared exponential is one of the most widely-used kernels within the kernel machines field [5]. The squared exponential kernel function for two inputs x and x' is shown in equation (4.15).

$$k_{se}(x, x') = \sigma^2 \exp(-0.5||x - x'||^2/l^2). \tag{4.15}$$

The squared exponential kernel has two parameters (l and σ). The former is known as the length-scale parameter, which controls the horizontal scale over which the function changes and the latter is the amplitude of the process, which controls the vertical scale changes[23]. This covariance function is infinitely differentiable, which means that the GP with this covariance function has derivatives of all orders, and the resulting Gaussian Process using this kernel is thus very smooth.

4.0.2. Learning the hyperparameters

Prediction of the Gaussian Process model will be depending on $\mathbf{k}$ i.e covariance function as seen equation (4.14). Instead of using fixed covariance function, we use a family of functions parameterized by few hyperparameters and then using maximization of the marginal likelihood, we tune the hyperparameters for these functions [4].

The log-likelihood function for a Gaussian Process regression model is easily evaluated using the standard form for a multivariate Gaussian distribution, giving

$$\ln p(\mathbf{t}|\boldsymbol{\theta}) = -\frac{1}{2}\ln|\mathbf{C}_N| - \frac{1}{2}\mathbf{t}^T\mathbf{C}_N^{-1}\mathbf{t} - \frac{N}{2}\ln(2\pi). \tag{4.16}$$

Here, $\boldsymbol{\theta}$ denotes the hyperparameters of the Gaussian Process model. Here, we want to maximize LHS with respect to $\boldsymbol{\theta}$. In practice, we do this by minimizing the negative log-likelihood. As equation (4.16) is non-convex, there is a chance of getting stuck in local minima if initialization is not good enough. Hence, we have to be careful while initializing the hyperparameters $\boldsymbol{\theta}$. For initialization, we find the value of $\ln p(\mathbf{t}|\boldsymbol{\theta})$ over a range of $\boldsymbol{\theta}$ values and use the best combination of $\boldsymbol{\theta}$ values for initialization.

In addition to the covariance kernel hyperparameters, there are two more hyperparameters to be taken care of. First, one is mean, but as we assume that the process is a zero-mean process, this parameter can be skipped from the optimization process. The second parameter is noise variance indicating the effect of noise in the system. If this parameter is initialized lower than the noise present in the system, the model might fail to fit the data. Whereas, if the model fits very well to the data, this parameter will be optimized to the value close to the noise level in the system.

4.1. Experiments with Gaussian Process regression

In this experiment, we use N^{th} and $N+1^{th}$ rows concatenated together as input and motion parameters (in this case rotation angle θ) as output. We have generated a noiseless dataset of 10,000 samples which then split to training and validation data. The size of image $N \times N$ is 128×128 from which rotated and non-rotated k-space rows are sampled, angle of rotation between consecutive rows is limited to $\pm 5^o$, the dimensions of rectangle object, A_x and A_y are in a range of 10-100.

4.1.1. Experiment 1

In this experiment, we will try to find if the simple squared exponential kernel can map the relationship between input-output for predicting rotation angle for noiseless data. As we know that the central part k-space has more information as lower frequencies contribute to k-space more than higher frequencies, we first try to build a model for predicting the rotation angle between row 64-65 (image size 128×128). For this experiment, we use Matlab R2019a, prtools [26] and Gaussian Process regression and classification Toolbox version 4.2 [6].

In the first part of this experiment, we try to explore how the size of the training dataset affects validation error. In figure 4.1, the validation error (MAE) for 1000 samples is shown for training dataset from 100 samples to 3000 samples. Here, you can see for more training samples, the training error is reducing and for 3000 training sample the validation error

(MAE) is 1.96^o. In equation (4.16), you can observe that for learning the hyperparameters, we need to calculate $\mathbf{C}_N^{-1}$, which has a computational complexity of $O(n^3)$. With the system hardware limitations, we have to limit the number of training samples to 3000.

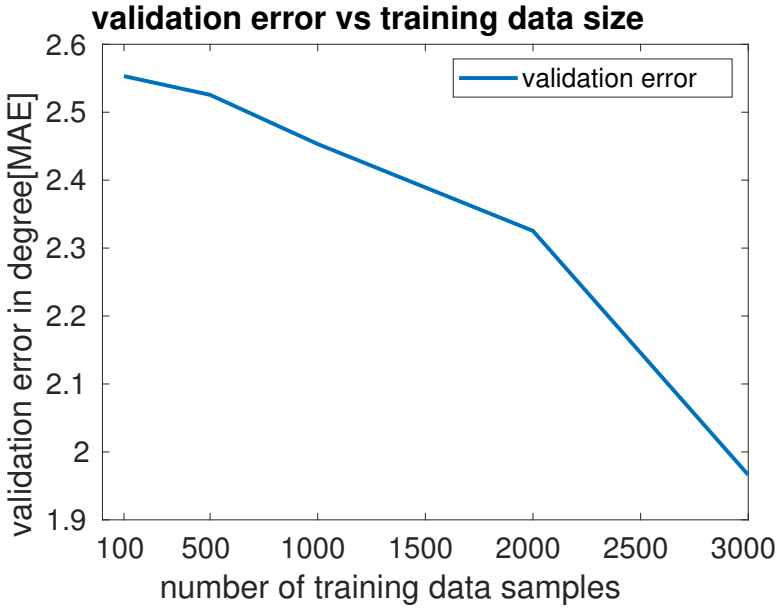


Figure 4.1: Validation error for Gaussian Process regression when training data size is increased. The experiment is done for central k-space row pair (64-65)

From figure 4.1, the error for predicting rotation motion parameter which is in range $\pm 5^o$, is 1.996^o for training data size of 3000 samples. The mean absolute value of actual output targets (validation set) is 2.56^o whereas validation error is 1.996^o indicating the performance of the model is poor (as the mean absolute value of targets is 2.5^o). Next, we try to analyze the reason behind high validation error.

In the squared exponential kernel, there are two hyperparameters to be tuned l and σ as seen in equation (4.15). In figure 4.2, we have shown the normalized log of negative log-likelihood against the log of two hyperparameters (l and σ). To plot this, first, we calculate the negative log-likelihood against the log of two hyperparameters. In this negative log-likelihood, it is difficult to see the minima and hence we calculate the log of the negative log-likelihood, then normalize it and plot it against the two hyperparameters. In this plot, it can be seen that for lower values of $\log(l)$, the normalized log of negative log-likelihood is minimum. After minimization, the value of l was observed around 0.13 which is small.

If the value of l parameter is small, according to equation (4.15), diagonal values for the kernel matrix will be close to σ^2 and non-diagonal values of kernel matrix will be small. We know that the Kernel matrix can be considered as a function of the correlation between inputs. So, smaller off-diagonal values indicate that the inputs are correlated just to themselves and not to the other inputs. This lack of correlation between inputs is a characteristic of noise. This shows that the model is unable to map the relation between inputs and hence considers input samples as noise. Another reason for the failure of model might be due to the shape of domain (figure 4.2) with multiple local minima values and minimization results depend on a good initialization (We used 0.5 steps for both log hyperparameter values to plot domain. Smaller steps might give better domain plot but is computationally expensive). Training error is $0.7015 \times 10^{-}9$ indicating that the model predicts the output perfectly only for training data samples.

From the results of this experiment, it was concluded that the data is too complex for a Gaussian Process and hence it maps the inputs as noise. The experiment was performed

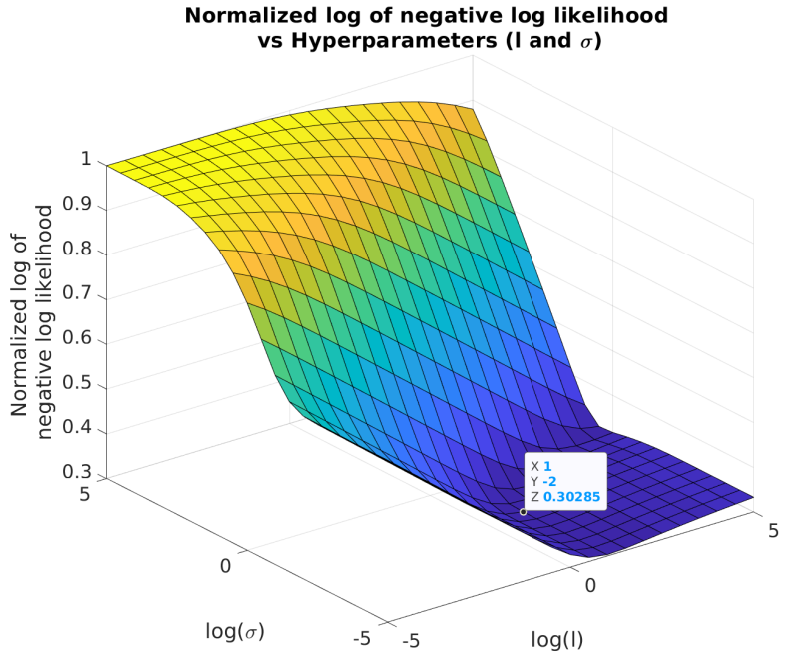


Figure 4.2: Plot of normalized log negative log likelihood vs hyperparameters

with noiseless data (initializing noise hyperparameter to a very small value) and for central pair rows (row number 64-65). With this scenario, if the model fails on validation data, it was assumed that the model will fail for the data away from k-space and also for noisy data.

Hence, we do not investigate further using Gaussian Process regression and go for the more complex model like fully connected neural networks and convolutional neural networks.

5

Fully Connected Neural Networks

After discussing the Gaussian processes, we now try to explore more complex and more expressive models like Artificial Neural Networks. In this section, we discuss model architecture, experiments, and results of fully connected regressive neural networks for predicting the motion parameters in k-space.

5.0.1. Introduction to fully connected neural networks

In fully connected neural networks, every neuron in a layer is connected to every other neuron in the previous and the next layer. The first layer is the input layer with dimensionality depending on input size and the final layer is called the output layer with dimensionality depending on output size. The layers in between these two layers are called hidden layers and the number of hidden layers and their dimensionality is dependent on model complexity requirements. Every connection between two neurons is given a weight. Weights are tuned by minimizing the loss function. The activation function is used to introduce non-linearity in the model.

Consider a fully connected neural network(one-layer) with input dimensions "m" and output dimensions "n". Here, $x \in \mathbb{R}^m$ represents the input and $y_i \in \mathbb{R}$ be the i^{th} output of fully connected layer. The output y_i can be represented as,

$$y_i = \sigma(w_{i,1}x_1 + \ldots + w_{i,m}x_m). \tag{5.1}$$

where σ is non-linear function and w_i are weight parameters which are to be learned. The complete output y(m dimensions) is represented as,

$$y = [\sigma(w_{1,1}x_1 + \ldots + w_{1,m}x_m); \sigma(w_{2,1}x_1 + \ldots + w_{2,m}x_m); \ldots; \sigma(w_{n,1}x_1 + \ldots + w_{n,m}x_m)]. \tag{5.2}$$

Equation (5.2) can also be represented as matrix multiplication,

$$y = \sigma(Wx). \tag{5.3}$$

where W is a matrix in $\mathbb{R}^{n \times m}$ and the non-linearity σ is applied component-wise. Such multiple layers are stacked together to create a multi-layer fully connected neural network. The output of such a multi-layer neural network with l layers can be described as,

$$y = \sigma_l(W_l(\sigma_{l-1}(W_{l-1}(\ldots\sigma_1(W_1x))))). \tag{5.4}$$

Model
We use a Fully connected Neural Network of 3 hidden layers of 128 neurons each with one output for predicting rotation angle. For all layers except the output layer, the ReLu activation function was used and for the output layer, the linear activation function was used. L2 regularization parameter (λ) was selected to be 0.005 and batch size was selected to be 512. Gaussian kernel initialization was selected. The "adam" optimizer was selected and "mean absolute error" was used as a loss function for training. 10,000 data samples were used for training each model and 1000 samples were used for validation. FNN model was trained a number of times on 10000 training samples and validated with 1000 samples(simple validation was used as cross-validation is computationally expensive and data generated is uniformly sampled data) by varying the hyperparameters and the model with the best performance on validation data was selected.

For training and testing the model, we use `python` 3.6.8 with `TensorFlow` version 1.7.0. NVIDIA TITAN X_p (12 GB) GPU was used to train and test the models.

5.1. Experiments with Fully Connected Neural Network

5.1.1. Experiment 1

In the first experiment, we try to find out if we can use "fully connected neural networks" to predict motion parameters for consecutively scanned k-space rows. The motion parameter, in this case, is the angle of rotation between consecutive rows. The datasets used here were the same as the previous experiment(A_x, A_y are in the range of 10-100 and angle of rotation between consecutive rows is in the range $\pm 5^o$). We start the experiment with central k-space row pair and then we move on to the k-space row pair away from the center. In the first part of this experiment, we use a noiseless dataset. In figure 5.1, you can observe that validation error(MAE) reduces to 0.65^o for fully connected neural network model trained on data near to central k-space rows(row pair 64-65) whereas, for model trained on data away from central k-space values(row pairs 1-2 and 127-128), validation error is in the range of $1.25^o - 1.5^o$. The mean absolute value of targets(rotation angle) is 2.5^o as targets are uniformly distributed in the range $\pm 5^o$. So the MAE of $1.25^o - 1.5^o$ indicates that the model is performing poorly with a very high validation error.

In the second part of this experiment, we try to find out if the same difficulty in learning for the model trained on k-space rows away from the center is observed in noisy data (SNR=50) as well. In both figures 5.1 and 5.2, it can be observed that the model trained on central k-space performs better than the model trained on data away from central k-space. In the noiseless case, it was observed that though the model trained on data away from central k-space performs worse than the model trained on central k-space data, the training and validation errors reduce when trained for more epochs (refer figures 5.3b and 5.3c). Whereas in the noiseless case for the model trained on data away from central k-space(figures 5.4b and 5.4c), after the first 5 epochs, the validation error does not reduce (indicating model is unable to learn) and reduction in the value of only training error can be observed. This indicates that the model fails to predict the target values (rotation angle) for unknown data(validation data) and can only learn targets for training data. When the validation error is higher than the training error it is considered as overfitting. This might be the effect of a complex model (FNN has a large number of training parameters) or less training data. Also due to the effect of noise, the model performance reduces and its effect is significant for models trained on data away from central k-space.

From the experiments performed on noiseless and noisy data, it was observed that the model works better on central k-space data. But for predicting the motion parameter values for k-space rows away from the center, the model fails and gives poor validation results.

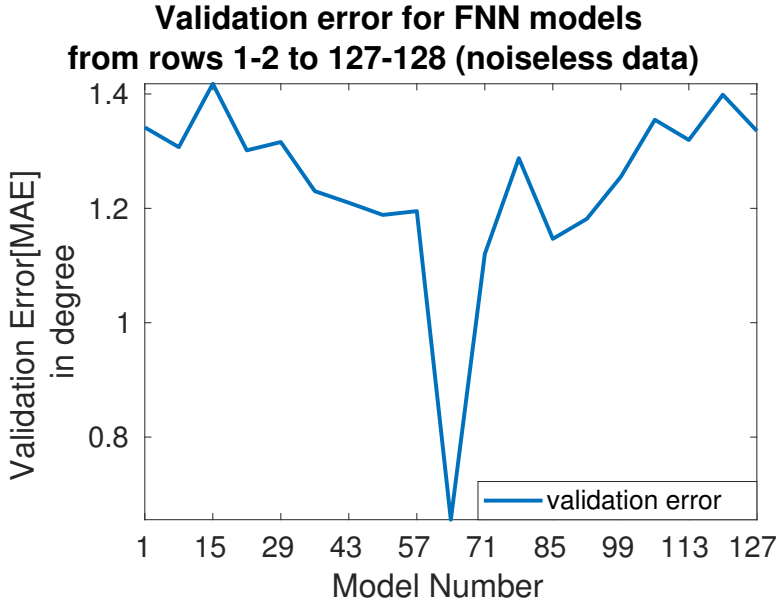


Figure 5.1: Mean Absolute error for 1000 validation samples for models trained on equally spaced noiseless k-space row pairs from 1-2 to 127-128

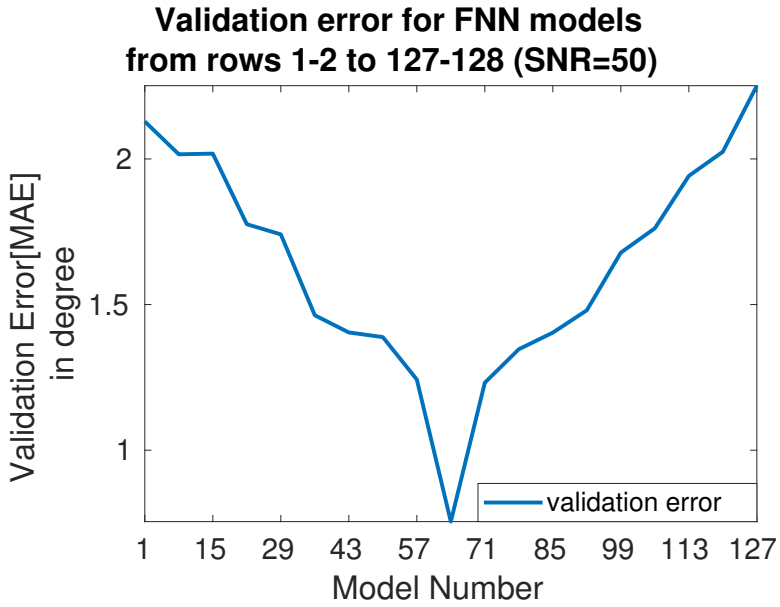
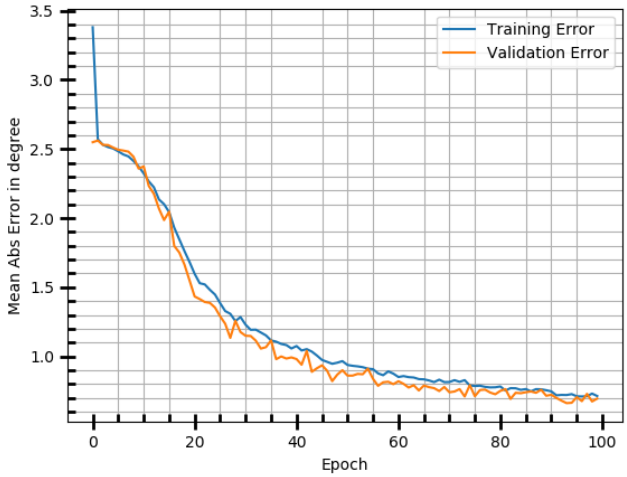
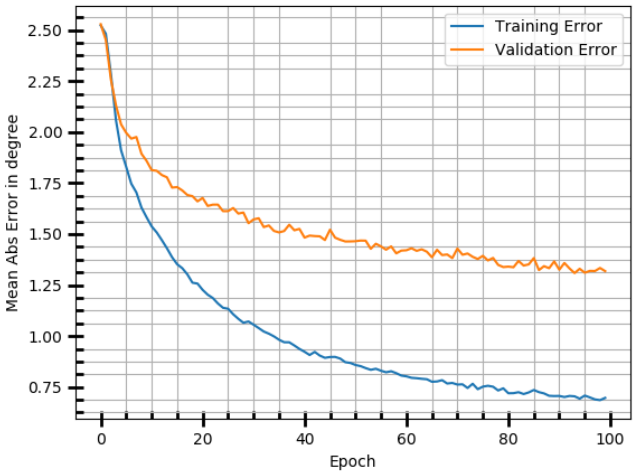


Figure 5.2: Mean Absolute error for 1000 validation samples for models trained of equally spaced k-space row pairs (SNR=50) from 1-2 to 127-128

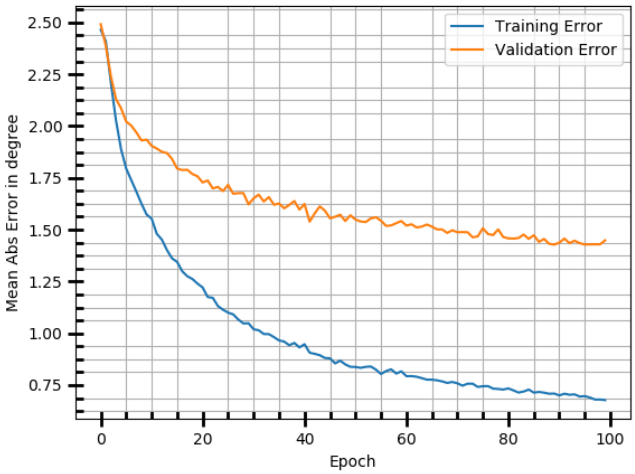
The datasets we are using are of rectangle objects. The k-space function for a rectangular object is a 2-D sinc function. From figure 3.1, it can be observed that the contribution of low-frequency data(central k-space) is higher than the contribution of high-frequency data in k-space. Due to this, the model trained on central k-space data can learn from the data and hence can predict with validation error(MAE) of 0.7^o. Due to small high-frequency components in k-space, the model trained on that data fails to learn any pattern or relationship in input and output data and hence fails to predict the output.
This model works better for non-scaled data rather than scaled data. When we scale the data feature-wise, the model loses the information about the shape of the k-space function(2D sinc



(a) Training and Validation error after each epoch for 100 epochs for row pair 64-65 i.e. central k-space row pair

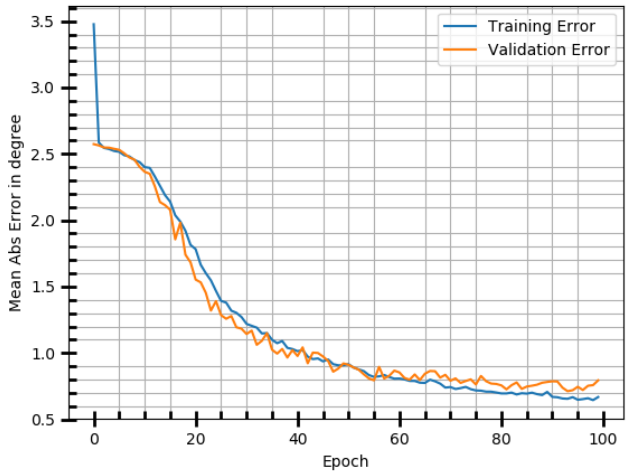


(b) Training and Validation error after each epoch for 100 epochs for row pair 1-2 i.e. row pair away from center

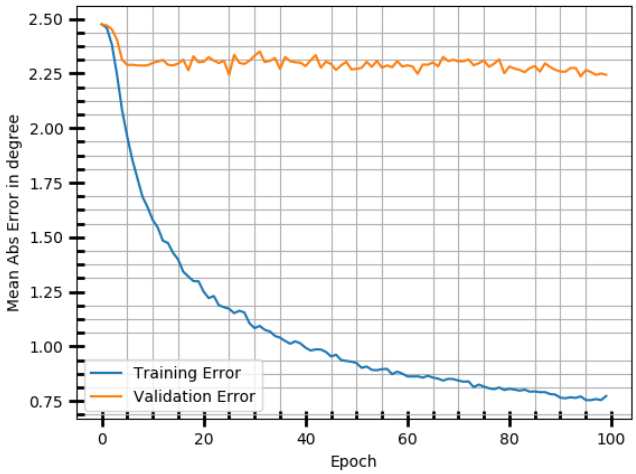


(c) Training and Validation error after each epoch for 100 epochs for row pair 127-128 i.e. row pair away from center

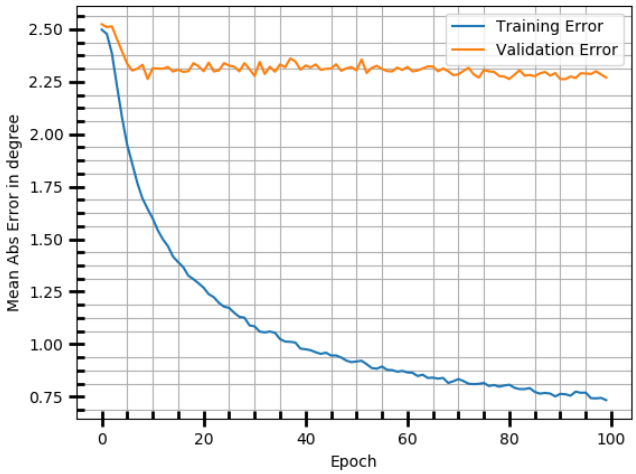
Figure 5.3: Training and validation error for noiseless data for a fully connected neural network model trained at central k-space and at extreme ends.



(a) Training and Validation error after each epoch for 100 epochs for row pair 64-65 i.e. central k-space row pair



(b) Training and Validation error after each epoch for 100 epochs for row pair 1-2 i.e. row pair away from center



(c) Training and Validation error after each epoch for 100 epochs for row pair 127-128 i.e. row pair away from center

Figure 5.4: Training and validation error for noisy data for a fully connected neural network model trained at central k-space and at extreme ends.

function in this case). We also tried scaling the complete k-space data rather than scaling the data feature-wise but in that case, also the central k-space components are significantly higher than the k-space components away from the center and hence the model does not work. One of the disadvantages of the fully connected neural network model is it does not takes advantage of the structure of data i.e. spatial relationship in nearby k-space values and considers every input individually.

In the next chapter, we will experiment with convolutional neural networks that can take advantage of the structure of data using convolutional filters.

6

Convolutional Neural Networks

In a fully connected neural network (CNN), every neuron in each layer is connected to every neuron in the previous and next layer. Due to this structure, the number of trainable parameters in the network is very high. The convolutional neural network has advantages over fully connected neural networks such as sparse interactions, parameter sharing, and equivariant representations[11]. In the next section, we discuss the CNN model, experiments and results of CNN for predicting the motion parameters in k-space.

6.0.1. Introduction to convolutional neural networks

Convolutional neural networks have similarities with fully connected neural networks (multilayer network of neurons with trainable parameters) except these networks assume that the input has some spatial structural properties. CNNs extract the features of an image using convolution and pooling operations and using these features as input to fully connected layer, assign a value or a label to the input.

CNNs are built from 3 main types of layers namely Convolutional Layer, Pooling Layer, and Fully-Connected Layer. Convolution layer consists of a number of small convolution filters that are passed over the input that learn the structural features of an image. The pooling layer is used to reduce the number of parameters by selecting the most important features. The pooling layer helps to reduce the number of trainable parameters and hence reducing the risk of overfitting. Fully connected layers flatten these 2-D inputs into a vector and then assign a label or a value to each input sample. Initial convolutional layers extract the low-level features like edge, color, gradient while as we go deep, the next convolutional layers give high-level features showing the wholesome understanding of the image/input [1].

Model:
After multiple experiments on training data, a CNN model with 3 convolution-pooling-dropout layers followed by a dense layer and then a single unit output layer was used. 64, 32 , 16 filters of size 2×2 were used in convolution layer respectively with stride of 1×1. L2 regularization parameter (λ) was 0.005 and dropout rate was 0.1 (10%). The model was trained for 100 epochs with ”mean absolute error” as a loss function and ”adam” optimizer. Using 10000 training samples and 1000 validation samples (simple validation was used as cross-validation is computationally expensive and data generated is uniformly sampled data) a CNN model with different set of hyperparameters was trained each time and the model with the best performance on validation data was selected.For training and testing the model, we use python 3.6.8 with TensorFlow version 1.7.0. NVIDIA TITAN X_p (12 GB) GPU was used to train and test the models.

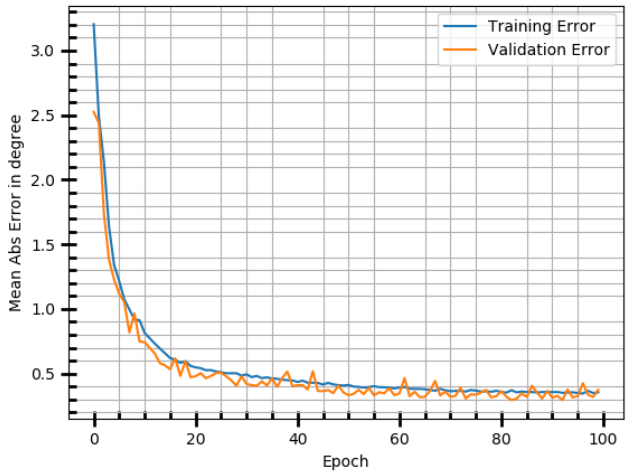
6.1. Experiments with Convolutional Neural Network

6.1.1. Experiment 1

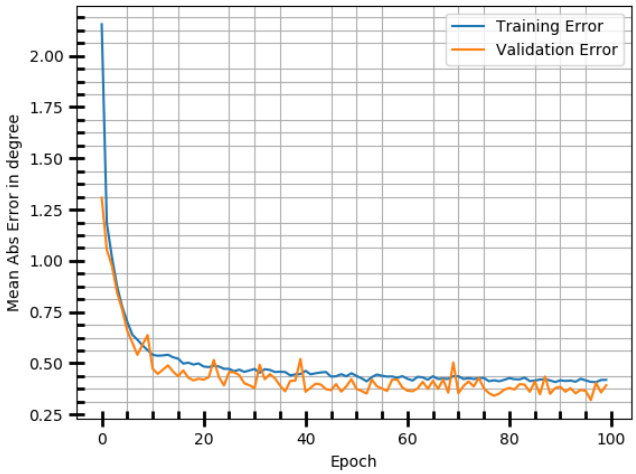
In the first experiment, we are trying to find out how accurate model can learn in the noiseless and noisy scenario. The dataset used is similar to the dataset used in the experiments in sections 4 and 5 i.e. simple rectangle centered at the origin with random rotation. For each pair of consecutive k-space rows, we train a separate CNN model. Each input for this model is of size 2×128 (two consecutive rows), with a single output θ (rotation angle between two input rows). For each pair of rows, 10000 such samples were created for training. The validation set of 1000 samples was created separately for each pair of rows. For each pair of rows, separate CNN models were trained, effectively training 127 CNN models. In figure 6.1 and 6.2 it can be observed that, model is able to learn the data well and is able to predict the rotation motion parameter with the validation error (MAE) 0.3^o in central k-space and 0.4^o in k-space region away from center for 1000 validation samples each in noiseless case. This is the best error rate we have achieved (compared to Gaussian Processes and FNNs) and we believe that this error rate is sufficient for a good reconstruction (will be discussed in 6.1.5).

Now, in the presence of noise (SNR=50), the results of training the models can be seen in figure 6.2. The validation error (MAE) for the model trained on the central row pair is lower (0.5^o) than the error away from the center (0.75^o). Here, the effect of noise is more for a model trained on data away from the center of k-space. The reason is the same, as mentioned in the previous section, that the low-frequency contribution is higher than high frequencies in k-space and hence the effect of noise is more on high-frequency data i.e. data away from the k-space center.

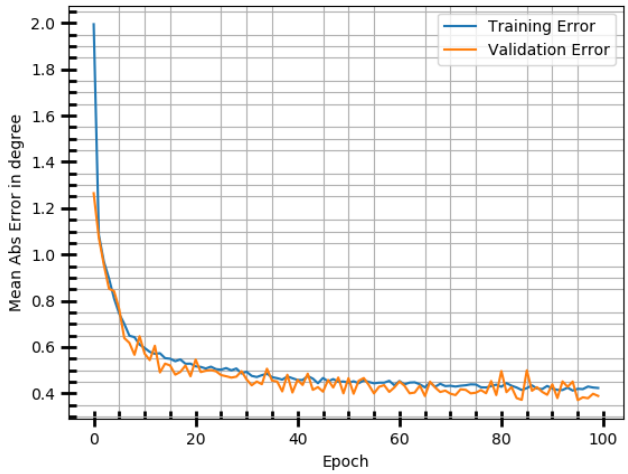
After this experiment, now we will check the effect of the amount of training data on validation error.



(a) Training and Validation error after each epoch for 100 epochs for central k-space row pair

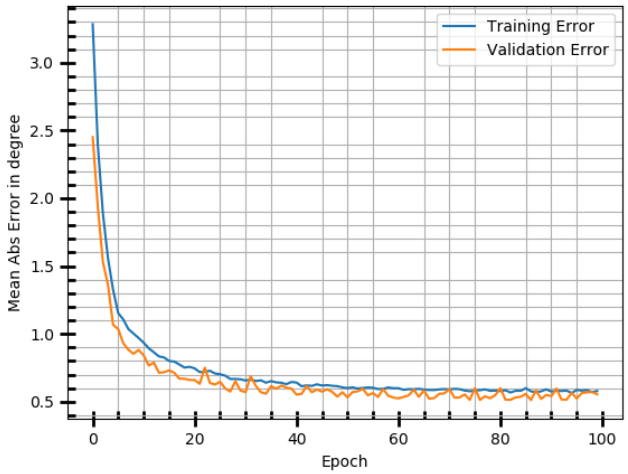


(b) Training and Validation error after each epoch for 100 epochs for row pair 1-2 i.e. row pair away from center

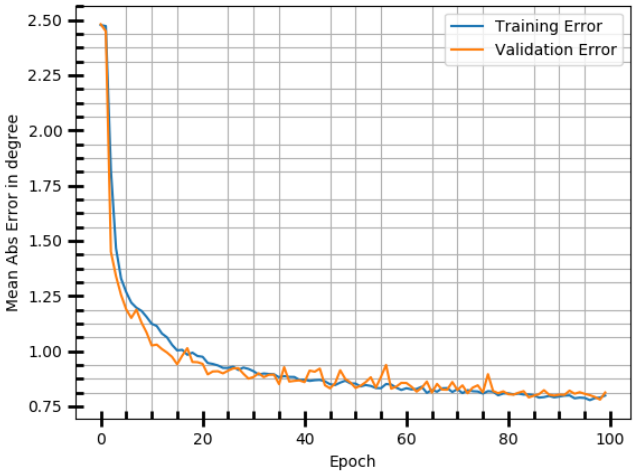


(c) Training and Validation error after each epoch for 100 epochs for row pair 127-128 i.e. row pair away from center

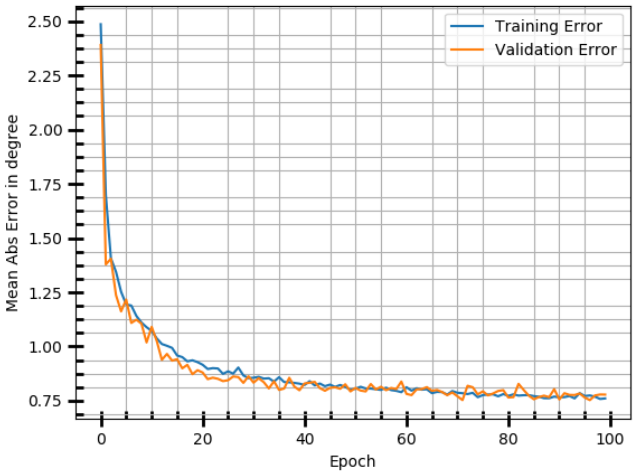
Figure 6.1: Training and validation error for noiseless data for model trained at central k-space and at extreme ends.



(a) Training and Validation error after each epoch for 100 epochs for central k-space row pair



(b) Training and Validation error after each epoch for 100 epochs for row pair 1-2 i.e. row pair away from center



(c) Training and Validation error after each epoch for 100 epochs for row pair 127-128 i.e. row pair away from center

Figure 6.2: Training and validation error for noisy data (SNR=50) for model trained at central k-space and at extreme ends.

6.1.2. Experiment 2

For this experiment, we check the effect of the amount of training data on validation error. We train a CNN model on central k-space data (row pair 64-65) in the presence of noise (SNR=50) by increasing the number of training samples from 1000 to 10000.

As we increase the number of training data samples, validation error for the CNN model reduces as shown in figure 6.3. We decided to use 10000 training samples for each model by considering the limitation of system storage capabilities (127×50 MB storage for the data of one experiment) and training time (127×80 seconds for training all models of one experiment).

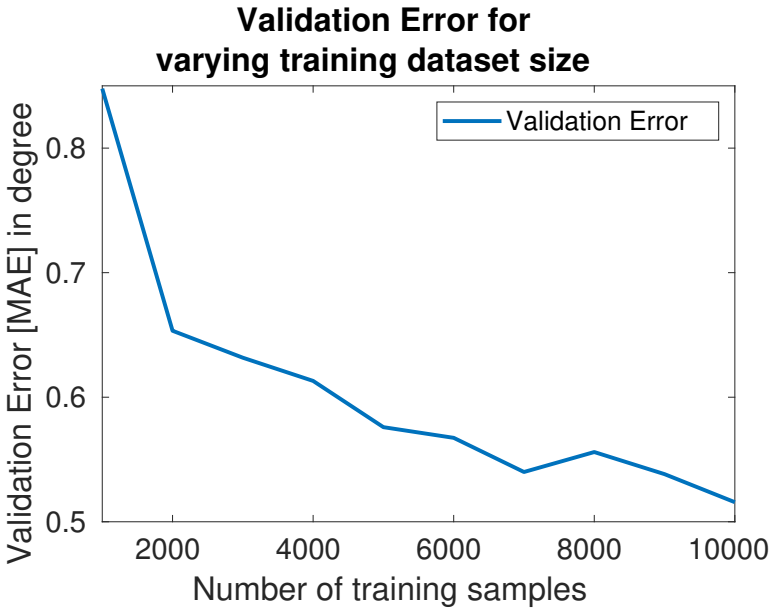


Figure 6.3: Mean Absolute error for 1000 validation samples for varying size of training dataset

6.1.3. Experiment 3

From the previous experiments, it was concluded that the CNN model works better as we start increasing the number of training data samples. The variation in A_x and A_y in the previous dataset was 10-100. Now, we decided to limit this variation to 18-22 for A_x and to 45-55 for A_y with an initial angle of rotation between -45^o to 45^o and angle between consecutive rows was limited in the range of $\pm 5^o$. We reduce the variability of the object as the real dataset (brain images) will have fewer variations in the size of the object.

In this experiment, we will train the model on noiseless data. For each pair of rows, we use a separate model ending up with 127 models for a k-space image of 128×128. As we know, more information is present at the center of k-space data as low frequencies contribute more to k-space data than higher frequencies, we first check the performance of CNN for central lines. After evaluating performance at the central k-space rows, we train a similar model on pair of rows away from the center and then evaluate the performance.

To obtain the graph below, we trained 127 models on 127 training datasets for each pair of rows having 10000 samples and validated on 1000 samples. In Figure 6.4, on Y-axis we have a mean absolute error in degree, for 1000 validation samples for 127 models each

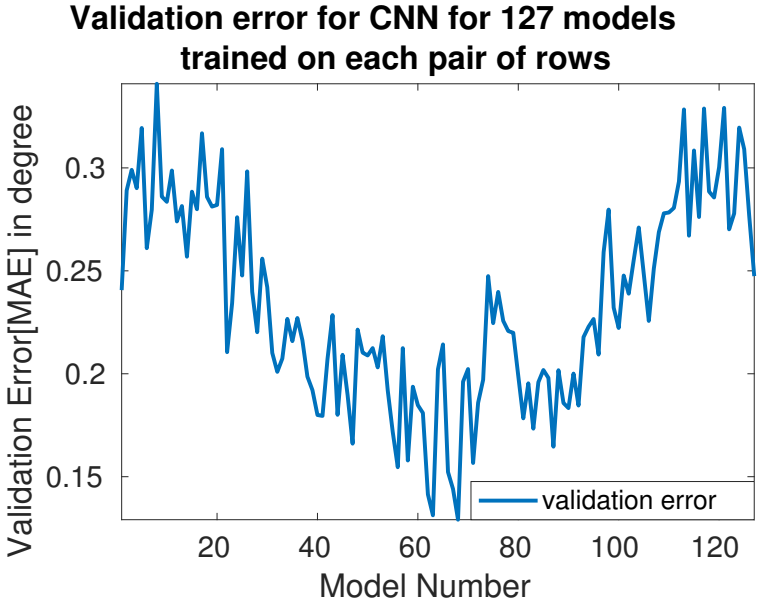


Figure 6.4: Mean Absolute error for 1000 validation samples for each model

trained on 10000 samples of one particular pair of rows. For the rows away from the center, a validation error is around 0.28^o whereas for the rows near to the center, the validation error is around 0.18^o. The reduction in validation error compared to the previous experiment is because of the reduction in the variability of object dimensions. Now, in the next experiment, we try to find the effect of noise on model accuracy trained on similar dataset as this experiment with added noise.

6.1.4. Experiment 4

In this experiment, model architecture, number of training and validation samples is the same as a previous experiment but now, we train the models on noisy data (SNR=50).
Now, in Figure 6.5, when we train the CNN models on noisy data (SNR=50), it was observed that the MAE is more than in the case of noiseless data. The effect of noise on validation error is more for rows away from the center. The error reduces to 0.3 for the central k-space rows from 0.7 for the k-space rows away from the center.

6.1.5. Experiment 5

After performing the previous experiments, it was clear that the effect of noise reduces the performance of CNN models. This hampers the performance more for the models trained on data away from the k-space center.

Now, in the next experiment, using the CNN models which were trained in the previous experiment (on noisy data), we try to correct the motion corrupted k-space data to reconstruct the object image. The noise level we used for training and testing is 50 SNR as MRI data has SNR levels around 50. This experiment aims to check how good is the error rate of rotation angle predictions we have achieved in the previous experiments for reconstructing the object using motion corrupted data. For this, now we try to test the prediction accuracy

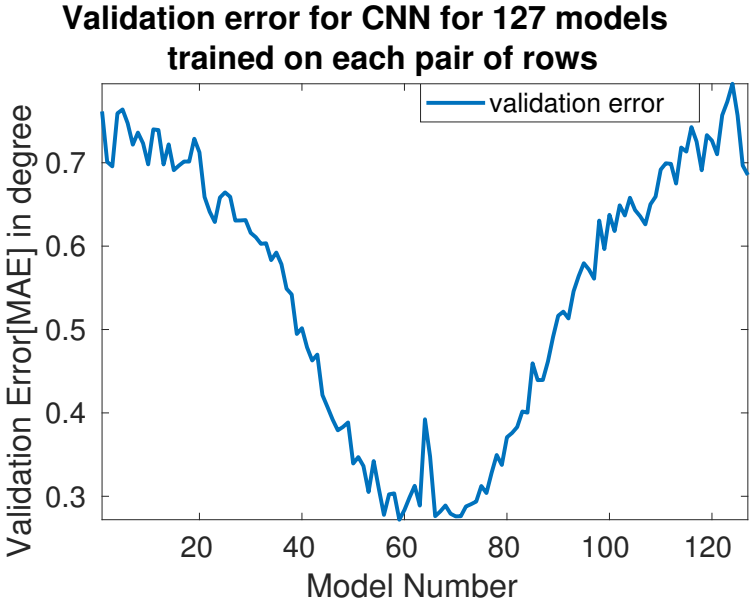


Figure 6.5: Mean Absolute error for 1000 validation samples for each model for data with SNR 50

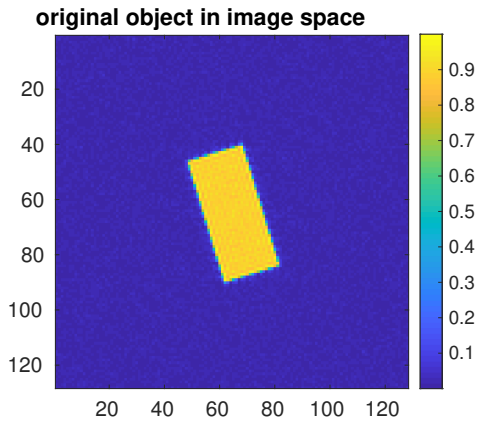
of data which is corrupted by continuous rotation motion. After predicting these values, we try to reconstruct the data to reduce motion artifacts by using iterative non-uniform Fourier transform[10]. First, we experimented with rotation parameter which is parabolic in nature (Figure 6.6e). In figure 6.6 you can observe the raw k-space data, predicted angles between rows and reconstructed object image using predicted angles and iterative NUFFT toolbox[10].

In figure 6.6b and 6.6c, the raw k-space data and reconstructed object using IFFT is shown. At the central part of k-space, the signal with higher amplitude is present showing dominance of lower frequencies in raw data. Due to rotation motion occurred between rows 40-60, the ghosting effect can be seen clearly in figure 6.6c. Now, using the models trained in the previous experiment, we try to predict the rotation parameter θ for all consecutive row pairs. In figure 6.6d, predicted and actual θ values are shown. Here, the predictions near the center are very good confirming the model behavior in figure 6.5. The predictions away from the center are less accurate due to the effect of noise and lower amplitude of high frequencies in the raw data. In figure 6.6e, you can observe the behavior of the rotation angle is parabolic for both predicted and actual values. Each model is trained separately and each test data row pair is fed separately to each model. So each model predicts the rotation angle separately and the resulting predictions for consecutive row angles are not smooth. Also, the models have some error in the prediction, hence the predictions for angles in consecutive rows are more fluctuating. We use low pass filtering in cumulative predicted values to avoid these sudden changes as we assume the nature of motion is smooth. We use the Matlab lowpass filter function with a normalized bandpass frequency of 0.3 (decided by multiple experiments) to filter the signal of row-wise cumulative predictions of rotation angle values. Using these low pass filtered values, we then predict the new co-ordinates of scanned points as shown in figure 6.6f. Using iterative NUFFT[10], the reconstructed object image is shown in figure 6.6g. The structural similarity matrix (SSIM) values of corrupted-original and reconstructed-original image pairs are 0.7608 and 0.7803 respectively. The significant improvement in the object image can be seen after reconstruction with motion effect correction. The reconstructed objects look a bit rotated as a whole, might be due to the difference in the cumulative value of actual and predicted theta.

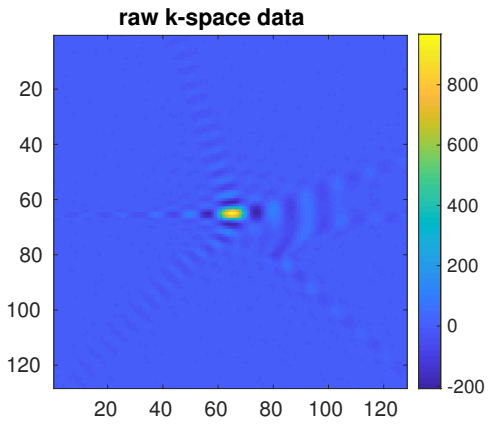
In figure 6.7, similar reconstruction is shown as the previous example but here, the rotation angle is linearly dependent on the row number. The SSIM values of corrupted-original

and reconstructed-original image pairs are 0.7089 and 0.7534 respectively

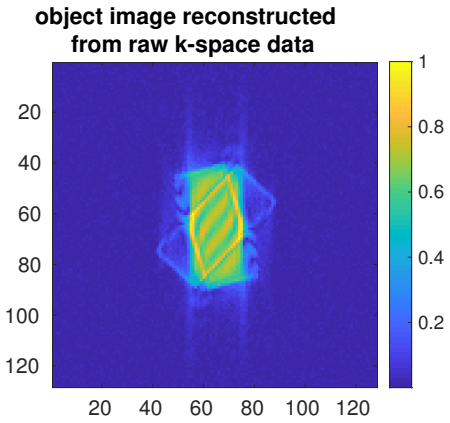
From figure 6.6 and figure 6.7 it was clear that using CNN models and iterative NUFFT, it was possible to improve the quality of reconstructed image using motion corrupted k-space data.



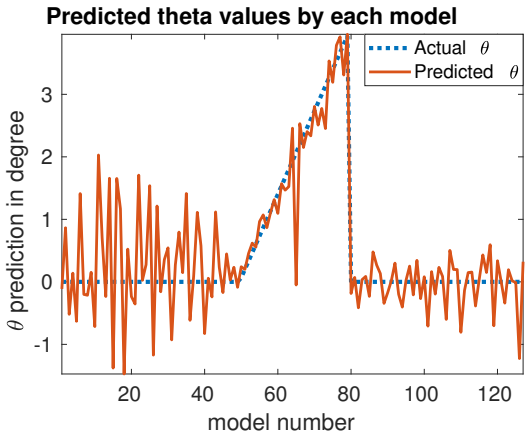
(a) original object



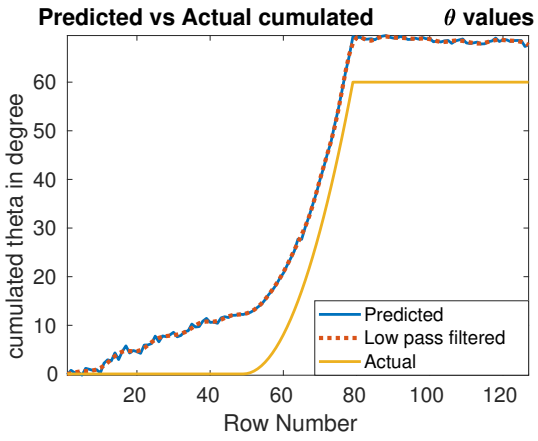
(b) k-space scanned data of moving object



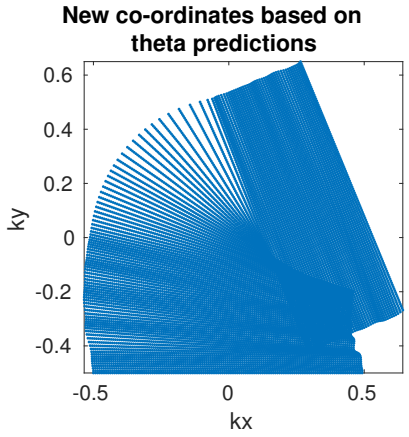
(c) reconstructed object using IFFT



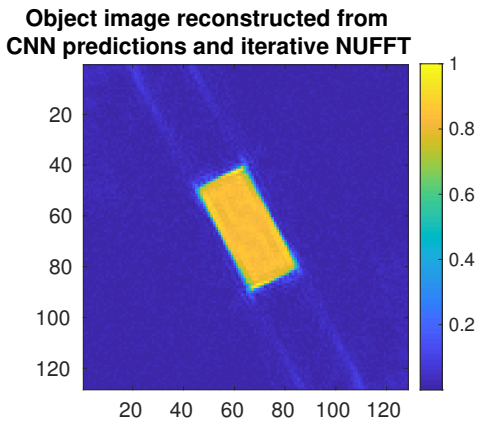
(d) Predictions by CNN models



(e) cumulated values of predicted rotation angle

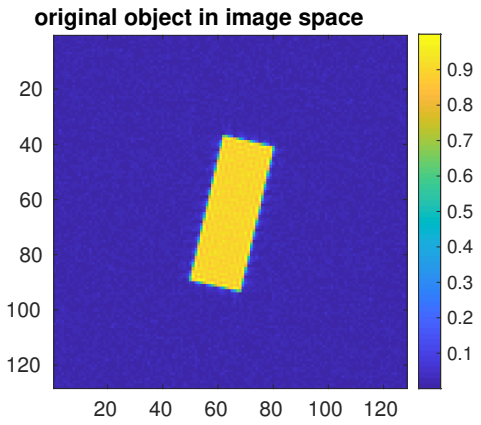


(f) New co-ordinates of scanned data based on filtered CNN predictions

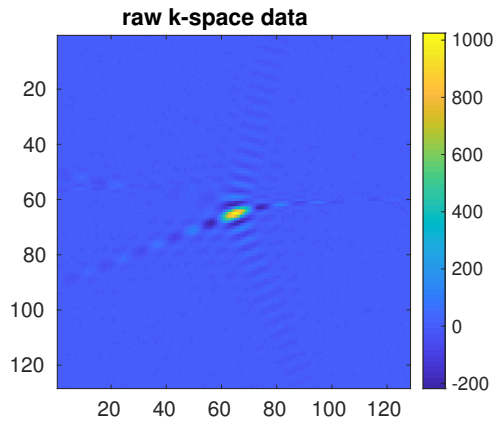


(g) reconstructed object using new co-ordinates, raw scanned data and iterative NUFFT

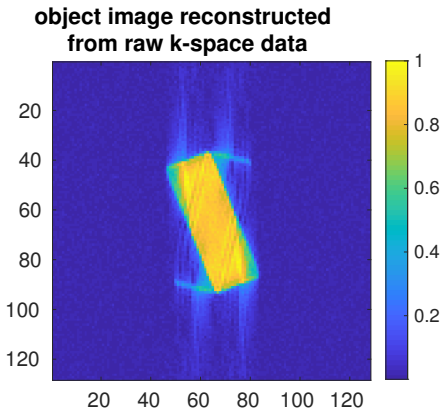
Figure 6.6: reconstructing the object image affected by rotation motion (non-linear motion)



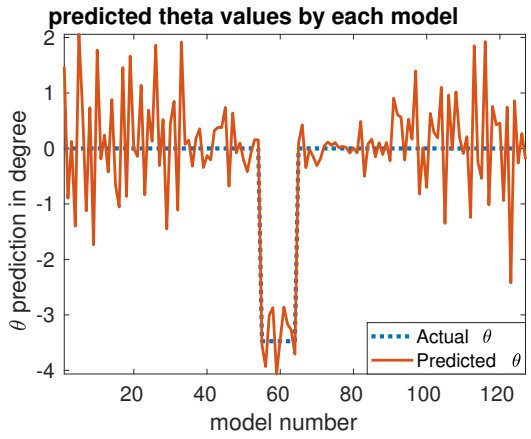
(a) original object



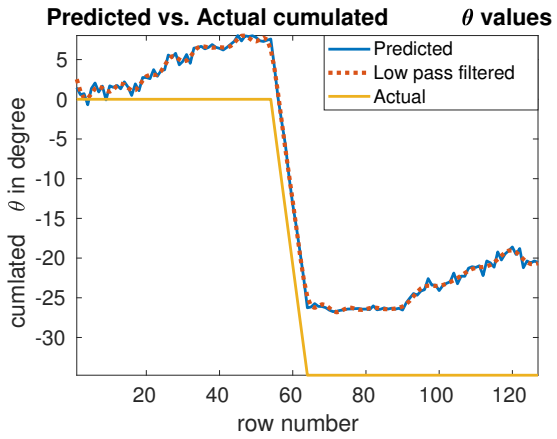
(b) k-space scanned data of moving object



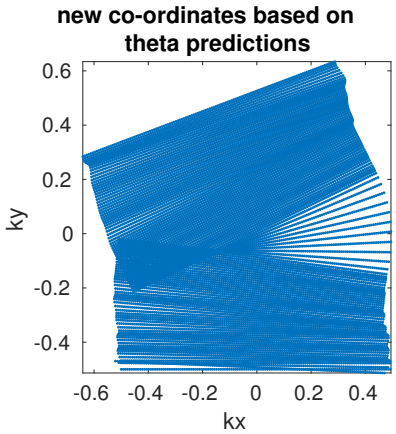
(c) reconstructed object using IFFT



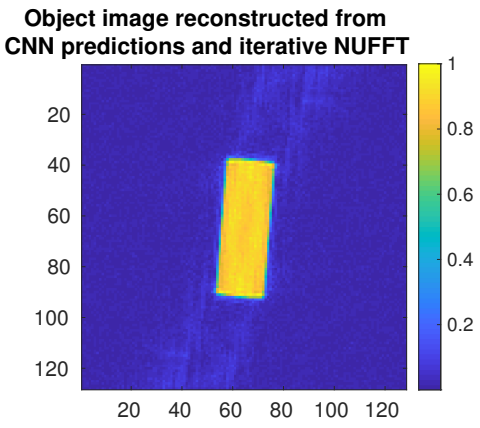
(d) Predictions by CNN models



(e) cumulated values of predicted rotation angle



(f) New co-ordinates of scanned data based on filtered CNN predictions



(g) reconstructed object using new co-ordinates, raw scanned data and iterative NUFFT

Figure 6.7: reconstructing the object image affected by rotation motion (linear motion)

7

Results on brain MRI dataset

7.1. MRI Data and experimental setup

OASIS-3 is the latest release in the Open Access Series of Imaging Studies (OASIS) [31].The dataset includes 2228 3D MR sessions of 1098 Subjects (age 42-95). Around 15 2D slices of every session are useful as training and test data for our motion correction framework since they display similar anatomic characteristics. The signal and noise distribution have an approximate mean (and standard deviation) of 156.21 (99.3883) and 4.88 (3.51) respectively. The mean and standard deviation for signal is calculated by masking the image to separate the intracranial area. The mean and standard deviation of noise is calculated by separating the background and calculating its mean and standard deviation. As the images only have absolute values (0 to 255) the noise has Rice distribution. All these images are assumed to be free of motion artifacts. To create motion affected images, we introduce simulated motion artifacts using a non-uniform Fourier transform. We choose arbitrary rotation angles (angle between consecutive k-space rows is limited to $\pm 5^o$) and rotate the k-space coordinates to mimic the effect of rotation of k-space. Subsequently, we compute NUFFT with respect to the rotated k-space coordinates to simulate the acquired data. Figure 7.1a is a 2D brain image of 128×128 and effects of simulated motion artifacts can be observed in figure 7.1b.

The CNN model architecture, the number of models and the number of training samples used is the same as chapter 6. In the synthetic dataset, the k-space data values are real whereas in the brain image dataset, k-space data values are complex. This is because the image is not symmetric and hence its k-space equivalent is complex. As the rotation in image space is equivalent to the rotation in k-space and rotation affects both amplitude and phase (unlike translation which affects only phase), using only absolute values of k-space data it is possible to determine rotation angle. By using absolute values of k-space data as inputs, CNN models with single input channel (similar to synthetic dataset) can be used for training and testing the brain image dataset.

The total images extracted from the OASIS [31] dataset were 33405. Out of these images, 10000 images were used for training and 1000 images used for validation. Remaining 22405 were used for testing and reconstruction.

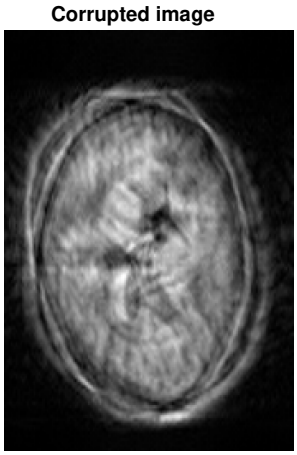
7.2. Results

The validation error for central k-space rows was around 0.3^o and for k-space data away from the center, the validation error was in the range of $0.7^o - 1.4^o$. The results of reconstructed

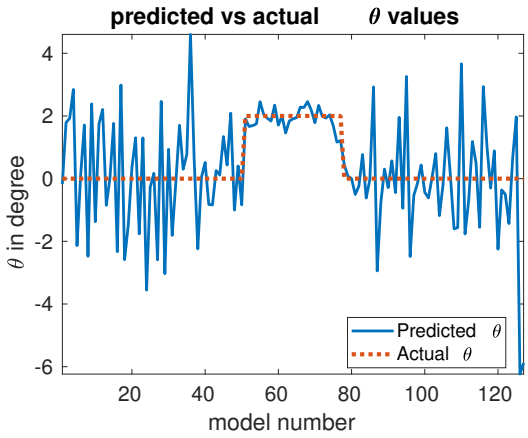
sample brain image from test dataset are as shown in figure 7.1.

In figure 7.1c, predictions of rotation parameters can be observed. At central k-space, these predictions have better accuracy than the prediction away from the center. In figure 7.1d, it can be observed that the predicted cumulated rotation angle values are close to the actual cumulated rotation angle values. To reduce the sudden changes in cumulated angle, we filter the predicted cumulated angle values as shown in 7.1d. The brain image was reconstructed using k-space data, new co-ordinates based on CNN θ predictions and iterative NUFFT as observed in figure 7.1f. The structural similarity index (SSIM) is 0.3251 for corrupted brain image-actual brain image whereas SSIM for reconstructed brain image -actual brain image is improved to 0.7325.

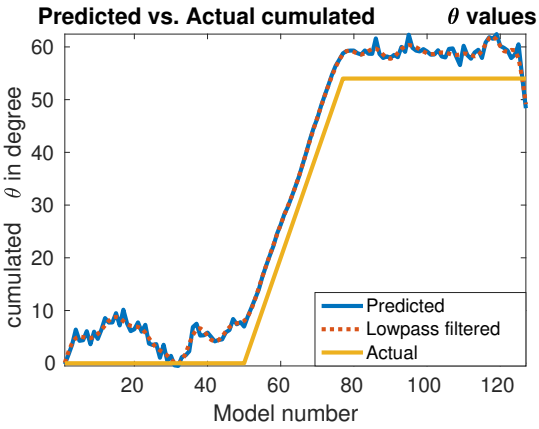
(a) brain image



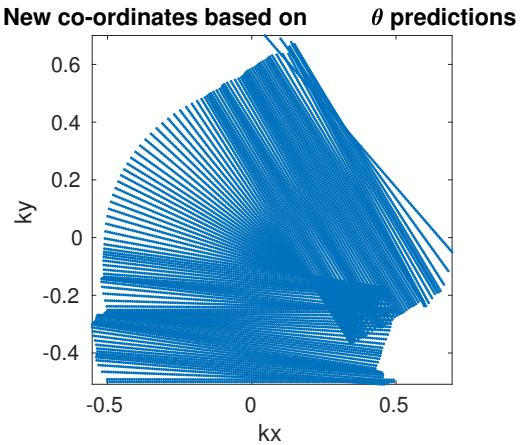
(b) reconstructed brain image using IFFT



(c) Predictions by CNN models



(d) Cumulated values of predicted rotation angle



(e) New co-ordinates of scanned data based on filtered CNN predictions

(f) reconstructed brain image using new co-ordinates, raw scanned data and iterative NUFFT

Figure 7.1: reconstructing the brain image affected by rotation motion(linear motion)

8

Conclusion, Discussion and Future Work

8.1. Discussion

In the previous chapters, experiments performed to predict motion parameters using non-linear least squares, Gaussian Processes, Fully Connected Neural Networks and, Convolutional Neural Networks were discussed. In this chapter, the results of all the experiments are analyzed and then the advantages and shortcomings of the methods implemented are discussed with possible future work.

To estimate the motion parameters, first, the object geometry is assumed to be known. Non-linear least squares is one of the simplest and most widely used method hence we start our experiments with this method. In section 3.1.1, using non-linear least squares, the rotation angle of an object was estimated when object geometry is known. From figure 3.2, it can be observed that the objective function has a clear minimum and hence can be minimized. This is because the model is simple (rectangle object) and the system is noiseless. Also, rotation is a global parameter that affects more at the lower frequencies than the higher frequencies. As all the data ($N \times N$) is used to predict a single rotation angle, the low-frequency data was also being used for prediction of the rotation angle. From this, it was concluded that, if object geometry is known and the object has a well-defined k-space function (simple setup) using non-linear least squares, the angle of rotation can be estimated from scanned k-space data.

Instead of estimating a single rotation angle value for a complete object, in the next experiment, the rotation angle for each row of k-space was estimated.In figure 3.3,the objective functions for central k-space rows have clear minima unlike objective functions away from center. In figure 3.1 it can be seen that the low-frequency components present in the image contribute more to the k-space than the high-frequency components. Also, as discussed earlier, the rotation being a global parameter affects the low frequencies more than the high frequencies. Hence predicting the rotation parameter is easier using low-frequency data. From figure 3.3, we concluded that minimizing the objective function is easier at central k-space. Due to multiple local minima for objective function away from central k-space, it is difficult to minimize this function with random initialization. As it was assumed that the motion is smooth and scanning time for each row is in order of ms, the angle between consecutive rows was limited in the range of $\pm 5^{o}$. Hence, the least-squares problem was minimized at the central k-space row pair with random initialization first and then using that prediction the minimization of the objective function for the pair of rows further away from the center was initialized. In figure 3.6, when there is noise in the k-space data, as the noise level increases, the accuracy of prediction reduces for the k-space rows away from the center. As we discussed earlier, low-frequency components are at the center of k-space and have a higher contribution, the effect of noise on low-frequency k-space data is smaller compared to the effect of noise on high-frequency k-space data.

When the noise in the system is increased, the prediction accuracy for the k-space data away from the center reduces. As observed in figure 3.6a, translation and rotation parameters were estimated and for lower SNR values, the prediction accuracy away from center reduces.
In the least-squares problem, it was assumed that the geometry of the object and its dimensions are known. In a practical scenario, this is hardly the case. We started with the least-squares problem as it is the most commonly used and simple method for regression. From the prediction error of least squares method in presence of noise and when motion parameter is changing fast and the fact that it needs the geometry of object completely known (model-driven), it was decided to explore data-driven regression methods.

Gaussian process regression was the second method used to estimate motion parameters. In equation (4.16) for training the hyperparameters, the inverse of the covariance matrix is to be calculated. For more than 3000 training points, calculating this matrix inverse is computationally expensive and hence the number of training points was limited to 3000. This is one of the big disadvantages of the Gaussian processes regression which limits its use for a large amount of training data. For the squared exponential kernel, the length-scale parameter decides the relationship between multiple input samples. After minimizing the marginal negative log-likelihood, the length-scale parameter goes to a very small value indicating failure of the model to map the correlation between input samples. Due to this, the model treats each input sample as noise and even if the training error is minimum, the model gives poor generalization performance with a very high validation error.

For learning the hyperparameters, the marginal negative log-likelihood is minimized. For this minimization, one of the critical aspects is the initialization of these hyperparameters. In figure 4.2, it can be observed that the domain is non-convex and has multiple local minima. If we use random initialization, there is a risk of getting stuck in local minima and it is computationally expensive to perform grid search for good initialization.

The poor validation results are obtained for a model trained on noiseless central k-space data and hence it was assumed that the model will also fail for noisy data as well as for the data away from the central k-space. This behavior of the model might be due to the high variance and complexity in the training data. If we wish to use more training samples we can either use better hardware or use faster and less computationally expensive training methods for training the Gaussian process.

Now, we move on to more complex and expressive models than Gaussian processes and opted for Artificial Neural Networks. We experimented with Fully Connected Neural Networks (FNN) and from figures, 5.3 and 5.4 it was concluded that FNN works well with central k-space data both in noisy and noiseless case but with k-space data away from the center, the value of validation error is high. In figures 5.3b and 5.3c, it can observed that though the validation error reduces after each epoch, it is still higher than training error and stabilizes around 100 epochs. This indicates that the FNN model fails to establish the relationship between inputs and targets for the k-space data away from the center. In figures 5.4b and 5.4c when noise is added in the systems, after first 5 epochs, the model trained on k-space data away from center fails to improve the validation error performance. One of the reasons behind this performance might be, the fully connected neural networks treat each input separately and do not take advantage of spatial relationships in the data. As the data, in this case, has a lower contribution of high frequencies, the information it contains is not sufficient for FNNs to derive the relationship between inputs and targets with a given amount of training data. Also, rotation is a global parameter and it affects the local frequencies as they determine the global properties of the image, and hence it is easier to predict the rotation angle with low-frequency values than higher frequencies. We also tried using scaled data (both feature scaling and image scaling) but the same issue is faced for FNNs.

CNNs use the spatial relation by means by convolutional filters and hence we use CNNs for the next experiment. In figures 6.1, 6.2, 6.4 and 6.5 it can be observed that though the validation error for the model trained on k-space data away from center is larger, it is still comparable to validation error for model trained on central k-space. Now we want to verify that how good is error rate (MAE) of $0.3^o - 0.7^o$ for target values ranging in $\pm 5^o$. This was done by reconstructing the image using predictions of CNN models. In figures 6.6 and 6.7 the reconstruction results can be seen. The reconstruction results show a clear improvement in the quality of the image from the image reconstructed by using a simple inverse Fourier transform. In figures 6.6 and 6.7, when the predictions are compared, the difference in actual and predicted values is observed. The variance in the data and noise (SNR=50) might be the reasons behind this difference. The filtering of the rotation angle's temporal behavior is necessary as we assume the motion to be smooth. In figures 6.6e and 6.7e it can be observed that the predicted rotation motion parameter curve follows closely the actual rotation motion parameter curve. Even if we predict the motion exactly, we will not be able to reconstruct the object perfectly as due to non-uniform k-space sampling as an effect of motion, some spatial frequency components are lost.

We performed some preliminary experiments with a more complex dataset of brain images and obtained some promising results as shown in section 7. The performance of the CNN models on the brain dataset is better for central k-space data (low frequencies) than k-space data away from the center (high frequencies) similar to that of the synthetic dataset. The significant improvement in the quality of reconstructed brain image indicates that the CNN models give good predictions for brain MRI dataset as well and can be used for reconstruction of brain images affected by rotation motion.

From the four methods we implemented, it was concluded that CNNs work better than the other three methods namely non-linear least squares, Gaussian processes and FNNs for predicting motion in k-space.

8.2. Conclusion and Outlook

With the methods discussed, we tried to detect, estimate and correct the effects of motion in the k-space of MRI data. Using CNN models for prediction of rotation parameters and iterative NUFFT for reconstruction, we get improved quality of reconstructed image for synthetic data consisting of simple rectangles. Least squares, Gaussian processes, and FNNs fail to predict motion parameters accurately for this synthetic dataset as explained in the discussion section. The CNNs show some potential for k-space motion parameters prediction for the simple setup as well as the complex dataset of brain images.

The average time for training a CNN model for a pair of k-space lines is 80 seconds and average prediction time using a pre-trained CNN model for one pair of rows is 0.05 seconds. As the prediction time is short and the CNN model predicts the rotation angle for each pair of row independently from other k-space data, it was concluded that the proposed method has some potential to detect the motion parameters in real-time.

The method we implemented was the first step towards correcting the motion artifacts directly from k-space data. Using the selected approach, good reconstruction results were obtained for the synthetic dataset and some promising preliminary results were obtained for brain images.

Suggested future work:

1. In the current implementation, we tried to predict rotation angle in k-space using CNN

models and obtain good reconstructions on a synthetic dataset. The input data we used to perform the experiments of CNNs was real. As a next step, we also want to predict translation motion in k-space. After introducing translation motion, the object image is not symmetric anymore and k-space data is now complex. To predict translation motion, we have to use this complex k-space data as input and hence need to use two input channels for CNN models with three outputs (rotation angle and x-y translation motion values). In the current set up, we are predicting a single parameter with some prediction error. For three parameters, the effect of the prediction error will be higher and needs to be analyzed further.

2. Many clinical MRI scans acquire 3D data. If we want to use a similar approach for 3D MRI k-space data$[N \times N \times N]$, we need to have $[N^2 - 1]$ models of CNN. Training and storing these many models is difficult even if they have a few thousand parameters. If we want to reduce the number of models, further research is necessary to find out if we can use the same model for multiple k-space row pairs for predicting motion parameters. In the case of 3D MRI data, three rotation motion parameters and three translation motion parameters are to be predicted for each row pair. The number of parameters to be predicted for the data of $[N \times N \times N]$ will be $6 \times (N^2 - 1)$. In case of 3D data, for each row, the number of inputs will be the same $(2 \times N)$ and the number of motion parameters to predicted will be higher (six parameters for each row pair). The possibility of error will be higher in this case and needs further research.

3. With the current implementation, we train each model separately and then use them for predicting the rotation angle. As we have observed, the models trained on the central k-space data have better performance than the models trained on k-space data away from the central k-space. All the models we use for prediction are independent and hence they do not use this information. Introducing dependency in predictions of consecutive models and combining the information of motion parameters predictions of low-frequency k-space data for predicting motion parameters of high-frequency k-space data can improve the results. One of the methods which could be used for this purpose is to put smoothness constraints on predictions of models. This can be done by training a model on multiple rows instead of a single row pair and putting a smoothness constraint on the predictions of this model. In the current setup, we are training a model on the input size of $2 \times N$ to predict the single output. Training a model on $m \times N$ input to predict $m - 1$ outputs with a smoothness constraint on predictions can help to improve the results and can be researched further.

4. The current approach can be be used to predict the motion parameters in real-time. This can be used to detect the motion parameters and if the motion predicted is very high, the re-scan could be started instantly without the necessity of reconstruction. This could be done by setting a threshold on predicted cumulative motion parameters. This application could be explored further to find the limits on motion parameters for recommending re-scan.

Bibliography

[1] URL http://cs231n.github.io/convolutional-networks/.

[2] Christian Baumgartner Wenjia Bai Daniel Rueckert Andreas Maier Benedikt Lorch, Ghislain Vaillant. Automated detection of motion artefacts in mr imaging using decision forests. *Journal of Medical Engineering*, 2017. URL https://doi.org/10.1155/2017/4501647.

[3] Gilardi MC et al. Bettinardi V, Scardaoni R. Head holder for pet, ct, and mr studies. *J Comput Assist Tomogr*, 15:886–892, 1991.

[4] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[5] C. K. I. Williams C. E. Rasmussen. *Gaussian Processes for Machine Learning*. the MIT Press, 2006.

[6] Hannes Nickisch Carl Edward Rasmussen. Gaussian process regression and classification toolbox version 4.2. URL http://gaussianprocess.org/gpml/code.

[7] V Edward, Christian Windischberger, R Cunnington, M Erdler, Rupert Lanzenberger, D Mayer, Walter Endl, and R Beisteiner. Quantification of fmri artifact reduction by a novel plaster cast head holder. *Human brain mapping*, 11:207–13, 12 2000. doi: 10.1002/1097-0193(200011)11:33.0.CO;2-J.

[8] Allen D. Elster. Geography of k-space, . URL http://mriquestions.com/parts-of-k-space.html.

[9] Allen D. Elster. Filling k-space, . URL http://mriquestions.com/data-for-k-space.html.

[10] Jeffrey A Fessler. Michigan image reconstruction toolbox. URL http://web.eecs.umich.edu/~fessler/irt/fessler.tgz.

[11] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[12] Jürgen Hennig, A Nauerth, and H Friedburg. Rare imaging: A fast imaging method for clinical mr. *Magnetic resonance in medicine : official journal of the Society of Magnetic Resonance in Medicine / Society of Magnetic Resonance in Medicine*, 3:823–33, 12 1986. doi: 10.1002/mrm.1910030602.

[13] Padmanabhan Menon James G. Pipe, Ergun Ahunbay. Effects of interleaf order for spiral mri of dynamic processes. *Magnetic Resonance in Medicine*, 41:417–422, 1999.

[14] Pipe JG. An optimized center-out k-space trajectory for multishot mri: comparison with spiral and projection reconstruction. *Magn Reson Med*, 42:714–720, 1999.

[15] Hosung Kim, Wenlu Zhang, Meng Law, Lu Zhao, Arthur Toga, and Ben Duffy. Retrospective correction of motion artifact affected structural mri images using deep learning of simulated motion. 05 2018.

[16] Thomas Küstner, Karim Armanious, Jiahuan Yang, Bin Yang, Fritz Schick, and Sergios Gatidis. Retrospective correction of motion-affected mr images using deep learning frameworks. *Magnetic Resonance in Medicine*, 82(4):1527–1540, 2019. doi: 10.1002/mrm.27783. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/mrm.27783.

[17] Benedikt Lorch, Ghislain Vaillant, Christian F. Baumgartner, Wenjia Bai, Daniel Rueckert, and Andreas K. Maier. Automated detection of motion artefacts in mr imaging using decision forests. In *Journal of medical engineering*, 2017.

[18] Julian R. Maclaren. Motion detection and correction in magnetic resonance imaging. URL https://core.ac.uk/download/pdf/35458658.pdf.

[19] Shobha Malviya, Terri Voepel-Lewis, Odd Eldevik, David Rockwell, J Wong, and AR Tait. Sedation and general anaesthesia in children undergoing mri and ct: Adverse events and outcomes. *British journal of anaesthesia*, 84:743–8, 06 2000. doi: 10.1093/oxfordjournals.bja.a013586.

[20] P Mansfield. Multiplanar image formation using nmr spin echos. *Journal of Physics C: Solid State Physics*, 10:L55, 02 2001. doi: 10.1088/0022-3719/10/3/004.

[21] Michael Herbst Maxim Zaitsev, Julian. Maclaren. Motion artefacts in mri: a complex problem with many partial solutions. *J Magn Reson Imaging*, 42(4):887–901, 2015. doi: 10.1002/jmri.24850.

[22] V. Menon, K. O. Lim, J. H. Anderson, J. Johnson, and A. Pfefferbaum. Design and efficacy of a head-coil bite bar for reducing movement-related artifacts during functional mri scanning. *Behavior Research Methods, Instruments, & Computers*, 29(4):589–594, Dec 1997. ISSN 1532-5970. doi: 10.3758/BF03210613. URL https://doi.org/10.3758/BF03210613.

[23] Kevin P. Murphy. Machine learning: A probabilistic perspective. In *Adaptive computation and machine learning series*, pages 575–576. MIT Press, 2012.

[24] Kamlesh Pawar, Zhaolin Chen, N. Shah, and Gary Egan. Moconet: Motion correction in 3d mprage images using a convolutional neural network approach, 07 2018.

[25] James G. Pipe. Motion correction with propeller mri: application to head motion and free-breathing cardiac imaging. *Magnetic resonance in medicine*, 42 5:963–9, 1999.

[26] P. Paclik E. Pekalska D. de Ridder D.M.J. Tax S. Verzakov R.P.W. Duin, P. Juszczak. Prtools4 a matlab toolbox for pattern recognition. URL http://prtools.tudelft.nl/files/prtools4.2.5.zip.

[27] Gordon Sarty. Single trajectory radial (star) imaging. *Magnetic resonance in medicine : official journal of the Society of Magnetic Resonance in Medicine / Society of Magnetic Resonance in Medicine*, 51:445–451, 03 2004. doi: 10.1002/mrm.20001.

[28] Alessandro Sbrizzi. Principles of magnetic resonance imaging. URL http://www.radiotherapie.nl/asbrizzi/webpage/isc_files/slides2.pdf.

[29] Yunhong Shu, Stephen Riederer, and Matt Bernstein. Three-dimensional mri with an undersampled spherical shells trajectory. *Magnetic resonance in medicine : official journal of the Society of Magnetic Resonance in Medicine / Society of Magnetic Resonance in Medicine*, 56:553–62, 09 2006. doi: 10.1002/mrm.20977.

[30] Keith Slifer, M Cataldo, M Cataldo, A Llorente, and Arlene Gerson. Behavior analysis of motion control for pediatric neuroimaging. *Journal of applied behavior analysis*, 26: 469–70, 02 1993. doi: 10.1901/jaba.1993.26-469.

[31] J. Morris T. Benzinger, D. Marcus. Oasis-3: Longitudinal neuroimaging, clinical, and cognitive dataset for normal aging and alzheimer's disease. URL https://www.oasis-brains.org/#about.

[32] Ghislain Vaillant, Claudia Prieto, Christoph Kolbitsch, Graeme Penney, and Tobias Schaeffter. Retrospective rigid motion correction in k-space for segmented radial mri. *IEEE transactions on medical imaging*, 33, 06 2013. doi: 10.1109/TMI.2013.2268898.

[33] Hans Chr van der Werf. Towards real-time imaging: a literature study on fast imaging by undersampling and smart reconstruction. URL https://dspace.library.uu.nl/bitstream/handle/1874/222736/literatuur.pdf?sequence=2.

[34] Ya-xiang Yuan. A review of trust region algorithms for optimization. *ICM99: Proceedings of the Fourth International Congress on Industrial and Applied Mathematics*, 09 1999.