UNIVERSITY OF COPENHAGEN

A thesis presented to the Faculty of Science in partial fulfillment of the requirements for the degree

Master of Science in Bio- and Medical Physics

(60 ECTS)

Multiparametric quantification of tissue integrity in the knee using MRI

Medical Imaging

Gidega Vijayakumar

nqj779alumni.ku.dk

Niels Bohr Institute

Supervisor: Bryan Haddock & Lise Arleth

27 Juli 2021

PREFACE

This thesis study was written for partial fulfillment of the requirements for the Master of Science degree in Bio- and Medical Physics, offered by the University of Copenhagen (UCPH) Faculty of Science in collaboration with Rigshospitalet Glostrup, Department of Clinical Physiology, Nuclear Medicine and PET.

The target group for which this thesis was written, is mainly physics students with a basic knowledge of principles in physics and with an interest for medical physics. Being that this thesis investigates a problem concerning the degenerative joint disease osteoarthritis, it could also present an interest for physiotherapists, who can treat the symptoms in the earlier stages and inhibit the development of the disease.

First of all, I would like to thank my central project supervisor, medical physics expert Bryan Haddock for not only introducing me to this thesis project, and thereby providing me the opportunity to work on this exciting and very relevant subject in nuclear medicine, but also for supporting and motivating me through the whole thesis process. It has been a great learning experience to work with the guidance of Bryan. Moreover I want to thank my UCPH-Science main supervisor, prof. Lise Arleth for all the support and advice, making sure the thesis work was heading the right direction throughout the whole process. Furthermore I am thankful for the help and advice that I received for the processing of the medical images, from the Stanford University team, Feliks Kogan, Lauren Watkins, Arjun Divyang Desai, Valentina Mazzoli. Finally I would like to thank Bryan, Linnéa Haugen, Mathilde Overgaard Lauersen, Tanne Stephanie Wiberg Larsson and Thomas Tassi Jørgensen for being volunteers for my own MRI knee loading experiment.

Gidega Vijayakumar

ABSTRACT

It is difficult to diagnose the degenerative joint disease Osteoarthritis (OA) in the early stages, as the joint will appear normal on the currently utilized diagnosis method, X-ray imaging. The pain and dysfunction in the joints, will mostly be experienced only when loading or any kind of activity is done, hence with PET/MRI imaging, the response to loading can be studied in the whole joint. The purpose of this thesis is to analyse the bone-cartilage interaction in loaded knee joints and to further investigate a detectable correlation between the response in knee cartilage and adjacent bone tissues, utilizing PET/MRI imaging. This was examined by processing a set of collected (by Stanford University) ^{18}F-NaF PET/MRI baseline and post-exercise knee scans, for 12 healthy subjects. The knee loading exercise consisted of one-legged step-up and drop-land exercise repeated 100 times. The change in response of the exercise was analysed for the different femoral cartilage regions' T_2 and $T_1\rho$ values and the adjacent subchondral bone ^{18}F-NaF uptake. A statistically significant change and thereby response of the loading in the adjacent subchondral bone tissue ^{18}F-NaF uptake was observed, but the same significant response was not achieved in terms of a change in the femoral cartilage tissue T_2 nor $T_1\rho$ values. This led to the second study of the thesis, the investigation of an optimized MRI protocol for measuring the response to loading of the knee joints in terms of a change in the T_2 value of the femoral cartilage regions. This was examined by acquiring a set of baseline and post-exercise scans for 5 healthy subjects, with a Turbo Spin Echo (TSE) MRI sequence, exercise consisting of 60 squats holding 2 kg hand weight, and with a shorter time span between executing this exercise and acquiring the post scans. Additionally, the effect of Blood Flow Restriction (BFR) was also examined. A significant change in the cartilage T_2 values and thereby response of the loading in only the knee with BFR band was measurable for this study. Therefore, it can be concluded, that by combining an optimized MRI protocol with the PET imaging, it will be possible to eventually further study, whether a correlation between cartilage and adjacent bone tissues response to loading can be detected with PET/MRI imaging.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

^{18}F-NaF [^{18}F]-sodium fluoride

AI Artificial Intelligence

ACD Annihilation Coincidence Detection

BFR Blood Flow Restriction

BMI Body Mass Index

CNN Convolutional Neural Networks

CT Computed Tomography

DESS Dual Echo Steady State

DICOM Digital Imaging and Communications in Medicine

DOSMA Deep Open-Source Medical Image Analysis

emf electromotive force

FID Free Induction Decay

FOV Field Of View

FSE Fast Spin Echo

GLME Generalized Linear Mixed-Effects

GRE Gradient Echo

IMRAD Introduction, Method, Result And Discussion

IR Inversion Recovery

MR Magnetic Resonance

MRA Magnetic Resonance Angiography

MRAC Magnetic Resonance-based Attenuation Correction

MRI Magnetic Resonance Imaging

NMR Nuclear Magnetic Resonance

OA Osteoarthritis

OAI Osteoarthritis Initiative

PET Positron Emission Tomography

RF Radio Frequency

ROI Region Of Interest

SE Spin Echo

SUV Standardized Uptake Value

TE Echo Time

TI Inversion Time

TR Repetition Time

TSE Turbo Spin Echo

Part I

INTRODUCTION

1

MOTIVATION

Globally, millions of people are affected by the degenerative joint disease Osteoarthritis (OA), causing pain, stiffness in the joints, and affecting the mobility [2]. In a normal joint, the articular cartilage acts as protection for the bone ends that rub against each other. In a joint with the chronic disease OA, the articular cartilage and the tissues of the joint gradually degrade, resulting in the loss of the smooth surface of the articular cartilage, which becomes irregular and frayed [24]. This affects the bone as the cartilage gradually disappears, failing to ensure that the bones move smoothly and painlessly in relation to each other.

OA (like other forms of arthritis) can not be cured, but depending on the stage, the right treatment can reduce symptoms and in many cases inhibit the development of the disease [25]. There are 4 stages of OA; early, mild, moderate and severe. It's difficult to make the diagnosis at an early stage, as the person will mostly not feel any symptoms and the joint will appear normal on an X-ray, which is the diagnosis method used right now [29]. The treatment in the mild and moderate stage would mainly be a combination of pain killers and exercises for the joints. But when the OA reach the severe stage, surgical treatment is often the only option, as the cartilage has completely worn away or there is very little left, which will affect the bones even more.

People diagnosed with OA, will not only be affected physically. Because of the limitations of daily living activities, they may also be affected mentally, which overall will diminish their quality of life. Aside from this, the disease is very expensive for the society; in health expenses, in social expenses and of course also in lost labor force, as they might have to either call in sick for a long period of time or even worse, retire early from the labor market [26]. Finding a method to diagnose OA in the early stages, thereby making it possible to give an appropriate treatment to prevent the degradation of the articular cartilage and the tissues of the joint, and monitor the response to this treatment, would be an essential solution for all of these problems. What this thesis will investigate, may pave the road for solving the problem of diagnosing OA in the early stages.

2

PROBLEM STATEMENT

Osteoarthritis (OA) is most commonly known to occur in hip, knees and hands, but in this thesis a step into finding a possible method to diagnose OA in the early stages of the knees will be studied. Studies have shown that OA can be understood much better, with the composition of simultaneous quantitative measurements of both bone and cartilage [8]. A suggestion from these studies is also that there is a connection between the degenerative changes and processes in the cartilage and the adjacent bone in the early stages. Finding this connection could be a step into exploring a new quantitative method to diagnose OA in the early stages, by being able to study both the bone and cartilage health simultaneously.

To study the early stages of OA, we accordingly need an optimal imaging technique that must be sensitive both to soft-tissue and bone-tissue health. Positron Emission Tomography / Magnetic Resonance Imaging (PET/MRI) imaging of the knee should be an ideal method for this study, since this imaging technique will fulfill these two requirements. Magnetic Resonance Imaging (MRI) will provide morphological information of the joint tissues with an exceptional image resolution [6]. Using MRI we have already been able to study the damage of the articular cartilage. Positron Emission Tomography (PET) will provide information of metabolic abnormalities of bone metabolism in OA [11]. Thus with PET imaging we would be able to examine remodeling in the bones, by using a PET tracer [^{18}F]-sodium fluoride (^{18}F-NaF), which is a well-established bone seeking agent [4]. Simultaneous PET and MRI scanning will then provide extensive imaging of the whole joint, including both the soft tissues and bone, which is exactly what we need to be able to study the complex disease process in OA.

In OA knees, most of the pain and dysfunction in the joints are linked to the altered mechanics [3], and is mostly experienced only when loading the joints or doing any kinds of activity. Currently used imaging techniques utilize static conditions with the person at rest, but with the PET/MRI imaging technique the whole joint function can be studied for any changes after a joint loading. Hence, the main purpose of this thesis is to study the bone-cartilage interaction in loaded knee joints, and more specifically to examine the following hypothesis: ***With a loading of the knee joints, the cartilage tissues deform and the bone tissues react very quickly to adapt to this load, consequently there must be a correlation between the change in these two kinds of tissues, which could be captured using PET/MRI imaging.*** Detecting a functional response or an area where the functions are affected, even when the structure has not changed (the structure of the OA diseased cartilage and bone may first change after several months or even years), could be an opportunity to pickup a joint dysfunction, that may be a new indicator for diagnosing early stages of OA. In this thesis, the correlation between the response of cartilage and bone tissue to loading, will be studied in healthy knee subjects without OA, which subsequently could be a baseline that may be compared with the function of a potentially sick knee joints, and by that tell us whether or not the cartilage and bone are responding to the loading the way they should. The indicator of a physiological response measure will be the $T_1\rho$ and T_2 parameters for cartilage and ^{18}F-NaF uptake for bone.

This hypothesis was examined primarily by processing a set of previously collected baseline scans followed by post-exercise scans for healthy subject data from ^{18}F-NaF PET/MRI examinations of the knees. These were acquired in the USA, by Stanford University and the data was thus already available prior to my thesis work. Therefore, the central focus of this thesis was to process these medical images and analyse the change in response to the exercise in the cartilage tissue $T_1\rho$ (T_1 relaxation time in the rotating frame) and T_2 relaxation times and the ^{18}F-NaF uptake in the adjacent bone tissue. $T_1\rho$ and T_2 will provide us with information about the changes in proteoglycan, collagen fibers and water content of the cartilage, which is why these parameters are used for evaluating degradation or integrity of cartilage [1] [15]. According to some researches, the cartilage integrity is linked to the loading effects in knees and due to this, the PET/MRI post-exercise scans on the healthy subjects, should show us changes compared to the baseline scans, that can be linked to the integrity of the cartilage [9].

An open-source Python library for MRI processing techniques, called DOSMA (Deep Open-Source Medical Image Analysis), was used to segment the cartilage in the knee and also to generate T_2 maps of the cartilage [37]. Furthermore, the free open source medical image viewer Horos, was used to e.g. view and resample the scans.

Based on the results of the ^{18}F-NaF PET/MRI data, an additional experiment was executed by me at Rigshospitalet Glostrup, to explore the prospect of an optimized MRI T_2 measure study, with the purpose of measuring the change in response of the strained knee cartilage, as this is a crucial point for being able to investigate the hypothesis. The data was collected from parametric MRI scans of T_2 for cartilage of joints, for 5 subjects with healthy knees. A baseline scan followed by a post-exercise scan, supported the investigation of optimizing the method for detecting a change in cartilage T_2 values in knees. The main parameters probed in this study were a different type of knee exercise and a shorter time span between the exercise and the acquisition of the post-exercise scan. Furthermore, the effect of BFR on T_2 values in knees were investigated, by giving the subjects a tight band around one of their legs before doing the exercises.

To analyse these data, a segmentation of the knee cartilage on the MRI scans were carried out utilizing a MATLAB code. Additionally, a calculation of the T_2 values were done based on the particular MRI T_2-weighted scans that were collected. Finally, the T_2 values of the knee cartilage regions were compared, in order to analyse the differences in response to the exercises, using the statistical analysis method called Mixed-Effects Model. This particular method will permit the use of both fixed effects and random effects.

The structure of this thesis will primarily follow the commonly known organizational structure in scientific writing; Introduction, Method, Result And Discussion (IMRAD). As a part of the introduction (i), the following chapter, theoretical background (3), will review the necessary theory underlying the investigation, which are about the medical imaging techniques utilized for this study, Magnetic Resonance Imaging (MRI) and Positron Emission Tomography (PET). Considering that this thesis investigation consists of two different studies, the next two parts (ii & iii) will be the following two studies: PET/MRI bone-cartilage interaction - Stanford Study and MRI cartilage T_2 change - Glostrup Study, each of which will include a chapter of the methods, where the experimental as well as the data processing methods used in the study, will be explained. Thereafter, it will include a chapter in which the results of the investigation will be presented, divided into three sections of: raw, processed and analysed data. The principal reason for this division of the data, is for making it clear what data was directly acquired from the imaging modalities and thereby was raw data, what were the processed data acquired by utilizing various image processing methods and lastly the data analysed based on the processed data, for investigating the hypothesis. The final chapter of these two parts will be a partial conclusion of each studies. These two parts will be followed by part iv with a discussion of both of the studies as a whole. Finally, to close off the thesis investigation, part v conclusion will emphasize clearly and concisely the answer to the problem statement, and thereby either confirm or deny the hypothesis written in the Introduction.

3

THEORETICAL BACKGROUND

In this section, the theory underlying the experiment and thus the answering of the problem statement will be reviewed. The thesis is mainly about solving a problem, using the diagnostic imaging techniques; PET and MRI, hence the theoretical background will mainly be based on the following basic textbooks; ***Magnetic Resonance Imaging - Physical Principles and Sequence Design*** by Haacke EM (1999) [18], ***Quantitative MRI of the Brain - Principles of Physical Measurement*** [17] by Cercignani M (2018) and ***The Essential Physics of Medical Imaging*** by Bushberg JT (2012) [16].

3.1 MAGNETIC RESONANCE IMAGING

Magnetic Resonance Imaging (MRI) is a medical imaging technique, that applies Nuclear Magnetic Resonance (NMR). The basic principle behind NMR, makes it possible to create internal anatomical medical images of the body tissues, by using the magnetic properties, called the nuclear spin, of the tissue molecules. In biological tissue, it is mainly the hydrogen nucleus (the proton) that has these properties, also known as the intrinsic angular momentum of an atomic nucleus. When the spinning proton in hydrogen is placed in a magnetic field, it will begin to precess about that field. This means that the axis of rotation of the spinning proton, will start to move in a circular motion around the fixed axis of the magnetic field, in other words it will have a wobbling motion like a spinning top. What is very interesting here, is how fast the precession happens, and the resonance frequency of this is called; the precessional frequency or Larmor frequency. The Larmor frequency is specific to each nucleus, and is proportional to the magnitude of the main magnetic field that it experiences.

NMR or MR imaging (the word *'Nuclear'* was suppressed, because of the extensive concern for any phrase containing this word) was made possible, because of the key element of encoding the data spatially. When the object is exposed to a spatially varying magnetic field, the Larmor frequency will accordingly vary spatially. The spatial information of the object can then be obtained, by separating the different frequency components of the signal. These are some of the first basic elements underlying the very powerful, well established and completely harmless imaging modality. More of the concepts and principles of MRI will be reviewed, in more details in the following sections.

3.1.1 *Fundamental Interaction of a Proton Spin with the Magnetic Field*

As mentioned above, in MRI the dominating nucleus is the proton in the hydrogen nucleus. The reason why hydrogen is dominant, is first an foremost because the human body consist of up to 60% water [27], and hydrogen is one of the components of water. Secondly, all individual elementary particles have an intrinsic spin, meaning that they rotate about their own axis [35]. According to how many proton and/or neutron the nucleus consist of, the spins of these will be added together and either have zero or a non zero net spin. The net spin will determine whether the nucleus of the atom has an overall spin. Generally the rule is, that any nucleus with either an odd atomic number or weight will have a net spin [20]. Since hydrogen only consist of one proton and one electron, its atomic number is 1 and its atomic weight is 1.008 u, thereby the nucleus of hydrogen has a net spin. This spinning motion produces a tiny magnetic field, due to the movement of the electrical charges and this is called the magnetic moment. The magnetic nucleus could be pictured as a tiny compass, which is a very sensitive detector of magnetic fields and as we know, it will point in the direction of the earth's magnetic field. The small nuclear magnets will act equivalently, so when placed inside a very strong magnetic field, they tend to point at the same direction as the magnetic field. A large amount of net nuclear magnetization will then occur, just as the individual nuclear magnetic fields will be added together. So all of the hydrogen nucleus will now act unified as a big compass, instead of millions of individual small compasses. But in contrast to the compass needle remaining still, the interaction of the spinning hydrogen nucleus (proton) with the external magnetic field, will result in the precession of the proton spin about that field direction. What causes this precession to happen, is a torque in the direction of the

precession. When the spinning proton interacts with the magnetic field, a rotational force will be produced, forcing the spin to precess about the fixed axis of the magnetic field named; $\vec{\mathrm{B}}_0$. So overall the imaging of humans is achievable because of the capability to manipulate with a combination of magnetic fields, and detecting the bulk precession of the hydrogen spins mainly in water but also in fat and other organic molecules.

As stated earlier, the precessional angular frequency of the magnetic moment vector for the proton around the external static magnetic field, $\vec{\mathrm{B}}_0$, is dependent on the magnitude of this magnetic field and the gyromagnetic ratio, γ, of the proton. This precession frequency, as previously mentioned, is also called the Larmor frequency, hence the following equation that defines this frequency is referred to as the Larmor equation:

$$\omega_0 = \gamma B_0 \tag{1}$$

The gyromagnetic ratio of a nuclear is defined as the ratio between the nuclear magnetic moment, μ, and the nuclear spin, I_N. The value of this constant, varies in terms of which atomic species it is [31]. For the proton in hydrogen, the γ value is approximately $2.6 \times 10^8 \mathrm{rad/s/T}$.

3.1.2 *Magnetization*

Magnetization is defined as the average magnetic dipole moment density, in which the magnetic dipole moment indicates the strength and the direction of the magnetic field. To put this in perspective to the images you can acquire of the macroscopic body using MRI, we will have to focus on the protons and their local magnetic moment per unit volume, referred to as $\vec{\mathrm{M}}(\vec{r}, t)$. By that magnetization is given by:

$$\vec{\mathrm{M}} = \frac{1}{V} \sum_{\substack{\text{protons} \\ \text{in } V}} \vec{\mu}_i \tag{2}$$

Here V indicates the volume of one volume element, also called voxel. Voxel is identical to a pixel (picture element), but just in a 3 dimensional space. Here we are assuming that the volume V is small enough in order for the external fields to be approximately constant over V, but still big enough to enclose a large number of protons. $\vec{\mu}_i$ denotes the magnetic dipole moment of each protons in the volume V. The microscopic collection of spins in this V has the same phase, which implies that they resonate at the same frequency, and this is called a spin isochromat. To understand NMR we can utilize the equation of motion from

classical mechanics. Since the angular momentum is the equivalence of linear momentum, but in rotation, Newton's second law can be interpreted to state that; the time rate of change of angular momentum of a nuclear dipole depends upon the torque applied on the dipole by the exerted magnetic field. The equation for a system's change of total angular momentum $\vec{\mathrm{J}}$, when the torque $\vec{\mathrm{N}}$ on the system is nonzero, is given by:

$$\frac{d\vec{\mathrm{J}}}{dt} = \vec{\mathrm{N}} \tag{3}$$

Applying firstly the definition of torque on a magnetic dipole moment caused by an external constant magnetic field $\vec{\mathrm{B}}$:

$$\vec{\mathrm{N}} = \vec{\mu} \times \vec{\mathrm{B}} \tag{4}$$

and secondly the direct relation between the magnetic dipole moment and the spin angular moment:

$$\vec{\mu} = \gamma \vec{\mathrm{J}} \tag{5}$$

eq. 3 can be reduced to the fundamental equation of motion:

$$\frac{d\vec{\mathrm{J}}}{dt} = \vec{\mu} \times \vec{\mathrm{B}} = \gamma \vec{\mathrm{J}} \times \vec{\mathrm{B}} \Rightarrow \frac{d\vec{\mu}}{dt} = \gamma \vec{\mu} \times \vec{\mathrm{B}} \tag{6}$$

This equation is a simpler form of the Bloch equation, also called the equations of motion of nuclear magnetization, which will be reviewed and derived in more details in a following section. Neglecting the interactions of the proton with their surroundings, we can take the sum over the fundamental equation of motion (eq. 6) for the individual spins per unit volume, which will give us the following equation:

$$\frac{1}{V}\sum_i \frac{d\vec{\mu}_i}{dt} = \frac{1}{V}\sum_i \left(\gamma \vec{\mu}_i \times \vec{\mathrm{B}}_{ext}\right) = \frac{\gamma}{V}\sum_i \vec{\mu}_i \times \vec{\mathrm{B}}_{ext} \tag{7}$$

Using the definition of magnetization (eq. 2), the equation above can be reduced to the following differential equation, which still only applies for non-interacting protons:

$$\frac{d\vec{\mathrm{M}}}{dt} = \gamma \vec{\mathrm{M}} \times \vec{\mathrm{B}}_{ext} \tag{8}$$

In this equation the external static main magnet field is defined as; $\vec{\mathrm{B}}_{ext} = B_0\hat{z}$, and it is most convenient to study the magnetization along with its differential equation, in terms of the parallel and perpendicular components defined in respect to $\vec{\mathrm{B}}_{ext}$. The parallel component, also named longitudinal component of the magnetization is denoted as; $\vec{\mathrm{M}}_{\parallel} = M_z\hat{z}$. The perpendicular or transverse components are then denoted as; $\vec{\mathrm{M}}_{\perp} = M_x\hat{x} + M_y\hat{y}$. These

components, when applied in eq. 8, will lead to the following decoupled equations, because of the cross product (the formula of cross product; $\vec{a} \times \vec{b} = \|\vec{a}\| \|\vec{b}\| \sin(\theta)$, where θ is the angle between $\vec{a}$ and $\vec{b}$):

$$\frac{d\vec{M}_{\parallel}}{dt} = \gamma \vec{M}_{\parallel} \times \vec{B}_{ext} \Rightarrow \frac{dM_z}{dt} = \gamma M_z B_0 \sin(0)$$
$$\frac{dM_z}{dt} = 0 \tag{9}$$

$$\frac{d\vec{M}_{\perp}}{dt} = \gamma \vec{M}_{\perp} \times \vec{B}_{ext} \tag{10}$$

These equations still only applies to the incidents, where the protons do not interact with their environment. To be able to model the interactions of the proton with their neighborhood, we have to acknowledge the decay parameters, which will be added to the equations above (eq. 9 & 10) as an extra term. These decay parameters will be different for the two equations, due to reason that the magnitude of the macroscopic magnetization is a vector sum of a large number of proton spins. This leads to different relaxations of the parallel and perpendicular components of $\vec{M}$ to the external field, in their approach to reach the equilibrium values. These decay parameters are also known as relaxation, will be elaborated and reviewed in more details in the following section.

To be capable of detecting the magnetization of the system and accordingly the signal from the macroscopic body, we need to tip away the magnetization vector, $\vec{M}$, from the external field direction, $\vec{B}_{ext}$. This will subsequently cause the magnetic field, produced by the collection of proton spins, into precessing along with the external magnetization. The precession leads to a change in flux, that will be detected by a nearby receiver coil. The changing flux is basically the disturbed spins, relaxing back to their equilibrium value. The equilibrium value, also mentioned above, is the longitudinal equilibrium magnetization, denoted with M_0, which is the component of the average magnetic dipole moment density vector, along the external field direction. If we consider a sample of protons in some tissue, with the number of protons per unit volume defined as ρ_0, known also as the spin density, multiplying the proton magnetic moment component $\gamma\hbar/2$ with the relative spin excess $\hbar\omega_0/2kT$ and the spin density ρ_0, will give the longitudinal equilibrium magnetization M_0. Here the spin excess is the number of spins parallel to the magnetic field, which exceeds the number anti-parallel to that field, and these are known to be very small.

Knowing that the precessional frequency ω_0 is given by eq. 1, the longitudinal equilibrium magnetization can thus be written as:

$$M_0 = \frac{\gamma\hbar}{2} \cdot \frac{\hbar(\gamma B_0)}{2kT} \cdot \rho_0 = \frac{\rho_0 \gamma^2 \hbar^2}{4kT} B_0 \tag{11}$$

This equilibrium value, is what makes it possible to measure the NMR effects, and thereby acquire images of the body.

3.1.3 *Relaxation*

The main measure in this thesis is the quantification of T_1 and T_2 relaxation times in cartilage tissue and how it changes under loading, thus this section will thoroughly review the term of relaxation. To understand the relaxation as well as the decay parameters, we have to take a closer look on how the magnetization gets tipped and what happens when this is done. As mentioned before, when spinning protons are placed in a strong magnetic field, their magnetic moment will align along the external magnetic field, that is exerted on them. This magnetization can be rotated away from its alignment along the axis of the external magnetic field, that is the longitudinal direction, by applying a Radio Frequency (RF) pulse for a short time. The RF pulse is basically a wave of magnetic field, that is transmitted in short bursts from a RF-transmit coil. The RF of this pulse, needs to be near the Larmor frequency, which means that the RF pulse will have the same frequency as the precessional frequency of the spinning protons, ensuring that they are in resonance. The resonance allows the protons to absorb the energy from the RF pulse, just enough to rotate their axes away from the longitudinal direction. When the RF pulse stops, the protons start to release the absorbed energy, and consequently return to their previous alignments, emitting a signal back to the coil, that is measured in MRI. This is the phenomenon known as relaxation and as previously stated it is composed of two processes happening simultaneously; longitudinal and transverse relaxation. Below the longitudinal relaxation followed by the transverse relaxation will be reviewed. These two combined gives the Bloch equation, which will be studied and solved in the following section.

For interacting protons, eq. 9 will fundamentally be wrong, as it is missing the term describing how the spins of the protons will seek back to align with the external field, by exchanging the excess energy with the surrounding lattice, that is the molecules within the neighbourhood.

During the longitudinal relaxation process, the longitudinal magnetisation M_z will grow back towards the equilibrium value M_0. The time constant T_1 is the longitudinal relaxation time (also called spin-lattice relaxation time), which accordingly indicates the rate at which M_z will return to M_0. So the growth rate from the constant interactions of the protons with the surroundings, consequently leads to proportionality between the rate of change of the longitudinal magnetization, $\frac{dM_z}{dt}$, and the difference; $M_0 - M_z$. This empirically determined proportionality constant characterizes the inverse of the time scale of the growth rate, T_1, which means that eq. 9 can be rewritten as:

$$\frac{dM_z}{dt} = \frac{1}{T_1}(M_0 - M_z) \tag{12}$$

In human tissue for B_0 field strengths of 0.01T or higher, the longitudinal relaxation parameter T_1 for protons will range from tens to thousands of milliseconds. In this thesis, the $T_1\rho$, rather than the T_1 relaxation parameter will be investigated. This is principally analogous to T_1, but instead of characterizing the relaxation along B_0, it characterizes the relaxation of the magnetization along the RF field of a spin locking pulse, applied in the rotating frame of reference [36]. Hence the greek letter ”rho” in the name of this parameter, refers to the ”ro”tating frame [34]. The $T_1\rho$ sequence consist of both T_1 and T_2 weighting.

Likewise for interacting protons, eq. 10 is missing the term that describes how the transverse magnetization decays, as a result of the varying local fields that the spins will experience. This variation in the local fields causes the local precessional frequencies to be different, which thereby leads the individual spins to begin fanning out (also called dephasing, as the spins will have different phases or angles relative to each other) in time. The dephasing will consequently reduce the total transverse magnetization vector, as it is a sum of all of the individual transverse components or spins. The rate of this transverse magnetization reduction is represented by the experimental time constant parameter T_2, that is the longitudinal relaxation time (also called spin-spin relaxation time). So T_2 indicates how fast the MR signal in the transverse plane will decay. A correction of eq. 10 to include this decay, will simply be done by adding the decay rate term $-\frac{1}{T_2}\vec{\mathrm{M}}_\perp$, which will give the following differential equation instead:

$$\frac{d\vec{\mathrm{M}}_\perp}{dt} = \gamma\vec{\mathrm{M}}_\perp \times \vec{\mathrm{B}}_{ext} - \frac{1}{T_2}\vec{\mathrm{M}}_\perp \tag{13}$$

This additional term is what leads to an exponential decay of the initial transverse magnetization vector $\vec{\mathrm{M}}_\perp$, which will be explained more in the following section, where the two

differential equations (eq. 12 & eq. 13) will be solved, as a part of solving the Bloch equation. In human tissue the transverse relaxation parameter T_2 for protons is on the order of tens of milliseconds. Essentially because of the inhomogeneities in the main external field, there will be an additional dephasing of $\vec{M}_\perp$, which can be indicated by another relaxation time constant $T_2^{'}$. Thence, T_2 is the intrinsic, while $T_2^{'}$ is the extrinsic parameter. The overall relaxation time T_2^*, can then be written in terms of the intrinsic and extrinsic decay time parameter as follows:

$$\frac{1}{T_2^*} = \frac{1}{T_2} + \frac{1}{T_2^{'}} \tag{14}$$

This equation is based on the definition of the relaxation rates, $R_2 = 1/T_2$, and that the total relaxation rate R_2^* is defined as the sum of the internal and external relaxation rates:

$$R_2^* = R_2 + R_2^{'} \tag{15}$$

The extrinsic reduction of $\vec{M}_\perp$ due to $T_2^{'}$ is recoverable, in the sense that by applying a rephasing pulse that reverses the dephasing of the spins caused by the external field inhomogeneities, the initial value of $\vec{M}_\perp$ can be recovered. This is described as creating an echo, which will be reviewed in more details in one of the following sections about among others, measuring T_2. But the intrinsic reduction due to T_2 are not recoverable, in that the variations of the field are first and foremost random and furthermore local and time-dependent.

3.1.4 *Bloch Equation*

As mentioned above, by combining the differential equations for magnetization in the presence of an external magnetic field with the relaxation terms for the according components (eq. 12 & eq. 13), we will be able to write an empirical one vector equation, known as the Bloch equation:

$$\frac{d\vec{M}}{dt} = \gamma\vec{M} \times \vec{B}_{ext} + \frac{1}{T_1}(M_0 - M_z)\hat{z} - \frac{1}{T_2}\vec{M}_\perp \tag{16}$$

Here the relaxation terms characterize the recovery back to equilibrium for a field pointing along the z-axis. In this section the Bloch equation will be solved for the case where the field is constant and thereby defined as; $\vec{B}_{ext} = B_0\hat{z}$. Solving the Bloch equation will give a better understanding of the dynamics of the magnetization under these circumstances, and thereby how it is possible to use these signals to acquire medical images of the human body.

Let us start by expanding eq. 16 and thereafter write the three equations of the different components separately, that yields from computing the cross product in the Bloch equation:

$$\frac{d\vec{\mathrm{M}}}{dt} = \left(M_x\hat{x} + M_y\hat{y} + M_z\hat{z}\right) \times \left(0\hat{x} + 0\hat{y} + \gamma B_0\hat{z}\right) + \frac{M_0 - M_z}{T_1}\hat{z} - \frac{M_x\hat{x} + M_y\hat{y}}{T_2} \tag{17}$$

A way of calculating the cross product of two vectors, when they start at the origin point $(0,0,0)$ is by using this definition:

$$\vec{\mathrm{a}} \times \vec{\mathrm{b}} = \begin{bmatrix} a_y b_z - a_z b_y \\ a_z b_x - a_x b_z \\ a_x b_y - a_y b_x \end{bmatrix} \tag{18}$$

In our case the terms with b_x or b_y will be zero, as the external magnetic field only has a z-component, and thus the following three component equations can be written from the Bloch equation by also using the Larmor equation (eq. 1) as follows:

$$\frac{dM_x}{dt} = \gamma(M_y B_0 - M_z \cdot 0) - \frac{M_x}{T_2} = \omega_0 M_y - \frac{M_x}{T_2} \tag{19}$$

$$\frac{dM_y}{dt} = \gamma(M_z \cdot 0 - M_x B_0) - \frac{M_y}{T_2} = -\omega_0 M_x - \frac{M_y}{T_2} \tag{20}$$

$$\frac{dM_z}{dt} = \gamma(M_x \cdot 0 - M_y \cdot 0) + \frac{M_0 - M_z}{T_1} = \frac{M_0 - M_z}{T_1} \tag{21}$$

To solve the Bloch equation, we just have to find the solution for the three differential equations above. For the differential equations in transverse plane, the last term in eq. 19 and eq. 20, is the relaxation term which can be solved with an exponential function. This implies that the additional relaxation term leads to an exponential decay of an initial value of $\vec{\mathrm{M}}_\perp$, which can be clearly seen in the rotating reference frame:

$$\left(\frac{d\vec{\mathrm{M}}_\perp}{dt}\right)' = -\frac{\vec{\mathrm{M}}_\perp}{T_2} \xrightarrow{solution} \vec{\mathrm{M}}_\perp(t) = \vec{\mathrm{M}}_\perp(0)e^{-t/T_2} \tag{22}$$

An illustration of this exponential decay can be seen presented in figure 1. Utilizing integrating factors to solve the transverse differential equations, we can, by changing the variables, $M_x = m_x e^{-t/T_2}$ and $M_y = m_y e^{-t/T_2}$, eliminate the relaxation terms. The solutions for m_x and m_y have the general form; $C_1 \cos\omega_0 t + C_2 \sin\omega_0 t$. This leads to the following set of solutions for the transverse equations in terms of the original variables:

$$M_x(t) = e^{-t/T_2}\left(M_x(0)\cos\omega_0 t + M_y(0)\sin\omega_0 t\right) \tag{23}$$

$$M_y(t) = e^{-t/T_2}\left(M_y(0)\cos\omega_0 t - M_x(0)\sin\omega_0 t\right) \tag{24}$$

To solve the differential equation of the parallel component (eq. 21) the general form of an exponential function can again be used, and furthermore using an integrating factor results in the following solution:

$$M_z = M_z(0)e^{-t/T_1} + M_0\left(1 - e^{-t/T_1}\right) \tag{25}$$

The first term in the equation above will be zero, assuming that the magnetization along the z-axis gets tipped away at the time $t = 0$, and thereafter M_z will have an exponential regrowth to the equilibrium value M_0, which is exactly what this solution expresses. This exponential regrowth is displayed in the sample curve in figure 1 below, where T_1 determines the time scale for the regrowth.

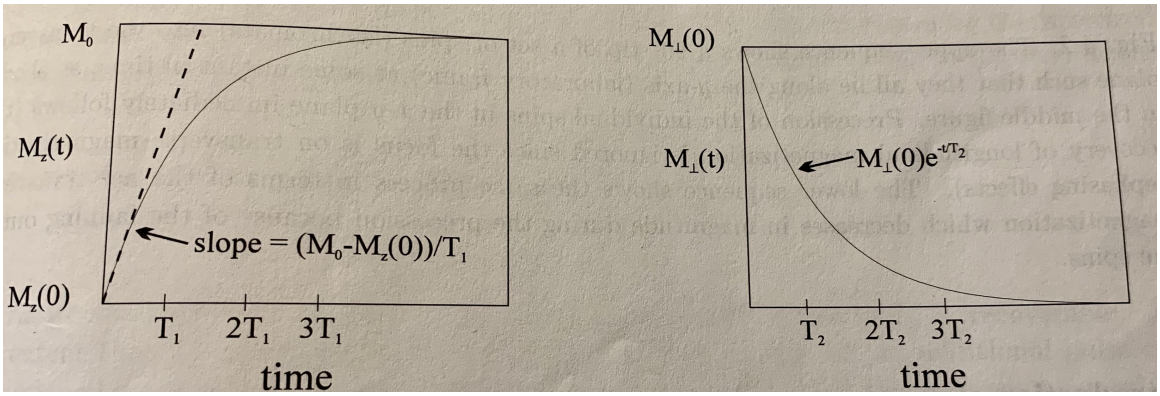

Figure 1: Illustration of the regrowth of $\vec{M}_{\parallel}$ from the initial value $M_z(0)$ to the equilibrium value M_0 (Left plot). Illustration of the decay of $\vec{M}_{\perp}$ from an initial value (Right plot). [18]

To find the solution in steady-state or equilibrium, we have to study the equations at the asymptotic limit $t \to \infty$. This limit leads to the vanishing of all the exponentials and thereby results in the following equilibrium solutions:

$$\lim_{t\to\infty} M_x(t) = \lim_{t\to\infty} M_y(t) = 0, \quad \lim_{t\to\infty} M_z(t) = M_0 \tag{26}$$

This agrees very well with the definition of how relaxation behaves; the parallel magnetization will relax back to the equilibrium value after some time, leaving no transverse magnetization behind.

3.1.5 *Signal Acquisition Method*

The signals from the tissues in the body, can be acquired utilizing different methods in MRI, regarding which tissue properties needs to be examined. For example, an MRI method or experiment called Free Induction Decay (FID) is used to among others locate water resonance peak and accordingly detect the RF amplitude and the time span needed for a maximum signal to be produced. Thus, a FID is performed as a routine on the MRI machines for tuning the RF coils and by that optimizing the system response. This is one of the simplest MRI signal acquisition methods, since by just applying a $\pi/2$ (90°) pulse, a global signal can be detected from a sample. In a static magnetic field with a macroscopic set of hydrogen atoms, the $\pi/2$-pulse will be applied uniformly to the proton spins, leading to the longitudinal magnetization being rotated into the transverse plane. All the tipped spins will then precess together and induce a small electromotive force (emf) due to the magnetic field obtained from the sum over these precessing proton spins fields. The induced emf will be detected in any nearby receiver coil, thereby a FID signal is acquired. The FID signal decays exponentially with the time constant T_2 or T_2^*.

Another very well known signal acquisition method used for measurement of T_2 is the Spin Echo (SE) method. This method is particularly advantageous in measuring T_2, considering that in some instances the external field will not be uniform, and here the time constant T_2' can usually be adequately small, so that $\frac{1}{T_2'}$ will dominate $\frac{1}{T_2}$, resulting in possible extreme losses of the signals. But this is exactly what can be altered by utilizing the SE sequence, which is based on applying two RF pulses; firstly a $\pi/2$-pulse and secondly a π-pulse (also named the refocusing pulse). The SE signal acquisition method can be described in 3 steps:

1. The first $\pi/2$-pulse will tip the magnetization into transverse plan instantly at the time $t = 0$. At first the spins will start point along the $\hat{y}'$ - axis, and afterwards a dephasing of the spins at the different positions will happen relative to each other.

2. The second RF π-pulse will then be applied along the $\hat{y}'$ - axis, thereby rotating the spins about this axis with the angle π at the time $t = \tau$. After this pulse all the spins will have the opposite phase.

3. The spins maintain the accumulation of the phase after the time τ, and will return to a phase $\phi = 0$ at the same time, since the rate of the phase accumulation is unchanged.

When the spins are realigned along the positive $\hat{y}'$ - axis, they will form an echo, which is why this time is defined as the Echo Time (TE), $t = T_E \equiv 2\tau$ (the TE indicates the time from the center of the RF pulse to the center of the echo [32]).

The T_2 data can thus be acquired by sampling at multiple time points. Meaning that if the spatial dependence is neglected, the spin-spin relaxation time can be determined by using the collection of single data points at two different TEs:

$$T_2 = \frac{T'_E - T_E}{\ln\left(s\left(T_E\right)/s\left(T'_E\right)\right)} \tag{27}$$

In eq. 27 the T'_E indicates another TE, which is achieved by varying the time of the π-pulse in a second experiment leading to $T'_E > T_E$. The s in the eq. above indicates the signal acquired at the TEs. Alternately of repeating the SE experiment with varied TE, a multiple spin echo experiment can be done, by applying multiple π-pulses after the single $\pi/2$-pulse. Spacing the n^{th} π-pulse uniformly at the time $(2n-1)\tau$, the signal at each echo can then be acquired and plotted, as a function of the time of the different echos, n, on a semi-log scale. Fitting a straight line to this plot will subsequently give the relaxation time through the slope of this line.

For measuring T_1 a more sensitive signal acquisition method than FID and SE is required. Here another method called Inversion Recovery (IR) can be utilized to acquire accurate values of T_1, but in comparison to the FID and SE sequences, the calculation can not be acquired from one experiment. The IR is identical to the FID sequence, but consists of two RF pulses; firstly a π-pulse and secondly a $\pi/2$-pulse. The first RF pulse will invert the longitudinal magnetization, at a time interval T_I before the second RF pulse, that will then tip the longitudinal magnetization into the transverse plane. In between these two pulses, the longitudinal magnetization will regrow and thereby lead to a strong FID-signal depended on T_I. The accurate method to calculate a defined value of T_1, is by employing the fact that the signal will vanish when:

$$T_{I,null} = T_1 \ln(2) \tag{28}$$

This point corresponds to where the longitudinal magnetization has regrown from $-M_0$ to 0. So to find T_1, T_I has to be varied until a zero in the signal is detected.

The MRI signal acquisition methods or sequences used in this thesis investigation were primarily Dual Echo Steady State (DESS) (Stanford study) and Turbo Spin Echo (TSE) (Glostrup study). DESS is a 3D coherent or steady state Gradient Echo (GRE) sequence, that has, just like the TSE sequence, been a well used signal acquisition method for many years for the imaging of cartilage [12]. With the GRE technique, the echos can be registered significantly more rapid, by employing only one RF pulse, also resulting in a shorter TE when compared to the SE method. The Repetition Time (TR), which is the time between the repetition of the series of pulses and echos [32], can also be short if a low flip angle is applied. The flip angle is the angle at which the longitudinal magnetization gets rotated or flipped. The major characteristics of the steady state sequences are that the TR is short enough for the transverse magnetization not to decay fully before the next evenly spaced RF pulse gets applied. A dynamic equilibrium between the transverse and parallel magnetization can be obtained, when phase coherent RF pulses with the same flip angle are applied with a TR that is constant and has a shorter value than the T_2 of the tissue. Two different types of signals will be created after this equilibrium is attained:

1. A post excitation signal arisen from the most recent RF pulse - A FID signal (S_+).
2. A prior excitation signal arisen from an echo reformation and is a product of the residual echo refocusing at the time of the succeeding RF pulse - A SE signal (S_-).

Two Magnetic Resonance (MR) images with very distinct contrasts can be achieved, from the simultaneous acquisition of the two separate steady state free precession echos. By employing the sum of squares calculation to both echos, the DESS sequence combines these two signals into one. The S_- signal causes a high T_2 contrast and the S_+ signal yields a T_1/T_2 ratio dominated contrast. TSE also known as Fast Spin Echo (FSE), is a modified version of the SE signal acquisition method, invented for reducing the acquisition time, but still providing tissue contrast very similar to the SE sequence [19]. In the SE sequence only a single echo will get detected over one TR, but in contrast TSE is further optimised due to the multiple echos measured per TR [33]. The reason why a train of echos will be generated over one TR, is because the series of π-pulses or the refocusing pulses applied after the first $\pi/2$-pulse, have a moderately different phase encoding gradient for each of these echos [30]. This leads to the acquisition of multiple lines of k-space, which is the phase-encoding steps during one TR, thereby reducing the imaging time.

3.1.6 *MRI of Articular Cartilage*

In this thesis the T_1 (or more specifically $T_1\rho$) and T_2 values will be analysed for the cartilage tissue. These two MRI parameters are essential for cartilage imaging, mainly because the composition of cartilage is water, collagen and proteoglycan, which studies have shown that with 3 T MRI can be evaluated via T_2 mapping, $T_1\rho$ imaging amongst other compositional assessment techniques [10]. The biochemical changes in terms of the structure of the collagen and the interaction of water with the cartilage extracellular matrix can be evaluated through T_2 relaxation time measurements. The biochemical changes of proteoglycan in cartilage can further be assessed by T_1 relaxation measurements. Additionally the macro-molecular cartilage changes can also be reflected in both T_1 and T_2, since they can be utilized to analyse the water protons slow motion that characterizes various magnetic resonance relaxation mechanisms.

3.1.7 *Blood flow restriction*

Blood Flow Restriction (BFR) is a very well known technique employed in the gym for effective muscle building. A BFR band will reduce the supply of oxygen to the muscles, leading to a higher metabolic stress on the muscles in work, which enhances the swelling of the muscles, enabling fast twitch fibers to respond more quickly and effectively to the load caused by an exercise. This implies that combining a low intensity exercise with occlusion of the blood flow can give comparable results to high intensity exercise. Studies have been showing the potential for this being an effective clinical musculoskeletal rehabilitation tool, and therefor have been gaining favourable use clinically [7]. This might be favourable for the investigation of this thesis hypothesis as well, for this reason it was probed in the Glostrup study.

3.2 POSITRON EMISSION TOMOGRAPHY

For this thesis, the main focus is on the MRI part of the medical imaging modality utilized for the study. Hence the theoretical background of the Positron Emission Tomography (PET) modality, will only be reviewed superficially. For further information and theory about this type of imaging modality, chapter 19.3 from the book: *The Essential Physics of Medical Imaging*, will be referred to [16].

3.2.1 *The PET imaging modality*

Positron Emission Tomography (PET) is a Computed Tomography (CT) imaging modality, utilizing nuclear medicine to acquire medical images representing the emitted positron nuclides' distribution in the patients. CT has the requirement of acquiring a set of projection images from no less than 180° arc encircling the patient. This is performed by the numerous detector rings, which encircles the patient in typical PET systems. The activity distribution of the radioactive tracer injected in the patient, will be detected in the PET scanners by attaining the projection through the use of Annihilation Coincidence Detection (ACD). The transverse images are then mathematically reconstructed by the PET system's computer from the attained projection data.

In matter, excitation and ionization leads to most of the kinetic energy of the emitted positrons to be lost. By annihilation, an interaction of the positron with an electron is initiated, when most of the kinetic energy is lost. Accordingly, a conversion of this electron-positron pair into 2 photons with the energy of 511 keV will occur. The photons are emitted in almost opposite directions, and will get detected and identified as pairs of interaction appearing at almost the same time, which is the so called process of Annihilation Coincidence Detection (ACD). Furthermore, the detected photons' trajectories will also be determined by connecting the line between these 2 interactions.

3.2.2 ^{18}*F-NaF PET*

Depending on the purpose of the PET imaging and what the target process within the body is, various radioactive tracers can be utilized. For this study, the purpose of the PET imaging was to study the bone remodeling, which can be done by utilizing [^{18}F]-sodium fluoride (^{18}F-NaF) tracer, that has the ability to be highly absorbed in the bony skeleton very rapidly. Moreover, having a relatively short half time of $T_{1/2} = 110\,\text{min}$, the isotope will get cleared out of the blood circulation rapidly as well and thereby be very advantageous for the PET imaging.

The nuclear equation for the Fluorine-18 β^+ decay, which further results in the 2 photons being detected by the PET aforementioned, can be seen below in eq. 29.

$$\begin{aligned} {}^{18}_{9}F &\xrightarrow{\beta^+\text{decay}} {}^{18}_{8}O + e^+ \\ e^+ + e^- &\xrightarrow{\text{Annihilation}} \gamma + \gamma \end{aligned} \tag{29}$$

The PET voxel values indicate the ^{18}F-NaF-uptake of the according tissue. The absorption of the ^{18}F-NaF tracer not only depends on the tissue type, but also the weight of the target (the knees), meaning that for each subject the ^{18}F-NaF-uptake will be diverse depending on the subjects' body weight too. When analysing the PET voxel values, this factor needs to be taken in consideration, because comparing the different subjects data will be influenced by this. Therefore, the ^{18}F-NaF-uptake acquired directly from the PET images, were converted to the well known nuclear medicine term: Standardized Uptake Value (SUV). This conversion will normalize the ^{18}F-NaF-uptake in proportion to the subject's body weight. The definition of the SUV is:

$$\text{SUV} = \frac{C_{img}}{C_{inj}} = \frac{C_{img}}{\text{ID}/\text{BW}} \tag{30}$$

where C_{img} indicates the voxel intensities (the ^{18}F-NaF-uptake) of the calibrated PET image, the C_{inj} is the ratio between the injected dose (ID) and the body weight (BW) of the subjects. Accordingly, for the data analysis of this study, the SUV of the PET scans will be utilized instead of the ^{18}F-NaF-uptake.

Part II

PET/MRI BONE-CARTILAGE INTERACTION - STANFORD STUDY

4

METHODS

4.1 STUDY DESIGN

The aim of this PET/MRI bone-cartilage interaction - Stanford Study is primarily to investigate the response of the knee joints to loading. Utilizing the hybrid PET/MRI imaging technique, the whole joint can be studied. Thereby, a quantitative technique to better study both the bone and cartilage simultaneously can be explored. Osteoarthritis (OA) patients experience the implications of the disease when moving the sick joint, which is why imaging of joints in movement or after an acute loading, rather than at rest, would be essential to investigate. The experimental acquisition of the data for this study was executed in the USA, by Stanford University after an ethical approval of the study was given by the Stanford University Institutional Review Board (Stanford University, Administrative Panels for the Protection of Human Subjects). ^{18}F-NaF PET/MRI scans of 12 subjects with healthy knee joints were acquired, pre and post completing a one-legged step-up and drop-land exercise by Dawn Holley and Harsh Gandhi [5]. The 12 healthy subjects who participated in this study, consisted of 7 females and five males with the age: 34 ± 7 years and the Body Mass Index (BMI): $23.1 \pm 33\,\mathrm{kg/m^2}$. When the participants of this study were recruited through public advertisement, the only criteria were that they had no former knee injuries or previous history of osteoporosis or other bone related diseases. The design of this study will be described in more details in the following sections.

4.1.1 *Exercise protocol*

In between the pre and post ^{18}F-NaF PET/MRI scan (the imaging protocol will be elaborated in the following section), the subjects performed an exercise for exposing the knee joints for loading, resulting in changes in the cartilage tissue physiology which correspondingly the bone tissue will quickly response to by also changing the physiology. These physiological response can in turn reveal areas of reduced tissue integrity. The exercise protocol consisted of stepping up on a stool of 25 cm height, with the right leg (the step-up), afterwards using the left leg a straight-legged drop jump landing (the drop-land) was performed with a rate of approximately 15 steps a minute. The step-up exercise will cause a low impact on the knee joints, whereas the drop-land exercise will cause a higher impact considering that the subjects tried their best to land with heel first or flat foot. The bone will response with an adaptation after the drop-land exercise with an effectively strengthening, as a high bone-strain magnitude an high strain rate will be induced by this exercise [13]. This exercise was repeated 100 times by each subjects with 30 s of resting two times (after 30 steps and after 60 steps). A visualisation of this exercise can be seen in the series of images in figure 2 [5] with the exercise cycle starting from the right to the left images.

Figure 2: The exercise protocol consisting of a step-up with the right leg followed by a drop-land on left leg from a 25 cm stool, repeated 100 times [5].

4.1.2 *Imaging protocol*

As aforementioned, the simultaneous PET and MRI scanning of each subjects were performed pre and post the exercise described in the previous section. The scanning was executed on a 3 T hybrid Signa PET/MRI system (GE Healthcare, Milwaukee, WI, USA), with the subjects wearing a 16-channel flexible phased-array wrap coil (NeoCoil, Pewaukee, WI, USA) around each of the knees. A prepared dosage of ^{18}F-NaF with the activity of 93 ± 2 MBq was hand injected, and immediately afterwards a bilateral (both knees) PET/MRI scan in a PET bed with the Field Of View (FOV) 26 cm was performed in list mode for 50 min. To determine the remaining activity from the first injection, an additional 3 min PET scan was executed after the subjects performed the exercise, and before the second injection was given for the post exercise scan. The measured residual ^{18}F-NaF activity was subtracted from the post exercise images. Simultaneously with the PET the MRI data was acquired also of both knees, including the sequences for Magnetic Resonance-based Attenuation Correction (MRAC) and Magnetic Resonance Angiography (MRA). The 3D steady state GRE sequence, DESS, was used for acquiring the MRA data with the following imaging parameters: TR/TE $= 21/2.1$ ms, slices $= 18$, slice thickness $= 1.2$ mm and the flip angle $= 15°$. Utilizing a 2-point Dixon fat-water, T_1-weighted fast spoiled GRE sequences the data for MRAC were acquired with the following parameters of acquisition: TR/TE1/TE2 $= 4.1/1.1/2.2$ ms, FOV $= 50 \times 37.5$ cm, matrix $= 256 \times 128$, slice thickness/overlap $= 5.2/2.6$ mm, 120 images/slab and scan time $= 18$ s.

4.2 DATA PROCESSING

The methods used for processing the PET/MRI data, acquired by Stanford University, will be explained thoroughly in this section. The major part of the thesis work was used on medical image processing the data acquired by Stanford. This is a very essential part of Medical Physics for the prevention, diagnosis and treatment of human diseases. The processing of medical images will lead to the further analysis of the acquired imaging data, which can give us the final results of the study of interest. Most of the techniques and methods for the medical image processing used for this thesis, were introduced to me in the following courses at University of Copenhagen, *"Signal and Image Processing"* and mainly *"Medical Image Analysis"*. This thesis made it possible for me to work with the different processing techniques in more depth, and on actual medical images of humans with the purpose of investigating a new diagnosis method.

The PET/MRI data for this study, was received as Digital Imaging and Communications in Medicine (DICOM) files, which is the most common and international standard data/file format used for storing, exchanging and transmitting medical images. This format ensures that the medical images can be exchanged with the data and quality required for clinical use [22]. DICOM arranges the information into collections of data (data sets). It does not only consists of the image data sets, but also a header with information of, for example the patients identification, the study parameters, etc. To get access to these information, an extraction of data from a series of standardized tags within the header is needed [14]. One DICOM file contains the image data set of one slice of the scan. The image processing of these PET/MRI DICOM files, was done using Python, DOSMA and Horos.

4.2.1 *Pre processing data*

To improve the MRI DESS image data from undesired distortions, a pre processing was done as an essential step to prepare for the further data processing, i.e. especially the automatic deep learning segmentation of the cartilage (which will be described in the next section). A lot of the knee DESS images for the different subjects, had artifacts on several of the slices, which may interfere and disturb the automatic segmentation, as these were done using the DESS scans. An example of the artifact, can be seen in the left plot in figure 7 presented in

section 5.2, with a plot of the pre processed and thereby corrected image to the right. The artifacts on these images involves misplaced parts of the leg in the opposite site, and noise in form of bright spots placed beside the leg. Thus to improve the automatic segmentation of the cartilage tissue, these artifacts where removed from the DESS scans, by running through all slices of the DESS images, manually locating the pixel containing these artifacts, defining these pixel to have the value zero and finally writing it back on the corresponding slices DICOM file. To make it more easy, fast and general, the whole column containing the artifact for all the slices in that scan, was defined to have zero pixel values. However this was done without interfering to much with the voxels containing the signals from the knee, which weren't misplaced, as this may result in more disturbing then helping the automatic segmentation.

4.2.2 *Segmentation of cartilage and bone*

For this study, the acquired data consist of 2 PET and 4 MRI scans for each of the 12 subjects. The PET scan consist of 89 slices, the DESS MRI scan consist of 220 slices and the T_1-weighted MRI scan consist of 110 slices. To analyse a possible correlation between the cartilage tissue and the adjacent bone tissue, a segmentation of these two types of tissues are needed in order to distinguish them from the PET and MRI scans. The most accurate way of doing this, would be to manually draw a Region Of Interest (ROI) mask on each slice for all scans, but this would take a large amount of time. To optimize the time used for processing the data, instead of drawing the mask manually "by hand", the Deep Open-Source Medical Image Analysis (DOSMA), an Artificial Intelligence (AI)-powered Python library was utilized for segmenting the cartilage tissue from the DESS scans [37]. As the name implies DOSMA is an open-source pipeline for musculoskeletal analysis, which apart from segmentation also can be utilized for other medical image processing techniques such as; registration, denoising, super-resolution etc. In addition to these image processing techniques, it can also operate quantitative fitting, anatomical visualization and analysis of patellar tilt, femoral cartilage thickness, etc. [23]. For this study DOSMA was employed only for segmentation of knee cartilage and calculation of T_2 map (T_2 map calculation will be explained in later section). The automatic deep learning segmentation method DOSMA uses, is pre-trained Convolutional Neural Networks (CNN), with Osteoarthritis Initiative (OAI) 2D U-Net as

the segmentation model. The design of the deep learning neural network in CNN makes it advantageous for processing images because of the data being structured in arrays [21]. It is very powerful for computer vision, as it has the property of picking up patterns very well on the input images. With up to 20 or 30 layers, CNN is a feed-forward neural network, and the power comes from the convolutional layers. Containing many of these convolutional layers stacked on top of each other, makes it capable to recognize more complex and advance shapes for each of this particular layers. The way convolutional layers are used in CNN, imitates the network of the human visual cortex, where an incoming image is processed by the series of layers and thereby identifying the more complex features. The OAI 2D U-Net model used for the CNN of DOSMA was trained on a down sampled rendition of the OAI iMorphics DESS data set, which was scans of sick knees. Being that the data set of this study are healthy knees, this could cause some complications in the segmentation of the cartilage, and may thereby result in not the most precise segmentation, but still usable.

Utilizing DOSMA [23], a segmentation of the different types of cartilage; femoral, tibial and patellar was done on a DESS knee scan for one subject, resulting in the femoral cartilage segmentation having the best outcome. Furthermore, it showed that the segmentation was needed to be done bilateral (both knees separately), hence the segmentation of only the femoral cartilage for all 12 subjects were done, after separating the DESS scans into left and right knee. Additionally to the femoral cartilage mask achieved from the automatic segmentation, a region mask for the femoral cartilage was obtained to be able to distinguish the different regions of the femoral cartilage including; trochlea, central and posterior regions for both the medial (inner half) and lateral (outer half) of the knee. This was obtained foremost using the segmented femoral cartilage mask and further a ROI mask already drawn based on the PET scans. These ROI masks needed to be resampled to the DESS space, which was done using Horos. Afterwards the resampled ROI mask was compared with the pixel of the automatic segmented femoral cartilage mask, to obtain the final femoral cartilage region mask (visualised in figure 9) used for the further data processing and analysis.

A segmentation of the subchondral bone (the bone tissue adjacent to the cartilage tissue) was also needed, but this was already done on the PET scan. The mask was given to me, which then was resampled to the DESS space again using Horos (bone mask visualised in figure 10), as all of the analysis will be done on the images in DESS space.

4.2.3 *Adjacent cartilage and bone voxel*

The main purpose of this study is to investigate a possible correlation between the cartilage tissue and the adjacent bone tissue after a loading of the knee joints. To analyse the correlation between the values of the cartilage voxel and the adjacent bone voxel, it is first and foremost highly essential to achieve an optimal method for finding these voxel. Different methods were tried, but the following voxel-wise method was the most optimal way of finding the adjacent voxel: Utilizing the acquired cartilage and bone mask described in the section above, a python code was written to run through each slice of the masks, for one voxel at the time to search through the neighbouring voxel of the cartilage voxel, whether it could meet the criteria for being assigned as the adjacent bone voxel. The code for the adjacent bone voxel finding was implemented (this python code can be found in the appendix), so that firstly two fundamental morphological image processing operations were performed on the femoral cartilage mask. The first morphological operation was erosion, which removes the outermost voxel, ensuring that the cartilage mask used for the search indeed is the voxel of the cartilage tissues, in case the automatic segmentation picked up something on the edges that is not. The second operation carried out on the cartilage mask was a dilation, that was used to ad the nearest region around the cartilage mask, as a part of the mask defining this whole region of the scan to not be regarded as possible bone voxel in the search. These morphological operations will lead to the creation of a thin region between the cartilage and bone voxel being left out, in the search for the adjacent bone voxel, which is the soul purpose of the operations. Afterwards utilizing the new eroded cartilage mask, for each slice and cartilage voxel at a time, the nearest neighbouring voxel within a certain radius (defined in order that mostly the deep cartilage will be chosen for the further analysis), were ran through all of the following criteria for being assigned as the adjacent bone voxel for that particular cartilage voxel:

- The voxel is part of the bone mask.
- The voxel is not a part of the dilated cartilage mask.
- The distance from the voxel to the mean point of the cartilage mask, has to be smaller then the threshold defined as; the distance between the current cartilage voxel and the cartilage mask mean point.

The threshold was defined to assure that the assigned bone voxel was indeed the femur/thigh bone, and not the tibia/shin bone or the patella/kneecap bone. The mean point of the cartilage mask is computed for each slice, defined in such a way that it was placed approximately in the middle of the femur knee bone. The criteria described above leads to the possibility that one cartilage voxel having several different adjacent bone voxel assigned. In addition to saving the voxel of the cartilage and the corresponding adjacent bone voxel, for then later to be able to analyse the MRI cartilage voxel values with the adjacent PET bone voxel values, the assigned adjacent bone voxel with the cartilage voxel were saved as a mask. This was done primarily for being able to visualize the final cartilage and bone mask used for the further data analysis, and to make sure that the masks are correct. The cartilage region mask was also used when saving the voxel values from the adjacent bone voxel finding, to later be able to distinguish the different regions of the cartilage in the analysis.

4.2.4 *Calculating T_2 map*

DOSMA was again utilized for calculating the T_2 map from the DESS scans. The T_2 is calculated using the dual echos, just as described in the theoretical background section 3.1.5 about Signal Acquisition Methods. DOSMA has the options to do the calculation with fat, fluid or no suppression of the knee. After trying the T_2 map calculation with; fat, fluid, fat & fluid and no suppression, the best outcome was observed to be with no suppression, since the different types of suppression resulted in some parts of the cartilage to have no T_2 values. Without the suppression the T_2 map had some noise, but this could easily be removed using a threshold for values that definitely could not be femoral cartilage values. Besides, only the femoral cartilage region of the T_2 map was used for the further data analysis. In figure 3 a slice from each of the different T_2 map calculation can be seen visualized.

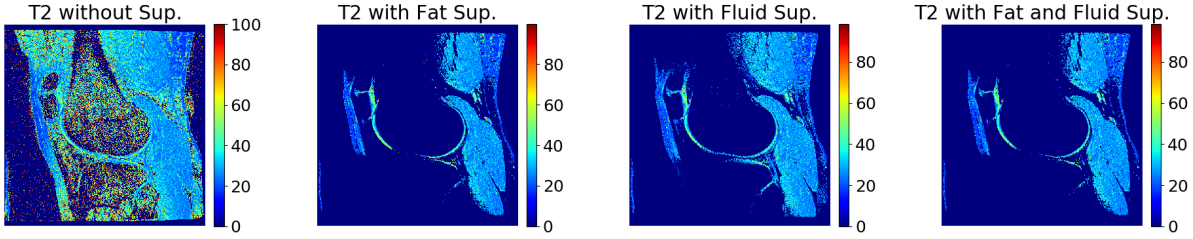

Figure 3: Plots of the same slice of the T_2 map calculation with fat, fluid, fat & fluid and no suppression for subject 2.

4.2.5 *Registration of PET and $T_1\rho$ map*

For answering the hypothesis of this thesis, that PET/MRI imaging can be employed to measure the correlation of the change in the cartilage and adjacent bone tissues after a loading of the knee, the acquired PET/MRI data needs to be analysed voxel wise. Being that the PET images has another dimension (both the number of slices and the resolution) than the two different MRI images acquired, as aforementioned the MRI DESS images were chosen to be the target image, which the two other medical images were geometrically aligned to, for the further voxel wise analysis. This was carried out by performing medical image registration, a process were different sets of data gets transformed into one coordinate system. With the help from Postdoc Valentina Mazzoli, a python code written by her was utilized to register the PET images and $T_1\rho$ map to the target high resolution DESS images. This registration uses an open source software called Elastix [38], which is a toolbox for image registration employing the rigid and nonrigid methods. For the PET and MRI images studied in this thesis, the registrations were done mainly with the simple rigid method, but some of the images or maps had better results using an affine or B-spline registration. Hence all of the $T_1\rho$ maps and PET images for all subjects, where run through the 3 different registration methods, to find the best possible outcome of the registration to the DESS space, right and left knees separately since not working bilateral gave even better results. The registration was still not completely flawless, but for the further analysis it was considerably acceptable, which is why this was chosen to continue to work with. For the PET images the ^{18}F-NaF uptake values were after the registration, converted to SUV (utilizing the conversion equation from the PET theory section). This conversion is necessary, in order to be able to compare the values of all 12 subject's PET data, without it being dependent on the subject's body weight, which will have an impact on how the radioactive tracer will be absorbed differently in the knees.

5

RESULTS

5.1 RAW DATA

This section of the results will present a sample of the raw data obtained by Stanford and what were further used for the data processing and subsequently for the data analysis. The sample that will be presented in this section, is the baseline data for one subject. The reason for only presenting the baseline data, is due to the fact that there will not be any major visible difference between the images of the baseline (pre-) and post-exercise scans of the knees. The change in response of the exercise should be more visible quantitatively, which will be presented in the Data analysis section (5.3). All of the figures in this section, are screenshots of the data viewed by means of the medical image viewer Horos.

5.1.1 ^{18}*F NaF - PET/MRI Scans*

Figure 4 shows a slice of the baseline PET knee scan in sagittal plane, acquired simultaneously with MRI. As previously stated, the PET tracer ^{18}F-NaF (NaF) is a very good bone seeking agent, and this is visualized very well in this figure. The dark pixels are where the radioactive tracer is absorbed in the knee. The darker the pixel of the scan are, the higher the NaF uptake will be in this pixel. The darkest area you can see on the figure, is the knee cap also called the patella. This indicates that, in this slice of the scan in sagittal plane, it is the patella that has the highest NaF uptake, hence the bone tissues in patella must have increased metabolism. This method of imaging has a very high sensitivity and can thus register uptake of radioactive tracers in very small amounts. This is exactly why the baseline PET knee scan looks like it

has picked up even the small amounts of the NaF uptake, absorbed in the tissues that is not bone.

Figure 4: Baseline PET knee scan of subject 2

Figure 5 shows a slice of the images acquired from both signals from the baseline MRI DESS knee scan in sagittal plane, acquired simultaneously with PET. The left image is the signals acquired from the 1. echo, thusly with no diffusion. The right image is the signals acquired from the 2. echo, and by that the image with diffusion. The images with diffusion can be utilized to study the movement of the water molecules, by comparing it to the image without diffusion. The image from the 1. echo has much better quality then the other image, and the cartilage is highlighted more on this image. Some of the DESS scans, like the one in figure 5, has an artifact which may result in complications of the automatic segmentation of the cartilage. The artifact can be seen very clearly on the left image, where a small part of the leg, in the right lower corner is cut off and placed on the opposite site of the image. This was easily corrected, as described in the Methods section, by defining the pixels, where the artifact is located, to be zero.

Figure 5: Baseline MRI DESS knee scan of subject 2 (Left: 1. echo & Right: 2. echo)

5.1.2 $T_1\rho$ - *map*

Figure 6 is the baseline $T_1\rho$ map, calculated from T_1-weighted scans, so in fact this is not a sample of raw data. But as the calculation was not made by me and it was handed to me already processed, this data will be treated as raw data. The T_1-weighted scan was repeated 4 times with the following Inversion Time (TI); 10 ms, 20 ms, 30 ms and 40 ms. The signal, S, from each of the 4 images, increases with the increasing TI and is given by the following equation:

$$S(TI) = M_0 \cdot \left(1 - \exp \frac{TI}{T_1\rho}\right) \tag{31}$$

The $T_1\rho$ map is accordingly calculated, by fitting the signals from each voxel over the 4 TI's to equation 31. This was done in MatLab for all 12 subjects, by my supervisor.

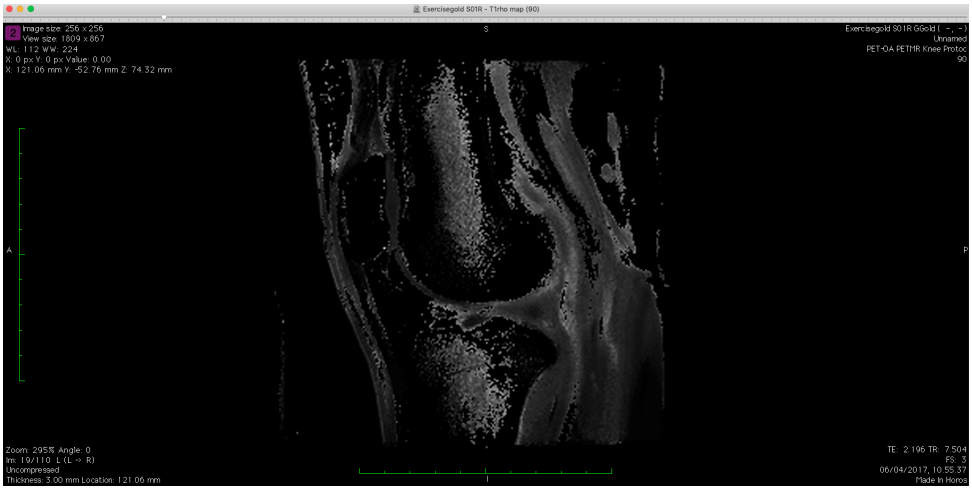

Figure 6: Baseline $T_1\rho$ map of the knee of subject 2

5.2 PROCESSED DATA

This section will present a sample of the processed data, which were further used for the data analysis. The sample that will be presented in this section, are the results of the medical image processing of baseline raw data for one subject from the PET/MRI bone-cartilage interaction - Stanford Study (the raw data presented in the previous section). All of the figures in this section and the following section, are the data plotted in Python (the Python code for these results, can be found in the Appendices section).

5.2.1 *Pre processed data*

To obtain the most optimal automatic segmentation of the knee cartilage, using the Python library DOSMA, a correction of the artifact, described in the previous section, was done on both baseline and post exercise DESS scans. An example of this pre processing, can be seen in figure 7, where the plot to the left is of the original DESS scan from the 1. echo without any corrections. The plot to the right is of the pre processed DESS scan with the correction of the artifact. Additionally a small noise, located above the 'cut off' artifact on the lower left corner, can also be seen removed, which should furthermore ensure that the automatic segmentation will not get disturbed. Similar noises were found on other subject's scans and even on other slices of the presented subject's scan, but sometimes these were even bigger and more bright then the one seen in figure 7. Hence removing these noises, even if it was outside of the knee, should only be optimizing the segmentation. At times these corrections were needed to be done on both sites of the knee, so all of the subject's DESS scans were inspected for these kinds of artifact or noise, and accordingly corrected, before running the scans through the automatic segmentation of the knee cartilage.

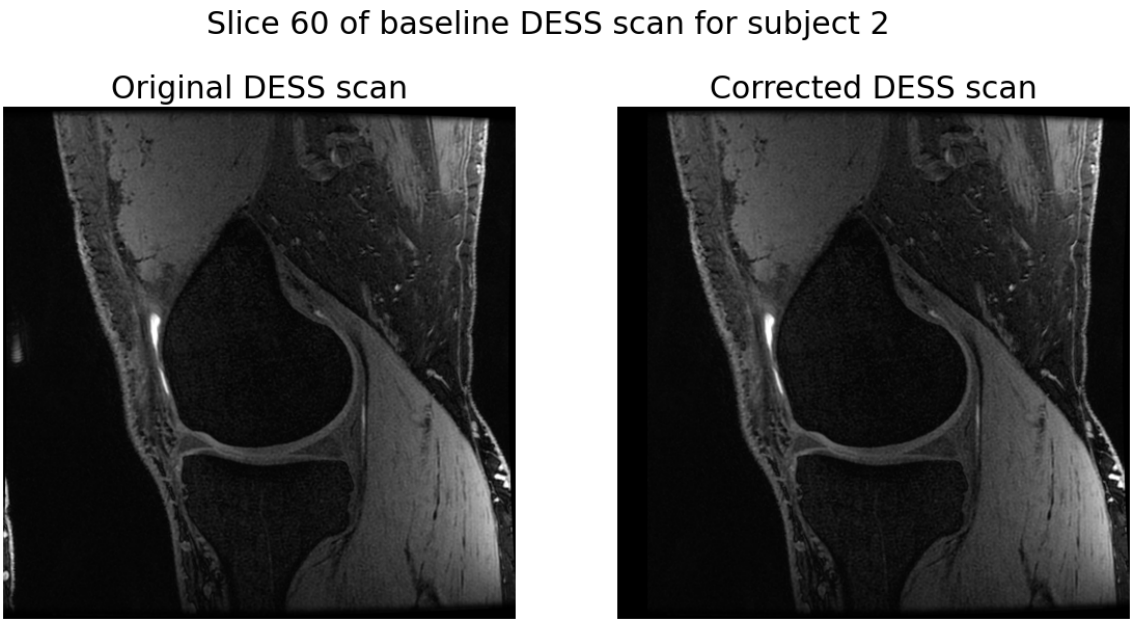

Figure 7: Original and corrected baseline DESS scan of subject 2

5.2.2 *Segmentation of cartilage and bone*

To be able to compare and study the correlation between the knee cartilage and the adjacent subchondral bone values, we have to distinct between the pixel of cartilage and bone on the PET/MRI scans. Thus a segmentation of the knee cartilage was performed firstly, utilizing the Deep Open-Source MRI Analysis pipeline, DOSMA. After segmenting the different types of the knee cartilage tissues using the pre-trained CNN for one subjects DESS scan, and subsequently inspecting the quality of these, the best segmentation was clearly archived of the femoral cartilage, being the case why only this particular tissue was further investigated in this thesis. The reason for this particular tissue having the best segmentation outcome from using DOSMA, is primarily due to the segmentation models being mostly trained on femoral cartilage. By means of this the model had more difficulties recognising the other types of cartilage tissues, consequently resulting in poor segmentation. Moreover the knee cartilage on the DESS scans, in some areas were not highlighted as much as they should, which also may have lead to challenges in the segmenting. To be able to segment the cartilage of both knees, it was necessary to separate the DESS scans into Left and Right knee, and then perform the segmentation for the knees separately. Below in figure 8, three different slices of the femoral cartilage mask from the segmentation can be seen plotted on the corresponding DESS images of a baseline scan. On these slices it is possible to see how the segmentation

are relatively acceptable, even though they are not perfect as they are missing some of the cartilage pixel. But it is more preferable, that it segmented with some missing cartilage pixel, rather then segmenting pixel that are not cartilage. On top of that the automatic segmentation spared a lot of time, compared to if the segmentation should have been done manually for all 12 subjects DESS scans.

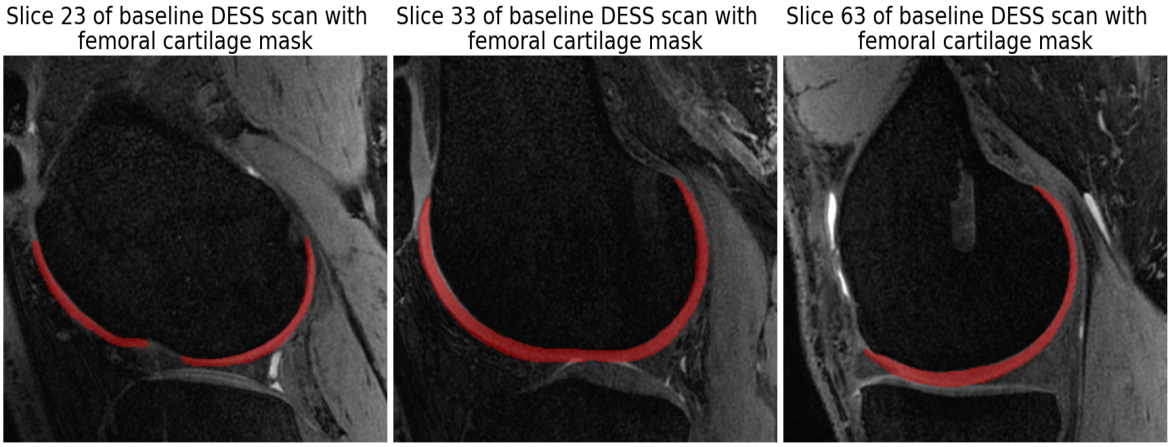

Figure 8: 3 slices of baseline DESS scan with femoral cartilage mask

In addition to distinguishing the knee cartilage tissue from the knee scans, it was also interesting and substantial to achieve distinction between the different regions of the femoral cartilage, because of the way this particular type of tissue is build. This may lead to different ways of receiving the load of an exercise in the different regions of the cartilage. In figure 9 some of the different regions in the femoral cartilage can be seen visualised, where the different colors represent the different types of regions. The first 2 plots in this figure are slices of the medial part of the knee and the last plot is the lateral part of the knee. This implies that the blue, green and red colors respectively indicates the trochlea, central and posterior regions of the medial knee. On the last plot only the central and posterior regions of the lateral knee are visualised with respectively blue and orange colors. Comparing figure 8 with 9, it is possible to see how some part of the cartilage mask is missing, which is because of the way the region mask is computed among others from the automatically segmented cartilage mask.

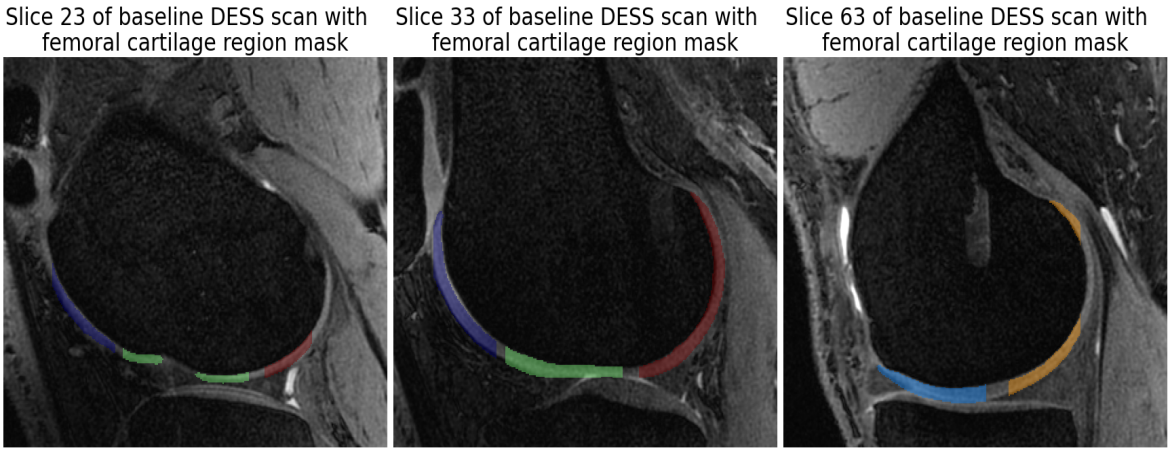

Figure 9: 3 slices of baseline DESS scan with femoral cartilage region mask (Dark blue: Trochlea Medial, Green: Central Medial, Red: Posterior Medial, Light Blue: Central Lateral & Orange: Posterior Lateral)

The values of the cartilage regions have to be compared with the values of the adjacent subchondral bone, implying that a segmentation of this region of the knee is needed as well. Segmentation of the subchondral bone was already made by my supervisor, Bryan, which was utilized for this study. The only complication with these segmented masks, was that it was made based on the PET scans that has another dimension than the DESS scans, hence a resampling of these bone masks was needed. The resampling was done using HOROS, and the result of it is seen in figure 10 for 3 slices. Due to the resampling from a smaller space to a bigger space with more slices, the mask in some of the slices did not fit well at all (seen in the first plot of figure 10), and generally the bone mask was not perfect on the DESS scans. Nonetheless, in relative to the time saved not having to manually draw the mask for all scans, the bone mask were still applicable for the further work.

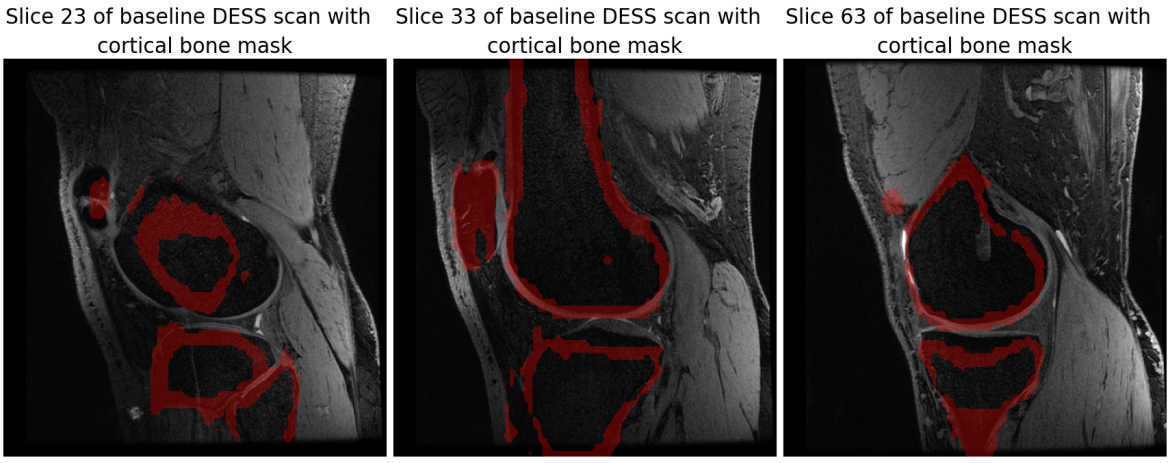

Figure 10: 3 slices of baseline DESS scan with subchondral bone mask

5.2.3 *Cartilage and adjacent bone voxel*

Utilizing the masks of the cartilage and subchondral bone, presented above in section 5.2.2, it was feasible to voxel-wise find the adjacent bone voxel for the cartilage voxel. The procedure of how this was found, is explained in more depth in the data processing section 4.2, but basically the way the code is written to search for these voxel, one cartilage voxel can have several adjacent bone voxel. Additionally a thin region between the cartilage and bone mask will be left out of this search for the adjacent voxel, essentially because the mask of cartilage and bone are not completely perfect as already mentioned. To be sure that the cartilage and bone voxel are indeed what they are assigned to be in this search, this gap between was created. All of these assigned cartilage and adjacent bone voxel, were for each slice of the scans saved as a mask, to be able to visualise exactly which part of the knee would be further analysed. An example of these masks can be seen for 3 slices of the baseline DESS scan in figure 11. In the first plot of this figure, there are no visual mask of cartilage and adjacent bone voxel, being that no adjacent bone voxel were found for any of the cartilage voxel for this slice of the knee scan. The major cause for this is the fact that the mask of the cartilage and mainly the bone is not correct and seems like they are not close enough to each other to meet the criteria for assigning the bone voxel as adjacent for this slice (this criteria is described in the data processing section 4.2), thereby leading the search of the adjacent bone voxel to find nothing. The 2 other plots in figure 11 shows very fine what is defined as the cartilage voxel (cyan mask) and the adjacent bone voxel (purple mask). In this search mostly the deep cartilage was assigned as the cartilage voxel, as mentioned before this region of the cartilage receives the most loading of the exercises.

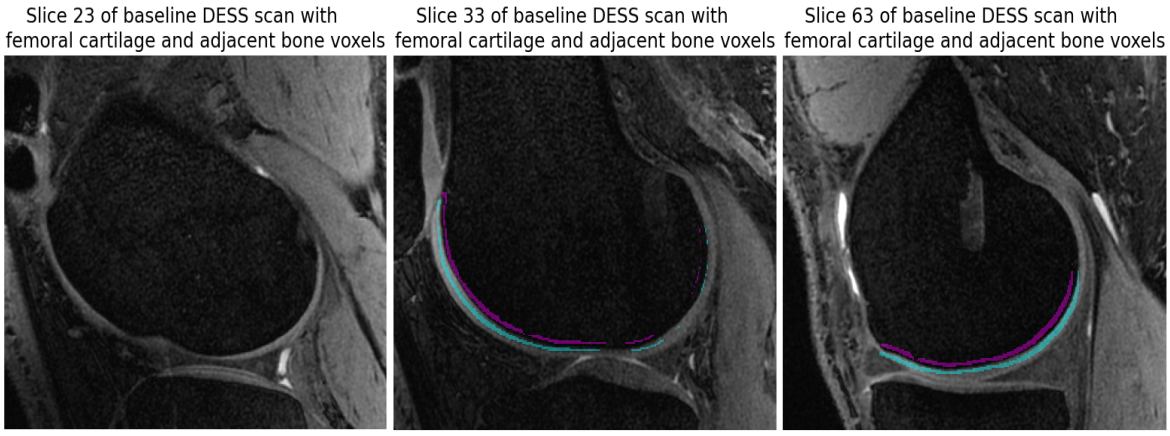

Figure 11: 3 slices of baseline DESS scan with mask of femoral cartilage (cyan) and adjacent subchondral bone (purple) voxel

5.2.4 *Calculated T_2 map*

The T_2 maps of the DESS scans were calculated using also DOSMA. DOSMA had the options of calculating T_2 map with suppression of fat, fluid and both at the same time. It was also possible to do it without suppression. The calculation with suppression resulted in T_2 maps, where major parts of the cartilage were missing, meaning that they had T_2 values equal to zero. The calculation without any suppression had some noise, but this was fairly easy to sort out as the T_2 values of cartilage is well known. For this reason the T_2 map calculation without any suppression was chosen for the further work. An example of this calculated T_2 map can be seen in figure 12 for one slice of a DESS scan. Her the T_2 map is plotted both by it self without the sorting of the noise (left plot), and with the sorting of the noise and just the femoral cartilage part plotted on top of the corresponding DESS scan (right plot). On the right plot it is clearly seen that the T_2 values of the femoral cartilage tissue are between $30 - 50$ ms, mostly around 35 ms which is what the expected T_2 value should be for baseline cartilage in knees.

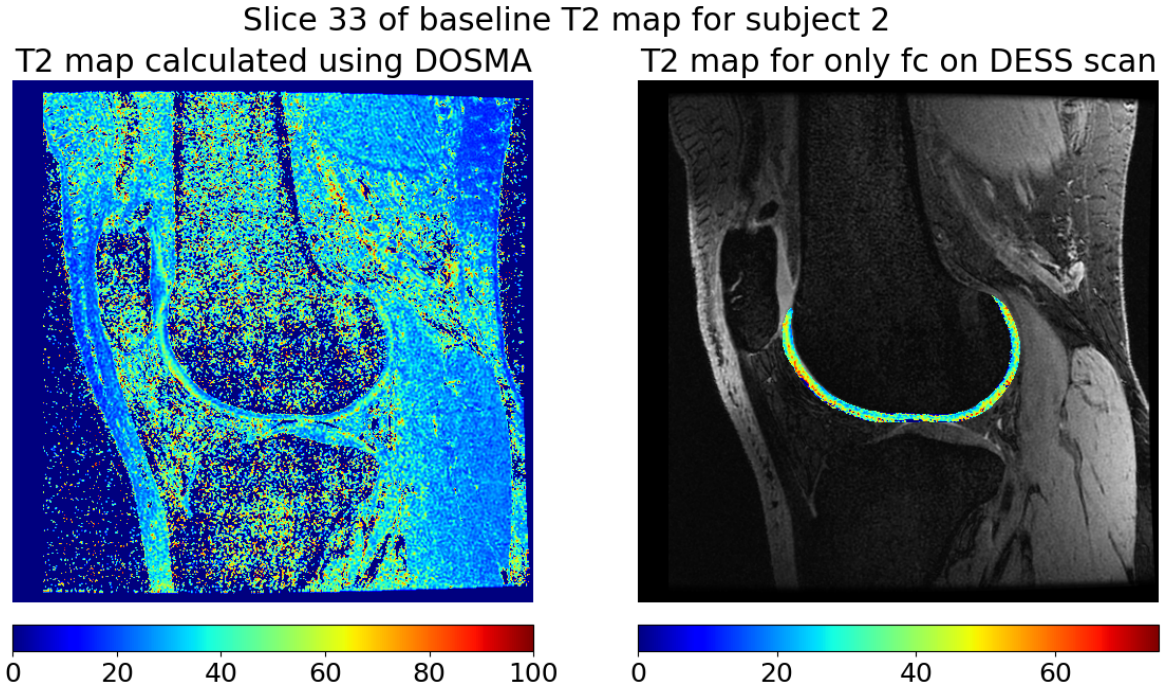

Figure 12: One slice of the calculated T_2 map for baseline DESS scan (Left: Calculated T_2 map & Right: Only the femoral cartilage part of the T_2 map on DESS scan)

5.2.5 *Registration of PET and $T_1\rho$ map*

The $T_1\rho$ map and PET scans have another dimension then the DESS scan, and to be able to compare the values between these 3, they have to have the same dimension. Thus a registration was done on the $T_1\rho$ map and PET scans to the DESS space. An example of the registration results for 3 slices of baseline PET scan and $T_1\rho$ map, can be seen in figure 13 and 14, plotted on top of the corresponding DESS scan slice.

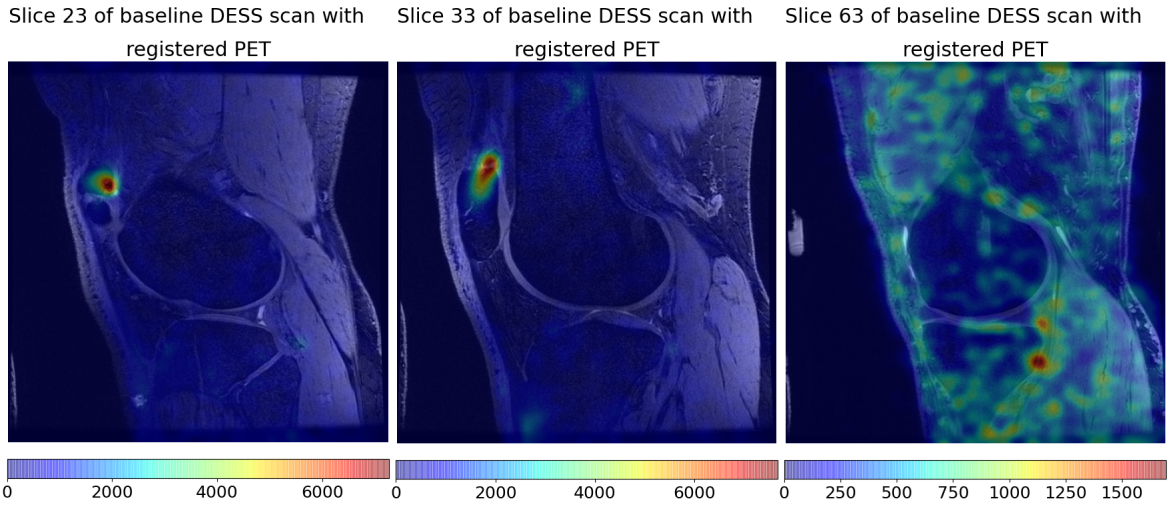

Figure 13: 3 slices of baseline DESS scan with registered PET scan

The pixel values from a PET scan, it is the NaF uptake, can vary from 0 to over 6000 as visualised in figure 13 and especially in the first 2 plots of this figure. Aforementioned the PET scan picks up even very small amounts of the NaF uptake, which gets absorbed by soft tissues as well, and this can be seen specifically in the last plot. Due to this large variation of NaF values in just one slice, it is more challenging to evaluate how well the registered PET scan agrees with the corresponding DESS scan even when plotted together. The highest values of NaF uptake in figure 13 can be seen at the patella, agreeing well with the PET scan presented in figure 4, although not being fully perfect but still acceptable. The particular high values of NaF uptake in the patella makes it difficult to distinguish the lower values in rest of the knee.

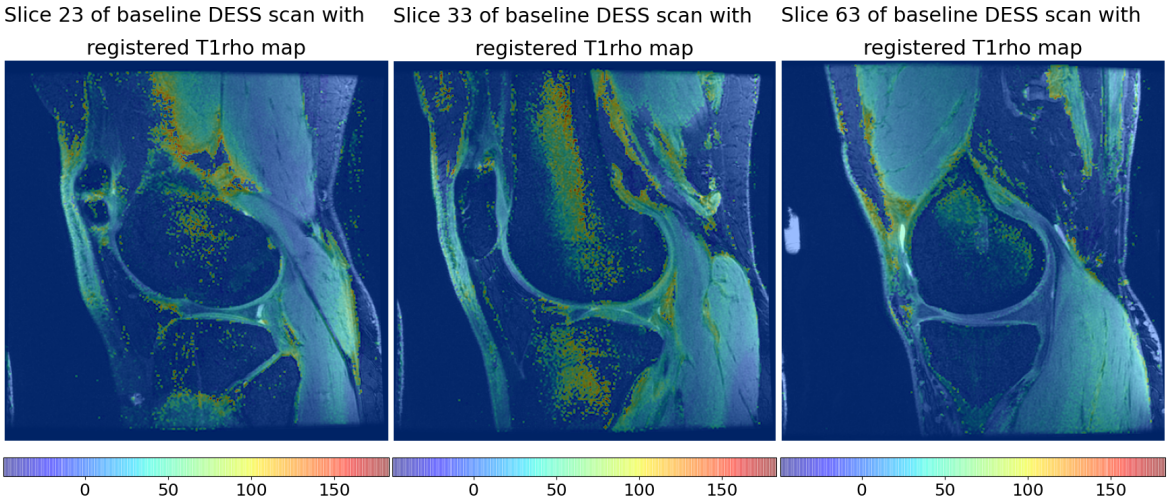

Figure 14: 3 slices of baseline DESS scan with registered $T_1\rho$ map

The pixel values of a $T_1\rho$ map does not vary as much as for the PET scan. Accordingly it is more distinct to see the different features of the knees in the registered $T_1\rho$ map and how well it agrees with the according DESS scan, which is visualised in figure 14. The ROI for these maps are of course the femoral cartilage, therefor when evaluating which registration method (rigid, affine or B-spline registration) has the best outcome, mainly this region was focused on. On some of the slices it may seem like the registration is perfect (the first plot in figure 14), while on other slices it may seem like the registration is not perfect and displaced (the second plot in figure 14), but the final decision on which registration method has the best outcome was chosen on an overall basis of the registered $T_1\rho$ map, and thereby used for the further processing and analysis.

5.3 DATA ANALYSIS

This section will present, the analysis of the data obtained from the medical image processing (example of this was presented in the previous section, for one of the subjects baseline data). More precisely, the data that will be analysed in this section, is the T_2 and $T_1\rho$ values of the femoral cartilage and the SUV of the adjacent bone from the baseline and post exercise PET/MRI scans. The main purpose of this analysis, will be to first of all examine whether it is possible to measure a change in the cartilage and adjacent bone of the knees after experiencing a load, and thereafter to study a possible correlation between these changed values of the cartilage and the adjacent bone, allowing to investigate the hypothesis of this thesis.

5.3.1 *Cartilage T_2 values*

To analyse whether it is possible to measure a correlation between the femoral cartilage and the adjacent subchondral bone after an acute loading of the knee, it requires that a change in these regions of the knee can be measured from the PET/MRI scan. For the femoral cartilage the change should be measured in the T_2 and/or the $T_1\rho$ values. First the change between the pre and post T_2 values will be analysed for the different femoral cartilage regions, being that the exercise executed by the subjects for loading the knee, should affect the femoral cartilage regions differently. To get a visualization of the obtained T_2 data from the different regions of the femoral cartilage for all 12 subjects pre and post exercise, a boxplot was made for each regions data separately. The exercise affects, not only the various regions within the femoral cartilage (the different regions are visualized in fig. 9) individually, but also the right and left knee experience and impact of diverse strength as aforementioned. For this reason, boxplots for the T_2 data of left and right knee are plotted separately into two different figures. In figure 15 a plot of the boxplots for the different regions of the femoral cartilage for the left knee can be seen plotted for pre and post exercise data and similarly for the right knee in figure 16. To make it easier to visualize, an image of the the cartilage regions (slice 33 of baseline DESS scan with the femoral cartilage region mask from fig. 9) is plotted right under the x-axis of the boxplots, pointing to the corresponding regions.

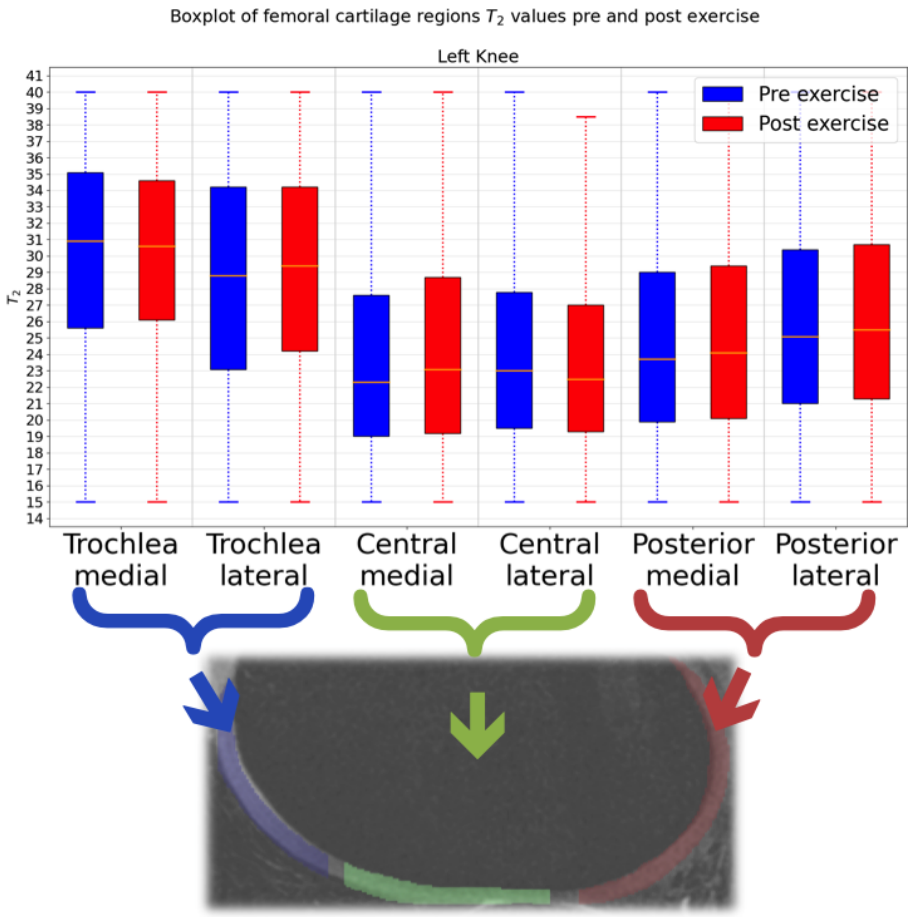

Figure 15: Boxplot to visualize the distribution of the T_2 values for the different regions of femoral cartilage, pre and post exercise for the left knee.

Apart from being able to see how the distribution of the T_2 values are for the different regions, it can also be utilized to examine whether there is a visual and significant change between the pre and post exercise T_2-values for all of the subjects data. In both figure 15 and 16 the pre and post T_2 data for each regions were plotted next to each other, making it more visual whether there is a possible change after the loading of the knee. The blue boxplots indicates the distribution of the T_2 data for all subjects pre exercise scans, and the red indicates post exercise. The boxplots for both the left and right knee, first and foremost displays clearly that the interquartile range or the midspread (middle 50%) of the data for the different regions of femoral cartilage does have a significant different T_2 range, thus confirming that the different regions of the femoral cartilage experience the loading individually. The trochlea regions seems to have a significant higher T_2, compared to the central and posterior regions. The posterior has a slightly higher range than the central regions. The more crucial analysis of the data distribution is to observe whether the boxplots indicates a significant change from pre to

post scans. Looking at both the left and right knees boxplots, it does not seem like the first, second (median) or third quartile have a significant difference from pre to post data for any of the regions. Moreover when comparing each regions left and right knee boxplots to each other, there is only a very small difference, which does not appear to be significant enough assuming that the left and right knees sustain a distinctive impact after the loading exercises. The median, that is the center of the data (assuming the data is normally distributed, median should be relatively the same as the mean of the data) is around the same for the medial and lateral for the various regions, implying that overall the exercise is considerably impacted balanced over the medial and lateral regions. The data is more skewed towards the lower values for the central and posterior regions, whereas for the trochlea it is skewed towards the higher values. This might be an indication that the data is not entirely normally distributed or could also be because the sample size for the different regions are not the same.

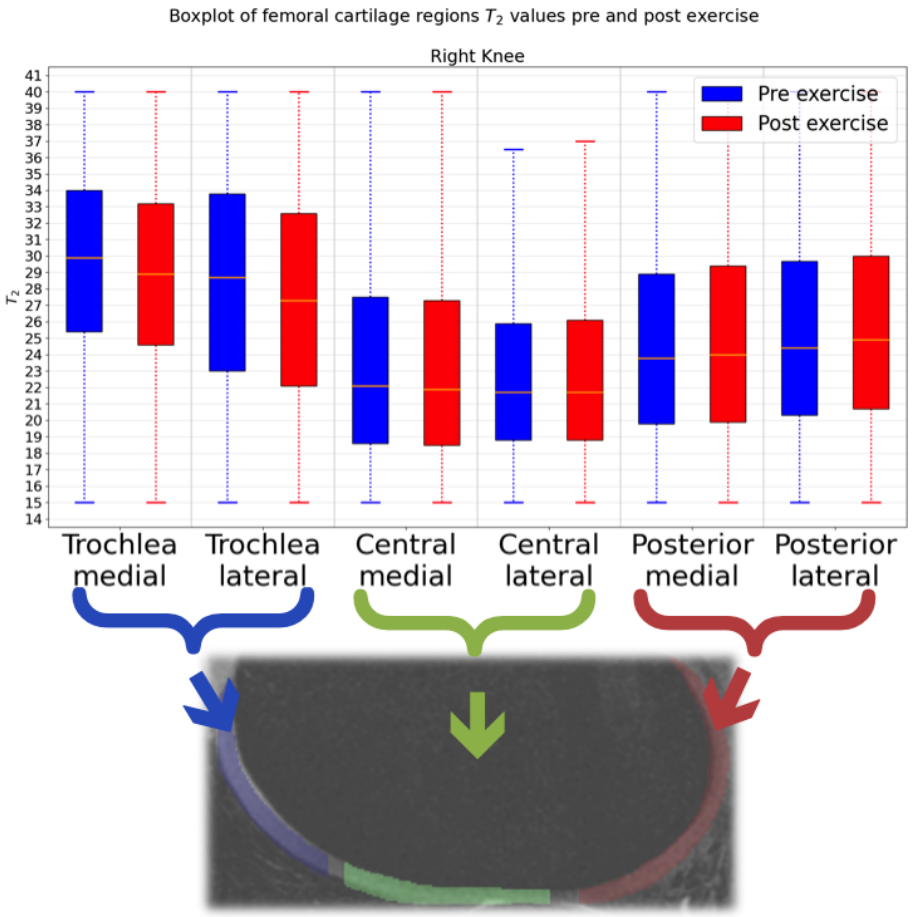

Figure 16: Boxplot to visualize the distribution of the T_2 values for the different regions of femoral cartilage, pre and post exercise for the right knee.

All these things considered, utilizing boxplots to visualize the distribution of the different regions T_2 data from pre & post exercise scans for left & right knees, illustrates that there certainly is measured a varying loading between the trochlea, central and posterior femoral cartilage regions, but no significant change after loading is measured, nor any difference between the left and right knees distinctive loading affects. This could be as a result of the subjects having different individual femoral cartilage T_2 values, that affects the data distribution. A way of visualising the data, taking this influence in consideration, is to plot the data for each subjects knees separately, but doing this for every data point would be too confusing to understand. This is why each subjects left and right knees separate mean T_2 values, for the different regions of femoral cartilage was plotted and can be seen in figure 17. Each of the 3 femoral cartilage regions have different markers and the medial and lateral have accordingly different colors, making it easier to visualise the data points (the legend in the lower right corner, displays which marker represent which region data). The way the plot is designed, having pre exercise mean T_2 values on the x-axis and the corresponding post exercise mean T_2 values on the y-axis, makes it clear how much the T_2 mean values of each subjects' separate knee has changed for the various regions, including the whole femoral cartilage (indicated with the yellow stars). The diagonal dashed black line, signifies the values where pre is equal to the post T_2 values, so the further away the data points are to this line, the larger a change after the loading is measured.

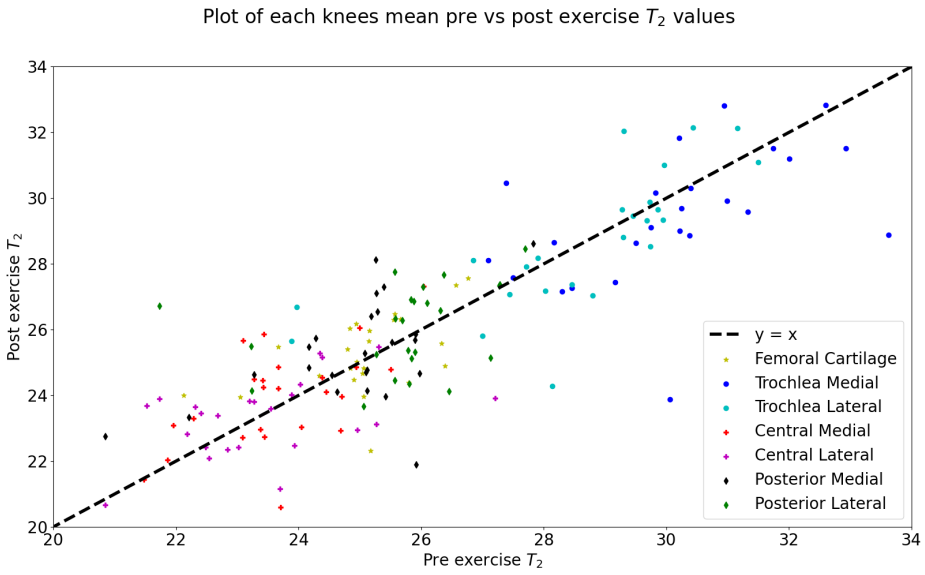

Figure 17: Plot of the mean T_2 values for each subjects different femoral cartilage regions (both left and right knee separate) pre versus post exercise.

First of all, it is visible that the data points are placed close to the line, corresponding with the observation made from the boxplots from figure 15 and 16, that there is no significant change of the T_2 value after the exercise. Additionally the plot shows how the data for the different regions are located in groups agreeing with the boxplots, central has lower T_2 values than the posterior, which has lower values than the trochlea region's T_2 values. These analysis are based on more visualising the data and thereby investigating the hypothesis, but to analyse the data and get a more quantitative answer, for whether there is measured a change in the cartilage, a statistical hypothesis test can be utilized. This type of analysis was done on the T_2 data, utilizing the Generalized Linear Mixed-Effects (GLME) model, that has the benefit of being able to analyse the relationship, between independent and dependent variables with the use of coefficients that could be varying with respect to one or more grouping variables [28]. The grouping variables for this data set can be the subject, time, region and the knee. After formatting the input data to the table data type (each row of the table represent one observation and each column represents one predictor variables), and using the build in fitting function *"fitglme"* in MATLAB, the relationship between pre and post T_2 mean and median values (since the data seemed to be not fully normally distributed, both the mean and median was probed) were analysed with the GLME model.

P-values for the T_2	**Mean**		**Median**	
difference pre vs post	**Left**	**Right**	**Left**	**Right**
Femoral	0,17	0,47	0,21	0,55
Trochlea medial	0,64	0,17	0,56	0,20
Trochlea lateral	0,42	0,31	0,43	0,25
Central medial	0,16	0,81	0,20	0,98
Central lateral	0,55	0,70	0,63	0,67
Posterior medial	0,46	0,63	0,51	0,54
Posterior lateral	0,45	0,58	0,48	0,57

Table 1: Table of p-values for the T_2 difference between pre and post exercise. The p-values were calculated for both the T_2 mean and median values for left and right knee separately.

The p-value was achieved from this analysis of the change for each regions mean and median T_2 values, which can be seen in table 1. A p-value ≤ 0.05 (5% level of significance) indicates that the null hypothesis should be rejected and thereby the alternative hypothesis, indicating

that there is a significant relationship between the 2 studied variables, will be accepted as the observed relationship is not a result of random cause. But what can be observed in table 1, is that none of the regions T_2 mean or median values have a significant change, considering all of the p-values are above 0.05, confirming the observations from the boxplots and mean plot above.

5.3.2 *Cartilage $T_1\rho$ values*

The change of $T_1\rho$ values for the cartilage regions were also analysed, in the same way the T_2 values were in the previous section. In figure 18 and 19 the data distribution of the different femoral cartilage regions can be seen plotted as boxplots for the left and right knee separate. Again the pre and post data for each regions are plotted next to each other, to make it more visible to see whether a change has been measured.

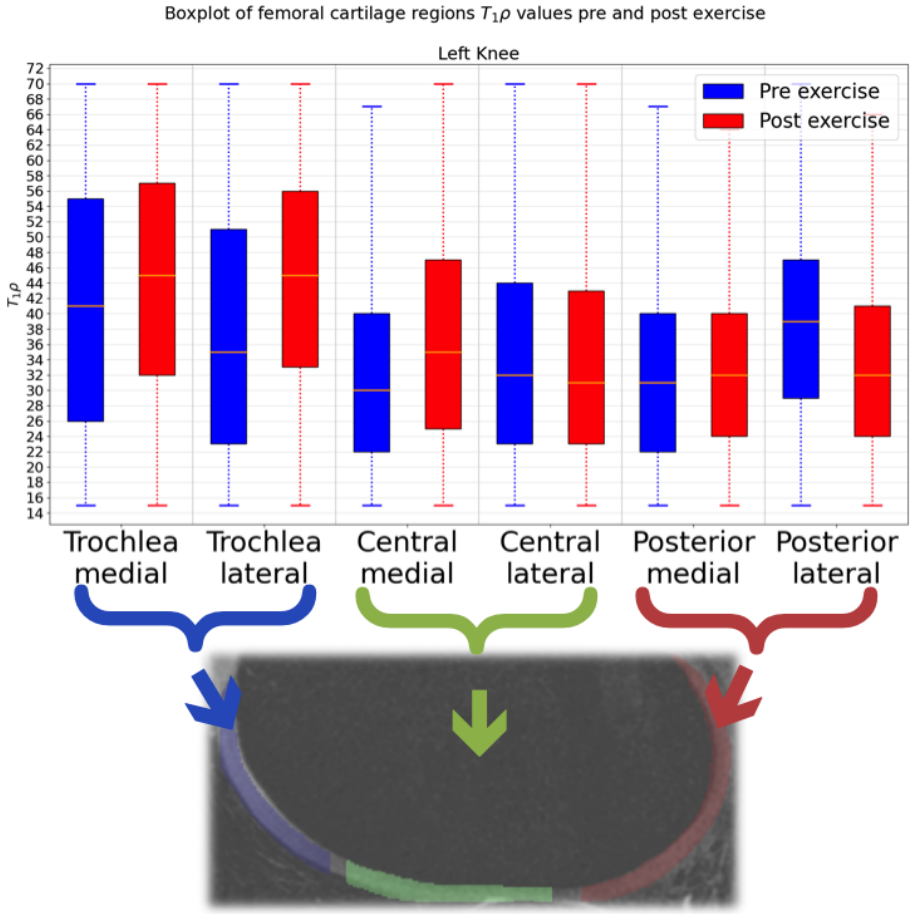

Figure 18: Boxplot to visualize the distribution of the $T_1\rho$ values for the different regions of femoral cartilage, pre and post exercise for the left knee.

For most of the regions, the data distribution suggest that a change indeed has been measured between the pre ad post $T_1\rho$ values. For the left knee all regions except central lateral and posterior medial, can be observed to have a considerable change after the loading. The change is positive in the trochlea and central regions, whereas for the posterior it is negative. Furthermore the interquartile range for the trochlea region is bigger than for the other regions. For the right knee all regions except trochlea medial and posterior medial, have a notable change, but the change is positive in the medial regions and negative in the lateral. The change being either negative or positive might be a sign, that it could be caused by randomness and a chance, rather than being a significant change, even tough that for the right knee the direction of the change appears to be dependent on whether the region is medial or lateral. In contrast to the T_2 data boxplots, the $T_1\rho$ shows a difference between the medial and lateral regions, suggesting that the exercise has not a balanced impact over the medial and lateral regions.

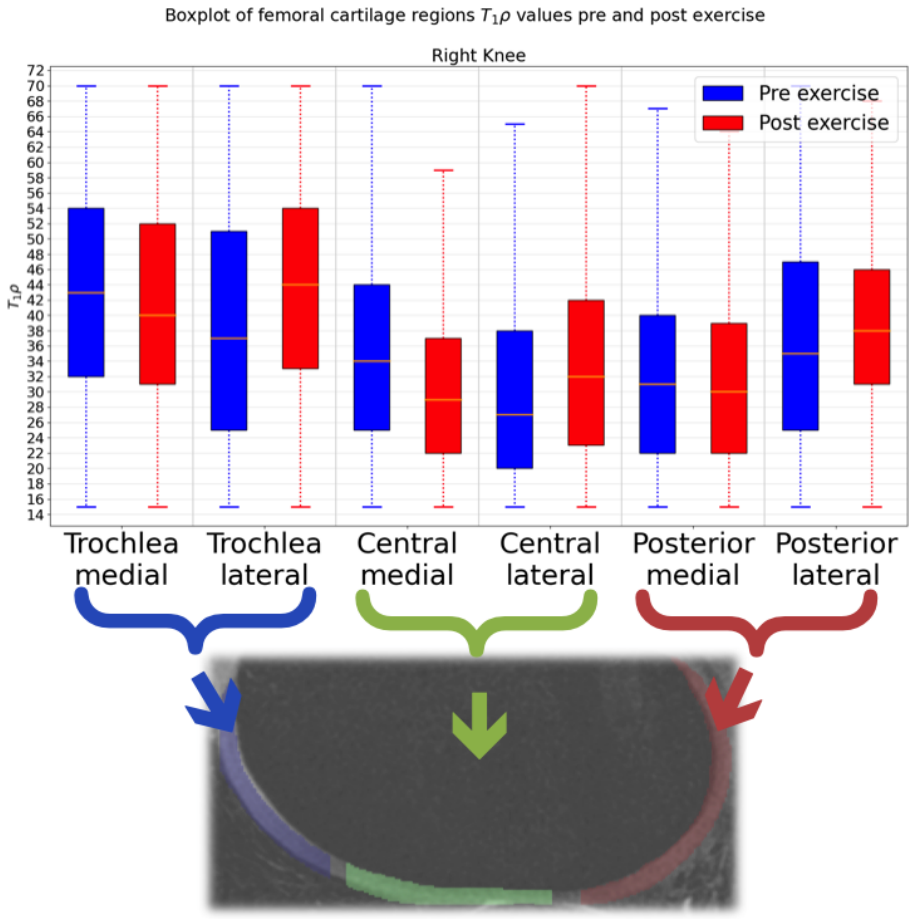

Figure 19: Boxplot to visualize the distribution of the $T_1\rho$ values for the different regions of femoral cartilage, pre and post exercise for the right knee.

The clear difference between the 3 regions of the T_2 data, can not be observed as clearly for the $T_1\rho$, but it is still observable agreeing with the T_2 data, that the loading impacts the 3 different regions individually. Identical to the T_2 data, the $T_1\rho$ data distribution is skewed towards the lower values for the central and posterior regions, but for the trochlea it seems to be more normally distributed and thereby has not that much of a skewness. When comparing the data for the left knee with the right knee, a slight difference can be observed for each regions, which was not observable for the T_2 data. In figure 20, a similar plot to what can be seen in figure 17, is plotted for the $T_1\rho$ data. In this plot it is distinctly seen, that each subjects left and right knees separate mean $T_1\rho$ values, for the different regions of femoral cartilage are placed further away from the diagonal line, by that indicating a change of the $T_1\rho$ value after exercise. Thus both the boxplots and the mean $T_1\rho$ plot implies that the data shows a change in the cartilage after the loading of the knee has been measured, although the data is placed on both upper and lower side of the diagonal. To be sure this is not caused by chance and randomness in the data, utilizing the GLME model, the relationship between pre and post $T_1\rho$ mean and median values were analysed with a hypothesis test.

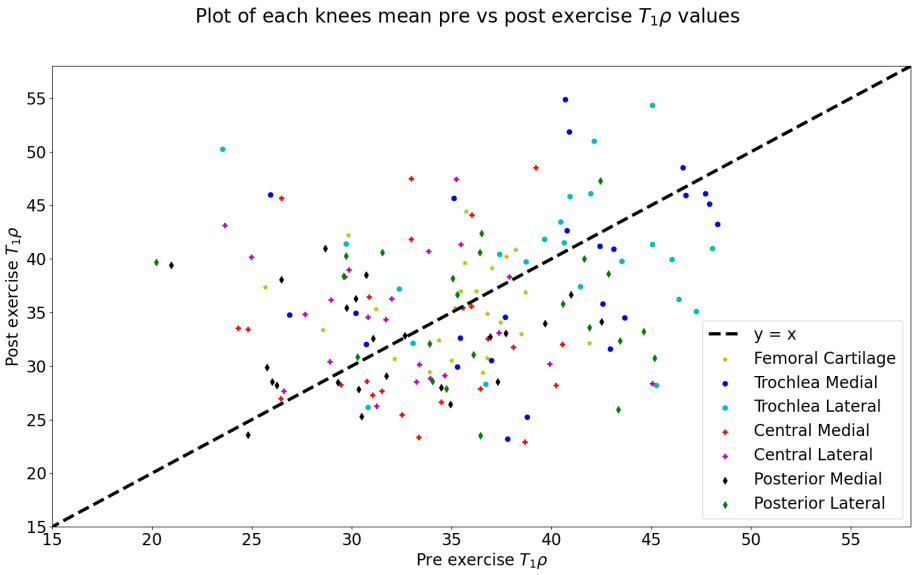

Figure 20: Plot of the mean $T_1\rho$ values for each subjects different femoral cartilage regions (both left and right knee separate) pre versus post exercise.

In table 2 the acquired p-values from the GLME model fitted to the $T_1\rho$ data can be seen visualised. The only region having a p-value under the 5% level of significance, is the central lateral specifically for the right knees. All the other regions have p-values above 0.05, thereby disproving that any significant change is measured between the pre and post $T_1\rho$ mean or median values for these regions. The right knee did get the highest impact, which could be the reason why the central lateral region in this knee has a measurable significant change after the exercise. But generally comparing the p-values to the values from table 1, they are much larger indicating that the data distribution or relationship is closer to be caused by randomness, rather than a statistically significant change in $T_1\rho$.

P-values for the $T_1\rho$ difference pre vs post	**Mean**		**Median**	
	Left	**Right**	**Left**	**Right**
Femoral	0,76	0,88	0,80	0,80
Trochlea medial	0,79	0,51	0,72	0,69
Trochlea lateral	0,93	0,99	0,91	0,87
Central medial	0,22	0,11	0,25	0,27
Central lateral	0,92	0,05	0,95	0,02
Posterior medial	0,55	0,82	0,53	0,70
Posterior lateral	0,12	0,47	0,11	0,64

Table 2: Table of p-values for the $T_1\rho$ difference between pre and post exercise. The p-values were calculated for both the $T_1\rho$ mean and median values for left and right knee separately.

The relationship between the left and right knees $T_1\rho$ mean and median values, were also analysed to study whether the difference observed in the boxplots were indeed significant. The analysis was done for the pre and post data separately, which should show no difference for the pre data. For the post data the analysis was also done for the delta $T_1\rho$ (the percentage difference between pre and post $T_1\rho$ mean values) in addition to the mean an median. The p-values for this analysis can be seen in table 3, which shows that a significant difference between left and right knee has been measured for only the central lateral region of pre $T_1\rho$ data and for the posterior lateral region of post $T_1\rho$ data. All other regions seems to have no difference between the two knees. Altogether for the $T_1\rho$ data a significant change has only been measured for the right knee central lateral, but this still does not seem to be convincing enough to be used for further analysis.

P-values for the $T_1\rho$ difference left vs right knee	Pre		Post		
	Mean	Median	Mean	Median	Delta
Femoral	0,7	0,56	0,48	0,56	0,39
Trochlea medial	0,80	0,75	0,54	0,67	0,38
Trochlea lateral	0,56	0,45	0,57	0,64	0,73
Central medial	0,08	0,19	0,23	0,30	0,04
Central lateral	0,02	0,01	0,92	0,90	0,17
Posterior medial	0,83	0,77	0,52	0,45	0,45
Posterior lateral	0,87	0,77	0,03	0,05	0,44

Table 3: Table of p-values for the $T_1\rho$ difference between left and right knee for pre and post exercise. For pre exercise the p-values were calculated for both the $T_1\rho$ mean and median values, whereas for the post exercise scans the p-values were also calculated for delta $T_1\rho$.

5.3.3 *Subchondral bone SUV*

Even though no significant and favorable change was measured in the cartilage T_2 or $T_1\rho$ values, the change of the adjacent subchondral bone was still analysed, to investigate whether the loading of the bone at least was measurable through the PET scan. As mentioned in the Theoretical background under the PET section, the ^{18}F-NaF uptake values acquired from the PET scan, were converted to Standardized Uptake Value (SUV) instead, as this would give a more normalized value with respect to each subjects body weight, which is more appropriate to use for the analysis. A similar plot, with the SUV data distribution for each femoral cartilage regions adjacent subchondral bones boxplot can be seen in figure 21 for the left knee and in figure 22 for the right knee. The first very apparent observation made from both plots, is that there is a very significant change from pre to post data. The upper quartile of the pre data boxplots have about the same value as the lower quartile of the post data boxplots, confirming that the change for all of the regions are positive, meaning that the SUV is measured to be higher after the loading of the bone. The size of the interquartile range for the post data being wider compared to the pre data, illustrates that each subject have a more diverse SUV after the exercise, indicating that the affect of the exercise in the bone can be individually different. This can be visualized better by plotting each subjects SUV mean values separately for each knee, as seen in figure 23.

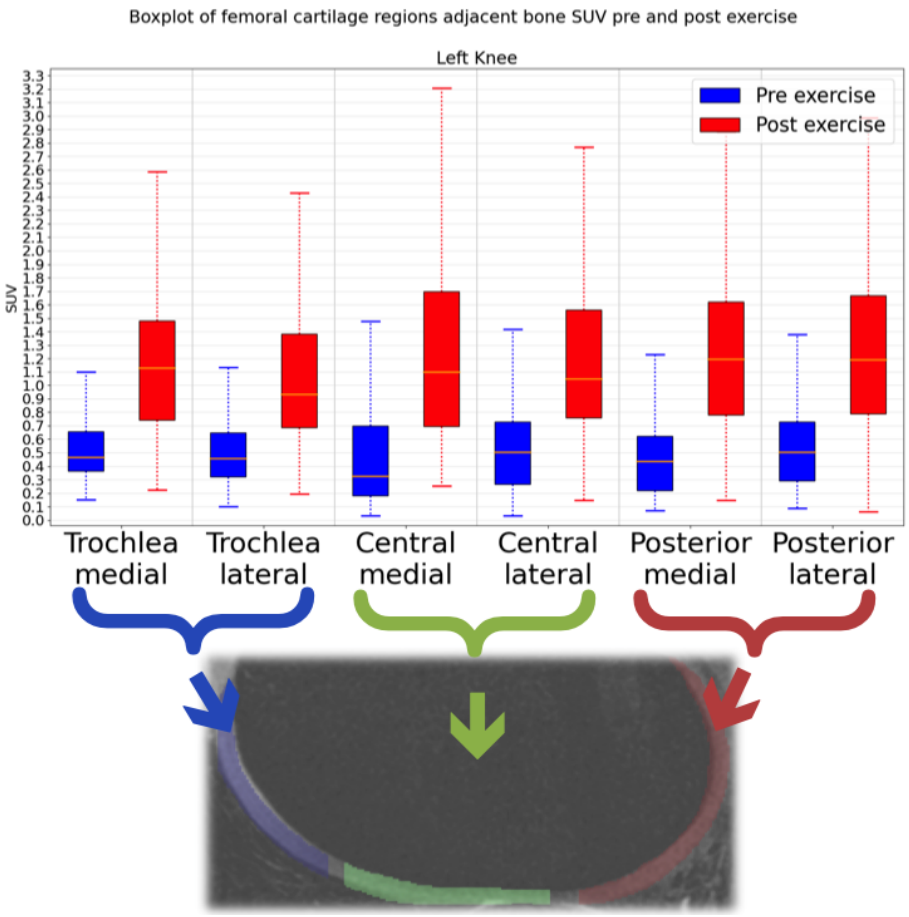

Figure 21: Boxplot to visualize the distribution of the SUV for the subchondral bone adjacent to the different regions of femoral cartilage, pre and post exercise for the left knee.

The plot of the mean SUV for each subjects subchondral bone adjacent to the different femoral cartilage regions, shows the same as the boxplot, a significant change in the positive direction. All of the data points (except for one single point placed very close to the diagonal) are placed on the upper side of the diagonal line. Further a tendency is showing that the higher the pre SUV are, the higher the post SUV are, illustrating that their might be a linear relationship between the change of the pre and post SUV data. The different regions does not appear to have distinct ranges, like what was observed in figure 17 for the T_2 data. The SUV data points for each regions are more widely spread amongst each other. Since no significant difference between the left and right knee, seems to be shown in the boxplots, only the pre and post relationship were analysed utilizing the GLME model for the SUV data.

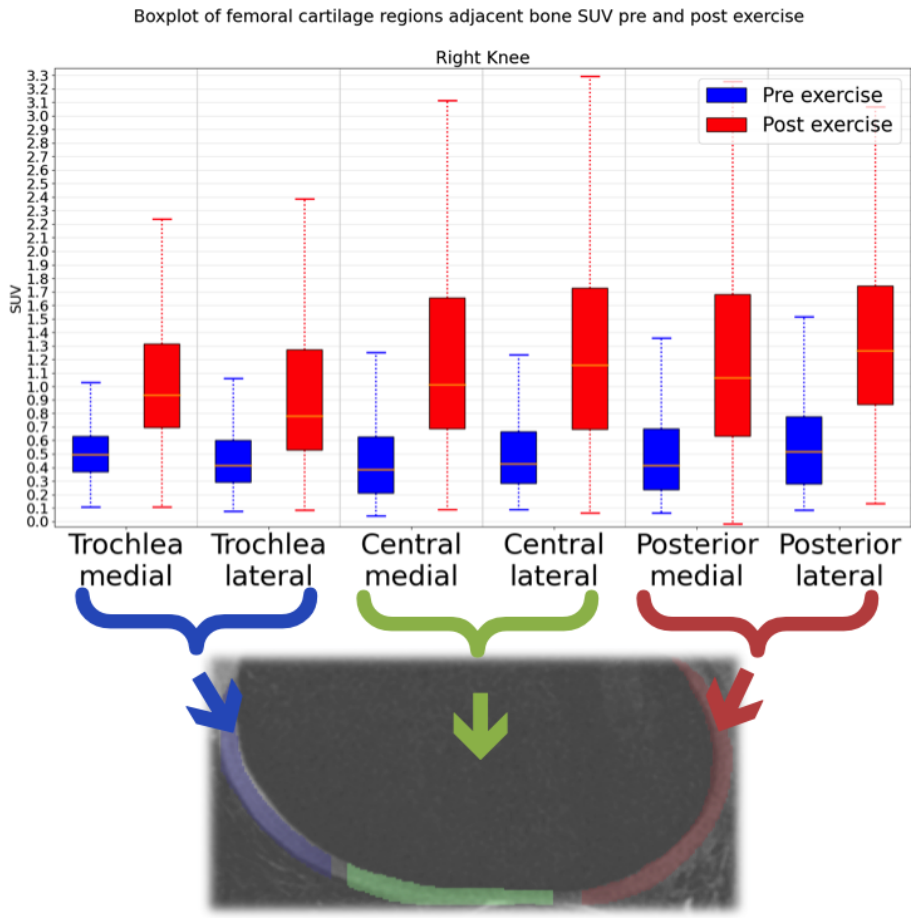

Figure 22: Boxplot to visualize the distribution of the SUV for the subchondral bone adjacent to the different regions of femoral cartilage, pre and post exercise for the right knee.

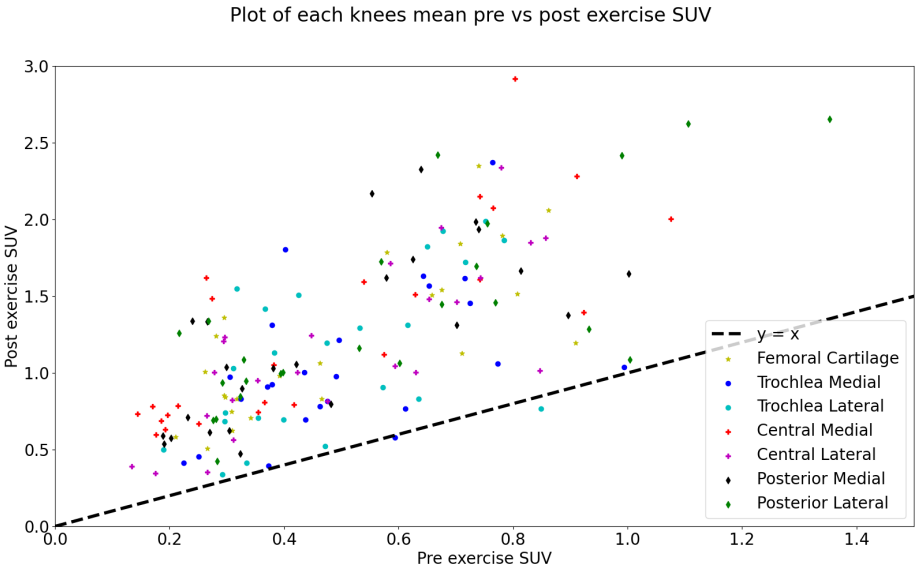

Figure 23: Plot of the mean SUV for each subjects subchondral bone adjacent to the different femoral cartilage regions (both left and right knee separate) pre versus post exercise.

The p-values from this analysis can be seen in table 4, for both the mean and median SUV and for the left and right knees separately. These values agree very well with the observation made so far for the SUV data. All of the regions have a very small p-value, ensuring that the change between pre and post SUV is very significant for all the adjacent bone regions.

P-values for the SUV	Mean		Median	
difference pre vs post	Left	Right	Left	Right
Femoral	$0{,}54e{-}9$	$0{,}14e{-}10$	$0{,}24e{-}9$	$0{,}47e{-}10$
Trochlea medial	$0{,}82e{-}7$	$0{,}15e{-}5$	$0{,}24e{-}6$	$0{,}12e{-}5$
Trochlea lateral	$0{,}92e{-}7$	$0{,}23e{-}8$	$0{,}43e{-}7$	$0{,}12e{-}7$
Central medial	$0{,}81e{-}9$	$0{,}11e{-}10$	$0{,}31e{-}9$	$0{,}49e{-}11$
Central lateral	$0{,}48e{-}8$	$0{,}13e{-}11$	$0{,}80e{-}9$	$0{,}12e{-}11$
Posterior medial	$0{,}16e{-}8$	$0{,}47e{-}9$	$0{,}33e{-}8$	$0{,}28e{-}8$
Posterior lateral	$0{,}77e{-}9$	$0{,}39e{-}8$	$0{,}82e{-}9$	$0{,}59e{-}8$

Table 4: Table of p-values for the SUV difference between pre and post exercise. The p-values were calculated for both the SUV mean and median values for left and right knee separately.

Even though a significant change has been measured in the bone SUV after the knee loading, because it was not possible to measure a significant change in the femoral cartilage T_2 nor $T_1\rho$ values, with the data processing and data analysis executed in this thesis, it is not reasonable to further analyse a possible correlation of the loading affect, between the cartilage and adjacent bone. Instead a more optimized protocol for detecting a change in the cartilage should be investigated, for then being able to in the future investigate the correlation between the cartilage and bone.

6

PARTIAL CONCLUSION

The purpose of the PET/MRI Bone-Cartilage Interaction - Stanford Study was primarily to study whether it is possible to measure a correlation between the response to loading of the cartilage tissue and the adjacent bone tissue. But in order to be able to measure this correlation, it requires that a response in terms of a significant change in the cartilage tissue values and the adjacent bone tissue values can be measured.

The analysis of the processed data showed a statistically significant change, and thereby response, of the loading in the adjacent subchondral bone tissue SUV, but the same significant response was not obtained in terms of a change in the femoral cartilage tissue T_2 , nor $T_1\rho$ values. Thus, it would not be reasonable to further examine a correlation of the response to loading between the knee cartilage and adjacent bone. It would be more appropriate to study an improved MRI protocol, for measuring a change in the cartilage tissue after a knee joint loading. This is exactly what led to the following MRI Cartilage T_2 Change - Glostrup Study.

Part III

MRI CARTILAGE T_2 CHANGE - GLOSTRUP STUDY

7

METHODS

7.1 STUDY DESIGN

The aim of this MRI cartilage T_2 change - Glostrup Study, is mainly to investigate an optimized MRI protocol for measuring the response to loading of the knee joints in terms of a change in the T_2 value of the femoral cartilage regions. The reason for the unsuccessful detecting of a change in T_2 or $T_1\rho$ in the femoral cartilage from the Stanford Study, could be first and foremost because of the time span between the knee exercise and the acquisition of the post exercise scan, or because of the exercise not being enough impact to be registered through the MRI DESS scan, which also could lead to the third reason, that the particular MRI imaging protocol might not be the best suited for this measurement. Hence, the 3 main parameters that were probed in my own experiment, with the purpose of measuring a T_2 change in the femoral cartilage, was firstly a shorter time span between acquiring the post scan after executing the knee exercise, secondly a different type of knee exercise possibly impacting the knee joints more and finally, a different MRI acquisition sequence that is more suited for MRI T_2 mapping. The effect of Blood Flow Restriction (BFR) on T_2 values in the knee was also explored, by making the subjects wear a tight band around one of their legs during the exercises. The experimental acquisition of the data for this study was executed by me at Rigshospitalet Glostrup, by performing parametric MRI scans of T_2 for the knee joints of 5 subjects with healthy knee joints. For each subject, a baseline (pre) scan and two post scans were acquired of each knee separately. The 5 healthy subjects who participated in this study consisted of 3 females and two males. The design of this study will be described in more details in the following sections.

7.1.1 *Exercise protocol*

In between the pre and post MRI scan (the imaging protocol will be elaborated in the following section), the subjects performed an exercise for exposing the knee joints to loading, resulting in changes in the knee cartilage. The exercise protocol consisted of squats repeated 60 times continuously without any rest in between. The exercise was done with the subjects holding a 2 kg hand weight in front of their chest, to make sure the knee joints would be impacted enough to be measured in terms of a change in the T_2 values. In addition, the subjects were wearing a tight band around their right leg during the exercise, and during the acquisition of the first set (left and right knee scanned separately) of post scans. The second set of post scans were acquired after making the subjects loosen up the band. A series of images of the exercise protocol can be seen in figure 24.

Figure 24: The exercise protocol consisting of squats repeated 60 times carrying a 2 kg hand weight and with a tight band around the right leg.

7.1.2 *Imaging protocol*

The pre and post exercise MRI scans were executed on a 3 T Philips MRI system, with the subjects wearing a cardiac coil (no knee coils were available) over both knees. The Turbo Spin Echo (TSE) sequence was used for acquiring the MRI data with the following imaging parameters: TR/TE1/TE2/TE3/TE4/TE5/TE6 = 1500/10/20/30/40/50/60 ms, FOV = 149 × 48 mm, matrix = 188 × 161, slices = 8 , slice thickness = 2.5 mm, slice gap = 4 mm, slice orientation = sagittal, flip angle = 90° and scan time = 123 s.

7.2 DATA PROCESSING

The methods utilized for processing the MRI data, acquired by me at Rigshospitalet Glostrup, will be explained in this section. After exporting the scanned data from the MRI computer in DICOM format, HOROS was used to distinguish and then save the scan into the different sequences for each knees' scans as DICOM files, for then being able to load them into Python and MATLAB for the data processing.

7.2.1 *Segmentation of cartilage*

The AI-powered Python library for medical image analysis presently only supports 4 MRI signal acquisition methods (Quantitative DESS, CubeQuant, MAPSS and UTE Cones), none of which consist of the TSE method applied in this Glostrup study. Accordingly, DOSMA was not applicable for the usage of an AI-powered automatic femoral cartilage segmentation on the TSE knee scans. Considering that the data for this study only consist of 3 scans for each knee for 5 subjects, the segmentation was instead accomplished manually by employing a ROI drawing code in MATLAB (the MATLAB code can be found in the appendices section). After drawing the femoral cartilage mask manually directly on each slice for every TSE scan, the masks were saved as DICOM files. This was challenging at times, with the reason being, that in some slices of the knee, what was cartilage, bone, or other types of tissue, would appear to be indistinguishable from one another. But this difficulty could sometimes be solved by using the slices of the knees from the last echo time scans, where the cartilage tissue was more easy to distinguish from the other types of tissues. Later, the whole femoral cartilage masks were classified into 3 additional different regions - trochlea, central and posterior. This was also done manually in python, by choosing threshold column values for separating the 3 different regions of the femoral cartilage. For each TSE knee scan, the 8 slices were run through, in order to find the 2 column threshold values, that best fit all the slices. The 3 masks of the regions were then saved and written back to the manually drawn femoral cartilage DICOM files.

7.2.2 *Calculation of T_2 map*

For the Stanford study, the calculation of the T_2 maps were computed utilizing DOSMA, but again, since it does not support the TSE sequence, the T_2 map for the Glostrup study was calculated manually voxel-wise using python. The Turbo Spin Echo (TSE) signal acquisition method applied for acquiring the MRI data for this study measures 6 echoes over one TR, leading to the acquisition of 6 images from every TE for each of the 8 slices of the knee scans. The signals acquired from the TSE sequence are dependent on the 6 TE, which can be described with the following equation:

$$S(TE) = S_0 \cdot \exp -\frac{TE}{T_2} + C \tag{32}$$

The equation indicates that the signal from the same voxel of the scan decays exponentially with the increasing TE. Fitting the signals from the 6 different TE to equation 32, voxel-wise for every slice of the knee scans, the T_2 map can be calculated. To do this with a python code for each and every voxel for all subjects TSE knee scans would take a very long time. Therefore, the T_2 map was only calculated for the Region Of Interest (ROI), the femoral cartilage. The computed T_2 values had a range up to 100 ms, which is too high to be a femoral cartilage T_2 value, thus T_2 values over 80 ms were sorted away for further data analysis.

8

RESULTS

8.1 RAW DATA

This section of the results will present the data that I acquired to test a part of the hypothesis, that the response to loading of the knee joints are measurable, in terms of a change in the T_2 value of the femoral cartilage regions. Like the Stanford study, there were not major visible differences between the images of the baseline (pre-) and post-exercise scans of the knees, therefor the sample that will be presented in this section, is the baseline data for one subject. The change in response to the exercise is apparent quantitatively, which will be presented in the Data analysis section (8.3). The figures in this section are screenshots of the data viewed by means of the medical image viewer Horos.

8.1.1 *MRI TSE Scan*

Figure 25 is one slice of the baseline MRI TSE scans acquired from all 6 echo times (TE: 10, 20 30, 40, 50 & 60). Since the knees where scanned separately, the images shown in figure 25 are only scans of the right knee of subject 3 from my own study. It is clearly seen in these scans, how the images of the same slice of knee gets more dark and little less detailed with each increasing echo time. This is exactly what was expected, as the signal should be decreasing for each echo time. Using the scans from all echo times, the T_2 map can then be calculated.

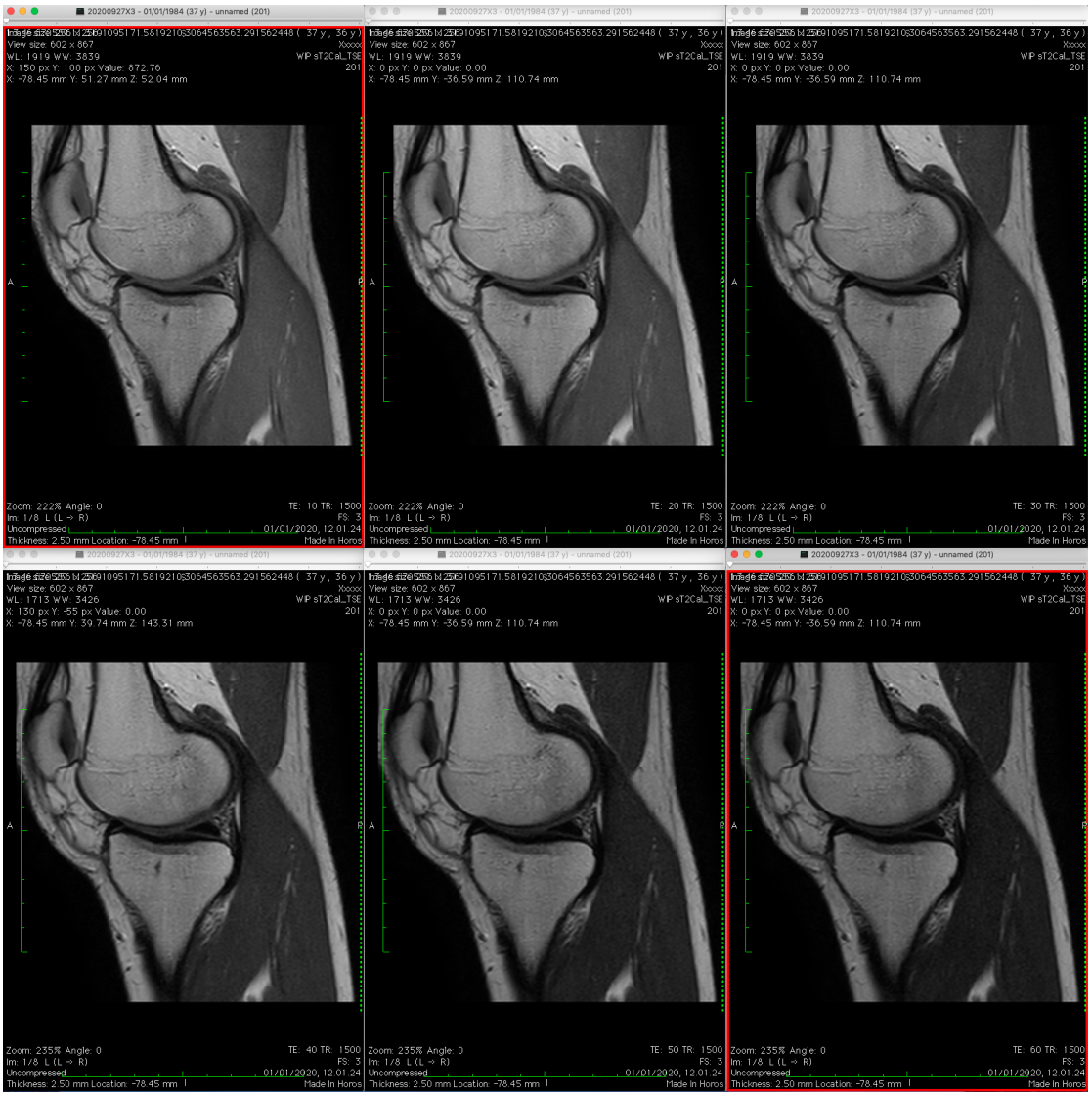

Figure 25: Baseline MRI TSE knee scans of subject 3 from all 6 echo times for right knee (First and second row of scans from left to right are scans from echo time: 10, 20, 30, 40, 50 & 60)

8.2 PROCESSED DATA

The data presented in this section, are the results of the medical image processing of baseline raw data for one subject from the MRI cartilage T_2 change - Glostrup Study (the data presented in the previous section). All of the figures in this section and the following sections, are screenshots of the data plotted in Python (the Python code for these results, can be found in the Appendices section).

8.2.1 *Segmentation of cartilage*

The segmentation of femoral cartilage for the TSE scans from this study, were drawn manually by myself for every slice across all the scans. As for the Stanford study, the femoral cartilage was classified into femoral trochlea, central and posterior regions. However, it was not classified into medial and lateral regions as well, since the classification was done manually as described in the data processing section 7.2, and the way the TSE scans were planned during the acquisition of the data, makes it too complicated to manually classify the knee into medial and lateral regions. Therefore, the femoral cartilage for this study was only classified into the three different regions mentioned above. An example of this classification of the different regions in the femoral cartilage can be seen in figure 26 for 3 slices of one knee TSE scan. The femoral cartilage region mask indicates the trochlea, central and posterior region, with respectively the dark blue, green and red mask. These femoral cartilage region masks were then utilized for the further data analysis of the femoral cartilage T_2 values.

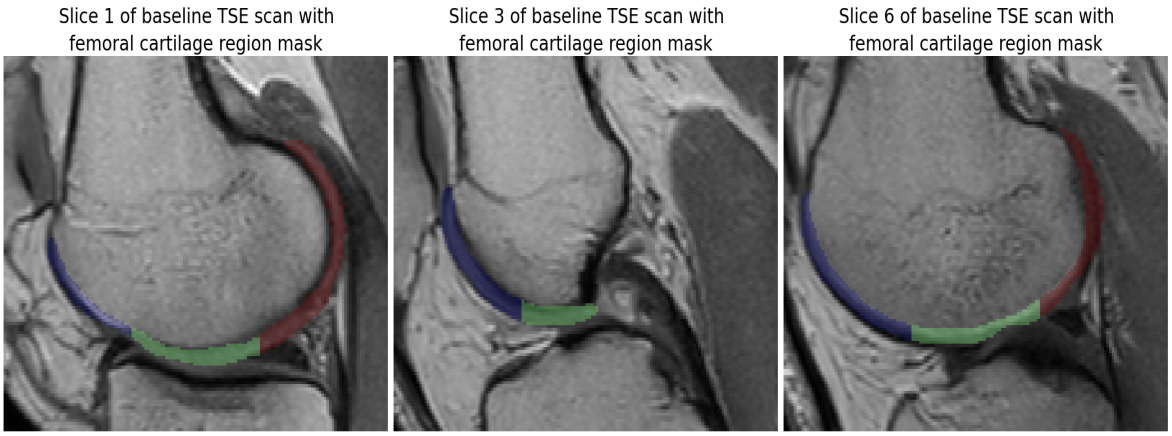

Figure 26: 3 slices of baseline TSE scan with femoral cartilage region mask (Dark blue: Trochlea, Green: Central, Red: Posterior)

8.2.2 *Calculated T_2 map*

The T_2 maps for this study where calculated using the TSE scans from each of the 6 echo times as mentioned prior, and as described in detail in the data processing section 7.2. On the grounds of the way the T_2 values were calculated voxel wise, the T_2 map was only calculated for the region of interest, the femoral cartilage, in order to save calculation time. 3 slices of a calculated T_2 map for the femoral cartilage can be seen visualised in figure 27, plotted on top of the corresponding TSE scan. Taking an overall look at these T_2 maps, it can be observed that the T_2 values for this region is calculated to be primarily between 30 to 60 ms, which is a range somewhat higher than the values calculated for the scans from the Stanford study, seen in figure 12. This could be because of the way the T_2 values were acquired and calculated differs for these 2 studies. What can also be seen in this figure, is that the areas where the T_2 values are over 80 ms, could be other types of tissue than cartilage, meaning that the femoral cartilage region mask has not been drawn entirely correct. For this purpose, and also because these are too high to be cartilage T_2 values, the values over 80 ms were sorted away for further data analysis. In this way it will be even more certain, that the data that will be further analysed is indeed cartilage values.

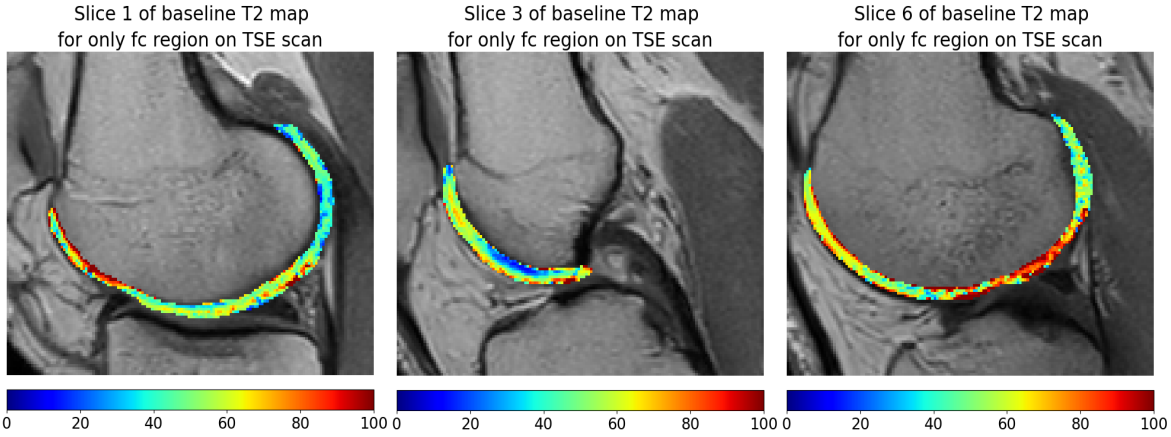

Figure 27: 3 slices of baseline T2 Map for only the femoral cartilage region on TSE scan

8.3 DATA ANALYSIS

The data presented in this section, is the analysis of the T_2 values of the femoral cartilage from the baseline and post exercise MRI scans. The main purpose of this analysis will be to determine whether it is possible to measure a change in the T_2 values of the knee cartilage, thereby allowing us to investigate a part of the hypothesis of this thesis.

8.3.1 *Cartilage T_2 values*

The distribution of the T_2 data acquired from the Glostrup study is visualized in similar manor to the Stanford study data, via boxplots which can be seen in figure 28 for the left knee, and figure 29 for the right knee. What can be seen plotted in these two figures, is the T_2 data from the pre, post 1 (data acquired immediately after the exercise) and post 2 exercise scans (data acquired after loosening the tight band around the right leg), indicated with the blue, red and green boxplots respectively, for each of the 3 separate femoral cartilage regions.

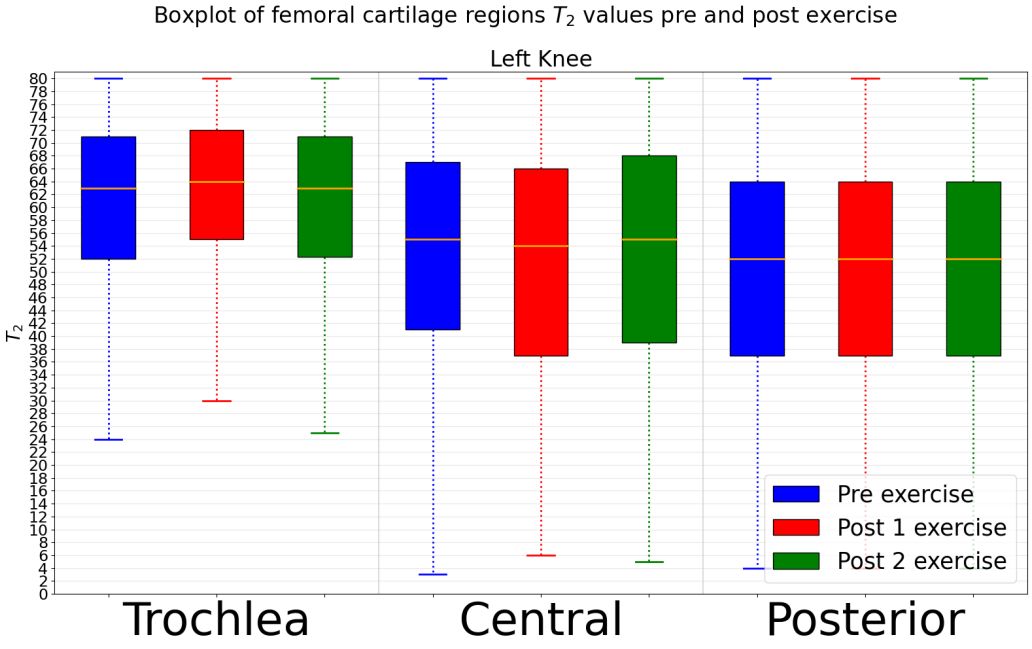

Figure 28: Boxplot to visualize the distribution of the T_2 values for the different regions of femoral cartilage, pre and post exercise for the left knee.

Firstly, it is very noticeable that the range of the T_2 data distribution in both figures (left and right knees) is considerably higher, when compared to the T_2 data from the Stanford study. Both the min, max and the interquartile range of the data are higher, which could come from

the significantly different method of acquiring and calculating the T_2 values for this study. Secondly, a fairly moderate difference in the T_2 data can be observed for the different regions of femoral cartilage, especially between the trochlea region and the 2 other regions. The Trochlea region also seems to have a smaller sized box, but placed in the higher T_2 values compared to the other two regions' data, indicating that the T_2 values of the trochlea regions are closer to each other and not as widespread. Further, data for all 3 regions are skewed towards the higher values. By generally looking at both knees' data, no significant difference can be observed between the pre and post data, except for the right knees' trochlea region, where a slight negative difference between the pre and post 1 data can be observed. The post 2 data appears to be around the same range and values of the pre data, suggesting that the T_2 values indeed do go back to the baseline T_2 value rather quickly. This implies that the time span between executing the knee loading with the exercise and the post scan, can be optimized further, for being able to measure the T_2 change in the cartilage.

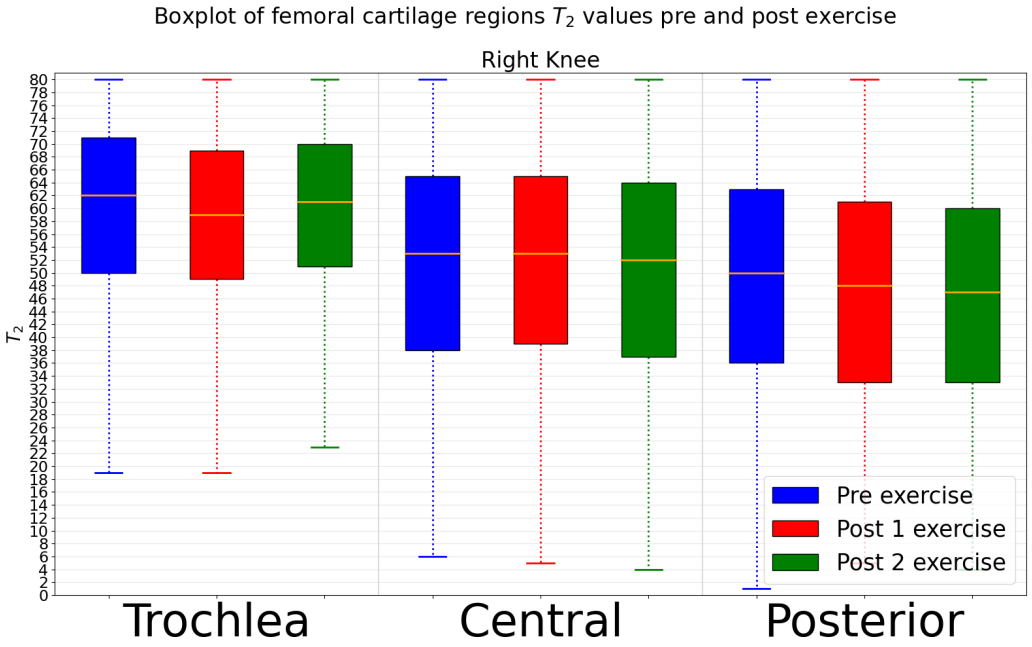

Figure 29: Boxplot to visualize the distribution of the T_2 values for the different regions of femoral cartilage, pre and post exercise for the right knee.

The Glostrup study experiment was designed to also investigate the effect of Blood Flow Restriction (BFR), which can be analysed by exploring a possible difference between the left and right knees' data, as the subjects were wearing a tight band on the right leg during the exercise and the first set of post MRI. Comparing the median values for the left with the right knees' data, a noticeable difference can be detected. This is when considering how the exercise for loading the knee (disregarding the tight band around on of the legs) should provide an equal impact on both knees, assuming that all test subjects performed the exercise in the correct manner, as visualized in the series of images showing the exercise protocol (fig. 24). Accordingly, this shows us that the BFR should have an impact on the measured T_2 values of the femoral cartilage. To be more sure that these observations made from the boxplots from figure 28 and 29 are significant, the hypothesis test utilizing GLME model was performed. Both the difference between the pre and post data and the difference between the left and right knee data was studied. The p-values from these hypothesis tests can be seen in the two tables below for both the whole femoral cartilage and for the different regions.

P-values for the T_2 difference	**pre vs post 1**				**pre vs post 2**			
	Mean		**Median**		**Mean**		**Median**	
	Left	**Right**	**Left**	**Right**	**Left**	**Right**	**Left**	**Right**
Femoral	0,82	0,50	0,51	0,21	0,96	0,17	0,81	0,20
Trochlea	0,44	0,05	0,43	0,07	0,50	0,69	0,55	0,74
Central	0,31	0,02	0,85	0,39	0,24	0,27	0,28	0,51
Posterior	0,94	0,79	0,89	0,79	0,99	0,34	0,95	0,22

Table 5: Table of p-values for the T_2 difference between pre and post 1 exercise (left part of table) and between pre and post 2 exercise (right part of table). The p-values were calculated for both the T_2 mean and median values for left and right knee separately.

In table 5 the p-values for the hypothesis test of a significant difference between the pre and post 1 T_2 mean and median values, can be seen in the left part of the table for left and right knee separately. The corresponding difference between pre and post 2 data can be seen on the right part of the table. No significant change of the T_2 values for any of the regions can be seen between the pre and post 2 data, confirming the observation made from the boxplots. However, a significant change between the pre and post 1 T_2 mean data can be seen for the right knee, which was the knee with the tight band on, in the trochlea and central regions.

This confirms the main aim for this study, that it is possible to measure the T_2 change in some parts of the femoral cartilage by optimizing the protocol, implying as well, that the BFR makes it easier to measure the change. These observations also align with the observed distribution of the T_2 values for at least the trochlea regions of the right knee. The p-values for the hypothesis test of a significant difference between the left and right knee can be seen in table 6. This table consists of the test for whether a difference has been measured between the left and right knee for the pre, post 1 and post 2 T_2 data. For the pre data, the relationship was analysed for T_2 mean and median values, whereas for the post 1 and post 2, it was also analysed for the T_2 delta values (the percentage difference between pre and post T_2 mean values).

	P-values for the T_2 difference between left and right knee							
	Pre		**Post 1**			**Post 2**		
	Mean	**Median**	**Mean**	**Median**	**Delta**	**Mean**	**Median**	**Delta**
Femoral	0,08	0,09	$0{,}18e{-}2$	$8{,}58e{-}5$	0,17	$0{,}19e{-}2$	$0{,}06e{-}2$	0,03
Trochlea	0,54	0,52	0,03	0,02	$0{,}08e{-}2$	0,51	0,39	0,81
Central	0,12	0,17	0,62	0,33	0,01	0,03	0,04	0,63
Posterior	0,56	0,46	0,08	0,04	0,26	0,07	0,03	$0{,}28e{-}2$

Table 6: Table of p-values for the T_2 difference between left and right knee for pre, post 1 and post 2 exercise. For pre exercise the p-values were calculated for both the T_2 mean and median values, whereas for the post exercise scans the p-values were also calculated for delta T_2.

For the pre data there are no significant differences between the left and right knee, which are the expected results, considering that the subjects should not have loaded the knees with any exercises or are not wearing the tight band yet. Looking at the whole femoral cartilage regions' p-value, for both the post 1 and post 2, there is a very significant difference between the left and right knee. When looking at the different regions' p-values, for post 1 data all regions have a significant difference between left and right knee, but for post 2 data the trochlea region seems to be the only region out of the 3, with no significant difference. This could be because the trochlea regions' T_2 values for the post 2 scan reached it's baseline value again. Generally, the results from the table 6 implies that it can be confirmed that the BFR band does impact the loading of the knee, which is measurable in the T_2 values of the femoral cartilage utilizing MRI.

Furthermore, the results from table 5, which are only showing a measurable significant change from pre to post 1 of the T_2 values in the knee wearing the tight band, confirms that the use of BFR can be very convenient for amplifying the load of the exercise in the cartilage regions, and by extension for rehabilitation of muscles around the joint, which is already being practised both clinically and at fitness centers.

9

PARTIAL CONCLUSION

The purpose of the MRI Cartilage T_2 change - Glostrup study, was essentially to study whether optimizing the MRI protocol could provide the possibility of measuring a change in the T_2 values of the femoral cartilage regions after exposing the knee joints to a loading. Furthermore, the effect of BFR was investigated, in order to explore whether it could be measured in the MRI T_2 values of the cartilage.

The analysis of the processed data displayed, that a significant change, and thereby response, of the loading in the trochlea and central femoral cartilage regions was measurable, but only in the knee with BFR band on. This not only confirms that it is possible to measure the response in cartilage, which couldn't be achieved in the Stanford study, but also that the BFR is usable for optimizing the measurements. The additional analysis of the BFR effect being measurable in the T_2 values of femoral cartilage was also confirmed, since a significant difference between the left (no BFR) and right (with BFR) leg was achievable.

Aside from being able to confirm a part of the thesis hypothesis, that the response of knee joint loading is detectable in the cartilage utilizing MRI, this study can also conclude that the MRI and exercise protocol can still be further optimized, for measuring an even more significant change in the cartilage. For future prospects then, this provides an opportunity for further investigation of the primary hypothesis of this thesis, which is concerned with whether a correlation between the change in the cartilage and adjacent bone tissues can be captured utilizing the dual modality PET/MRI imaging.

Part IV

DISCUSSION

10

DISCUSSION

The main goal of the thesis was to study the bone-cartilage interaction in loaded knee joints. More specifically, as written in the introduction, the following hypothesis was examined: ***With a loading of the knee joints, the cartilage tissues deform and the bone tissues react very quickly to adapt to this load, so consequently there must be a correlation between the change in these two kinds of tissues, which could be captured using PET/MRI imaging.***

The first study, PET/MRI Bone-Cartilage Interaction - Stanford Study, had the aim of investigating this hypothesis. By processing the ^{18}F-NaF PET/MRI medical images acquired by Stanford University and subsequently analysing the T_2 and $T_1\rho$ cartilage values and the SUV adjacent subchondral bone values, the main findings observed were, that the response to the knee joint loading was only significantly measurable in the adjacent bone. The failed measurement of the expected load response in the cartilage could have been caused by several different factors. First and foremost, the time span between executing the knee loading exercise and then acquiring the data, could pose a major influence in detecting a change in the cartilage regions. The cartilage values analysed for this thesis have a limited time interval before they restitute back to the baseline values, which may have then been exceeded for this experiment protocol. Shortening the time span between the exercise and the data acquisition could be a possible optimization, as well as a sequence with a shorter scan time. This leads to the second potential factor, which is the method utilized for acquiring the MRI data. A better suited sequence for measuring the response of loading in cartilage, could then constitute a possible optimization. In addition to the scan time being shorter, utilizing a data acquisition method with multiple echoes over a single or dual echo, could be more appropriate for this study. The third potential, and perhaps most obvious, factor, could be, that the type of exercise utilized did not present enough of an impact on the knee joints. The

exercise protocol followed in this study should impact the right knee with a longer and higher load directed more to the trochlea femoral cartilage region, whereas for the left knee, it was a shorter shock directed more to the central regions, and a smaller amount to the trochlea. This is to be expected, considering that the subjects completed the exercise correctly. Even though the response to the loading was not significantly measurable in terms of a change in the cartilage MRI values (especially the T_2 values), a significant difference between the range of the values for the different cartilage regions were measurable, agreeing with the expected amount of impact described above. So a different type of exercise, one that would provide more of an impact on the knee joints, could be an extra factor of optimization for measuring the response in cartilage.

These are all factors that could be optimized in the imaging and exercise protocols, both of which were designed by Stanford. What was more influenced by myself, was the medical image processing of the acquired data, performed before the indicators of a physiological response measure (T_2, $T_1\rho$ and SUV) were analysed. In this case as well, a number of factors could have influenced the failed detection of load response in the knee cartilage. Taking it from the first steps of processing, the segmentation of knee joint cartilage had a major limitation, since the utilized automatic segmentation with DOSMA was limited to only giving usable segmentation of the femoral cartilage. The patella and tibial cartilage would have been an interesting joint region to analyse as well, since these regions of cartilage are not as curved, and with the type of exercise executed, they are expected to have a higher and more uniform direct impact. So for future work, these two types of knee cartilage would provide an interesting perspective for further study of the hypothesis. Another part of the segmentation, which could have caused the complication of detecting the load response in the femoral cartilage was, that the bone segmentation was not flawless. The incorrect bone mask, for some of the slices, resulted in the failed detection of adjacent cartilage and bone voxels, as visualized in figure 11. This led to the loss of a relatively large amount of data for the further analysis. Correcting the bone mask could therefore provide an optimization for detecting the change in response in the knee joint. Lastly, the registration performed on both the PET and $T_1\rho$ map, could be improved as well, which may have resulted in the wrong $T_1\rho$ values being analysed for some of the cartilage regions, causing data disturbance. A fine tuning of the registration, potentially with types of methods other than, the rigid, affine or

B-spline registration, could yield a better result, thereby contributing to a more correct data analysis.

Considering that it was not possible to detect a significant response of knee loading in the cartilage, as with the adjacent subchondral bone, there was no groundwork for further analysis of the main hypothesis, when concerned with a possible correlation between the cartilage and adjacent bone loading response. Consequently, the second study, Cartilage T_2 change - Glostrup Study, was aimed to study an optimized imaging an exercise protocol, for being able to measure the response in knee joints in terms of a change in the T_2 values. By confirming that it is possible to detect the cartilage loading response with an optimized MRI protocol, it will provide the opportunity to work with the main hypothesis of this thesis in the future. The parameters probed in this study were primarily the factors discussed above: a shorter time span between knee exercise and knee scanning, another type of exercise for loading the knee joints with higher impact, and finally another MRI sequence for the T_2 data acquisition. Additionally, the effect of the well-known BFR was also studied, in order to confirm whether or not it can be measured in the cartilage as well. The main findings from this study was, that the response to knee loading was only detectable in the right knee with the BFR band on, and that a very significant difference was measured in the femoral cartilage T_2 values between the left and right knees. This confirmed, that the effect of BFR can not only be measured in the cartilage, but that it also seems to have helped to detect the loading response in terms of a change in the T_2 values, by amplifying the impact of the exercise on this particular knee. The change was still not as statistically significant as was the case for the adjacent bone from the Stanford study, showing that there is room for more optimization, when trying to detect a stronger statistically significant loading response. What could be investigated for further optimization, is different types of MRI sequences, that might be better suited for detecting the T_2 change in cartilage. On top of that, a more impacting and straining knee exercise, or even by utilizing the BFR, as it did show an amplification of the change, could also be a factor of optimization. Furthermore, what was not done in this experiment, was letting the subjects rest lying down on the MRI table before the baseline scan, making sure that the knee were not exposed to any strain or loading, as this could affect the measurement of the baseline T_2 values.

To summarize, the findings from this thesis work can be employed as groundwork for further investigating the best way to detect the knee loading response in cartilage simultaneously with bone. The next logical step would then be, to examine with the optimized protocol whether the dual modality PET/MRI can be utilized to detect a correlation between the cartilage and adjacent bone response to loading. If such a correlation can be detected, and thereby confirm the hypothesis of this thesis, what could be of interest would be to further examine the same for sick knees, afflicted with Osteoarthritis (OA). This will provide the possibility of comparing the detected correlation between the cartilage and adjacent bone response for healthy knees, with sick knees, leading to the next step into investigating a possible method for diagnosing early stages of OA. This could then be utilized to detect a joint dysfunction, that may be a new indicator, as mentioned in the introduction.

Part V

CONCLUSION

11

CONCLUSION

By analysing the T_2 and $T_1\rho$ cartilage values and the adjacent subchondral bone SUV, from a set of PET/MRI data acquired by Stanford University for pre and post exercise, **it was not possible to confirm the hypothesis of this thesis**, which stated that by using PET/MRI dual modality imaging, a correlation between the change in response of knee joint loading in cartilage tissues and adjacent bone tissues can be detected. This was mainly because the response of knee loading in cartilage was not significantly measurable, like the response in the adjacent bone was. But with the optimized MRI protocol and the effect of BFR, it was possible to measure a significant change in T_2 cartilage values as a response for knee loading, for the set of data acquired at Rigshospitalet Glostrup. The MRI protocol could still be further optimized, when seeking to measure a statistically stronger response in cartilage, simultaneously with the PET scan of the adjacent bone. Obtaining this in the future would lead to the prospect of further inspecting the main hypothesis of this thesis, stated above.

Part VI

BIBLIOGRAPHY

12

BIBLIOGRAPHY

ARTICLES

[1] Chen M, Qiu L, Shen S, et al. (2017 November). *The influences of walking, running and stair activity on knee articular cartilage: Quantitative MRI using T1 rho and T2 mapping*. PLOS ONE. 12(11): e0187008.

[2] Deshpande BR, Katz JN, Solomon DH, et al. (2016 December). *The number of persons with symptomatic knee osteoarthritis in the United States: Impact of race/ethnicity, age, sex, and obesity*. Arthritis Care Res (Hoboken). 68(12): 1743–1750.

[3] Guilak F. (2011 December). *Biomechanical factors in osteoarthritis*. Best Pract Res Clin Rheumatol. 25(6): 815–823.

[4] Haddock B, Fan AP, Jørgensen NR, et al. (2019 May). *Kinetic [18F]-Fluoride of the Knee in Normal Volunteers*. Clin Nucl Med. 44(5): 377-385.

[5] Haddock B, Fan AP, Uhlrich SD, et al. (2019 November). *Assessment of acute bone loading in humans using [18F]NaF PET/MRI*. Eur J Nucl Med Mol Imaging. 46(12):2452-2463.

[6] Hayashi D, Guermazi A, Kwoh CK (2014 January). *Clinical and translational potential of MRI evaluation in knee osteoarthritis*. Curr Rheumatol Rep. 16(1): 391.

[7] Hughes L,Paton B, Rosenblatt B, et al. (2017 Marts). *Blood flow restriction training in clinical musculoskeletal rehabilitation: a systematic review and meta-analysis*. British Journal of Sports Medicine. 51:1003-1011.

[8] Kogan F, Fan AP, McWalter EJ, et al. (2016 October). *PET/MRI of metabolic activity in osteoarthritis: A feasibility study*. J Magn Reson Imaging. 45(6): 1736-1745.

[9] Mamisch TC, Trattnig S, Quirbach S, et al. (2010 Marts). *Quantitative T2 mapping of knee cartilage: differentiation of healthy control cartilage and cartilage repair tissue in the knee with unloading - initial results*. Radiology. 254(3):818-26.

[10] Mittal S, Pradhan G, Singh S, et al. (2019 December). *T1 and T2 mapping of articular cartilage and menisci in early osteoarthritis of the knee using 3-Tesla magnetic resonance imaging*. Pol J Radiol. 84: e549–e564.

[11] Savic D, Pedoia V, Seo Y, et al. (2016 January). *Imaging bone-cartilage interactions in osteoarthritis using [(18)F]-NaF PET-MRI*. Mol Imaging. 15: 1–12.

[12] Thakkar RS, Flammang AJ, Chhabra A, et al. (2011 Marts). *3T MR Imaging of Cartilage using 3D Dual Echo Steady State (DESS)*. MAGNETOM Flash.

[13] Turner CH. (1998 November). *Three rules for bone adaptation to mechanical stimuli*. Bone. 23(5): 399-407.

[14] Varma DR. (2012 Jan-Mar). *Managing DICOM images: Tips and tricks for the radiologist*. Indian J Radiol Imaging. 22(1): 4–13.

[15] Waldenmeier L, Evers C, Uder M, et al. (2019 July). *Using Cartilage MRI T2-Mapping to Analyze Early Cartilage Degeneration in the Knee Joint of Young Professional Soccer Players*. Cartilage. 10(3): 288–298.

BOOKS

[16] Bushberg JT, Seibert JA, Leidholdt EM, et al. (2012). *The Essential Physics of Medical Imaging*. Third Edition. USA: Lippincott Williams & Wilkins.

[17] Cercignani M, Dowell NG, Tofts PS, et al. (2018). *Quantitative MRI of the Brain Principles of Physical Measurement*. Second Edition. USA: CRC Press Taylor & Francis Group.

[18] Haacke EM, Brown RW, Thompson MR, et al. (1999). *Magnetic Resonance Imaging Physical Principles and Sequence Design*. First Edition. USA: Wiley-Liss.

[19] Kiefer B (1998). *Turbo Spin-Echo Imaging. In: Echo-Planar Imaging*. First Edition. Germany: Springer-Verlag Berlin Heidelberg.

WEBSITES

[20] Chickscope (1996). *Chickscope Overview: MRI Introduction For High School Students*, [accessed 2021 July 27]. http://chickscope.beckman.illinois.edu/about/overview/mrihs.html

[21] DeepAI (2019). *Convolutional Neural Network*, [accessed 2021 July 27]. https://deepai.org/machine-learning-glossary-and-terms/convolutional-neural-network

[22] DICOM (1993). *About DICOM: Overview*, [accessed 2021 July 27]. https://www.dicomstandard.org/about

[23] DOSMA (2019). *DOSMA: Deep Open-Source Medical Image Analysis*, [accessed 2021 July 27]. https://dosma.readthedocs.io/en/stable/index.html

[24] Gigtforeningen (2016 February). *Hvad er slidgigt–artrose–og hvordan udvikler sygdommen sig?*, [accessed 2021 July 27]. https://www.gigtforeningen.dk/viden-om-gigt/diagnoser/slidgigt/hvad-er-slidgigt/.

[25] Gigtforeningen (2018 January). *Behandling af slidgigt–dine behandlingsmuligheder*, [accessed 2021 July 27]. https://www.gigtforeningen.dk/viden-om-gigt/diagnoser/slidgigt/behandling-af-slidgigt/.

[26] Gigtforeningen (2018 November). *Smerter i led, ryg og muskler koster os milliarder i tabt arbejdskraft*, [accessed 2021 July 27]. https://www.gigtforeningen.dk/for-pressen/pressemeddelelser/2018/2-halvaar/smerter-i-led-ryg-og-muskler-koster-os-milliarder-i-tabt-arbejdskraft/

[27] Healthline (2019 July). *What Is the Average (and Ideal) Percentage of Water in Your Body?*, [accessed 2021 July 27]. https://www.healthline.com/health/body-water-percentage.

[28] MathWorks (1994-2021). *Generalized Linear Mixed-Effects Models*, [accessed 2021 July 27]. https://se.mathworks.com/help/stats/generalized-linear-mixed-effects-models.html.

[29] Medical News Today (2018 August). *The stages of osteoarthritis of the knee*, [accessed 2021 July 27]. https://www.medicalnewstoday.com/articles/310579.

[30] Questions and Answers in MRI (2001). *Fast spin-echo*, [accessed 2021 July 27]. https://www.mriquestions.com/what-is-fsetse.html.

[31] Questions and Answers in MRI (2001). *Gyromagnetic Ratio (λ)*, [accessed 2021 July 27]. http://mriquestions.com/gyromagnetic-ratio-gamma.html.

[32] Questions and Answers in MRI (2001). *TR and TE*, [accessed 2021 July 27]. http://mriquestions.com/tr-and-te.html.

[33] Radiopaedia (2015). *Fast spin echo*, [accessed 2021 July 27]. https://radiopaedia.org/articles/fast-spin-echo.

[34] Radiopaedia (2015). *T1 rho*, [accessed 2021 July 27]. https://radiopaedia.org/articles/t1-rho.

[35] Scientific Explorer (2014 November). *Why Do Particles Have Spin?*, [accessed 2021 July 27]. http://sciexplorer.blogspot.com/2014/11/why-do-particles-have-spin_18.html.

[36] University of Ottawa NMR Facility Blog (2009 March). *What is T1 and How is it Measured?*, [accessed 2021 July 27]. http://u-of-o-nmr-facility.blogspot.com/2009/03/what-is-t-1-and-how-is-it-measured.html.

OPEN-SOURCE CODE - SOFTWARE

[37] Desai AD, Barbieri M, Mazzoli V, et al. (2019). *DOSMA: A deep-learning, open-source framework for musculoskeletal MRI analysis*. Proc. Intl. Soc. Mag. Reson. Med. 27(1106).

[38] Klein S, Staring M, Murphy K, et al. (2010 January). *elastix: a toolbox for intensity based medical image registration*. IEEE Transactions on Medical Imaging, vol. 29, no. 1, pp. 196 - 205.

Part VII

APPENDICES

A

PYTHON CODE - STANFORD STUDY

A.1 PRE PROCESSING

```
8
9   # Importing necersary packages:
10  #------------------------------
11  from __future__ import print_function
12  import matplotlib.pyplot as plt
13  from natsort import natsorted
14  import matplotlib as mpl
15  from pathlib import Path
16  import numpy as np
17  import pydicom
18  import os
19
20
21  # Setting som plotting standards:
22  #--------------------------------
23  font = {'weight' : 'normal', 'size' : 18}
24  mpl.rc('font', **font)
25
26
27  # Defining functions used in the code:
28  #-------------------------------------
29  # Function to collect all dcm files from a folder into lstFilesDCM:
30  def DCMFiles_List(DCM_Path):
31
32      # Create an empty list to collect all .dcm files into it;
33      lstFilesDCM = []
34
35      # Loop to traverse the directory & collect all dcm files into lstFilesDCM;
36      for dirName, subdirList, fileList in os.walk(DCM_Path):
37          # Sorting the fileList;
38          fileList = natsorted(fileList) # naturally sort
39          for filename in fileList:
40              # Check whether the file's DICOM;
41              if ".dcm" in filename.lower():
42                  # Storing files in list;
43                  lstFilesDCM.append(os.path.join(dirName,filename))
44
45      return lstFilesDCM
```

```
46
47
48  # Function to import raw DICOM data into an Pixel array:
49  def DCM_Import(DCM_Path):
50
51      # Collecting all dcm files from the folder into a list;
52      lstFilesDCM = DCMFiles_List(DCM_Path)
53
54      # Reference to extract metadata:
55      # Get refference file (first slice);
56      RefDs = pydicom.dcmread(lstFilesDCM[0])
57
58      # Load dimensions based on the number of rows, columns, and slices;
59      ConstPixelDims = (int(RefDs.Rows), int(RefDs.Columns), len(lstFilesDCM))
60
61      # Storring the raw DICOM data:
62      # Creating an array size is based on 'ConstPixelDims';
63      ArrayDicom = np.zeros(ConstPixelDims, dtype=RefDs.pixel_array.dtype)
64
65      # Creating array to store the Rescale Slope & Rescale Intercept for all slices;
66      RescaleSlope = np.ones(len(lstFilesDCM))
67      RescaleIntercept = np.zeros(len(lstFilesDCM))
68
69      # Loop through all the DICOM files;
70      for filenameDCM in lstFilesDCM:
71
72          # Read the file;
73          ds = pydicom.dcmread(filenameDCM)
74
75          # Storing raw data;
76          ArrayDicom[:, :, lstFilesDCM.index(filenameDCM)] = ds.pixel_array
77
78          # Correcting for Rescale tags if they exist;
79          if ("RescaleIntercept" in ds) == True:
80
81              RescaleSlope[lstFilesDCM.index(filenameDCM)] = ds.RescaleSlope
82              RescaleIntercept[lstFilesDCM.index(filenameDCM)] = ds.RescaleIntercept
83
84      # Correcting for the rescale tags;
85      ArrayDicom = ArrayDicom * RescaleSlope + RescaleIntercept
86
87      return ArrayDicom
88
89
90  # Function to remove Ghosting artifact and save to DCM files:
91  def DCM_Ghosting(DCM_Path, col1, col2):
92
93      # Collecting all dcm files from the folder into a list;
94      lstFilesDCM = DCMFiles_List(DCM_Path)
95
96      # Reference to extract metadata:
97      # Get refference file (first slice);
98      RefDs = pydicom.dcmread(lstFilesDCM[0])
99
100     # Load dimensions based on the number of rows, columns, and slices;
101     ConstPixelDims = (int(RefDs.Rows), int(RefDs.Columns), len(lstFilesDCM))
102
103     # Storring the raw DICOM data:
104     # Creating an array size is based on 'ConstPixelDims';
105     DESS_Ghost = np.zeros(ConstPixelDims, dtype=RefDs.pixel_array.dtype)
106
107     # Loop through all the DCM files to read-store-correct-write to files again;
108     for filenameDCM in lstFilesDCM:
109
110         # Read the file;
111         ds = pydicom.dcmread(filenameDCM)
112
```

```python
        # Store the raw image data;
        DESS_Ghost[:, :, lstFilesDCM.index(filenameDCM)] = ds.pixel_array

        # Correcting for Ghost Artefact (setting to zero);
        DESS_Ghost[:, col1:col2, lstFilesDCM.index(filenameDCM)] = 0

        # Saving/writing the modification back into the files again;
        ds.PixelData = DESS_Ghost[:, :, lstFilesDCM.index(filenameDCM)].tobytes()
        ds.save_as(filenameDCM)


# Class to plot and scroll trough the slices using key_press_event:
class IndexTracker:
    # Initial definitions for plot;
    def __init__(self, fig, ax, X, title):

        self.fig = fig
        self.ax = ax
        self.X = X
        rows, cols, self.slices = X.shape
        self.ind = 0

        self.im = ax.imshow(self.X[:, :, self.ind], cmap='gray')
        self.ax.set_title(title)
        self.ax.axis('off')
        self.update()

    # Key definition for navigation of slices;
    def on_key(self, event):

        if event.key == 'up':
            self.ind = (self.ind + 1) % self.slices
        elif event.key == 'down':
            self.ind = (self.ind - 1) % self.slices
        self.update()

    # Function to update plot;
    def update(self):

        self.im.set_data(self.X[:, :, self.ind])
        self.fig.suptitle('Slice ' + str(self.ind+1) + ' of baseline DESS scan for subject '
                          + subject[1:2])
        self.im.axes.figure.canvas.draw()

#-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-#

# Importing the uncorrected DESS scan:
#-----------------------------------
NaF = 'NaF_sub'
subject = '02A'
Patient_folder = NaF + subject

# Importing the DESS scans, using the function from above;
DESS_DCM = DCM_Import(Patient_folder + '/DESSLow/')


# Sizes of the imported DESS images;
Rows, Cols, Slices = DESS_DCM.shape

# Array to store the DESS scans without difussion (Echo 1);
E1_DCM = np.zeros((Rows, Cols, int(Slices/2)))


# Echo1, The odd no. images/slices [I0001.dcm,I0003.dcm,I0005.dcm,...]:
# Intialising index for divided array;
idx_1 = 0
for i in range(Slices):
```

```
    # Even no. in file list (incl. 0) will be stored in not diffused array;
    if (i % 2) == 0: # Even numbers in file list are odd no. images
        E1_DCM[:, :, idx_1] = DESS_DCM[:, :, i]
        idx_1 += 1


# Correcting for Ghost Artifact:
#-------------------------------
#Defining the threshold of columns that needs to be corrected;
col1 = 0
col2 = 65

#Definig value for the corrected columns to be set as;
gray = DESS_DCM.max()/10

# Loop to show where the Ghosting correction will be;
for s in range(E1_DCM.shape[2]):

    E1_DCM[:,col1:col2,s] = gray


# Displaying the uncorrected DESS scans:
#----------------------------------------
# Plotting DESS files;
fig1, ax1 = plt.subplots(1, 1, figsize=(14, 7))

track1_1 = IndexTracker(fig1, ax1, E1_DCM,
                        'DESS scans with columns that will be corrected')

fig1.canvas.mpl_connect('key_press_event', track1_1.on_key)

plt.show()

#-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-#

# Removing Ghosting artifact;
#---------------------------
# Using function to remove Ghosting artifact and save to DCM files:
#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#
DCM_Ghosting(Patient_folder + '/DESSLow_Correct/', col1, col2)
#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#


# Importing the corrected DESS scan and displaying:
#--------------------------------------------------
# Importing the DESS scans, using the function from above;
DESS_Corrected = DCM_Import(Patient_folder + '/DESSLow_Correct/')

# Sizes of the imported DESS images;
Rows, Cols, Slices = DESS_Corrected.shape

# Array to store the DESS scans without difussion (Echo 1);
E1_Corrected = np.zeros((Rows, Cols, int(Slices/2)))


# Echo1, The odd no. images/slices [I0001.dcm,I0003.dcm,I0005.dcm,...]:
# Intialising index for divided array;
idx_1 = 0
for i in range(Slices):
    # Even no. in file list (incl. 0) will be stored in not diffused array;
    if (i % 2) == 0: # Even numbers in file list are odd no. images
        E1_Corrected[:, :, idx_1] = DESS_Corrected[:, :, i]
        idx_1 += 1


# Plotting corrected DESS files;
fig2, ax2 = plt.subplots(1, 1, figsize=(14, 7))
```

```
247
248 track2_1 = IndexTracker(fig2, ax2, DESS_Corrected,
249                          'DESS scans that are corrected')
250
251 fig2.canvas.mpl_connect('key_press_event', track2_1.on_key)
252
253
254 plt.show()
255
256 #-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-#
257
258 # Displaying the original and the corrected DESS scans:
259 #------------------------------------------------------
260 # Plotting DESS E1 of original and corrected files beside each other;
261 fig3, ax3 = plt.subplots(1, 2, figsize=(14, 7))
262
263 track3_1 = IndexTracker(fig3, ax3[0], E1_DCM, 'Original DESS scan')
264 track3_2 = IndexTracker(fig3, ax3[1], E1_Corrected, 'Corrected DESS scan')
265
266 fig3.canvas.mpl_connect('key_press_event', track3_1.on_key)
267 fig3.canvas.mpl_connect('key_press_event', track3_2.on_key)
268
269 plt.show()
```

A.2 FEMORAL CARTILAGE ROI MASK

```
8  # Importing necersary packages:
9  #-------------------------------
10 from scipy.ndimage.morphology import binary_dilation, binary_erosion
11 import matplotlib.pyplot as plt
12 from natsort import natsorted
13 from scipy.io import loadmat
14 import matplotlib as mpl
15 from pathlib import Path
16 from scipy import stats
17 import nibabel as nib
18 import numpy as np
19 import pydicom
20 import h5py
21 import os
22
23
24 # Setting som plotting standards:
25 #--------------------------------
26 font = {'weight' : 'normal', 'size'   : 18}
27 mpl.rc('font', **font)
28
29
30 # Defining functions/classes used in the code:
31 #---------------------------------------------
32 # Function to import Nifti data into Pixel array:
33 def NIfTI_Import(Path, filename):
34
35     # Setting the path to the NIfTI file;
36     nii = os.path.join(Path, filename)
```

```
    # Loading and getting the pixel data for the NIfTI file;
    ArrayNIfTI = nib.load(nii).get_fdata()

    return ArrayNIfTI


# Function to collect all dcm files from a folder into lstFilesDCM:
def DCMFiles_List(DCM_Path):

    # Create an empty list to collect all .dcm files into it;
    lstFilesDCM = []

    # Loop to traverse the directory & collect all dcm files into lstFilesDCM;
    for dirName, subdirList, fileList in os.walk(DCM_Path):
        # Sorting the fileList;
        #fileList.sort()
        fileList = natsorted(fileList)
        for filename in fileList:
            # Check whether the file's DICOM;
            if ".dcm" in filename.lower():
                # Storing files in list;
                lstFilesDCM.append(os.path.join(dirName,filename))

    return lstFilesDCM


# Function to import raw DICOM data into an Pixel array:
def DCM_Import(DCM_Path):

    # Collecting all dcm files from the folder into a list;
    lstFilesDCM = DCMFiles_List(DCM_Path)

    # Reference to extract metadata:
    # Get refference file (first slice);
    RefDs = pydicom.dcmread(lstFilesDCM[0])

    # Load dimensions based on the number of rows, columns, and slices;
    ConstPixelDims = (int(RefDs.Rows), int(RefDs.Columns), len(lstFilesDCM))

    # Storring the raw DICOM data:
    # Creating an array size is based on 'ConstPixelDims';
    ArrayDicom = np.zeros(ConstPixelDims, dtype=RefDs.pixel_array.dtype)

    # Creating array to store the Rescale Slope & Rescale Intercept for all slices;
    RescaleSlope = np.ones(len(lstFilesDCM))
    RescaleIntercept = np.zeros(len(lstFilesDCM))

    # Loop through all the DICOM files;
    for filenameDCM in lstFilesDCM:

        # Read the file;
        ds = pydicom.dcmread(filenameDCM)

        # Storing raw data;
        ArrayDicom[:, :, lstFilesDCM.index(filenameDCM)] = ds.pixel_array

        # Correcting for Rescale tags if they exist;
        if ("RescaleIntercept" in ds) == True:

            RescaleSlope[lstFilesDCM.index(filenameDCM)] = ds.RescaleSlope
            RescaleIntercept[lstFilesDCM.index(filenameDCM)] = ds.RescaleIntercept

    # Correcting for the rescale tags;
    ArrayDicom = ArrayDicom * RescaleSlope + RescaleIntercept

    return ArrayDicom

```

```
104
105 # Class to plot and scroll trough the slices using key_press_event:
106 class IndexTracker:
107
108     # Initial definitions for plot;
109     def __init__(self, fig, ax, ind, X, alpha, cmap, title):
110
111         self.fig = fig
112         self.ind = ind
113         self.cmap = cmap
114         self.alpha = alpha
115         self.ax = ax
116         self.X = X
117         rows, cols, self.slices = X.shape
118
119         self.im = ax.imshow(self.X[:, :, self.ind], alpha=self.alpha,
120                             cmap=self.cmap)
121         self.ax.set_title(title)
122         self.ax.axis('off')
123         self.update()
124
125     # Key definition for navigation of slices;
126     def on_key(self, event):
127
128         if event.key == 'up':
129             self.ind = (self.ind + 1) % self.slices
130         elif event.key == 'down':
131             self.ind = (self.ind - 1) % self.slices
132         self.update()
133
134     # Function to update plot;
135     def update(self):
136
137         self.im.set_data(self.X[:, :, self.ind])
138         self.fig.suptitle('Slice ' + str(self.ind+1) + ' of baseline DESS scan with')
139         self.im.axes.figure.canvas.draw()
140
141
142 # Function to find the first slice of the segmentation that is not 0:
143 def FirstSlice_Segment(Segment):
144
145     # Defining the size of the Segment array;
146     rows, cols, slices = Segment.shape
147
148     # Defining the initial index,if there is no segmentation this will be used;
149     Slice_index = 0
150
151     # Loop to find the 1st slice which has the segmentation;
152     for i in range(slices):
153         if np.nansum(Segment[:,:,i]) != 0:
154             Slice_index = i
155             break
156
157     return Slice_index
158
159
160 # Function to get Region ROI dataset from Mat-file:
161 def Region_ROI(path):
162
163     # Getting the ROI dataset from the .mat file;
164     ROI_WrongShape = file[path][()]
165
166     # Array to store the ROI in correct shape;
167     ROI = np.zeros((256,256,89),dtype=int)
168
169     # Reshapping ROI from (89x256x256) to (256x256x89) & rotating & flipping;
170     for i in range(89):
```

```
171
172         # Rotating the array 90 degrees & flipping;
173         ROI[:,:,i] = np.fliplr(np.rot90(ROI_WrongShape[i,:,:], k=1, axes=(1,0)))
174
175     return ROI
176
177
178 # Function to write the Region ROI Mask into coppied DICOM files:
179 def WritingDCM_ROIMask(ROI_Path, ROI_Mask, Region):
180
181     # Collecting all dcm files from the folder into a list;
182     lstFilesDCM = DCMFiles_List(ROI_Path)
183
184     # Get refference file (first slice);
185     RefDs = pydicom.dcmread(lstFilesDCM[0])
186
187     # Load dimensions based on the number of rows, columns, and slices;
188     ConstPixelDims = (int(RefDs.Rows), int(RefDs.Columns), len(lstFilesDCM))
189
190     # Creating an array to store dcm data size is based on 'ConstPixelDims';
191     Region_ROI = np.zeros(ConstPixelDims, dtype=RefDs.pixel_array.dtype)
192
193     # Loop through all the DCM files to read-store-write to files again;
194     for filenameDCM in lstFilesDCM:
195
196         # Read the dcm file;
197         ds = pydicom.dcmread(filenameDCM)
198
199         # Modifying some of the tags from the coppied dcm file:
200         if Region == 'Troclea':
201
202             ds.SeriesDescription = Region + ' Region mask'
203             ds.SeriesNumber = ds.SeriesNumber + 5
204
205         elif Region == 'Central':
206
207             ds.SeriesDescription = Region + ' Region mask'
208             ds.SeriesNumber = ds.SeriesNumber + 6
209
210         elif Region == 'Posterior':
211
212             ds.SeriesDescription = Region + ' Region mask'
213             ds.SeriesNumber = ds.SeriesNumber + 7
214
215         ds.RescaleSlope = ds.RescaleSlope/ds.RescaleSlope #makes it 1 in the right format
216         ds.RescaleIntercept = ds.RescaleIntercept - ds.RescaleIntercept;
217
218         # Storing the Region ROI Mask;
219         Region_ROI[:, :, lstFilesDCM.index(filenameDCM)] = ROI_Mask[:, :, lstFilesDCM.index(filenameDCM)]
220
221         # Writing the ROI_Region Mask into the files;
222         ds.PixelData = Region_ROI[:, :, lstFilesDCM.index(filenameDCM)].tobytes()
223
224         # Saving the files;
225         ds.save_as(filenameDCM)
226
227
228 # Function to correct Region ROI values and save to DCM files:
229 def CorrectingDCM_ROIMask(ROI_Path, ROI_Mask, Region):
230
231     # If loop to choose the Regions Value;
232     if Region == 'Troclea':
233
234         Medial_value = 1
235         Lateral_value = 2
236
237     elif Region == 'Central':
```

```
238
239         Medial_value = 3
240         Lateral_value = 4
241
242     elif Region == 'Posterior':
243
244         Medial_value = 5
245         Lateral_value = 6
246
247     # Correcting the resampled ROI Masks values;
248     for s in range(ROI_Mask.shape[2]):
249         for x in range(ROI_Mask.shape[1]):
250             for y in range(ROI_Mask.shape[0]):
251
252                 if ROI_Mask[x,y,s] > 1:
253
254                     ROI_Mask[x,y,s] = Lateral_value
255
256                 elif ROI_Mask[x,y,s] == 1:
257
258                     ROI_Mask[x,y,s] = Medial_value
259
260                 else:
261
262                     ROI_Mask[x,y,s] = 0
263
264
265     # Collecting all dcm files from the folder into a list;
266     lstFilesDCM = DCMFiles_List(ROI_Path)
267
268     # Get refference file (first slice);
269     RefDs = pydicom.dcmread(lstFilesDCM[0])
270
271     # Load dimensions based on the number of rows, columns, and slices;
272     ConstPixelDims = (int(RefDs.Rows), int(RefDs.Columns), len(lstFilesDCM))
273
274     # Creating an array to store dcm data size is based on 'ConstPixelDims';
275     Region_ROI = np.zeros(ConstPixelDims, dtype=RefDs.pixel_array.dtype)
276
277     # Loop through all the DCM files to read-store-write to files again;
278     for filenameDCM in lstFilesDCM:
279
280         # Read the file;
281         ds = pydicom.dcmread(filenameDCM)
282
283         # Modifying the Rescale tags in the dcm file:
284         ds.RescaleSlope = ds.RescaleSlope/ds.RescaleSlope #makes it 1 in the right format
285         ds.RescaleIntercept = ds.RescaleIntercept - ds.RescaleIntercept;
286
287         # Storing the Region ROI Mask;
288         Region_ROI[:, :, lstFilesDCM.index(filenameDCM)] = ROI_Mask[:, :, lstFilesDCM.index(filenameDCM)]
289
290         # Writing the ROI_Region Mask into the files;
291         ds.PixelData = Region_ROI[:, :, lstFilesDCM.index(filenameDCM)].tobytes()
292
293         # Saving the files;
294         ds.save_as(filenameDCM)
295
296
297 # Function to Combine the ROI Mask into coppied TRMask DICOM files:
298 def Combining_ROIMask(ROIMask_Path, CRMask_Path, PRMask_Path):
299
300     # Importing ROI_Mask with the TR Mask;
301     ROI_Mask = DCM_Import(ROIMask_Path)
302
303     # Importing ROI_Mask with the TR Mask;
304     CR_Mask = DCM_Import(CRMask_Path)
```

```
305
306     # Importing ROI_Mask with the TR Mask;
307     PR_Mask = DCM_Import(PRMask_Path)
308
309     # Combining the 3 resampled ROI Masks;
310     for s in range(ROI_Mask.shape[2]):
311         for x in range(ROI_Mask.shape[1]):
312             for y in range(ROI_Mask.shape[0]):
313
314                 if ROI_Mask[x,y,s] == 0:
315
316                     ROI_Mask[x,y,s] = CR_Mask[x,y,s]
317
318                     if ROI_Mask[x,y,s] == 0:
319
320                         ROI_Mask[x,y,s] = PR_Mask[x,y,s]
321
322
323     # Collecting all dcm files from the ROI_Mask folder into a list;
324     lstFilesDCM = DCMFiles_List(ROIMask_Path)
325
326     # Get refference file (first slice);
327     RefDs = pydicom.dcmread(lstFilesDCM[0])
328
329     # Load dimensions based on the number of rows, columns, and slices;
330     ConstPixelDims = (int(RefDs.Rows), int(RefDs.Columns), len(lstFilesDCM))
331
332     # Creating an array to store dcm data size is based on 'ConstPixelDims';
333     Region_ROI = np.zeros(ConstPixelDims, dtype=RefDs.pixel_array.dtype)
334
335     # Loop through all the DCM files to read-store-write to files again;
336     for filenameDCM in lstFilesDCM:
337
338         # Read the dcm file;
339         ds = pydicom.dcmread(filenameDCM)
340
341         # Modifying the Rescale tags in the dcm file:
342         ds.RescaleSlope = ds.RescaleSlope/ds.RescaleSlope #makes it 1 in the right format
343         ds.RescaleIntercept = ds.RescaleIntercept - ds.RescaleIntercept;
344
345         # Storing the Region ROI Mask;
346         Region_ROI[:, :, lstFilesDCM.index(filenameDCM)] = ROI_Mask[:, :, lstFilesDCM.index(filenameDCM)]
347
348         # Writing the ROI_Region Mask into the files;
349         ds.PixelData = Region_ROI[:, :, lstFilesDCM.index(filenameDCM)].tobytes()
350
351         # Saving the files;
352         ds.save_as(filenameDCM)
353
354
355 # Function to Optimize ROI Mask and save to DCM files:
356 def Optimizing_ROIMask(ROI_Path, ROI_Mask, fc):
357
358     # Looping through slices to optimize Lateral Regions;
359     for s in range(ROI_Mask.shape[2]):
360
361         # Array to stor col. indices of Lateral Regions;
362         TL_col = []
363         CL_col = []
364         PL_col = []
365
366         # Indices for Lateral Regions in slice s;
367         _, TL_col = np.where(ROI_Mask[:,:,s] == 2)
368         _, CL_col = np.where(ROI_Mask[:,:,s] == 4)
369         _, PL_col = np.where(ROI_Mask[:,:,s] == 6)
370
371         # Setting all the columns to the region values;
```

```
        if TL_col.size:
            ROI_Mask[:,(TL_col.min()-1):(TL_col.max()+1),s] = 2
        if CL_col.size:
            ROI_Mask[:,(CL_col.min()-1):(CL_col.max()+1),s] = 4
        if PL_col.size:
            ROI_Mask[:,(PL_col.min()-1):(PL_col.max()+1),s] = 6


    # Looping through slices to optimize Medial Regions;
    for s in range(ROI_Mask.shape[2]):

        # Array to stor col. indices of Medial Regions;
        TM_col = []
        CM_col = []
        PM_col = []

        # Indices for Medial Regions in slice s;
        _, TM_col = np.where(ROI_Mask[:,:,s] == 1)
        _, CM_col = np.where(ROI_Mask[:,:,s] == 3)
        _, PM_col = np.where(ROI_Mask[:,:,s] == 5)

        # Setting all the columns to the region values;
        if TM_col.size:
            ROI_Mask[:,(TM_col.min()-1):(TM_col.max()+1),s] = 1
        if CM_col.size:
            ROI_Mask[:,(CM_col.min()-1):(CM_col.max()+1),s] = 3
        if PM_col.size:
            ROI_Mask[:,(PM_col.min()-1):(PM_col.max()+1),s] = 5


    # Sorting out only the fc segmentation - Optimization;
    ROI_Mask[fc < 1] = 0


    # Collecting all dcm files from the folder into a list;
    lstFilesDCM = DCMFiles_List(ROI_Path)

    # Get refference file (first slice);
    RefDs = pydicom.dcmread(lstFilesDCM[0])

    # Load dimensions based on the number of rows, columns, and slices;
    ConstPixelDims = (int(RefDs.Rows), int(RefDs.Columns), len(lstFilesDCM))

    # Creating an array to store dcm data size is based on 'ConstPixelDims';
    Region_ROI = np.zeros(ConstPixelDims, dtype=RefDs.pixel_array.dtype)

    # Loop through all the DCM files to read-store-write to files again;
    for filenameDCM in lstFilesDCM:

        # Read the file;
        ds = pydicom.dcmread(filenameDCM)

        # Modifying the Rescale tags in the dcm file:
        ds.RescaleSlope = ds.RescaleSlope/ds.RescaleSlope #makes it 1 in the right format
        ds.RescaleIntercept = ds.RescaleIntercept - ds.RescaleIntercept;

        # Storing the Region ROI Mask;
        Region_ROI[:, :, lstFilesDCM.index(filenameDCM)] = ROI_Mask[:, :, lstFilesDCM.index(filenameDCM)]

        # Writing the ROI_Region Mask into the files;
        ds.PixelData = Region_ROI[:, :, lstFilesDCM.index(filenameDCM)].tobytes()

        # Saving the files;
        ds.save_as(filenameDCM)


#-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-#
```

```
                    # Defining Subject folder
#-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-#


# Defining the subject folder (Patient-folder):
#----------------------------------------------
NaF = 'NaF_sub'
subject = '02A'
Patient_folder = NaF + subject


#-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-#
                        # Importing ROIs from Matlab file
#-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-#


# Defining the region:
# ---------------------
# Troclea;
Region = 'Troclea'

# # Central;
# Region = 'Central'

# # Posterior;
# Region = 'Posterior'


# Loading .mat file with ROIS:
#-----------------------------
# Reading the file;
file = h5py.File(Patient_folder + '/TAC_' + subject[0:2] + 'TAC_data.mat','r')


# Importing the ROIs from MatLab data file into Python Numpy array:
#------------------------------------------------------------------
# If loop to choose which region that will be Imported from the mat.file;
if Region == 'Troclea':

    # Regions in right leg;
    Right_Medial = Region_ROI('results/ROIS/' + subject[-1] +
                            '_right_femurTroclea_medial') * 1
    Right_Lateral = Region_ROI('results/ROIS/' + subject[-1] +
                          '_right_femurTroclea_lateral') * 34464
    # Regions in left leg;
    Left_Medial = Region_ROI('results/ROIS/' + subject[-1] +
                            '_left_femurTroclea_medial') * 1
    Left_Lateral = Region_ROI('results/ROIS/' + subject[-1] +
                          '_left_femurTroclea_lateral') * 34464

elif Region == 'Central':

    # Regions in right leg;
    Right_Medial = Region_ROI('results/ROIS/' + subject[-1] +
                            '_right_femurcentral_medial')*1
    Right_Lateral = Region_ROI('results/ROIS/' + subject[-1] +
                          '_right_femurcentral_lateral')*34464

    # Regions in left leg;
    Left_Medial = Region_ROI('results/ROIS/' + subject[-1] +
                            '_left_femurcentral_medial')*1
    Left_Lateral = Region_ROI('results/ROIS/' + subject[-1] +
                          '_left_femurcentral_lateral')*34464

elif Region == 'Posterior':

    # Regions in right leg;
```

```
    Right_Medial = Region_ROI('results/ROIS/' + subject[-1] +
                              '_right_femurposterior_medial')*1
    Right_Lateral = Region_ROI('results/ROIS/' + subject[-1] +
                             '_right_femurposterior_lateral')*34464

    # Regions in left leg;
    Left_Medial = Region_ROI('results/ROIS/' + subject[-1] +
                              '_left_femurposterior_medial')*1
    Left_Lateral = Region_ROI('results/ROIS/' + subject[-1] +
                             '_left_femurposterior_lateral')*34464


# Combining medial & lateral region with dif. values in one array;
# Regions in right leg;
ROI_right = Right_Medial + Right_Lateral

# Regions in left leg;
ROI_left = Left_Medial + Left_Lateral

# Both leg
ROI_both = ROI_right + ROI_left

# First slice of Region ROI:
ROI_ind = FirstSlice_Segment(ROI_both)


# Displaying ROI:
#-----------------
# Plotting ROI imported from MatLab:
fig, ax = plt.subplots(1, 1, figsize=(10, 20))

track1 = IndexTracker(fig, ax, ROI_ind, ROI_both, 1, 'jet',
                      Region + ' Region ROI imported from mat-file')

fig.canvas.mpl_connect('key_press_event', track1.on_key)

plt.show()


#-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-#
                # Writting ROIs to DICOM files and Displaying
#-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-#


# Write Region ROI Mask imported form MatLab to DICOM file:
#----------------------------------------------------------
# Path to folder with the coppied DCM files to write the Region ROI Mask onto;
if Region == 'Troclea':

    RegionROI_Path = Patient_folder + '/' + Region + 'Region_ROI/'

elif Region == 'Central':

    RegionROI_Path = Patient_folder + '/' + Region + 'Region_ROI/'

elif Region == 'Posterior':

    RegionROI_Path = Patient_folder + '/' + Region + 'Region_ROI/'


# #-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#
# #-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#

# # Writting the Region ROI Mask to the coppied DICOM files;
# WritingDCM_ROIMask(RegionROI_Path, ROI_both, Region)

# #-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#
```

```
# #-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#

# Importing DCM files from Region_ROI folder;
RegionROI = np.float64(DCM_Import(RegionROI_Path))

# The first slice with imported ROI Mask;
RegionROI_ind = FirstSlice_Segment(RegionROI)


# Displaying ROI:
#----------------
# Plotting DICOM files from Region_ROI folder:
fig1, ax1 = plt.subplots(1, 1, figsize=(10, 20))

track1_1 = IndexTracker(fig1, ax1, RegionROI_ind, RegionROI, 1, 'jet',
                        'DCM files from ' + Region + 'Region_ROI folder displayed')

fig1.canvas.mpl_connect('key_press_event', track1_1.on_key)

plt.show()


#-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-#
                    # Correcting the Resampled ROI values
#-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-#


# Defining the region:
#---------------------
# Troclea;
Region = 'Troclea'

# # Central;
# Region = 'Central'

# # Posterior;
# Region = 'Posterior'


# Importing the E1 DESS scan for both Knees:
#-------------------------------------------
DESS_DCM = DCM_Import(Patient_folder + '/DESSLow/')

# Dividing into not Diffused (Echo1) and Diffused (Echo2) Array;
# Sizes of the imported DESS images;
Rows, Cols, Slices = DESS_DCM.shape
# Array to store the DESS scans without difussion (Echo 1);
E1_DCM = np.zeros((Rows, Cols, int(Slices/2)))

# Echo1, The odd no. images/slices [I0001.dcm,I0003.dcm,I0005.dcm,...]:
# Intialising index for divided array;
idx_1 = 0
for i in range(Slices):
    # Even no. in file list (incl. 0) will be stored in not diffused array;
    if (i % 2) == 0: # Even numbers in file list are odd no. images
        E1_DCM[:, :, idx_1] = DESS_DCM[:, :, i]
        idx_1 += 1


# Importing the resampled Region ROI Mask:
#-----------------------------------------
# Path to folder with resampled DCM files of Region ROI Mask;
if Region == 'Troclea':

    ResampledROI_Path = Patient_folder + '/' + Region[0] + 'RMask2DESS_' + subject[1:3] + '/'

```

```
640 elif Region == 'Central':
641
642     ResampledROI_Path = Patient_folder + '/' + Region[0] + 'RMask2DESS_' + subject[1:3] + '/'
643
644 elif Region == 'Posterior':
645
646     ResampledROI_Path = Patient_folder + '/' + Region[0] + 'RMask2DESS_' + subject[1:3] + '/'
647
648
649 # Importing resampled Region ROI Mask;
650 RMask = DCM_Import(ResampledROI_Path)
651
652 # The first slice with ROI Mask;
653 RMask_ind = FirstSlice_Segment(RMask)
654
655
656 #-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#
657 #-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#
658
659 # Correcting the ROI values and saving to the resampeled dcm files:
660 CorrectingDCM_ROIMask(ResampledROI_Path, RMask, Region)
661
662 #-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#
663 #-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#
664
665
666 # Plotting Resampled ROI mask on top of DESS E1:
667 fig2, ax2 = plt.subplots(1, 1, figsize=(10, 20))
668
669 track2_2 = IndexTracker(fig2, ax2, RMask_ind, E1_DCM, 1, 'gray',
670                         'Resampled ' + Region + ' Region ROI Mask on top of DESS E1')
671 track2_1 = IndexTracker(fig2, ax2, RMask_ind, RMask, 0.4, 'jet',
672                         'Resampled ' + Region + ' Region ROI Mask on top of DESS E1')
673
674 fig2.canvas.mpl_connect('key_press_event', track2_1.on_key)
675 fig2.canvas.mpl_connect('key_press_event', track2_2.on_key)
676
677 plt.show()
678
679
680 #-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-#
681                     # Combing the 3 Regions to 1 ROI Mask
682 #-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-#
683
684
685 # Importing the E1 DESS scan for both Knees:
686 #-----------------------------------------
687 DESS_DCM = DCM_Import(Patient_folder + '/DESSLow/')
688
689 # Dividing into not Diffused (Echo1) and Diffused (Echo2) Array;
690 # Sizes of the imported DESS images;
691 Rows, Cols, Slices = DESS_DCM.shape
692 # Array to store the DESS scans without difussion (Echo 1);
693 E1_DCM = np.zeros((Rows, Cols, int(Slices/2)))
694
695 # Echo1, The odd no. images/slices [I0001.dcm,I0003.dcm,I0005.dcm,...]:
696 # Intialising index for divided array;
697 idx_1 = 0
698 for i in range(Slices):
699     # Even no. in file list (incl. 0) will be stored in not diffused array;
700     if (i % 2) == 0: # Even numbers in file list are odd no. images
701         E1_DCM[:, :, idx_1] = DESS_DCM[:, :, i]
702         idx_1 += 1
703
704
705 # Combining the 3 Region Masks to 1 ROI Mask:
706 #-------------------------------------------
```

```
# Path to folder with ROI Mask (TRMask);
ROIMask_Path = Patient_folder + '/ROI_Mask/'

# Path to folder with ROI Mask (TRMask);
CRMask_Path = Patient_folder + '/CRMask2DESS_' + subject[1:3] + '/'

# Path to folder with ROI Mask (TRMask);
PRMask_Path = Patient_folder + '/PRMask2DESS_' + subject[1:3] + '/'


#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#
#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#

# Combining the 3 Region Masks to 1 ROI Mask and saving to the dcm files:
Combining_ROIMask(ROIMask_Path, CRMask_Path, PRMask_Path)

#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#
#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#


# Importing ROI Mask;
ROI_Mask = DCM_Import(ROIMask_Path)

# The first slice with ROI Mask;
ROIMask_ind = FirstSlice_Segment(ROI_Mask)

# Plotting Combined ROI mask on top of DESS E1:
fig3, ax3 = plt.subplots(1, 1, figsize=(10, 20))

track3_2 = IndexTracker(fig3, ax3, ROIMask_ind, E1_DCM, 1, 'gray',
                        'Combined ROI Mask on top of DESS E1')
track3_1 = IndexTracker(fig3, ax3, ROIMask_ind, ROI_Mask, 0.4, 'jet',
                        'Combined ROI Mask on top of DESS E1')

fig3.canvas.mpl_connect('key_press_event', track3_1.on_key)
fig3.canvas.mpl_connect('key_press_event', track3_2.on_key)

plt.show()


#-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-#
                        # Final Optimization of ROI Mask
#-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-#


# # Defining the subject folder (Patient-folder):
# #----------------------------------------------
# NaF = 'NaF_sub'
# subject = '13A'
# Patient_folder = NaF + subject


# Importing the E1 DESS scan for both Knees:
#-------------------------------------------
DESS_DCM = DCM_Import(Patient_folder + '/DESSLow_Correct/')

# Dividing into not Diffused (Echo1) and Diffused (Echo2) Array;
# Sizes of the imported DESS images;
Rows, Cols, Slices = DESS_DCM.shape
# Array to store the DESS scans without difussion (Echo 1);
E1_DCM = np.zeros((Rows, Cols, int(Slices/2)))

# Echo1, The odd no. images/slices [I0001.dcm,I0003.dcm,I0005.dcm,...]:
# Intialising index for divided array;
idx_1 = 0
for i in range(Slices):
    # Even no. in file list (incl. 0) will be stored in not diffused array;
```

```
    if (i % 2) == 0: # Even numbers in file list are odd no. images
        E1_DCM[:, :, idx_1] = DESS_DCM[:, :, i]
        idx_1 += 1


# Importing the ROI Mask:
#------------------------
# Path to folder with ROI Mask (TRMask);
ROIMask_Path = Patient_folder + '/ROI_Mask/'

# Importing ROI Mask;
ROI_Mask = DCM_Import(ROIMask_Path)

# The first slice with ROI Mask;
ROIMask_ind = FirstSlice_Segment(ROI_Mask)


# Importing the Segmentations (NIfTI-data):
#------------------------------------------
# Importing the segmentation (NIfTI-data);
fc_K1 = NIfTI_Import(Patient_folder + '/DOSMAKnee1/fc', 'fc.nii.gz')
fc_K2 = NIfTI_Import(Patient_folder + '/DOSMAKnee2/fc', 'fc.nii.gz')

# Combining the fc for knee1 and knee2 to whole fc;
fc = np.zeros((ROI_Mask.shape[0],ROI_Mask.shape[1],ROI_Mask.shape[2]))

fc[:,:,0:fc_K1.shape[2]] = fc_K1
fc[:,:,fc_K1.shape[2]:ROI_Mask.shape[2]] = fc_K2


#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#
#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#

# Optimizing the ROI Mask and saving to the dcm files:
Optimizing_ROIMask(ROIMask_Path, ROI_Mask, fc)

#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#
#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#-*-#


# The first slice with ROI Mask after Optimization;
ROIMask_ind = FirstSlice_Segment(ROI_Mask)

# Defining zero in segmentation as nan;
ROI_Mask[ROI_Mask == 0] = np.nan



# Plotting ROI mask on top of DESS E1:
#-------------------------------------
fig4, ax4 = plt.subplots(1, 1, figsize=(10, 20))

track4_2 = IndexTracker(fig4, ax4, ROIMask_ind+4, E1_DCM, 1, 'gray',
                        'femoral cartilage region mask')
track4_1 = IndexTracker(fig4, ax4, ROIMask_ind+4, ROI_Mask, 0.4, 'jet',
                        'femoral cartilage region mask')

fig4.canvas.mpl_connect('key_press_event', track4_1.on_key)
fig4.canvas.mpl_connect('key_press_event', track4_2.on_key)

plt.show()
```

A.3 CARTILAGE AND ADJACENT BONE VOXEL

```
# Importing necersary packages:
#------------------------------
from scipy.ndimage.morphology import binary_dilation, binary_erosion
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
from natsort import natsorted
import matplotlib as mpl
from pathlib import Path
from scipy import stats
import nibabel as nib
import numpy as np
import pydicom
import os


# Setting som plotting standards:
#--------------------------------
font = {'weight' : 'normal', 'size'   : 14}
mpl.rc('font', **font)


# Defining functions/classes used in the code:
#---------------------------------------------
# Function to import Nifti data into Pixel array:
def NIfTI_Import(Path, filename):

    # Setting the path to the NIfTI file;
    nii = os.path.join(Path, filename)
    # Loading and getting the pixel data for the NIfTI file;
    ArrayNIfTI = nib.load(nii).get_fdata()

    return ArrayNIfTI


# Function to collect all dcm files from a folder into lstFilesDCM:
def DCMFiles_List(DCM_Path):

    # Create an empty list to collect all .dcm files into it;
    lstFilesDCM = []

    # Loop to traverse the directory & collect all dcm files into lstFilesDCM;
    for dirName, subdirList, fileList in os.walk(DCM_Path):
        # Sorting the fileList;
        #fileList.sort()
        fileList = natsorted(fileList)
        for filename in fileList:
            # Check whether the file's DICOM;
            if ".dcm" in filename.lower():
                # Storing files in list;
                lstFilesDCM.append(os.path.join(dirName,filename))

    return lstFilesDCM


# Function to import raw DICOM data into an Pixel array:
def DCM_Import(DCM_Path):

    # Collecting all dcm files from the folder into a list;
    lstFilesDCM = DCMFiles_List(DCM_Path)

    # Reference to extract metadata:
```

```
    # Get refference file (first slice);
    RefDs = pydicom.dcmread(lstFilesDCM[0])

    # Load dimensions based on the number of rows, columns, and slices;
    ConstPixelDims = (int(RefDs.Rows), int(RefDs.Columns), len(lstFilesDCM))

    # Storring the raw DICOM data:
    # Creating an array size is based on 'ConstPixelDims';
    ArrayDicom = np.zeros(ConstPixelDims, dtype=RefDs.pixel_array.dtype)

    # Creating array to store the Rescale Slope & Rescale Intercept for all slices;
    RescaleSlope = np.ones(len(lstFilesDCM))
    RescaleIntercept = np.zeros(len(lstFilesDCM))

    # Loop through all the DICOM files;
    for filenameDCM in lstFilesDCM:

        # Read the file;
        ds = pydicom.dcmread(filenameDCM)

        # Storing raw data;
        ArrayDicom[:, :, lstFilesDCM.index(filenameDCM)] = ds.pixel_array

        # Correcting for Rescale tags if they exist;
        if ("RescaleIntercept" in ds) == True:

            RescaleSlope[lstFilesDCM.index(filenameDCM)] = ds.RescaleSlope
            RescaleIntercept[lstFilesDCM.index(filenameDCM)] = ds.RescaleIntercept

    # Correcting for the rescale tags;
    ArrayDicom = ArrayDicom * RescaleSlope + RescaleIntercept

    return ArrayDicom


# Class to plot and scroll trough the slices using key_press_event:
class IndexTracker:

    # Initial definitions for plot;
    def __init__(self, fig, ax, ind, X, alpha, cmap, title, subject):

        self.fig = fig
        self.ind = ind
        self.cmap = cmap
        self.subject = subject
        self.alpha = alpha
        self.ax = ax
        self.X = X
        rows, cols, self.slices = X.shape

        self.im = ax.imshow(self.X[:, :, self.ind],
                            alpha=self.alpha, cmap=self.cmap)
        self.ax.set_title(title)
        self.ax.axis('off')
        self.update()

    # Key definition for navigation of slices;
    def on_key(self, event):

        if event.key == 'up':
            self.ind = (self.ind + 1) % self.slices
        elif event.key == 'down':
            self.ind = (self.ind - 1) % self.slices
        self.update()

    # Function to update plot;
    def update(self):
```

```
136
137         self.im.set_data(self.X[:, :, self.ind])
138         self.fig.suptitle('Slice ' + str(self.ind+1) + ' of baseline DESS scan with')
139         # self.fig.suptitle('Slice ' + str(self.ind+1) + ' for subject ' +
140         #                   self.subject + ' ' + knee)
141         self.im.axes.figure.canvas.draw()
142
143
144 # Function to find the first slice of the segmentation that is not 0:
145 def FirstSlice_Segment(Segment):
146
147     # Defining the size of the Segment array;
148     rows, cols, slices = Segment.shape
149
150     # Defining the initial index,if there is no segmentation this will be used;
151     Slice_index = 0
152
153     # Loop to find the 1st slice which has the segmentation;
154     for i in range(slices):
155         if np.nansum(Segment[:,:,i]) != 0:
156             Slice_index = i
157             break
158
159     return Slice_index
160
161
162 # Function to find the adjacent subchondral bone voxel to femoral cartilage:
163 def BoneCartilageVoxel(fc,r,CMask,T2fc,PET,RMask):
164
165     # fc mask used to search;
166     fc_search = binary_erosion(fc).astype(fc.dtype)
167
168     # fc mask used to define where not to assign adjacent bone voxel;
169     bone_not = binary_dilation(fc).astype(fc.dtype)
170
171     # Array to store the adjacent bone and cartilage mask for display;
172     fc_adj = np.zeros(fc.shape)
173
174     # Array to store centrum/mean point of segmentation;
175     mean_fc = np.zeros((fc.shape[2],2), dtype = int)
176
177     # The radius of the voxel around the fc that will be checked;
178     radius = r
179
180     # Array to store the T2 and Naf Values for cartilage-bone voxel;
181     # For the whole Femur Cartilage;
182     T2 = []
183     Naf = []
184
185     # For different Regions of the fermur cartilage;
186     T2_TM = [] # Troclea_Medial
187     T2_TL = [] # Troclea_Lateral
188     Naf_TM = [] # Troclea_Medial
189     Naf_TL = [] # Troclea_Lateral
190
191     T2_CM = [] # Central_Medial
192     T2_CL = [] # Central_Lateral
193     Naf_CM = [] # Central_Medial
194     Naf_CL = [] # Central_Lateral
195
196     T2_PM = [] # Posterior_Medial
197     T2_PL = [] # Posterior_Lateral
198     Naf_PM = [] # Posterior_Medial
199     Naf_PL = [] # Posterior_Lateral
200
201
202     # Looping through slices;
```

```
    for s in range(fc.shape[2]):

        # Index counter for the array that stores the fc & bone voxel;
        voxel = 0

        # Arrays to store index of fc segmentations - fc row and col;
        fc_row = []
        fc_col = []

        # Index for fc segmentation in slice s:
        fc_row, fc_col = np.where(fc[:,:,s] > 0)

        # Mean point for the fc segmentation on each slice;
        if fc_row.size:

            # Midt point of min & max in both row and col;
            mean_fc[s,0] = int((fc_row.max()- fc_row.min())/2 + fc_row.min())
            mean_fc[s,1] = int((fc_col.max()- fc_col.min())/2 + fc_col.min())


        # Arrays to store index of serch fc segmentations;
        search_row = []
        search_col = []

        # Index for search fc segmentation in slice s:
        search_row, search_col = np.where(fc_search[:,:,s] > 0)


        # Looping through every fc voxel in slice s to find adjacent bone voxel;
        for i in range(len(search_row)):

            # Array with the row and col indices of neighbouring voxel;
            row_nb = np.arange(search_row[i] - radius, search_row[i] + 1)
            col_nb = np.arange(search_col[i] - radius, search_col[i] + (radius+1))

            # Defining the threshold (radius) for the adjacent bone voxel search;
            # The distance between the fc_mean point and the current fc voxel ;
            Threshold = np.sqrt( (mean_fc[s,0] - search_row[i])**2 +
                                 (mean_fc[s,1] - search_col[i])**2 )

            # Array to store the indices of the neighbouring voxel;
            Nb_voxel = np.zeros((len(row_nb)*len(col_nb),2),dtype=int)

            # Array with the distance from the neighbour voxel to the fc_mean point;
            Distance = np.zeros(Nb_voxel.shape[0])

            #Counter for index in Neighbour_voxel;
            v = 0

            # Looping through row_nb;
            for x in range(len(row_nb)):
                # Looping through col_nb;
                for y in range(len(col_nb)):

                    # Storing the indices of the neighbouring voxel in an array;
                    Nb_voxel[v,0] = row_nb[x]
                    Nb_voxel[v,1] = col_nb[y]

                    v += 1

            # Computing the distance for each neighbour to the fc mean point;
            for d in range(len(Distance)):

                Distance[d] = np.sqrt( (mean_fc[s,0] - Nb_voxel[d,0])**2 +
                                       (mean_fc[s,1] - Nb_voxel[d,1])**2 )

            # Finding adjacent bone voxel for the current fc segment;
```

```
            for d in range(len(Distance)):
                if Distance[d]<Threshold and bone_not[Nb_voxel[d,0],Nb_voxel[d,1],s]!=1:
                    if CMask[Nb_voxel[d,0],Nb_voxel[d,1],s] == 1:

                        # "Drawing" the fc voxel and adjacent bone voxel;
                        fc_adj[search_row[i],search_col[i],s] = 1 #Cartilage
                        fc_adj[Nb_voxel[d,0],Nb_voxel[d,1],s] = 2 #Bone

                        # Cartilage T2 value and the adjacent bone voxels Naf Uptake;
                        if T2fc[search_row[i],search_col[i],s] != 0:

                            # Storing T2 and corresponding NaF values in array;
                            # For the whole femur cartilage
                            T2.append(T2fc[search_row[i], search_col[i],s])
                            Naf.append(PET[Nb_voxel[d,0], Nb_voxel[d,1],s])

                            # Cartilage T2 value and Naf for regions of fc;
                            if RMask[search_row[i],search_col[i],s] == 1:

                                # Storing T2 and corresponding NaF values in array;
                                T2_TM.append(T2fc[search_row[i], search_col[i],s]) # Troclea_Medial
                                Naf_TM.append(PET[Nb_voxel[d,0], Nb_voxel[d,1],s])

                            elif RMask[search_row[i],search_col[i],s] == 2:

                                # Storing T2 and corresponding NaF values in array;
                                T2_TL.append(T2fc[search_row[i], search_col[i],s]) # Troclea_Lateral
                                Naf_TL.append(PET[Nb_voxel[d,0], Nb_voxel[d,1],s])

                            elif RMask[search_row[i],search_col[i],s] == 3:

                                # Storing T2 and corresponding NaF values in array;
                                T2_CM.append(T2fc[search_row[i], search_col[i],s]) # Central_Medial
                                Naf_CM.append(PET[Nb_voxel[d,0], Nb_voxel[d,1],s])

                            elif RMask[search_row[i],search_col[i],s] == 4:

                                # Storing T2 and corresponding NaF values in array;
                                T2_CL.append(T2fc[search_row[i], search_col[i],s]) # Central_Lateral
                                Naf_CL.append(PET[Nb_voxel[d,0], Nb_voxel[d,1],s])

                            elif RMask[search_row[i],search_col[i],s] == 5:

                                # Storing T2 and corresponding NaF values in array;
                                T2_PM.append(T2fc[search_row[i], search_col[i],s]) # Posterior_Medial
                                Naf_PM.append(PET[Nb_voxel[d,0], Nb_voxel[d,1],s])

                            elif RMask[search_row[i],search_col[i],s] == 6:

                                # Storing T2 and corresponding NaF values in array;
                                T2_PL.append(T2fc[search_row[i], search_col[i],s]) # Posterior_Lateral
                                Naf_PL.append(PET[Nb_voxel[d,0], Nb_voxel[d,1],s])


    # List of T2 and NaF values converted into one numpy array (T2Naf);
    # For whole femur cartilage
    T2Naf = np.zeros((len(T2),2))
    T2Naf[:,0] = np.array(T2)
    T2Naf[:,1] = np.array(Naf)

    # For femur Troclea medial
    T2Naf_TM = np.zeros((len(T2_TM),2))
    T2Naf_TM[:,0] = np.array(T2_TM)
    T2Naf_TM[:,1] = np.array(Naf_TM)

    # For femur Troclea lateral
    T2Naf_TL = np.zeros((len(T2_TL),2))
```

```
    T2Naf_TL[:,0] = np.array(T2_TL)
    T2Naf_TL[:,1] = np.array(Naf_TL)

    # For femur Central medial
    T2Naf_CM = np.zeros((len(T2_CM),2))
    T2Naf_CM[:,0] = np.array(T2_CM)
    T2Naf_CM[:,1] = np.array(Naf_CM)

    # For femur Troclea lateral
    T2Naf_CL = np.zeros((len(T2_CL),2))
    T2Naf_CL[:,0] = np.array(T2_CL)
    T2Naf_CL[:,1] = np.array(Naf_CL)

    # For femur Posterior medial
    T2Naf_PM = np.zeros((len(T2_PM),2))
    T2Naf_PM[:,0] = np.array(T2_PM)
    T2Naf_PM[:,1] = np.array(Naf_PM)

    # For femur Posterior lateral
    T2Naf_PL = np.zeros((len(T2_PL),2))
    T2Naf_PL[:,0] = np.array(T2_PL)
    T2Naf_PL[:,1] = np.array(Naf_PL)

    return fc_adj,T2Naf,T2Naf_TM,T2Naf_TL,T2Naf_CM,T2Naf_CL,T2Naf_PM,T2Naf_PL


# Function to bin af set of data and get binned mean:
def BinnedData(nbins,x_data,y_data):

    # Computing binned mean;
    bin_means,bin_edges,_ = stats.binned_statistic(x_data,y_data,
                                                   statistic='mean',bins=nbins)

    # Computing binned std;
    bin_std,_,_ = stats.binned_statistic(x_data,y_data,statistic='std',
                                         bins=nbins)

    # x bins;
    bin_x = (bin_edges[1:] + bin_edges[:-1])/2

    return bin_x,bin_means,bin_std


# Function to make a linear fit for a set of data:
def LinearFit(x_data,y_data):

    # Data to fit - raw data;
    x = x_data.reshape((-1, 1))
    y = y_data

    # Fitting the linear model to the data;
    model = LinearRegression().fit(x,y)

    # The coefficient of determination (R^2)
    R2 = model.score(x,y)

    # The slope and intercept of the linear regression;
    b0 = model.intercept_
    b1 = model.coef_[0]

    # Predict response;
    y_pred = model.predict(x)

    return x,y_pred,b0,b1,R2


# Function to Import registered images and reshaping:
```

```
404 def Importing_RegisteredImage(path, filename):
405 
406     # Importing the registered image;
407     Reg_WrongShape =  NIfTI_Import(path,filename)
408 
409     # Array to store the ROI in correct shape;
410     Reg = np.zeros((512,512,Reg_WrongShape.shape[0]),dtype=int)
411 
412     # Reshapping Reg from (220x512x512) to (512x512x220) & rotating & flipping;
413     for i in range(Reg_WrongShape.shape[0]):
414 
415         # Rotating the array 90 degrees & flipping;
416         Reg[:,:,i] = np.fliplr(np.rot90(Reg_WrongShape[i,:,:], k=1, axes=(0,1)))
417 
418     return Reg
419 
420 
421 #-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-#
422                             # Defining subject folder
423 #-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-#
424 
425 
426 # Defining the subject folder & Knee (Patient-folder):
427 #-----------------------------------------------------
428 NaF = 'NaF_sub0'
429 knee = 'Knee1'
430 subjectA = '2A'
431 subjectB = '2B'
432 Patient_folderA = NaF + subjectA
433 Patient_folderB = NaF + subjectB
434 
435 
436 #-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-#
437                 # Importing PET / MRI images and segmentations
438 #-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-#
439 
440 
441 # Importing the Segmentations (NIfTI-data):
442 #-------------------------------------------
443 # Importing the segmentation (NIfTI-data);
444 A_fc = NIfTI_Import(Patient_folderA + '/DOSMA' + knee + '/fc', 'fc.nii.gz')
445 B_fc = NIfTI_Import(Patient_folderB + '/DOSMA' + knee + '/fc', 'fc.nii.gz')
446 
447 
448 # Importing the resampled Region ROI mask:
449 # ----------------------------------------
450 # Importing resampled Region ROI Mask;
451 A_RMask = DCM_Import(Patient_folderA + '/ROI_Mask/' )
452 B_RMask = DCM_Import(Patient_folderB + '/ROI_Mask/' )
453 
454 # Getting the needed knee only for the data processing;
455 if knee == 'Knee1':
456 
457     # Subtracting only knee 1
458     A_RMask = A_RMask[:,:,0:110]
459     B_RMask = B_RMask[:,:,0:110]
460 
461 elif knee == 'Knee2':
462 
463     # Subtracting only knee 2
464     A_RMask = A_RMask[:,:,110:A_RMask.shape[2]]
465     B_RMask = B_RMask[:,:,110:B_RMask.shape[2]]
466 
467 
468 # Importing DESS scan Echo 1:
469 #----------------------------
470 # Importing the DESS (NIfTI-data);
```

```
A_E1 = NIfTI_Import(Patient_folderA + '/DOSMA' + knee + '/qdess', 'echo1.nii.gz')
B_E1 = NIfTI_Import(Patient_folderB + '/DOSMA' + knee + '/qdess', 'echo1.nii.gz')


# Importing the resampled Cortical Mask of bone:
#-----------------------------------------------
# Importing the resampled scans, using the function from above;
A_CMask = np.float64(DCM_Import(Patient_folderA + '/CMask2DESS_' + subjectA + '/'))
B_CMask = np.float64(DCM_Import(Patient_folderB + '/CMask2DESS_' + subjectB + '/'))

# Getting the needed knee only for the data processing;
if knee == 'Knee1':

    # Subtracting only knee 1
    A_CMask = A_CMask[:,:,0:110]
    B_CMask = B_CMask[:,:,0:110]

elif knee == 'Knee2':

    # Subtracting only knee 2
    A_CMask = A_CMask[:,:,110:A_CMask.shape[2]]
    B_CMask = B_CMask[:,:,110:B_CMask.shape[2]]


# Importing the resampled PET (PET2DESS):
#----------------------------------------
# Importing the registered scans, using the function from above;
A_PET = Importing_RegisteredImage(Patient_folderA, 'PET.nii.gz')
B_PET = Importing_RegisteredImage(Patient_folderB, 'PET.nii.gz')

# Getting the needed knee only for the data processing;
if knee == 'Knee1':

    # Subtracting only knee 1
    A_PET = A_PET[:,:,0:110]
    B_PET = B_PET[:,:,0:110]

elif knee == 'Knee2':

    # Subtracting only knee 2
    A_PET = A_PET[:,:,110:A_PET.shape[2]]
    B_PET = B_PET[:,:,110:B_PET.shape[2]]


# Importing T2 Map (NIfTI file):
#-------------------------------
# Importing the T2 map for knee 1 & 2 (NIfTI-data);
A_T2fc = NIfTI_Import(Patient_folderA + '/DOSMA' + knee + '/fc/t2', 't2.nii.gz')
B_T2fc = NIfTI_Import(Patient_folderB + '/DOSMA' + knee + '/fc/t2', 't2.nii.gz')

# Reducing noise in T2 map :
A_T2fc[A_T2fc > 40] = 0
A_T2fc[A_T2fc < 15] = 0

B_T2fc[B_T2fc > 40] = 0
B_T2fc[B_T2fc < 15] = 0


# Importing the resampled T1rho Map:
#-----------------------------------
# Importing the registered scans, using the function from above;
A_T1p = Importing_RegisteredImage(Patient_folderA, 'T1p.nii.gz')
B_T1p = Importing_RegisteredImage(Patient_folderB, 'T1p.nii.gz')

# Getting the needed knee only for the data processing;
if knee == 'Knee1':

```

```
    # Subtracting only knee 1
    A_T1p = A_T1p[:,:,0:110]
    B_T1p = B_T1p[:,:,0:110]

elif knee == 'Knee2':

    # Subtracting only knee 2
    A_T1p = A_T1p[:,:,110:A_T1p.shape[2]]
    B_T1p = B_T1p[:,:,110:B_T1p.shape[2]]

# Reducing noise in T1p map :
A_T1p[A_T1p > 70] = 0
A_T1p[A_T1p < 15] = 0

B_T1p[B_T1p > 70] = 0
B_T1p[B_T1p < 15] = 0


#-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-#
    # Finding adjacent bone voxel to fc and saving to compressed file - T2
#-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-#


# Finding adjacent Subchondral bone voxel to fc:
# ------------------------------------------------
# T2NaF;
A_fc_adj,A_T2Naf,A_T2Naf_TM,A_T2Naf_TL,A_T2Naf_CM,A_T2Naf_CL,A_T2Naf_PM,A_T2Naf_PL = BoneCartilageVoxel(A_fc
B_fc_adj,B_T2Naf,B_T2Naf_TM,B_T2Naf_TL,B_T2Naf_CM,B_T2Naf_CL,B_T2Naf_PM,B_T2Naf_PL = BoneCartilageVoxel(B_fc


# Saving data to file in compressed .npz format:
# ------------------------------------------------
# T2NaF;
np.savez_compressed(subjectA + knee + '_T2Naf',T2Naf=A_T2Naf,T2Naf_TM=A_T2Naf_TM,T2Naf_TL=A_T2Naf_TL,
                    T2Naf_CM=A_T2Naf_CM,T2Naf_CL=A_T2Naf_CL,T2Naf_PM=A_T2Naf_PM,T2Naf_PL=A_T2Naf_PL)
np.savez_compressed(subjectB + knee + '_T2Naf',T2Naf=B_T2Naf,T2Naf_TM=B_T2Naf_TM,T2Naf_TL=B_T2Naf_TL,
                    T2Naf_CM=B_T2Naf_CM,T2Naf_CL=B_T2Naf_CL,T2Naf_PM=B_T2Naf_PM,T2Naf_PL=B_T2Naf_PL)

# Checking if the saved arrays are correct;
loaded = np.load( subjectA + knee + '_T2Naf.npz' )
print(np.array_equal(A_T2Naf, loaded['T2Naf']))
print(np.array_equal(A_T2Naf_TM, loaded['T2Naf_TM']))


#-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-#
    # Finding adjacent bone voxel to fc and saving to compressed file - T1p
#-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-#


# # Finding adjacent Subchondral bone voxel to fc:
# #-----------------------------------------------
# # T1pNaF;
# A_fc_adj,A_T1pNaf,A_T1pNaf_TM,A_T1pNaf_TL,A_T1pNaf_CM,A_T1pNaf_CL,A_T1pNaf_PM,A_T1pNaf_PL = BoneCartilageV
# B_fc_adj,B_T1pNaf,B_T1pNaf_TM,B_T1pNaf_TL,B_T1pNaf_CM,B_T1pNaf_CL,B_T1pNaf_PM,B_T1pNaf_PL = BoneCartilageV


# # Saving data to file in compressed .npz format:
# # ------------------------------------------------
# # T1pNaF;
# np.savez_compressed(subjectA + knee + '_T1pNaf',T1pNaf=A_T1pNaf,T1pNaf_TM=A_T1pNaf_TM,T1pNaf_TL=A_T1pNaf_T
#                     T1pNaf_CM=A_T1pNaf_CM,T1pNaf_CL=A_T1pNaf_CL,T1pNaf_PM=A_T1pNaf_PM,T1pNaf_PL=A_T1pNaf_P
# np.savez_compressed(subjectB + knee + '_T1pNaf',T1pNaf=B_T1pNaf,T1pNaf_TM=B_T1pNaf_TM,T1pNaf_TL=B_T1pNaf_T
#                     T1pNaf_CM=B_T1pNaf_CM,T1pNaf_CL=B_T1pNaf_CL,T1pNaf_PM=B_T1pNaf_PM,T1pNaf_PL=B_T1pNaf_P

# # Checking if the saved arrays are correct;
# loaded = np.load( subjectA + knee + '_T1pNaf.npz' )
# print(np.array_equal(A_T1pNaf, loaded['T1pNaf']))
```

```python
# print(np.array_equal(A_T1pNaf_TM, loaded['T1pNaf_TM']))


#-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-#
            # Displaying the Cartilage and adjacent Bone voxel
#-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-#


# The index of the first slice of the segmentations:
#---------------------------------------------------
A_fc_ind = FirstSlice_Segment(A_fc_adj)
B_fc_ind = FirstSlice_Segment(B_fc_adj)


# Defining zero as nan for plotting;
#-----------------------------------
A_fc_adj[A_fc_adj == 0] = np.nan
B_fc_adj[B_fc_adj == 0] = np.nan


# Displaying the Cartilage and adjacent Bone voxel with DESS E1:
#---------------------------------------------------------------
# Subject A - Knee 1
fig1, ax1 = plt.subplots(1, 1, figsize=(14, 7))

track1_1 = IndexTracker(fig1, ax1, A_fc_ind, A_E1, 1, 'gray',
                        'femoral cartilage and adjacent bone voxels', subjectA)
track1_2 = IndexTracker(fig1, ax1, A_fc_ind, A_fc_adj, 0.40, 'cool',
                        'femoral cartilage and adjacent bone voxels', subjectA)

fig1.canvas.mpl_connect('key_press_event', track1_1.on_key)
fig1.canvas.mpl_connect('key_press_event', track1_2.on_key)

plt.show()


# Subject B - Knee 1
fig2, ax2 = plt.subplots(1, 1, figsize=(14, 7))

track2_1 = IndexTracker(fig2, ax2, B_fc_ind, B_E1, 1, 'gray',
                        'DESS with adjacent Cartilage and Bone Voxel', subjectB)
track2_2 = IndexTracker(fig2, ax2, B_fc_ind, B_fc_adj, 0.40, 'cool',
                        'DESS with adjacent Cartilage and Bone Voxel', subjectB)

fig2.canvas.mpl_connect('key_press_event', track2_1.on_key)
fig2.canvas.mpl_connect('key_press_event', track2_2.on_key)

plt.show()
```

A.4 REGISTRATION OF PET AND $T_1\rho$ MAP

```
# %%
"""
# Jupyter Notebook to Register Images to DESS scan
"""

# %%
import os
print(os.getcwd())

%matplotlib inline
%load_ext autoreload
%autoreload 2

from knee_DWI.functions_knee_DWI import *
import os
import nibabel as nib
import numpy     as np
import SimpleITK as sitk
import glob
from dipy.core.gradients import gradient_table
nib.openers.Opener.default_compresslevel = 9

# %%
"""
## Defining Subject folder
"""

# %%
# Defining data folder:
subject = '13B'
infolder = '//Users/Gidega/Desktop/NaF_sub' + subject
print(infolder)

# %%
"""
## Creating folder to save nifti files
"""

# %%
outfolder = infolder + '/Nifti_data'
if not os.path.exists(outfolder):
        os.mkdir(outfolder)

if not os.path.exists(outfolder + '/DESS'):
        os.mkdir(outfolder + '/DESS')
if not os.path.exists(outfolder + '/PET'):
        os.mkdir(outfolder + '/PET')
if not os.path.exists(outfolder + '/T1p'):
        os.mkdir(outfolder + '/T1p')

# %%
"""
## Converting to nifti (run only once)
"""

# %%
dicom_folder = [infolder + '/DESSLow', infolder + '/End_of_' + subject[-1] + '_coreg', infolder + '/T1p-map'
nifti_folder = [outfolder + '/DESS', outfolder + '/PET', outfolder + '/T1p']

# %%
for i in range(len(dicom_folder)):
```

```
    dicom_to_nifti(dicom_folder[i],nifti_folder[i])

# %%
"""
## Import  data (in Nifti format)
"""

# %%
files = []
for i in range(len(nifti_folder)):
    file = glob.glob(nifti_folder[i]+'/*.nii.gz')[0]
    files.append(file)
print(files)
dess, dess_vox, dess_affine = read_nifti_data(files[0], rotate = True)
pet, pet_vox, pet_affine = read_nifti_data(files[1], rotate = True)
t1p, t1p_vox, t1p_affine = read_nifti_data(files[2], rotate = True)


show_dess_images(dess,'sagittal', 34) #  data, ori, slice number to be plotted
show_dess_images(t1p,'sagittal', 17) #  data, ori, slice number to be plotted
show_dess_images(pet,'sagittal', 60) #  data, ori, slice number to be plotted

# %%
"""
## Image registration to DESS scan
"""

# %%
temp = "/Users/Gidega/Desktop/knee_DWI"
os.chdir(temp)

print(dess_vox)
print(pet_vox)
print(t1p_vox)

# %%
"""
## PET Registration
"""

# %%
"""
### Rigid
"""

# %%
# Register PET to DESS, rigid:
pet_to_dess_rig = register_images(dess, dess_affine, pet, pet_affine,'rigid')

# %%
# Display PET registration:
show_registration_results(dess,pet_to_dess_rig,55)

# %%
"""
### Affine
"""

# %%
# Register PET to DESS, affine:
pet_to_dess_aff = register_images(dess, dess_affine, pet, pet_affine,'affine')

# %%
# Display PET registration:
show_registration_results(dess,pet_to_dess_aff,55)

# %%
```

```
"""
## T1p Registration
"""

# %%
"""
### Rigid
"""

# %%
# Register T1rho to DESS, rigid:
t1p_to_dess_rig = register_images(dess, dess_affine, t1p, t1p_affine,'rigid')

# %%
show_registration_results(dess,t1p_to_dess_rig,50)

# %%
"""
### Affine
"""

# %%
# Register T1rho to DESS, affine:
t1p_to_dess_aff = register_images(dess, dess_affine, t1p, t1p_affine,'affine')

# %%
show_registration_results(dess,t1p_to_dess_aff,50)

# %%
"""
### Bspline
"""

# %%
# Register T1rho to DESS, bspline:
t1p_to_dess_bsp = register_images(dess, dess_affine, t1p, t1p_affine,'bspline')

# %%
show_registration_results(dess,t1p_to_dess_bsp,50)

# %%
```

A.5 CONVERSION OF NAF VALUES TO SUV

```
# Importing necersary packages:
#------------------------------
from scipy.ndimage.morphology import binary_dilation, binary_erosion
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
from datetime import datetime
from natsort import natsorted
import matplotlib as mpl
from pathlib import Path
from scipy import stats
import nibabel as nib
```

```
import numpy as np
import scipy.io
import pydicom
import os


# Setting som plotting standards:
#--------------------------------
font = {'weight' : 'normal', 'size'   : 14}
mpl.rc('font', **font)


# Defining functions/classes used in the code:
#---------------------------------------------
# Function to collect all dcm files from a folder into lstFilesDCM:
def DCMFiles_List(DCM_Path):

    # Create an empty list to collect all .dcm files into it;
    lstFilesDCM = []

    # Loop to traverse the directory & collect all dcm files into lstFilesDCM;
    for dirName, subdirList, fileList in os.walk(DCM_Path):
        # Sorting the fileList;
        #fileList.sort()
        fileList = natsorted(fileList)
        for filename in fileList:
            # Check whether the file's DICOM;
            if ".dcm" in filename.lower():
                # Storing files in list;
                lstFilesDCM.append(os.path.join(dirName,filename))

    return lstFilesDCM


# Function to load T2 & Naf data:
def Load_T2Naf(file):

    # Loading data file;
    loaded_file = np.load(file)

    # Loading T2Naf arrays from the file;
    T2Naf = loaded_file['T2Naf']
    T2Naf_TM = loaded_file['T2Naf_TM']
    T2Naf_TL = loaded_file['T2Naf_TL']
    T2Naf_CM = loaded_file['T2Naf_CM']
    T2Naf_CL = loaded_file['T2Naf_CL']
    T2Naf_PM = loaded_file['T2Naf_PM']
    T2Naf_PL = loaded_file['T2Naf_PL']

    return T2Naf, T2Naf_TM, T2Naf_TL, T2Naf_CM, T2Naf_CL, T2Naf_PM, T2Naf_PL


# Function to load T1p & Naf data:
def Load_T1pNaf(file):

    # Loading data file;
    loaded_file = np.load(file)

    # Loading T1pNaf arrays from the file;
    T1pNaf = loaded_file['T1pNaf']
    T1pNaf_TM = loaded_file['T1pNaf_TM']
    T1pNaf_TL = loaded_file['T1pNaf_TL']
    T1pNaf_CM = loaded_file['T1pNaf_CM']
    T1pNaf_CL = loaded_file['T1pNaf_CL']
    T1pNaf_PM = loaded_file['T1pNaf_PM']
    T1pNaf_PL = loaded_file['T1pNaf_PL']

```

```
    return T1pNaf, T1pNaf_TM, T1pNaf_TL, T1pNaf_CM, T1pNaf_CL, T1pNaf_PM, T1pNaf_PL


#-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-#


# Defining the subject folder & Knee (Patient-folder):
#------------------------------------------------------
NaF = 'NaF_sub'
knee = 'Knee1'
subject = 13
subjectA = str(subject) + 'A'
subjectB = str(subject) + 'B'
Patient_folderA = NaF + subjectA
Patient_folderB = NaF + subjectB


# Reading the activity and body weight of the subjects from mat.file:
#---------------------------------------------------------------------
# Loading inj.mat file;
file = scipy.io.loadmat('inj.mat')

# Getting the data from the .mat file;
subA = (file['Inj'][0,subject-1])['A'][0,0]
A_act = int(subA[0])*(10**(-3)) #Activity in kBq (k; 10^3)

subB = (file['Inj'][0,subject-1])['B'][0,0]
B_act = int(subB[0])*(10**(-3)) #Activity in kBq (k; 10^3)

BW = float((file['Inj'][0,subject-1])['weight']) # Subjects Body weight in kg


# Computing the Cinj using intial activity (injection Dose) and  (Body weight);
A_Cinj = A_act/BW
B_Cinj = B_act/BW


# #-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-#
#                              # T2Naf_data Converting
# #-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-#


# Path to T2Naf_data npz-file:
#-------------------------------
A_T2Naf_path = 'T2Naf_data/' + subjectA + knee + '_T2Naf.npz'
B_T2Naf_path = 'T2Naf_data/' + subjectB + knee + '_T2Naf.npz'


# Loading the T2Naf data:
#-------------------------
A_T2Naf, A_T2Naf_TM, A_T2Naf_TL, A_T2Naf_CM, A_T2Naf_CL, A_T2Naf_PM, A_T2Naf_PL = Load_T2Naf(A_T2Naf_path)
B_T2Naf, B_T2Naf_TM, B_T2Naf_TL, B_T2Naf_CM, B_T2Naf_CL, B_T2Naf_PM, B_T2Naf_PL = Load_T2Naf(B_T2Naf_path)


# Converting NaF to SUV and saving to T2SUV variabel:
#-----------------------------------------------------
# Saving T2Naf data into new array called T2SUV;
A_T2SUV = A_T2Naf
A_T2SUV_TM = A_T2Naf_TM
A_T2SUV_TL = A_T2Naf_TL
A_T2SUV_CM = A_T2Naf_CM
A_T2SUV_CL = A_T2Naf_CL
A_T2SUV_PM = A_T2Naf_PM
A_T2SUV_PL = A_T2Naf_PL

B_T2SUV = B_T2Naf
B_T2SUV_TM = B_T2Naf_TM
```

```
B_T2SUV_TL = B_T2Naf_TL
B_T2SUV_CM = B_T2Naf_CM
B_T2SUV_CL = B_T2Naf_CL
B_T2SUV_PM = B_T2Naf_PM
B_T2SUV_PL = B_T2Naf_PL

# Converting Naf to SUV;
A_T2SUV[:,1] = A_T2SUV[:,1]/A_Cinj
A_T2SUV_TM[:,1] = A_T2SUV_TM[:,1]/A_Cinj
A_T2SUV_TL[:,1] = A_T2SUV_TL[:,1]/A_Cinj
A_T2SUV_CM[:,1] = A_T2SUV_CM[:,1]/A_Cinj
A_T2SUV_CL[:,1] = A_T2SUV_CL[:,1]/A_Cinj
A_T2SUV_PM[:,1] = A_T2SUV_PM[:,1]/A_Cinj
A_T2SUV_PL[:,1] = A_T2SUV_PL[:,1]/A_Cinj

B_T2SUV[:,1] = B_T2SUV[:,1]/B_Cinj
B_T2SUV_TM[:,1] = B_T2SUV_TM[:,1]/B_Cinj
B_T2SUV_TL[:,1] = B_T2SUV_TL[:,1]/B_Cinj
B_T2SUV_CM[:,1] = B_T2SUV_CM[:,1]/B_Cinj
B_T2SUV_CL[:,1] = B_T2SUV_CL[:,1]/B_Cinj
B_T2SUV_PM[:,1] = B_T2SUV_PM[:,1]/B_Cinj
B_T2SUV_PL[:,1] = B_T2SUV_PL[:,1]/B_Cinj


# Saving T2SUV data to file in compressed .npz format:
# ----------------------------------------------------
np.savez_compressed(subjectA + knee + '_T2SUV',T2SUV=A_T2SUV,T2SUV_TM=A_T2SUV_TM,T2SUV_TL=A_T2SUV_TL,
                    T2SUV_CM=A_T2SUV_CM,T2SUV_CL=A_T2SUV_CL,T2SUV_PM=A_T2SUV_PM,T2SUV_PL=A_T2SUV_PL)
np.savez_compressed(subjectB + knee + '_T2SUV',T2SUV=B_T2SUV,T2SUV_TM=B_T2SUV_TM,T2SUV_TL=B_T2SUV_TL,
                    T2SUV_CM=B_T2SUV_CM,T2SUV_CL=B_T2SUV_CL,T2SUV_PM=B_T2SUV_PM,T2SUV_PL=B_T2SUV_PL)


#-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-#
                            # T1pNaf_data Converting
#-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-#


# # Path to T1pNaf_data npz-file:
# #------------------------------
# A_T1pNaf_path = 'T1pNaf_data/' + subjectA + knee + '_T1pNaf.npz'
# B_T1pNaf_path = 'T1pNaf_data/' + subjectB + knee + '_T1pNaf.npz'


# # Loading the T1pNaf data:
# #-------------------------
# A_T1pNaf, A_T1pNaf_TM, A_T1pNaf_TL, A_T1pNaf_CM, A_T1pNaf_CL, A_T1pNaf_PM, A_T1pNaf_PL = Load_T1pNaf(A_T1p
# B_T1pNaf, B_T1pNaf_TM, B_T1pNaf_TL, B_T1pNaf_CM, B_T1pNaf_CL, B_T1pNaf_PM, B_T1pNaf_PL = Load_T1pNaf(B_T1p


# # Converting NaF to SUV and saving to T1pSUV variabel:
# #-----------------------------------------------------
# # Saving T1pNaf data into new array called T1pSUV;
# A_T1pSUV = A_T1pNaf
# A_T1pSUV_TM = A_T1pNaf_TM
# A_T1pSUV_TL = A_T1pNaf_TL
# A_T1pSUV_CM = A_T1pNaf_CM
# A_T1pSUV_CL = A_T1pNaf_CL
# A_T1pSUV_PM = A_T1pNaf_PM
# A_T1pSUV_PL = A_T1pNaf_PL

# B_T1pSUV = B_T1pNaf
# B_T1pSUV_TM = B_T1pNaf_TM
# B_T1pSUV_TL = B_T1pNaf_TL
# B_T1pSUV_CM = B_T1pNaf_CM
# B_T1pSUV_CL = B_T1pNaf_CL
# B_T1pSUV_PM = B_T1pNaf_PM
# B_T1pSUV_PL = B_T1pNaf_PL
```

```
220
221 # # Converting Naf to SUV;
222 # A_T1pSUV[:,1] = A_T1pSUV[:,1]/A_Cinj
223 # A_T1pSUV_TM[:,1] = A_T1pSUV_TM[:,1]/A_Cinj
224 # A_T1pSUV_TL[:,1] = A_T1pSUV_TL[:,1]/A_Cinj
225 # A_T1pSUV_CM[:,1] = A_T1pSUV_CM[:,1]/A_Cinj
226 # A_T1pSUV_CL[:,1] = A_T1pSUV_CL[:,1]/A_Cinj
227 # A_T1pSUV_PM[:,1] = A_T1pSUV_PM[:,1]/A_Cinj
228 # A_T1pSUV_PL[:,1] = A_T1pSUV_PL[:,1]/A_Cinj
229
230 # B_T1pSUV[:,1] = B_T1pSUV[:,1]/B_Cinj
231 # B_T1pSUV_TM[:,1] = B_T1pSUV_TM[:,1]/B_Cinj
232 # B_T1pSUV_TL[:,1] = B_T1pSUV_TL[:,1]/B_Cinj
233 # B_T1pSUV_CM[:,1] = B_T1pSUV_CM[:,1]/B_Cinj
234 # B_T1pSUV_CL[:,1] = B_T1pSUV_CL[:,1]/B_Cinj
235 # B_T1pSUV_PM[:,1] = B_T1pSUV_PM[:,1]/B_Cinj
236 # B_T1pSUV_PL[:,1] = B_T1pSUV_PL[:,1]/B_Cinj
237
238
239 # # Saving T1pSUV data to file in compressed .npz format:
240 # # ----------------------------------------------------
241 # np.savez_compressed(subjectA + knee + '_T1pSUV',T1pSUV=A_T1pSUV,T1pSUV_TM=A_T1pSUV_TM,T1pSUV_TL=A_T1pSUV_T
242 #                     T1pSUV_CM=A_T1pSUV_CM,T1pSUV_CL=A_T1pSUV_CL,T1pSUV_PM=A_T1pSUV_PM,T1pSUV_PL=A_T1pSUV_P
243 # np.savez_compressed(subjectB + knee + '_T1pSUV',T1pSUV=B_T1pSUV,T1pSUV_TM=B_T1pSUV_TM,T1pSUV_TL=B_T1pSUV_T
244 #                     T1pSUV_CM=B_T1pSUV_CM,T1pSUV_CL=B_T1pSUV_CL,T1pSUV_PM=B_T1pSUV_PM,T1pSUV_PL=B_T1pSUV_P
```

B

MATLAB & PYTHON CODE - GLOSTRUP STUDY

B.1 FEMORAL CARTILAGE ROI DRAWING

```
%% Clearing and closing to free space:

clear
clc
close all

%% Drawing ROI's on all slices loaded from folder:

% Folder where the dcm files choosen to draw on will be selected from:
folder=uigetdir('/Users/Gidega/Desktop/Bryan/','choose folder with files');
files=ev_ls(fullfile(folder,'*.dcm'));

% Loop to store all slices of the scan in matrix:
for x=1:length(files)
    D_info(x).header=dicominfo(files{x});
    mat(:,:,x)=dicomread(files{x});
end

mat(mat>prctile(mat(:),80))=prctile(mat(:),80);
rip(mat)
display('Press any key to continue')
pause

%% Saving the drawed ROI mask to matrix:

ROI=getmask(RipRoi);
mask=ROI{1};


%% Writting the ROI mask to dcm files:

dim=size(mat);
w = warning ('off','all');
numstr='0000';
SeriesInstanceUID = dicomuid; %gives new ID to series

[path,file]=fileparts(folder);

```

```
mkdir(fullfile(path,[file,'_ROI']))
for x=1:dim(3)
    num=[numstr,num2str(x)];
    num=num(end-4:end);
    info=D_info(x).header;
    info.NumberOfTimeSlices=int16(1);
    info.SeriesDescription='Cartilage ROI';
    info.SeriesNumber=info.SeriesNumber+1;
    info.SeriesInstanceUID = SeriesInstanceUID;
    info.RescaleSlope=info.RescaleSlope/info.RescaleSlope; %makes it 1 in the right format
    info.RescaleIntercept=info.RescaleIntercept-info.RescaleIntercept;
    names=fieldnames(info);
    ind=regexpi(names,'xx_Creator');
    ind=cellfun(@isempty,ind);
    info=rmfield(info,{names{ind==0}});
    dicomwrite(int16(mask(:,:,x)),[fullfile(path,[file,'_ROI']),'/Im',num,'.dcm'],info, 'CreateMode', 'copy'
end
```

B.2 ROI MASK

```
# Importing necersary packages:
#------------------------------
from scipy.ndimage.morphology import binary_dilation, binary_erosion
import matplotlib.pyplot as plt
from natsort import natsorted
import matplotlib as mpl
import numpy as np
import pydicom
import os


# Setting som plotting standards:
#--------------------------------
font = {'weight' : 'normal', 'size' : 18}
mpl.rc('font', **font)


# Defining functions used in the code:
#-------------------------------------
# Function to collect all dcm files from a folder into lstFilesDCM:
def DCMFiles_List(DCM_Path):

    # Create an empty list to collect all .dcm files into it;
    lstFilesDCM = []

    # Loop to traverse the directory & collect all dcm files into lstFilesDCM;
    for dirName, subdirList, fileList in os.walk(DCM_Path):
        # Sorting the fileList;
        #fileList.sort()
        fileList = natsorted(fileList) # naturally sort
        for filename in fileList:
            # Check whether the file's DICOM;
            if ".dcm" in filename.lower():
                # Storing files in list;
                lstFilesDCM.append(os.path.join(dirName,filename))
```

```
45
46      return lstFilesDCM
47
48
49  # Function to import raw DICOM data into an Pixel array:
50  def DCM_Import(DCM_Path):
51
52      # Collecting all dcm files from the folder into a list;
53      lstFilesDCM = DCMFiles_List(DCM_Path)
54
55      # Reference to extract metadata:
56      # Get refference file (first slice);
57      RefDs = pydicom.dcmread(lstFilesDCM[0])
58
59      # Load dimensions based on the number of rows, columns, and slices;
60      ConstPixelDims = (int(RefDs.Rows), int(RefDs.Columns), len(lstFilesDCM))
61
62      # Storring the raw DICOM data:
63      # Creating an array size is based on 'ConstPixelDims';
64      ArrayDicom = np.zeros(ConstPixelDims, dtype=RefDs.pixel_array.dtype)
65
66      # Creating array to store the Rescale Slope & Rescale Intercept for all slices;
67      RescaleSlope = np.ones(len(lstFilesDCM))
68      RescaleIntercept = np.zeros(len(lstFilesDCM))
69
70      # Loop through all the DICOM files;
71      for filenameDCM in lstFilesDCM:
72
73          # Read the file;
74          ds = pydicom.dcmread(filenameDCM)
75
76          # Storing raw data;
77          ArrayDicom[:, :, lstFilesDCM.index(filenameDCM)] = ds.pixel_array
78
79          # Correcting for Rescale tags if they exist;
80          if ("RescaleIntercept" in ds) == True:
81
82              RescaleSlope[lstFilesDCM.index(filenameDCM)] = ds.RescaleSlope
83              RescaleIntercept[lstFilesDCM.index(filenameDCM)] = ds.RescaleIntercept
84
85      # Correcting for the rescale tags;
86      ArrayDicom = ArrayDicom * RescaleSlope + RescaleIntercept
87
88      return ArrayDicom
89
90
91  # Class to plot and scroll trough the slices using key_press_event:
92  class IndexTracker:
93
94      # Initial definitions for plot;
95      def __init__(self, fig, ax, ind, X, alpha, cmap, title):
96
97          self.fig = fig
98          self.ind = ind
99          self.cmap = cmap
100         self.alpha = alpha
101         self.ax = ax
102         self.X = X
103         rows, cols, self.slices = X.shape
104
105         self.im = ax.imshow(self.X[30:150, 25:175, self.ind], alpha=self.alpha,
106                             cmap=self.cmap)
107         self.ax.set_title(title)
108         self.ax.axis('off')
109         self.update()
110
111         # # Creates colorbar if cmap is 'jet';
```

```
        # if self.cmap == 'jet':
        #     cbar = self.fig.colorbar(self.im, orientation="horizontal",
        #                              ax=self.ax, fraction=0.046, pad=0.04)
        #     cbar.mappable.set_clim(vmin=-10,vmax=300)

    # Key definition for navigation of slices;
    def on_key(self, event):

        if event.key == 'up':
            self.ind = (self.ind + 1) % self.slices
        elif event.key == 'down':
            self.ind = (self.ind - 1) % self.slices
        self.update()

    # Function to update plot;
    def update(self):

        self.im.set_data(self.X[30:150, 25:175, self.ind])
        #self.fig.suptitle('Slice ' + str(self.ind+1) + ' of subject ' + subject)
        self.fig.suptitle('   Slice ' + str(self.ind+1) + ' of baseline TSE scan with')
        self.im.axes.figure.canvas.draw()


# Function to classify regions of ROI mask:
def Regions_ROI(ROI,Threshold):

    # Array to store corrected ROI mask with 3 regions;
    ROI_regions = np.zeros((ROI.shape[0],ROI.shape[1],ROI.shape[2]))

    # Loop to classify the regions of femural cartilage (ROI):
    for s in range(ROI.shape[2]): # Slice
        for x in range(ROI.shape[1]): # Row
            for y in range(ROI.shape[0]): # Column

                # Femur Troclea Cartilage;
                if y <= Threshold[0] and ROI[x,y,s] > 0:

                    ROI_regions[x,y,s] = 1

                # Femur Central Cartilage;
                if y >= Threshold[0] and y <= Threshold[1]:
                    if ROI[x,y,s] > 0:

                        ROI_regions[x,y,s] = 2

                # Femur Posterior Cartilage;
                if y >= Threshold[1] and ROI[x,y,s] > 0:

                        ROI_regions[x,y,s] = 3

    return ROI_regions


# Function to write the Region ROI Mask back to the DICOM files:
def WritingDCM_ROIMask(ROI_Path, ROI_Mask):

    # Collecting all dcm files from the folder into a list;
    lstFilesDCM = DCMFiles_List(ROI_Path)

    # Get refference file (first slice);
    RefDs = pydicom.dcmread(lstFilesDCM[0])

    # Load dimensions based on the number of rows, columns, and slices;
    ConstPixelDims = (int(RefDs.Rows), int(RefDs.Columns), len(lstFilesDCM))

    # Creating an array to store dcm data size is based on 'ConstPixelDims';
    Region_ROI = np.zeros(ConstPixelDims, dtype=RefDs.pixel_array.dtype)
```

```
179
180     # Loop through all the DCM files to read-store-write to files again;
181     for filenameDCM in lstFilesDCM:
182
183         # Read the dcm file;
184         ds = pydicom.dcmread(filenameDCM)
185
186         # Storing the Region ROI Mask;
187         Region_ROI[:, :, lstFilesDCM.index(filenameDCM)] = ROI_Mask[:, :, lstFilesDCM.index(filenameDCM)]
188
189         # Writing the ROI_Region Mask into the files;
190         ds.PixelData = Region_ROI[:, :, lstFilesDCM.index(filenameDCM)].tobytes()
191
192         # Saving the files;
193         ds.save_as(filenameDCM)
194
195 #-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-#
196                             # Importing Dicom files
197 #-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-#
198
199
200 # Defining the subject folder(Patient-folder):
201 #---------------------------------------------
202 seq = '2'
203 subject = 'Thomas'
204 T2Map_path = subject + '/T2Map_' + seq + '01'
205 TSE_path = subject + '/TSE_' + seq + '01'
206 ROI_path = subject + '/ROI_' + seq + '01'
207
208
209 # Defining the threhold of the fc regions:
210 #-----------------------------------------
211 # Threshold for the 3 regions of fc (Troclea-Central-Posterior);
212 Threshold = [75,125]
213
214
215 # Importing the T2-map:
216 #----------------------
217 # Importing DICOM files, using the function from above;
218 T2 =  np.float64(DCM_Import(T2Map_path + '/'))
219
220
221 # Importing the TSE-scan:
222 #------------------------
223 # Importing DICOM files, using the function from above;
224 TSE =  np.float64(DCM_Import(TSE_path + '/'))
225
226
227 # Importing the ROI mask:
228 #------------------------
229 # Importing DICOM files, using the function from above;
230 ROI =  np.float64(DCM_Import(ROI_path + '/'))
231
232
233 #-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-#
234                         # Dispalying ROI on TSE & T2
235 #-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-#
236
237
238 # Displaying DICOM files :
239 #-------------------------
240
241 # Defining all zero values in ROI to nan for plotting;
242 ROI[ROI == 0] = np.nan
243
244 # Plotting the ROI mask drawn in Matlab on T2 Map;
245 fig, ax = plt.subplots(1, 1, figsize=(10, 20))
```

```
246
247 track_1 = IndexTracker(fig, ax, 0, T2, 1, 'jet',
248                         'ROI mask of fc on T2 Map - seq. ' + seq)
249 track_2 = IndexTracker(fig, ax, 0, ROI, 0.3, 'hsv',
250                         'ROI mask of fc on T2 Map - seq. ' + seq)
251
252 fig.canvas.mpl_connect('key_press_event', track_2.on_key)
253 fig.canvas.mpl_connect('key_press_event', track_1.on_key)
254
255 plt.show()
256
257
258 # Plotting the ROI mask drawn in Matlab on TSE scan;
259 fig1, ax1 = plt.subplots(1, 1, figsize=(10, 20))
260
261 track1_1 = IndexTracker(fig1, ax1, 0, TSE, 1, 'gray',
262                         'femoral cartilage region mask')
263 track1_2 = IndexTracker(fig1, ax1, 0, ROI, 0.3, 'jet',
264                         'femoral cartilage region mask')
265
266 fig1.canvas.mpl_connect('key_press_event', track1_2.on_key)
267 fig1.canvas.mpl_connect('key_press_event', track1_1.on_key)
268
269
270 plt.show()
271
272
273 #-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-#
274             # Classifying the regions of the fc ROI and displaying
275 #-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-#
276
277
278 # Classifying the regions of fc ROI mask:
279 #----------------------------------------
280 ROI_regions = Regions_ROI(ROI,Threshold)
281
282
283 # Displaying DICOM files :
284 #-------------------------
285 # Defining all zero values in ROI to nan for plotting;
286 ROI_regions[ROI_regions == 0] = np.nan
287
288 # Plotting the ROI mask drawn in Matlab on TSE scan;
289 fig1, ax1 = plt.subplots(1, 1, figsize=(10, 20))
290
291 track1_1 = IndexTracker(fig1, ax1, 0, TSE, 1, 'gray',
292                         'Corrected ROI mask on TSE scan - seq. ' + seq)
293 track1_2 = IndexTracker(fig1, ax1, 0, ROI_regions, 0.15, 'gist_rainbow',
294                         'Corrected ROI mask on TSE scan - seq. ' + seq)
295
296 fig1.canvas.mpl_connect('key_press_event', track1_2.on_key)
297 fig1.canvas.mpl_connect('key_press_event', track1_1.on_key)
298
299
300 plt.show()
301
302
303 #-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-#
304             # Writting the regions of fc ROI back to the dcm files
305 #-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-#
306
307
308 # Classifying the regions of fc ROI mask:
309 #----------------------------------------
310 ROI_regions = Regions_ROI(ROI,Threshold)
311
312
```

```
# Writting the corrected ROI to the dcm files:
#----------------------------------------------
#-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-#
WritingDCM_ROIMask(ROI_path, ROI_regions)
#-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-#
```

B.3 T_2 MAP CALCULATION

```
# Importing necersary packages:
#------------------------------
from scipy.optimize import curve_fit
import matplotlib.pyplot as plt
from natsort import natsorted
import matplotlib as mpl
import numpy as np
import pydicom
import os


# Setting som plotting standards:
#--------------------------------
font = {'weight' : 'normal', 'size' : 18}
mpl.rc('font', **font)


# Defining functions used in the code:
#-------------------------------------
# Function to collect all dcm files from a folder into lstFilesDCM:
def DCMFiles_List(DCM_Path):

    # Create an empty list to collect all .dcm files into it;
    lstFilesDCM = []

    # Loop to traverse the directory & collect all dcm files into lstFilesDCM;
    for dirName, subdirList, fileList in os.walk(DCM_Path):
        # Sorting the fileList;
        #fileList.sort()
        fileList = natsorted(fileList) # naturally sort
        for filename in fileList:
            # Check whether the file's DICOM;
            if ".dcm" in filename.lower():
                # Storing files in list;
                lstFilesDCM.append(os.path.join(dirName,filename))

    return lstFilesDCM


# Function to import raw DICOM data into an Pixel array:
def DCM_Import(DCM_Path):

    # Collecting all dcm files from the folder into a list;
    lstFilesDCM = DCMFiles_List(DCM_Path)

    # Reference to extract metadata:
    # Get refference file (first slice);
```

```
    RefDs = pydicom.dcmread(lstFilesDCM[0])

    # Load dimensions based on the number of rows, columns, and slices;
    ConstPixelDims = (int(RefDs.Rows), int(RefDs.Columns), len(lstFilesDCM))

    # Storring the raw DICOM data:
    # Creating an array size is based on 'ConstPixelDims';
    ArrayDicom = np.zeros(ConstPixelDims, dtype=RefDs.pixel_array.dtype)

    # Creating array to store the Rescale Slope & Rescale Intercept for all slices;
    RescaleSlope = np.ones(len(lstFilesDCM))
    RescaleIntercept = np.zeros(len(lstFilesDCM))

    # Loop through all the DICOM files;
    for filenameDCM in lstFilesDCM:

        # Read the file;
        ds = pydicom.dcmread(filenameDCM)

        # Storing raw data;
        ArrayDicom[:, :, lstFilesDCM.index(filenameDCM)] = ds.pixel_array

        # Correcting for Rescale tags if they exist;
        if ("RescaleIntercept" in ds) == True:

            RescaleSlope[lstFilesDCM.index(filenameDCM)] = ds.RescaleSlope
            RescaleIntercept[lstFilesDCM.index(filenameDCM)] = ds.RescaleIntercept

    # Correcting for the rescale tags;
    ArrayDicom = ArrayDicom * RescaleSlope + RescaleIntercept

    return ArrayDicom


# Class to plot and scroll trough the slices using key_press_event:
class IndexTracker:

    # Initial definitions for plot;
    def __init__(self, fig, ax, ind, X, alpha, cmap, title):

        self.fig = fig
        self.ind = ind
        self.cmap = cmap
        self.alpha = alpha
        self.ax = ax
        self.X = X
        rows, cols, self.slices = X.shape

        self.im = ax.imshow(self.X[30:150, 25:175, self.ind], alpha=self.alpha,
                            cmap=self.cmap)
        self.ax.set_title(title)
        self.ax.axis('off')
        self.update()

        # Creates colorbar if cmap is 'jet';
        if self.cmap == 'jet':
            cbar = self.fig.colorbar(self.im, orientation="horizontal",
                                     ax=self.ax, fraction=0.0566, pad=0.04)
            cbar.mappable.set_clim(vmin=0,vmax=100)

    # Key definition for navigation of slices;
    def on_key(self, event):

        if event.key == 'up':
            self.ind = (self.ind + 1) % self.slices
        elif event.key == 'down':
            self.ind = (self.ind - 1) % self.slices
```

```
124         self.update()
125 
126     # Function to update plot;
127     def update(self):
128 
129         self.im.set_data(self.X[30:150, 25:175, self.ind])
130         self.fig.suptitle('   Slice ' + str(self.ind+1) + ' of baseline T2 map')
131         self.im.axes.figure.canvas.draw()
132 
133 
134 # Exponential function that will be fitted to the Signals of 6 Echotimes:
135 def Exp_func(TE, S_0, T2, C):
136     return S_0 * np.exp(-TE / T2) + C
137 
138 # Function to Compute the T2 Map from TSE scans:
139 def Comp_T2Map(ROI,TSE):
140 
141     # Array to stor T2 map of the one knee slice with ROI:
142     T2_map = np.zeros((ROI.shape[0],ROI.shape[1],ROI.shape[2]))
143     # Array to store error of T2 map for the one knee slice with ROI:
144     T2_err = np.zeros((ROI.shape[0],ROI.shape[1],ROI.shape[2]))
145 
146 
147     # Loop to calculate T2 Map of knee fc ROI:
148     for s in range(ROI.shape[2]):
149 
150         # Arrays to store index of fc ROI - fc index row and col;
151         fc_row = []
152         fc_col = []
153 
154         # Index for fc ROI in ROI_slice:
155         fc_row, fc_col = np.where(ROI[:,:,s] > 0)
156 
157 
158         for pixel in range(len(fc_row)):
159 
160             # Storing the Signals(y) and the corresponding Echotimes(x) for one pixel:
161             # Arrays with the 6 Echotimes (TE):
162             x = np.array((10,20,30,40,50,60),dtype=int)
163 
164             # Array to store Signals from the 6 different Echotimes:
165             y = np.zeros(len(x),dtype=float)
166 
167             # Slice indicator for the one slice from different TE scans:
168             slice_TE = s # defining the initial slice - for TSE TE = 10
169 
170             # Loop to store signal from same slice of knee for all 6 Echotimes:
171             for TE in range(len(y)):
172 
173                 # Storing signals (y) for the 6 different Echotimes:
174                 y[TE] = TSE[fc_row[pixel],fc_col[pixel],slice_TE]
175 
176                 # Updating slice indicator for next Echotime scan:
177                 slice_TE += 8 # One Echotime scan consist of 8 slices
178 
179 
180             # Fitting the exponential function to the data stored above (Signal & TE):
181             # Initial values for the parameters:
182             Init_vals = [y[0] * 1.1, 40, 100]
183             # Minimum values for the parameters:
184             Min_vals = [y[0] / 2, 0, -100]
185             # Maximum values for the parameters:
186             Max_vals = [y[0] * 1.7, 100, 500]
187 
188             # Fitting the data to the exponential function:
189             p_opt, p_cov = curve_fit(Exp_func, x, y, p0=Init_vals, bounds=(Min_vals, Max_vals), method='trf'
190 
```

```
            # Standard deviation errors on the parameters
            p_err = np.sqrt(np.diag(p_cov))

            # Storring the T2 and corresponding error in arrays:
            T2_map[fc_row[pixel],fc_col[pixel],s] = p_opt[1]
            T2_err[fc_row[pixel],fc_col[pixel],s] = p_err[1]

    return T2_map


# Function to write the Computed T2_Map back to DICOM files:
def WritingDCM_T2Map(ComputedT2Map_Path, ComputedT2_Map):

    # Collecting all dcm files from the folder into a list;
    lstFilesDCM = DCMFiles_List(ComputedT2Map_Path)

    # Get refference file (first slice);
    RefDs = pydicom.dcmread(lstFilesDCM[0])

    # Load dimensions based on the number of rows, columns, and slices;
    ConstPixelDims = (int(RefDs.Rows), int(RefDs.Columns), len(lstFilesDCM))

    # Creating an array to store dcm data size is based on 'ConstPixelDims';
    T2_Map = np.zeros(ConstPixelDims, dtype=RefDs.pixel_array.dtype)

    # Loop through all the DCM files to read-store-write to files again;
    for filenameDCM in lstFilesDCM:

        # Read the dcm file;
        ds = pydicom.dcmread(filenameDCM)

        # Storing the T2_Map;
        T2_Map[:, :, lstFilesDCM.index(filenameDCM)] = ComputedT2_Map[:, :, lstFilesDCM.index(filenameDCM)]

        # Writing the ROI_Region Mask into the files;
        ds.PixelData = T2_Map[:, :, lstFilesDCM.index(filenameDCM)].tobytes()

        # Modyfing the description and series tag;
        ds.SeriesDescription = 'Computed T2 Map'
        ds.SeriesNumber = ds.SeriesNumber + 2

        # Saving the files;
        ds.save_as(filenameDCM)


#-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-#
                            # Importing Dicom files
#-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-#


# Defining the subject folder(Patient-folder) and seq. and slice:
#----------------------------------------------------------------
seq = '2'
subject = 'Thomas'
TSE_path = subject + '/TSE_' + seq + '01'
ROI_path = subject + '/ROI_' + seq + '01'
Comp_T2Map_path = subject + '/Comp_T2Map_' + seq + '01'


# Importing the TSE-scan:
#------------------------
# Importing DICOM files, using the function from above;
TSE =  np.float64(DCM_Import(TSE_path + '/'))


# Importing the ROI mask:
#------------------------
```

```
# Importing DICOM files, using the function from above;
ROI =  np.float64(DCM_Import(ROI_path + '/'))


# Importing the T2-map:
#-----------------------
# Importing DICOM files, using the function from above;
Comp_T2 =  np.float64(DCM_Import(Comp_T2Map_path + '/'))


#-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-#
                            # Dispalying TSE and ROI
#-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-#


# Displaying scans and ROI/T2 :
#------------------------------
# Defining all zero values in ROI to nan for plotting;
Comp_T2[Comp_T2 == 0] = np.nan

# Plotting the ROI mask drawn in Matlab on TSE scan;
fig1, ax1 = plt.subplots(1, 1, figsize=(10, 20))

track1_1 = IndexTracker(fig1, ax1, 0, TSE, 1, 'gray',
                        'for only fc region on TSE scan')
track1_2 = IndexTracker(fig1, ax1, 0, Comp_T2, 1, 'jet',
                        'for only fc region on TSE scan')

fig1.canvas.mpl_connect('key_press_event', track1_2.on_key)
fig1.canvas.mpl_connect('key_press_event', track1_1.on_key)


plt.show()


#-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-#
        # Calculate T2 Map of fc ROI from TSE scan and Displaying
#-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-#


# Calculate T2 Map of fc ROI from TSE scan:
#------------------------------------------
T2_Map = Comp_T2Map(ROI,TSE)


# Plotting TSE with the calculated T2 map on top:
#------------------------------------------------
# Defining all zero values in ROI to nan for plotting;
T2_Map[T2_Map == 0] = np.nan

# Plotting the ROI mask drawn in Matlab on TSE scan;
fig1, ax1 = plt.subplots(1, 1, figsize=(10, 20))

track1_1 = IndexTracker(fig1, ax1, 0, TSE, 1, 'gray',
                        'Computed T2 Map of fc on TSE scan - seq. ' + seq)
track1_2 = IndexTracker(fig1, ax1, 0, T2_Map, 0.4, 'jet',
                        'Computed T2 Map of fc on TSE scan - seq. ' + seq)

fig1.canvas.mpl_connect('key_press_event', track1_2.on_key)
fig1.canvas.mpl_connect('key_press_event', track1_1.on_key)

plt.show()


#-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-#
                    # Write the computed T2 map to dcm files
#-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-#
```

```
325
326
327 # Calculate T2 Map of fc ROI from TSE scan:
328 #-----------------------------------------
329 T2_Map = Comp_T2Map(ROI,TSE)
330
331
332 # Writting the computed T2 Map to the dcm files:
333 #-----------------------------------------------
334 #-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-#
335 WritingDCM_T2Map(Comp_T2Map_path, T2_Map)
336 #-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-#
```