Die approbierte Originalversion dieser Diplom-/Masterarbeit ist an der Hauptbibliothek der Technischen Universität Wien aufgestellt (http://www.ub.tuwien.ac.at).

The approved original version of this diploma or master thesis is available at the main library of the Vienna University of Technology (http://www.ub.tuwien.ac.at/englweb/).

Unterschrift (Betreuer)

TECHNISCHE
UNIVERSITÄT
WIEN

VIENNA
UNIVERSITY OF
TECHNOLOGY

DIPLOMARBEIT

Computer-Assisted Localization of Mice Organs in Micro Positron Emission Tomography

ausgeführt am

Institut für Festkörperphysik
der Technischen Universität Wien
und
Austrian Research Centers, Seibersdorf

unter der Anleitung von

Ao. Univ.-Prof. DI. Dr. Norbert Gurker
und
DI. Dr. Claudia Kuntner

durch

Robert Kremslehner
Waidhofnerstraße 65
3300 Amstetten

Wien, August 2007

Unterschrift (Student)

Abstract

MicroPET (Positron Emission Tomography) is an imaging method used for small animals. It is a small scale equivalent to PET applied in medical diagnosis. The major drawback of PET is, that it does not provide precise anatomical information, because it just shows the distribution of the injected radio labeled molecules and no further body structure.

In order to gain quantitative information about the distribution of the tracer in organs, ROIs (Regions of Interest) have to be defined. Due to observing the measured activity in ROIs for a series of time frames, the time dependent distribution of the tracer can be calculated. Normally the ROIs have to be manually defined. Unfortunately the selected regions greatly depend on the user defining them, and selecting the regions may also be quite time consuming.

The software developed during this thesis simplifies and objectivizes the selection of ROIs of reconstructed PET images for mice. First, the user selects one or more clearly visible organs. On the basis of these user defined regions, the software calculates the positions of the remaining organs. This is done by adjusting the PET image to the digital map of an “average mouse” derived from microMRT (Magnetic Resonance Tomography). Therefore the organs provided by this mouse phantom are rescaled and rotated, until the organs reach their best overlap with the actual PET image. Thereby the location of the remaining organs can be estimated.

This document starts with a short introduction into the aim of the work. Chapter 2 provides a general survey of PET and a short review of tracer production. In Chapter 3 the developed software is introduced, and its functional principles are discussed in detail. In Section 4.1 the microPET-instrument and measurement preparations are presented, and in Section 4.2 software-based results are summarized and compared to those resulting from user-defined selection. In Chapter 5 a concluding discussion is given. Finally, in Chapter 6 possible further use and development of the software is outlined.

Kurzfassung

MikroPET (Positron Emission Tomography) ist ein Abbildungsverfahren für Kleintiere. Es stellt ein verkleinertes Äquivalent zu der in der Humanmedizin gebräuchlichen PET dar. Der größte Nachteil von PET ist, dass es keine genauen anatomischen Informationen liefert, sondern nur die Verteilung der radioaktiv markierten Moleküle und keine Körperstrukturen darstellen kann.

Um eine quantitative Analyse der Verteilung eines Radiopharmakons zu ermöglichen, müssen ROIs (Regions of Interest) ausgewählt werden. Indem man die in den ROIs gemessene Aktivität über einen größeren Zeitraum beobachtet, kann man die zeitabhängige Verteilung des Tracers errechnen. Normalerweise müssen die ROIs händisch markiert werden. Unglücklicherweise hängt diese Art, Regionen zu markieren, stark von der Person ab, die diese Regionen einzeichnet. Zudem ist das Auswählen der ROI manchmal sehr zeitaufwändig.

Die Software, die im Zuge dieser Diplomarbeit entwickelt wurde, vereinfacht und objektiviert das Definieren von ROIs in PET-Bildern von Mäusen. Zuerst muss der Benutzer ein oder mehrere gut sichtbare(s) Organ(e) auswählen. Auf dieser Grundlage kann die Software die Position der übrigen Organe errechnen. Dies wird durch einen Abgleich des PET-Bildes mit einem digitalen Modell einer “Durchschnittsmaus”, welches von MikroMRT (Magnetic Resonance Tomography) Bildern abgeleitet wurde, gewährleistet. Dazu werden die Organe des Mausphantoms der Größe nach angepasst und rotiert, bis sie bestmöglich mit dem PET-Bild überlappen. Daraus kann die Position der übrigen Organe errechnet werden.

Das Dokument beginnt mit einer kurzen Einleitung über die Zielsetzung der Diplomarbeit. Kapitel 2 bietet einen kurzen Überblick über PET und einen kurzen Auszug über die Herstellung von Tracern. In Kapitel 3 wird das Programm im Detail vorgestellt. In Abschnitt 4.1 werden das MikroPET-System und Messvorkehrungen besprochen. Die automatisiert erzielten Messergebnisse werden dann in Abschnitt 4.2 präsentiert und mit jenen verglichen, die aus einer benutzerdefinierten Auswertung folgen. Anschließend werden die Messergebnisse im Kapitel 5 noch näher analysiert. Schlussendlich wird in Kapitel 6 die weitere Verwendung und Weiterentwicklung des Programms diskutiert.

Contents

1 Introduction 7
1.1 Problem Formulation . . . 7
1.2 Aim . . . 7

2 General Survey of PET 9
2.1 Principle . . . 9
2.1.1 PET in Human Medicine . . . 10
2.1.2 Small Animal PET . . . 10
2.2 Radioactive Tracer . . . 11
2.2.1 Production of Radionuclide . . . 12
2.2.2 Synthesis of Radiopharmacon . . . 12
2.3 Measurement Technique . . . 13
2.3.1 Detectors . . . 13
2.3.2 Coincidence Detection . . . 13
2.4 Correction Methods . . . 15
2.4.1 Normalization . . . 15
2.4.2 Decay Correction . . . 15
2.4.3 Isotope Branching Fraction . . . 16
2.4.4 Attenuation Correction . . . 16
2.4.5 Dead-time correction . . . 17
2.4.6 Scatter Correction . . . 18
2.5 Reconstruction . . . 18
2.5.1 Filtered Back Projection . . . 18
2.5.2 Numerical Methods . . . 19

3 Software Details 21
3.1 Data Types Used . . . 22
3.1.1 PET Data . . . 22
3.1.2 Phantom . . . 24
3.2 Fitting Process . . . 25
3.2.1 Defining Organs . . . 26
3.2.2 Extract Phantom Organs . . . 27
3.2.3 Resizing Organs . . . 29
3.2.4 Rescaling Distancies . . . 31
3.2.5 Determine Angles . . . 32
3.2.6 Affiliate Organs . . . 35
3.3 Output . . . 37
3.3.1 Saving Organ Maps . . . 38
3.3.2 Generating TAC Files . . . 38
3.4 Image Processing . . . 42
3.4.1 2D Slices . . . 43
3.4.2 3D Projection . . . 45

4 Software Evaluation 47
4.1 Materials and Methods 47
4.1.1 Positron Emission Tomograph Used 47
4.1.2 Evaluation Software 47
4.1.3 Animals Used 47
4.2 Results 48
4.2.1 Correlation 50
4.2.3 Indirect Correlation 53
4.2.2 Reproducibility 57

5 Measurement Discussion 59
5.1 Discussion on Correlation 59
5.2 Discussion on Indirect Correlation 59
5.3 Discussion on Reproducibility 59

6 Conclusion & Outlook 61

A References 63

B Abbreviations 65

C Figures 67

D Tables 69

Acknowledgements 71

1 Introduction

1.1 Problem Formulation

Positron Emission Tomography (PET) (introduced in detail in Chapter 2 “General Survey of PET”) has established as an imaging method next to Computer Tomography (CT) and Magnetic Resonance Tomography (MRT). PET is commonly used to diagnose tumors, metastases or metabolic dysfunctions. The fabrication of the used tracers is illustrated in Section 2.2.

The major drawback of PET is, that the measured data usually does not provide anatomical information. In order to gain anatomical information CT or MRT are often used. Therefore CT or MRT images are often merged with the PET data, providing the location of the applied dose.

CT has the advantage, that it is less expensive than MRT, but CT only provides useful anatomical information, if adjacent body regions have different attenuation coefficients. Another disadvantage is, that CT exposes animals to significant additional radiation dose, which can distort metabolism or time evolution of tumors.

In MRT one can even distinguish between regions, which have the same X-ray density (as for instance parts of the brain) and it does not affect metabolism, but instrumentation costs are very high.

If neither CT nor MRT are used to grant anatomical information, ROIs (Region of Interest) have to be selected by hand. This procedure is quite time consuming and subjective. Therefore a semiautomated software[1] was developed at the UCLA[1], which provides an alternative to CT and MRT. This computer program computes the remaining organs on the basis of the user defined ones. Afterwards it can calculate the TACs (Time Activity Curves), which show the time dependent distribution of the tracer, of each organ. The anatomical information for this software is provided by a digital mouse phantom[2].

1.2 Aim

The aim of this thesis was to fasten and to improve the semiautomated software, written by Adam Kesner et al. But instead of adapting the source code of this computer program, a new approach was taken, and so a completely new software was written. The new software’s algorithm is both simpler and faster as well as more stable. A detailed description of this computer program will be presented in “Software Details”.

---

[1] abbr., University of California, Los Angeles

2 General Survey of PET

2.1 Principle

Positron Emission Tomography (PET) is an imaging technique in nuclear medicine. Its principle is shown in Fig. 2.1. PET is based on imaging radio labeled substances within living organisms. Therefore these substances are labeled with radioactive isotopes with a β^+-decay (positron emission), such as ^{11}C, ^{18}F or ^{124}I. These proton rich nuclei decay, emitting positrons, which travel through matter, till they reach thermal energies and annihilate by recombining with electrons. Each annihilation process is followed by two back-to-back 511 keV gamma photons. Partially these photons are scattered and absorbed on the way to the detectors. A coincidence processing (see Subsection 2.3.2) unit separates real coincidences from scattered photons. This coincidences provide the information of the original location of the annihilation, because the source of radiation must be situated in a straight line between the two detectors (also known as Line of Response (LOR)).

PET uses two different recording modes. The 2D (two-dimensional) mode only allows coincidences along one detector ring (photons emitted in radial direction), whereas 3D (three-dimensional) mode allows coincidencies over all detector rings. The advantage of 2D mode is, that it reduces random events, but at the cost of a highly reduced sensitivity. The collected data is acquired in listmode format. Therein the time of each counting event and the detector cell, which absorbed the photon, is stored. The listmode data is saved on hard disk and is used for dynamic image reconstruction (see Section 2.5).

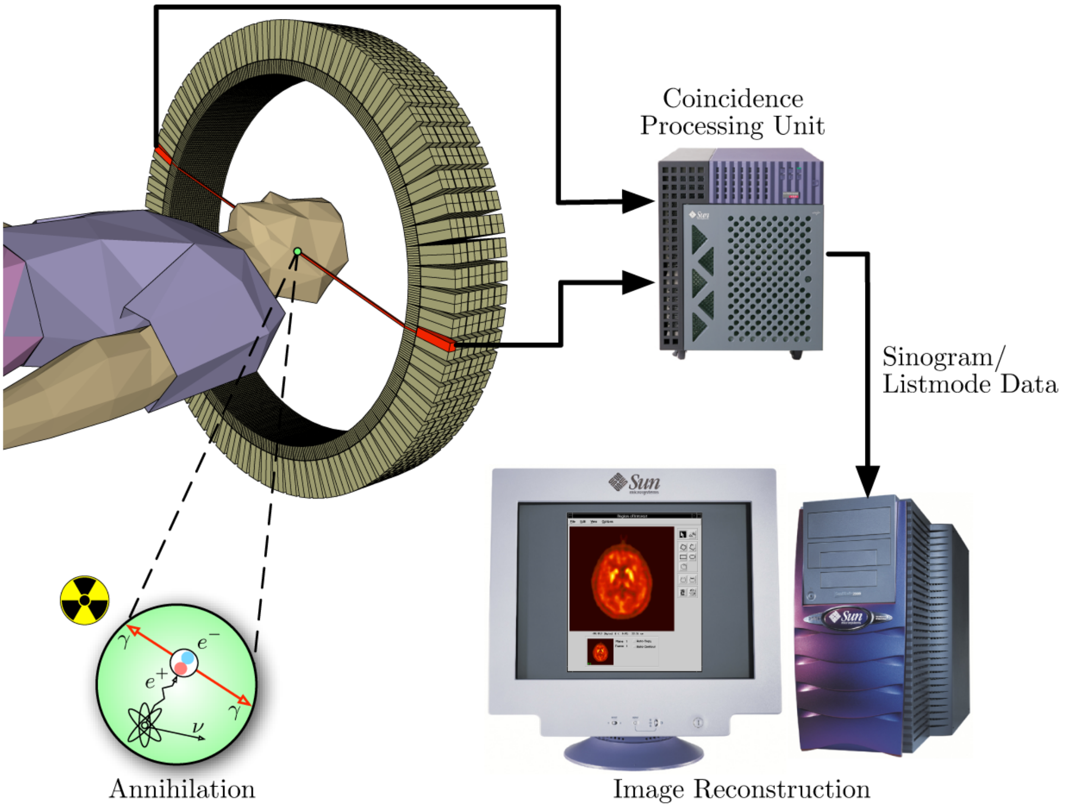


Figure 2.1 Scheme of Positron Emission Tomography[3]

In preclinical research PET is used in three main areas: for the development of new tracers for nuclear medicine, to follow treatment therapies and direct labeling of new promising drugs. The major drawback of PET is, that it “only” provides physiological information of organs, where the radio labeled molecules accumulate but no anatomical information. Therefore PET is often supported by Computer Tomography (CT), which allows to localize the measured activity more accurately.

2.1.1 PET in Human Medicine

In human medicine PET is among the most expensive imaging methods. Commonly a PET investigation takes about one to two hours. Because of the ephemerality of the used radionuclides, the radioactive isotopes have to be produced just in time and can only be transported on short distances. Therefore next to a PET-facility a cyclotron, which produces the radionulides, has to be established, hence associated with high initial and maintenance costs. A typical PET facility is shown in Fig. 2.2.

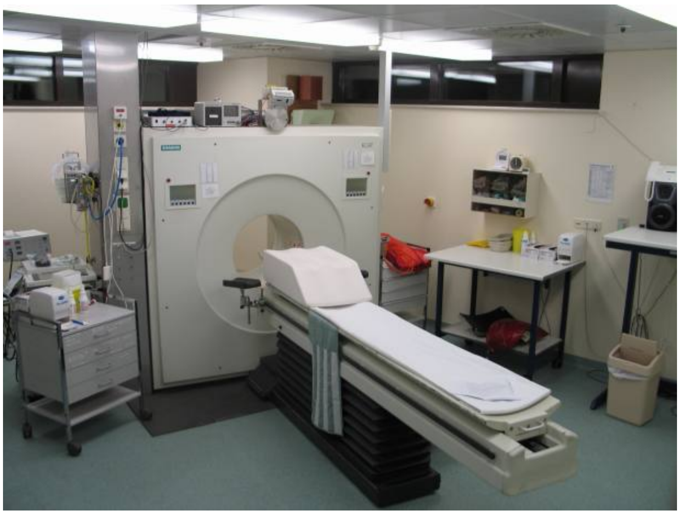


Figure 2.2 Human PET facility[4]

In modern medicine, PET is mainly used in oncology, neurology and cardiology. It helps to diagnose and locate tumors, metastases or metabolic dysfunctions. The radiopharmacon ^{18}F-Fluor-Deoxyglucose (FDG) accumulates in many malignant tumors (metabolic trapping) and helps to diagnose the course of cancer. FDG is also used for imaging the metabolic activity of the brain or for detection of minor perfused heart regions.

2.1.2 Small Animal PET

Usually laboratory animals like rodents (mice or rats) are used for preclinical studies. The positron emission tomographs (shown in Fig. 2.3) developed for small animals (often also referred to as microPET) normally use a smaller ring of detectors, because it improves sensitivity. In miscellaneous studies promising therapies or new drugs are tested. The biodistribution of radio labeled drugs can be recorded in real time, which allows to draw a conclusion on the drug’s effectivity. This method reduces the quantity of animals needed and is more cost efficient and faster than dissecting several animals’ organs at a given time.

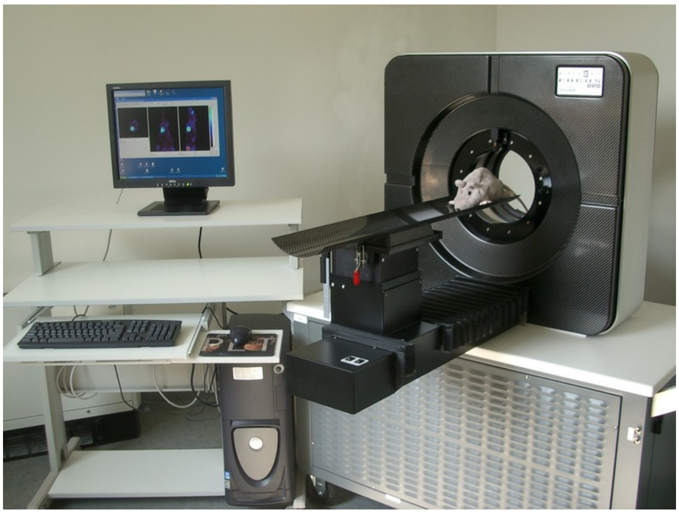


Figure 2.3 Small animal PET facility in ARC (Austrian Research Centers), Seibersdorf

2.2 Radioactive Tracer

There are many tracers, that serve different purposes. In Table 2.1 commonly used isotopes, their half life and some compounds carrying this isotope are shown. The chemical structure lends each radiopharmacon different metabolic effects. ^{18}F-Fluor-Deoxyglucose (FDG) for instance is used to highlight glucose metabolism, whereas the fluoride ion is enriched in bones. In living organisms the radio labeled molecules are exposed to decomposition. Metabolism cracks up compounds, freeing for example the radioactive fluor from FDG. This is called biological half life and leads to unwanted side effects, which have to be taken into account in the final evaluation. It may happen, that a ^{124}I-labeled molecule separates from ^{124}I, which enriches in the thyroid afterwards. At that point the track of these previous, radio labeled molecules is lost, but it looks, as if the molecule accumulates in the thyroid. If not taken into consideration, this may result in wrong conclusions.

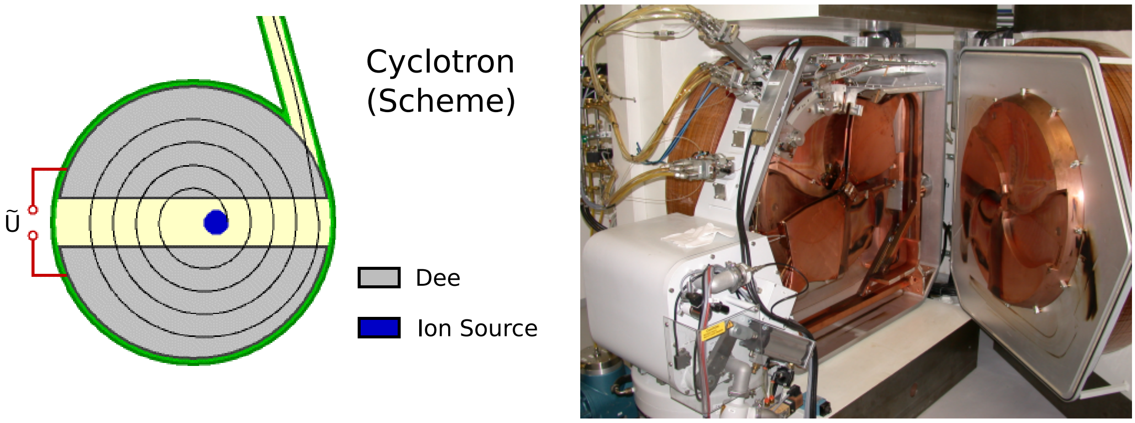
isotope	half life	compound	diagnostic usage
^{11}C	20.3 min	^{11}C-Choline ^{11}C-Pittsburgh compound B	prostate cancer Alzheimer's disease
^{13}N	9.97 min	^{13}N-L-Glutamate acid ^{13}N-Ammonia	amino acid metabolism myocardial perfusion
^{15}O	2.03 min	^{15}O-Water $^{15}O_2$	blood circulation oxygen metabolism
^{18}F	109.8 min	^{18}F-Fluor-Deoxyglucose ^{18}F-Fluoride ^{18}F-6-Fluoro-DOPA	glucose metabolism bone metabolism dopamine metabolism
^{124}I	13.27 h	^{124}I	thyroid function

Table 2.1 Examples of radioactive tracers[5]

2.2.1 Production of Radionuclide

There exist several kinds of production facilities for radionuclides. The most common ones are cyclotrons, reactors and generators[6]. In the following only the cyclotron will be discussed briefly.

A cyclotron is a particle accelerator. It accelerates electrically charged particles. Commonly protons (${}^{1}_{1}\mathrm{H}$), deuterons (${}^{2}_{1}\mathrm{H}$) and α-particles (${}^{4}_{2}\mathrm{He}$) are used. It consists of two D-shaped chambers, so called Dees. Between the two Dees there is a gap, wherein the charged particle is accelerated by an electric field. A magnetic field (Lorentz-force) keeps charged particles on a circular path. For each full rotation, the particle passes the electric field twice and therefore polarity of this field has to be reversed twice for each full rotation. The schema of a cyclotron is shown in Fig. 2.4.



(a) Scheme of a cyclotron[7]

(b) Picture of the cyclotron used at ARC, Seibersdorf

Figure 2.4 Illustration of a Cyclotron

Cyclotrons are used to produce ${}^{11}\mathrm{C}$, ${}^{13}\mathrm{N}$, ${}^{15}\mathrm{O}$, ${}^{18}\mathrm{F}$ and ${}^{124}\mathrm{I}$. Therefore the accelerated particles are targeted at nuclei like for instance ${}^{14}\mathrm{N}$, which grabs an accelerated proton, emits an α-particle and becomes ${}^{11}\mathrm{C}$.

2.2.2 Synthesis of Radiopharmacon

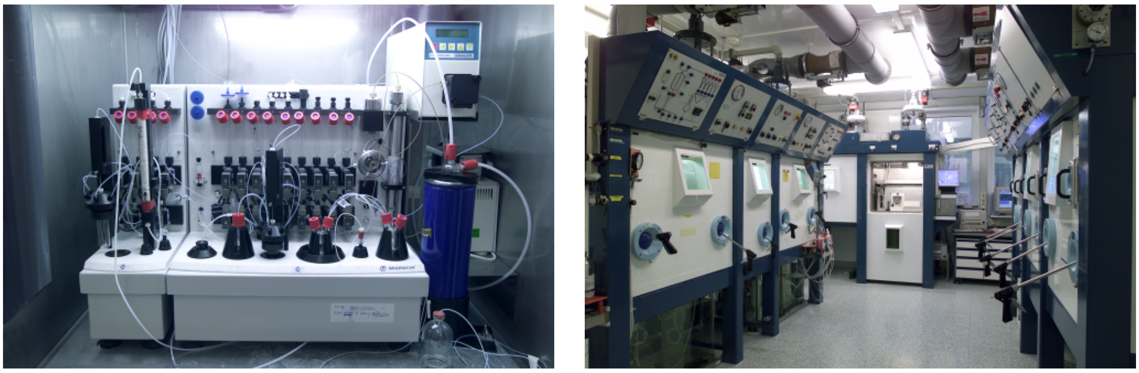


Figure 2.5 Pictures of the synthesis unit and hot laboratory facilities at ARC, Seibersdorf

For the production of ${}^{18}\mathrm{F}$-Fluor-Deoxyglucose commonly electrophile addition or nucleophile substitution is used. In electrophile addition a pi-bound is replaced by two covalent bounds (see Fig. 2.6).

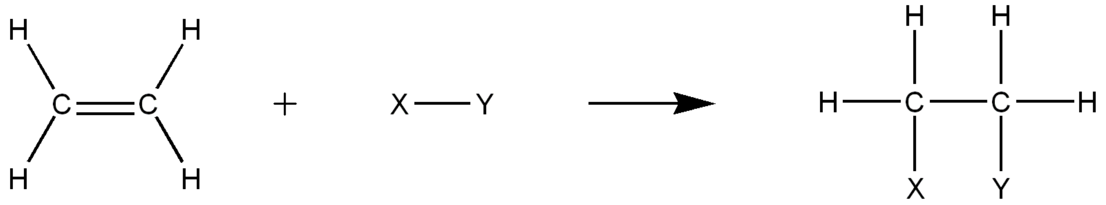


Figure 2.6 Electrophile addition[8]

In nucleophile substuition the group X (see Fig. 2.7) is separated from the compound. Now an electron rich nucleophile uses the vacant position to recombine with the compound.

It is important to keep in mind, that the synthesis of the radionuclide with a molecule may determine its biological half life.

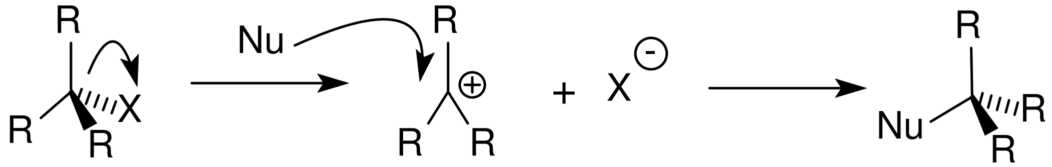


Figure 2.7 Nucleophile substitution[9]

2.3 Measurement Technique

2.3.1 Detectors

For the detection of γ-rays scintillating crystals coupled to a Photomultiplier Tube (PMT) are used. Scintillators are materials like NaI(Tl) (thallium doped sodium iodide), BaF_2 (barium fluoride), BGO (bismuth germanate) and LSO(Ce) (cerium doped lutetium oxyortho-silicate)[10]. They are excited by γ-rays or charged particles passing through it. Afterwards the resulting excitation energy is emitted by multiple photons with longer wavelength. These photons are usually in the UV range and their number is proportional to the energy of the absorbed particle. As Fig. 2.8 shows, every resulting light photon excites an electron in the photo cathode. These electrons are accelerated to a dynode, releasing several electrons for every electron. The resulting electrons are accelerated to the next dynode and so on, till the electrons finally reach the anode, resulting in a measurable current, which is proportional to the absorbed particle's energy.

Several scintillating crystals and photo multipliers are grouped to a detector block. These blocks now form several concentric detector rings (as shown in Figs. 2.1 and 2.9).

2.3.2 Coincidence Detection

There exist several ways the emitted photons can reach the detectors (as shown in Fig. 2.10). A "true" coincidence occurs, when two back-to-back photons, which both

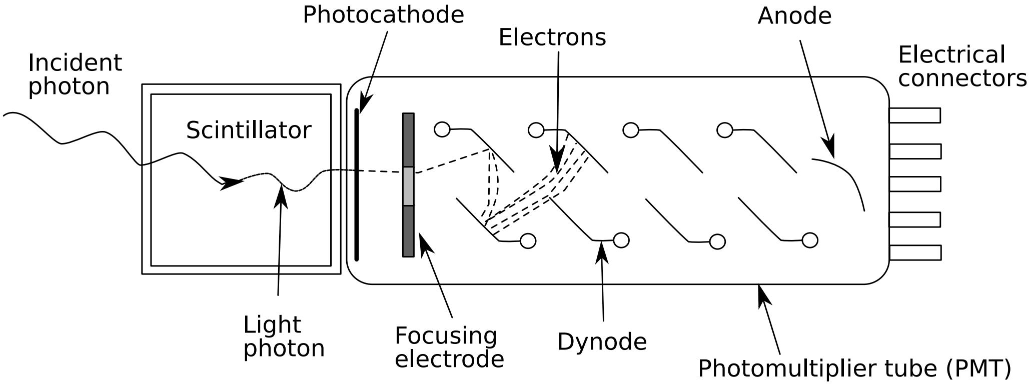


Figure 2.8 Scintillating crystal coupled to a PMT[11]

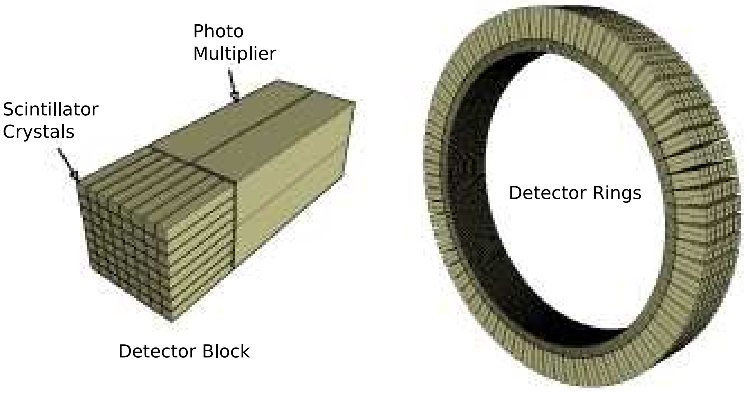
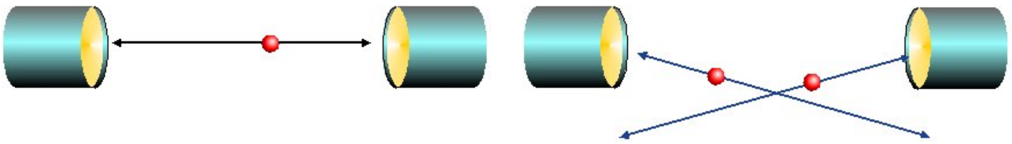


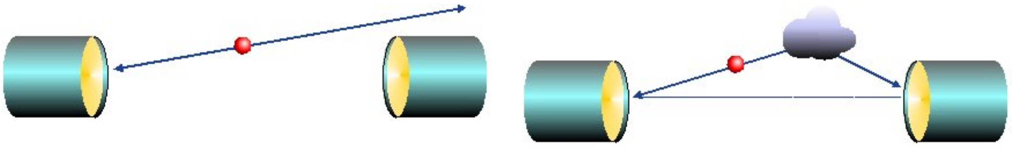
Figure 2.9 PET detector system[12]

emerge from an electron-positron-annihilation, reach the detectors without being scattered. “Random” events take place, if two photons, having two different origins, reach the detectors incidentally within the same time window. If only one photon reaches the detector, within a time frame, it is called “single”. “Scatter” coincidences appear, if both photons descend from a common origin, but one or both of them are scattered before they reach the detector.

As it is well known, image reconstruction in CT is based on line informations. In CT and SPECT (Single Photon Emission Computed Tomography) these lines are defined by collimators, whereas in PET these lines are found by coincidence counting. Therefore PET is only interested in “true” coincidences. These coincidences events provide the information of the original location of the annihilation, because the source of radiation must be situated along a straight line between the two detectors (also known as Line of Response (LOR)). “Singles”, “randoms” and “scatter” events are just unwanted side effects that should be avoided. And so detected “singles” are discarded. “Randoms” and “scatter” events are similar to “trues” and can easily be mistaken for “trues”. In order to avoid “randoms”, the coincidence time frame can be reduced. “Scattered” events are reduced by limiting the energy window for coincidence detection (Compton effect).



“Trues” “Randoms”



“Singles” “Scatter”

Figure 2.10 Different types of coincidences[13, 14, 15, 16]

2.4 Correction Methods

There are some physical effects, that result in a discrepancy between the measured and the real radioactive activity. These effects have to be numerically corrected before image reconstruction. In the following the most important correction steps will be presented.

2.4.1 Normalization

The sensitivity of LOR is influenced by the angle between the LOR and the axis of the detector ring as well as by the detectors’ sensitivities. Each detector block consists of several unique scintillator crystals and slightly different PMTs. This causes different response of the detectors, which can also change in time. Additionally geometric effects[17] have to be taken into account (as shown in Fig. 2.11). Coincidences of the red source of radiation can only be detected by a few detector blocks, whereas coincidences of the green source can be monitored by all detectors resulting in a disproportional high detection of the green source of radiation. The process of conditioning the sensitivities of all LORs is called normalization. For each LOR one Normalisation Coefficient (NC) is calculated. These NCs are used to correct for the sensitivities of all LORs. Fig. 2.12 illustrates normalization correction with the help of three sinograms. A sinogram consists of several rows. Each row represents the number of measured coincidences (the more coincidences the brighter) at every angle of acquisition (eg. row $1 \mathrel{\widehat{=}} 1°$, row $2 \mathrel{\widehat{=}} 2°$, ...). In Fig. 2.12a the sinogram of a homogeneous cylinder filled with a ^{18}F-water solution in the central field of view without normalization is shown. A correction mask (shown in Fig. 2.12b) equalizes the sensitivities of the LORs by multiplying the unnormalized sinogram with the correction mask, which leads to the normalized sinogram, shown in Fig. 2.12c.

2.4.2 Decay Correction

During measurement the radioactive isotopes decay. This results in a decrease of measured activity. But in PET mainly the distribution of the radiopharmacon is important, and therefore the measured activity is decay corrected, which keeps the measured dose constant. This approach enables the comparison of absorbed doses in organs.

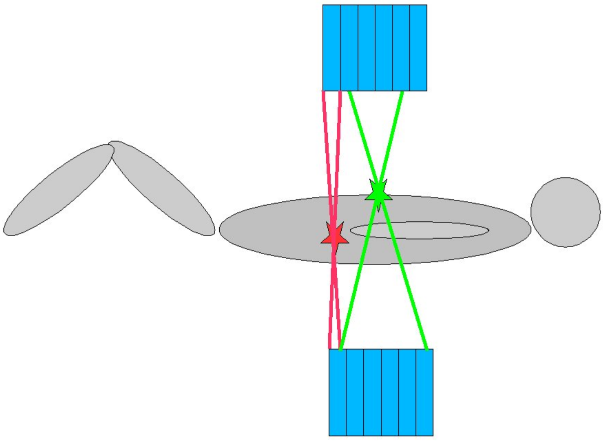
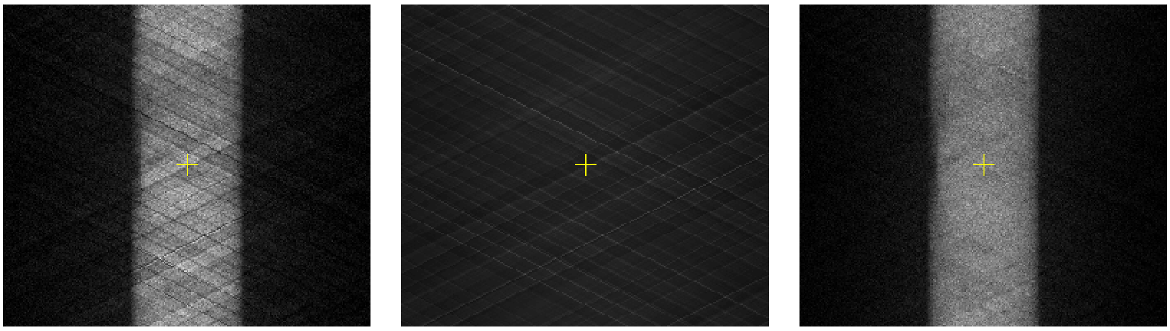


Figure 2.11 Different sensitivities due to geometric effects in 3D mode[18]



(a) Unnormalized (b) Correction Mask (c) Normalized

Figure 2.12 Illustrating the effect of normalization

2.4.3 Isotope Branching Fraction

Isotopes may have different decay channels, but in PET only β^+-decays (positron-decays) are measured by detecting coincidences. In order to compute the actual dose (including all other decay channels), the measured activity has to be divided by the probability of the nuclide's β^+-decay.

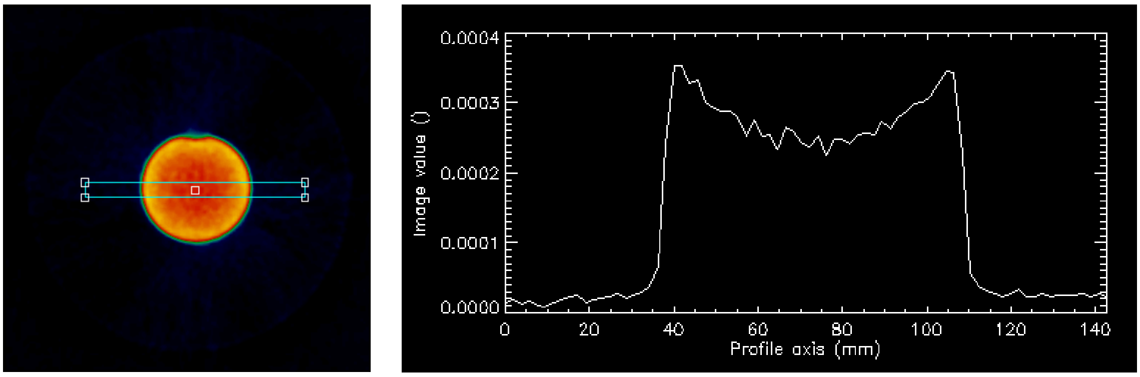
2.4.4 Attenuation Correction

The photons created during the positron-electron-annihilation are subject to attenuation processes such as scattering or absorption. The level of attenuation varies, because the photons have to travel along different distances through the imaged object, until they reach the detectors. If the origin of the photons lies deeper within the imaged object, fewer photons will reach the detectors (as shown in Fig. 2.13). In small animal PET this effect is considerably smaller than in human PET, but for quantitative analysis these effects cannot be neglected. There are several attenuation correction methods[19]:

- **Scaling method:** This very simple method assumes, that the imaged object is a homogeneous cylinder filled with water. By calculating the estimated attenuation the measured data can be adapted to this values.

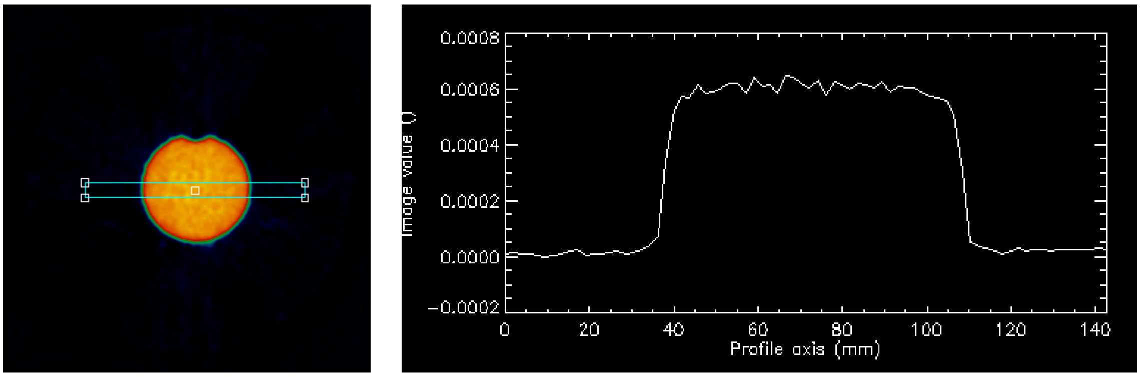
- **PET-transmission method:** The PET itself is used as a CT. A radiation source (eg. ^{68}Ge or ^{57}Co) is rotated around the imaged object. Thereby the attenuation can be directly measured.

- **CT-transmission method:** in principle just the same as PET-transmission, with the advantage of a higher resolution.



activity tomogram of a cylinder with no attenuation correction

cross section values (without correction)



attenuation corrected activity tomogram of a cylinder

cross section values (corrected)

Figure 2.13 Attenuation correction for cylinder filled with 500 ml ^{18}F-water solution

2.4.5 Dead-time correction

Every detection system has a certain dead time. This is the time span within two independent detection events cannot be resolved. If two events occur within the dead time, it leads to a pulse pileup in a PMT[6]. Too big pulse amplitudes may exceed the defined energy frame and may be discarded. These losses are called dead time losses. They become significant at higher count rates. In order to reduce these losses, detectors with shorter dead time can be used. Additionally the percentage of overloaded detectors can be estimated to extrapolate the actually detected counts.

2.4.6 Scatter Correction

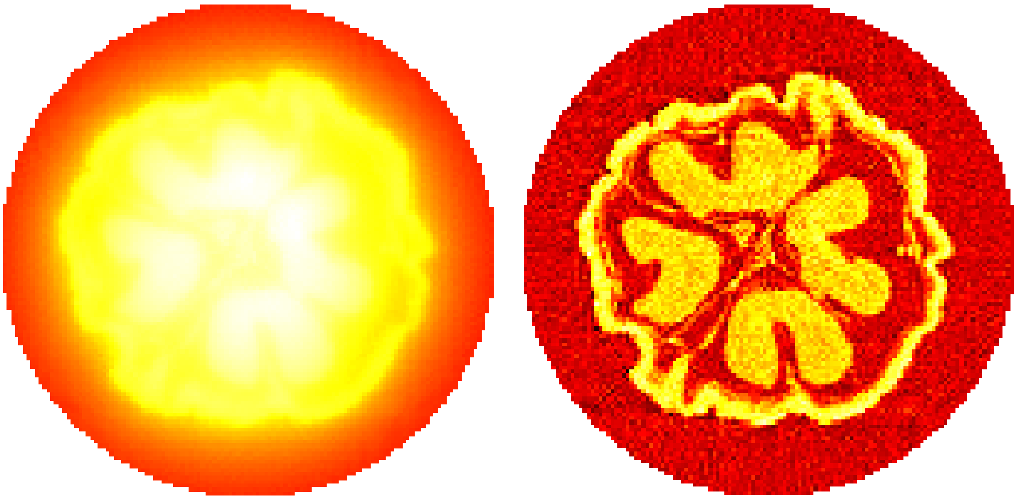
Scatter correction tries to minimize the scattering events caused by Compton scattering. There exist several different schemes and techniques to calculate scattered events. The methods range from the Gaussian fit technique over Monte Carlo simulations to single scatter simulation algorithms[17].

2.5 Reconstruction

In 1917 Radon[20] formed the mathematical bases for reconstructing objects from measured projection data (sinogram). There exists a wide range of different reconstruction methods[21]. A rather fast reconstruction method is the Filtered Back Projection (FBP). Iterative numerical methods are superior to filtered back projection, but also more time consuming. In the following the principles of these reconstruction methods will be shown very briefly.

2.5.1 Filtered Back Projection

FBP is by far the best known reconstruction method. An object, measured through several angles, can be reconstructed by back projecting the measured projections to the image plane. This approach is similar to mathematical shadow casting. But if the measured projections are directly projected into the image plane, the resulting image will be blurry (see Fig. 2.14a). In order to eliminate this smearing effects the resulting object has to be deconvoluted. As a replacement for deconvolution high-pass filters are used. Due to convolution of the projections with an appropriate filter function the reconstructed image is sharpened and negative values at edges may occur (Fig. 2.14b).



(a) Unfiltered Back Projection (BP)

(b) FBP using a bandlimiting-filter

Figure 2.14 Illustrating the use of filters in FBP (mircoCT image of a nut[22])

2.5.2 Numerical Methods

Numerical reconstruction methods are based on sets of linear equations. The projected values along several angles (sinogram) are known. By an iterative process the unknown matrice's elements have to be calculated. The estimated matrice's elements are adjusted, till the projections through the reconstructed matrix equal the measured projection values.

A very simple iterative method, called Algebraic Reconstruction Technique (ART)[23], is illustrated in Fig. 2.15. During this iterative process the matrice's elements (A', B', C', D') are adjusted to match the measured projection values. This is done for every projection angle. After having compared all angles, the first iteration has finished. This iteration process has to be done several times till the matrice's elements converge.

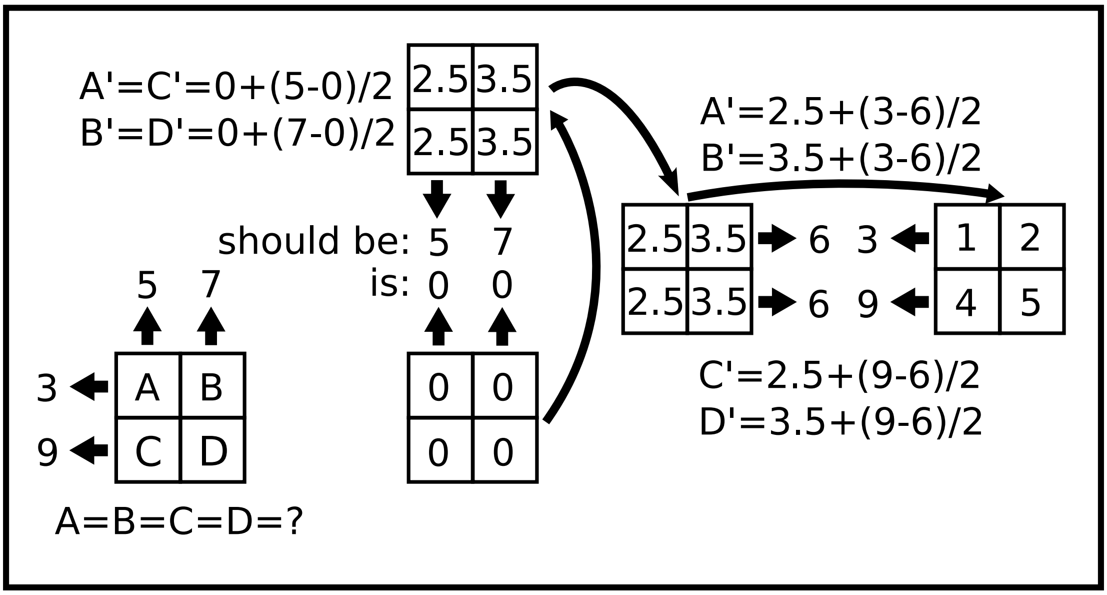


Figure 2.15 Illustration of the ART algorithm

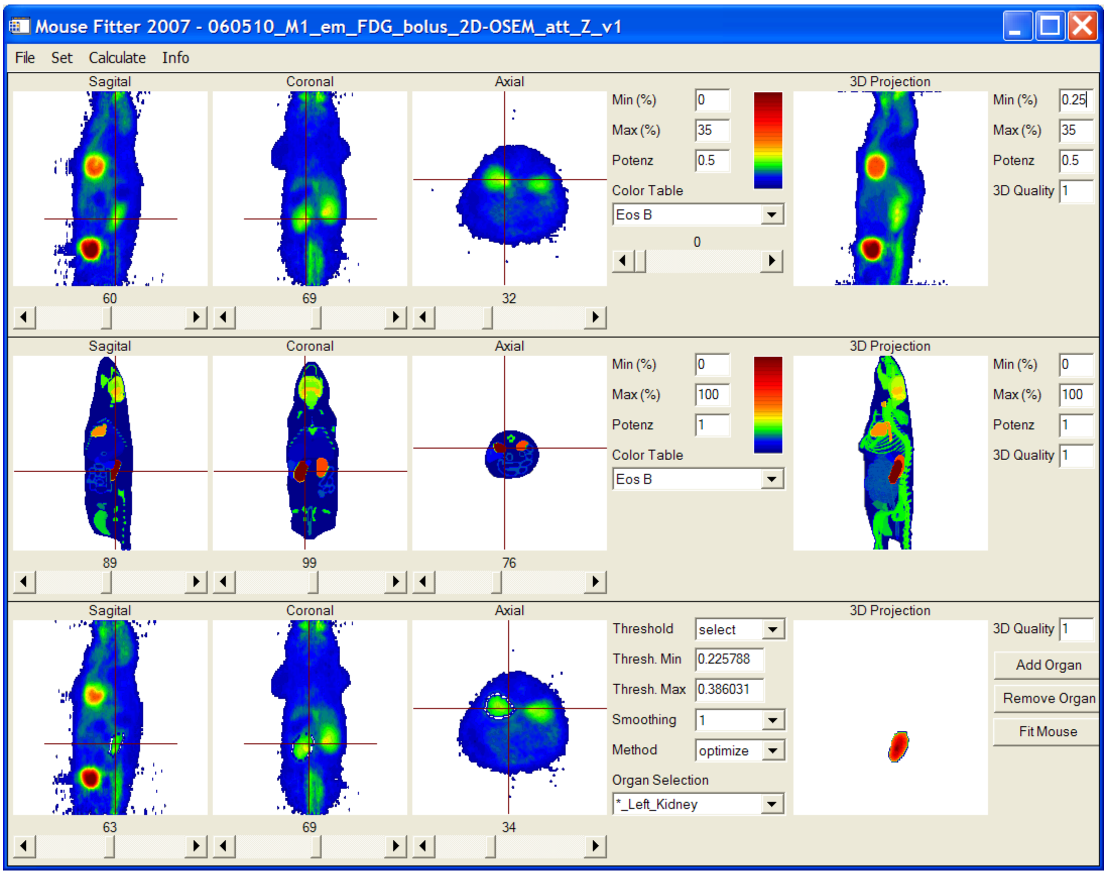
Though this method is very descriptive it converges fairly slow. Other methods like Expectation Maximization[24] (EM) are more efficient. Ordered Subset Expectation Maximization[25] (OSEM) provides the same results as EM and converges more quickly. Two other commonly used variants of EM are: the Maximum Likelihood Expectation Maximization[24] (MLEM) and the Maximum a Posteriori[23] (MAP) algorithm.

3 Software Details

The software is completely written in IDL (Interactive Data Language), an object oriented, vectorized programming language, whose syntax is related to Fortran and C.

In Fig. 3.1 the GUI (Graphical User Interface) of the computer program, which has been called “Mouse Fitter 2007”, is shown. It consists of three major, vertically aligned blocks. At the top the image of the loaded PET data, in the middle the phantom file and at the bottom the selected ROI are displayed. Each block is divided into a set of two-dimensional (2D) images, which are made up of coronal, sagital and axial planes and a three-dimensional (3D) projection.

Organ selection takes place by choosing the desired region in the “Organ Selection” combo box and selecting the organ’s maximum with a left mouse and the minimum with a right mouse click. Pressing the button “Add Organ” now defines the organ mask.



Mouse Fitter 2007 - 060510_M1_em_FDG_bolus_2D-OSEM_att_Z_v1

Min (%) 0
Max (%) 35
Potenz 0.5
Color Table Eos B
0
60
69
32
Min (%) 0.25
Max (%) 35
Potenz 0.5
3D Quality 1

Min (%) 0
Max (%) 100
Potenz 1
Color Table Eos B
89
99
76
Min (%) 0
Max (%) 100
Potenz 1
3D Quality 1

Threshold select
Thresh. Min 0.225788
Thresh. Max 0.386031
Smoothing 1
Method optimize
Organ Selection *_Left_Kidney
63
69
34
3D Quality 1

Figure 3.1 Program’s GUI

After having defined all desired ROIs, the software calculates the most likely positions of the remaining organs. In combo box “Method” two reconstruction methods can be selected. The method “optimize” determines the angle between phantom and PET image, that guarantees the best possible overlap, whereas method “customize” lets the user choose the right angle.

In order to simplify organ selection the perceptibility can be enhanced by changing the color table or by adjusting the minimum and maximum values of the image to be displayed. By clicking into a 3D projection, while keeping the left mouse button pressed, and dragging the mouse, the 3D projection can freely be rotated around the z-axis (axial).

3.1 Data Types Used

In the following PET and phantom file formats will be discussed and it is shown how they are imported into the computer program.

3.1.1 PET Data

The PET data are stored in concorde file format. This is a two file format and consists of a raw data file (img-file) and a header file (img.hdr-file) in ASCII (American Standard Code for Information Interchange) format. The header file can be edited in a normal text editor, which allows to easily change settings. In the header file lines are commented with “`#`”. Each setting is made up of `setting_description` and `setting_value` using the following format: “`setting_description setting_value`” (eg. “`number_of_dimensions 3`”). An extract of a typical header file is shown below.

```
...
# Number of dimensions in data set (integer)
#     Order from fastest to slowest is XYZW
#
number_of_dimensions 3
#
# Size of X dimension in data set (integer)
#
x_dimension 128
#
# Size of Y dimension in data set (integer)
#
y_dimension 128
#
# Size of Z dimension in data set (integer)
#
z_dimension 95
...
```

The raw data file consists of one to several time frames. Each frame is made up of a three dimensional array of 32 bit floats. Before the raw data file can be imported, the header file has to be read in, to be able to import the raw data correctly. The most important passages of function `readHdrFile` are shown below. In a `WHILE`-loop each line is imported with `READF`. If the line’s first character is not equal to “`#`”, the first part of the line’s contents is compared to the `setting_description` (as for instance “`x_dimension`”) and its `setting_value` is saved. Not all settings in the header file are important for the software. The necessary settings are: the PET file’s dimensions (`x_dimension`, `y_dimension`, `z_dimension`), the number of frames (`total_frames`), the voxel sizes (`pixel_size`), the injected dose (`dose`, `dose_units`), the mouse weight (`subject_weight`, `subject_weight_units`) and additional calibration factors (`calibration_factor`, `calibration_units`, `isotope_branching_fraction`).

After the `WHILE`-loop the imported values are adapted to computer program units, which are mCi (millicurie), g (gram) and cc (cubic centimeter) (lines 25 and 26). This prevents a mix up of different units within the software.

```
1   FUNCTION readHdrFile, path, fileName
2    ...
3    WHILE (~ EOF(inLun)) AND (endOfHeaderCounter NE 0) DO BEGIN
4     READF, inLun, inFileData
5     IF (STRMID(inFileData,0,1) NE "#") THEN BEGIN
6      inFileDataArray = STR_SEP(inFileData, " ")
7      CASE inFileDataArray(0) OF
8       "x_dimension": dimensions(0) = inFileDataArray(1)
9       "z_dimension": dimensions(1) = inFileDataArray(1)
10      "total_frames": dimensions(2) = inFileDataArray(1)
11      "dose": cInjectedDose = inFileDataArray(1)
12      "dose_units": ...
13      "subject_weight": cMouseWeight = inFileDataArray(1)
14      "subject_weight_units": ...
15      "end_of_header": endOfHeaderCounter = endOfHeaderCounter - 1
16      "pixel_size": magnificationXY = inFileDataArray(1)
17      "calibration_factor": ...
18      "calibration_units": ...
19      "isotope_branching_fraction": ...
20      ELSE:
21     ENDCASE
22    END
23   END
24   ...
25   calibration.injectedDose = cInjectedDose * units(0)
26   calibration.mouseWeight = cMouseWeight * units(1)
27   ...
28  END
```

After having read out the dimensions from the header file, the `petFile`-struct is defined, because arrays cannot be rescaled in IDL. This struct stores all imported PET data (`petFile.cache`) and other important variables.

```
petFile =  {fileName:’’, path:’’,$
 magnification:FLTARR(2), dim:INTARR(3),$
 cache:FLTARR(dimensions(0), dimensions(0),$
  dimensions(1), dimensions(2)),$
 isCached:INTARR(dimensions(2)), type:’img’,$
 frameDuration:FLTARR(dimensions(2)),$
 scaleFactor:FLTARR(dimensions(2)),$
 isotopeBranchingFraction:1.}
```

Each frame has a different frame duration (`frame_duration`) and a different scaling factor (`scale_factor`). Both have to be imported from the header file and are stored in the previous defined `petFile` struct. This is done in a `WHILE` loop, which aborts if the end of the header file is encountered or the frame counter, called `endOfHeaderCounter`, reaches the absolute number of time frames (`dimensions(2)`).

```
WHILE (~ EOF(inLun)) AND (endOfHeaderCounter LT dimensions(2)) $
DO BEGIN
 READF, inLun, inFileData
 IF (STRMID(inFileData,0,1) NE "#") THEN BEGIN
  inFileDataArray = STR_SEP(inFileData, " ")
  CASE inFileDataArray(0) OF
   "frame_duration": petFile.frameDuration(endOfHeaderCounter) = $
    inFileDataArray(1)
   "scale_factor": petFile.scaleFactor(endOfHeaderCounter) = $
    inFileDataArray(1)
   "end_of_header": endOfHeaderCounter = endOfHeaderCounter + 1
   ELSE:
  ENDCASE
 END
END
```

Now procedure `readImgFile` reads in the current frame of the img-file (lines 6-9). The variable `offset` equals the offset in bytes of the opened file, because not the whole img-file is read but just the current frame. The raw data of each frame has to be multiplied with the frame's `scale_factor` and the global `isotope_branching_fraction` (for more information on the isotope branching fraction see Subsection 2.4.3).

```
1   PRO readImgFile, frame
2    ...
3    inFileData = FLTARR(petFile.dim(0),petFile.dim(0),petFile.dim(1))
4    offset = SIZE(inFileData, /N_ELEMENTS)*frame*4ULL
5    ...
6    OPENR, inLun, inFile, /GET_LUN
7    POINT_LUN, inLun, offset
8    READU, inLun, inFileData
9    CLOSE, inLun, /ALL
10   ...
11   FOR i=0,2 DO IF calibration.flipPetFile(i) THEN
12    inFileData = REVERSE(inFileData, i+1, /OVERWRITE)
13   petFile.cache(*,*,*, frame) =
14    inFileData(*,*,*) * petFile.scaleFactor(frame) /
15    petFile.isotopeBranchingFraction
16   petFile.isCached(frame) = 1
17  END
```

3.1.2 Phantom

The phantom file is in raw data format. It consists of a three dimensional array of 16 bit integers. The function `readBinFile` shows, how the data is read out. In equivalence to struct `petFile`, a new struct `phantom` file is defined, which stores all phantom data.

```
FUNCTION readBinFile
 ...
 dimensions = INTARR(3)
 dimensions = [190, 190, 1]
 ...
 phantom = {fileName:'BFnormalmap_act_av', path:'.\Phantom\',$
  magnification:FLTARR(2), dim:INTARR(3),$
  cache:BYTARR(dimensions(0), dimensions(0),$
   dimensions(1), dimensions(2)),$
  isCached:INTARR(dimensions(2)), type:'bin'}
 phantom.dim = dimensions
 ...
 inFileData = BYTARR(phantom.dim(0), phantom.dim(0), phantom.dim(1))
 OPENR, inLun, inFile, /GET_LUN, /COMPRESS
 READU, inLun, inFileData
 CLOSE, inLun, /ALL
 ...
 phantom.cache(*,*,*,0) = inFileData
 phantom.isCached(0) = 1
 RETURN, 0
END
```

3.2 Fitting Process

By the help of Fig. 3.2, the main principle of the software is demonstrated. First of all the user has to select a certain organ by clicking at the minimum and the maximum intensity value of the organ region. The difference between these values is taken as threshold. The computer program now determines, if the value of the current pixel compared to the maximum value, lies within the selected threshold (Subsection 3.2.1). The phantom data consists of discrete values (eg. heart = 28, Bladder = 12, ...), providing an anatomical map. With the phantom data the software creates a mask of each organ (Subsection 3.2.2).

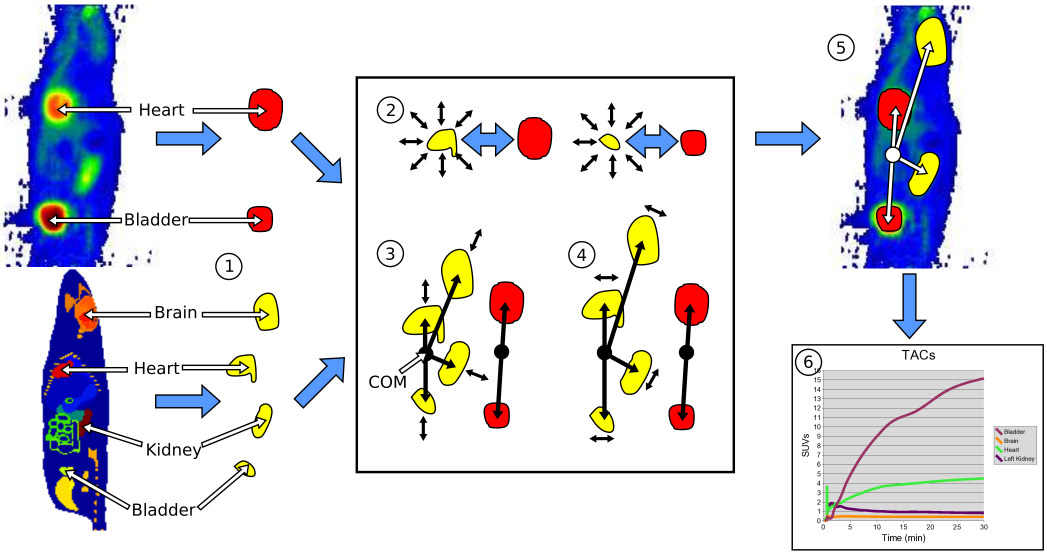


Figure 3.2 Functional principle of the fitting process

Then calculation begins, and the phantom and PET masks are compared to each other. The organ masks defined by the phantom enlarge or shrink, till they have reached the same volumes as the selected PET organ (2) (Subsection 3.2.3). Then the distances between the center of mass (COM) and the phantom masks are adapted to the defined PET organs (3) (Subsection 3.2.4). The remaining organs are rescaled by the mean relative change of distance. Because the orientation of the mouse slightly differs, the organs are rotated around the x-,y- and z-axis (4) (Subsection 3.2.5), until the phantom and PET organ masks meet their best possible overlap.

When the calculation is finished, the masks (the originally selected and the calculated ones) are affiliated with the PET data as shown in (5) (Subsection 3.2.6). By knowledge of the most likely position of each organ the accumulated dose in each organ can be calculated.

The dynamic PET data consists of several time frames. Every frame represents the measured activity in predefined time intervals. With the help of these frames, it is possible to gain knowledge about the biodistribution of the injected radioactive tracer. Afterwards the distribution can be visualized in TACs (6) (Time Activity Curves) (Subsection 3.3.2).

3.2.1 Defining Organs

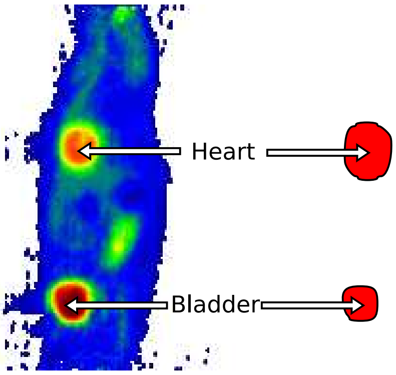


Figure 3.3 Defining Organs

In order to select an organ, the organ's maximum has to be selected by a left mouse button click. Afterwards the organ's minimum has to be selected by a right mouse button click. In procedure `selectPetOrgan` the difference between these two values is called `threshold`. The `referenceValue` represents the value of the organ's defined maximum. `pos` represents the current position. `organ(organSelected).mask(pos)` is the selected organ's mask (shown in Fig. 3.3). Voxel values are set to zero, if they do not belong to this organ, and set to one, if they are part of the organ.

Procedure `selectPetOrgan` is called recursively. If the current voxel (`pos`) of the organ mask is already part of the organ, do nothing and exit. Otherwise this voxel is set to be part of the organ and its neighbours[2] are tested. If the adjacent position is valid[3] and "|`referenceValue` – left/right_neighbour| < `threshold`" then `selectPetOrgan` is called recursively with this new adjacent position (eg. `scaledData3D(pos(0)+1,pos(1),pos(2))`).

---

[2] Because of simplicity, only adjacent voxels in x-direction are shown below. For the y and z-direction it is nearly the same.

[3] i.e. if all the coordinates lie within the dimensions of the PET file.

```
PRO selectPetOrgan, referenceValue, pos, scaledData3D, threshold
 ...
 organ(organSelected).mask(pos(0), pos(1), pos(2)) = 1
 IF (pos(0) LT (petFile.dim(0)-1)) THEN BEGIN
  IF ABS(referenceValue - scaledData3D(pos(0)+1, pos(1), pos(2))) $
  LT threshold THEN BEGIN
   selectPetOrgan,referenceValue,pos+[1,0,0],scaledData3D,threshold
  END
 ENDIF
 IF (pos(0) GT 0) THEN BEGIN
  IF ABS(referenceValue - scaledData3D(pos(0)-1, pos(1), pos(2))) $
  LT threshold THEN BEGIN
   selectPetOrgan,referenceValue,pos-[1,0,0],scaledData3D,threshold
  END
 ENDIF
 ...
END
```

3.2.2 Extract Phantom Organs

The phantom data consists of discrete values, which are shown in Table 3.1. IDL provides a specially optimized function `WHERE` (as shown below), which allows to find regions, that have a certain value (shown in Fig. 3.4). Therefore organ allocation can quickly be done. It is useful to additionally define groups of organs. These defined organ groups are shown in Table 3.2. They are not part of the phantom file, they are just arbitrarily set up to ease organ selection and consist of several organs provided by the phantom file.

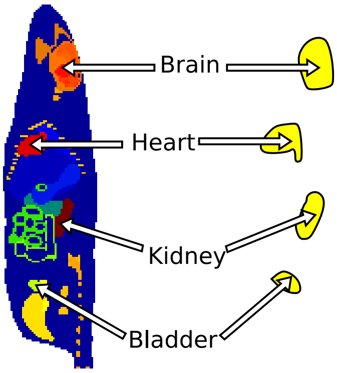


Figure 3.4 Extract Phantom Organs

With a `FOR`-loop over all defined organs, wherein `organ(i).value` accord to the values shown in Tables 3.1 and 3.2, the location of each organ (`organLocationPhantom`) is computed using function `WHERE`. If an organ group is selected, several organs are chosen (lines 9-20). The function's output, `organLocationPhantom`, is a list containing every voxel the selected organ consists of. "`phantomMask(organLocationPhantom) = 1`" sets every organ voxel to one, the rest remains zero. In order to save time and memory each `phantomMask` is cut to a minimal cuboid. For not losing track of the overall picture, the vector to the organ's COM (Center of Mass) has to be stored in `phantomMaskLocation`. Therefore the built-in function `ARRAY_INDICES` is used, which returns the x, y and z-coordinate of the organ voxels. Then the minimum and maximum x, y and z-coordinates are calculated and the organ mask is cut and stored in `*ptrPhantomMask`.

organname	value
Body	1
Liver	2
Lungs	3
Stomach	5
Pancreas	7
Spleen	9
Small Intestine	10
Large Intestine	11
Bladder	12
Spermatic Duct	13
Testes	14
Ribs & Legs	15
Spine	16

organname	value
Skull	17
Ventrical Space	18
Neocortical White	19
Neocortical Gray	20
Cerebellum White	21
Cerebellum Gray	22
Thalamus	24
Hippocampus	25
Thyroid	26
Heart	28
Right Kidney	35
Left Kidney	36

Table 3.1 Some organs and values defined by phantom file

organgroup	values included	value
Everything	0-36	100
Kidneys	35,36	101
Brain	18-25	102
Bones	15-17	103

Table 3.2 Additionally defined organgroups

```
1   ptrPhantomMask = PTRARR( N_ELEMENTS(organ) )
2   ...
3   phantomMaskLocation = FLTARR(N_ELEMENTS(organ), 3)
4   ...
5   phantomMask =$
6    BYTARR(phantom.dim(0), phantom.dim(0), phantom.dim(1))
7   FOR i = 1, N_ELEMENTS(organ)-1 DO BEGIN
8    phantomMask(*,*,*) = 0
9    CASE organ(i).value OF
10    1:    organLocationPhantom=WHERE(phantom.cache(*,*,*,0),$
11          organSizePhantom)
12    101:  organLocationPhantom=WHERE((phantom.cache(*,*,*,0) EQ 35)$
13          OR (phantom.cache(*,*,*,0) EQ 36), organSizePhantom)
14    102:  organLocationPhantom=WHERE((phantom.cache(*,*,*,0) GE 18)$
15          AND (phantom.cache(*,*,*,0) LE 25), organSizePhantom)
16    103:  organLocationPhantom=WHERE((phantom.cache(*,*,*,0) GE 15)$
17          AND (phantom.cache(*,*,*,0) LE 17), organSizePhantom)
18    ELSE: organLocationPhantom=WHERE(phantom.cache(*,*,*,0)$
19          EQ organ(i).value, organSizePhantom)
20   ENDCASE
21   phantomMask(organLocationPhantom) = 1
22   organLocationPhantomIndices = $
23    ARRAY_INDICES(phantom.cache(*,*,*,0), organLocationPhantom)
24   Fig:Resizing_OrgansminX = MIN(organLocationPhantomIndices(0, *))
25   maxX = MAX(organLocationPhantomIndices(0, *))
26   minY = MIN(organLocationPhantomIndices(1, *))
27   ...
28   FOR ii=0,2 DO $
29    phantomMaskLocation(i,ii)=MEAN(organLocationPhantomIndices(ii,*))
30   (*ptrPhantomMask(i))=phantomMask(minX:maxX,minY:maxY,minZ:maxZ)
31  END
```

3.2.3 Resizing Organs

Before the actual resizing takes place (as illustrated in Fig. 3.5), the COM of all selected organs is computed. Fist, only the selected organ voxels[4] are summed up in a `FOR` loop (lines 3-6). Then the COM of each PET file organ is calculated and stored in the array `petFileMaskLocation`. This is done by averaging over all voxels x-,y- and z-coordinates separately. Then the COM of all selected organs for both phantom (`phantomCoM`) and pet file (`petFileCoM`) are computed. Therefore the mask location vectors are weighted by the mask sizes and summed up, providing the COMs (one for the PET and one for the phantom file) of all organs (lines 12-15).

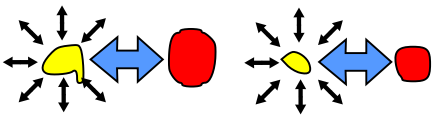


Figure 3.5 Resizing Organs

The calculation of the COMs is done, because in a final step `phantomCoM` and `petFileCoM` will be positioned on top of each other, providing a starting point for the reconstructed organs.

---

[4] (`organ(i).defined= '*_'`) is true for selected organs only (see Table 3.3).

```
FOR i = 1, N_ELEMENTS(organ)-1 DO BEGIN
 IF organ(i).defined EQ ’*_’ THEN BEGIN
  organLocationPetFile=WHERE(organ(i).mask,organSizePetFile)
  organLocationPhantom=WHERE((*ptrPhantomMask(i)),organSizePhantom)
  summedOrganSizePetFile=summedOrganSizePetFile+organSizePetFile
  summedOrganSizePhantom=summedOrganSizePhantom+organSizePhantom
  organLocationPetFileIndices = $
   ARRAY_INDICES(organ(i).mask,organLocationPetFile)
  FOR ii=0,2 DO BEGIN
   petFileMaskLocation(i, ii) = $
    MEAN(organLocationPetFileIndices(ii, *))
   petFileCoM(ii) = petFileCoM(ii) + $
    petFileMaskLocation(i, ii)*organSizePetFile
   phantomCoM(ii) = phantomCoM(ii) + $
    phantomMaskLocation(i, ii)*organSizePhantom
  END
 END
END
...
petFileCoM = petFileCoM/summedOrganSizePetFile
phantomCoM = phantomCoM/summedOrganSizePhantom
```

Before the organ sizes between the PET and phantom file can be compared, they have to be rescaled. This has to be done, because the phantom and PET file voxels have different dimensions. The phantom data consists of cubical voxels, whereas the PET file voxels are cuboids. `scaleXY` and `scaleZ` are defined as shown below, to get the ratio between xy and z dimensions. The overall `scalingFactor`, `scaleXY` and `scaleZ` will be used to resize organ and organ location vectors afterwards.

```
scale =1.*MIN([petFile.magnification(0),petFile.magnification(1)])
scaleXY =1./(petFile.magnification(0)/scale)
scaleZ =1./(petFile.magnification(1)/scale)
scalingFactor = (1.*summedOrganSizePetFile/ $
 (summedOrganSizePhantom*scaleXY^2*scaleZ))^(1./3)
```

In the following `FOR` loop the actual resizing takes place. First the size, the mask should be resized to (`maskSize`), is calculated. `maskSize` is a three dimensional integer array. A new temporary pointer `tmpPtrPhantomMask` to a byte array of `maskSize`'s size is defined. If `maskSize` is too small, a flag in the binary array `toSmallOrgans` is set[5]. With function `CONGRID` each phantom mask (`*ptrPhantomMask(i)`) is resized to `maskSize` and the resulting mask is stored in `*tmpPtrPhantomMask`. Afterwards the unscaled phantom mask is deleted with `PTR_FREE` and `ptrPhantomMask(i)` is set to point at the resized organ.

---

[5] see Table 3.3

```
FOR i = 1, N_ELEMENTS(organ)-1 DO BEGIN
 maskSize = ROUND(SIZE((*ptrPhantomMask(i)), /DIMENSIONS) * $
  [scaleXY, scaleXY, scaleZ] * scalingFactor)
 tempPtrPhantomMask = PTR_NEW( BYTARR(maskSize) )
 FOR ii=0,2 DO IF maskSize(ii) LT 2 THEN toSmallOrgans(i) = 1
 (*tempPtrPhantomMask) = ROUND( $
  CONGRID(1.*(*ptrPhantomMask(i)), $
  maskSize(0), maskSize(1), maskSize(2)) )
 PTR_FREE, ptrPhantomMask(i)
 ptrPhantomMask(i) = tempPtrPhantomMask
END
```

3.2.4 Rescaling Distancies

Before resizing the vectors to the COMs of all organs (`phantomMaskLocation`) (shown in Fig. 3.6) and to the COM of all selected organs (`phantomCoM`), they have to be rescaled to be comparable to the PET file vectors. Therefore `scaleXY`, `scaleZ` and `scalingFactor` (as shown in Subsection 3.2.3) are used.

```
FOR i = 1, N_ELEMENTS(organ)-1 DO BEGIN
 phantomMaskLocation(i,*) = phantomMaskLocation(i,*)* $
  [scaleXY,scaleXY,scaleZ]*scalingFactor
END
phantomCoMScaled = phantomCoM*[scaleXY,scaleXY,scaleZ]* $
 scalingFactor
```

The variable `lengthOrganToCoM` is used to rescale distances between organs. In a `FOR` loop all distances between the COM of all selected organs and the COM of each organ, both for PET (`lengthOrganToCoMPetFile`) and phantom masks (`lengthOrganToCoMPhantom`), are calculated. The mean ratio between `lengthOrganToCoMPetFile` and `lengthOrganToCoMPhantom` is stored in `lengthOrganToCoM`. If only one organ is defined, there is no need to calculate `lengthOrganToCoM`, because in this case the vectors `phantomCoM` and `phantomMaskLocation` are the same and therefore `lengthOrganToCoM` is set to one.

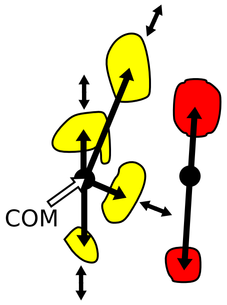


Figure 3.6 Rescaling Distancies

Now all organ vectors starting from `phantomCoM` to the organ's COM will be multiplied with `lengthOrganToCoM`. Normally the `lengthOrganToCoM` factor should be close to one. If `lengthOrganToCoM` equals one, the investigated mouse corresponds to a rescaled phantom mouse, keeping all proportions. But for mice, whose distances between the organs are above-average, compared to the organ size, (eg. fat mice) the factor `lengthOrganToCoM` will slightly be greater than one.

```
1   lengthScaleOrganToCoM = 0.
2   IF definedOrgans NE 1 THEN BEGIN
3    FOR i=1, N_ELEMENTS(organ)-1 DO BEGIN
4     IF organ(i).defined EQ '*_' THEN BEGIN
5      lengthOrganToCoMPhantom = 0.
6      lengthOrganToCoMPetFile = 0.
7      FOR ii=0,2 DO BEGIN
8       lengthOrganToCoMPhantom = lengthOrganToCoMPhantom + $
9        (phantomCoMScaled(ii) - phantomMaskLocation(i, ii))^2
10      lengthOrganToCoMPetFile = lengthOrganToCoMPetFile + $
11       (petFileCoM(ii) - petFileMaskLocation(i, ii))^2
12     END
13     lengthScaleOrganToCoM = lengthScaleOrganToCoM + $
14      (lengthOrganToCoMPetFile/lengthOrganToCoMPhantom)^(1./2)
15    END
16   END
17  END ELSE BEGIN
18   lengthScaleOrganToCoM = 1.
19  END
20  lengthScaleOrganToCoM = lengthScaleOrganToCoM/definedOrgans
```

3.2.5 Determine Angles

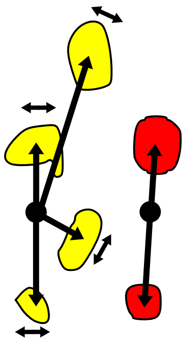


Figure 3.7 Determine Angles

Normally the investigated mouse does not have the same orientation as the phantom mouse. In order to provide a better overlap between the reconstructed masks and the defined masks, the phantom masks are rotated around the x, y and z-axis (shown in Fig. 3.7). In three nested `FOR` loops each angle is varied in an interval of ten degrees ([-5,+5]) with a step width of one degree[6]. The overlap of each position is measured and stored in variable `overallQuality`. Then the angle settings that provide the best overlap are used to move all organs into their most likely position. Therefore the organ masks and their location vectors are rotated around the COM.

All organ masks are rotated, resized and brought into the correct position. Then the overlap of the reconstructed masks with the user defined organs is calculated. `overallQuality` is defined as an array of floats. It keeps the mean overlap of the selected organ masks (`organ(i).mask`) and the reconstructed organ masks (`tempOrganMask`) for every angle combination.

`organ(i).mask` and `tempOrganMask` are binary. Their values are zero, if a voxel is not part of this organ, or one, if a voxel is part of this organ. So each element of `organ(i).mask` just has to be multiplied with `tempOrganMask` to get the total overlap (lines 12-13). With the help of function `TOTAL` all values in an array are summed up, thus giving us the quantity of all overlapping organ's voxels. After dividing through the organ sizes and the number of organs, `overallQuality` equals the percental overlap. `overallQuality` is calculated for all combinations of different angles. Finally the desired angle (`desiredAngle`) is found by locating the maximum value of `overallQuality`.

---

[6] The starting angles can be defined by the user.

```
overallQuality = FLTARR(11, 11, 11)
...
FOR angleX = -5,5 DO BEGIN
 FOR angleY = -5,5 DO BEGIN
  FOR angleZ = -5,5 DO BEGIN
   FOR i=1,  N_ELEMENTS(organ)-1 DO BEGIN
    IF organ(i).defined EQ '*_' THEN BEGIN
     ...
     ;; ROTATING MASKS AND VECTORS
     ;; RESIZE MASKS AND VECTORS
     ...
     quality = TOTAL(organ(i).mask * tempOrganMask)/ $
      ((TOTAL(organ(i).mask) + TOTAL(tempOrganMask))/2)
     overallQuality(angleX+5, angleY+5, angleZ+5) = $
      overallQuality(angleX+5, angleY+5, angleZ+5) + quality
     ...
    END
   END
  END
 END
END
overallQuality = overallQuality/definedOrgans
maximumQuality = MAX( overallQuality, location )
desiredAngle = ARRAY_INDICES( overallQuality, location ) - [5,5,5]
```

Before calculating `overallQuality` the organs have to be turned into the correct position. The organ masks are rotated using the `rotateMask` function, which is shown below. Before rotation, the mask that exactly fits the organ has to be enlarged, so that there is enough space to turn the mask around without touching the array's borders. Therefore the array is enlarged to serve the worst case, where the mask exactly lies diagonally in the mask's array cuboid. The bigger mask (`tempMask`), which temporarily stores the mask, is a cube with a side length as long as the diagonal of the old mask (`mask`). The rotation of `tempMask` is done by the optimized, IDL-function `ROT`. After rotation, the mask is cut to a minimal cuboid (lines 16-20) and stored into `*ptrMask`. Finally function `rotateMask` returns the pointer `ptrMask` to the rotated mask.

Fig. 3.8 shows an 2D example for `rotateMask`. The dark-red object represents the organ. The box on the left and the dotted box at the right represent the old mask. The green line is the diagonal length of the old mask. The dashed box is the temporary mask `tempMask` and the solid frame shows the returned cut mask (`*ptrMask`).

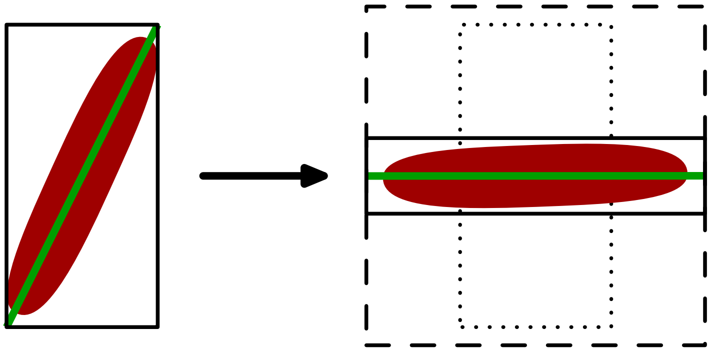


Figure 3.8 The wost-case scenario for function rotateMask

```
FUNCTION rotateMask, mask, degreeX, degreeY, degreeZ
 maskDim = SIZE((mask), /DIMENSIONS)
 tempMaskDim = CEIL( (TOTAL((1.*maskDim)^2))^(1./2) ) + 1.
 tempMask = BYTARR( tempMaskDim, tempMaskDim, tempMaskDim )
 offset = 1.*(tempMaskDim - maskDim)/2
 tempMask( offset(0), offset(1), offset(2) ) = mask
 ...
 FOR i = 0, tempMaskDim-1 DO tempMask(i,*,*) = $
  ROT( REFORM(tempMask(i,*,*)), degreeX )
 FOR i = 0, tempMaskDim-1 DO tempMask(*,i,*) = $
  ROT( REFORM(tempMask(*,i,*)), degreeY )
 FOR i = 0, tempMaskDim-1 DO tempMask(*,*,i) = $
  ROT( REFORM(tempMask(*,*,i)), degreeZ )
 ...
 tempMaskNotZero = WHERE( tempMask )
 tempMaskNotZeroIndices = ARRAY_INDICES(tempMask, tempMaskNotZero)
 minX = MIN(tempMaskNotZeroIndices(0, *))
 maxX = MAX(tempMaskNotZeroIndices(0, *))
 minY = MIN(tempMaskNotZeroIndices(1, *))
 ...
 ptrMask = PTR_NEW(BYTARR(maxX-minX+1, maxY-minY+1, maxZ-minZ+1))
 (*ptrMask) = tempMask(minX:maxX, minY:maxY, minZ:maxZ)
 RETURN, ptrMask
END
```

After having turned the organ masks, the vectors from the COM of all defined organs to each organ's COM are rotated. This task is done by function `rotateVector`. With a simple rotation operator (`rotationMatrix`) these vectors easily can be rotated into the right position.

```
FUNCTION rotateVector, vec, degreeX, degreeY, degreeZ
 VecX = vec(0)
 VecY = vec(1)
 VecZ = vec(2)
 a = rotationMatrix(degreeX) ## [[VecY], [VecZ]]
 VecY = a(0)
 VecZ = a(1)
 a = rotationMatrix(degreeY) ## [[VecX], [VecZ]]
 VecX = a(0)
 VecZ = a(1)
 a = rotationMatrix(degreeZ) ## [[VecX], [VecY]]
 VecX = a(0)
 VecY = a(1)
 RETURN, [VecX, VecY, VecZ]
END
```

Function `rotationMatrix` is used by `rotateVector` and represents an euclidic, orthogonal, two dimensional rotation matrix.

$$\texttt{rotationMatrix}(\beta) = \begin{pmatrix} \cos\beta & \sin\beta \\ -\sin\beta & \cos\beta \end{pmatrix}$$

```
FUNCTION rotationMatrix, a
 b = a*!PI/180
 RETURN, [[COS(b),SIN(b)],[-SIN(b),COS(b)]]
END
```

After mask- and vector rotation the masks have to be rescaled and resized as seen before in Chapters 3.2.3 and 3.2.4. Then the masks with the maximum overlap are taken and the rescaled and resized phantom organ masks are used to define the remaining non selected organs. This is shown in Subsection 3.2.6.

3.2.6 Affiliate Organs

After having determined the optimal angle, resizing and rescaling, the organ masks have to be put together (shown in Fig. 3.9). How this is done, is shown below. All organ masks are merged to organ masks with the dimensions of the PET file. This is done for reasons of simplicity and performance, because with this method all PET data referring to an organ can be received by simply multiplying each PET data matrix value with the organ's mask (this will be shown in Subsection 3.3.2 and Section 3.4).

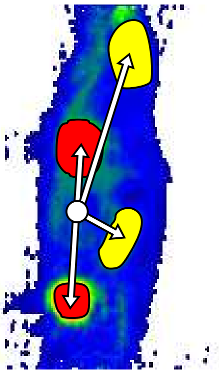


Figure 3.9 Affiliate Organs

In order to place the small organs, the coordinate offset in the bigger mask (PET file sized) must be known. The vectors used to calculate the offset coordinates (`coord`) are shown in Fig. 3.10.[7] The vectors `petFileCOM` and `phantomMaskLocation` are know. The vector `center` has to be calculated first and represents the COM of the small organ mask. By simple vector addition one gets `coord`.

Because the organ might only be partially seen or even not be visible, it is necessary to check, if the small mask fits into the PET file size mask. Protruding organ parts have to be truncated and `organ(i).defined` flags are set (the legend is shown in Table 3.3).

flag	meaning
?_	organ mask is not defined
*_	organ mask has been defined and selected by the user
+_	organ mask has been calculated and fits in entirely
!_	organ mask has been calculated but does not fit entirely
-_	organ mask has been calculated but cannot be seen

Table 3.3 Legend of organ.defined-flags

---

[7] The vectors named in Fig. 3.10 equal the syntax names to preserve consistence

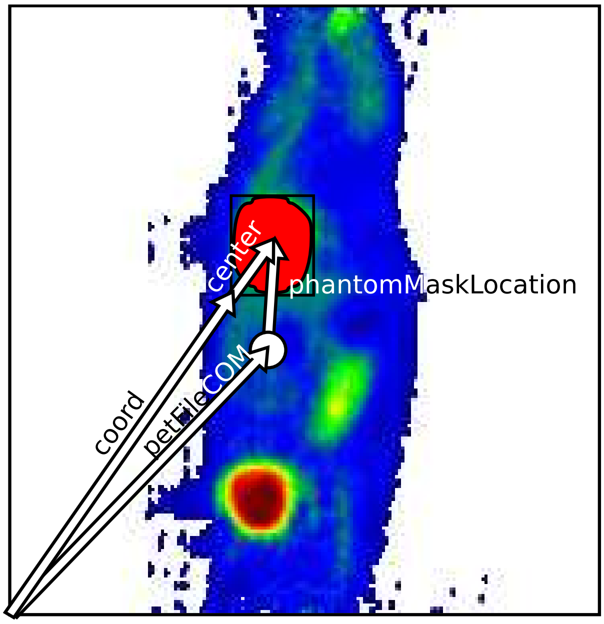


Figure 3.10 Illustration of transforming the phantom organ masks to globally used organ masks

Merging is done in a `FOR` loop over all organs. The temporary used PET sized mask `tempOrganMask` is cleared before processing each organ. It stores the organ mask, until the mask is passed over to `organ(i).mask`. If the organ is big enough to be fitted[8], the COM of all small organ masks, that have not been defined by the user, are calculated (lines 6-9). Then the offset coordinates `coord` are computed in a vector addition (as illustrated in Fig. 3.10). If the organ mask fits (lines 12-19) then the small organ mask `*ptrPhantomMask` can easily be stored in `organ(i).mask` using `coord` as offset. If it does not fit, the mask has to be truncated (lines 25-36). Therefore the following IDL syntax is used:

$$\texttt{tempOrganMask}(X_{Offset}, ...) = \texttt{*ptrPhantomMask}(X_{Index}^{Min} : X_{Index}^{Max}, ...)$$

In order to ensure that $\left(X_{Offset} + X_{Index}^{Max}\right)$ does not exceed the x-dimension of `tempOrganMask`, X_{Index}^{Max} has to be harmonized with the x-dimension of `tempOrganMask`.[9] This is done using the arrays `minX`, `maxX`, `minY` and so on (see below). This avoids that `tempOrganMask` exceeds positive or negative x-, y- or z-dimension.

---

[8] It must at least be a three dimensional array.

[9] It is the same for x-,y- and z-coordinate. Therefore explanation is limited to the x-coordinate.

```
FOR i = 1, N_ELEMENTS(organ)-1 DO BEGIN
 tempOrganMask(*,*,*) = 0
 IF ~toSmallOrgans(i) THEN BEGIN
  ...
  IF organ(i).defined NE ’*_’ THEN BEGIN
   location = WHERE((*ptrPhantomMask(i)), maskSize)
   locationIndices = ARRAY_INDICES((*ptrPhantomMask(i)), location)
   FOR ii=0,2 DO center(ii) = MEAN(locationIndices(ii, *))
   FOR ii=0,2 DO coord(ii) = ROUND(petFileCoM(ii) + $
    phantomMaskLocation(i, ii) - center(ii))
   maskSize = SIZE((*ptrPhantomMask(i)), /DIMENSIONS)
   IF $
      (coord(0) GE 0) AND (coord(1) GE 0) AND (coord(2) GE 0) AND $
    ( (coord(0) + maskSize(0)) LT petFile.dim(0) ) AND ... $
   THEN BEGIN
    tempOrganMask( coord(0), coord(1), coord(2) ) $
     = (*ptrPhantomMask(i))
    organ(i).mask = tempOrganMask
    organ(i).defined = ’+_’
   END ELSE BEGIN
    minX=([coord(0),0])
    maxX=([maskSize(0)-1,MAX([petFile.dim(0)-coord(0)-1,0])])
    minY=([coord(1),0])
    ...
    IF  (-MIN(minX) GE MIN(maxX)) OR ... OR $
     (MAX(minX) GE petFile.dim(0)) OR ... $
    THEN BEGIN
     organ(i).mask = tempOrganMask
     organ(i).defined = ’-_’
    END ELSE BEGIN
     tempOrganMask( MAX(minX), MAX(minY), MAX(minZ) ) = $
      (*ptrPhantomMask(i))(-MIN(minX):MIN(maxX), $
      -MIN(minY):MIN(maxY),-MIN(minZ):MIN(maxZ))
     organ(i).mask = tempOrganMask
     organ(i).defined = ’!_’
    END
   END
  END
 END ELSE BEGIN
  organ(i).mask = tempOrganMask
  organ(i).defined = ’-_’
 END
END
```

3.3 Output

The defined and reconstructed orgran maps can be saved to disk. The saving process will be shown in Subsection 3.3.1. Afterwards the calculation of TACs (Time Activity Curves) and how these TACs are saved, is shown Subsection 3.3.2.

3.3.1 Saving Organ Maps

The calculated organ masks can be saved to continue working with the user defined or calculated organ masks. They are stored in a file, which is named like the PET file, in the progam's subdirectory "Output". If you click "File" → "Save Organ Masks" at the menu bar procedure `on_W_MENU_SAVEORGANMASKS` will be called. This procedure executes `writeMaskFile` and prompts a message, if saving has finished successfully.

```
PRO on_W_MENU_SAVEORGANMASKS, Event
 ...
 writeMaskFile, ".\Output\"+petFile.fileName+".mask.gz"
 dialog = DIALOG_MESSAGE("Masks saved successfully.", /INFORMATION)
END
```

Procedure `writeMaskFile` uses the widely used GZIP[10] format for compression. All organ masks are binary files that consist just of "0" and "1" making compression highly efficient. First the PET file's name and dimension is stored. Then all organ masks are saved.

```
PRO writeMaskFile, outFile
 ...
 OPENW, outLun, outFile, /GET_LUN, /COMPRESS
 WRITEU, outLun, STRING(petFile.fileName, FORMAT='(A255)')
 WRITEU, outLun, petFile.dim
 outFileData = organ
 ...
 WRITEU, outLun, outFileData
 CLOSE, outLun, /ALL
END
```

3.3.2 Generating TAC Files

TACs (Time Activity Curves) are used to follow time dependent biodistribution of the radiopharmacon. Fig. 3.11 shows the tracer's distribution in bladder, brain, heart and left kidney within an interval of thirty minutes. In this example the TACs are measured in Standardized Uptake Values (SUVs), which are explained below. The software is also capable to calculate other common units like "mCi", "mCi/cc"[11], "MBq", "MBq/cc" and "%ID/g"[12]. Because all other units can more or less be calculated by conversion factors, this Subsection just focuses on "SUVs".

A SUV (Standardized Uptake Value) is the mean value of all organ voxels divided through the injected dose and mulpiplied with the mouse's weight.

---

[10] See www.zlib.org for details.

[11] cc means cubic centimeters.

[12] percentage Injected Dose per gram

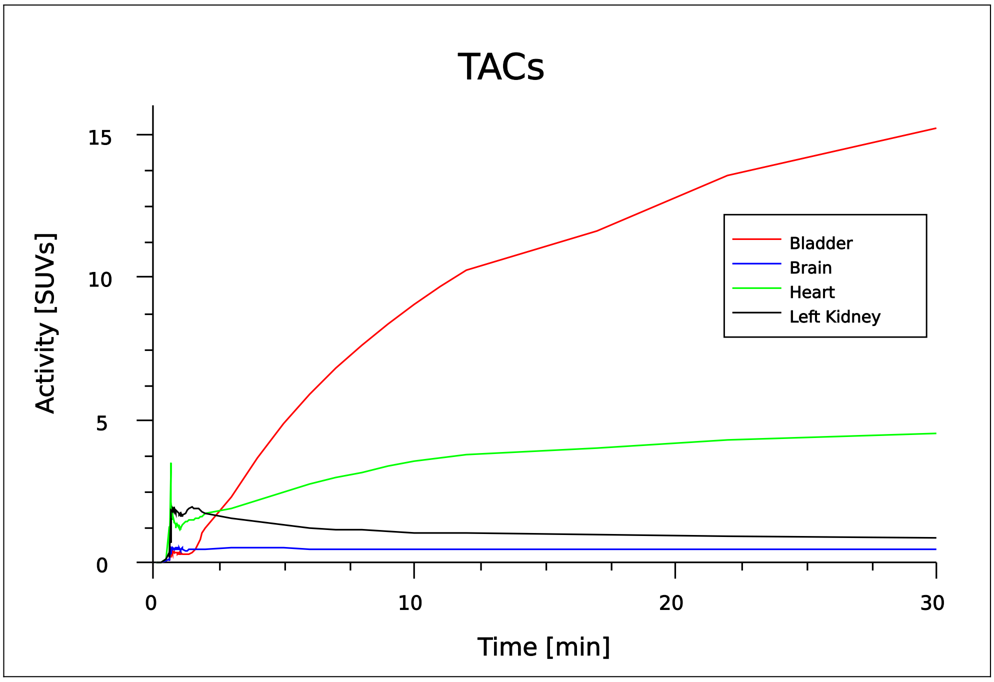


Figure 3.11 Time Activity Curves

By clicking “Calculate” → “TACs [SUVs]” at the menu bar, procedure `on_W_MENU_TACS_SUVS` will be called. The procedure checks, if a PET image file is opened. If no file is opened, it exits. The calculation itself is done by function `calculateTACs`, which returns a struct consisting of TACs and organ sizes. The calculated TAC files are stored in a file, which is named like the PET file plus ending “_SUVs.txt”, in the progam’s subdirectory “Output”. If everything was successful, “TAC Calculation finished.” is prompted.

```
PRO on_W_MENU_TACS_SUVS, Event
 ...
 IF ~petFileOpen() THEN RETURN
 calculatedOrgan = calculateTACs("SUVs")
 IF N_ELEMENTS( calculatedOrgan ) EQ 1 THEN RETURN
 writeTACFile, ".\Output\"+petFile.fileName+"_SUVs.txt", $
  calculatedOrgan, "SUVs"
 dialog = DIALOG_MESSAGE("TAC Calculation finished.",/INFORMATION)
END
```

Function `calculateTACs` returns a struct (`calculatedOrgan`), which consits of the measured TAC and organ size (in voxel and cubical centimeters) for each organ. First all organ sizes are computed using the IDL function `WHERE`, which provides the organ size in voxels. Multiplying with the voxel’s dimensions the size in cubical centimeters is obtained.

The following calculation depends on which TAC unit has been selected. In the following description SUVs are used. For each frame[13], the frame data is read in, if not already cached. This is provided by function `readFile`. In a second `FOR` loop over all organs, all measured voxel values of the current organ are summed up by `TOTAL`, multiplied with a

---

[13] `petFile.dim(2)` equals the number of time frames.

calibration factor[14], divided through the organ size, divided through the injected dose and multiplied with the mouse's weight.

After calculation the struct `calculatedOrgan` is returned to the calling procedure.

```
FUNCTION calculateTACs, unit
 ...
 calculatedOrgan = ""
 calculatedOrgan = CREATE_STRUCT('TAC',FLTARR(petFile.dim(2)), $
  'SizeVoxel', LONARR(1), 'SizeCC',FLTARR(1) )
 calculatedOrgan = REPLICATE(calculatedOrgan, N_ELEMENTS(organ))
 FOR organNr = 0, N_ELEMENTS(organ)-1 DO BEGIN
  location = WHERE( organ(organNr).mask, organSizeVoxel )
  calculatedOrgan(organNr).SizeVoxel = organSizeVoxel
  calculatedOrgan(organNr).SizeCC = $
   calculatedOrgan(organNr).SizeVoxel*petFile.magnification(0)^2*$
   petFile.magnification(1)
 END
 CASE unit OF
  "mCi": ...
  "mCi/cc": ...
  "MBq": ...
  "MBq/cc": ...
  "SUVs": BEGIN
    FOR frame = 0, petFile.dim(2)-1 DO BEGIN
     IF readFile(petFile, frame) THEN RETURN, 1
     FOR organNr = 0, N_ELEMENTS(organ)-1 DO BEGIN
      calculatedOrgan(organNr).TAC(frame) = $
       (1.*TOTAL(petFile.cache(*,*,*,frame)*organ(organNr).mask)*$
       calibration.factor/calculatedOrgan(organNr).SizeVoxel)/ $
       (calibration.injectedDose/calibration.mouseWeight)
     END
    END
   END
  "%ID/g": ...
  ELSE: RETURN, 2
 ENDCASE
 RETURN, calculatedOrgan
END
```

Procedure `writeTACFile` now saves all TAC data provided by `calculateTACs`. This is quite a large procedure, so only the more important passages are shown. `timeArray`, which is an array providing the absolute measurement times, is generated based on the frame durations (`petFile.frameDuration(i)`).

First the file name is written by `PRINTF`. With "`PRINTF, outLun, FORMAT='(80("-"))'`" a horizontal line consisting of 80 "-" is printed. Then the calibration factor, the mouse weight and the injected dose are stored. After another horizontal line the elements of `timeArray` are fused to a solid string, using function `STRJOIN` and printed. The following lines are filled with the SUVs of each organ.

---

[14] The calibration factor transforms machine values to physical units.

At the end of the file the organ sizes (in cubical centimeters and voxels) are written to the file.[15]

```
PRO writeTACFile, outFile, calculatedOrgan, unit
 timeArray = FLTARR(petFile.dim(2))
 timeArray(0) = petFile.frameDuration(0)
 FOR i=1, petFile.dim(2)-1 DO timeArray(i) = $
  timeArray(i-1) + petFile.frameDuration(i)
 OPENW, outLun, outFile, /GET_LUN
 outString = petFile.fileName
 PRINTF, outLun, "Filename: ", outString, FORMAT=’(A0, A0)’
 PRINTF, outLun, FORMAT=’(80("-"))’
 outString = STRING(FORMAT=’(G15.7)’, calibration.factor*10.^6)
 PRINTF, outLun, FORMAT=’(A0, T30, A0, T60, A0)’, $
 "Calibration_Factor", "([nCi]/[cc])/[PET-Units]", outString
 ...
 outString = STRJOIN( $
  STRING(FORMAT=’(G15.7)’, timeArray ), /SINGLE )
 PRINTF, outLun, FORMAT=’(A0, T30, A0, T60, A0)’, $
  "Time", "[s]", outString
 FOR i=0, N_ELEMENTS(organ)-1 DO BEGIN
  outString = STRJOIN( $
   STRING(FORMAT=’(G15.7)’, calculatedOrgan(i).TAC(*)), /SINGLE )
  PRINTF, outLun, FORMAT=’(A0, T30, A0, T60, A0)’, $
   organ(i).defined+organ(i).name, "["+unit+"]", outString
 END
 ...
 CLOSE, outLun, /ALL
END
```

An example TAC is shown below.

```
Filename: 060510_M1_em_FDG_bolus_dynamic_v1
-----------------------------------------------------------------
Calibration_Factor ([nCi]/[cc])/[PET-Units]     7437390.
Mouse_Weight       [g]                          33.50000
Injected_Dose      [mCi]                       0.5520000
-----------------------------------------------------------------
Time               [s]           10.00000       20.00000     ...
?_choose           [SUVs]    0.0008017067   0.0008581656     ...
!_Body             [SUVs]     0.001376659    0.001497061     ...
+_Liver            [SUVs]     0.001477350    0.001535296     ...
...                ...                ...            ...     ...
----------------------------------------------------------------
Organ                         Volume [cc]   Volume [Vox]
?_choose                         198.2492       1556480.
!_Body                           38.98878       306106.0
+_Liver                          3.413012       26796.00
...                                   ...            ...
```

---

[15] This is not shown in the syntax extract.

3.4 Image Processing

This Section gives a brief overview on how images are scaled and displayed in the thesis' software. Not all graphical capabilities of the computer program will be covered here. 2D and 3D images are drawn by the procedures `draw2D` and `draw3D`, which demand three variables: `baseNr`, `data` and `frame`.

```
draw2D, baseNr, data, frame
draw3D, baseNr, data, frame
```

Draw widgets (as shown in Fig. 3.1) are horizontally aligned. The first row has `baseNr` "0", the second "1" and the third "2". If `baseNr` is set to "1" the second row of draw widgets will be filled with `data` at time frame number `frame`.

For both `draw2D` and `draw3D` data sets are rescaled first.[16] This is done by the procedure `scaleDrawCache` (as shown below). First the 3D data set's minimal value is set to zero (the modified data is stored in `*scaledDrawData`), because for some reconstruction algorithms negative voxel values (doses) can occur. Then `*scaledDrawData` is divided by its maximum value. Now all data values lie in between "0" and "1". In the graphical user interface the minimum and the maximum value (in percent of the (maximum – minimum) value), that should be displayed, can be entered. Function `BYTSCL` assigns each value between `MIN` and `MAX` a value between "0" and "255", whereat values smaller than `MIN` are set to "0" and values larger then `MAX` are set to "255". Again `*scaledDrawData` is divided by its maximum value. Finally each element of the rescaled data is raised to a user-set power. This is very useful to discriminate regions with high values from those with lower ones.

```
PRO scaleDrawCache, baseNr, is2Dor3D, unscaledData
 ...
 IF ~PTR_VALID( scaledDrawData(baseNr, is2Dor3D) ) THEN $
  scaledDrawData(baseNr, is2Dor3D) = $
  PTR_NEW( FLTARR(SIZE(unscaledData, /DIMENSIONS)) )
 *scaledDrawData(baseNr, is2Dor3D) = $
  1.* unscaledData-MIN(unscaledData)
 *scaledDrawData(baseNr, is2Dor3D) = $
  *scaledDrawData(baseNr, is2Dor3D)/$
  MAX(*scaledDrawData(baseNr, is2Dor3D))
 *scaledDrawData(baseNr, is2Dor3D) = $
  1.*BYTSCL(*scaledDrawData(baseNr, is2Dor3D), $
  MIN=settings(0)/100., MAX=settings(1)/100., TOP=!D.TABLE_SIZE)
 *scaledDrawData(baseNr, is2Dor3D) = $
  *scaledDrawData(baseNr, is2Dor3D)/$
  MAX(*scaledDrawData(baseNr, is2Dor3D))
 *scaledDrawData(baseNr, is2Dor3D) = $
  (*scaledDrawData(baseNr, is2Dor3D))^(settings(2))
 scaledDrawIsCached(baseNr, is2Dor3D) = 1
 ...
END
```

---

[16] The scaled data (`scaledDrawData`) is only used for image display.

3.4.1 2D Slices

Procedure `draw2D` (shown below) can be divided into three parts. First the procedure checks, if the current data set has already been scaled. If not, procedure `scaledDrawCache` is used to scale the data. Second the current cursor position, provided by the slider bars, is read out. And third the 2D slices (sagital, coronal, axial) are drawn by `drawSlice2D`.

```
PRO draw2D, baseNr, data, frame
 ...
 IF ~scaledDrawIsCached(baseNr, 0) THEN $
  scaleDrawCache, baseNr, 0, data.cache(*,*,*,frame)
 ...
 pos = INTARR(3)
 FOR i=0,2 DO BEGIN
  WIDGET_CONTROL, id.slider(baseNr, i), GET_VALUE=a
  pos(i) = a
 ENDFOR
 ...
 drawSlice2D, baseNr, 0, REFORM( (*scaledDrawData(baseNr, 0))$
    (pos(0), *, *) ), data.dim(0), data.dim(1),$
    data.magnification(0), data.magnification(1),$
    pos(1), pos(2), offset(BaseNr,1), offset(BaseNr,2),$
    windowSize.xy, windowSize.z, settings
 drawSlice2D, baseNr, 1, ..
 drawSlice2D, baseNr, 2, ...
END
```

Procedure `drawSlice2D` (shown below) is used for drawing 2D slices. Because `drawSlice2D` is quite unimaginative and difficult to understand, Fig. 3.12 is used to illustrate the procedure.

The unscaled data does not smoothly fit into the draw widget. It has to be rectified, enlarged and centered in the draw widget (shown on the left hand side of Fig. 3.12). This is done by `drawSlice2D`.

First the procedure calculates how often the image will fit into the draw widget. This is stored into `scaleX` and `scaleY`. A `scaleX` factor of “2” means that the draw widget’s x-size is double the image’s x-size. `windowX` and `windowY` represent the width and height of the draw widget (the black boxes in Fig. 3.12). `dimX` and `dimY` are the width and height of the image. `magX` and `magY` represent the x- and y-dimension of every voxel’s orthogonal projection along the displayed axis. Factor `scale` equals the minimum of `scaleX` and `scaleY`. `scale` is the enlargement factor that has to be used to fit the image into the draw widget. `scalingOffset` is a two dimensional array, that represents the needed drawing offset to center rescaled image. `zoom` is a user defined zoom factor, that enables the user to enlarge the picture.

`paintingFactor` is used to draw a frame in background color around the image (symbolized by the dashed boxes on the right hand side of Fig. 3.12). This is done, because, if the picture is zoomed or moved in the draw widget, the old picture fragments are overdrawn by background color. The frame’s thickness is calculated by the `paintingFactor` multiplied by the x- and y-dimensions of the image and is stored in `paintingOffset`. The image including the frame is stored in `painting`.

Then the final dimensions of the picture including the frame (`dimPaintingWithFrame`) is computed. The IDL function `CONGRID` resizes `painting`. Afterwards a cursor is drawn, showing the current cursor position (`cursorX` and `cursorY`). Then the draw widget, which should be painted on, is selected by `WIDGET_CONTROL` and by `WSET`, and finally the painting is drawn by `TVSCL` using the calculated x- and y-offsets.[17]

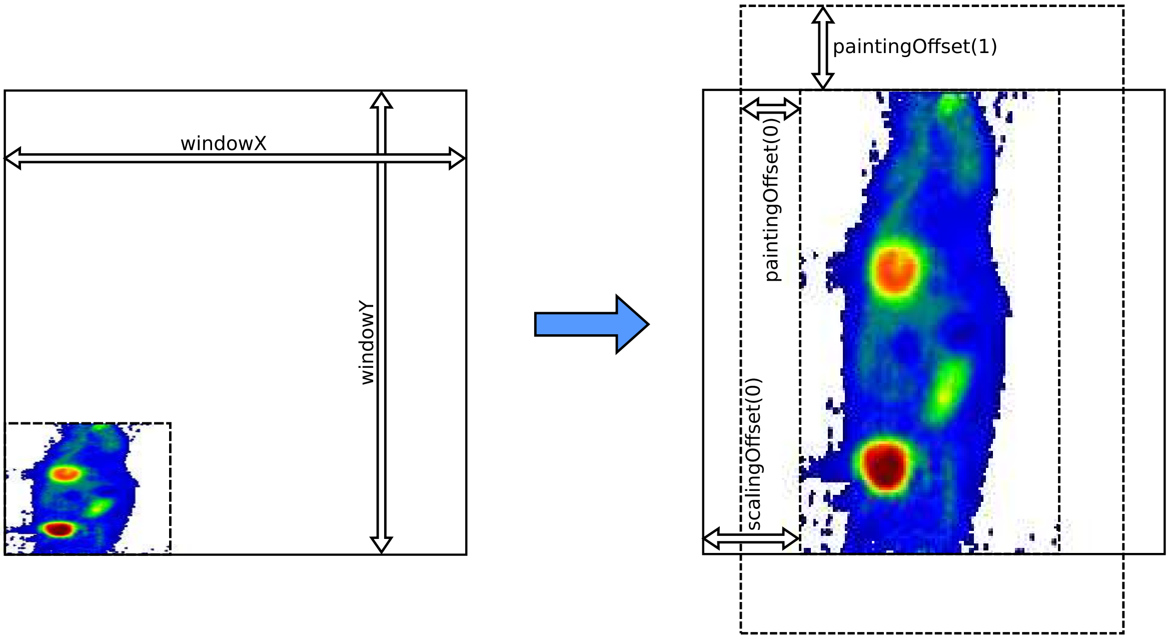


Figure 3.12 Illustration of `drawSlice2D`

---

[17] `offsetX` and `offsetY` are additional user defined offsets.

```
PRO drawSlice2D, baseNr, drawNr, slice, dimX, dimY, magX, magY,$
   cursorX, cursorY, offsetX, offsetY, windowX, windowY, settings

 scalingOffset=FLTARR(2)
 scaleX = 1.*windowX/(dimX*magX)
 scaleY = 1.*windowY/(dimY*magY)
 scale = MIN([scaleX, scaleY])
 scalingOffset(0)=(windowX - dimX*zoom(baseNr)*scale*magX)/2.
 scalingOffset(1)=(windowY - dimY*zoom(baseNr)*scale*magY)/2.
 paintingFactor = 0.2
 paintingOffset = FLTARR(2)
 painting = FLTARR(dimX*(paintingFactor*2+1),$
  dimY*(paintingFactor*2+1))
 paintingOffset(0) = dimX*paintingFactor
 paintingOffset(1) = dimY*paintingFactor
 painting(paintingOffset(0), paintingOffset(1)) = slice
 paintingOffset(0) = paintingOffset(0)*scaleX*magX
 paintingOffset(1) = paintingOffset(1)*scaleY*magY
 ...
 dimPaintingWithFrame = INTARR(2)
 dimPaintingWithFrame(0) = $
  ROUND((1+paintingFactor*2)*dimX*magX*scale*zoom(baseNr))
 dimPaintingWithFrame(1) = $
  ROUND((1+paintingFactor*2)*dimY*magY*scale*zoom(baseNr))
 painting = CONGRID(painting, dimPaintingWithFrame(0), $
  dimPaintingWithFrame(1), /INTERP)
 ...
 ;; PAINT CURSOR
 ...
 WIDGET_CONTROL, id.draw(baseNr, drawNr), GET_VALUE = index
 WSET, index
 TVSCL, painting,$
  (scalingOffset(0)*(paintingFactor*2+1))-paintingOffset(0)+$
   offsetX*zoom(baseNr)*magX*dimX*scale,$
  (scalingOffset(1)*(paintingFactor*2+1))-paintingOffset(1)+$
   offsetY*zoom(baseNr)*magY*dimY*scale
END
```

3.4.2 3D Projection

The 3D projection is done by `draw3D`. First the current draw widget is selected by `WIDGET_CONTROL` and `WSET`. Then the scale-factors `scaleXY`, `scaleZ` and `scale` are computed (as shown in Subsection 3.4.1). Then `scaleXY` and `scaleZ` are divided through `scale`.

The IDL procedure `T3D` is used to set up the right scaling for the draw widget. In order to fit the image smoothly into the draw widget, the system arrays `!X.S`, `!Y.S` and `!Z.S` are set up as shown in lines 16-18. `TRANSLATE=[-0.5,-0.5,-0.5]` translates the middle of the image to the origin. By `ROTATE` the image is rotated around the origin. After rotating the image, it is translated to the user defined offset position (`offset3D`). `SCALE` rectifies the image. With the help of `TRANSLATE=[0.5,0.5,0.5]` the picture is moved to its final position. `VOXEL_PROJ` now calculates the projected image, which is displayed by `TVSCL`.

```
PRO draw3D, baseNr, data, frame
 ...
 IF ~scaledDrawIsCached(baseNr, 1) THEN $
  scaleDrawCache, baseNr, 1, data.cache(*,*,*,frame)
 ...
 WIDGET_CONTROL, id.draw(baseNr,3), GET_VALUE = index
 WSET, index
 ...
 scaleXY = 1.*windowSize.xy/(data.dim(0)*data.magnification(0))
 scaleZ = 1.*windowSize.z/(data.dim(1)*data.magnification(1))
 scale = MIN([scaleXY, scaleZ])
 scaleXY = scaleXY/scale
 scaleZ = scaleZ/scale
 ...
 T3D, /RESET
 !X.S = [0, 1.]/(data.dim(0)-1)
 !Y.S = [0, 1.]/(data.dim(0)-1)
 !Z.S = [0, 1.]/(data.dim(1)-1)
 T3D, TRANSLATE=[-0.5,-0.5,-0.5]
 T3D, ROTATE=[90,180+angle3D(baseNr),180]
 T3D, TRANSLATE=[offset3D(baseNr,0),offset3D(baseNr,1), 0]
 T3D, SCALE=zoom3D(baseNr)*[1./scaleXY,1./scaleZ,1.]
 T3D, TRANSLATE=[0.5,0.5,0.5]
 TVSCL, VOXEL_PROJ(BYTE(255* *scaledDrawData(baseNr, 1)), $
  /MAXIMUM_INTENSITY, STEP=[1.,1.,1.]*quality, /INTERPOLATE)
END
```

4 Software Evaluation

In this Chapter the results of the mouse fitting software are evaluated. First all measurement devices and the used software will be presented in Section 4.1. Then the evaluation results will be presented in Section 4.2 and afterwards in Chapter 5 the outcome will be interpreted.

4.1 Materials and Methods

In this Section the used PET instrument, the evaluation software and the animals used for measurement are discussed.

4.1.1 Positron Emission Tomograph Used

For measurements a Siemens microPET Focus 220 scanner[26] (shown in Fig. 4.1) is used. This is a dedicated small animal PET system with a resolution of 1.3 mm FWHM (Full Width at Half Maximum) and an absolute sensitivity (is a measure for the system's aperture and represents the fraction of radioactive decays that produce a valid coincidence event[17]) of 3.4 % in the central field of view for an energy window of 250-750 keV and a timing window of 10 ns. The scanner acquires data in three-dimensional list mode format (explained in Section 2.1). Image Reconstruction is done by a dual core Pentium Xenon computer with 2 GB main memory. The reconstruction software is provided by Siemens and offers different reconstruction algorithms such as FBP, OSEM, MAP (see Section 2.5).

Because laboratory animals must not move during data acquisition, the mice are scanned under constant isofluorane anesthesia in a temperature stabilized imaging chamber (as shown in Fig. 4.2), which prevents the animals from cooling down in the air-conditioned laboratory.

4.1.2 Evaluation Software

For software evaluation the open source software Amide[27] was used. With the help of this software ROIs can be selected using simple geometric forms like cuboids, ellipsoids and elliptical cylinders or 2D- and 3D-isocontours, which will add voxels to the ROI, if the voxel values lie in between userdefined values. This allows to easily define ROIs. The major drawback of this computer program is, that regions without sharp edges only can be hardly and imprecisely selected. Therefore only well definable regions will be used to grant a more precise evaluation.

4.1.3 Animals Used

About twenty mouse data sets recorded within August 2005 through to April 2007 were used to evaluate the computer program developed in this thesis. Some mice originate from different Transgenic (TG) breeds, the others are Wild Type (WT) mice used for relation.

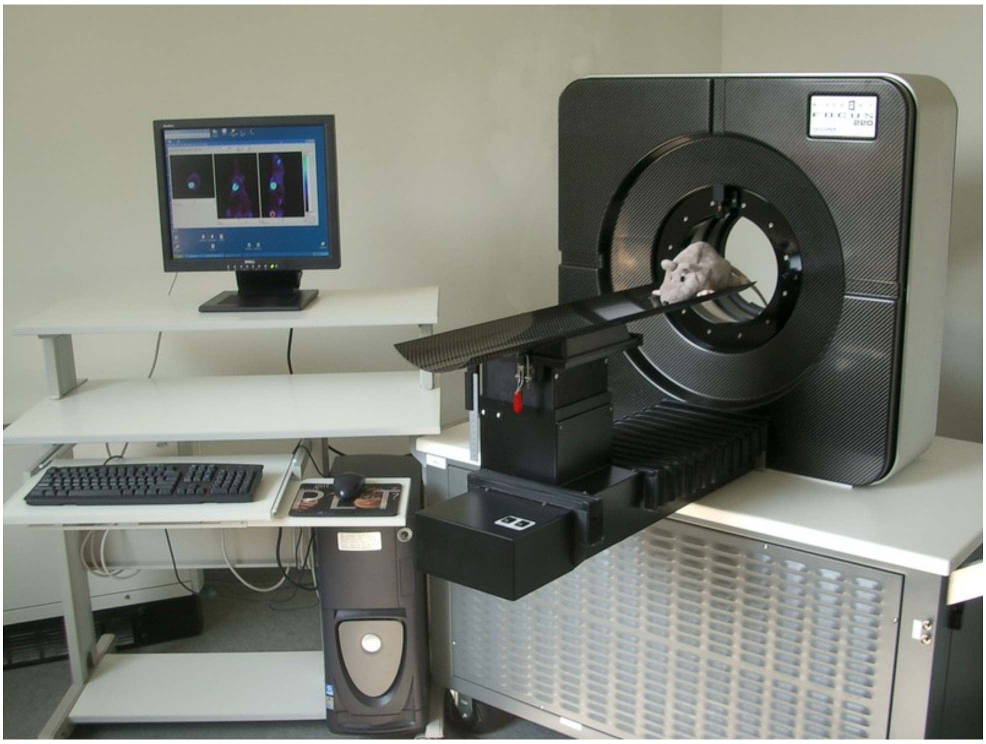


Figure 4.1 Siemens microPET Focus 220 scanner

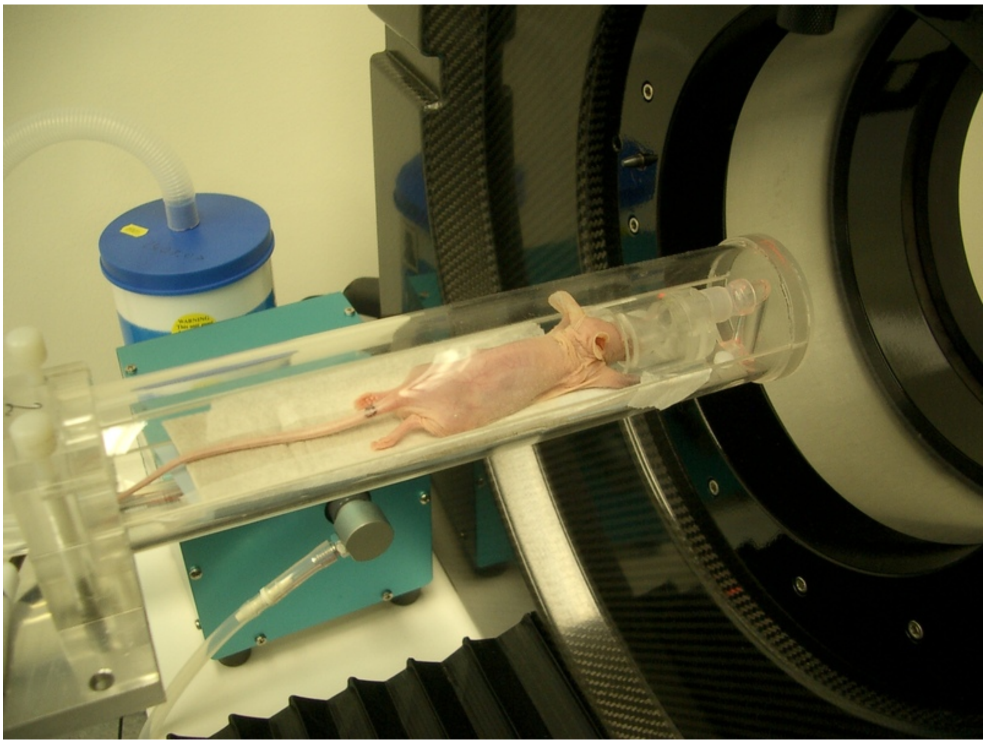


Figure 4.2 Mouse chamber used for measurements

4.2 Results

The presentation and evaluation of the measurement results is subdivided into three parts. First in “Correlation” the software results for user selected organs will be compared to the results provided by Amide. Second the reliability of organ acitvity calculated by the computer program is presented in “Indirect Correlation”. Third in “Reproducibility” the deviation of selecting the same organ multiple times with the software, developed in this thesis, and Amide is shown.

4.2.1 Correlation

^{18}F-Fluoride enriches in bones and is excreeted over kidneys and bladder. It is used to show bone metabolism. Therefore the time dependent activity of bladder and spine of dynamically reconstructed data sets of three mice measured with MF2007 ("Mouse Fitter 2007" - software developed in this thesis) and Amide are compared to each other. In order to get a quick overview on correlation Figs. 4.3 through 4.8 are shown in XY-plot. The X-axis represents the weighted activity in %ID/g, which is explained below, measured by MF2007 and the Y-axis the weighted activity measured by hand selection with Amide. The closer the plotted values lie to function "x=y", the more they correlate.

%ID/g is the mean value of all organ voxels divided through the injected dose. Fig. 4.3 shows the measured activity of fluoride in the bladder for all three mice (M1, M2, M3). The number of mice used for measurement is shown in the caption of each Figure (i.e. n=3). For both M1 and M2, the activity calculated by Amide is greater than the activity computed by MF2007, whereas for mouse M3 there is perfect correlation.

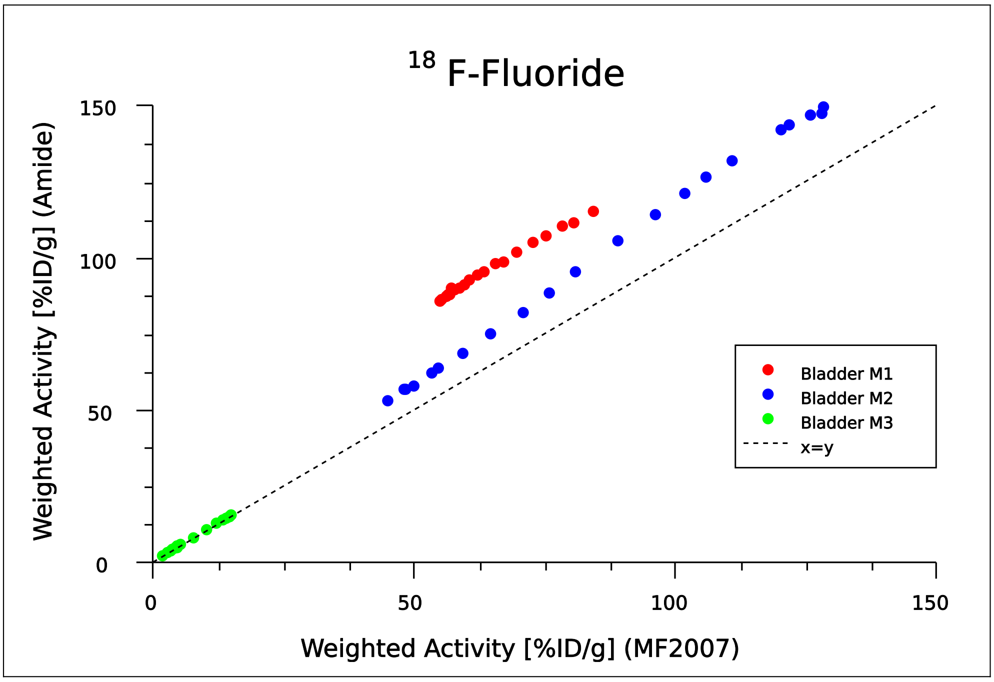


Figure 4.3 Correlation between the manually and MF2007 defined bladder ROI in ^{18}F-Fluoride microPET images (n=3)

Fig. 4.4 shows the measured activity of fluoride in the spine for all three mice. It shows the same behavior for all three mice and nearly perfect correlation, but with increasing activity the correlation gets weaker.

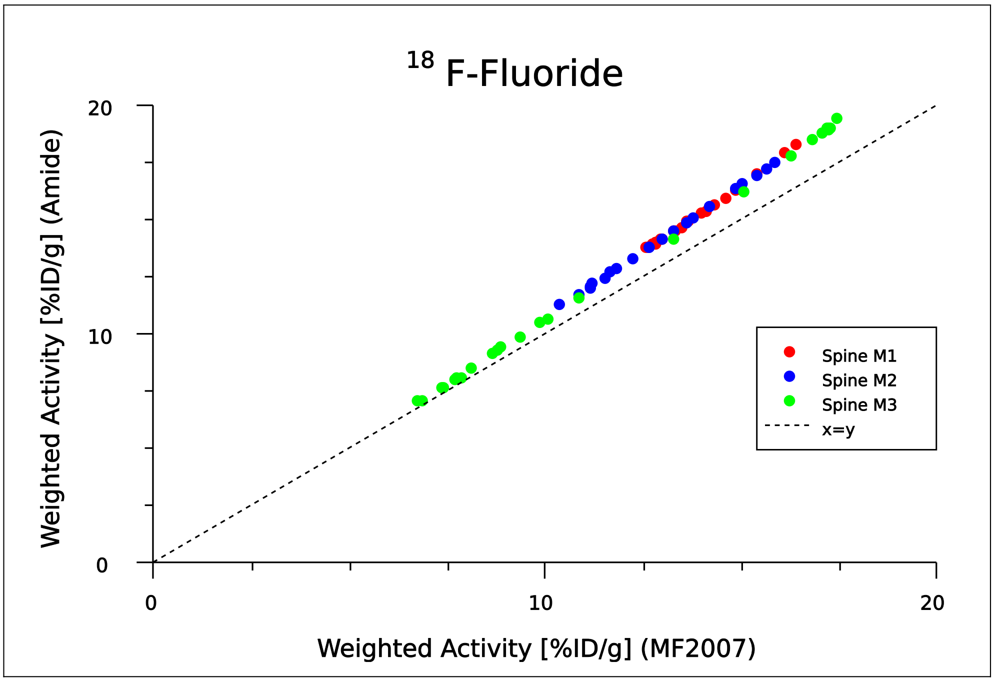


Figure 4.4 Correlation between the manually and MF2007 defined spine ROI in ^{18}F-Fluoride microPET images (n=3)

The activity in the spine of one mouse with a ^{68}Ga-EDTMP injection is illustrated in Fig. 4.5 and shows nearly perfect correlation. ^{68}Ga-EDTMP is also used to gain information about bone metabolism.

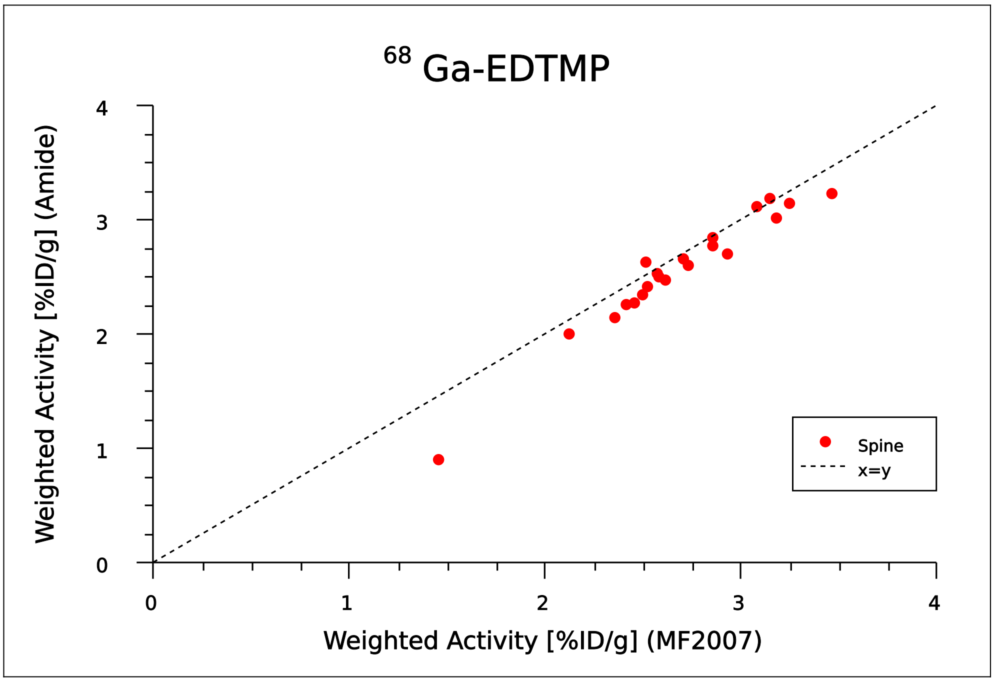


Figure 4.5 Correlation between the manually and MF2007 defined spine ROI in ^{68}Ga-EDTMP microPET images (n=1)

Sodium Iodide (NaI) enriches in thyroid and stomach. Fig. 4.6 shows perfect correlation for the thyroid region. For the stomach region the activity calculated by Amide increases compared to MF2007 for higher values. It is the same effect as for ^{18}F-Fluoride shown before.

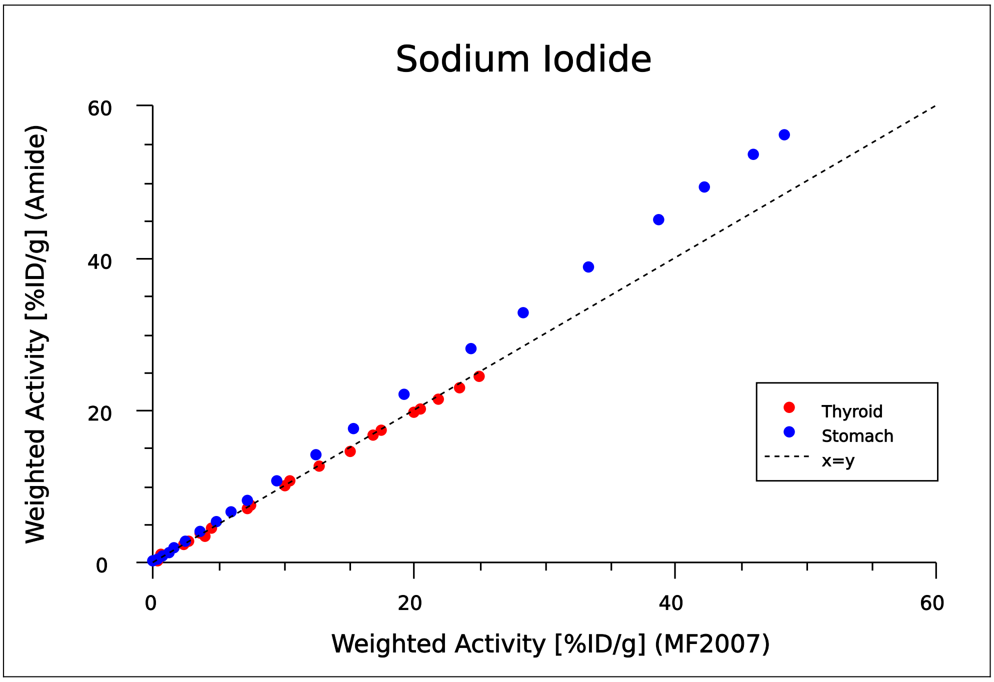


Figure 4.6 Correlation between the manually and MF2007 defined ROIs (thyroid and stomach) in ^{124}I-NaI microPET images (n=1)

^{18}F-FDG, which is one of the most frequently used tracers in PET, accumulates in heart and brain and is excreted over the kidneys. Three mice were evaluated. The resulting comparison diagram is presented in Fig. 4.7.

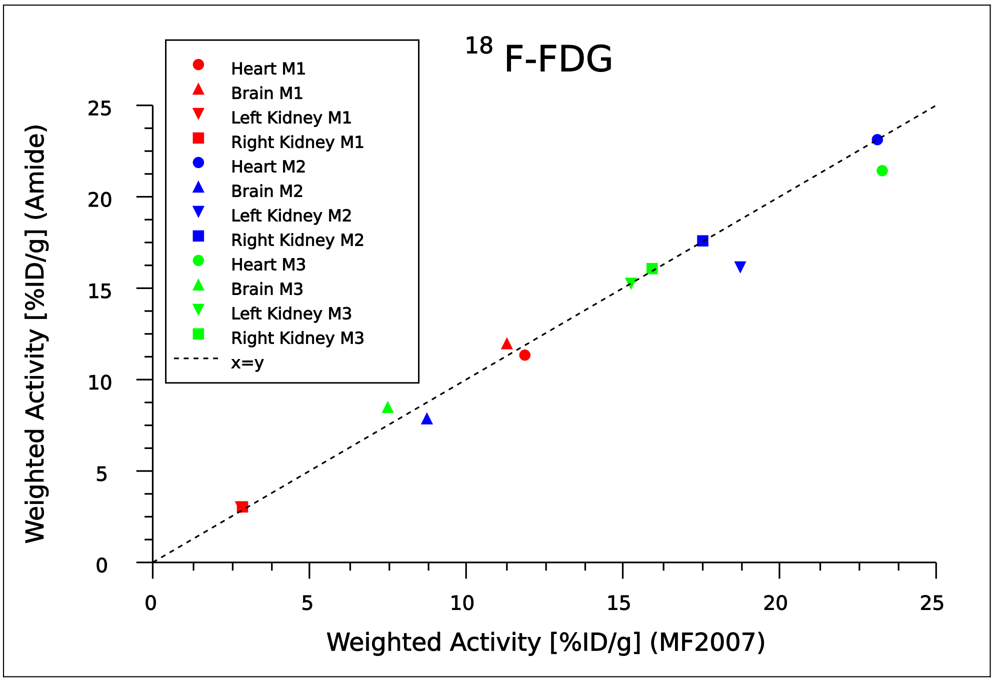


Figure 4.7 Correlation between the manually and MF2007 defined ROIs in ^{18}F-FDG microPET images (n=3)

4.2.3 Indirect Correlation

Due to the knowledge of the location of user defined organs the developed MF2007 calculates the position of the other organs. In this Subsection the degree of reliance of the calculated activity of the reconstructed organs is evaluated. Therefore the left and the right kidney were selected by the user and the remaining organs were computed by MF2007. For comparison the mean activities of heart, kidneys and brain were calculated with Amide. The results of the indirect correlation are shown in Figs. 4.8 and the respective TACs are displayed in 4.9 and 4.10. The results for the brain and the kidneys are equal for both Amide and MF2007. The activity in the heart region diverges from perfect correlation.

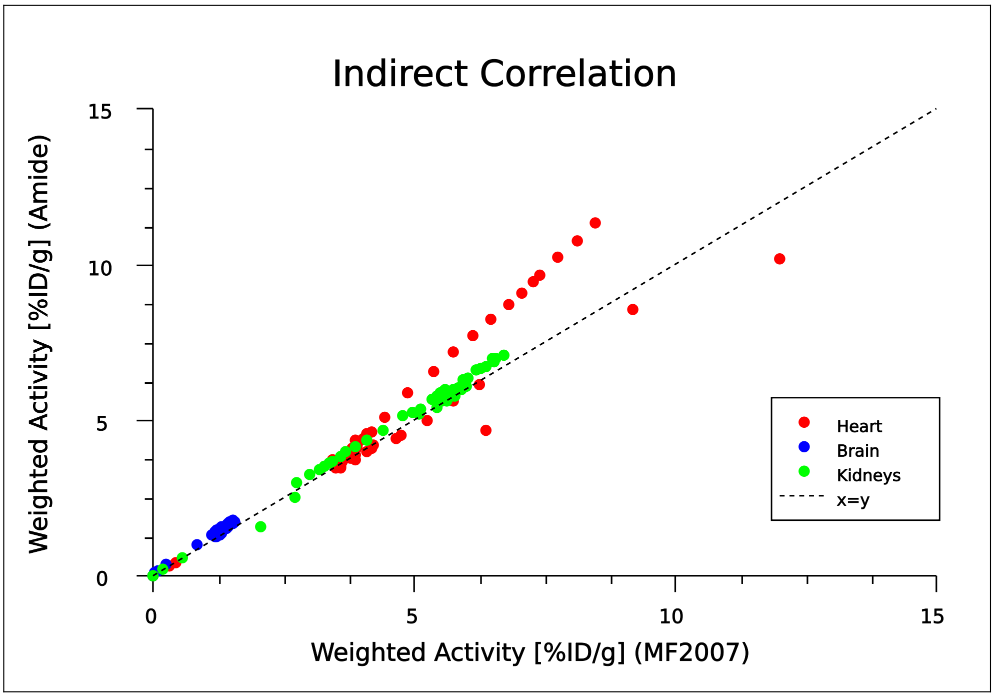


Figure 4.8 Indirect correlation between the manually and reconstructed organs calculated by MF2007 in ^{18}F-FDG microPET images (n=1)

In Figs. 4.9 and 4.10 we can see the perfusion peak in the heart's TAC. While the activity curves for the brain and the kidneys do not exhibit major differences, the manualy defined heart's TAC tends to be higher than the MF2007-TAC.

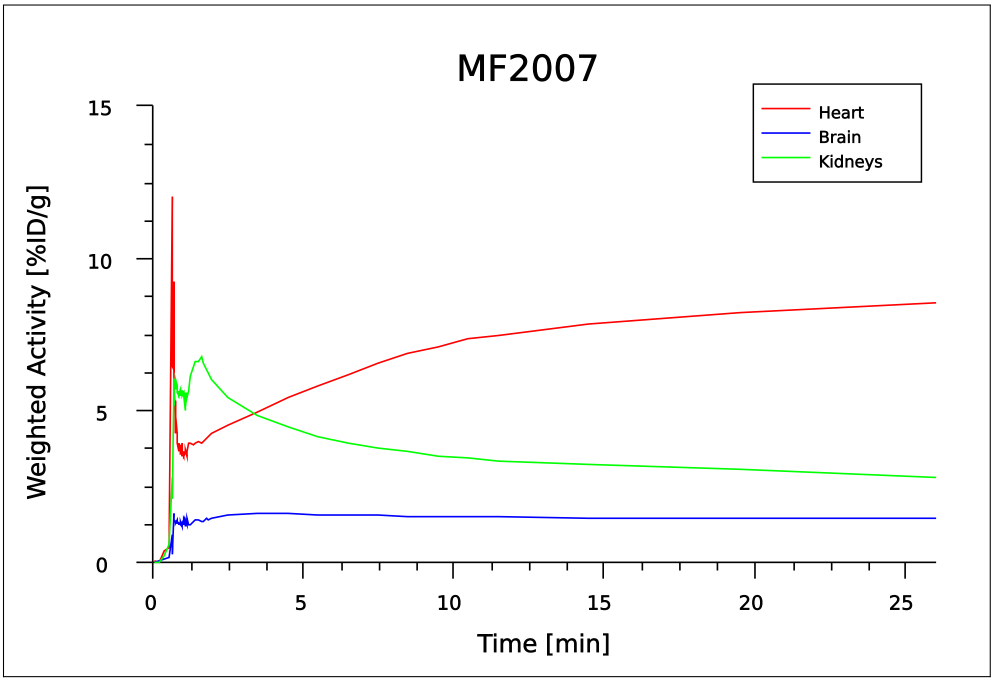


Figure 4.9 TACs calculated by MF2007 in ^{18}F-FDG microPET images (n=1)

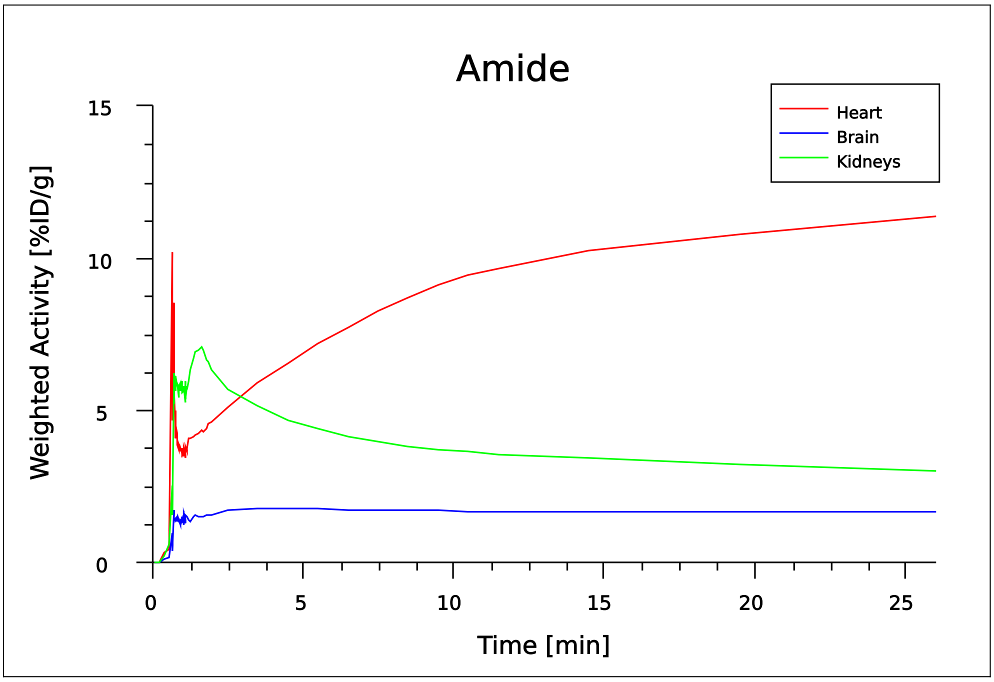


Figure 4.10 TACs calculated by Amide in ^{18}F-FDG microPET images (n=1)

If there is little contrast between the selected organ and its surroundings, regions defined by hand selection are generally inaccurate. Particularly brain regions are elaborate and improper to select. MF2007 offers a better alternative for defining ROIs. In Fig. 4.11 and Fig. 4.12 the uptake of the cerebellum, the hippocampus and the frontal cortex of ^{18}F-FDDNP is illustrated. The mean values of six TG and five WT mice are compared to each other. The TACs evaluated by hand (Amide) exhibit much more spread than the TACs calculated by MF2007.

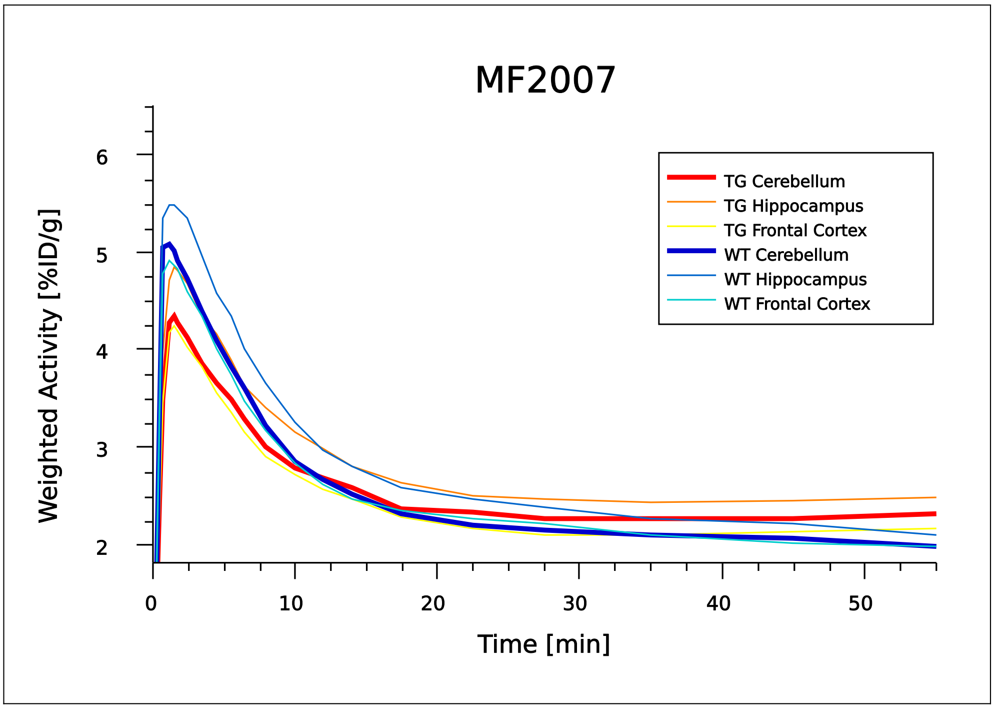


Figure 4.11 TACs calculated by MF2007 in ^{18}F-FDDNP microPET images (n(TG)=6, n(WT)=5)

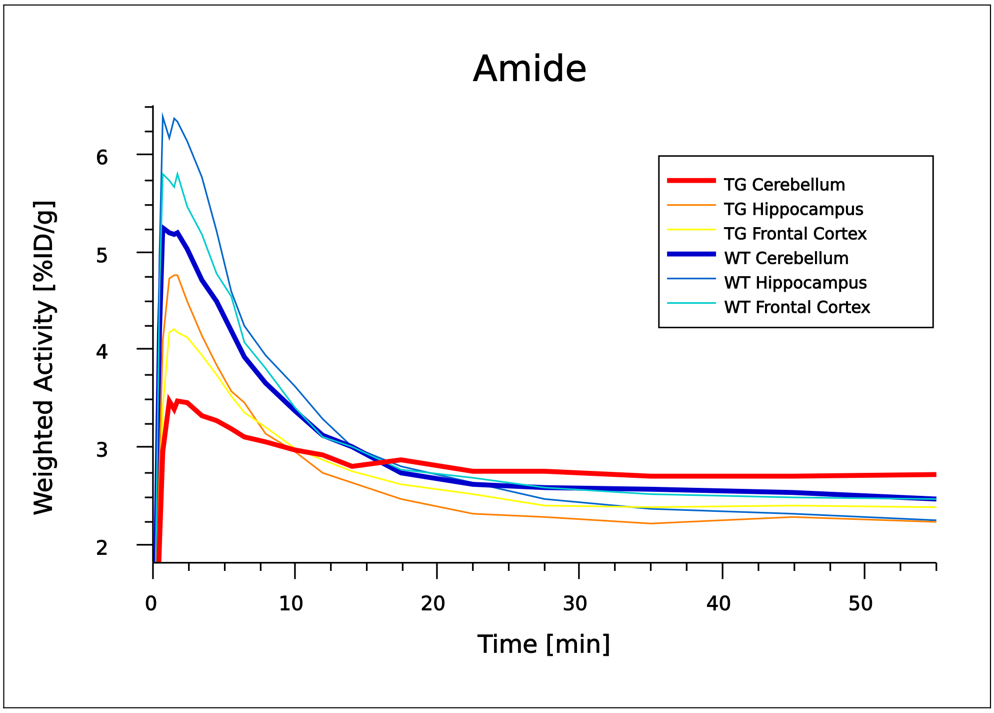


Figure 4.12 TACs calculated by Amide in ^{18}F-FDDNP microPET images (n(TG)=6, n(WT)=5)

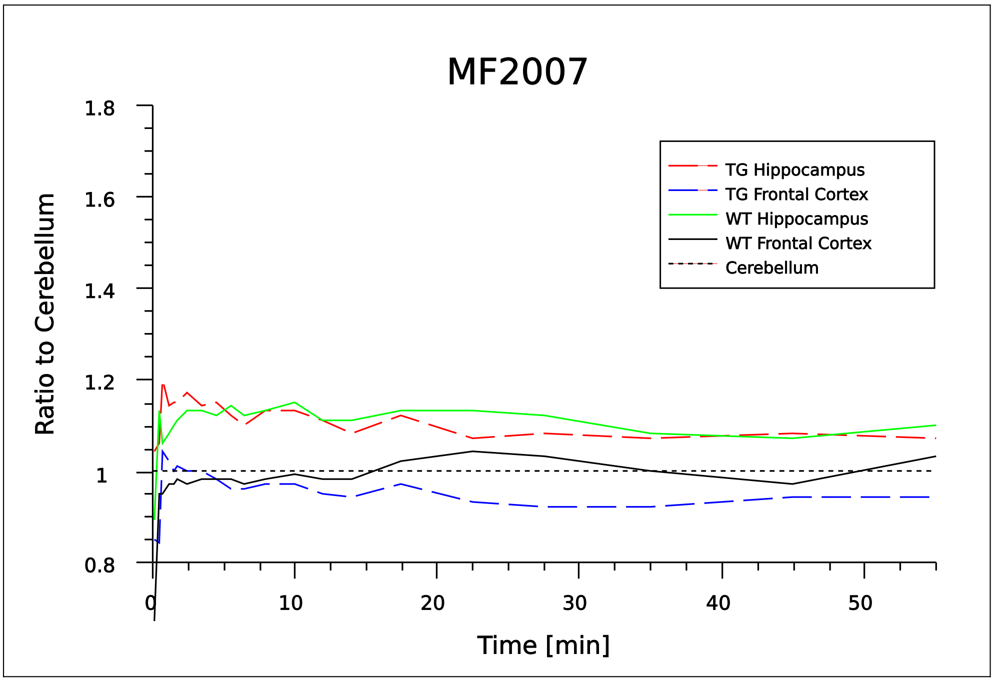


Figure 4.13 TACs (in ratio to cerebellum) calculated by MF2007 in ^{18}F-FDDNP microPET images (n(TG)=6, n(WT)=5)

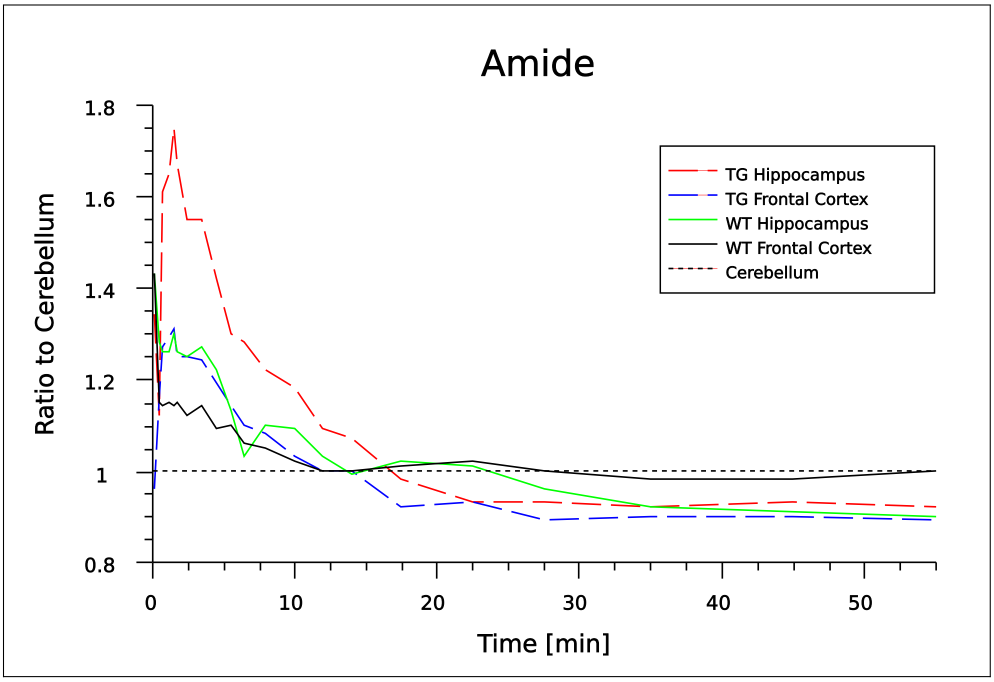


Figure 4.14 TACs (in ratio to cerebellum) calculated by Amide in ^{18}F-FDDNP microPET images (n(TG)=6, n(WT)=5)

If the activity ratios between the brain regions to the cerebellum region calculated by Amide and MF2007 are compared to each other, the ratio calculated by MF2007 almost level off (shown in Fig. 4.13), whereas the ratio of Amide is subject to severe fluctuations (Fig. 4.14).

4.2.2 Reproducibility

In order to evaluate the reproducibility, when defining ROIs by hand (Amide) or with the help of MF2007, the same organ of one mouse is selected independently several times. The resulting Standard Deviation (SD) is used as a quality parameter. The smaller the degree of deviation the more stable is the method for defining the organs. Fig. 4.15 shows, that MF2007's SDs as percentage of mean activity are clearly smaller than those determined by hand evaluation.

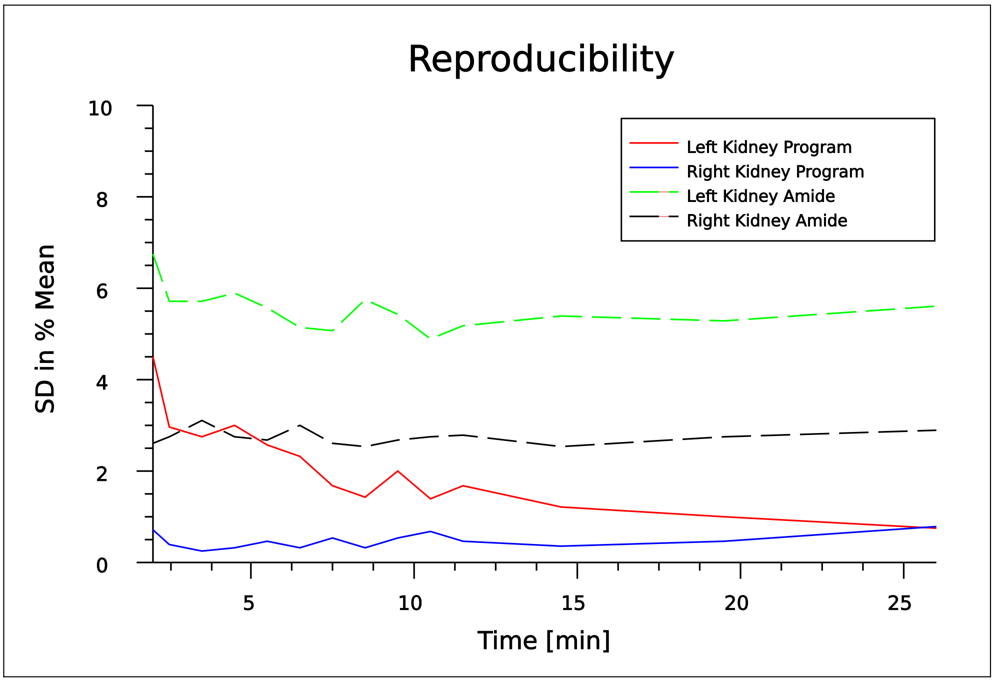


Figure 4.15 Reproducibility of defining a ROI by hand or with the help of MF2007

5 Measurement Discussion

5.1 Discussion on Correlation

Fig. 4.3 shows, that the activity calculated by Amide is generally higher than for regions selected by MF2007. This characteristics occurs, because the computed activity matches the mean value of all selected voxels. In case the bladder has to be selected, the activity decreases from its center like a Gauss error distribution curve. The selected volume increases by r^3. The outer layers mostly influence the volume and therefore the number of the selected voxels. Because the activity decreases rapidly with increasing distance from the bladders center, the mean voxel value highly depends on the radius of the selected region. Since PET images do not show sharp edges it is difficult to decide, where the bladder really ends. The smaller the selected region is (with its center round the highest activity voxels), the disproportionately higher the resulting mean values are. When selecting regions manually, one tends to select undersized regions, thus resulting in higher mean activity. The higher the mean voxels value gets, the higher this discrepancy will be (shown in Figs. 4.3, 4.4 and 4.6). If the ROIs have been selected to large, MF2007 points to the fact, that there is a mismatch between the distance and size of organs.

5.2 Discussion on Indirect Correlation

Fig. 4.8 illustrates, that the reconstructed organs like brain and kidneys equal the regions manually selected, but the values for the heart show a great discrepancy. This is due to the fact, that the heart, which is provided by the mouse phantom, is somehow illshaped. It was suprising, that although the distance from the brain to the COM is large, such a good correlation could be achieved.

The reconstructed organs rely on the distance to the COM of the user defined organs. The farther the organs lie away from the COM the greater the discrepancies are. If the mouse has a crooked back, this cannot be taken into account, and the position of a certain organ might fail. In small regions the reconstructions used in Figs. 4.13 and 4.14 point out, that the manual selection of brain regions is to imprecise and shows a strange behaviour, whereas the ROI reconstructed by MF2007 present a constant ratio.

5.3 Discussion on Reproducibility

The reproducibility (shown in Fig. 4.15) for organs selected by MF2007 is significantly higher than defining the same ROI by hand. This is only guaranteed, if the ROI matches the dimensions of the mouse phantom. Organ TACs as for instance for the bladder, which change size or location, are subject to broader fluctuations.

6 Conclusion & Outlook

Summarizing it can be concluded, that the software developed in this thesis offers a faster as well as a more precise method for evaluating the activity in desired regions, reducing subjective factors. The ROIs can be selected more accurately and more user-independently compared to manual selection. ROIs can be selected more easily, thus saving time.

Due to the software's modular structure the development of this computer program can be continued and new features can be implemented, eg. in order to provide a better overlap between the reconstructed and the PET file ROIs, the ROIs could be further rotated around their own COM or their shape can be altered.

Additionally the program can easily be adopted to support different kinds of anatomical phantoms (eg. for a rat phantom). For that purpose a binary rat phantom with discrete organ values must be available. The software's code has to be adopted to the rat phantoms dimensions and the discrete values assigned to each organ must be changed. This only requires about a dozen lines of code to be modified.

A References

[1] Adam L. Kesner, Magnus Dahlbom, Sung-Cheng Huang, Wei-Ann Hsueh, Betty S. Pio, Johannes Czernin, Michael Kreissl, Hsiao-Ming Wu and Daniel H. S. Silverman. Semiautomated analysis of small-animal PET data. *Journal of Nuclear Medicine*, 47:1181-1186, 2006.

[2] William P. Segars, Benjamin M. W. Tsui, Eric C. Frey, G. Allan Johnson and Stuart S. Berr. Development of a 4-D digital mouse phantom for molecular imaging research. *Molecular Imaging and Biology*, 6(3):149-159, 2004.

[3] http://de.wikipedia.org/wiki/Bild:PET-schema.png.

[4] http://de.wikipedia.org/wiki/Bild:ECAT-Exact-HR--PET-Scanner.jpg.

[5] Robert R. Kinsey. The NUDAT/PCNUDAT program for nuclear data. 1996 Data extracted from NUDAT database (Jan. 14/1999).

[6] Simon R. Cherry, James A. Sorenson and Michael E. Phelps. *Physics in Nuclear Medicine*. Saunders, third edition, 2003.

[7] http://de.wikipedia.org/wiki/Bild:Zyklotron_Schema.gif.

[8] http://en.wikipedia.org/wiki/Image:Ear.png.

[9] http://de.wikipedia.org/wiki/Bild:Sn1_Substitution.svg.

[10] Claudia Kuntner. *Evaluation of New Inorganic Scintillators for Application in a Prototype Small Animal PET Scanner*. PhD thesis, TU Vienna, 2003.

[11] http://en.wikipedia.org/wiki/Image:Photomultipliertube.svg.

[12] http://en.wikipedia.org/wiki/Image:PET-detectorsystem_2.png.

[13] http://de.wikipedia.org/wiki/Bild:Trues.jpg.

[14] http://de.wikipedia.org/wiki/Bild:Randoms.jpg.

[15] http://de.wikipedia.org/wiki/Bild:Singles.jpg.

[16] http://de.wikipedia.org/wiki/Bild:Scatter.jpg.

[17] Michael E. Phelps. *PET - Molecular Imaging and Its Biological Applications*. Springer, 2004.

[18] http://de.wikipedia.org/wiki/Bild:PET_3D_mode.jpg.

[19] Patrick L. Chow, Fernando R. Rannou and Arion F. Chatziioannou. Attenuation correction for small animal PET tomographs. *Physics in Medicine and Biologie*, 50(8):1837-1850, 2005.

[20] J. Radon. Über die Bestimmung von Funktionen längs gewisser Mannigfaltigkeiten. *Berichte der mathematisch-physikalischen Kl. der Sächsischen Gesellschaft der Wissenschaften*, 59, 1917.

[21] Thorsten M. Buzug. *Einführung in die Computertomographie – Mathematisch-physikalische Grundlagen der Bildrekonstruktion*. Springer, 2005.

[22] N. Gurker, R. Nell, G. Seiler and J. Wallner. A tunable focusing beamline for desktop x-ray microtomography. *Review of Scientific Instruments*, 70(7):2935-2949, 1999.

[23] Philippe P. Bruyant. Analytic and iterative reconstruction algorithms in SPECT. *Journal of Nuclear Medicine*, 43:1343-1358, 2002.

[24] A. P. Dempster, N. M. Laird and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39(1):1-38, 1977.

[25] H. Malcolm Hudson and Richard S. Larkin. Accelerated image reconstruction using ordered subsets of projection data. *IEEE Transactions on Medical Imaging*, 13(4):601-609, 1994.

[26] Yuan-Chuan Tai, Ananya Ruangma, Douglas Rowland, Stefan Siegel, Danny F. Newport, Patrick L. Chow and Richard Laforest. Performance evaluation of the micropet focus: A third-generation micropet scanner dedicated to animal imaging. *Journal of Nuclear Medicine*, 46:455-463, 2005.

[27] http://amide.sourceforge.net/.

B Abbreviations

2D	two-dimensional
3D	three-dimensional
ARC	Austrian Research Centers
ART	Algebraic Reconstruction Technique
ASCII	American Standard Code for Information Interchange
BP	Back Projection
COM	Center of Mass
CT	Computer Tomography
EM	Expectation Maximization
FBP	Filtered Back Projection
FDG	^{18}F-Fluor-Deoxyglucose
FWHM	Full Width at Half Maximum
GUI	Graphical User Interface
IDL	Interactive Data Language
LOR	Line of Response
MAP	Maximum a Posteriori
MF2007	"Mouse Fitter 2007" - software developed in this thesis
MLEM	Maximum Likelihood Expectation Maximization
MRT	Magnetic Resonance Tomography
NC	Normalisation Coefficient
OSEM	Ordered Subset Expectation Maximization
PET	Positron Emission Tomography
PMT	Photomultiplier Tube
ROI	Region of Interest
SD	Standard Deviation
SPECT	Single Photon Emission Computed Tomography
SUV	Standardized Uptake Value
TAC	Time Activity Curve
TG	Transgenic
WT	Wild Type

C Figures

2.1 Scheme of Positron Emission Tomography[3] 9

2.2 Human PET facility[4] 10

2.3 Small animal PET facility in ARC (Austrian Research Centers), Seibersdorf . 11

2.4 Illustration of a Cyclotron 12

2.5 Pictures of the synthesis unit and hot laboratory facilities at ARC, Seibersdorf 12

2.6 Electrophile addition[8] 13

2.7 Nucleophile substitution[9] 13

2.8 Scintillating crystal coupled to a PMT[11] 14

2.9 PET detector system[12] 14

2.10 Different types of coincidences[13, 14, 15, 16] 15

2.11 Different sensitivities due to geometric effects in 3D mode[18] 16

2.12 Illustrating the effect of normalization 16

2.13 Attenuation correction for cylinder filled with 500 ml ^{18}F-water solution 17

2.14 Illustrating the use of filters in FBP (mircoCT image of a nut[22]) 18

2.15 Illustration of the ART algorithm 19

3.1 Program's GUI 21

3.2 Functional principle of the fitting process 25

3.3 Defining Organs 26

3.4 Extract Phantom Organs 27

3.5 Resizing Organs 29

3.6 Rescaling Distancies 31

3.7 Determine Angles 32

3.8 The wost-case scenario for function rotateMask 33

3.9 Affiliate Organs 35

3.10 Illustration of transforming the phantom organ masks to globally used organ masks 36

3.11 Time Activity Curves 39

3.12 Illustration of `drawSlice2D` 44

4.1 Siemens microPET Focus 220 scanner 48

4.2 Mouse chamber used for measurements 48

4.3 Correlation between the manually and MF2007 defined bladder ROI in ^{18}F-Fluoride microPET images (n=3) 50

4.4 Correlation between the manually and MF2007 defined spine ROI in ^{18}F-Fluoride microPET images (n=3) 51

4.5 Correlation between the manually and MF2007 defined spine ROI in ^{68}Ga-EDTMP microPET images (n=1) 51

4.6 Correlation between the manually and MF2007 defined ROIs (thyroid and stomach) in ^{124}I-NaI microPET images (n=1) 52

4.7 Correlation between the manually and MF2007 defined ROIs in ^{18}F-FDG microPET images (n=3) 52

4.8 Indirect correlation between the manually and reconstructed organs calculated by MF2007 in ^{18}F-FDG microPET images (n=1) 53

4.9 TACs calculated by MF2007 in ^{18}F-FDG microPET images (n=1) 54

4.10 TACs calculated by Amide in ^{18}F-FDG microPET images (n=1) 54

4.11 TACs calculated by MF2007 in ^{18}F-FDDNP microPET images (n(TG)=6, n(WT)=5) 55

4.12 TACs calculated by Amide in ^{18}F-FDDNP microPET images (n(TG)=6, n(WT)=5) 55

4.13 TACs (in ratio to cerebellum) calculated by MF2007 in ^{18}F-FDDNP microPET images (n(TG)=6, n(WT)=5) 56

4.14 TACs (in ratio to cerebellum) calculated by Amide in ^{18}F-FDDNP microPET images (n(TG)=6, n(WT)=5) 56

4.15 Reproducibility of defining a ROI by hand or with the help of MF2007 57

D Tables

2.1 Examples of radioactive tracers[5] 11
3.1 Some organs and values defined by phantom file 28
3.2 Additionally defined organgroups 28
3.3 Legend of organ.defined-flags 35

Acknowledgements

First of all, I want to thank everyone assisting me in writing my diploma thesis. In particular DI. Dr. Claudia Kuntner for supervising me, for giving me the opportunity to work in her group at Seibersdorf, for reviewing my thesis and for helping me designing an awarded poster. I also want to thank Univ.-Prof. DI. Dr. Norbert Gurker, who assisted me in writing my thesis and provided me a contact point for all my open questions. Furthermore I want to thank Ing. Thomas Wanek, Mag. Dr. Oliver Langer and the entire team of “Radiation Safety and Applications in Seibersdorf” for supporting me.

Special thanks are due to my parents, Eva and Reinhard, who offered me the opportunity to study physics, my siblings, Daniel and Ute, and Sabine for their social backup.