4-2016

Registration and Segmentation of Multimodality Images for Post Processing of Skeleton in Preclinical Oncology Studies

Vineeth Radhakrishnan
Grand Valley State University

Registration and segmentation of multimodality images for post processing of skeleton in preclinical oncology studies

Vineeth Radhakrishnan

A Thesis Submitted to the Graduate Faculty of

GRAND VALLEY STATE UNIVERSITY

In

Partial Fulfillment of the Requirements

For the Degree of

Master of Science in Engineering

Biomedical Engineering
School of Engineering

April 2016

Dedications

To my parents for the love, care and support in all my endeavors. I appreciate all the sacrifices they made to put me through the best education possible.

To my mentor Dr. Rajesh Kannan Megalingam, who set a new standard for teaching through his passion to nurture young talent and bring out the best in his students. He inspired me to pursue Masters and to set higher goals in life. He taught me through his life that success comes through hard work and perseverance.

Acknowledgements

I would like to thank Dr. Rhodes, for all the support and advice throughout the course of my thesis work. Thanks for providing me the opportunity to work at Van Andel Institute and apply the theoretical knowledge I learnt in your classes to solve real world problems. I would like to thank Dr. Jeffrey Ward and Dr. Bruce Dunne in providing valuable feedback for my work.

I would like to thank Dr. Anthony Chang and all the members of Small Animal Imaging Facility (SAIF) at Van Andel Institute (VAI) for introducing me to the world of molecular imaging and providing me all the access to the infrastructure in order to carry out my research work.

I would like to thank my family, most notably my wife Neethu, who supported me throughout the graduate work with unlimited love and encouragement.

I would like to thank my friends, Pradeep, Deepak and Nanda Kumar for the support and friendship during my study at Grand Valley State University.

I am grateful to God for everything that I have in my life and giving me the strength to complete this work.

Abstract

Advancements in medical imaging techniques provide biomedical researchers with quality anatomical and functional information inside preclinical subjects in the fields of cancer, osteopathic, cardiovascular, and neurodegenerative research. The throughput of the preclinical imaging studies is a critical factor which determines the pace of small animal medical research. The time involved in manual analysis of large amount of imaging data prior to data interpretation by the researcher, limits the number of studies in a time frame.

In the proposed solution, an automated image segmentation method was used to segment individual vertebrae in mice. Individual vertebrae of MOBY atlas were manually segmented and registered to the CT data. The PET activity for L1-L5 vertebrae was measured by applying the CT registered atlas vertebrae ROI.

The algorithm was tested on three datasets from a PET/CT bone metastasis study using ^{18}F-NaF radiotracer. The algorithm was found to reduce the analysis time threefold with a potential to further reduce the automated analysis time by use of computer system with better specification to run the algorithm. The manual analysis value can vary each time the analysis is performed and is dependent on the individual performing the analysis. Also the error percent was recorded and showed an increasing trend as the analysis moves down the spine from skull to caudal vertebrae. This method can be applied to segment the rest of the bone in the CT data and act as the starting point for the registration of the soft tissues.

Table of Contents

1. Introduction 10
   1.1. Research Objective 11
2. Literature Review 14
   2.1. Imaging Modalities 14
      2.1.1. Computed Tomography (CT) 14
      2.1.2. Positron Emission Tomography (PET) 15
   2.2. Image registration 16
   2.3. Image registration algorithms 16
      2.3.2. Surface based registration 17
      2.3.3. Intensity based registration techniques: 19
   2.4. Atlases in medical imaging research 19
   2.5. Related Work 22
3. Methodology 23
   3.1. Marching cube algorithm 27
   3.2. Iterative Closest Point Algorithm 33
      3.2.1. Point selection 34
      3.2.2. Point matching 35
      3.2.3. Error metric and minimization of error metric 37
   3.3. Manual method using MIPAV 40
4. Results 41
5. Discussion 49
6. Future Work 53
7. Appendices 54
   7.1. Software Code: 54
8. References 85

List of Figures

Figure 3-1 Generalized block diagram for the algorithm. The segmented atlas is loaded and joints are defined as either ball and socket or hinge joints.. 23

Figure 3-2 Figure showing the various anatomical joints and the joint type on the Moby. Ball and socket joints are identified as B and hinge joints as H in the Figure on the left. 24

Figure 3-3 Figure showing the atlas regions for automatic segmentation. 25

Figure 3-4 Figure showing the application of the atlas segmentation hierarchy. 26

Figure 3-5 Figure showing the registration of the point cloud and the data cloud using the standard ICP algorithm. 27

Figure 3-6 Triangulation cubes in marching cube algorithm (Lorensen and Cline 1987)..... 29

Figure 3-7 Figure showing the continuous surface rendering on both cubes by sharing the vertices 1, 2, 3 and 4. 30

Figure 3-8 Figure showing the surface rendered μCT image of the mice using marching cube algorithm. 30

Figure 3-9 Figure showing the registration of the μCT mesh and the un-segmented atlas spine.32

Figure 3-10 Figure showing the skeletal anatomy of LAC grey mouse illustrating the number of vertebrae of each category 32

Figure 3-11 Figure showing the manually segmented atlas spine showing 7 Cervical, 13 Thoracic, 6 Lumbar, 4 Sacral and 5 Caudal vertebrae. 33

Figure 3-12 Figure showing the alignment of the CT and atlas after applying the initial input conditions. 34

Figure 3-13 Figure showing K-DTree formation to find the nearest neighbor for the data point cloud Di in the model point cloud Mi. 36

Figure 3-14. Figure showing (a) the registered skull atlas in 3D coordinate format (b) the volumetric data of the registered atlas skull. 39

Figure 3-15. Figure showing the settings for the PET/CT registration in MIPAV tool for PET radiotracer activity using manual method. 39

Figure 4-1 Figure showing the atlas prior to registration (on the left) and the atlas image after registration on the μCT image and segmentation (on the right). 41

Figure 4-2 Figure showing the registered atlas and μCT data at different azimuth angles. 42

Figure 4-3 Figure showing the CT image with manually drawn ROI in MIPAV. 42

Figure 4-4 Figure showing the PET image with manually drawn ROI copied from the CT image. 43

Figure 4-5 Figure showing the comparison of the PET activity measured by manual and automated method for the 1st dataset..... 44

Figure 4-6 Figure showing the comparison of the PET activity measured by manual and automated method for the 2nd dataset. 44

Figure 4-7 Figure showing the comparison of the PET activity measured by manual and automated method for the 3rd dataset. 45

Figure 4-8 Figure showing the error percent for all the three datasets for L1-L5 vertebrae 45

Figure 4-9 Figure showing the registered atlas vertebrae to the CT data for three different data sets..... 46

Figure 4-10 Figure showing the step by step registration of the atlas vertebrae onto the CT dataset. 46

Figure 4-11. Figure showing the saturation of error metric after certain number of iterations. ... 47

Figure 4-12. Figure showing the number of iterations before the error metric becomes significantly small. 47

Figure 4-13 Fluctuation in error after skull registration due to high number of iterations than required (Reqd Iteration:70, Set value:150)..... 48

List of Tables

Table 2-1 Table showing list of rat/mouse atlases used in pre-clinical studies 21

Table 4-1 Table showing the PET radiotracer activity measured using the algorithm..... 43

1. Introduction

The rapid growth of computer aided diagnosis and computerized medical image analysis has propelled the need for advanced image processing techniques in the medical field. The discovery of X-rays by German physicist Wilhelm Conrad Roentgen in 1895 starts the timeline of medical imaging history. Following this discovery, came the widespread application of X-rays in medical diagnosis including the use of fluoroscopy to study the blood vessels by Dr. Francis Henry Williams (Linton Summer 1995). The application of imaging in oncology dates back to the 1910's when Marie Curie published the theory of radioactivity and the investigation of X-ray radiation on patient therapy. From then, different imaging modalities have been developed for various clinical purposes. These imaging modalities became an integral part in oncology research and its clinical diagnostics. Imaging modalities like the ultrasound, Computed Tomography (CT) and Magnetic Resonance Imaging (MRI) provide the anatomical information about bones as well as soft tissues inside human body, and the Positron Emission Tomography (PET), Single Photon Emission Computed Tomography (SPECT) provide the functional information about the tissue metabolic activity as it corresponds to any biologically active molecule of interest, non-invasively. Apart from the imaging techniques, currently major research is being done to enhance the computational speed of various algorithms used in processing these images.

The necessity for a distributed environment that provides image processing services, over integrated service networks has previously been identified following the widespread application of various imaging systems in the medical field. One of the early outcomes of the research in this field of developing a software architecture to handle the image processing tools, user applications, and the handshake protocols, involved data transfer between different software modules and was done in the 1990's by M. Zikos et.al (M. Zikos 1997). This described a

distributed environment that provides image processing services over integrated tele-radiology services networks. This environment facilitated the integration of new image processing software with the existing tools and provided scheduling mechanisms for efficient management of computational resources. Although millions of imaging studies are conducted worldwide, there does not exist a universal image processing algorithm for applications such as image segmentation, as each study is specific to the imaging modality and the body part being studied (Sharma and Aggarwal 2010). Preclinical imaging deals with visualization of living animals at organ, tissue, cell or molecular level for research purposes such as drug development, which involves multimodality imaging techniques at multiple time points over large number of samples. In order to facilitate an accelerated discovery process in oncology drug development, advanced image processing techniques play a significant role.

1.1. Research Objective

The Small Animal Imaging Facility (SAIF) at the Van Andel Research Institute (VARI) in Grand Rapids MI, focuses on the development of imaging technologies that can provide biomedical researchers with quality anatomical and functional information inside preclinical subjects in the fields of cancer, osteopathic, cardiovascular, and neurodegenerative research. This research focuses on using a technique called atlas based segmentation which was previously used at SAIF, for analysis of [F^{18}] FDG radiotracer uptake in soft tissues of mice in preclinical oncology studies. The data for validating the algorithm developed for automatic registration and segmentation in this thesis is obtained from a study focusing on using a different radiotracer [F^{18}] NaF, for early detection of myeloma, whose physiology closely mimics metabolic process in the bone. The [F^{18}] NaF absorbed is an indicator of the amount of bone formation (osteoblast) and bone breakdown (osteoclast) activity during the metastasis of cancer cells. This research focuses

on quantification of the radiotracer activity on individual vertebrae of the spine, which was previously accounted as a single bone structure in preclinical image segmentation (VanOss 2012) (M. B. Artem Khmelinskii 2010). The study was conducted on 6 week old Nob-Obese Diabetic (NOD) Severe Combined Immuno-Deficiency (SCID) mice with human multiple myeloma cell line injected via tail vein injection procedure. F[18] NaF radiotracer was injected and the mice was scanned at several time points in order to study the disease progression and its effect on osseous tissue. The mice were scanned with μCT in the Feet First Prone (FFP) position and reconstructed with a voxel size of 0.4x0.4x0.4mm for μCT and 0.46x0.46x0.46mm for PET. The image is stored in DICOM format (Digital Imaging and Communications in Medicine), where the information about the subject and study is stored in its header. The post processing of images include the quantification of the radioactivity recorded in the PET data in the regions where F[18] NaF is absorbed due to bone remodeling. This algorithm optimizes the post processing of the images in the study which shortens the time frame required in analysis. This algorithm has the following steps,

1) Segmentation of the spine despite varying posture of the mice during scan. To achieve this aim:
    a. Define Moby atlas skeletal hierarchy and the architecture to identify connected bones in the skeletal system.
    b. Define Moby atlas skeletal joints and apply hinge and ball & socket joints for these skeletal joints
    c. Develop the algorithm to register CT and Moby atlas by applying a smoothing technique to the CT data, loading MOBY atlas as surface mesh using the

Marching Cube algorithm, and developing a variant of the Iterative Closest Point (ICP) to align the atlas with the CT.

d. Skeletal segmentation and manually identifying the vertebrae joints to define joint movements on the atlas data.

e. Apply the ICP registration algorithm to align the atlas vertebrae onto the CT for each vertebra.

2) Measurement of PET image pixel radioactivity (in Becquerel) in the segmented bones. To achieve this aim,

    a. Register PET/CT (Segmented CT) data to measure the activity in each vertebra.

3) Evaluation of accuracy by comparing with the manual method. To achieve this aim,

    a. Manually draw Region Of Interest (ROI) on the vertebrae of 3 PET/CT image to measure the activity.

    b. Identify the activity based on the automated segmentation method.

    c. Statistically validate the accuracy of the automated segmentation method.

2. Literature Review

2.1. Imaging Modalities

Currently a number of Imaging modalities are available to study the anatomical as well as functional activity in a subject. Also, in the research field, these imaging modalities are widely used in pre-clinical imaging. Some of the major modalities are X-ray Computed Tomography, Positron Emission Tomography (PET), Single Photon Emission Computed Tomography (SPECT), Magnetic Resonance Imaging (MRI) and Optical Imaging.

2.1.1. Computed Tomography (CT)

X-ray Computed Tomography is one of the medical imaging procedure in which multiple 2D X-ray are taken from different angles around the subject and reconstructed to form a 3D image of the anatomical structure of the subject's body. The X-ray image is formed based on the variation in mass attenuation co-efficient of the different components of the body based on their density. Lower density body parts like water and fat have lower attenuation coefficient compared to dense components like bone. The 2D images are normally reconstructed by an algorithm called the filtered back projection algorithm. Sir Godfrey Newbold Hounsfield shared the 1979 Nobel Prize Physiology or Medicine with Allan McLeod Cormack for developing the diagnostic technique of X-ray CT. For this contribution, the unit for quantitative measure of radio density was named as Hounsfield Units (HU). Thus in an X-ray CT, air takes a value of -1000 HU, water has 0 HU and the dense cortical bones has a value of +1000 HU. The resolution of the image formed under an X-ray CT depends on many factors like the bore size of the CT machine, scan period, X-ray energy, collimator design etc. Also, for better imaging of some of the anatomical features like blood vessels, renal system, GI system, and the liver, contrast agents are also used. In the United States alone, more than 50 million CT studies are performed annually, and about

50% of CT studies use intravenous iodinated contrast media such as Gastrografin, iohexol, ioxilan, etc (Saravanan Namasivayam 2006).

Micro CT's are CT machines designed for small animal imaging. μCT's have significantly higher resolution then clinical CT's but this comes at the requirement of a reduced bore diameter. The Small Animal Imaging Facility (SAIF) has a nanoSPECT/CT (Bioscan 2011) scanner which is designed to scan mice and rats. Mice scans are typically reconstructed at a resolution of 200μm.

2.1.2.Positron Emission Tomography (PET)

PET images provide information about the functional activity of a specified target inside the subject. PET imaging is based on the phenomenon of spontaneous positron emission by the nuclei of some unstable ultra-short-lived radio nuclides (USLRs), in which the number of protons exceeds that of neutrons (Anatoliy Granov 2013). The unstable radio nuclides are injected into the body along with a biologically active molecule which is either metabolized by specific regions of the body or it binds to certain target organs. The unstable nuclide emits positrons by combining with an electron which is called annihilation. The distance travelled by the positron through the medium, called the positron range, depends on the density of the medium and the energy of the emitted positron. On annihilation they emit two 511KeV photons moving approximately 180˚ apart which are detected by the detector ring of the scanner. The detector-blocks in the ring consist of several crystals connected with a Photomultiplier tube (PMT). The resolution of the PET image is directly affected by the positron range, the size, form and material property of the detector blocks as well as the diameter of the ring. The signal output from the PMTs is used to reconstruct a tomographic image of the subject.

SAIF uses a bench top μPET for animal imaging, which has a higher resolution than clinical PET. This increased resolution is due to the smaller scintillation crystals used to detect photons (Bioscience n.d.).

2.2. Image registration

Image registration is the process of identification of spatial correspondence (Hajnal and Hill 2001). Medical imaging is about establishing shape, structure, size, and spatial relationships of anatomical structures, together with spatial information about function. Thus image registration is necessary for establishing the correspondence of spatial information in medical images and equivalent structures in the body in order to pinpoint the location of functional and structural abnormalities. This forms the basis for image interpretation and analysis. In a clinical scenario, images from multiple modalities may have to be fused in order to draw useful conclusions from the data. This requires mental compensation by the interpreter or the clinician for changes in subject position. Image registration aligns the images from these modalities and establishes correspondences between various features seen on different imaging modalities. Also, the registration of an atlas or computer models aids in the delineation of anatomical and pathological structures in medical images and is considered as an important precursor to detailed analysis (Hajnal and Hill 2001).

2.3. Image registration algorithms

Image registration algorithms can be divided into those that depend on the corresponding points between the images, on the corresponding surfaces, and on image intensities.

2.3.1. Corresponding landmark based registration

One of the simplest methods of image registration employs what are called “fiducial markers” in the two images. The landmark can be a physical marker or a pin attached to the surface of the

patient's skin or which is screwed in and attached to the bones. The second method gives an accurate registration but is highly invasive and can cause discomfort or infection to the underlying tissues. Attaching the marker to the skin can cause error because of the skin movement. These markers have to be attached firmly to the skin in order to reduce the error.

More than three markers are generally used as landmarks as the error reduces with the number of markers. In this method, the algorithm involves the calculation of the centroid of each set of points and translating the 3-D image based on the difference between the new and the old centroid points (Hajnal and Hill 2001). This point set is then rotated along its new centroid until the squared distances between the point pairs is minimized. The algorithm is validated based on error parameters like the Fiducial Registration Error (FRE) or the Target registration Error (TRE) or the more accurate and widely accepted Fiducial localization error (FLE) (Hajnal and Hill 2001).

2.3.2.Surface based registration

In surface based registration, corresponding surfaces are delineated in the two imaging modalities and the transformation that minimizes distance between the surfaces is computed. The two major algorithms based on identifying surface correspondences are the "head hat algorithm" and the "Iterative Closest point algorithm", the latter being the more widely accepted one.

2.3.2.1.Head and hat algorithm

In the "Head and Hat" algorithm, the contour of a particular surface is drawn in one modality, which is called the 'Head'. Then the corresponding set of points for the same surface from the other modality is also drawn which forms the 'Hat'. Then the algorithm tries to fit the 'hat' set of points onto the 'head' contour and calculates the difference between the various points on the contour through iterations for identifying the best fit. This method tends to fail when the

anatomical structures show symmetry of rotation. “Distance transform” is a method in which the distance from every point in space to one of the points in the surface to be registered is used as an input to the iterative computation, which can make the algorithm faster and more efficient (Hajnal and Hill 2001).

2.3.2.2.Iterative Closest Point algorithm (ICP)

ICP refers to the use of an iterative algorithm to estimate the best alignment of two point clouds. ICP is used in many applications like object identification in a scene, estimating motion correction in sensor data, merging observations into a map, to name a few. The algorithm exists in many different variants and can be tailored to use different features of the point clouds depending on the circumstances. Several research studies have been done on ICP optimization, thus a large number of variants are available in the literature. The basic form of ICP involves the following steps:

1. **Selection:** Initially, it is good to select the appropriate model points and data points to apply ICP. The data points refer to the cloud set which is transformed to match the reference model set.
2. **Matching:** It refers to the matching of the data points and the model points using nearest neighbor algorithms.
3. **Weighting:** Matched point pairs have to be weighted based on their compatibility.
4. **Rejecting:** Based on a statistical evaluation of the nearest neighbor distances, some of the point pairs may be rejected.
5. **Error metrics:** It defines the objective function that is minimized in every iteration of the algorithm.

6. **Minimization of error metric:** The error metric has to be minimized in the consecutive iterations by improving upon the matching algorithm.

Another set of iterative image registration technique which 'image pixel/voxel intensity' information instead of input image landmark/surface features, to register the input set of images are known as 'Intensity based' registration techniques.

2.3.3.Intensity based registration techniques:

2.3.3.1.Mutual information based registration

In this method, "information" is chosen as a registration metric. The shared information in two images is calculated using joint entropy. This was based on the method developed by Claude Shannon and Norbert Weiner, as a part of communication theory in 1940's. In this method, we calculate the amount of information in the two images combined. If these two images are totally unrelated then their entropies are equal to the sum of the entropies of individual images. As the images become similar, their joint entropy gets smaller. The joint entropy can be obtained through joint histogram of the images and calculating the joint probability density function of the two images to be registered. Mutual Information is applied to multi-modality image registration as in (1) (Hajnal and Hill 2001)

$$I(A,B) = H(A) + H(B) - H(A,B) \quad (1)$$

2.4.Atlases in medical imaging research

In biomedical research, human as well as small animal atlases have been used for defining geometric references and for making useful comparisons between anatomical structures and physiological function. Some are organ-dedicated atlases restricted to a single organ or organ system, while some are whole body atlases. The available clinical atlases are used in population imaging, image segmentation, image registration, and follow up studies (M. B. Artem

Khmelinskii 2010). The main three human atlases in clinical research are the Talairach brain atlas (Talairach, Rayport and Tournoux 1988), Visible Human Project whole body atlas (The Visible Human Project® 2013) and the4D NCAT torso phantom (W. Paul Segars 2001). The Talairach brain atlas provides a 3D coordinate space with labeled regions of the brain and it is clinically used in functional neurosurgery, human brain mapping, neuroradiology, medical image analysis and in neuroscience education. The Visible Human Project consists of MRI, CT and cryosection images of both male and female human bodies. It is used in a wide range of educational, diagnostic, treatment planning, and other industrial uses (The Visible Human Project® 2013). Visible Human data set is used in projects such as BCS Grid Data Blade, a Java applet that provides 2D views of the Visible Human male; I Voxel Browser, Java based web browser showing voxel data, surface models, annotations, body system relationships, volume rendering, and stereo 3D viewing, developed at University of Michigan. The 4D NCAT models the 4D motion of the lungs, heart, diaphragm, and ribs with time as the 4th dimension. This was modeled to study the motion artifacts during respiration in CT images.

For preclinical applications numerous other mice and rat atlas models are available derived from various techniques with different characteristics. Many of them are made available to the public and thoroughly defined. These atlases are enumerated in the Table 2-1 below.

Registration of an atlas to another anatomical image requires non-rigid transformation due to the postural variation and the variation in mice size that may occur during imaging studies. There are no standard protocols for imaging a mouse, as it can vary based on the type of study involved. The available atlases are non-articulated and they have to be articulated by defining joint types and degree of freedom, prior to using them for the purpose of anatomical segmentation.

Table 2-1 Table showing list of rat/mouse atlases used in pre-clinical studies

S.No	Atlas	Method	Developed at	Application
1	LONI Rat Atlas (Arthur W. Toga 1995)	Computerized cryomicrotome	UCLA Laboratory of Neuro-Imaging	Brain Mapping and neuroscience studies
2	Edinburgh Mouse Atlas (R.M. Brune 1999)	Histological Imaging	Biomedical Sciences, University of Edinburgh	Interpretation and understanding of spatial data in mouse embryos
3	MRI Atlas of Mouse Development (Marc Dhenain 2001)	11.7 T MR Imaging	California Institute of Technology	Study relationships between the components within a developing system.
4	Mouse Cochlea Database (Peter A. Santi 2008)	Orthogonal-plane fluorescence optical sectioning microscopy (OPFOS) imaging and Amira reconstruction	University of Minnesota	Establish morphometric parameters of cochlear structures in normal and mutant mice
5	MOBY mouse (Segars WP 2004)	High-resolution 3-D magnetic resonance microscopy (MRM)	Department of Radiology, Johns Hopkins University,	development of new imaging instrumentation, image acquisition strategies, and image processing and reconstruction methods
6	Digimouse (Belma Dogdas 2007)	Coregistered x-ray CT and cryosection data	Signal and Image Processing Institute, University of Southern California	Study of anatomy, Computer phantom studies to simulate imaging systems, labeling of anatomical structures.
7	Sprague–Dawley (SD) rat atlas (Xueling Bai 2006)	Cryosection milling imaging system	Huazhong University of Science and Technology, China	Integrative study of the physiological and pathological phenotype of the rat.

2.5. Related Work

Baiker et al in 2010, have described a method for fully automated segmentation of μCT image using an articulated MOBY mouse atlas. The joints in the atlas are given anatomically realistic joint types and are defined in a hierarchical atlas tree (Martin Baikera 2010). It uses the ICP algorithm and constraints on the Degrees of Freedom at the joints for local registration based on the model tree hierarchy.

Baiker et al in 2012, described a method in which the bones from the SPECT image were segmented and applied to an Articulated Planar Reformation (APR) algorithm (H. C. Artem Khmelinskii 2012) for side by side change visualization and comparison. The results were shown to be robust for even "incomplete" (large chunks of bone missing) data as can be the case in bone metastasis studies.

Langerak et al in 2013, proposed a method where, multiple atlases are clustered together prior to the registration to ensure robustness of the segmentation procedure (Thomas R. Langerak 2013). Though the process reduces computation time, the authors mentioned that multi-atlas based segmentation could reduce the accuracy of segmentation in certain applications.

3. Methodology

Figure 3-1, shows the block diagram for the registration of the MOBY atlas onto the CT image of the mouse and the steps to quantify the radioactivity in skull and vertebrae regions. The Moby atlas was used as a reference for the segmentation of the μCT data. The Moby atlas was chosen over other atlas data because the articulated version of atlas developed by Baiker et al was available online on request for research. The articulated version of the atlas has the following anatomical segments: inside and outside skull surfaces, upper and lower forelimb, upper and lower hind limb, front and hind paw, spine, rib, sternum, clavicle and pelvis, with a total of 21 segments. Also, the different joints were defined as either ball and socket joints, or hinge joints. Figure 3-2, shows all the defined joint types on the Moby atlas.

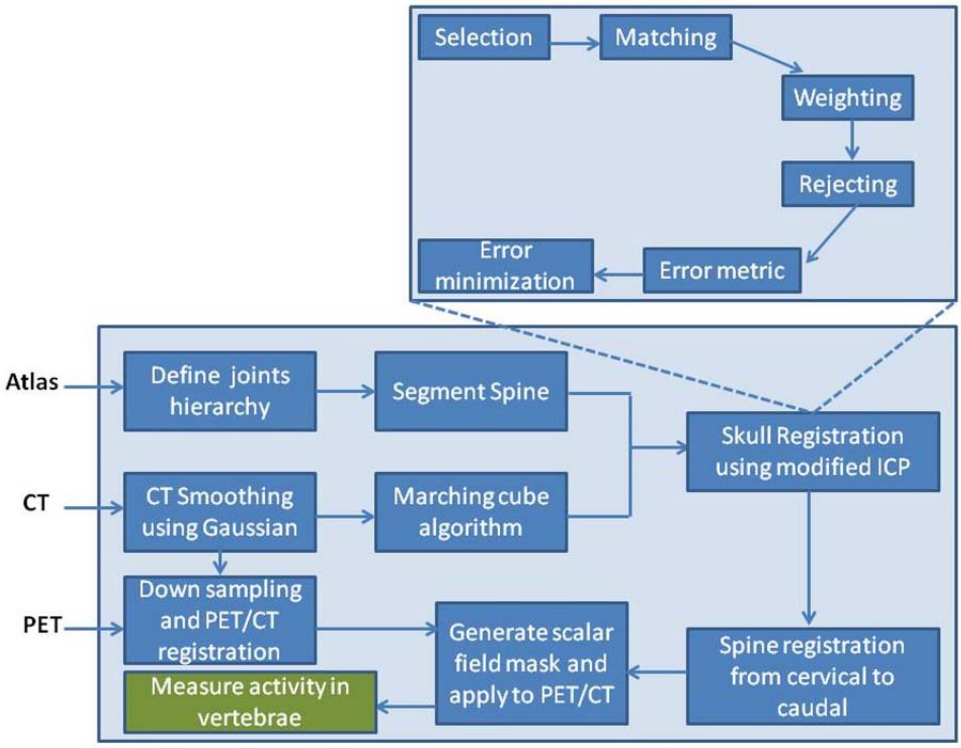


Figure 3-1 Generalized block diagram for the algorithm. The segmented atlas is loaded and joints are defined as either ball and socket or hinge joints. The spine is segmented using an anatomical reference of a 15 week old male mouse. μCT image is converted to mesh using the marching cube algorithm. The segmented atlas skull and spine and the μCT mesh data is registered together using the modified ICP algorithm. It generates a scalar field mask of the atlas and applies it to the PET image to measure the activity in vertebrae.

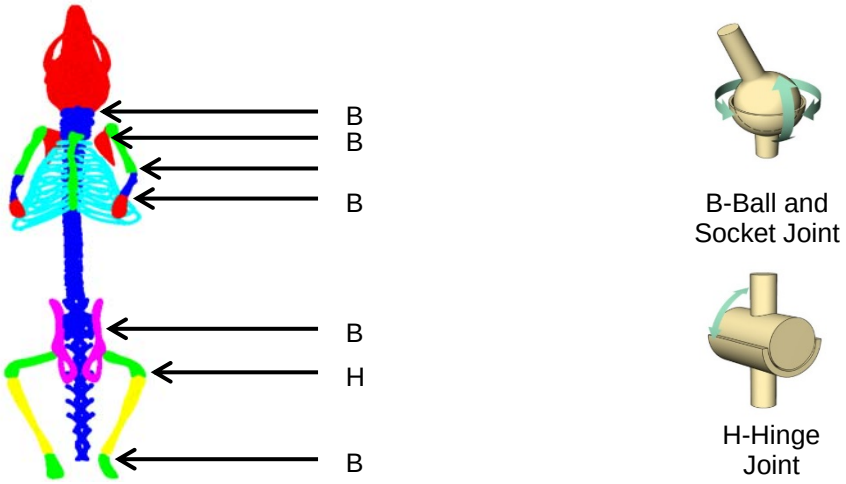


Figure 3-2 Figure showing the various anatomical joints and the joint type on the Moby. Ball and socket joints are identified as B and hinge joints as H in the Figure on the left.

Also, In the proposed algorithm, an architecture and nomenclature was assigned to label each articulated segment of the atlas as shown in Figure 3-3. This was done to identify the anatomical connection between different articulated segments on the atlas. The atlas segments were divided into four regions - left, right, top, and bottom as shown in Figure 3-3. Also, the interconnected segments were sequentially numbered in each region. Each segment on the atlas has a specific name based on its position in the hierarchy as in R-T-6 for the right forepaw where, R stands for Right, T for Top and 6 is its position in the hierarchy. Thus, in the algorithm, if a parent segment (segment higher in the hierarchy) is moved during automatic registration, the algorithm could automatically identify the daughter segments to be registered next. This way, the same parent transformations can be automatically applied to the daughter segments prior to their registration, thereby reducing user interaction and registration time.

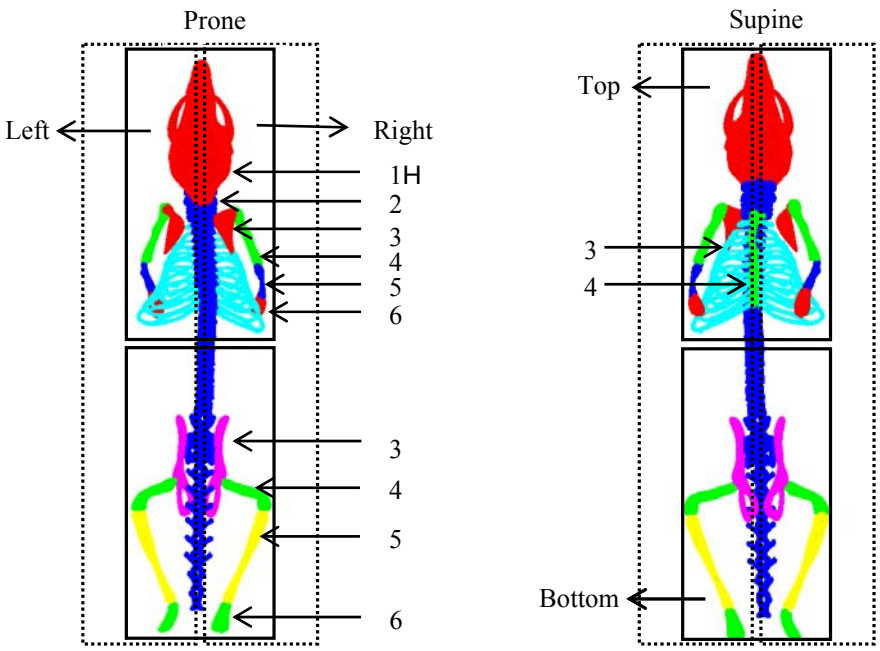


Figure 3-3 Figure showing the atlas regions for automatic segmentation.

The application of the architecture is shown in Figure 3-4, where the movement of the clavicle by a specified degree of angular measurement results in the movement of upper and lower forelimb by the same degree in consecutive steps automatically, ending with the left paw. Thus if the automatic registration algorithm generates a transformation R for a parent segment on the atlas to register with the μCT, the same transformation will be applied to the interconnected segments based on the defined hierarchy and nomenclature.

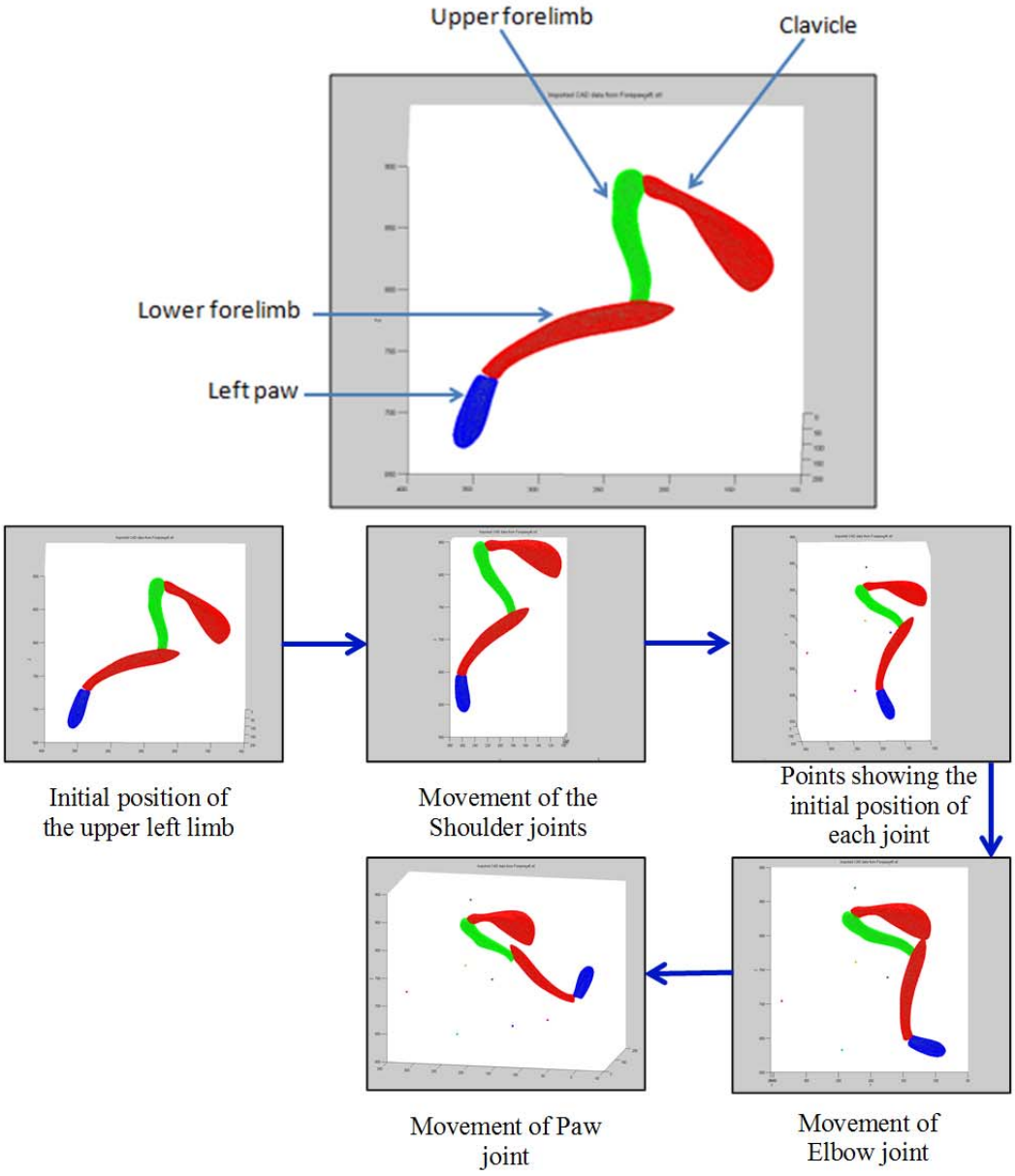


Figure 3-4 Figure showing the application of the atlas segmentation hierarchy.

In order to test the accuracy of the ICP registration algorithm, two copies of the atlas rib was plotted at 60 degree orientation from one another. The ICP algorithm was used to register both the rib segments of the atlas. The ICP algorithm applies the transformation on point cloud to register with the data cloud. Figure 3-5, shows the initial orientation and the registered states of

the two rib segments point clouds. It shows the magnified version of the registered 3D point cloud to show the closeness on the registered points.

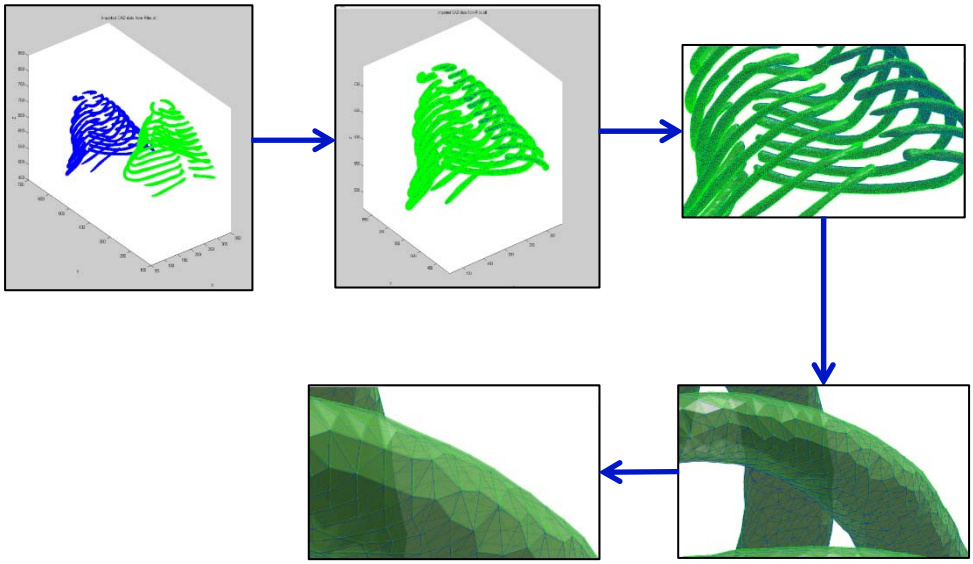


Figure 3-5 Figure showing the registration of the point cloud and the data cloud using the standard ICP algorithm.

3.1. Marching cube algorithm

The marching cube algorithm is currently the standard used for 3D surface reconstruction in medical visualization industry (Kalyankar and Apte 2013). It can be used to generate 3D structures from 2D CT scan dataset. Since 2D images cannot convey the underlying complexity of human anatomy, we rely on 3D reconstruction for interpretation of the acquired medical image. It allows medical professionals to properly visualize the volume and shape of features that they are interested in analyzing, like bone metastasis and remodeling features on specific vertebrae, or a particular tumor and its vascularization..

Rendering is a technology used in visualizing 3D datasets by displaying volumetric data as a meaningful two dimensional image. Based on the structure and data type, several techniques are used for rendering. Rendering is crucial for this algorithm as it uses a surface based registration technique for registering the Moby atlas to the μCT data for segmentation. Through rendering it

creates a surface around the μCT volumetric dataset with similar properties which is later used by Iterative Closest Point (ICP) algorithm for registration. There are two types of 3D rendering in medical industry: namely the cross-section rendering and threshold rendering. In cross-section rendering the volume is considered opaque and the user selects areas to render by adding new light sources illuminating the 2D cross-sectional slices. In this algorithm, threshold rendering is performed where the surfaces are rendered based on the tissue density selected by the user. Various anatomy regions can be rendered based on their tissue density. Tissue density can be obtained from CT images and the medical professional can easily use it to select the specific region to be rendered.

In this algorithm, eight pixel values are considered from two adjacent slices of the CT image. The density value of the tissues correspond to Hounsfield Unit (HU) in CT. Areas of the tissue with the same HU are known as the iso-surface value for the rendering algorithm. Each of the image pixel values are compared against the iso-surface value to determine if they are above or below the iso-surface value. This determines if each of the pixel in 2D CT dataset falls within or outside the rendered surface. The eight vertex forms an imaginary cube and we have to determine how the surface intersects the cube. Since there are eight vertices and two possible logical states (inside/outside) per vertex, there are 256 ways the surface could intersect the cube. By triangulating the 256 possibilities the method becomes error-prone and tedious. When the relationships of the surface values are inverted, the topology of the triangulated surface remains the same. This means that in case 1 of Figure 3-6, the vertex under consideration can either be inside or outside the iso-surface based on its value being higher or lower than the iso-surface value. Both cases the topology/surface intersection looks the same. Also, there is a rotational symmetry among some of the cases, as in case 1 of Figure 3-6, has one vertex inside/outside the

surface. The same surface intersection possibility will occur if any of the other eight vertex was inside/outside the iso-surface, thus eliminating the possibility of 7 other cases. This reduces the surface intersection possibilities from 256 to 14. As shown in figure 3-6, by analyzing just 8 vertices a precise surface can be expressed as a combination of 5 or less triangles.

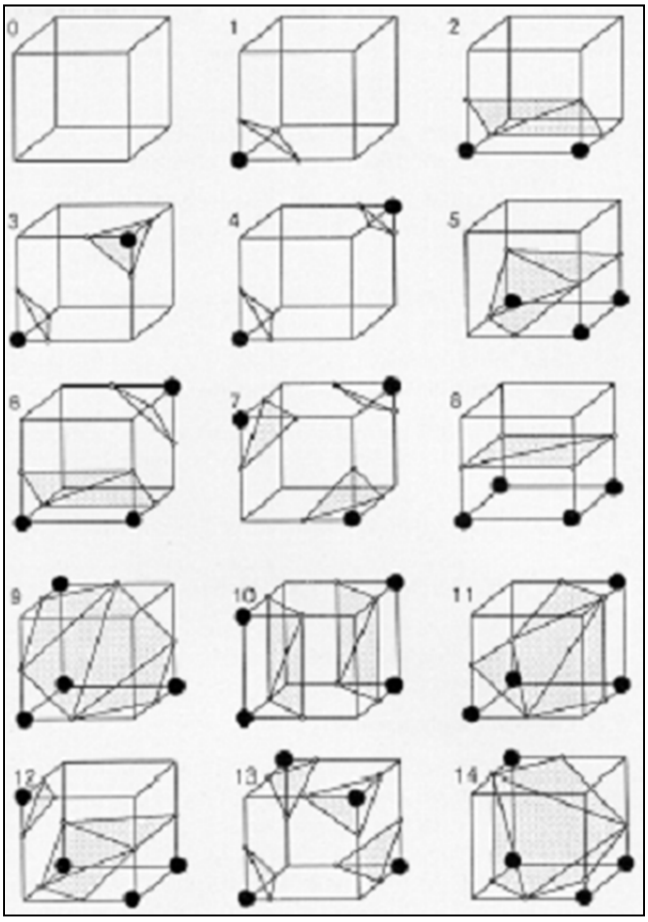


Figure 3-6 Triangulation cubes in marching cube algorithm (Lorensen and Cline 1987)

The relation between the cubes is such that, every cube shares four vertices with the adjacent cube to form continuous surface rendering. As shown in Figure 3-7, both the cubes share the vertices 1, 2, 3 and 4. The marching cube algorithm was implemented in MATLAB using the built in 'isosurface' function. Figure 3-8, shows the surface rendering of the μCT data of the mice using the marching cube algorithm with iso-surface value set to the Hounsfield Unit (HU) of the bone density, i.e. 750 HU.

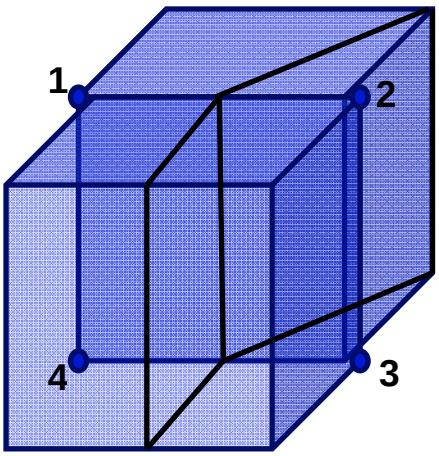


Figure 3-7 Figure showing the continuous surface rendering on both cubes by sharing the vertices 1, 2, 3 and 4.

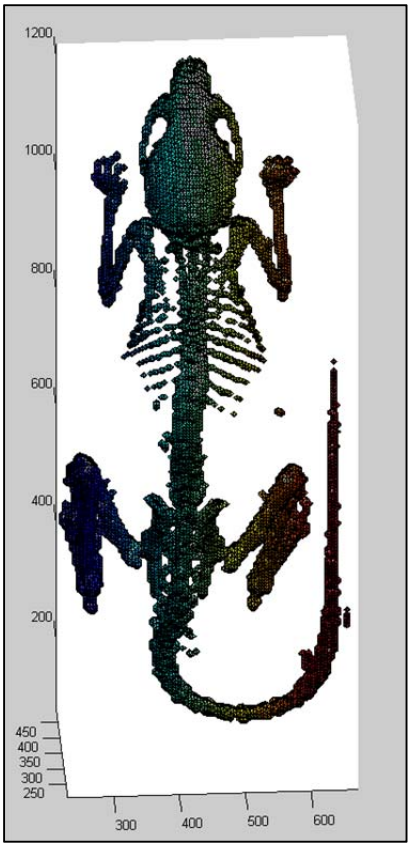


Figure 3-8 Figure showing the surface rendered μCT image of the mice using marching cube algorithm.

Prior to segmentation of the atlas spine, the registration algorithm is applied to the μCT mesh data obtained from the marching cube algorithm to register with the atlas skull and spine. The ICP algorithm was used to register the atlas spine and the μCT spine as shown in Figure 3-9. The

output of the algorithm shows that with the un-segmented rigid atlas spine structure, only a few regions of the cervical and caudal spine were registered. This problem was not tackled in the current version of the software as well as current research in atlas mouse segmentation procedure. Baiker et al, modeled the spine as a 3D curve curve between the neck and pelvis and it was not registered or segmented from the CT (M. B. Artem Khmelinskii 2010). In the work done by VanOss et all, the spine was segmented from the volumetric data as the 3D region between the neck joint and the hip joint (VanOss 2012). The 3D region grown starting from the neck to the posterior direction was segmented as the spine. This greatly depends on the static threshold value initially set to filter the bones from the soft tissue. An error in setting the threshold CT pixel value could affect the uptake value of spine in the results. The spine in the MOBY mouse atlas was manually segmented based on the work by Margaret J. Cook (.Cook 1965) as shown in Figure 3-11. To segment each atlas vertebrae manually, each vertebrae joint was identified. A Matlab script was written to segment every vertex between the two joints in the axial direction of the spine and was stored as one vertebra. But few of the vertebrae as in C1-C2, C3-C7, T1-T3, T4-T8, T9-T13 were found to be fused to one another and difficult to segment manually. Thus these vertebrae were stored as single segment and used for registration with the CT. Figure 3-10, shows the skeletal anatomy of a LAC grey mouse. This way the vertebrae could be moved in the atlas to register with the μCT data.

Figure 3-9 Figure showing the registration of the µCT mesh and the un-segmented atlas spine.

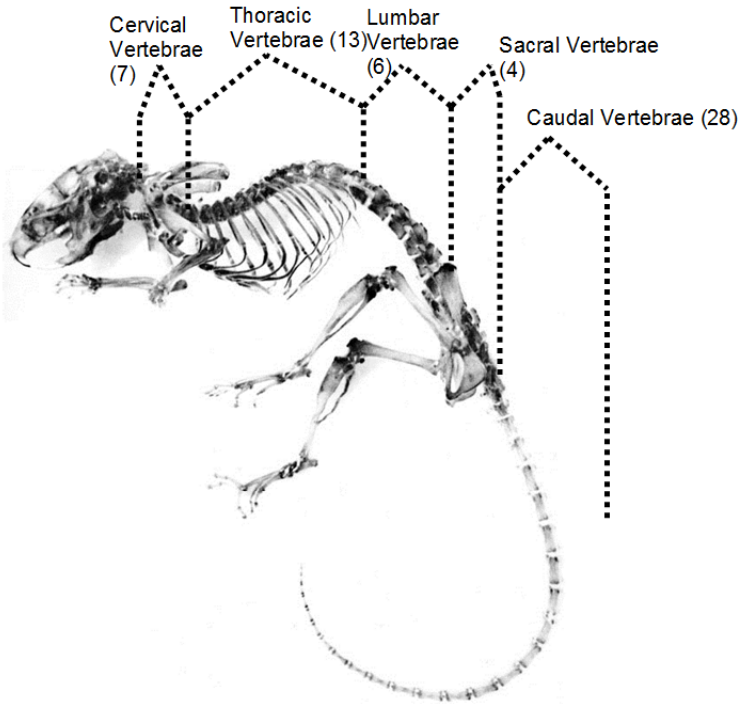


Figure 3-10 Figure showing the skeletal anatomy of LAC grey mouse illustrating the number of vertebrae of each category

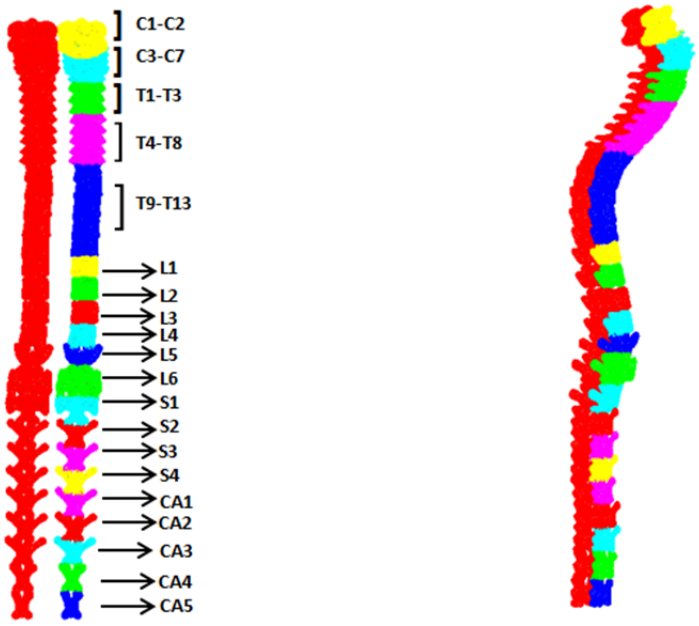


Figure 3-11 Figure showing the manually segmented atlas spine showing 7 Cervical, 13 Thoracic, 6 Lumbar, 4 Sacral and 5 Caudal vertebrae.

3.2. Iterative Closest Point Algorithm

Iterative Closest Point (ICP) algorithm is used for geometric alignment of point clouds when an estimate on their initial positions is known. The efficiency or performance of the algorithm can be enhanced if a better estimate on the initial position is known. Several variants on the ICP algorithm have been described that affect all phases of the algorithm namely, selection, matching, error metric and minimization of the error metric. Here we use a variant of ICP algorithm to increase the computational efficiency and cut down the processing time to register the segmented spine atlas and the μCT mesh point cloud. In ICP taxonomy, the atlas point cloud is called the data point cloud ‘D’ and the μCT mesh point cloud is the model point cloud ‘M’. The transformation is applied on the data point cloud ‘D’ to register to the model point cloud ‘M’. The steps for the ICP algorithm include point selection to reduce the computation complexity on each iteration, point matching to identify the nearest neighbors, calculation of error metric and minimization to improve registration on each iteration.

3.2.1.Point selection

Initially, the points to be registered have to be selected so that they are in close proximity and reduce the computation time over each iteration. To ensure the proximity of the vertebrae, the skull registration is performed prior to the spine. For skull registration, the same ICP algorithm is used with all the points from the Data and Model point clouds. The skull has a well-defined structure which helps in fast registration compared to the vertebrae. Once the spine gets registered, the other bones can be located close to the μCT by following the joint types and the atlas segmentation hierarchy as shown in Figure3-2 and Figure 3-3.

Figure 3-12 Figure showing the alignment of the CT and atlas after applying the initial input conditions.

The identification of initial point of registration for the atlas skull segment to the CT is based on the following input conditions:

a. The anterior and posterior end of the mouse is known and given as input to the algorithm.

b. Scan orientation of the mice is known and it remains the same for all the subjects in a study. The study from which this data is collected has all the mice scanned in Feet First Prone (FFP) position. This information can also be fetched from the DICOM header of the image file.

c. A rough estimate of the size scale required for matching the atlas and CT is known and is given as input.

This provides the rough alignment of the CT skull in close proximity to the atlas data as shown in Figure 3-12. In regular cases, all the three input conditions stated above, required for initial alignment, remains the same. This makes the "point matching" step easier, where every point in the atlas skull is matched with another in CT data. Following the initial alignment, the nearest neighbor for majority of the points in the atlas skull lies on the CT skull.

3.2.2.Point matching

Equation 2, shows the algorithm used for point matching.

$$C_{k(i)} = \min_{1 \le j \le N_M; 1 \le i \le N_D} \left(\left| \left| M_j - T_{k-1} D_i \right| \right|^2 \right) \qquad (2)$$

where, k is the number of ICP iteration, T_k is the transform from the previous iteration, N_M is the number of points in the model point cloud and N_D is the number of points in the data point cloud. Euclidean distance of the points from Model cloud and the Data cloud is calculated and is used as an error metric to determine the transformation for the next iteration.

In this step, a nearest neighbor search is performed to get the data points closest to the model point set. A basic approach is to find out the distance between all the points in data and model points to identify the shortest distance. This is known as the naïve approach, or brute-force

approach, or exhaustive approach. Although simple to implement, the computational complexity scales linearly with the number of points, f (N) in the point clouds.

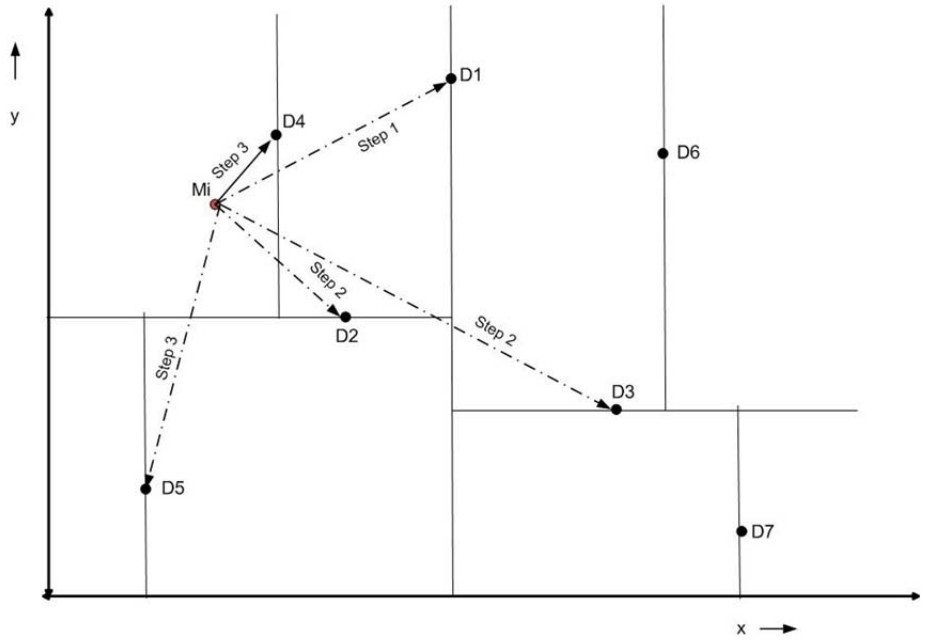


Figure 3-13 Figure showing K-DTree formation to find the nearest neighbor for the data point cloud Di in the model point cloud Mi.

In this algorithm, the kDtree method was used to find the nearest neighbor for the data cloud in the model cloud. To find the nearest neighbor using kDtree algorithm, the data point cloud, D is split by finding the median of all the points' first coordinates. The median point becomes the root of the tree. The entire data point cloud is divided into kDtree, which is the preprocessing required for this algorithm. To determine the nearest neighbor, as we move down the root, the distance of the model point to the data point is recorded. The nearest of the two branches from the root is determined and finally the closest branch with minimal distance is determined as the nearest point. The computational complexity of creating a kDtree with dimension 'k' is no higher than $O(kN\log_2 N)$ (Friedman, Bentley and Finkel 1977), where O is the asymptotic notation or the 'Big O' notation used to describe the limiting behavior of a function when the arguments tend to

a particular value or infinity. This way by using the kDtree, the computational complexity of finding the nearest neighbor in Euclidean matrix is greatly reduced.

3.2.3.Error metric and minimization of error metric

Error metric refers to the objective function that is minimized in every iteration of the algorithm. In this algorithm, point to point minimization is performed. The sum of squared distances of data points to model points is determined as in equation (3).

$$f(z) = \sum_{n=0}^{N} \left|\left|Rdi + T - mj\right|\right|^2 \qquad (2)$$

Where di refers to the data cloud points and mj refers to the model cloud points.

A closed form solution can be obtained for the point to point minimization using Singular Value Decomposition (SVD) as shown below.

The centroids for the data and model point cloud can be defined as

$$\bar{d} = \frac{1}{p}\sum d \qquad (3)$$

$$\bar{m} = \frac{1}{q}\sum m \qquad (4)$$

Where, p and q are the number of number of data and model point clouds. Then the point deviation from centroid is given as,

$$d_{i=}^{'} d_i - \bar{d} \qquad (5)$$

$$m_i^{'} = m_i - \bar{m} \qquad (6)$$

To determine the rotation matrix **R** and translation matrix $\vec{T}$,

$$E = \sum_{i=1}^{N} \left\| Rd_i + \vec{T} - m_i \right\|^2 \qquad (7)$$

Substituting (6) & (7) in (8),

$$E = \sum_{i=1}^{N} \left\| R\left(d_i^{'} + \bar{d}\right) + \vec{T}(m_i^{'} + \bar{m}) \right\|^2 = \sum_{i=1}^{N} \left\| R\bar{d_i^{'}} - m_i^{'} + (R\bar{d} - \bar{m} + \vec{T}) \right\|^2 \quad (8)$$

The error can be minimized by translating the data point centroid to the centroid of the model point. Thus,

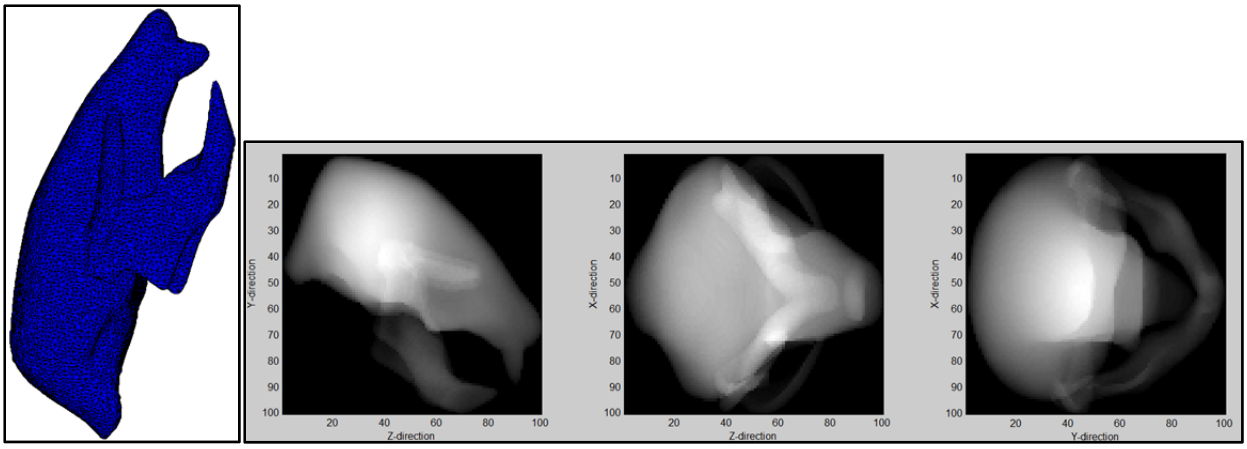
$$\vec{T} = \bar{q} - R\bar{p} \qquad (9)$$

Substituting (10) into (9) and simplifying, we can see that the error can be reduced by maximizing 11.

$$tr(RN) = \sum_{i=1}^{3} r_i \,.\, c_i \leq \sum_{i=1}^{3} \|r_i\| \|c_i\| \qquad (10)$$

Where, $\boldsymbol{N} = \sum_{i=1}^{N} d_i' \,.\, m_i'^{T}$

This can be achieved by considering the SVD of $N=U\sum V^T$ and choosing $R=VU^T$ (Kjer and Wilm 2010). This gives the value on right hand side of (11) as $tr(\sqrt{N^T N}$, is the maximum value and it occurs when $R=VU^T$. Applying the transformation to the data points registers it to the model point cloud with a minimized error E. Several iterations can be performed based on the initial alignment of the data and model point clouds to obtain a convergence result. The registered atlas vertebrae form the ROI for measuring the activity from the PET data. The ROI in the 3D coordinate format is converted to volumetric data as shown in Figure 3-14. The Figure shows the registered atlas skull and the volumetric data of the same.



(a) (b)

Figure 3-14. Figure showing (a) the registered skull atlas in 3D coordinate format (b) the volumetric data of the registered atlas skull.

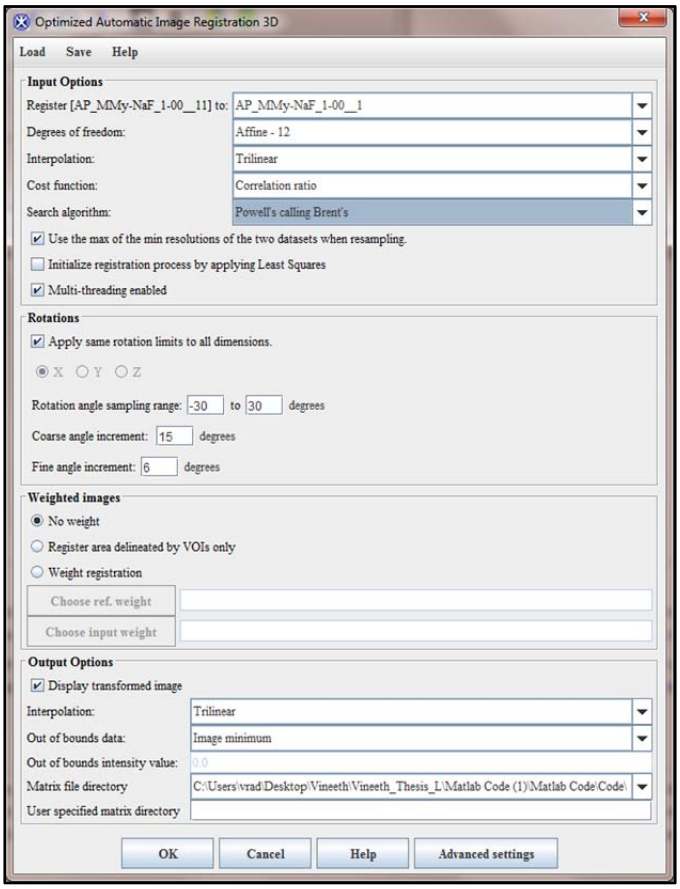


Figure 3-15. Figure showing the settings for the PET/CT registration in MIPAV tool for PET radiotracer activity using manual method.

3.3. Manual method using MIPAV

Medical Image Processing, Analysis and Visualization (MIPAV), is a software tool developed by National Institute of Health (NIH), to develop computational methods to analyze and quantify biomedical data. MIPAV v7.3.0 tool was used to register the PET and CT input data and perform manual ROI drawing on the data set. The MIPAV tool uses “Powell’s calling Brent’s” as the nearest neighbour search algorithm and trilinear interpolation for sampling along with other parameters as shown in Figure 3-15. The Volume Of Interest (VOI), was drawn on L1 to L5 vetebrae on the CT image as they are more visually discerable compared to the PET data. The VOI are then opened on the registered PET image for quantization of the activity in the PET data. MIPAV tool is used as the gold standard tool for measuring the activity on the PET data by manual method.

4. Results

Figure 4-1(a), shows the atlas spine prior to segmentation and Figure 4-1(b), shows the segmented atlas after registration onto the μCT. The two images show the segmented atlas spine orientation before and after the registration algorithm has been applied to the μCT data. Figure 4-2, shows the different azimuth orientation of registered atlas and μCT data.

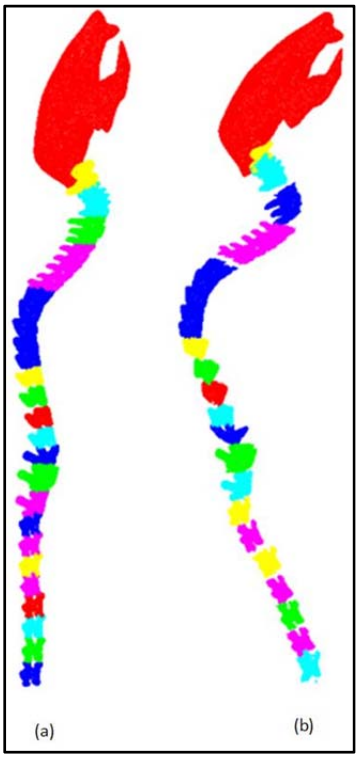


Figure 4-1 Figure showing the atlas prior to registration (on the left) and the atlas image after registration on the μCT image and segmentation (on the right).

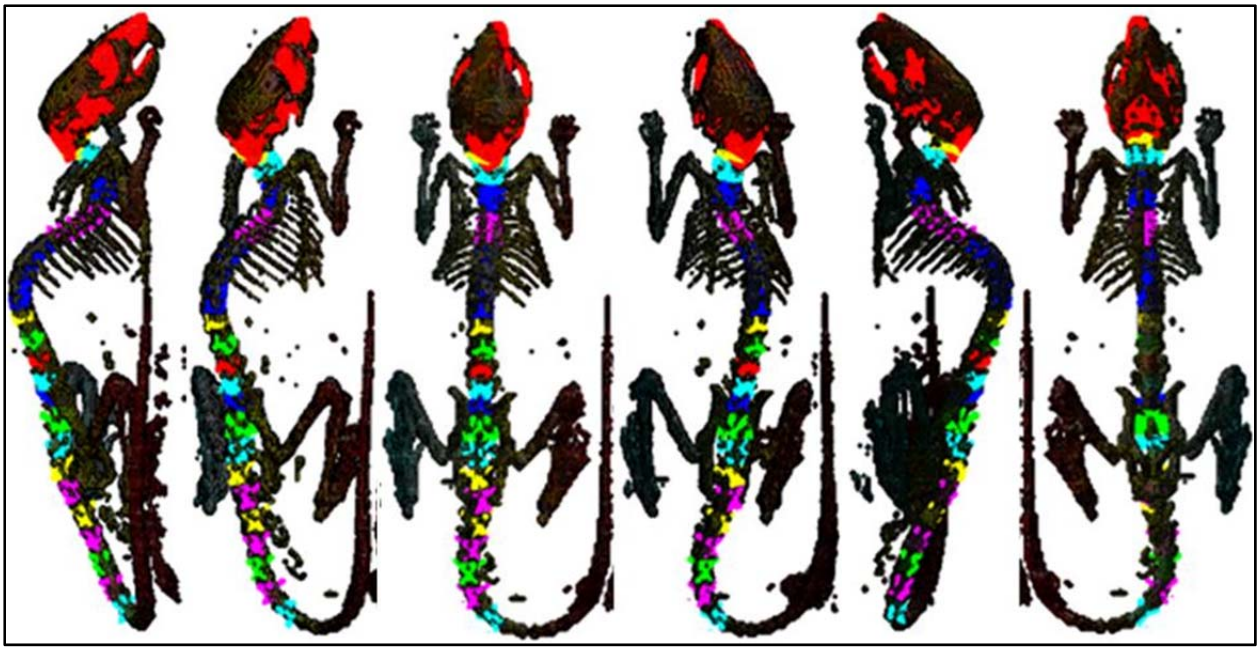


Figure 4-2 Figure showing the registered atlas and μCT data at different azimuth angles.

The ROI was drawn on the CT image as shown in Figure 4-4 and the ROI was moved to the registered PET data as shown in Figure 4-5.

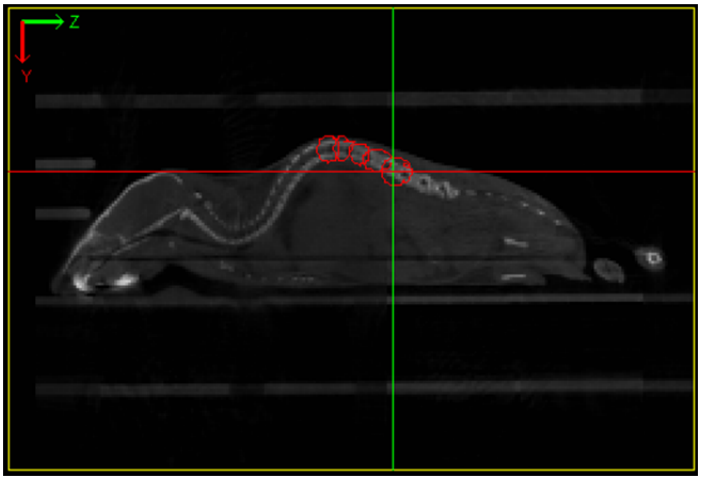


Figure 4-3 Figure showing the CT image with manually drawn ROI in MIPAV.

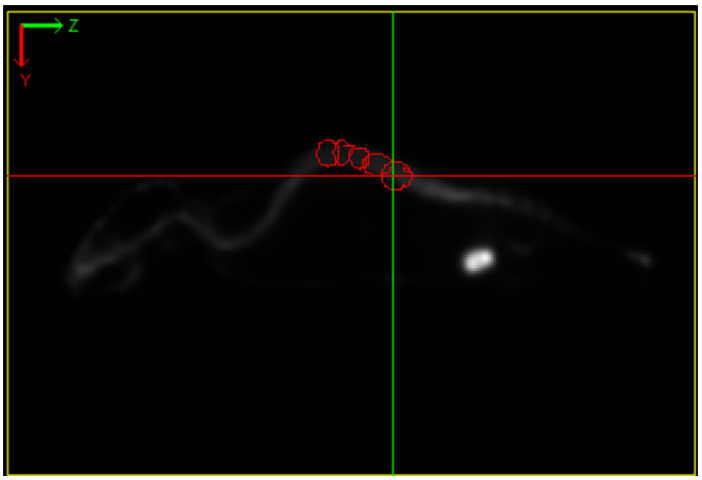


Figure 4-4 Figure showing the PET image with manually drawn ROI copied from the CT image.

Table 4-1, shows the PET activity in BQML (becquerel per ml), where one Bq, is the activity of a quantity of radioactive material in which one nucleus decays per second. Table 1.2, shows the radiotracer activity on the L1-L5 vetebrae measured by manual method and by using the segmentation algorithm. Error rate was calculated using (12) and recorded in Figure 4-8.

$$\text{Error Rate (E)} = \frac{||Manually\ calculated\ value - Automated\ value||}{Manually\ calculated\ value} * 100 \quad (12)$$

Table 4-1Table showing the PET radiotracer activity measured using the algorithm

Subject 1	Manual	9.41E+07	4.32E+07	5.11E+07	9.28E+07	1.62E+07
	Algorithm	1.02E+08	4.12E+07	3.53E+07	7.56E+07	1.09E+07
Subject 2	Manual	5.89E+07	2.95E+07	1.82E+07	3.42E+07	2.75E+07
	Algorithm	6.74E+07	2.44E+07	1.32E+07	2.52E+07	2.06E+07
Subject 3	Manual	1.79E+07	1.99E+07	1.49E+07	5.75E+06	5.68E+06
	Algorithm	1.60E+07	1.88E+07	1.63E+07	5.90E+06	6.72E+06

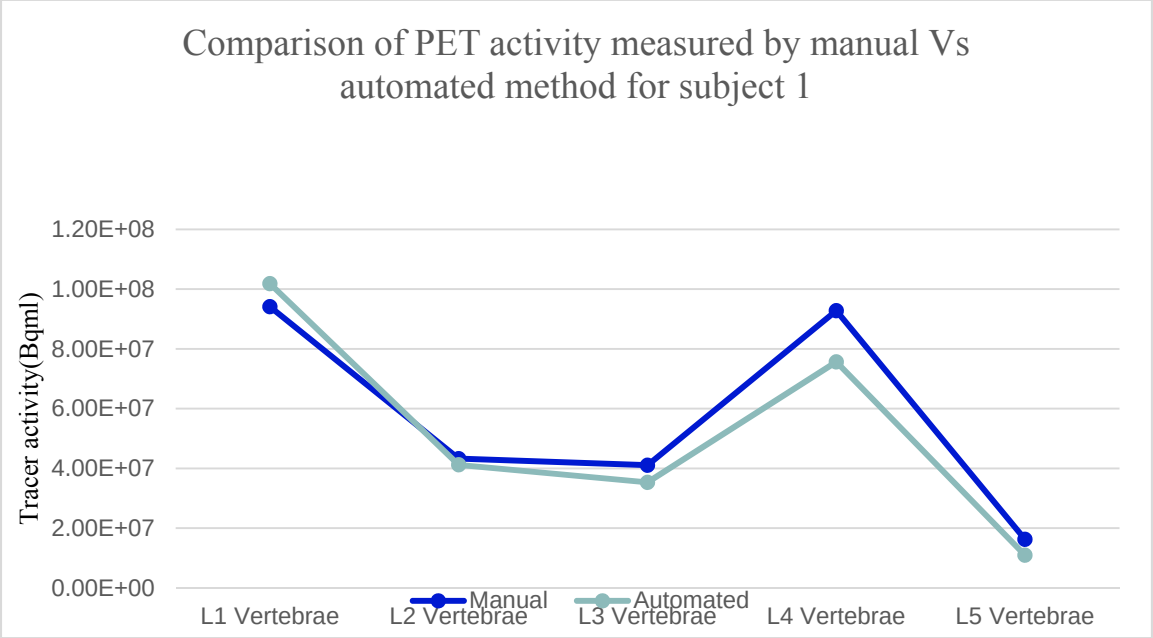


Figure 4-5 Figure showing the comparison of the PET activity measured by manual and automated method for the 1st dataset.

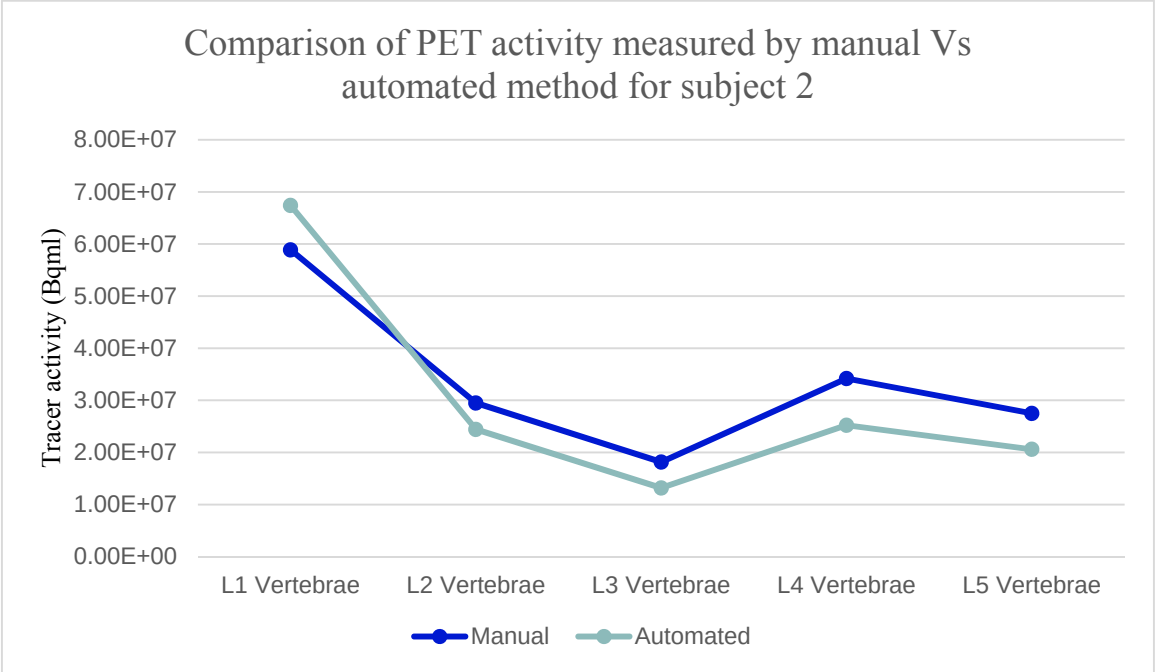


Figure 4-6 Figure showing the comparison of the PET activity measured by manual and automated method for the 2nd dataset.

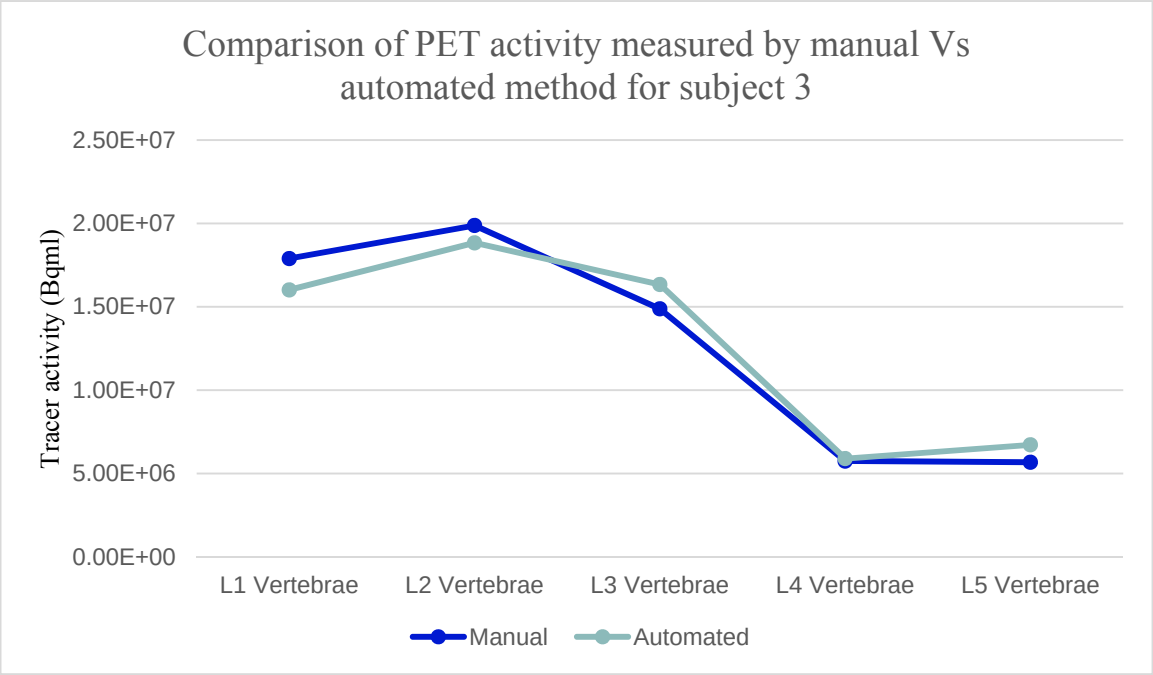


Figure 4-7 Figure showing the comparison of the PET activity measured by manual and automated method for the 3rd dataset.

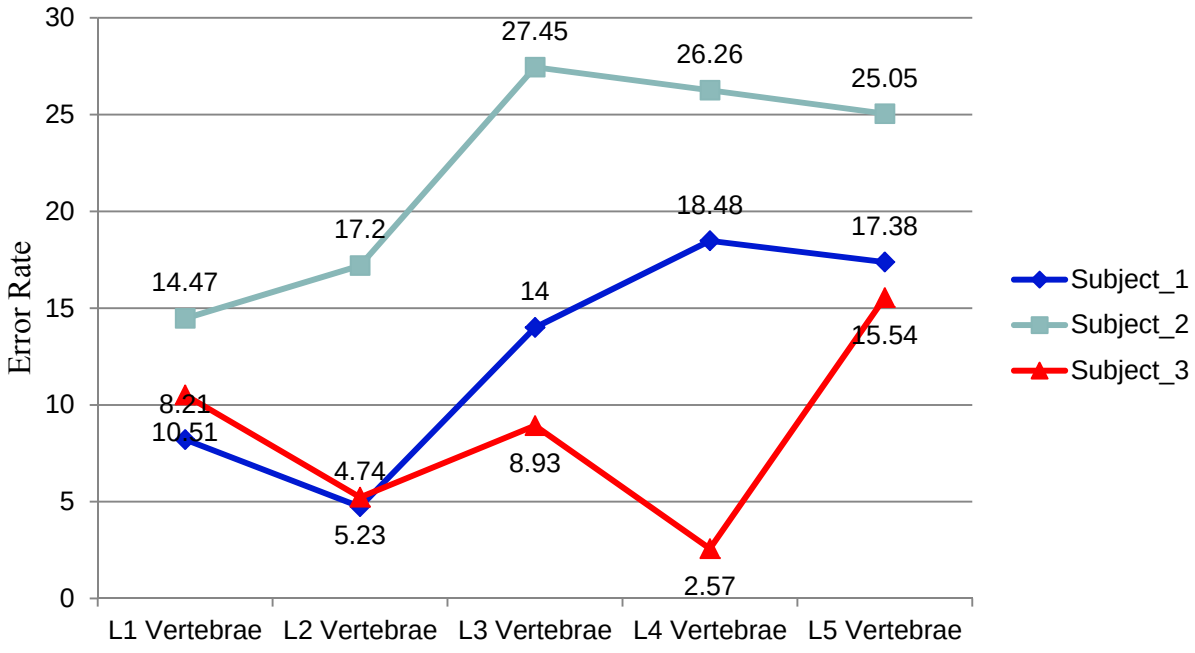


Figure 4-8 Figure showing the error percent for all the three datasets for L1-L5 vertebrae

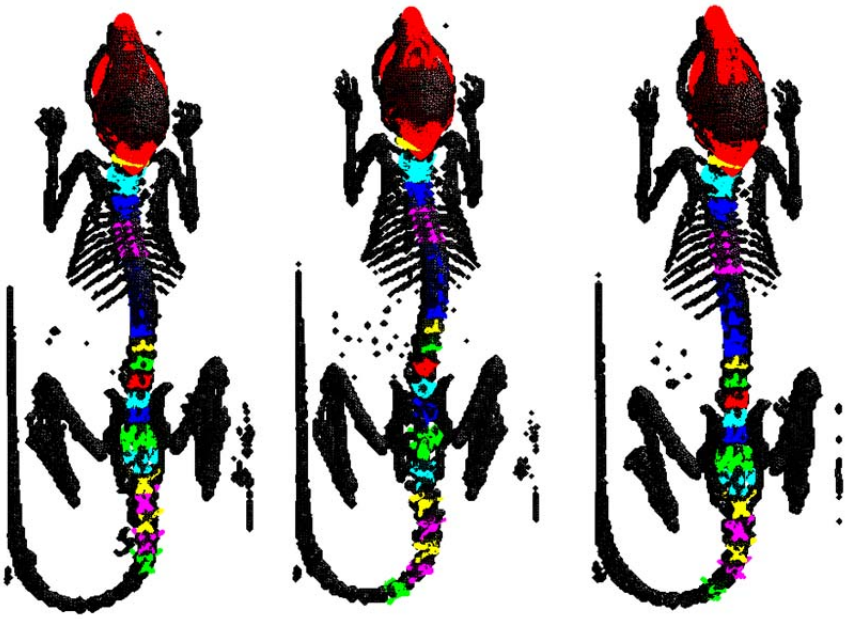


Figure 4-9 Figure showing the registered atlas vertebrae to the CT data for three different data sets.

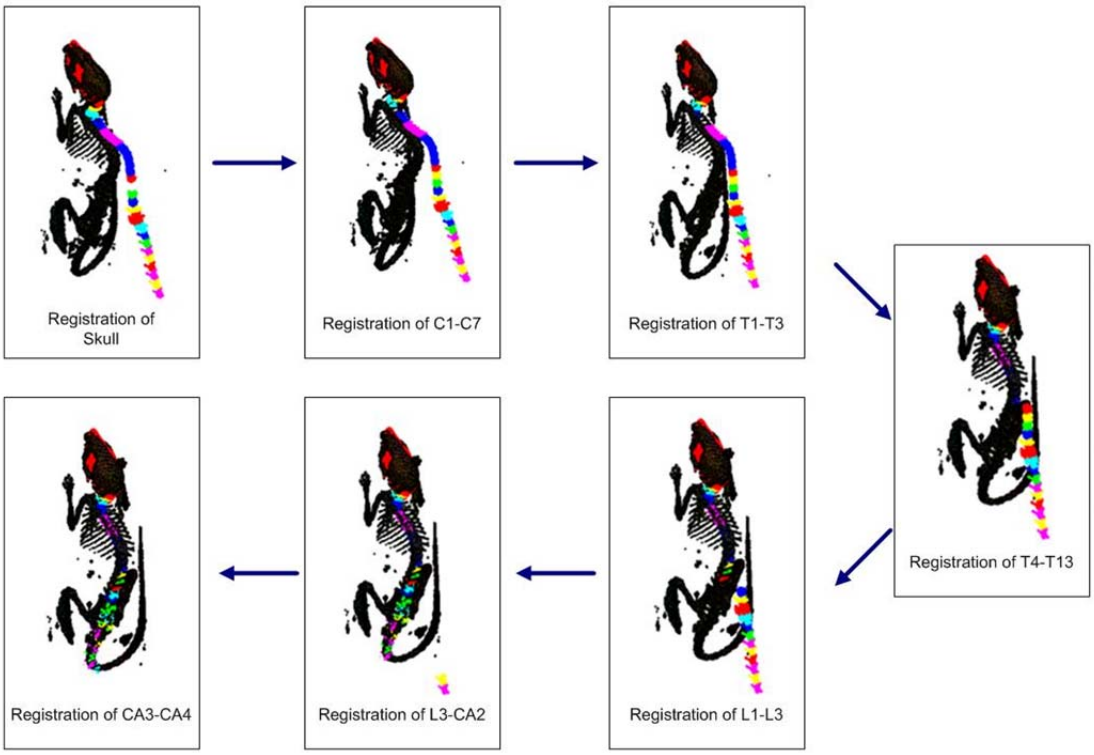


Figure 4-10 Figure showing the step by step registration of the atlas vertebrae onto the CT dataset.

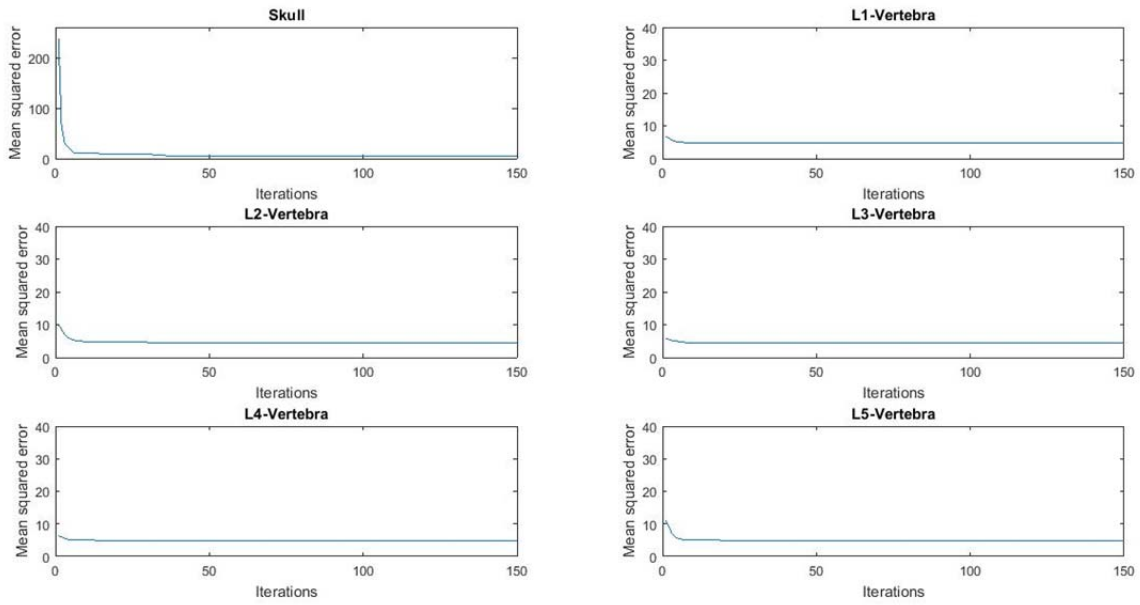


Figure 4-11. Figure showing the saturation of error metric after certain number of iterations.

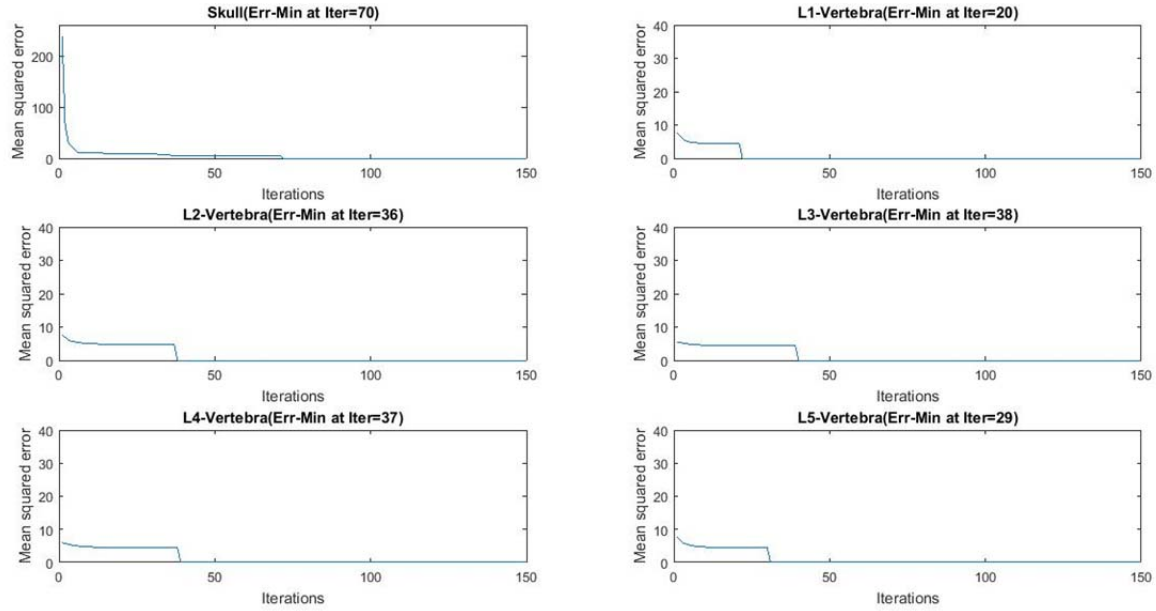


Figure 4-12. Figure showing the number of iterations before the error metric becomes significantly small.

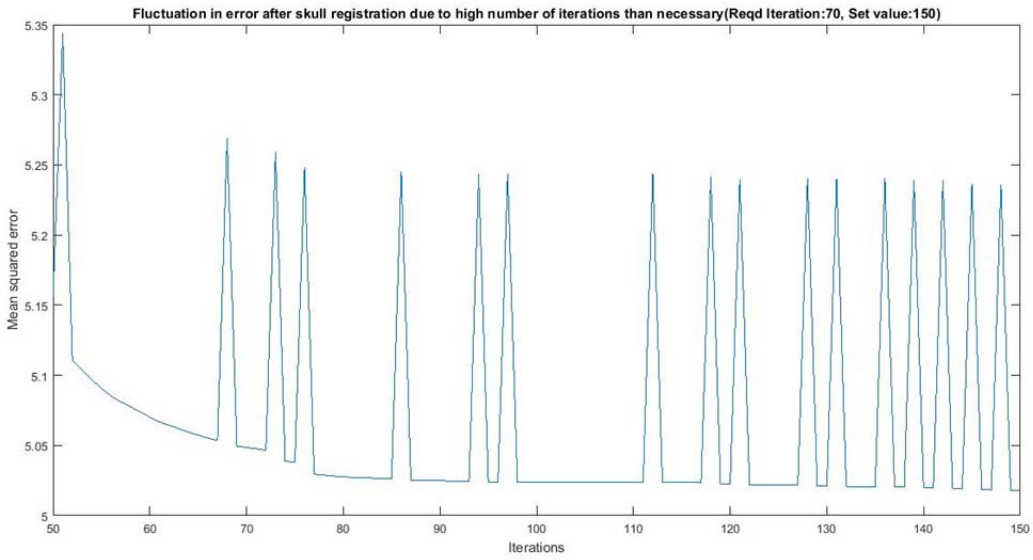


Figure 4-13 Fluctuation in error after skull registration due to high number of iterations than required (Reqd Iteration:70, Set value:150)

5. Discussion

Jadvar et. al (Jadvar, Desai and Conti 2015), performed studies on use of ^{18}F-NaF as PET/CT radiotracer for bone and joints osseous metabolic activity. It was found that, ^{18}F-NaF PET/CT was found to be diagnostically superior to other ^{99m}Tc-based bone scintigraphy for more accurate detection of extent osseous metastatic disease in a variety of cancers such as head and neck cancer, thyroid cancer, lung cancer, breast cancer, hepatocellular carcinoma, multiple myeloma, bladder cancer and prostate cancer. Accurate identification of the tumor viability was useful not only in diagnosis but also in treatment planning and follow-up.

The American Cancer Society estimates 26,850 new cases of multiple myeloma in the United States for the year of 2015 and an estimated 11,240 deaths (Society 2015). The median age at diagnosis is 69 and median age at death was estimated to be 75. 46.6% of the people was estimated to survive for 5 years. High mortality rate due to multiple myeloma was attributed to late diagnosis of the disease. Multiple myeloma causes bone metastasis which causes an imbalance in the ratio of the osteoblast to osteoclast activity of bones. During bone metastasis the osteoclast activity takes over osteoblast activity and causes bone erosion. Also, it was found that 5-10% of all cancer patients develop spinal metastasis during the course of their disease. Early diagnosis of bone/spine specific metastasis can preserve/improve neurologic functionality, achieve functional stability, optimize local tumor growth and improve the quality of life of the patients.

At the Small Animal Imaging Facility, the study was conducted to identify the precursors for early detection of multiple myeloma based on bone remodeling. The study was conducted on 6 week old Nob-Obese Diabetic (NOD) Severe Combined Immuno-Deficiency (SCID) mice with human multiple myeloma cell line injected via tail vein injection procedure. ^{18}F -NaF radiotracer was injected and the mice were scanned at several time points in order to study the disease

progression and its effect on osseous tissue. Three image dataset from the study was used to evaluate the accuracy and throughput of automated atlas based segmentation algorithm against the manual method. For the manual method, MIPAV image analysis software was used to draw ROI on individual spinal vertebrae L1-L5 on each slice and obtain the radiotracer activity value in each vertebra. The manually obtained value and the value obtained using the algorithm was tabulated as shown in Table 4.1. It shows the ^{18}F-NaF radiotracer activity value in vertebrae L1-L5. Figure 4-5 to 4-7, shows the plot of uptake value in (Bqml) in each of the vertebrae from L1-L5 for all the three subjects.

Vertebrae L1-L5 was selected as they are easy to identify for drawing the ROI manually. Figure 4-10 shows the sequential steps involved in registration of the whole atlas to the μCT dataset. Initially the skull is registered, followed by Cervical, Thoracic, Lumbar, Sacral and Caudal vertebrae. Figure 4-8, shows the percentage error in the automated method considering the manual ROI method using MIPAV tool as gold standard. It was seen that, the error percent varies from 2.57% on L4 of dataset 3 to 27.45% on L3 of dataset 2. Dataset 3 had the least error percent with a minimum and maximum value of 2.57% on L4 to 15.54% on L5. Also, dataset 2 had the highest error rate with a minimum and maximum value of 14.47% on L1 and 25.05% on L5. Also, a general trend in the variation of error percent was an increment in the error percent as we move down the spinal column. The error rate on L1and L2 on all three dataset was found to be less than L5 vertebra. This can be attributed to the cumulative error on each vertebrae registration from the skull to the caudal vertebrae. An error in the registration of skull results in an incremental error in the registration of the 1st cervical vertebrae as they are connected together in the atlas.

The algorithm was run on a Windows 8, Intel Core 64 bit, i5 processor laptop with 12GB DDR3 RAM laptop on Matlab® R2014a. The time taken by the algorithm was measured using the 'tic-

toc' function in Matlab and it was found to be 1414.34 seconds or 24 minutes (approximately). The time taken for the manual method to register the PET and CT dataset, draw the ROI on each slice and each vertebrae takes 60-65 minutes approx. Thus the time taken for the post processing of image dataset was reduced to one-third by automated method accelerating the image analysis workflow. Also, the time taken by the automated method can be reduced further by using a computer with a better configuration (eg: i7 processor with16GB RAM). Also, it was seen that activity measured during manual method depends on the subject drawing the ROI and the values slightly vary during each attempt. But the automated method is user independent on each attempt of image analysis and thus more robust compared to the manual method.

An analysis on the logic to stop the iteration for the ICP algorithm was performed. A static number of iteration, 150 was initially set to the algorithm, where algorithm attempts to register the atlas to the CT during 150 iterations. As in Figure 4-11, the error metric was found to be saturated after a particular number of iteration. It was observed that, the error metric saturates at different iteration for different segments based on their initial alignment. Following this, an arbitrary threshold of 1/1000 was set for the variation in error metric to be called as in saturated state. The algorithm stops the iteration when the difference of error metric from the previous iteration is less than 1/1000. Figure 4-12, shows that the registration gets completed at an early stage than the set value of 150 iterations. Skull segment was found to have a high initial error metric because it was the first segment to be registered. As expected, the number of iterations required for a saturated error metric was found to be 70 iterations. The number of iterations were found to lower for the L1-L5 segments and were in the rage of 29-38 iterations. The algorithm stops the registration once the error metric gets saturated, reducing the registration time (in the milli-seconds (ms) range). Since the algorithm takes 15-20 minutes from beginning till the end, the improvement in overall throughout due of the lower number of iterations were not

significant. Also, from Figure 4-11 and 4-12, it was found that, in the case of a set number of iterations, a slightly higher initial error metric was observed in certain segments like L2 and L5. This can be attributed to the fluctuation in the error metric in its saturated state and when the set value for the iteration coincides with a high fluctuation in error metric, the initial alignment for the following segment will result in higher misalignment. Figure 4-13, shows the fluctuation in the error metric for the skull segment in its saturated state. In this case, the required number of iteration for registration based on the arbitrary threshold was 70 and the set value was higher at 150.

In this thesis, an automated image segmentation method was used to segment individual vertebrae in mice. The algorithm was tested on three datasets from a PET/CT bone metastasis study using ^{18}F-NaF radiotracer. The algorithm was found to reduce the analysis time threefold with a potential to further reduce the automated analysis time by use of better specification for system to run the algorithm. The manual analysis value can vary each time the analysis is performed is dependent on the individual performing the analysis. Also the error percent was recorded and found that it tends to increases as the analysis moves down the spine from skull to caudal vertebrae.

6. Future Work

The performance of the algorithm can be improved by the use of an automated method to scale the μCT dataset to the size of atlas data instead of using a static scaling value. The software can include an optional manual intervention to add markers to specify each vertebra, making the tool semi-automatic and more reliable. The software can be written in another language like C++ using the ITK (Insight Segmentation and Registration Toolkit) and VTK (Visualization Toolkit) for developing better Graphical User Interface (GUI) and image processing algorithms. The above work can be integrated to the existing work done by VanOss et al, to segment all the bones in the CT data. In the future work, a semi automated analysis method can be developed where the user marks the approximate center of mass of the low contrast soft tissue and the software completes the remaining steps in registration. A more detailed study on the degree of freedom including the scaling of the subject data to the atlas can provide more robust registration of the high contrast soft tissues. Thus in the future version of the software, foremost, the skull can be registered based on the initial conditions laid out in this work or by the identification of anteroposterior axis by Principal Component Analysis method followed by VanOss et al. The registration of spine and sternum assists the registration of rib cage and high contrast lung soft tissue. The registration of low contrast soft tissue like liver has to be studied by semi-automatic method for high accuracy prior to fully automated analysis.

7. Appendices

7.1. Software Code:

Moby_icp_demo_2.m

```
%%Author: Vineeth Radhakrishnan
%%Data:   3/4/2016
clear all;
close all;
CT_file='AAAP_MMy-NaF_1-00_T5^^^^_CT.dcm';
PET_file='AAAP_MMy-NaF_1-00_T5^^^^_PT.dcm';
marker_init=zeros(170,170,252);
marker_array=add_marker(marker_init);
[m,n,p] = size(marker_array);
[xm,ym,zm]= meshgrid(1:m,1:n,1:p);
[fm,vm,cm] = isosurface(xm,ym,zm,marker_array,10,xm);
scale=5.0;
vm=vm.*scale;
stlwrite('Marker.stl',fm,vm,'mode','ascii');
plotct(vm,fm,cm);
marker_init_1=zeros(170,170,252);
marker_array_1=add_marker_1(marker_init_1);
[m,n,p] = size(marker_array_1);
[xm1,ym1,zm1]= meshgrid(1:m,1:n,1:p);
[fm1,vm1,cm1] = isosurface(xm1,ym1,zm1,marker_array_1,10,xm1);
vm1=vm1.*scale;
stlwrite('Marker_1.stl',fm1,vm1,'mode','ascii');
plotct(vm1,fm1,cm1);
CT_bone=threshold_bone(CT_file,-550);
CT_bone=flipdim(CT_bone,3);
[m,n,p] = size(CT_bone);
[x,y,z]= meshgrid(1:m,1:n,1:p);
[f,v,c] = isosurface(x,y,z,CT_bone,10,x);
v=v.*scale; %Scaling factor for the CT data.
filename='Skull_outside.stl';
[SkullF,SkullV,SkullC]=stl_file_read(filename);
wr=0.01;
[Ricp, Ticp, ER, t] = icp(v', SkullV',70,'Matching', 'kDtree',
'Minimize','point','Extrapolation', true,'WorstRejection',wr);
[SkullF,SkullV,SkullC]=transform_apply(filename,SkullV,Ricp, Ticp);
plotct(v,f,c);
hold on;
plot_segment(SkullF,SkullV,SkullC,filename,1,0,0);
display('C1-C2');
filename='C3-C7.stl';
[C3C7F,C3C7V,C3C7C]=transform_apply_primary(filename,Ricp,Ticp);
filename='T1-T3.stl';
[T1T3F,T1T3V,T1T3C]=transform_apply_primary(filename,Ricp,Ticp);
filename='T4-T8.stl';
[T4T8F,T4T8V,T4T8C]=transform_apply_primary(filename,Ricp,Ticp);
filename='T9-T13.stl';
[T9T13F,T9T13V,T9T13C]=transform_apply_primary(filename,Ricp,Ticp);
filename='L1.stl';
[L1F,L1V,L1C]=transform_apply_primary(filename,Ricp,Ticp);
filename='L2.stl';
```

```
[L2F,L2V,L2C]=transform_apply_primary(filename,Ricp,Ticp);
filename='L3.stl';
[L3F,L3V,L3C]=transform_apply_primary(filename,Ricp,Ticp);
filename='L4.stl';
[L4F,L4V,L4C]=transform_apply_primary(filename,Ricp,Ticp);
filename='L5.stl';
[L5F,L5V,L5C]=transform_apply_primary(filename,Ricp,Ticp);
filename='L6.stl';
[L6F,L6V,L6C]=transform_apply_primary(filename,Ricp,Ticp);
filename='S1.stl';
[S1F,S1V,S1C]=transform_apply_primary(filename,Ricp,Ticp);
filename='S2.stl';
[S2F,S2V,S2C]=transform_apply_primary(filename,Ricp,Ticp);
filename='S3.stl';
[S3F,S3V,S3C]=transform_apply_primary(filename,Ricp,Ticp);
filename='S4.stl';
[S4F,S4V,S4C]=transform_apply_primary(filename,Ricp,Ticp);
filename='CA-1.stl';
[CA1F,CA1V,CA1C]=transform_apply_primary(filename,Ricp,Ticp);
filename='CA-2.stl';
[CA2F,CA2V,CA2C]=transform_apply_primary(filename,Ricp,Ticp);
filename='CA-3.stl';
[CA3F,CA3V,CA3C]=transform_apply_primary(filename,Ricp,Ticp);
filename='CA-4.stl';
[CA4F,CA4V,CA4C]=transform_apply_primary(filename,Ricp,Ticp);
filename='CA-5.stl';
[CA5F,CA5V,CA5C]=transform_apply_primary(filename,Ricp,Ticp);

filename='C1-C2.stl';
[C1C2F,C1C2V,C1C2C]=transform_apply_primary(filename,Ricp,Ticp);
[Ricp Ticp ER t] = icp(v', C1C2V', 40,'Matching', 'kDtree',
'Minimize','point','WorstRejection',wr);
[C1C2F,C1C2V,C1C2C]=transform_apply_update(filename,C1C2V,Ricp,Ticp);
plot_segment(C1C2F,C1C2V,C1C2C,filename,1,1,0);
[v]=Update_ct(C1C2V,v);
display('C3-C7');
filename='T1-T3.stl';
[T1T3F,T1T3V,T1T3C]=transform_apply_update(filename,T1T3V,Ricp,Ticp);
filename='T4-T8.stl';
[T4T8F,T4T8V,T4T8C]=transform_apply_update(filename,T4T8V,Ricp,Ticp);
filename='T9-T13.stl';
[T9T13F,T9T13V,T9T13C]=transform_apply_update(filename,T9T13V,Ricp,Ticp);
filename='L1.stl';
[L1F,L1V,L1C]=transform_apply_update(filename,L1V,Ricp,Ticp);
filename='L2.stl';
[L2F,L2V,L2C]=transform_apply_update(filename,L2V,Ricp,Ticp);
filename='L3.stl';
[L3F,L3V,L3C]=transform_apply_update(filename,L3V,Ricp,Ticp);
filename='L4.stl';
[L4F,L4V,L4C]=transform_apply_update(filename,L4V,Ricp,Ticp);
filename='L5.stl';
[L5F,L5V,L5C]=transform_apply_update(filename,L5V,Ricp,Ticp);
filename='L6.stl';
[L6F,L6V,L6C]=transform_apply_update(filename,L6V,Ricp,Ticp);
filename='S1.stl';
[S1F,S1V,S1C]=transform_apply_update(filename,S1V,Ricp,Ticp);
filename='S2.stl';
```

```
[S2F,S2V,S2C]=transform_apply_update(filename,S2V,Ricp,Ticp);
filename='S3.stl';
[S3F,S3V,S3C]=transform_apply_update(filename,S3V,Ricp,Ticp);
filename='S4.stl';
[S4F,S4V,S4C]=transform_apply_update(filename,S4V,Ricp,Ticp);
filename='CA-1.stl';
[CA1F,CA1V,CA1C]=transform_apply_update(filename,CA1V,Ricp,Ticp);
filename='CA-2.stl';
[CA2F,CA2V,CA2C]=transform_apply_update(filename,CA2V,Ricp,Ticp);
filename='CA-3.stl';
[CA3F,CA3V,CA3C]=transform_apply_update(filename,CA3V,Ricp,Ticp);
filename='CA-4.stl';
[CA4F,CA4V,CA4C]=transform_apply_update(filename,CA4V,Ricp,Ticp);
filename='CA-5.stl';
[CA5F,CA5V,CA5C]=transform_apply_update(filename,CA5V,Ricp,Ticp);
filename='C3-C7.stl';
[C3C7F,C3C7V,C3C7C]=transform_apply_update(filename,C3C7V,Ricp,Ticp);
[Ricp Ticp ER t] = icp(v', C3C7V', 40,'Matching', 'kDtree',
'Minimize','point','WorstRejection',wr);
[C3C7F,C3C7V,C3C7C]=transform_apply_update(filename,C3C7V,Ricp,Ticp);
plot_segment(C3C7F,C3C7V,C3C7C,filename,0,1,1);

[v]=Update_ct(C3C7V,v);
display('T1-T3');
filename='T4-T8.stl';
[T4T8F,T4T8V,T4T8C]=transform_apply_update(filename,T4T8V,Ricp,Ticp);
filename='T9-T13.stl';
[T9T13F,T9T13V,T9T13C]=transform_apply_update(filename,T9T13V,Ricp,Ticp);
filename='L1.stl';
[L1F,L1V,L1C]=transform_apply_update(filename,L1V,Ricp,Ticp);
filename='L2.stl';
[L2F,L2V,L2C]=transform_apply_update(filename,L2V,Ricp,Ticp);
filename='L3.stl';
[L3F,L3V,L3C]=transform_apply_update(filename,L3V,Ricp,Ticp);
filename='L4.stl';
[L4F,L4V,L4C]=transform_apply_update(filename,L4V,Ricp,Ticp);
filename='L5.stl';
[L5F,L5V,L5C]=transform_apply_update(filename,L5V,Ricp,Ticp);
filename='L6.stl';
[L6F,L6V,L6C]=transform_apply_update(filename,L6V,Ricp,Ticp);
filename='S1.stl';
[S1F,S1V,S1C]=transform_apply_update(filename,S1V,Ricp,Ticp);
filename='S2.stl';
[S2F,S2V,S2C]=transform_apply_update(filename,S2V,Ricp,Ticp);
filename='S3.stl';
[S3F,S3V,S3C]=transform_apply_update(filename,S3V,Ricp,Ticp);
filename='S4.stl';
[S4F,S4V,S4C]=transform_apply_update(filename,S4V,Ricp,Ticp);
filename='CA-1.stl';
[CA1F,CA1V,CA1C]=transform_apply_update(filename,CA1V,Ricp,Ticp);
filename='CA-2.stl';
[CA2F,CA2V,CA2C]=transform_apply_update(filename,CA2V,Ricp,Ticp);
filename='CA-3.stl';
[CA3F,CA3V,CA3C]=transform_apply_update(filename,CA3V,Ricp,Ticp);
filename='CA-4.stl';
[CA4F,CA4V,CA4C]=transform_apply_update(filename,CA4V,Ricp,Ticp);
filename='CA-5.stl';
```

```
[CA5F,CA5V,CA5C]=transform_apply_update(filename,CA5V,Ricp,Ticp);
filename='T1-T3.stl';
[T1T3F,T1T3V,T1T3C]=transform_apply_update(filename,T1T3V,Ricp,Ticp);
[Ricp Ticp ER t] = icp(v', T1T3V', 40,'Matching', 'kDtree',
'Minimize','point','WorstRejection',wr);
[T1T3F,T1T3V,T1T3C]=transform_apply_update(filename,T1T3V,Ricp,Ticp);
plot_segment(T1T3F,T1T3V,T1T3C,filename,0,0,1);

display('T4-T8');
[v]=Update_ct(T1T3V,v);
filename='T9-T13.stl';
[T9T13F,T9T13V,T9T13C]=transform_apply_update(filename,T9T13V,Ricp,Ticp);
filename='L1.stl';
[L1F,L1V,L1C]=transform_apply_update(filename,L1V,Ricp,Ticp);
filename='L2.stl';
[L2F,L2V,L2C]=transform_apply_update(filename,L2V,Ricp,Ticp);
filename='L3.stl';
[L3F,L3V,L3C]=transform_apply_update(filename,L3V,Ricp,Ticp);
filename='L4.stl';
[L4F,L4V,L4C]=transform_apply_update(filename,L4V,Ricp,Ticp);
filename='L5.stl';
[L5F,L5V,L5C]=transform_apply_update(filename,L5V,Ricp,Ticp);
filename='L6.stl';
[L6F,L6V,L6C]=transform_apply_update(filename,L6V,Ricp,Ticp);
filename='S1.stl';
[S1F,S1V,S1C]=transform_apply_update(filename,S1V,Ricp,Ticp);
filename='S2.stl';
[S2F,S2V,S2C]=transform_apply_update(filename,S2V,Ricp,Ticp);
filename='S3.stl';
[S3F,S3V,S3C]=transform_apply_update(filename,S3V,Ricp,Ticp);
filename='S4.stl';
[S4F,S4V,S4C]=transform_apply_update(filename,S4V,Ricp,Ticp);
filename='CA-1.stl';
[CA1F,CA1V,CA1C]=transform_apply_update(filename,CA1V,Ricp,Ticp);
filename='CA-2.stl';
[CA2F,CA2V,CA2C]=transform_apply_update(filename,CA2V,Ricp,Ticp);
filename='CA-3.stl';
[CA3F,CA3V,CA3C]=transform_apply_update(filename,CA3V,Ricp,Ticp);
filename='CA-4.stl';
[CA4F,CA4V,CA4C]=transform_apply_update(filename,CA4V,Ricp,Ticp);
filename='CA-5.stl';
[CA5F,CA5V,CA5C]=transform_apply_update(filename,CA5V,Ricp,Ticp);
filename='T4-T8.stl';
[T4T8F,T4T8V,T4T8C]=transform_apply_update(filename,T4T8V,Ricp,Ticp);
[Ricp Ticp ER t] = icp(v', T4T8V', 40,'Matching', 'kDtree',
'Minimize','point','WorstRejection',wr);
[T4T8F,T4T8V,T4T8C]=transform_apply_update(filename,T4T8V,Ricp,Ticp);
plot_segment(T4T8F,T4T8V,T4T8C,filename,1,0,1);

display('T9-T13');
[v]=Update_ct(T4T8V,v);
filename='L1.stl';
[L1F,L1V,L1C]=transform_apply_update(filename,L1V,Ricp,Ticp);
filename='L2.stl';
[L2F,L2V,L2C]=transform_apply_update(filename,L2V,Ricp,Ticp);
filename='L3.stl';
[L3F,L3V,L3C]=transform_apply_update(filename,L3V,Ricp,Ticp);
```

```
filename='L4.stl';
[L4F,L4V,L4C]=transform_apply_update(filename,L4V,Ricp,Ticp);
filename='L5.stl';
[L5F,L5V,L5C]=transform_apply_update(filename,L5V,Ricp,Ticp);
filename='L6.stl';
[L6F,L6V,L6C]=transform_apply_update(filename,L6V,Ricp,Ticp);
filename='S1.stl';
[S1F,S1V,S1C]=transform_apply_update(filename,S1V,Ricp,Ticp);
filename='S2.stl';
[S2F,S2V,S2C]=transform_apply_update(filename,S2V,Ricp,Ticp);
filename='S3.stl';
[S3F,S3V,S3C]=transform_apply_update(filename,S3V,Ricp,Ticp);
filename='S4.stl';
[S4F,S4V,S4C]=transform_apply_update(filename,S4V,Ricp,Ticp);
filename='CA-1.stl';
[CA1F,CA1V,CA1C]=transform_apply_update(filename,CA1V,Ricp,Ticp);
filename='CA-2.stl';
[CA2F,CA2V,CA2C]=transform_apply_update(filename,CA2V,Ricp,Ticp);
filename='CA-3.stl';
[CA3F,CA3V,CA3C]=transform_apply_update(filename,CA3V,Ricp,Ticp);
filename='CA-4.stl';
[CA4F,CA4V,CA4C]=transform_apply_update(filename,CA4V,Ricp,Ticp);
filename='CA-5.stl';
[CA5F,CA5V,CA5C]=transform_apply_update(filename,CA5V,Ricp,Ticp);
filename='T9-T13.stl';
[T9T13F,T9T13V,T9T13C]=transform_apply_update(filename,T9T13V,Ricp,Ticp);
[Ricp Ticp ER t] = icp(v', T9T13V', 40,'Matching', 'kDtree',
'Minimize','point','WorstRejection',wr);
[T9T13F,T9T13V,T9T13C]=transform_apply_update(filename,T9T13V,Ricp,Ticp);
plot_segment(T9T13F,T9T13V,T9T13C,filename,0,0,1);

[v]=Update_ct(T9T13V,v);
display('L1');
filename='L2.stl';
[L2F,L2V,L2C]=transform_apply_update(filename,L2V,Ricp,Ticp);
filename='L3.stl';
[L3F,L3V,L3C]=transform_apply_update(filename,L3V,Ricp,Ticp);
filename='L4.stl';
[L4F,L4V,L4C]=transform_apply_update(filename,L4V,Ricp,Ticp);
filename='L5.stl';
[L5F,L5V,L5C]=transform_apply_update(filename,L5V,Ricp,Ticp);
filename='L6.stl';
[L6F,L6V,L6C]=transform_apply_update(filename,L6V,Ricp,Ticp);
filename='S1.stl';
[S1F,S1V,S1C]=transform_apply_update(filename,S1V,Ricp,Ticp);
filename='S2.stl';
[S2F,S2V,S2C]=transform_apply_update(filename,S2V,Ricp,Ticp);
filename='S3.stl';
[S3F,S3V,S3C]=transform_apply_update(filename,S3V,Ricp,Ticp);
filename='S4.stl';
[S4F,S4V,S4C]=transform_apply_update(filename,S4V,Ricp,Ticp);
filename='CA-1.stl';
[CA1F,CA1V,CA1C]=transform_apply_update(filename,CA1V,Ricp,Ticp);
filename='CA-2.stl';
[CA2F,CA2V,CA2C]=transform_apply_update(filename,CA2V,Ricp,Ticp);
filename='CA-3.stl';
[CA3F,CA3V,CA3C]=transform_apply_update(filename,CA3V,Ricp,Ticp);
```

```
filename='CA-4.stl';
[CA4F,CA4V,CA4C]=transform_apply_update(filename,CA4V,Ricp,Ticp);
filename='CA-5.stl';
[CA5F,CA5V,CA5C]=transform_apply_update(filename,CA5V,Ricp,Ticp);
filename='L1.stl';
[L1F,L1V,L1C]=transform_apply_update(filename,L1V,Ricp,Ticp);
[Ricp Ticp ER t] = icp(v', L1V', 40,'Matching', 'kDtree',
'Minimize','point','WorstRejection',wr);
[L1F,L1V,L1C]=transform_apply_update(filename,L1V,Ricp,Ticp);
plot_segment(L1F,L1V,L1C,filename,1,1,0);

[v]=Update_ct(L1V,v);
display('L2');
filename='L3.stl';
[L3F,L3V,L3C]=transform_apply_update(filename,L3V,Ricp,Ticp);
filename='L4.stl';
[L4F,L4V,L4C]=transform_apply_update(filename,L4V,Ricp,Ticp);
filename='L5.stl';
[L5F,L5V,L5C]=transform_apply_update(filename,L5V,Ricp,Ticp);
filename='L6.stl';
[L6F,L6V,L6C]=transform_apply_update(filename,L6V,Ricp,Ticp);
filename='S1.stl';
[S1F,S1V,S1C]=transform_apply_update(filename,S1V,Ricp,Ticp);
filename='S2.stl';
[S2F,S2V,S2C]=transform_apply_update(filename,S2V,Ricp,Ticp);
filename='S3.stl';
[S3F,S3V,S3C]=transform_apply_update(filename,S3V,Ricp,Ticp);
filename='S4.stl';
[S4F,S4V,S4C]=transform_apply_update(filename,S4V,Ricp,Ticp);
filename='CA-1.stl';
[CA1F,CA1V,CA1C]=transform_apply_update(filename,CA1V,Ricp,Ticp);
filename='CA-2.stl';
[CA2F,CA2V,CA2C]=transform_apply_update(filename,CA2V,Ricp,Ticp);
filename='CA-3.stl';
[CA3F,CA3V,CA3C]=transform_apply_update(filename,CA3V,Ricp,Ticp);
filename='CA-4.stl';
[CA4F,CA4V,CA4C]=transform_apply_update(filename,CA4V,Ricp,Ticp);
filename='CA-5.stl';
[CA5F,CA5V,CA5C]=transform_apply_update(filename,CA5V,Ricp,Ticp);
filename='L2.stl';
[L2F,L2V,L2C]=transform_apply_update(filename,L2V,Ricp,Ticp);
[Ricp Ticp ER t] = icp(v', L2V', 40,'Matching', 'kDtree',
'Minimize','point','WorstRejection',wr);
[L2F,L2V,L2C]=transform_apply_update(filename,L2V,Ricp,Ticp);
plot_segment(L2F,L2V,L2C,filename,0,1,0);

[v]=Update_ct(L2V,v);
display('L3');
filename='L4.stl';
[L4F,L4V,L4C]=transform_apply_update(filename,L4V,Ricp,Ticp);
filename='L5.stl';
[L5F,L5V,L5C]=transform_apply_update(filename,L5V,Ricp,Ticp);
filename='L6.stl';
[L6F,L6V,L6C]=transform_apply_update(filename,L6V,Ricp,Ticp);
filename='S1.stl';
[S1F,S1V,S1C]=transform_apply_update(filename,S1V,Ricp,Ticp);
filename='S2.stl';
```

```
[S2F,S2V,S2C]=transform_apply_update(filename,S2V,Ricp,Ticp);
filename='S3.stl';
[S3F,S3V,S3C]=transform_apply_update(filename,S3V,Ricp,Ticp);
filename='S4.stl';
[S4F,S4V,S4C]=transform_apply_update(filename,S4V,Ricp,Ticp);
filename='CA-1.stl';
[CA1F,CA1V,CA1C]=transform_apply_update(filename,CA1V,Ricp,Ticp);
filename='CA-2.stl';
[CA2F,CA2V,CA2C]=transform_apply_update(filename,CA2V,Ricp,Ticp);
filename='CA-3.stl';
[CA3F,CA3V,CA3C]=transform_apply_update(filename,CA3V,Ricp,Ticp);
filename='CA-4.stl';
[CA4F,CA4V,CA4C]=transform_apply_update(filename,CA4V,Ricp,Ticp);
filename='CA-5.stl';
[CA5F,CA5V,CA5C]=transform_apply_update(filename,CA5V,Ricp,Ticp);
filename='L3.stl';
[L3F,L3V,L3C]=transform_apply_update(filename,L3V,Ricp,Ticp);
[Ricp Ticp ER t] = icp(v', L3V', 40,'Matching', 'kDtree',
'Minimize','point','WorstRejection',wr);
[L3F,L3V,L3C]=transform_apply_update(filename,L3V,Ricp,Ticp);
plot_segment(L3F,L3V,L3C,filename,1,0,0);

display('L4');
[v]=Update_ct(L3V,v);
filename='L5.stl';
[L5F,L5V,L5C]=transform_apply_update(filename,L5V,Ricp,Ticp);
filename='L6.stl';
[L6F,L6V,L6C]=transform_apply_update(filename,L6V,Ricp,Ticp);
filename='S1.stl';
[S1F,S1V,S1C]=transform_apply_update(filename,S1V,Ricp,Ticp);
filename='S2.stl';
[S2F,S2V,S2C]=transform_apply_update(filename,S2V,Ricp,Ticp);
filename='S3.stl';
[S3F,S3V,S3C]=transform_apply_update(filename,S3V,Ricp,Ticp);
filename='S4.stl';
[S4F,S4V,S4C]=transform_apply_update(filename,S4V,Ricp,Ticp);
filename='CA-1.stl';
[CA1F,CA1V,CA1C]=transform_apply_update(filename,CA1V,Ricp,Ticp);
filename='CA-2.stl';
[CA2F,CA2V,CA2C]=transform_apply_update(filename,CA2V,Ricp,Ticp);
filename='CA-3.stl';
[CA3F,CA3V,CA3C]=transform_apply_update(filename,CA3V,Ricp,Ticp);
filename='CA-4.stl';
[CA4F,CA4V,CA4C]=transform_apply_update(filename,CA4V,Ricp,Ticp);
filename='CA-5.stl';
[CA5F,CA5V,CA5C]=transform_apply_update(filename,CA5V,Ricp,Ticp);
filename='L4.stl';
[L4F,L4V,L4C]=transform_apply_update(filename,L4V,Ricp,Ticp);
[Ricp Ticp ER t] = icp(v', L4V', 40,'Matching', 'kDtree',
'Minimize','point','WorstRejection',wr);
[L4F,L4V,L4C]=transform_apply_update(filename,L4V,Ricp,Ticp);
plot_segment(L4F,L4V,L4C,filename,0,1,1);

display('L5');
[v]=Update_ct(L4V,v);
filename='L6.stl';
[L6F,L6V,L6C]=transform_apply_update(filename,L6V,Ricp,Ticp);
```

```
filename='S1.stl';
[S1F,S1V,S1C]=transform_apply_update(filename,S1V,Ricp,Ticp);
filename='S2.stl';
[S2F,S2V,S2C]=transform_apply_update(filename,S2V,Ricp,Ticp);
filename='S3.stl';
[S3F,S3V,S3C]=transform_apply_update(filename,S3V,Ricp,Ticp);
filename='S4.stl';
[S4F,S4V,S4C]=transform_apply_update(filename,S4V,Ricp,Ticp);
filename='CA-1.stl';
[CA1F,CA1V,CA1C]=transform_apply_update(filename,CA1V,Ricp,Ticp);
filename='CA-2.stl';
[CA2F,CA2V,CA2C]=transform_apply_update(filename,CA2V,Ricp,Ticp);
filename='CA-3.stl';
[CA3F,CA3V,CA3C]=transform_apply_update(filename,CA3V,Ricp,Ticp);
filename='CA-4.stl';
[CA4F,CA4V,CA4C]=transform_apply_update(filename,CA4V,Ricp,Ticp);
filename='CA-5.stl';
[CA5F,CA5V,CA5C]=transform_apply_update(filename,CA5V,Ricp,Ticp);
filename='L5.stl';
[L5F,L5V,L5C]=transform_apply_update(filename,L5V,Ricp,Ticp);
[Ricp Ticp ER t] = icp(v', L5V', 40,'Matching', 'kDtree',
'Minimize','point','WorstRejection',wr);
[L5F,L5V,L5C]=transform_apply_update(filename,L5V,Ricp,Ticp);
plot_segment(L5F,L5V,L5C,filename,0,0,1);

display('L6');
[v]=Update_ct(L5V,v);
filename='S1.stl';
[S1F,S1V,S1C]=transform_apply_update(filename,S1V,Ricp,Ticp);
filename='S2.stl';
[S2F,S2V,S2C]=transform_apply_update(filename,S2V,Ricp,Ticp);
filename='S3.stl';
[S3F,S3V,S3C]=transform_apply_update(filename,S3V,Ricp,Ticp);
filename='S4.stl';
[S4F,S4V,S4C]=transform_apply_update(filename,S4V,Ricp,Ticp);
filename='CA-1.stl';
[CA1F,CA1V,CA1C]=transform_apply_update(filename,CA1V,Ricp,Ticp);
filename='CA-2.stl';
[CA2F,CA2V,CA2C]=transform_apply_update(filename,CA2V,Ricp,Ticp);
filename='CA-3.stl';
[CA3F,CA3V,CA3C]=transform_apply_update(filename,CA3V,Ricp,Ticp);
filename='CA-4.stl';
[CA4F,CA4V,CA4C]=transform_apply_update(filename,CA4V,Ricp,Ticp);
filename='CA-5.stl';
[CA5F,CA5V,CA5C]=transform_apply_update(filename,CA5V,Ricp,Ticp);
filename='L6.stl';
[L6F,L6V,L6C]=transform_apply_update(filename,L6V,Ricp,Ticp);
[Ricp Ticp ER t] = icp(v', L6V', 40,'Matching', 'kDtree',
'Minimize','point','WorstRejection',wr);
[L6F,L6V,L6C]=transform_apply_update(filename,L6V,Ricp,Ticp);
plot_segment(L6F,L6V,L6C,filename,0,1,0);

display('S1');
[v]=Update_ct(L6V,v);
filename='S2.stl';
[S2F,S2V,S2C]=transform_apply_update(filename,S2V,Ricp,Ticp);
filename='S3.stl';
```

```
[S3F,S3V,S3C]=transform_apply_update(filename,S3V,Ricp,Ticp);
filename='S4.stl';
[S4F,S4V,S4C]=transform_apply_update(filename,S4V,Ricp,Ticp);
filename='CA-1.stl';
[CA1F,CA1V,CA1C]=transform_apply_update(filename,CA1V,Ricp,Ticp);
filename='CA-2.stl';
[CA2F,CA2V,CA2C]=transform_apply_update(filename,CA2V,Ricp,Ticp);
filename='CA-3.stl';
[CA3F,CA3V,CA3C]=transform_apply_update(filename,CA3V,Ricp,Ticp);
filename='CA-4.stl';
[CA4F,CA4V,CA4C]=transform_apply_update(filename,CA4V,Ricp,Ticp);
filename='CA-5.stl';
[CA5F,CA5V,CA5C]=transform_apply_update(filename,CA5V,Ricp,Ticp);
filename='S1.stl';
[S1F,S1V,S1C]=transform_apply_update(filename,S1V,Ricp,Ticp);
[Ricp Ticp ER t] = icp(v', S1V', 40,'Matching', 'kDtree',
'Minimize','point','WorstRejection',wr);
[S1F,S1V,S1C]=transform_apply_update(filename,S1V,Ricp,Ticp);
plot_segment(S1F,S1V,S1C,filename,0,1,1);

display('S2');
[v]=Update_ct(S1V,v);
filename='S3.stl';
[S3F,S3V,S3C]=transform_apply_update(filename,S3V,Ricp,Ticp);
filename='S4.stl';
[S4F,S4V,S4C]=transform_apply_update(filename,S4V,Ricp,Ticp);
filename='CA-1.stl';
[CA1F,CA1V,CA1C]=transform_apply_update(filename,CA1V,Ricp,Ticp);
filename='CA-2.stl';
[CA2F,CA2V,CA2C]=transform_apply_update(filename,CA2V,Ricp,Ticp);
filename='CA-3.stl';
[CA3F,CA3V,CA3C]=transform_apply_update(filename,CA3V,Ricp,Ticp);
filename='CA-4.stl';
[CA4F,CA4V,CA4C]=transform_apply_update(filename,CA4V,Ricp,Ticp);
filename='CA-5.stl';
[CA5F,CA5V,CA5C]=transform_apply_update(filename,CA5V,Ricp,Ticp);
filename='S2.stl';
[S2F,S2V,S2C]=transform_apply_update(filename,S2V,Ricp,Ticp);
[Ricp Ticp ER t] = icp(v', S2V', 40,'Matching', 'kDtree',
'Minimize','point','WorstRejection',wr);
[S2F,S2V,S2C]=transform_apply_update(filename,S2V,Ricp,Ticp);
plot_segment(S2F,S2V,S2C,filename,1,1,0);

display('S3');
[v]=Update_ct(S2V,v);
filename='S4.stl';
[S4F,S4V,S4C]=transform_apply_update(filename,S4V,Ricp,Ticp);
filename='CA-1.stl';
[CA1F,CA1V,CA1C]=transform_apply_update(filename,CA1V,Ricp,Ticp);
filename='CA-2.stl';
[CA2F,CA2V,CA2C]=transform_apply_update(filename,CA2V,Ricp,Ticp);
filename='CA-3.stl';
[CA3F,CA3V,CA3C]=transform_apply_update(filename,CA3V,Ricp,Ticp);
filename='CA-4.stl';
[CA4F,CA4V,CA4C]=transform_apply_update(filename,CA4V,Ricp,Ticp);
filename='CA-5.stl';
[CA5F,CA5V,CA5C]=transform_apply_update(filename,CA5V,Ricp,Ticp);
```

```
filename='S3.stl';
[S3F,S3V,S3C]=transform_apply_update(filename,S3V,Ricp,Ticp);
[Ricp Ticp ER t] = icp(v', S3V', 40,'Matching', 'kDtree',
'Minimize','point','WorstRejection',wr);
[S3F,S3V,S3C]=transform_apply_update(filename,S3V,Ricp,Ticp);
plot_segment(S3F,S3V,S3C,filename,1,0,1);

display('S4');
[v]=Update_ct(S3V,v);
filename='CA-1.stl';
[CA1F,CA1V,CA1C]=transform_apply_update(filename,CA1V,Ricp,Ticp);
filename='CA-2.stl';
[CA2F,CA2V,CA2C]=transform_apply_update(filename,CA2V,Ricp,Ticp);
filename='CA-3.stl';
[CA3F,CA3V,CA3C]=transform_apply_update(filename,CA3V,Ricp,Ticp);
filename='CA-4.stl';
[CA4F,CA4V,CA4C]=transform_apply_update(filename,CA4V,Ricp,Ticp);
filename='CA-5.stl';
[CA5F,CA5V,CA5C]=transform_apply_update(filename,CA5V,Ricp,Ticp);
filename='S4.stl';
[S4F,S4V,S4C]=transform_apply_update(filename,S4V,Ricp,Ticp);
[Ricp Ticp ER t] = icp(v', S4V', 40,'Matching', 'kDtree',
'Minimize','point','WorstRejection',wr);
[S4F,S4V,S4C]=transform_apply_update(filename,S4V,Ricp,Ticp);
plot_segment(S4F,S4V,S4C,filename,1,1,0);

display('CA1');
[v]=Update_ct(S4V,v);
filename='CA-2.stl';
[CA2F,CA2V,CA2C]=transform_apply_update(filename,CA2V,Ricp,Ticp);
filename='CA-3.stl';
[CA3F,CA3V,CA3C]=transform_apply_update(filename,CA3V,Ricp,Ticp);
filename='CA-4.stl';
[CA4F,CA4V,CA4C]=transform_apply_update(filename,CA4V,Ricp,Ticp);
filename='CA-5.stl';
[CA5F,CA5V,CA5C]=transform_apply_update(filename,CA5V,Ricp,Ticp);
filename='CA-1.stl';
[CA1F,CA1V,CA1C]=transform_apply_update(filename,CA1V,Ricp,Ticp);
[Ricp Ticp ER t] = icp(v', CA1V', 40,'Matching', 'kDtree',
'Minimize','point','WorstRejection',wr);
[CA1F,CA1V,CA1C]=transform_apply_update(filename,CA1V,Ricp,Ticp);
plot_segment(CA1F,CA1V,CA1C,filename,1,0,1);
display('CA2');
[v]=Update_ct(CA1V,v);
filename='CA-3.stl';
[CA3F,CA3V,CA3C]=transform_apply_update(filename,CA3V,Ricp,Ticp);
filename='CA-4.stl';
[CA4F,CA4V,CA4C]=transform_apply_update(filename,CA4V,Ricp,Ticp);
filename='CA-5.stl';
[CA5F,CA5V,CA5C]=transform_apply_update(filename,CA5V,Ricp,Ticp);
filename='CA-2.stl';
[CA2F,CA2V,CA2C]=transform_apply_update(filename,CA2V,Ricp,Ticp);
[Ricp Ticp ER t] = icp(v', CA2V', 40,'Matching', 'kDtree',
'Minimize','point','WorstRejection',wr);
[CA2F,CA2V,CA2C]=transform_apply_update(filename,CA2V,Ricp,Ticp);
plot_segment(CA2F,CA2V,CA2C,filename,0,1,0);
stlwrite('Skull_roi.stl',SkullF,SkullV,'mode','ascii');
```

```
stlwrite('C1C2_roi.stl',C1C2F,C1C2V,'mode','ascii');
stlwrite('C3C7_roi.stl',C3C7F,C3C7V,'mode','ascii');
stlwrite('T1T3_roi.stl',T1T3F,T1T3V,'mode','ascii');
stlwrite('T4T8_roi.stl',T4T8F,T4T8V,'mode','ascii');
stlwrite('T9T13_roi.stl',T9T13F,T9T13V,'mode','ascii');
stlwrite('L1_roi.stl',L1F,L1V,'mode','ascii');
stlwrite('L2_roi.stl',L2F,L2V,'mode','ascii');
stlwrite('L3_roi.stl',L3F,L3V,'mode','ascii');
stlwrite('L4_roi.stl',L4F,L4V,'mode','ascii');
stlwrite('L5_roi.stl',L5F,L5V,'mode','ascii');
stlwrite('L6_roi.stl',L6F,L6V,'mode','ascii');
stlwrite('S1_roi.stl',S1F,S1V,'mode','ascii');
stlwrite('S2_roi.stl',S2F,S2V,'mode','ascii');
stlwrite('S3_roi.stl',S3F,S3V,'mode','ascii');
stlwrite('S4_roi.stl',S4F,S4V,'mode','ascii');
stlwrite('CA1_roi.stl',CA1F,CA1V,'mode','ascii');
stlwrite('CA2_roi.stl',CA2F,CA2V,'mode','ascii');
stlwrite('CA3_roi.stl',CA3F,CA3V,'mode','ascii');
stlwrite('CA4_roi.stl',CA4F,CA4V,'mode','ascii');
L1_list=Update_flist('L1.stl');
L2_list=Update_flist('L2.stl');
L3_list=Update_flist('L3.stl');
L4_list=Update_flist('L4.stl');
L5_list=Update_flist('L5.stl');
L6_list=Update_flist('L6.stl');
PET_PixelData=dicomread(PET_file);
PET_HeaderInformation = dicominfo(PET_file);
PET_data = ((squeeze(PET_PixelData)));
PET_PixelData = int16(PET_PixelData);
PET_data=im2double(PET_data);
sliceomatic(PET_data);
size(PET_data)

CT_bone=threshold_bone(CT_file,-550);
 [m,n,p] = size(CT_bone)
CT_bone=add_marker(CT_bone);
o1=CT_bone(:,85,:);
c=squeeze(o1);
figure(1);
imagesc(c);
hold on;
o=PET_data(:,45,:);
c1=squeeze(o);
figure(2);
imagesc(c1);

figure(3);
imshowpair(c1, c);
title('Unregistered');

[optimizer, metric]=imregconfig('multimodal');
movingRegisterDefault=imregister(c1, c, 'affine', optimizer, metric);
figure(4);
imshowpair(movingRegisterDefault, c);
title('A:Default Registration');
```

```
disp(optimizer)
disp(metric)

optimizer.InitialRadius=optimizer.InitialRadius/5
movingRegisterDefault=imregister(c1, c, 'affine', optimizer, metric);
figure(5);
imshowpair(movingRegisterDefault, c);
title('A:Adjusted Initial Radius Registration');

optimizer.MaximumIterations=500;
movingRegisterDefault=imregister(c1, c, 'affine', optimizer, metric);
figure(6);
imshowpair(movingRegisterDefault, c);
title('A:Adjusted Initial Radius Registration, MaximumIterations=300');

optimizer.InitialRadius=optimizer.InitialRadius/1.2
movingRegisterDefault=imregister(c1, c, 'affine', optimizer, metric);
figure(7);
imshowpair(movingRegisterDefault, c);
title('A:Adjusted Initial Radius Registration, MaximumIterations=300');

tformSimilarity = imregtform(c1, c,'similarity',optimizer,metric);
Rfixed = imref2d(size(c));
movingRegisteredRigid = imwarp(c1,tformSimilarity,'OutputView',Rfixed);
figure(8);
imshowpair(movingRegisteredRigid, c);
title('C: Registration based on similarity transformation model.');

movingRegisteredAffineWithIC = imregister(c1,c,'affine',optimizer,metric,...
    'InitialTransformation',tformSimilarity);
figure(9);
imshowpair(movingRegisteredAffineWithIC,c);
title('D: Registration from affine model based on similarity initial
condition.');
size(c)
size(c1)
size(movingRegisteredAffineWithIC)
size(L1_list)
L1_Vert_sq=squeeze(sum(L1_list,1));
imagesc(L1_Vert_sq);
figure(10);
imshowpair(L1_Vert_sq, movingRegisteredAffineWithIC);
size(L1_Vert_sq)
size(movingRegisteredAffineWithIC)
[moby] = VOXELISE(170,170,252,'Marker.stl','xyz');
e=double(moby);
for i=1:170
        for j=1:170
            for k=1:252
                if (e(i,j,k)==-1)
                    e(i,j,k)=0;
                elseif(e(i,j,k)>0)
                    e(i,j,k)=1;
                end
            end
        end
```

```
end
figure(1);
imagesc(squeeze(sum(e,1)));
colormap(gray(256));
xlabel('Z-direction');
ylabel('X-direction');
axis equal tight;

%%Manually Draw ROI on PET
PET_PixelData=dicomread(PET_file);
PET_HeaderInformation = dicominfo(PET_file);
PET_data = ((squeeze(PET_PixelData)));
PET_PixelData = int16(PET_PixelData);
PET_data=im2double(PET_data);
sliceomatic(PET_data);
size(PET_data)
CT_bone=threshold_bone(CT_file,-550);
%CT_bone=flipdim(CT_bone,3);
[m,n,p] = size(CT_bone)
CT_bone=add_marker(CT_bone);
o1=CT_bone(:,80,:);
c=squeeze(o1);
figure(1);
imagesc(c);
hold on;
o=PET_data(:,45,:);
c1=squeeze(o);
figure(2);
imagesc(c1);

figure(3);
imshowpair(c1, c);
title('Unregistered');

[optimizer, metric]=imregconfig('multimodal');
movingRegisterDefault=imregister(c1, c, 'affine', optimizer, metric);
figure(4);
imshowpair(movingRegisterDefault, c);
title('A:Default Registration');

disp(optimizer)
disp(metric)

optimizer.InitialRadius=optimizer.InitialRadius/5
movingRegisterDefault=imregister(c1, c, 'affine', optimizer, metric);
figure(5);
imshowpair(movingRegisterDefault, c);
title('A:Adjusted Initial Radius Registration');

optimizer.MaximumIterations=500;
movingRegisterDefault=imregister(c1, c, 'affine', optimizer, metric);
figure(6);
imshowpair(movingRegisterDefault, c);
title('A:Adjusted Initial Radius Registration, MaximumIterations=300');

optimizer.InitialRadius=optimizer.InitialRadius/1.2
```

```
movingRegisterDefault=imregister(c1, c, 'affine', optimizer, metric);
figure(7);
imshowpair(movingRegisterDefault, c);
title('A:Adjusted Initial Radius Registration, MaximumIterations=300');

tformSimilarity = imregtform(c1, c,'similarity',optimizer,metric);
Rfixed = imref2d(size(c));
movingRegisteredRigid = imwarp(c1,tformSimilarity,'OutputView',Rfixed);
figure(8);
imshowpair(movingRegisteredRigid, c);
title('C: Registration based on similarity transformation model.');

movingRegisteredAffineWithIC = imregister(c1,c,'affine',optimizer,metric,...
    'InitialTransformation',tformSimilarity);
figure(9);
imshowpair(movingRegisteredAffineWithIC,c);
title('D: Registration from affine model based on similarity initial
condition.');
size(L1_list)
L1_Vert_sq=squeeze(sum(L1_list,1));
imagesc(L1_Vert_sq);
figure(10);
imshowpair(L1_Vert_sq, movingRegisteredAffineWithIC);
size(L1_Vert_sq)
size(movingRegisteredAffineWithIC)
ct_mip=MIP(CT_bone);
imshow(ct_mip);

PET_PixelData=dicomread(PET_file);
PET_HeaderInformation = dicominfo(PET_file);
PET_data=PET_PixelData*PET_HeaderInformation.RescaleSlope+PET_HeaderInformati
on.RescaleIntercept;
Reg_PET=Temp_PET(PET_file, CT_file, 30,65, L1_Vert_sq);
%Apply ROI
for i=1:n
    Reg_PET_L1(:,i,:)=squeeze(Reg_PET(:,i,:)).*im2bw(L1_Vert_sq,0.01);
end
L1_Bqml=PET_HeaderInformation.RescaleSlope*sum(sum(sum(Reg_PET_L1)))
L2_Vert_sq=squeeze(sum(L2_list,1));
for i=1:n
    Reg_PET_L2(:,i,:)=squeeze(Reg_PET(:,i,:)).*im2bw(L2_Vert_sq,0.01);
end
L2_Bqml=PET_HeaderInformation.RescaleSlope*sum(sum(sum(Reg_PET_L2)))
L3_Vert_sq=squeeze(sum(L3_list,1));
for i=1:n
    Reg_PET_L3(:,i,:)=squeeze(Reg_PET(:,i,:)).*im2bw(L3_Vert_sq,0.01);
end
L3_Bqml=PET_HeaderInformation.RescaleSlope*sum(sum(sum(Reg_PET_L3)))
L4_Vert_sq=squeeze(sum(L4_list,1));
for i=1:n
    Reg_PET_L4(:,i,:)=squeeze(Reg_PET(:,i,:)).*im2bw(L4_Vert_sq,0.01);
end
L4_Bqml=PET_HeaderInformation.RescaleSlope*sum(sum(sum(Reg_PET_L4)))
L5_Vert_sq=squeeze(sum(L5_list,1));
for i=1:n
    Reg_PET_L5(:,i,:)=squeeze(Reg_PET(:,i,:)).*im2bw(L5_Vert_sq,0.01);
end
```

```
L5_Bqml=PET_HeaderInformation.RescaleSlope*sum(sum(sum(Reg_PET_L5)))
```

add_marker.m

```
function [ CT_bone] = add_marker( CT_bone )
[a,b,c]=size(CT_bone);
max(max(max(CT_bone)));
CT_bone(1:10,1:10,1:10)=1000;
CT_bone(160:170, 160:170, 1:10)=1000;
CT_bone(1:10, 160:170, 1:10)=1000;
CT_bone(160:170,1:10,1:10)=1000;
CT_bone(160:170,1:10,242:252)=1000;
CT_bone(160:170,160:170,242:252)=1000;
CT_bone(1:10,1:10,242:252)=1000;
CT_bone(1:10,160:170,242:252)=1000;
end
```

threshold_bone.m

```
function [ CT_bone ] = threshold_bone( filename,x)

CT=dicomread(filename);
 CT=int16(squeeze(CT));
 CT_Header=dicominfo(filename);
 Rescale_Slope=CT_Header.RescaleSlope;
 Rescale_Intercept=CT_Header.RescaleIntercept;
 CT=CT*Rescale_Slope+Rescale_Intercept;
[l,m,n]=size(CT)
 CT_bone=zeros(size(CT));
for i=1:l
    for j=1:m
        for k=1:n
            if (CT(i,j,k)>x)
            CT_bone(i,j,k)=1000;
            elseif((CT(i,j,k)<x))
            CT_bone(i,j,k)=0;
            end
        end
    end
end
CT_bone(1:l,1:40,1:n)=0;
CT_bone(100:l,1:m,1:n)=0;
```

Transform_apply.m

```
function [F,V,C]=transform_apply(filename,V,Ricp,Ticp)

    [F,V1,C]=stl_file_read(filename);
    Dicp = Ricp * V' + repmat(Ticp, 1, size(V',2));
    V=Dicp';
end
```

plot_segment.m

```
function plot_segment( F,V,C,filename,c1,c2,c3)
[F,V1,C]=stl_file_read(filename);
O = patch('faces', F, 'vertices' ,V);
```

```
set(O, 'facec', 'flat');
set(O, 'FaceVertexCData', C);
set(O, 'facealpha',.4);
set(O, 'EdgeColor',[c1 c2 c3]);
light
daspect([1 1 1])
view(3);
xlabel('X'),ylabel('Y'),zlabel('Z');
title(['Imported CAD data from ' filename]);
xlim([0 800]);
ylim([0 800]);
zlim([0 1200]);
disp(('Complete moby mouse atlas'))
pause(1);
end
```

transform_apply_primary.m

```
function [F,V,C]=transform_apply_primary(filename,Ricp,Ticp)

    [F,V,C]=stl_file_read(filename);
    Dicp = Ricp * V' + repmat(Ticp, 1, size(V',2));
    V=Dicp';



end
```

transform_apply_update.m

```
function [F,C1C2V,C]=transform_apply_update(filename,C1C2V,Ricp,Ticp);

    [F,V1,C]=stl_file_read(filename);
    Dicp = Ricp * C1C2V' + repmat(Ticp, 1, size(C1C2V',2));
    C1C2V=Dicp';
end
```

algo_test.m

```
clear all;
close all;
R_model=moby_joint;           %Ribs
R_model.Heir_state=['C','C','3'];
R_model.Color_mat=[0 0 1];
R_model.filename='Ribs.stl';
R_model.parent_joint_cordinate=[0,0,100];
R_model.draw();
R_model.Vertices=R_model.Vertices';
R_model.Vertices = [R_model.Vertices(1,:); R_model.Vertices(2,:);
R_model.Vertices(3,:); ones(1,length(R_model.Vertices))];
k = tl(R_model.parent_joint_cordinate)*Rx(30)*R_model.Vertices;
set(R_model.object,'Vertices',k(1:3,:)');
drawnow;
R_model.Vertices=k(1:3,:)';
```

```
R_data=moby_joint;           %Ribs

R_data.Heir_state=['C','C','3'];
R_data.Color_mat=[0 1 0];
R_data.filename='Ribs.stl';
R_data.parent_joint_cordinate=[328.8,279.5,697];

R_data.draw();



    [Ricp Ticp ER t] = icp(R_model.Vertices', R_data.Vertices', 20,'Matching',
'kDtree', 'Minimize','plane','Extrapolation', true);
    Dicp = Ricp * R_data.Vertices' + repmat(Ticp, 1,
size(R_data.Vertices',2));
    R_data.Vertices=Dicp';
    cla(R_model.object);
    R_data.object = patch('faces', R_data.Faces, 'vertices' ,R_data.Vertices)

        set(R_data.object, 'Marker','h');
        set(R_data.object, 'facec', 'flat');
        set(R_data.object, 'FaceVertexCData', R_data.Color);
        set(R_data.object, 'facealpha',.4);
        set(R_data.object, 'EdgeColor',R_data.Color_mat);
        light
        daspect([1 1 1])
        view(3);
        xlabel('X'),ylabel('Y'),zlabel('Z');
        title(['Imported CAD data from ' R_data.filename]);
        drawnow
        disp(['Complete moby mouse atlas'])
        pause(1);
```

Joint_define.m

```
%Script to assign properties of each part like its position in heirarchy,
parent joint cordinate (point of 3D rotation),
%color matrix for each part, filename of each part.
%In the heirarchy name, first alphabet refers to the part position(up-->A,
%down-->B, Center-->C), Right/left(Right-->R, Left-->L, Center-->C), joint
%location(1-->skull to 6-->paws)

%Author: Vineeth Radhakrishnan
%Date: 3/10/2014

clear all;
close all;

SI=moby_joint;          %Skull Inside
SO=moby_joint;          %Skull Outside
S=moby_joint;           %Spine
SL=moby_joint;          %Scapula left
SR=moby_joint;          %Scapuka Right
```

```
R=moby_joint;           %Ribs
PL=moby_joint;          %Pelvis Left
PR=moby_joint;          %pelvis Right
UFL=moby_joint;         %Upper Forelmb Left
UFR=moby_joint;         %Upper Forelimb Right
UHL=moby_joint;         %Upper hindlimb Left
UHR=moby_joint;         %Upper hindlimb Right
LFL=moby_joint;         %Lower forelimb Left
LFR=moby_joint;         %Lower forelimb Right
LHL=moby_joint;         %Lower Hindlimb Left
LHR=moby_joint;         %Lower Hindlimb Right
FPL=moby_joint;         %Forepaw Left
FPR=moby_joint;         %orepaw Right
HPL=moby_joint;         %Hindpaw Left
HPR=moby_joint;         %Hindpaw Right
St=moby_joint;          %Sternum

Obj_array=[SI,SO,S,SL,SR,R,PL,PR,UFL,UFR,UHL,UHR,LFR,LHL,LHR,FPR,HPL,HPR,St,L
FL,FPL];
SI.Heir_state=['A','C','1'];
SI.Color_mat=[0 0 1];
SI.parent_joint_cordinate=[328.8,279.5,697];
SI.filename='Skull_inside.stl';

SO.Heir_state=['A','C','1'];
SO.Color_mat=[1 0 0];
SO.parent_joint_cordinate=[328.8,279.5,697];
SO.filename='Skull_outside.stl';

S.Heir_state=['A','C','2'];
S.Color_mat=[0 0 1];
S.parent_joint_cordinate=[328.8,279.5,697];
S.filename='Spine.stl';

SL.Heir_state=['A','L','3'];
SL.Color_mat=[1 0 0];
SL.parent_joint_cordinate=[157.1,154.6,842.2];
SL.filename='Scapula_left.stl';
SL.draw();

SR.Heir_state=['A','R','3'];
SR.Color_mat=[1 0 0];
SR.parent_joint_cordinate=[328.8,279.5,697];
SR.filename='Scapula_right.stl';

R.Heir_state=['C','C','3'];
R.Color_mat=[0 1 1];
R.filename='Ribs.stl';
R.parent_joint_cordinate=[328.8,279.5,697];

PL.Heir_state=['B','L','3'];
PL.Color_mat=[1 0 1];
PL.filename='Pelvis_left_part.stl';
PL.parent_joint_cordinate=[328.8,279.5,697];

PR.Heir_state=['B','R','3'];
```

```
PR.Color_mat=[1 0 1];
PR.filename='Pelvis_right_part.stl';
PR.parent_joint_cordinate=[328.8,279.5,697];

UFL.Heir_state=['A','L','4'];
UFL.Color_mat=[0 1 0];
UFL.filename='Upper_forelimb_left.stl';
UFL.parent_joint_cordinate=[127,221.5,869.8];
UFL.draw();

UFR.Heir_state=['A','R','4'];
UFR.Color_mat=[0 1 0];
UFR.filename='Upper_forelimb_right.stl';
UFR.parent_joint_cordinate=[328.8,279.5,697];

UHL.Heir_state=['B','L','4'];
UHL.Color_mat=[0 1 0];
UHL.filename='Upper_hindlimb_left.stl';
UHL.parent_joint_cordinate=[328.8,279.5,697];

UHR.Heir_state=['B','R','4'];
UHR.Color_mat=[0 1 0];
UHR.filename='Upper_hindlimb_right.stl';
UHR.parent_joint_cordinate=[328.8,279.5,697];

LFR.Heir_state=['A','R','5'];
LFR.Color_mat=[1 0 0];
LFR.filename='Lower_forelimb_right.stl';
LFR.parent_joint_cordinate=[328.8,279.5,697];

LHL.Heir_state=['B','L','5'];
LHL.Color_mat=[1 0 0];
LHL.filename='Lower_hindlimb_left.stl';
LHL.parent_joint_cordinate=[328.8,279.5,697];

LHR.Heir_state=['B','R','5'];
LHR.Color_mat=[1 0 0];
LHR.filename='Lower_hindlimb_right.stl';
LHR.parent_joint_cordinate=[328.8,279.5,697];

FPR.Heir_state=['A','R','6'];
FPR.Color_mat=[0 0 1];
FPR.parent_joint_type='BallnSocket';
FPR.parent_joint_cordinate=[328.8,279.5,697];
FPR.filename='Forepaw_right.stl';

HPL.Heir_state=['B','L','6'];
HPL.Color_mat=[0 0 1];
HPL.parent_joint_type='BallnSocket';
HPL.parent_joint_cordinate=[328.8,279.5,697];
HPL.filename='Hindpaw_left.stl';

HPR.Heir_state=['B','R','6'];
HPR.Color_mat=[0 0 1];
HPR.parent_joint_type='BallnSocket';
```

```
HPR.filename='Hindpaw_right.stl';
HPR.parent_joint_cordinate=[328.8,279.5,697];

LFL.Heir_state=['A','L','5'];
LFL.Color_mat=[1 0 0];
LFL.filename='Lower_forelimb_left.stl';
LFL.parent_joint_type='Hinge';
LFL.parent_joint_cordinate=[80.26,221.5,761.6];
LFL.draw();

FPL.Heir_state=['A','L','6'];
FPL.Color_mat=[0 0 1];
FPL.parent_joint_type='Hinge';
FPL.parent_joint_cordinate=[106.2,335.3,704.4];
FPL.filename='Forepaw_left.stl';
FPL.draw();

St.Heir_state=['C','C','4'];
St.Color_mat=[0 1 1];
St.filename='Sternum.stl';
St.parent_joint_cordinate=[328.8,279.5,697];

SL.rotate(Obj_array);
UFL.rotate(Obj_array);
FPL.rotate(Obj_array);
FPL.parent_joint_cordinate
LFL.rotate(Obj_array);
FPL.parent_joint_cordinate
FPL.rotate(Obj_array);
LFL.rotate(Obj_array);
```

Update_flist.m

```
function [ diff_image ] = Update_flist( filename )

switch filename
    case 'L1.stl'
    f_list_1 =
{'Skull_roi.stl','Marker.stl','C1C2_roi.stl','C3C7_roi.stl','T1T3_roi.stl','T
4T8_roi.stl','T9T13_roi_roi.stl','L1_roi.stl','L2_roi.stl','L3_roi.stl','L4_r
oi.stl','L5_roi.stl','L6_roi.stl','S1_roi.stl','S2_roi.stl','S3_roi.stl','S4_
roi.stl','CA1_roi.stl','CA2_roi.stl','CA3_roi.stl','CA4_roi.stl'};
    f_list_2 =
{'Skull_roi.stl','Marker_1.stl','C1C2_roi.stl','C3C7_roi.stl','T1T3_roi.stl',
'T4T8_roi.stl','T9T13_roi_roi.stl','L2_roi.stl','L3_roi.stl','L4_roi.stl','L5
_roi.stl','L6_roi.stl','S1_roi.stl','S2_roi.stl','S3_roi.stl','S4_roi.stl','C
A1_roi.stl','CA2_roi.stl','CA3_roi.stl','CA4_roi.stl'};
    e1=Volumetric_conv(f_list_1);
    e2=Volumetric_conv(f_list_2);
    diff_image=(e1-e2);
    %size(diff_image);
    for i=1:170
        for j=1:170
            for k=1:252
                if (diff_image(i,j,k)==-1)
                    diff_image(i,j,k)=0;
```

```
                elseif(diff_image(i,j,k)>0)
                    diff_image(i,j,k)=1;
                end
            end
        end
    end
figure(1);
imagesc(squeeze(sum(diff_image,1)));
colormap(gray(256));
xlabel('Z-direction');
ylabel('X-direction');
axis equal tight;

    case 'L2.stl'
    f_list_1 =
{'Skull_roi.stl','Marker.stl','C1C2_roi.stl','C3C7_roi.stl','T1T3_roi.stl','T
4T8_roi.stl','T9T13_roi_roi.stl','L1_roi.stl','L2_roi.stl','L3_roi.stl','L4_r
oi.stl','L5_roi.stl','L6_roi.stl','S1_roi.stl','S2_roi.stl','S3_roi.stl','S4_
roi.stl','CA1_roi.stl','CA2_roi.stl','CA3_roi.stl','CA4_roi.stl'};
    f_list_2 =
{'Skull_roi.stl','Marker_1.stl','C1C2_roi.stl','C3C7_roi.stl','T1T3_roi.stl',
'T4T8_roi.stl','T9T13_roi_roi.stl','L1_roi.stl','L3_roi.stl','L4_roi.stl','L5
_roi.stl','L6_roi.stl','S1_roi.stl','S2_roi.stl','S3_roi.stl','S4_roi.stl','C
A1_roi.stl','CA2_roi.stl','CA3_roi.stl','CA4_roi.stl'};
    e1=Volumetric_conv(f_list_1);
    e2=Volumetric_conv(f_list_2);
    diff_image=(e1-e2);
    %size(diff_image);
    for i=1:170
        for j=1:170
            for k=1:252
                if (diff_image(i,j,k)==-1)
                    diff_image(i,j,k)=0;
                end
            end
        end
    end

figure(2);
imagesc(squeeze(sum(diff_image,1)));
colormap(gray(256));
xlabel('Z-direction');
ylabel('X-direction');
axis equal tight;

 case 'L3.stl'
    f_list_1 =
{'Skull_roi.stl','Marker.stl','C1C2_roi.stl','C3C7_roi.stl','T1T3_roi.stl','T
4T8_roi.stl','T9T13_roi_roi.stl','L1_roi.stl','L2_roi.stl','L3_roi.stl','L4_r
oi.stl','L5_roi.stl','L6_roi.stl','S1_roi.stl','S2_roi.stl','S3_roi.stl','S4_
roi.stl','CA1_roi.stl','CA2_roi.stl','CA3_roi.stl','CA4_roi.stl'};
    f_list_2 =
{'Skull_roi.stl','Marker_1.stl','C1C2_roi.stl','C3C7_roi.stl','T1T3_roi.stl',
'T4T8_roi.stl','T9T13_roi_roi.stl','L1_roi.stl','L2_roi.stl','L4_roi.stl','L5
_roi.stl','L6_roi.stl','S1_roi.stl','S2_roi.stl','S3_roi.stl','S4_roi.stl','C
A1_roi.stl','CA2_roi.stl','CA3_roi.stl','CA4_roi.stl'};
    e1=Volumetric_conv(f_list_1);
```

```
    e2=Volumetric_conv(f_list_2);
    diff_image=(e1-e2);
    %size(diff_image);
    for i=1:170
        for j=1:170
            for k=1:252
                if (diff_image(i,j,k)==-1)
                    diff_image(i,j,k)=0;
                end
            end
        end
    end

figure(3);
imagesc(squeeze(sum(diff_image,1)));
colormap(gray(256));
xlabel('Z-direction');
ylabel('X-direction');
axis equal tight;

 case 'L4.stl'
    f_list_1 =
{'Skull_roi.stl','Marker.stl','C1C2_roi.stl','C3C7_roi.stl','T1T3_roi.stl','T
4T8_roi.stl','T9T13_roi_roi.stl','L1_roi.stl','L2_roi.stl','L3_roi.stl','L4_r
oi.stl','L5_roi.stl','L6_roi.stl','S1_roi.stl','S2_roi.stl','S3_roi.stl','S4_
roi.stl','CA1_roi.stl','CA2_roi.stl','CA3_roi.stl','CA4_roi.stl'};
    f_list_2 =
{'Skull_roi.stl','Marker_1.stl','C1C2_roi.stl','C3C7_roi.stl','T1T3_roi.stl',
'T4T8_roi.stl','T9T13_roi_roi.stl','L1_roi.stl','L2_roi.stl','L3_roi.stl','L5
_roi.stl','L6_roi.stl','S1_roi.stl','S2_roi.stl','S3_roi.stl','S4_roi.stl','C
A1_roi.stl','CA2_roi.stl','CA3_roi.stl','CA4_roi.stl'};
    e1=Volumetric_conv(f_list_1);
    e2=Volumetric_conv(f_list_2);
    diff_image=(e1-e2);
    %size(diff_image);
    for i=1:170
        for j=1:170
            for k=1:252
                if (diff_image(i,j,k)==-1)
                    diff_image(i,j,k)=0;
                end
            end
        end
    end
figure(4);
imagesc(squeeze(sum(diff_image,1)));
colormap(gray(256));
xlabel('Z-direction');
ylabel('X-direction');
axis equal tight;

 case 'L5.stl'
    f_list_1 =
{'Skull_roi.stl','Marker.stl','C1C2_roi.stl','C3C7_roi.stl','T1T3_roi.stl','T
4T8_roi.stl','T9T13_roi_roi.stl','L1_roi.stl','L2_roi.stl','L3_roi.stl','L4_r
oi.stl','L5_roi.stl','L6_roi.stl','S1_roi.stl','S2_roi.stl','S3_roi.stl','S4_
roi.stl','CA1_roi.stl','CA2_roi.stl','CA3_roi.stl','CA4_roi.stl'};
```

```
    f_list_2 =
{'Skull_roi.stl','Marker_1.stl','C1C2_roi.stl','C3C7_roi.stl','T1T3_roi.stl',
'T4T8_roi.stl','T9T13_roi_roi.stl','L1_roi.stl','L2_roi.stl','L3_roi.stl','L4
_roi.stl','L6_roi.stl','S1_roi.stl','S2_roi.stl','S3_roi.stl','S4_roi.stl','C
A1_roi.stl','CA2_roi.stl','CA3_roi.stl','CA4_roi.stl'};
    e1=Volumetric_conv(f_list_1);
    e2=Volumetric_conv(f_list_2);
    diff_image=(e1-e2);
    %size(diff_image);
    for i=1:170
        for j=1:170
            for k=1:252
                if (diff_image(i,j,k)==-1)
                    diff_image(i,j,k)=0;
                end
            end
        end
    end

figure(5);
imagesc(squeeze(sum(diff_image,1)));
colormap(gray(256));
xlabel('Z-direction');
ylabel('X-direction');
axis equal tight;

 case 'L6.stl'
    f_list_1 =
{'Skull_roi.stl','Marker.stl','C1C2_roi.stl','C3C7_roi.stl','T1T3_roi.stl','T
4T8_roi.stl','T9T13_roi_roi.stl','L1_roi.stl','L2_roi.stl','L3_roi.stl','L4_r
oi.stl','L5_roi.stl','L6_roi.stl','S1_roi.stl','S2_roi.stl','S3_roi.stl','S4_
roi.stl','CA1_roi.stl','CA2_roi.stl','CA3_roi.stl','CA4_roi.stl'};
    f_list_2 =
{'Skull_roi.stl','Marker_1.stl','C1C2_roi.stl','C3C7_roi.stl','T1T3_roi.stl',
'T4T8_roi.stl','T9T13_roi_roi.stl','L1_roi.stl','L2_roi.stl','L3_roi.stl','L4
_roi.stl','L5_roi.stl','S1_roi.stl','S2_roi.stl','S3_roi.stl','S4_roi.stl','C
A1_roi.stl','CA2_roi.stl','CA3_roi.stl','CA4_roi.stl'};
e1=Volumetric_conv(f_list_1);
    e2=Volumetric_conv(f_list_2);
    diff_image=(e1-e2);
    %size(diff_image);
    for i=1:170
        for j=1:170
            for k=1:252
                if (diff_image(i,j,k)==-1)
                    diff_image(i,j,k)=0;
                end
            end
        end
    end

figure(6);
imagesc(squeeze(sum(diff_image,1)));
colormap(gray(256));
xlabel('Z-direction');
ylabel('X-direction');
axis equal tight;
```

```
end
```

icp.m (Code inspired from Jakob Wilm & Hans Martin Kjer with modification (Copyright (c) 2012, Jakob Wilm & Hans Martin Kjer))

```
function [TR, TT, ER, t] = icp(q,p,varargin)

inp = inputParser;

inp.addRequired('q', @(x)isreal(x) && size(x,1) == 3);
inp.addRequired('p', @(x)isreal(x) && size(x,1) == 3);

inp.addOptional('iter', 10, @(x)x > 0 && x < 10^5);

inp.addParamValue('Boundary', [], @(x)size(x,1) == 1);

inp.addParamValue('EdgeRejection', false, @(x)islogical(x));

inp.addParamValue('Extrapolation', false, @(x)islogical(x));

validMatching = {'bruteForce','Delaunay','kDtree'};
inp.addParamValue('Matching', 'bruteForce',
@(x)any(strcmpi(x,validMatching)));

validMinimize = {'point','plane','lmapoint'};
inp.addParamValue('Minimize', 'point', @(x)any(strcmpi(x,validMinimize)));

inp.addParamValue('Normals', [], @(x)isreal(x) && size(x,1) == 3);

inp.addParamValue('NormalsData', [], @(x)isreal(x) && size(x,1) == 3);

inp.addParamValue('ReturnAll', false, @(x)islogical(x));

inp.addParamValue('Triangulation', [], @(x)isreal(x) && size(x,2) == 3);

inp.addParamValue('Verbose', false, @(x)islogical(x));

inp.addParamValue('Weight', @(x)ones(1,length(x)),
@(x)isa(x,'function_handle'));

inp.addParamValue('WorstRejection', 0, @(x)isscalar(x) && x > 0 && x < 1);

inp.parse(q,p,varargin{:});
arg = inp.Results;
clear('inp');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Actual implementation

% Allocate vector for RMS of errors in every iteration.
t = zeros(arg.iter+1,1);
```

```
% Start timer
tic;

Np = size(p,2);

% Transformed data point cloud
pt = p;

% Allocate vector for RMS of errors in every iteration.
ER = zeros(arg.iter+1,1);

% Initialize temporary transform vector and matrix.
T = zeros(3,1);
R = eye(3,3);

% Initialize total transform vector(s) and rotation matric(es).
TT = zeros(3,1, arg.iter+1);
TR = repmat(eye(3,3), [1,1, arg.iter+1]);

% If Minimize == 'plane', normals are needed
if (strcmp(arg.Minimize, 'plane') && isempty(arg.Normals))
    arg.Normals = lsqnormest(q,4);
end

% If Matching == 'Delaunay', a triangulation is needed
if strcmp(arg.Matching, 'Delaunay')
    DT = DelaunayTri(transpose(q));
end

% If Matching == 'kDtree', a kD tree should be built (req. Stat. TB >= 7.3)
if strcmp(arg.Matching, 'kDtree')
    kdOBJ = KDTreeSearcher(transpose(q));
end

% If edge vertices should be rejected, find edge vertices
if arg.EdgeRejection
    if isempty(arg.Boundary)
        bdr = find_bound(q, arg.Triangulation);
    else
        bdr = arg.Boundary;
    end
end

if arg.Extrapolation
    % Initialize total transform vector (quaternion ; translation vec.)
    qq = [ones(1,arg.iter+1);zeros(6,arg.iter+1)];
    % Allocate vector for direction change and change angle.
    dq = zeros(7,arg.iter+1);
    theta = zeros(1,arg.iter+1);
end

t(1) = toc;

% Go into main iteration loop
```

```
% k=1;
% while (k<=arg.iter)
for k=1:arg.iter
    % Do matching
    switch arg.Matching
        case 'bruteForce'
            [match mindist] = match_bruteForce(q,pt);
        case 'Delaunay'
            [match mindist] = match_Delaunay(q,pt,DT);
        case 'kDtree'
            [match mindist] = match_kDtree(q,pt,kdOBJ);
    end

    % If matches to edge vertices should be rejected
    if arg.EdgeRejection
        p_idx = not(ismember(match, bdr));
        q_idx = match(p_idx);
        mindist = mindist(p_idx);
    else
        p_idx = true(1, Np);
        q_idx = match;
    end

    % If worst matches should be rejected
    if arg.WorstRejection
        edge = round((1-arg.WorstRejection)*sum(p_idx));
        pairs = find(p_idx);
        [~, idx] = sort(mindist);
        p_idx(pairs(idx(edge:end))) = false;
        q_idx = match(p_idx);
        mindist = mindist(p_idx);
    end

    if k == 1
        ER(k) = sqrt(sum(mindist.^2)/length(mindist));
    end

    switch arg.Minimize
        case 'point'
            % Determine weight vector
             weights = arg.Weight(match);
%             size(match)
%              size(weights)
%             size(p_idx)
%              size(weights(p_idx))
%              sum(weights)
            [R,T] = eq_point(q(:,q_idx),pt(:,p_idx), weights(p_idx));
        case 'plane'
            weights = arg.Weight(match);
            [R,T] =
eq_plane(q(:,q_idx),pt(:,p_idx),arg.Normals(:,q_idx),weights(p_idx));
        case 'lmaPoint'
            [R,T] = eq_lmaPoint(q(:,q_idx),pt(:,p_idx));
    end

    % Add to the total transformation
```

```
    TR(:,:,k+1) = R*TR(:,:,k);
    TT(:,:,k+1) = R*TT(:,:,k)+T;

    % Apply last transformation
    pt = TR(:,:,k+1) * p + repmat(TT(:,:,k+1), 1, Np);

    % Root mean of objective function
    ER(k+1) = rms_error(q(:,q_idx), pt(:,p_idx));

    % If Extrapolation, we might be able to move quicker
    if arg.Extrapolation
        qq(:,k+1) = [rmat2quat(TR(:,:,k+1));TT(:,:,k+1)];
        dq(:,k+1) = qq(:,k+1) - qq(:,k);
        theta(k+1) =
(180/pi)*acos(dot(dq(:,k),dq(:,k+1))/(norm(dq(:,k))*norm(dq(:,k+1))));
        if arg.Verbose
            disp(['Direction change ' num2str(theta(k+1)) ' degree in
iteration ' num2str(k)]);
        end
        if k>2 && theta(k+1) < 10 && theta(k) < 10
            d = [ER(k+1), ER(k), ER(k-1)];
            v = [0, -norm(dq(:,k+1)), -norm(dq(:,k))-norm(dq(:,k+1))];
            vmax = 25 * norm(dq(:,k+1));
            dv = extrapolate(v,d,vmax);
            if dv ~= 0
                q_mark = qq(:,k+1) + dv * dq(:,k+1)/norm(dq(:,k+1));
                q_mark(1:4) = q_mark(1:4)/norm(q_mark(1:4));
                qq(:,k+1) = q_mark;
                TR(:,:,k+1) = quat2rmat(qq(1:4,k+1));
                TT(:,:,k+1) = qq(5:7,k+1);
                % Reapply total transformation
                pt = TR(:,:,k+1) * p + repmat(TT(:,:,k+1), 1, Np);
                % Recalculate root mean of objective function
                % Note this is costly and only for fun!
                switch arg.Matching
                    case 'bruteForce'
                        [~, mindist] = match_bruteForce(q,pt);
                    case 'Delaunay'
                        [~, mindist] = match_Delaunay(q,pt,DT);
                    case 'kDtree'
                        [~, mindist] = match_kDtree(q,pt,kdOBJ);
                end
                ER(k+1) = sqrt(sum(mindist.^2)/length(mindist));
            end
        end
    end

    if (k>1)
        diff=abs(ER(k-1)-ER(k))
        if(diff<=0.001)
            diff
            break;
        end
    t(k+1) = toc;

    end
```

```
    %k=k+1;
end
k
if not(arg.ReturnAll)
    TR = TR(:,:,end);
    TT = TT(:,:,end);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [match mindist] = match_kDtree(~, p, kdOBJ)
    [match mindist] = knnsearch(kdOBJ,transpose(p));
    match = transpose(match);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [R,T] = eq_point(q,p,weights)

m = size(p,2);
n = size(q,2);

% normalize weights
weights = weights ./ sum(weights);

% find data centroid and deviations from centroid
q_bar = q * transpose(weights);
q_mark = q - repmat(q_bar, 1, n);
% Apply weights
q_mark = q_mark .* repmat(weights, 3, 1);

% find data centroid and deviations from centroid
p_bar = p * transpose(weights);
p_mark = p - repmat(p_bar, 1, m);
% Apply weights
%p_mark = p_mark .* repmat(weights, 3, 1);

N = p_mark*transpose(q_mark); % taking points of q in matched order

[U,~,V] = svd(N); % singular value decomposition

R = V*diag([1 1 det(U*V')])*transpose(U);

T = q_bar - R*p_bar;

function [R,T] = eq_lmaPoint(q,p)

Rx = @(a)[1     0       0;
          0     cos(a)  -sin(a);
          0     sin(a)  cos(a)];

Ry = @(b)[cos(b)    0   sin(b);
          0         1   0;
          -sin(b)   0   cos(b)];
```

```
Rz = @(g)[cos(g)    -sin(g) 0;
          sin(g)    cos(g)  0;
          0         0       1];

Rot = @(x)Rx(x(1))*Ry(x(2))*Rz(x(3));

myfun = @(x,xdata)Rot(x(1:3))*xdata;%+repmat(x(4:6),1,length(xdata));


options = optimset('Algorithm', 'levenberg-marquardt');
x = lsqcurvefit(myfun, zeros(6,1), p, q, [], [], options);


R = Rot(x(1:3));
T = x(4:6);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Extrapolation in quaternion space. Details are found in:
%
% Besl, P., & McKay, N. (1992). A method for registration of 3-D shapes.
% IEEE Transactions on pattern analysis and machine intelligence, 239?256.

function [dv] = extrapolate(v,d,vmax)

p1 = polyfit(v,d,1); % linear fit
p2 = polyfit(v,d,2); % parabolic fit
v1 = -p1(2)/p1(1); % linear zero crossing
v2 = -p2(2)/(2*p2(1)); % polynomial top point

if issorted([0 v2 v1 vmax]) || issorted([0 v2 vmax v1])
    disp('Parabolic update!');
    dv = v2;
elseif issorted([0 v1 v2 vmax]) || issorted([0 v1 vmax v2])...
        || (v2 < 0 && issorted([0 v1 vmax]))
    disp('Line based update!');
    dv = v1;
elseif v1 > vmax && v2 > vmax
    disp('Maximum update!');
    dv = vmax;
else
    disp('No extrapolation!');
    dv = 0;
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Determine the RMS error between two point equally sized point clouds with
% point correspondance.
% ER = rms_error(p1,p2) where p1 and p2 are 3xn matrices.

function ER = rms_error(p1,p2)
dsq = sum(power(p1 - p2, 2),1);
```

```
ER = sqrt(mean(dsq));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Converts (orthogonal) rotation matrices R to (unit) quaternion
% representations
%
% Input: A 3x3xn matrix of rotation matrices
% Output: A 4xn matrix of n corresponding quaternions
%
% http://en.wikipedia.org/wiki/Rotation_matrix#Quaternion

function quaternion = rmat2quat(R)

Qxx = R(1,1,:);
Qxy = R(1,2,:);
Qxz = R(1,3,:);
Qyx = R(2,1,:);
Qyy = R(2,2,:);
Qyz = R(2,3,:);
Qzx = R(3,1,:);
Qzy = R(3,2,:);
Qzz = R(3,3,:);

w = 0.5 * sqrt(1+Qxx+Qyy+Qzz);
x = 0.5 * sign(Qzy-Qyz) .* sqrt(1+Qxx-Qyy-Qzz);
y = 0.5 * sign(Qxz-Qzx) .* sqrt(1-Qxx+Qyy-Qzz);
z = 0.5 * sign(Qyx-Qxy) .* sqrt(1-Qxx-Qyy+Qzz);

quaternion = reshape([w;x;y;z],4,[]);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Converts (unit) quaternion representations to (orthogonal) rotation
matrices R
%
% Input: A 4xn matrix of n quaternions
% Output: A 3x3xn matrix of corresponding rotation matrices
%
%
http://en.wikipedia.org/wiki/Quaternions_and_spatial_rotation#From_a_quaterni
on_to_an_orthogonal_matrix

function R = quat2rmat(quaternion)
q0(1,1,:) = quaternion(1,:);
qx(1,1,:) = quaternion(2,:);
qy(1,1,:) = quaternion(3,:);
qz(1,1,:) = quaternion(4,:);

R = [q0.^2+qx.^2-qy.^2-qz.^2 2*qx.*qy-2*q0.*qz 2*qx.*qz+2*q0.*qy;
     2*qx.*qy+2*q0.*qz q0.^2-qx.^2+qy.^2-qz.^2 2*qy.*qz-2*q0.*qx;
     2*qx.*qz-2*q0.*qy 2*qy.*qz+2*q0.*qx q0.^2-qx.^2-qy.^2+qz.^2];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% Least squares normal estimation from point clouds using PCA
%
% H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle.
% Surface reconstruction from unorganized points.
% In Proceedings of ACM Siggraph, pages 71:78, 1992.
%
% p should be a matrix containing the horizontally concatenated column
% vectors with points. k is a scalar indicating how many neighbors the
% normal estimation is based upon.
%
% Note that for large point sets, the function performs significantly
% faster if Statistics Toolbox >= v. 7.3 is installed.
%
% Jakob Wilm 2010

function n = lsqnormest(p, k)
m = size(p,2);
n = zeros(3,m);

v = ver('stats');
if str2double(v.Version) >= 7.5
    neighbors = transpose(knnsearch(transpose(p), transpose(p), 'k', k+1));
else
    neighbors = k_nearest_neighbors(p, p, k+1);
end

for i = 1:m
    x = p(:,neighbors(2:end, i));
    p_bar = 1/k * sum(x,2);

    P = (x - repmat(p_bar,1,k)) * transpose(x - repmat(p_bar,1,k)); %spd
matrix P
    %P = 2*cov(x);

    [V,D] = eig(P);

    [~, idx] = min(diag(D)); % choses the smallest eigenvalue

    n(:,i) = V(:,idx);   % returns the corresponding eigenvector
end

function bound = find_bound(pts, poly)

%Correcting polygon indices and converting datatype
poly = double(poly);
pts = double(pts);

%Calculating freeboundary points:
TR = TriRep(poly, pts(1,:)', pts(2,:)', pts(3,:)');
FF = freeBoundary(TR);

%Output
bound = FF(:,1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

8. References

[1]. Cook, Margaret J. *The anatomy of laboratory mouse.* 1965.

[2]. Anatoliy Granov, Leonid Tiutin, Thomas Schwarz. *Positron Emission Tomography.* Springer, 2013.

[3]. Artem Khmelinskii, Harald C. Groen, Martin Baiker, Marion de Jong, Boudewijn P. F. Lelieveldt. "Segmentation and Visual Analysis of Whole-Body Mouse Skeleton microSPECT." *PLOS one* 7, no. 11 (2012).

[4]. Artem Khmelinskii, Martin Baiker, Eric L. Kaijzel, Josette Chen, Johan H. C. Reiber, Boudewijn P. F. Lelieveldt. "Articulated Whole-Body Atlases for Small Animal Image Analysis: Construction and Applications." *Molecular Imaging and Biology*, 2010: 898-910.

[5]. Arthur W. Toga, Emily M. Santor!, Ron Hazani And Karen Ambach. "A 3D Digital Map of Rat Brain." *Elsevier Science Ltd* 38, no. 1 (1995): 77-85.

[6]. Belma Dogdas, David Stout, Arion F Chatziioannou, and Richard M Leahy. "Digimouse: a 3D whole body mouse atlas from CT and cryosection data." *Physics in Medicine and Biology* 52, no. 3 (2007): 577–587.

[7]. Bioscan. *NanoSPECT/CT In Vivo Preclinical Imager.* Bioscan. 2011. http://www.bioscan.com/molecular-imaging/nanospect-ct (accessed 10 13, 2013).

[8]. Bioscience, Sofie. *GENISYS4: A new generation of PET.* Sofie Bioscience. http://sofiebio.com/genisys (accessed 10 13, 2013).

[9]. Friedman, Jerome, Jon Bentley, and Raphael Finkel. "An Algorithm for Finding Best Matches in Logarithmic Expected Time." *ACM Transactions on Mathematical Software (TOMS)* 3, no. 3 (1977).

[10]. Hajnal, Joseph V., and Derek L.G. Hill. *Medical Image Registration.* Taylor and Francis, 2001.

[11]. Jadvar, Hossein, Bhushan Desai, and Peter S. Conti. "Sodium 18F-Fluoride PET/CT of Bone, Joint, and Other Disorders." *Elsevier* 45, no. 1 (2015): 58-65.

[12]. Kalyankar, Pravin P, and S S Apte. "3D Volume Rendering Algorithm." *International Journal of Engineering and Advanced Technology (IJEAT)* 2, no. 5 (2013): 268-270.

[13]. Kjer, Hans M artin, and Jakob Wilm. "Evaluation of surface registration algorithm for PET motion correction." Denmark, 2010.

[14]. Linton, Otha W. "Medical Applications of X-Rays." *Beamline*, Summer 1995.

[15].M. Zikos, E. Kaldoudi, S. C. Orphanoudakis. "DIPE: A Distributed Environment for Medical Image Processing." *Proceedings of Medical Informatics Europe*, 1997: 465 - 469.

[16].Marc Dhenain, Seth W. Ruffins, and Russell E. Jacobs. "Three-Dimensional Digital Mouse Atlas Using High-Resolution MRI." *Elsevier* 232, no. 2 (2001): 458–470.

[17].Martin Baikera, Julien Milles, Jouke Dijkstra, Tobias D. Henning, Axel W. Weber, Ivo Que, Eric L. Kaijzel, Clemens W.G.M. Löwik, Johan H.C. Reiber, Boudewijn P.F. Lelieveldt. "Atlas-based whole-body segmentation of mice from low-contrast Micro-CT data." *Elsevier*, no. 14 (2010): 723–737.

[18].Peter A. Santi, Ian Rapson, Arne Voie. "Development of the mouse cochlea database (MCD)." *Elsevier*, 2008: 11-17.

[19].R.M. Brune, 1, J.B.L. Bard, C. Dubreuil, E. Guest, W. Hill, M. Kaufman, M. Stark, D. Davidson, R.A. Baldock. "A Three-Dimensional Model of the Mouse at Embryonic Day 9." *Elsevier Science* 216, no. 2 (1999): 457–468.

[20].Saravanan Namasivayam, Mannudeep K. Kalra, William E. Torres, William C. Small. "Adverse reactions to intravenous iodinated contrast media:a primer for radiologists." *Am Soc Emergency Radiol* 12 (2006): 210–215.

[21].Segars WP, Tsui BM, Frey EC, Johnson GA, Berr SS. "Development of a 4-D digital mouse phantom for molecular imaging research." *Molecular Imaging Biology*, 2004: 149-159.

[22].Sharma, Neeraj , and Lalit M Aggarwal. "Automated medical image segmentation techniques." *Journal of Medical Physics* 35, no. 1 (2010): 3-14.

[23].Society, American Cancer. "Cancer Facts and Figures 2015." 2015.

[24].Talairach, Jean, Mark Rayport, and Pierre Tournoux. "Co-planar stereotaxic atlas of the human brain: 3-dimensional proportional system: an approach to cerebral imaging." VIII, 122. Germany: Thieme [u.a.], 1988.

[25].*The Visible Human Project®.* U.S. National Library of Medicine. August 27, 2013. http://www.nlm.nih.gov/research/visible/visible_human.html (accessed October 19, 2013).

[26].Thomas R. Langerak, Floris F. Berendsen, Uulke A. Van der Heide, Alexis N. T. J. Kotte, Josien P. W. Pluim,. "Multiatlas-based segmentation with preregistration atlas selection." *American Association of Physicists in Medicine* 40, no. 9 (2013).

[27].VanOss, Jeffrey Lee. "Automatic Atlas Based Analysis of Radiotracer Uptake in Bones from Fused Nuclear Imaging/CT Data Sets of Mice." *ScholarWorks GVSU*, 2012.

[28].W. Paul Segars, David S. Lalush, and Benjamin M. W. Tsui. "Modeling Respiratory Mechanics in the MCAT and Spline-B ased MCAT Phantoms ." *IEEE TRANSACTIONS ON NUCLEAR SCIENCE* 48, no. 1 (2001): 89-97.

[29].Xueling Bai, Li Yu, Qian Liu, Jie Zhang, Anan Li, Dao Han, Qingming Luo and Hui Gong. "A high-resolution anatomical rat atlas." *Journal compilation: Anatomical Society of Great Britain and Ireland* 209 (2006): 707–708.