TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

IEE70LT

Shuaibu Ali Gombe 144954IVEM

3D VISUALIZATION AND SEGMENTATION OF LUNGS USING ITK/VTK/QT FRAMEWORK

Master's Thesis

Supervisor: Olev Märtens

PhD
Lead Researcher

Tallinn 2017

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Shuaibu Ali Gombe 144954IVEM

KOPSUDE 3D VISUALISEERIMINE JA SEGMENTEERIMINE ITK/VTK/QT RAAMISTIKUS

Magistritöö

Juhendaja: Olev Märtens

PhD
Lead Researcher

Tallinn 2017

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Shuaibu Ali Gombe

15.05.2017

Abstract

Due to the increasing prevalence of cancer in our societies, several researches are being carried out to provide enhanced computer assisted diagnostics to be used by the radiologist. The Insight Toolkit is an open source cross-platform application toolkit widely used by researchers in the field of medical image processing. Despite ITK's widespread usage by experienced researches, early stage and inexperienced researchers find it quite challenging to get up and running with all the necessary tools for medical image processing. As such, in this thesis a bundled virtual environment (Ubuntu Linux virtual machine image) containing the newest versions of ITK integrated together with Visualization Toolkit (VTK) and Qt framework was prepared. Moreover, detailed literature survey relating to 3D lung segmentation was also carried out which resulted in developing a semi-automated region growing-based lung segmentation method implemented and tested using the CT scan datasets from LIDC-IDRI database. The first chapter begins with introduction and motivation, followed by thesis task specification. Second chapter dives into the current literature and state of the art techniques for lung segmentation. Chapter three gives an overview of medical imaging and modalities while concluding with a brief discussion regarding the Digital Imaging and Communication in Medicine (DICOM) standard. Chapter four gives details regarding the bundled environment and setup of ITK, VTK and Qt. Chapter 5 contains the implementation of 3D lung segmentation and visualization. The last chapter includes conclusion, summary and future work.

This thesis is written in English and is 65 pages long, including 6 chapters, 37 figures and 2 tables.

Annotatsioon

Suureneva vähi levimuse tõttu ühiskonnas viiakse läbi uuringuid, et pakkuda radioloogidele diagnostikakssuuremat arvuti abi. Insight Toolkit (ITK) on avatud lähtekoodiga platvorm ja rakenduste tööriistakomplekt ning kasutatakse laialdaselt meditsiini pilditöötluses uurijate poolt. Vaatamata ITK laiale levikule kogenud uurijate seas, varajases staadiumis ja kogenematute teadlaste arvates on üsna keeruline seada ja töötada läbi kõik vajalikud vahendid meditsiiniliseks pilditöötluseks. Käesoleva töös on valmis ja kokku kompileeritud virtuaalne keskkond (Ubuntu Linux virtuaalse masina failide kujutis), mis sisaldab uusimaid versioone ITK-st, integreerituna koos visualiseerimine Toolkit'i (VTK) ja Qt raamistikuga. Enamgi veel, on esitatud üksikasjalik kirjanduse ülevaade 3D kopsu segmenteerimine meetoditest ning selle põhjal on arendatud poolautomaatne pildi-piirkonnas kasvatamisel põhinev kopsu segmenteerimise meetod, mida rakendati ja kontrolliti, kasutades kompuutertomograafia andmekogumit LIDC-IDRI andmebaasist. Esimene peatükk algab sissejuhatuse ja motivatsiooniosaga, millele järgneb lõputöö ülesande spetsifikatsioon. Teine peatükk sukeldub kirjanduse ja tehnikataseme üksikasjalikku ülevaatesse, ka kopsu kujutise segmenteerimise osas. Kolmas peatükk annab ülevaate meditsiinirakendustest ja modaalsustest ning lõpeb DICOM standardi ülevaatega. Neljas peatükk kirjeldab üksikasjades tarkvara keskkonna kompileerimist ja seadistust ITK, VTK ja Qt osades. Viies peatükk kirjeldab 3D kopsu segmenteerimist ja visualiseerimist. Viimane peatükk sisaldab järeldusi, kokkuvõtet ja tulevast tööd.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 65 leheküljel, 6 peatükki, 37 joonist, 2 tabelit.

List of abbreviations and terms

3D	Three-dimensional
ACR	American College of Radiology
ANN	Artificial Neural Network
CAD	Computer Aided Diagnosis
CT	Computed Tomography
DICOM	Digital Imaging and COmmunications in Medicine
EHR	Electronic Health Record
EMR	Electronic Medical Record
FBP	Filtered Back Projection
FDA	Food and Drug Administration
FDG	Fluorodeoxyglucose
FFNN	Feed Forward Neural Networks
FFT	Fast Fourier Transform
fMRI	Functional Magnetic Resonance Imaging
FN	False Negative
FNIH	Foundation for the National Institutes of Health
FP	False Positives
GCC	GNU Compiler Collection
GGN	Ground Glass Density Nodule
GUI	Graphical User Interface
HU	Hounsfield Unit
IDRI	Image Database Resource Initiative
IOD	Information Object Definition - DICOM
ITK	Inisight Toolkit
JPEG	Joint Photographic Experts Group
k-NN	K-Nearest Neighbors
LDA	Linear Discriminant Analysis
LIDC	The Lung Image Database Consortium
MHz	Megahertz

MRA	Magnetic Resonance Angiography
MRI	Magnetic Resonance Imaging
MRS	Magnetic Resonance Spectroscopy
NEMA	National Electrical Manufacturers Association
NIC	National Cancer Institute
NMI	Nuclear Medicine Imaging
OS	Operating System
PACS	Picture Archiving and Communication System
PET	Positron Emission Tomography
PSN	Part-solid Lung Nodule
PGGN	Pure Ground Glass Density Nodule
PyQt	Python Qt
RF	Radiofrequency
ROI	Region of Interest
SPECT	Single-photon Emission Computed Tomography
SVM	Support Vector Machines
TP	True Positive
VR	Value Representation - DICOM
VTK	Visualization Toolkit
Qt	Cute Cross Platform Application Framework

Table of Contents

List of figures

List of tables

1 Introduction

Lung cancer is one of the leading causes of cancer related death worldwide and more so in developed countries [1]. As such giving rise to the continuous need for enhanced and more accurate automated lung segmentation and detection methods which is an important part of computer aided diagnosis (CAD) to be used by the radiologist.

A lung or pulmonary nodule (which sometimes also referred to as “spot on the lung”, “shadow”) is usually some specific round-like area that is more solid than the usual surrounding lung tissues. They are commonly found in CT scans of the chest although most lung nodules are benign (non-cancerous). A lung nodule varies according to its location, size, shape and appearance. More specifically, a nodule refers to a spot on the lungs with diameter of around 3 centimetres and anything above that could be referred to as a mass. With benign pulmonary nodules, there is slight change in their size over time which contrasts with cancerous nodules’ ability to grow quickly (doubling in about 4 months or sometimes even much less). A cancerous pulmonary nodule is some sort of lesion that engulfs (overtime) the structures of the lung and thus leading to patient experiencing pains and other difficulties such as shortness of breath. Early detection of cancerous pulmonary lung nodules tend to be very challenging as normally there seems to be no clear symptoms from the beginning stages. Cancerous nodules can also be distinguished from benign nodules through its calcification. Calcification refers to the nodules’ development based on its surface and shape. Cancerous nodules tend to have irregular shape and rough surface while benign nodules are more likely to be regular in shape with even colour.

In this MSc research project, the author focussed on providing readymade and easily accessible/extensible application framework to be used in lung segmentation (the process of nodule detection). This constitutes setting up some sort of virtual open source-based medical image processing research environment bundled into a Linux virtual machine image which is easily distributable. Moreover, a segmentation algorithm was implemented and tested using the provided framework/environment. With the inherent

difficulty for early stage researchers to get up to speed in integrating the necessary tools for efficient medical image research, a lot of time is spent in learning the tools and appropriate setup/environment. In particular, setting up/integrating the full Insight Toolkit (ITK), Visualization Toolkit (VTK) and Qt complexity leave early stage researchers with no option than using the SimpleITK with limited functionality as the full setup of ITK/VTK/Qt bundle requires enormous amount of effort to set the environment up (configuration, custom build from source) leading to endless hours and days of trial and error which could have been spent doing main research activity.

In order to test the bundled environment through implementing specific lung segmentation from the literature, the author used CT scans freely available from LIDC-IDRI database. The LIDC-IDRI database consists of marked-up annotated lesions of thoracic CT scans including diagnostic and lung cancer screening. It was an initial effort by NIC which was further advanced by FNIH and accompanied by FDA. The dataset contains about 1018 cases thanks to the collaboration between academic centres and medical imaging companies [2].

1.1 Thesis task specification

The main tasks include:

- Perform literature survey relating to lung segmentation methods particularly 3D
- Provide readily accessible environment enabling 3D medical image analysis of the lungs using full featured ITK/VTK/Qt bundled into an easily distributable Linux virtual machine image. This involves custom configuration and building from source including wrappers for interfacing C++ base with higher level languages such as Python
- Implement and test effective lung segmentation from literature with the bundled environment ready from the previous step using sample thoracic CT scans from LIDC-IDRI database. This involves processing and visualization of segmented lungs in 3D

2 Literature Review

Several researches and detailed surveys have been carried out in the literature related to lung nodule segmentation and CAD for thorasic CT scan images [3], [4], [5].

The survey in [3] reviewed thoroughly the literature specifically relating to automated 3D detection of pulmonary nodules in CT images. As the authors pointed out, the accuracy in being able to determine precisely the nodule size is critical because it is directly related to the nodule's malignancy. The appearance of lung nodules differs based on its type, malignancy (or benign), size, internal structure and its location. As such pulmonary nodule detection and segmentation appears to be quite challenging task which most of the time involves the use of various methodologies (diverse levels of complexity) with each particular methodology able to handle only a particular aspect of the entire problem as whole.

In order for any computer aided diagnostic system to be of useful to the practicing radiologist in lung nodule segmentation and detection, it is expected that such a system meets certain requirements as specified in [6]. One of the key requirements is to be able to maintain high sensitivity and thereby improving the overall efficiency of the evaluation. High sensitivity means higher true positive rate (TP). The sensitivity is defined as the ratio between the true positive rate to the sum of true positive rate and false negative rate(FN). Basically, TP implies the ability of the CAD system to detect positive outcome from a sample CT scan containing malignant nodule while TF implies the inability of the system to detect the presence of malignant nodules in a CT scan which does contain tumours. Another requirement for such systems to be useful to the practicing radiologist is the having low false positive rate. False positive refers to the situation whereby the CAD system signals to presence of malignant nodule when the particular sample contain non or benign nodules and a such leading to errors in diagnosis and detection by radiologist. Other requirements include: fast processing time; high-level of automation avoiding the need for manual intervention; low cost, maintenance and support requirement including training; ability to shapes, sizes and types including juxta-pleural and juxta-vascular nodules.

Generally speaking, CAD systems for lung nodules detection composes of five major stages: data acquisition phase, pre-processing, lung segmentation stage, followed lung nodule detection and subsequently the reduction of false positives rate. The main stages are described in Table 1.

Table 1. Processing stages generally used in literature [3]

Authors	Acquisition of data	Pre-processing	Lung Segmentation	Nodule detection	FP reduction
Choi and Choi [7]	*Yes*	*No*	*Yes*	*Yes*	*Yes*
Santos et al. [8]	*Yes*	*No*	*Yes*	*Yes*	*Yes*
Badura and Pietka [9]	*Yes*	*No*	*Yes*	*Yes*	*Yes*
El-Baz et al. [10]	*Yes*	*No*	*Yes*	*Yes*	*Yes*
Wang et al. [11]	*Yes*	*No*	*Yes*	*Yes*	*Yes*
Cascio et al. [12]	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*
Chen et al. [13]	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*
Soltaninejad et al. [14]	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*
Suiyuan and Junfeng [15]	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*
Riccardi et al. [16]	*Yes*	*No*	*Yes*	*Yes*	*Yes*
Liu et al. [17]	*Yes*	*No*	*Yes*	*Yes*	*Yes*
Taghavi et al. [18]	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*
Matsumoto et al. [19]	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*
Ozekes and Osman [20]	*Yes*	*No*	*No*	*Yes*	*Yes*
Ozekes et al. [21]	*Yes*	*No*	*Yes*	*Yes*	*Yes*
Yang et al. [22]	*Yes*	*No*	*No*	*Yes*	*Yes*
Ge et al. [23]	*Yes*	*No*	*Yes*	*Yes*	*Yes*
Matsumoto et al. [24]	*Yes*	*No*	*Yes*	*Yes*	*Yes*
Hara et al. [25]	*Yes*	*No*	*Yes*	*Yes*	*Yes*
Mekada et al. [26]	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*
Dehmeshki et al. [27]	*Yes*	*No*	*No*	*Yes*	*No*
Armato III et al. [28]	*Yes*	*No*	*Yes*	*Yes*	*Yes*

The data acquisition step mainly involves the process of retrieving the input images to be processed by the system. In the ideal scenario, such data acquisition could be done by integrating picture archiving and communication system (PACS), electronic health record (EHR) or electronic medical record (EMR) and a computer aided diagnostic system. With this, images can be processed before analysis by the radiologist. On the other hand, for the purpose of research development and design, there are some publicly available database containing cases with lung nodules such as the Lung Image Database Consortium (LIDC) [29].

In order to achieve best possible segmentation results, it is necessary to pre-process the input image (CT scan) prior to the lung segmentation. There are several types of processing techniques that can be applied such as: median filtering [13], [14]; linear interpolation [12], [15], [24]; morphological operation [14]; Gaussian filtering [18], [27]; weighted-sum filtering [26]; contrast limited adaptive histogram equalization [30]; enhancement filter [31], [32]; wiener filter [33]; auto-enhancement [33]; Fast Fourier Transform (FFT) [34]; wavelet transform [34]; anti-geometric diffusion [34]; erosion filter [35]; noise correction [36] and smoothing filters [36].

After the pre-processing stage comes the next crucial step which is lung segmentation. This step contributes significantly to the detection of malignant nodules. It is complicated by the fact that the lung region does not seem to be heterogenous. Moreover, it is also affected by the existence of structures such as veins, arteries, bronchi/bronchioles all with similar densities [3]. There several kinds of techniques used in lung segmentation can be broadly classified as: thresholding based approach, shape based approach, deformable models approach and edge based [10].

After carving out the region of interest which corresponds to segmenting the lungs from the entire thoracic CT scan comes the nodule detection step. The nodule detection step constitutes the systematic process of identifying the presence of candidate lung nodules from the carved-out region of interest (ROI). The process of detecting nodules is quite challenging and the main difficulty lies in the separating real nodules from vessels and other structures within the lungs. Some of the techniques used in the literature for nodule detection are: genetic algorithm template matching [10], [37]; Hessian matrix-based [7]; [8], [13], [38]: 3D mass-spring models [12]; thresholding [15]; sieve filter [39]; variable n-quoit filter [40]; adaptive weighted k-means clustering [23]; connected component [26]; multiple gray-level thresholding [11], [41]; and 3D labelling method [42].

Following nodule detection stage is the false positive reduction step. Presence of false positive in the output of nodule detection affects the accuracy in interpreting medical diagnostics and examination. The challenge here lies in reducing false positive rate and at the same time maintaining high sensitivity. It is necessary to further analyse the candidate nodules' features after detection to determine the existence of false positive or not. The features of interest mainly include texture, morphology and pixel intensity values around the candidate nodules. Machine learning algorithms and techniques are used in

analysing the features and possibly detecting non-nodules from true nodules. This poses another challenge in terms of number of samples to be used depending on nodules' location, type and shape. Some of the machine learning techniques used in literature includes: support vector machines (SVM) [7], [8], [11], artificial neural network (ANN) [12], Bayesian supervised [10], k-nearest neighbours (k-NN) [14], feed forward neural networks (FFNN) [20], and linear discriminant analysis (LDA) [41].

The authors in [43] used iterative thresholding method as segmentation algorithm together with freeman chain code algorithm in order to repair the boundary of lung regions that are separated. For the node detection step, they used region growing method to extract the candidate lung nodules from the segmented region of interest. Furthermore, in [44], a new variational level set approach for nodule detection and segmentation was proposed. The authors also proposed the use of implicit spaces (signed distant function) to model the shape of long nodule. The model is fused together with the statistical information from the image intensity using variational segmentation frame. The model of the nodule is then mapped to image domain by some form of transformations (scaling, rotation, translation). Gradient descent optimization was employed to handle the shape model alignment and subsequently marking the nodules out. The technique is independent of the location and type of nodule which makes it more efficient in different scenarios.

Similarly, in [45] the authors presented an automated method for lung segmentation from CT scan images. Thresholding method was used at first in separating the lungs region from the CT scan. Series of morphological operations were applied to fully separate the left and right lungs. Prior to lung segmentation, Gaussian filter was used to smooth the input image in order to reduce noise. The threshold value was found by first defining an interval between -950 to 0 Hounsfield units which covers all the range of intensities of the lungs. It was then followed by a search for local minimum from the image histogram and creating a binary image from that. Further processing was carried out to remove the trachea and separate the two lungs.

Another method for segmenting lung parenchyma was described in [46]. The first step involved converting the CT image into binary image after thresholding. Adaptive thresholding [48] was used to obtain optimal threshold using the following iterative process and formula:

$$T^{i+1} = \frac{1}{2}(\mu_b + \mu_n)$$

Where μ_b is the average gray-level of the body voxels, μ_n be the average gray-level of the non-body voxels at the output of segmentation step having the threshold T^i. The new threshold T^{i+1} is then calculated in the next i+1. Thus this iterative process is repeated until no more changes in the threshold. Meaning, the threshold $T^{i+1} \approx T^i$. This is followed by applying 3D connected component labelling to eliminate the background from the binary image. Further processing was necessary to remove the trachea and subsequently smoothing followed by applying the binary mask to the original CT image to extract the lungs region.

The authors in [47] presented an algorithm for extracting region of interest (which is the long tissue) from the CT image in the context of content-based retrieval. As the authors pointed out that for content-based retrieval, image quality is less critical as compared to missing any other important part of the lungs. The algorithm is able to handle both JPEG and DICOM file types. The algorithm consists of five main steps as follows:

1) Adaptive thresholding technique is applied to determine optimal threshold value [48]. This allows for the separation of high and low density tissues

2) The removal of surrounding air (background) using the technique in [48], [49]

3) Performing cleaning so as to remove airways and reduce noise

4) The use of a rolling-ball operator in rebuilding the lung borders which basically constitutes the use of some morphological operation followed by hole filling

5) The last step involves separating the left and right lungs

More recently, the authors in [50] proposed methods for segmenting lung tissue and detection of lung nodules in the initial stage. The lung tissue segmentation method is based on combination of binary masks, flood fill algorithm, together with some morphological closing operation. The lung nodule detection step consists of multi-level thresholding process followed by applying some various forms of feature extraction techniques.

3 Medical Imaging

When it comes to medical imaging of the human body, it basically requires energy in certain form to begin with. For the use in radiology, the techniques mostly used in medical imaging must be such that the energy is able to penetrate the human body and tissues. As the case with frequencies of light in the visible spectrum, it has some limited capacity in penetrating human body and tissues and therefore not of much use in radiology applications. Although light in the visible spectrum has its own uses in the case of skin photography, endoscopy, light microscopy and many other fields. For the purpose of medical imaging in radiology, the spectrum of electromagnetic radiation outside of the visible light range is more suitable. As such, x-rays are used in computed tomography (CT) imaging and mammography. Likewise, for magnetic resonance imaging (MRI), radiofrequency (RF) range of the spectrum is used while in the case of nuclear medicine gamma rays are often used. On the other hand, ultrasound imaging uses mechanical energy from high frequency sound waves.

Apart from the case of nuclear medicine, virtually other medical imaging techniques necessarily need to have the energy generated and transmitted from the source to penetrate the human body and tissues while interacting with them. For if the energy only passes through the tissues without any form of interaction such scattering or absorption, then on the receiver side the detected signal from the energy might not constitute much information with regards to internal looks and anatomy of the body tissue. With such little information, it would not be possible to reconstruct the internal anatomy of the issue. Radioactive substances have to be introduced into the human body in the case of nuclear medicine to aid the physiological interactions and hence more information produced on the images.

When it comes to medical imaging, both the conditions under which the image is acquired and its technical quality are of critical importance. As such it is necessary to be able to technically (and precise) assess and evaluate the image quality and conditions under which it is acquired. The quality of images acquired in medical imaging devices requires some form of compromise. For example, better images from x-ray could be obtained by simply increasing the radiation dose which could be very harmful to the patient.

3.1 Medical imaging modalities

There are diverse ways and acquisition techniques in which medical images of the body or tissues can be captured through the use of different kinds of energies be it electromagnetic radiation (x-rays, gamma rays etc), high frequency sound waves in the case of ultrasound, or perhaps nuclear medicine requiring ingestion of radioactive substances. These different methods and modes by which medical imaging can be done are generally referred to as imaging modalities. Different modalities are more suited for varying applications. The discussion below provides details regarding some of the common imaging modalities. The discussions in this are mostly according to [51] - [54].

3.1.1 Radiography

Radiography happens to be the first kind of medical imaging modality which was possible due to the discovery of x-rays by Wilhelm Roentgen in 1895. Roentgen was able to make early radiographic image of human body as shown in Figure 1 his wife's hand. Radiography is also referred to as roentgenography. Radiography gave rise to the field of radiology which in turn lead to having radiologist as physicians specializing solely in interpreting medical images. In order to take radiographic images, there needs to a source of x-ray on particular side of a patient and an x-ray detector usually situate on the other side. A pulse with short duration of around half of a second is emitted from the x-ray source tube causing some part of the x-ray to interact with human (patient's) body while some eventually passing through the body and reaching the x-ray detector on the other side where an image of the body is formed. Before the x-rays interact with the human body, it has some sort of homogenous distribution and that will be modified due to the interaction through absorption or scattering and thus experiencing some form of attenuation. Different constituents of the body such as bones tissue or air causes attenuation in a different way and thus resulting in a heterogenous distribution of the x-rays on the side of the detector. The detector used in x-ray could be photographic film or digitally based electronic system.

Imaging could be transmission or projection based. Transmission based imaging refers to situation where by the source of energy is outside the patient's body and after interaction which the body it is detected on the other side. Projection based imaging techniques constitutes a situation where by each point on the detector forming the image corresponds to information in straight trajectory along the patient's body. Radiography is both

transmission and projection based. Figure shows a simplified setup of planar radiography including anti-scatter grid.

Figure 1. Radiographic image of Roentgen's wife's hand, 1895 [51]

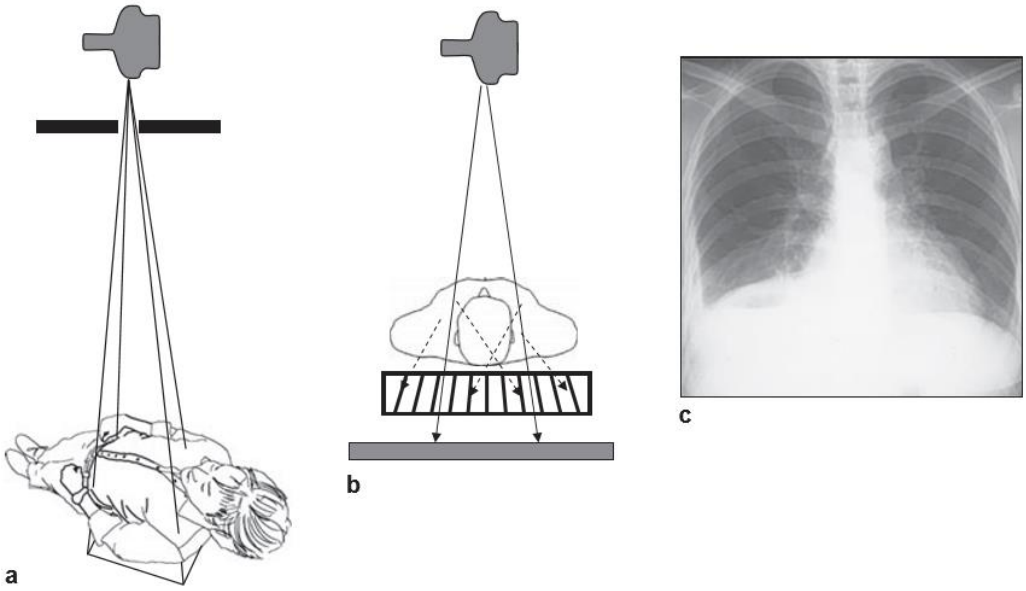

Figure 2. Simplified view of planar radiography (a), showing anti-scatter grid (b) and sample image of the chest (c) [52]

3.1.2 Fluoroscopy

Fluoroscopy constitutes the process of continuously acquiring some series of x-ray images over some time interval which essentially gives to real-time movie of the patient's body. It is both transmission and projection based imaging modality and can be viewed as real-time radiography. This kind of imaging systems make use of an x-ray detectors with the possibility to capture fast temporal sequence of images. Fluoroscopy is well suited for application like invasive procedures (therapeutics) and also in making movies of motion of the heart or other organs.

3.1.3 Mammography

Mammography refers to taking the radiographic images of the breast. As such, it is both transmission and projection based imaging modality. Very low x-ray energies are used in mammography and so requires some specially designed x-ray detectors suitable for the breast imaging. Screening mammography involves screening women with possible breast cancer while diagnostic mammography involves diagnosing women with presence of lump in the breast. More recently, some of the mammographic systems are able to produce breast images using tomosynthesis. Figure 3 shows an image of the breast produced through mammography and the low energy x-ray spectrum.

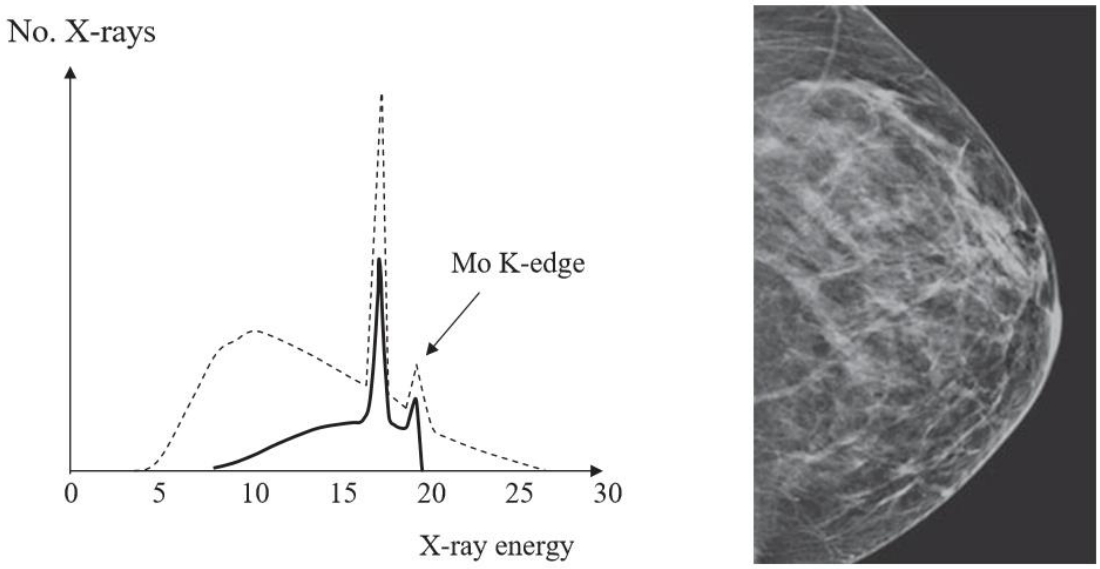

Figure 3. Spectrum of low energy x-ray used in mammography (left) and high resolution mammogram (right) [52]

3.1.4 Computed Tomography

The computed tomography (CT) happens to be first medical imaging modality to be used by the computer. It became available for clinical use in early 1970s. Computed tomography produces images of body parts by passing x-rays at several angles through rotation of the x-ray tube around the patient's body. A detector is placed in the opposite direction with respect to the source to receive the data. Numerous data are collected and passed to the computer for synthesis which results into tomographic images of patient's body. Tomography constitutes numerous slices of pictures. Unlike radiography, computed tomography has the advantage of being able to produce three-dimensional (3D) slices of patient's anatomy of interest. With the advent of CT, there was less need for conventional exploratory surgery anymore. CT scanners are able to acquire within few seconds tomographic images of up to 0.5mm in thickness. With 50cm patient's length corresponding to about 800 image slices. Images from CT are able to reveal many pathologies including cancer tumours. Figure 4 shows a helical CT scanner.

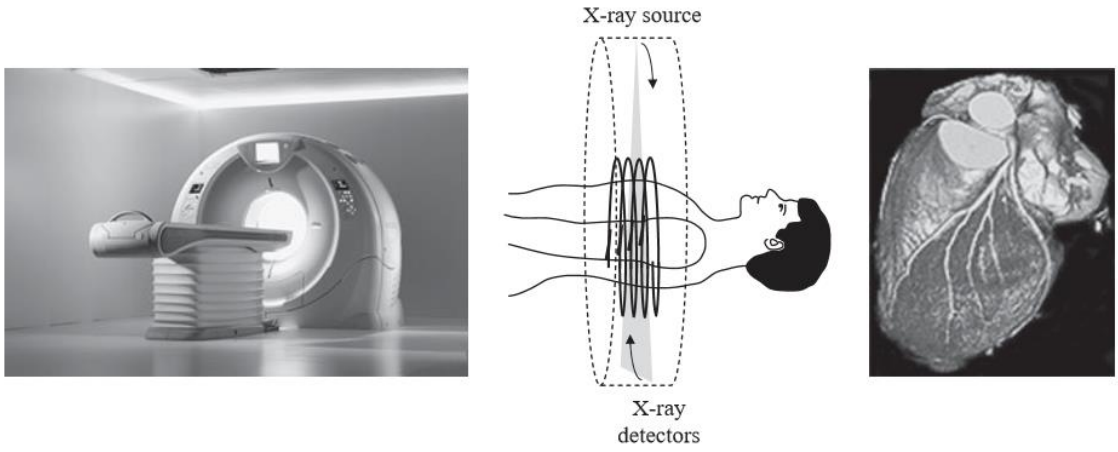

Figure 4. Helical CT scan, x-ray source/detector and 3D surface rendering of human heart [52]

In addition to axial images, it is also possible to produce sagittal and coronal images from CT scanners due to the fact that the volumetric data of CT is isotropic. CT scanner could operate in various modes such as: organ perfusion, dual-energy and gated cardiac CT. Figure 5 shows both coronal and sagittal views from an abdomen-pelvis CT scan.

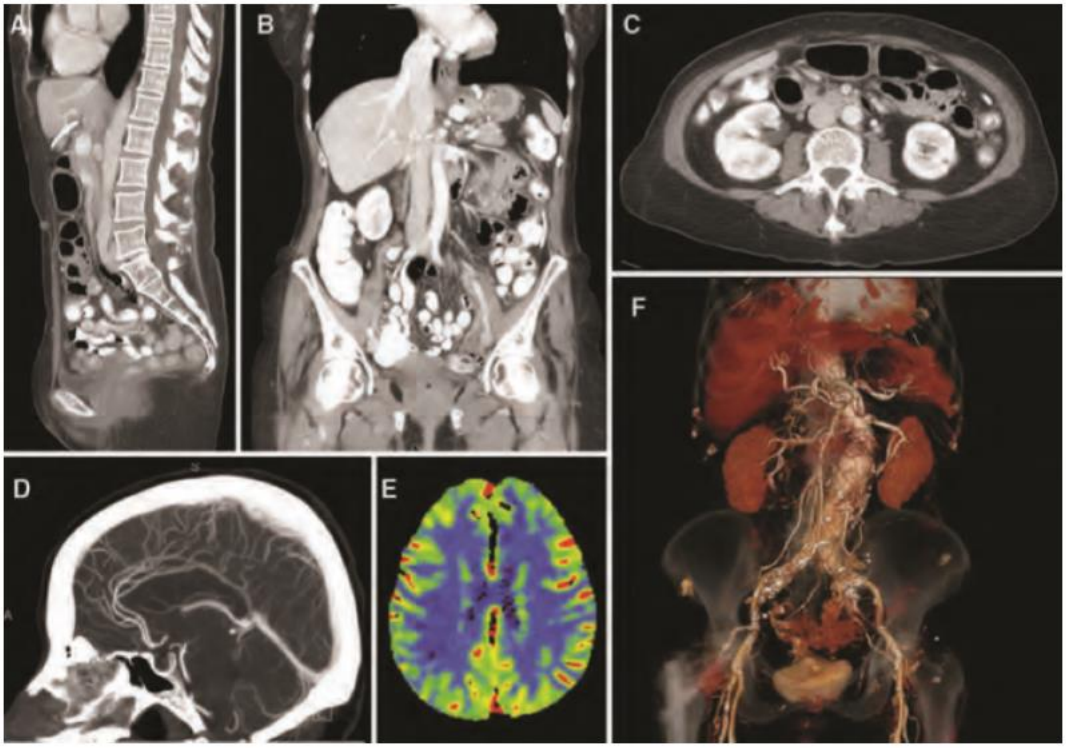

Figure 5. CT scan for human abdomen and head showing sagittal (A, D), coronal (B, E) and axial (C) views together with a 3D coloured surface (F) [51]

3.1.5 Magnetic Resonance Imaging

In the case of magnetic resonance imaging (MRI), very strong magnetic fields which are thousands of times that of the earth's are normally used. Majority of MRI scanners use some sort of nuclear magnetic properties of proton (hydrogen atom's nucleus which is abundant in human body tissues). When the proton is placed in a magnetic field, it wobbles on its axis and thus selectively absorbing some specific electromagnetic wave energy (radio) of around 64 megahertz (MHz). MRI requires the patient to be placed inside a magnetic field and a radio wave pulses are generated by the antenna coil located near the patient. The patient body interact with the generated waves (absorption) and afterwards the waves are re-emitted within some amount of time depending on the

magnetic properties of that specific tissue. Antennas surrounding the patient detects the re-emitted waves from the protons in the patient's body. Through slightly changing the magnetic field strength with position, the resonance frequency of the proton will also change with the position. In order to estimate the position of each signal coming from the patient's body, the frequency and phase information from the re-emitted waves are used. One of the widely used operating modes for MRI is spin echo imaging.

Just like in CT, MR also tomographic image slices of the patient whereby every point inside the image is determined by the magnetic properties of that particular tissue at the specified point. Medical images produced by MRI have better sensitivity to variations and higher contrast because various kinds of tissues in the human body (white/gray matter, cancer tumour, fat, etc) are having distinct magnetic properties. MRI are used predominantly in neurological imagine of the spine/head and musculoskeletal imaging such as knee injury as shown in Figure 6.

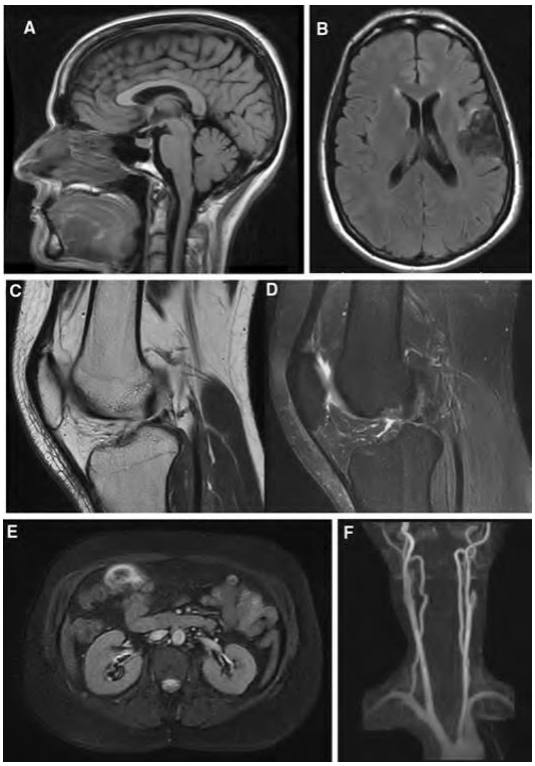

Figure 6. High contrast MRI images of the brain (A, B), knee (C, D), MRA (E) and abdomen (F) [51]

MRI technology basically serves as a strong competitor for the CT in several clinical applications. In contrast to CT scan which requires few seconds to complete, MRI generally tend to require much longer time (several minutes) to complete. This makes MRI not suitable for paediatrics or scenarios where motion tend to be difficult to be controlled. In such cases CT is normally used in place of MRI. Moreover, due to the strong magnetic field involved in MRI only certain kind of monitoring equipment are used during scanning of patients. On the other hand despite the drawbacks of MRI, nowadays faster acquisition time for images is possible due to the usage of enhanced coils. MRI is able to detect motion which makes it suitable taking images of arteries (blood flow) and brain (fMR). Magnetic resonance is also used in analysing tissue metabolism (MRS).

3.1.6 Ultrasound Imaging

Sound waves are a form of mechanical energy produced due to disturbances causing pressure to develop and propagate through air which can then be heard a point farther away from the wave source. This form of high frequency waves generated by sound are being used in forming images of human body by generating short sound pulses. The pulses are generated by an ultrasound transducer which is directly in contact with the body part or tissue of which image is being produced. The generated waves from the ultrasound transducer are reflected upon hitting the internal tissue structures of the body and thus resulting in echoed signal. The transducer is then able to receive the reflected sound wave signal which is an operating mode called pulse echo imaging. Beam of sound waves are being send in slices over the patient in straight lines using some form of transducer arrays or other advanced technique such as phased array transducer. The amplitudes of the resulting echoed waves (line by line) are then taken and used to calculate the brightness and subsequently converted into grayscale tomographic images slices of the body tissues. Ultrasound finds application in obstetrics for taking pictures of foetus because it is less harmful than ionising radiation. Ultrasound is not suitable for interfaces with high echoes (due to tissue-air filled interface) such as the lungs and the thorax. With specialized techniques, tomographic slices from ultrasound can also be converted into 3D volume representation.

3.1.7 Nuclear Medicine Imaging

Nuclear medicine imaging (NMI) basically involves giving some sort of radioactive chemical substance to patient through the process of inhaling, injection or even orally. Upon the distribution of the radioactive chemical substance inside the patient's body, then radiation detector will be making image projections using the gamma rays or x-rays as a result of the radioactive decay of the chemical substances' isotope. Nuclear medicine imaging is emission-based because of the radiation emitted from the chemical isotopes inside the patient's body. With nuclear medicine imagine, not only does it provide details regarding the patient's anatomical issue of interest, but it also provides details regarding the physiological situation inside of the patient. One of the chemical substances used in nuclear medicine is iodine which makes it possible to produce images of the thyroid due to its strong affinity for iodine or its equivalent. Nuclear medicine finds its usage in functional imaging (imaging physiological processes).

3.1.8 Single Photon Emission Computed Tomography

Single photon emission computed tomography (SPECT) can be viewed as the tomographic equivalent of nuclear medicine imaging (planar). Cameras are situated in several angles in order to capture the gamma or x-rays emitted from the patient's body and such data is them used in reconstruction tomographic image slices of the anatomical tissue of interest. As such, just like in conventional nuclear medicine imaging, it also possible to produce functional images using SPECT and more so with the tomographic advantages giving medical radiologist the ability to see clearly the functional situations of tissues and organs of interest inside the body. NMI and SPECT both use similar radioactive substances.

3.1.9 Positron Emission Tomography

In positron emission tomography (PET), isotopes of some radioactive substances emit positrons (positively charge electrons) and these chemical substances are combined with others to form suitable compounds such as 18F-fluorodeoxyglucose (18FDG) having the ability to localize inside the human body. Positrons are emitted which then interact with the electrons in the body tissues in the process called annihilation resulting into pure energy (radiation). The photons generated due to annihilation are 180 degrees apart which requires ring of detectors to be situated around the patient to capture the proton pair. From that tomographic images are constructed representing the internal tissues.

3.2 Image reconstruction techniques

In the case of CT, SPECT and PET imaging techniques, it is necessary to reconstruct a form of two-dimensional image from the acquired series of one-dimensional projections (say $\{p_1, p_2,.., p_n\}$). Many projections are required from all distinct detector angles in order to be able to effectively reconstruct the image representation as shown in Figure 7.

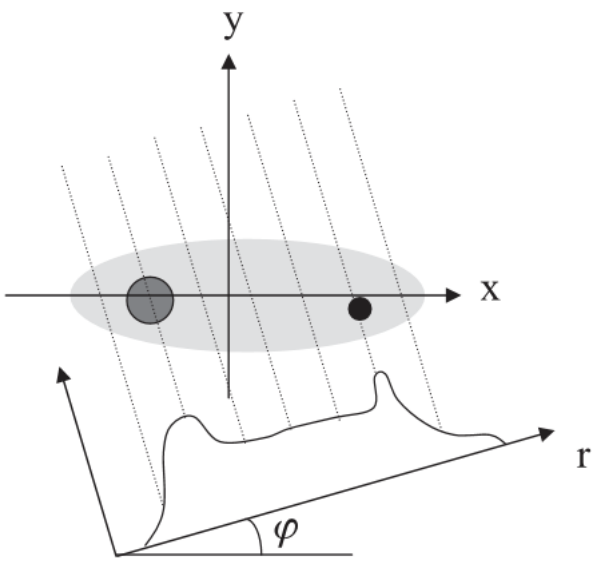

Figure 7. Projections of Image f(x,y), with projection p(r,φ) [52]

Reconstruction is dependent on the modality type. CT images values are represented in the Hounsfield units (HU) which is relates to the attenuation coefficient of the x-ray due to interaction with the body tissues as it passes though. The value of interest in the case of SPECT and PET is the distribution of chemical substance (radioactive) inside the patient's body.

Once all the necessary projections of the object of interest has been taken, the image will then be reconstructed using the method of backprojection or filtered back projection (FBP). Several other reconstruction techniques are used which are all inspired in away by the Radon transform and the Fourier slice theorem.

Figure 8 shows a very simple object backprojected with different number of projections taken. As it can be clearly seen, the higher the number of projections the better reconstruction of the original objects.

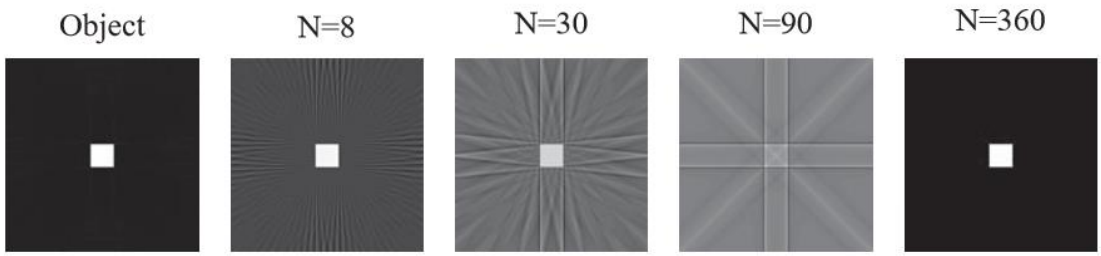

Figure 8. Backprojection reconstruction using different number of projections [52]

Values in CT Images are displayed in terms of the CT number (CT_o) expressed in Hounsfield unit (HU) which is related to attenuation coefficient as follows:

$$CT_o = 1000 \times \frac{\mu_o - \mu_{H_2o}}{\mu_{H_2o}}$$

Where μ_o and μ_{H_2o} represents the linear attenuation coefficient for the tissue and water respectively.

The range of values in HU of some common tissue types is shown in Table 2.

Table 2. Hounsfield values of some tissues [53]

Tissue Type	**Hounsfield Value Interval**
Air	-1000
Lung tissue	-900 to -170
Fat tissues	-220 to -30
Water	0
Pancreas	10 to 40
Liver	20 to 60
Heart	20 to 50
Kidney	30 to 50
Bones	45 to 3000

3.3 The DICOM standard

DICOM is an acronym for Digital Imaging and Communications in Medicine. DICOM can be viewed as the outcome of several years of effort in creating a universal and interoperable standard in all aspects of digital medical imaging initially lead by American College of Radiology (ACR) and National Electrical Manufacturers Association (NEMA) joint committee starting in 1983. It constitutes virtually all the important tools need for representing and processing accurately of medical imaging data. DICOM is more than just file format or image format but rather comprises the means of transferring medical data, how to store such data, display and other functional necessities of modern day medicine. Simply put, DICOM could be viewed as a set of all-encompassing standards.

Picture Archiving and Communication Systems (PACS) are related to DICOM in the sense that they are driven by the DICOM standards. Main components of PACS are shown in Figure 9.

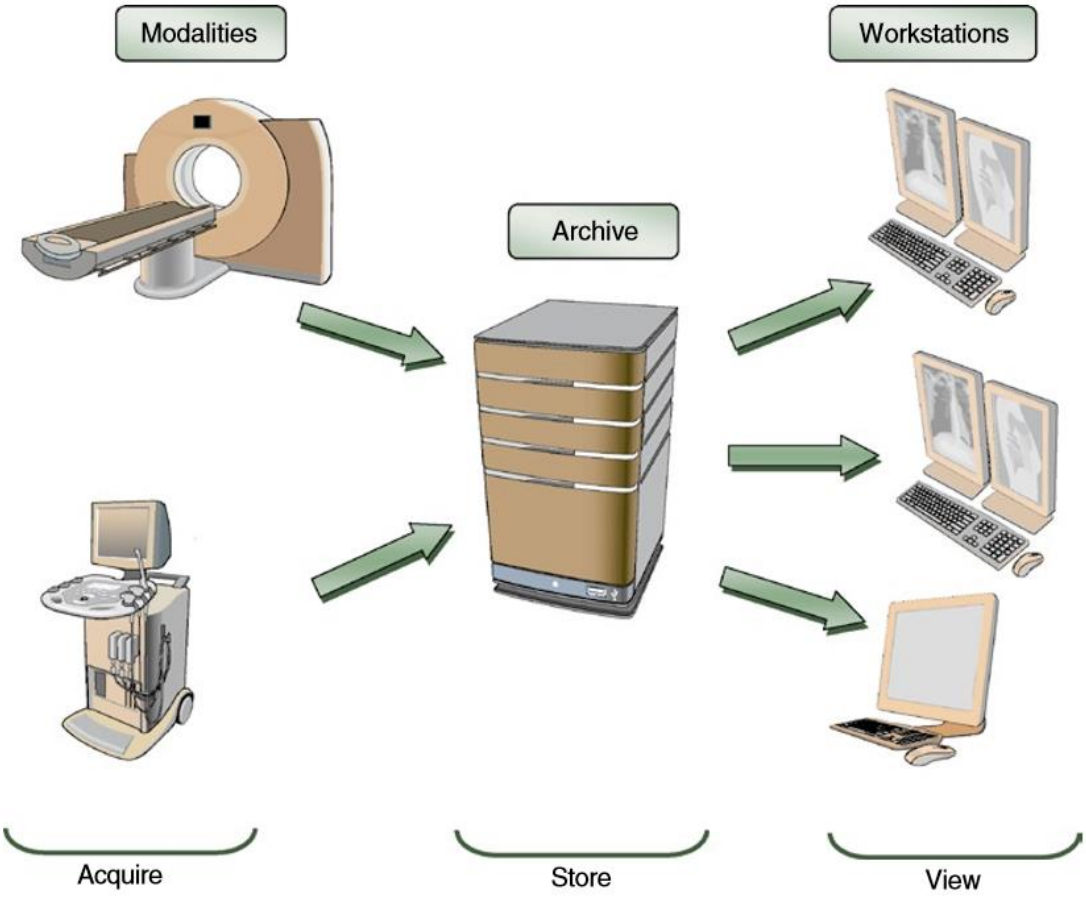

Figure 9. Main components of PACS [54]

All DICOM-driven PACS software and hardware devices comes with its own set of DICOM conformance statement. The conformance statement specifies to what extent a particular device supports and is compliant with the DICOM standard. The current DICOM standard consist of 20 volumes or parts usually written as PS3.1 to PS3.20 with PS3.9 and PS3.13 retired.

3.3.1 DICOM Information Model

DICOM models real world data such as devices, patients and studies based on the DICOM information model. The real-world data are represented as objects having attributes (or properties). Object together with their attributes are standardized by DICOM Information Object Definitions (IODs). Figure 10 shows a patient IOD which is consists of name, ID, date of birth, etc capturing all the necessary clinical information. The DICOM standards contains list of standard attributes for objects in order to be consistent in terms of formatting, processing and naming. The list of standard attributes is referred to as DICOM Data Dictionary. The attributes can have about 27 formats referred to as Value Representation (VR). VR types includes names, dates, times, etc.

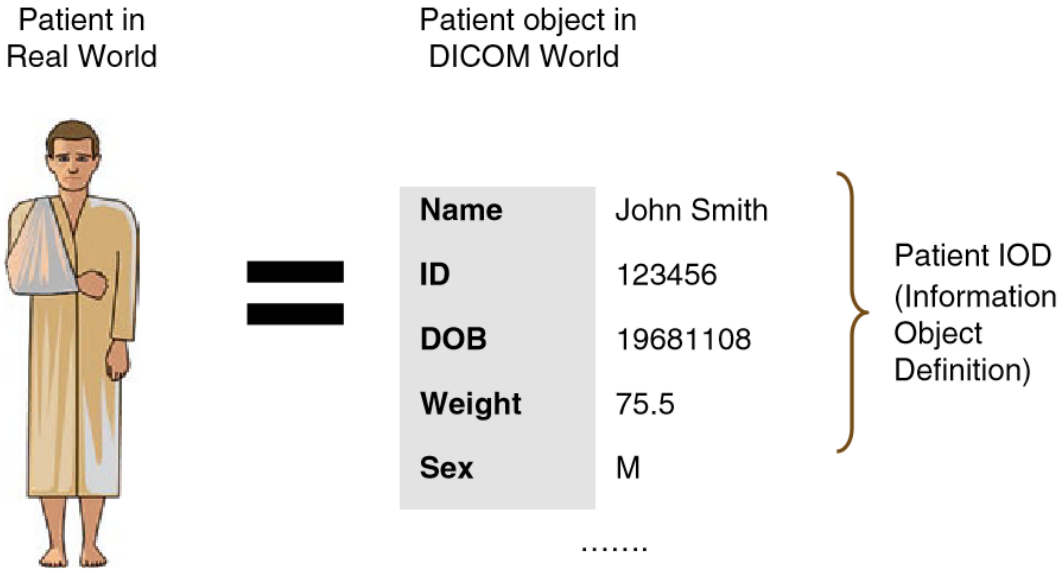

Figure 10. DICOM Information Object Definition (IOD) of a patient [54]

DICOM uses the service rendering model in order to exchange data between entities. Some DICOM terminologies relating to the service rendering model are as follows:

- Application Entities (AE): Application entities refers to all the DICOM driven devices and software exchange. AEs are able to provide services to each other
- Service-Object Pairs (SOPs): SOPs represents particular association of IODs with some service types which results in SOP classes
- Service Class Users (SCUs): These are entities requesting services from other DICOM-driven entities
- Service Class Providers (SCPs): SCPs receives and responds to requests from SCUs

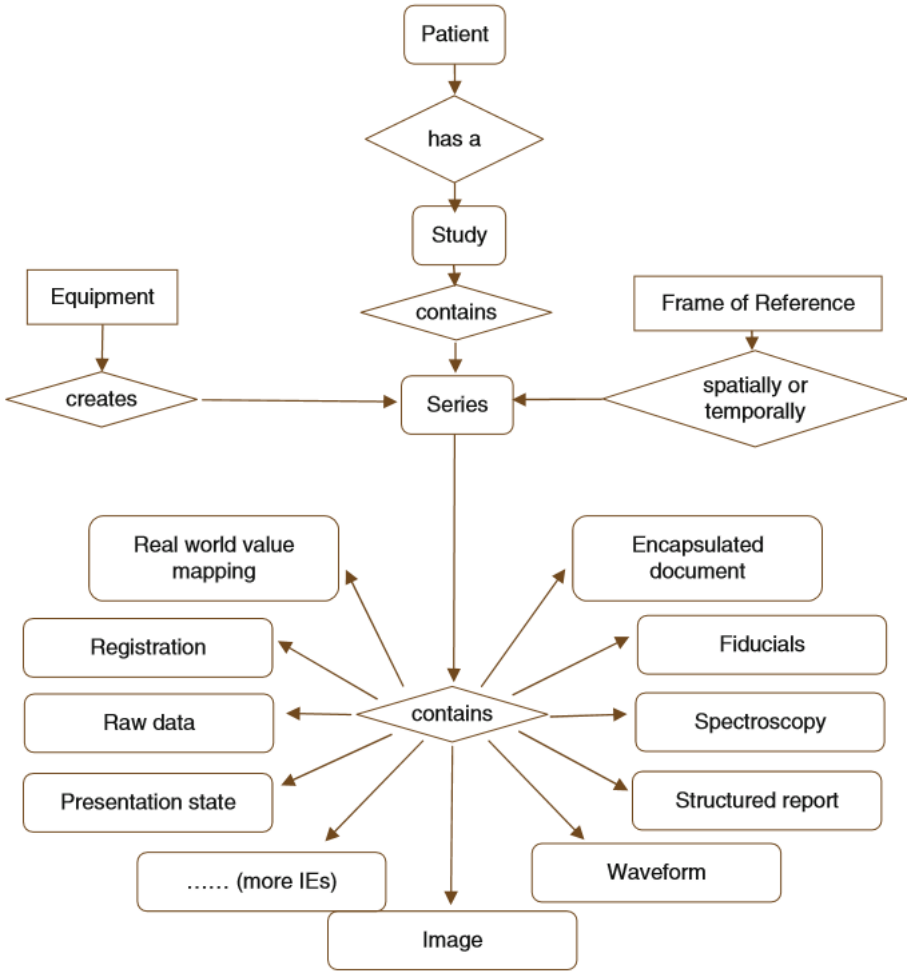

Figure 11. Overview of DICOM information model [54]

4 Integration of ITK/VTK/Qt

This section introduces the Insight Toolkit (ITK), the Visualization Toolkit (VTK) and the Qt framework. Moreover, it provides detailed instruction regarding the bundled environment (Ubuntu Linux virtual machine image) and its setup/configuration process in order to effectively integrate ITK/VTK/Qt together and perform 3D medical image segmentation and visualization.

4.1 The Insight Toolkit

The Insight Toolkit (ITK) is a comprehensive open-source software framework and toolkit for performing segmentation and registration. Segmentation can be viewed as the process for a digitally sampled representation of data undergoes identification and classification. The sampled data could be an image acquired from some medical equipment such as the CT scanners. Registration involves the process of making correspondences between data. ITK is cross-platform and uses the robust build environment known as CMake so as to manage all the platform specific intricacies during generation of project files and compilation process to allow for platform independence. ITK is developed using C++ and uses advanced generic (templated) programming paradigm. The use of templates makes the code highly efficient and most of ITK's algorithm can easily be applied to higher spatial dimension and different pixel types. ITK comes with a wrapping system that enables generation of interface between C++ and higher level languages such as Python.

4.2 The Visualization Toolkit

The visualization Toolkit (VTK) is an object-oriented based open-source software for computer graphics, image processing and visualization. Just like ITK, VTK also uses CMake as build tool. It compiles and runs on several platforms and operating systems (OS). CMake takes in independent a file named "CMakeLists.txt" from a specified source directory which describes the build process and all its possible dependencies. Basically, for the case of ITK and VTK, CMake is able to produce native build files that matches the operating system and compiler it is running on.

4.3 Qt Framework

Qt is a widely used cross-platform application framework for developing software which is intended to work on various combinations of software/hardware computing platforms with relative ease. Although Qt is mainly used in developing applications requiring graphical user interface (GUI), it can also be used in making console applications. Qt extends the standard C++ by introducing concepts such as signals and slots which makes it easier and more efficient to handle events in the case of GUIs or server applications. Qt supports several compilers such as the popular the GNU Compiler Collection for C++ (GCC C++) compiler and the Microsoft Visual Studio suite.

4.4 Configuration and installation process

The following outlines the steps necessary configure, build and install ITK/VTK and integrate with Qt for an Ubuntu Linux virtual machine running inside Oracle VM VirtualBox. The process was a combination of information from [55] - [72] and several individual changes/tweaking/trials until everything worked as expected.

1. In order to setup the environment, the following are required:

    a. Ubuntu ISO image (***Ubuntu 16.04.2 LTS***) [57]
    b. Oracle VM Virtual box binary (***VirtualBox-5.1.16-113841-Win***) [58] and install Ubuntu in it
    c. CMake binary (***cmake-3.7.2-Linux-x86_64.sh***) [59]
    d. Qt binary (***qt-opensource-linux-x64-5.8.0.run***) [60]
    e. VTK source code (***VTK 7.1.0***) [61]
    f. ITK source code (***ITK 4.11.0***) [62]

2. Steps to install Qt on Ubuntu (offline installer, Qt 5.8.0):

    a. From terminal run the following
        i. *sudo apt-get install build-essential libgl1-mesa-dev libfontconfig1*
        ii. *sudo apt-get install libglu1-mesa-dev -y*
        iii. *sudo apt-get install libxt-dev*

    b. Allow executing Qt installer qt-opensource-linux-x64-5.8.0.run and then install Qt
        i. *chmod +x qt-opensource-linux-x64-5.8.0.run*
        ii. *./qt-opensource-linux-x64-5.8.0.run*
        iii. Note the Qt install directory: /home/user/Qt5.8.0

3. Install CMake. CMake installation directory /home/user/cmake-3.7.2-Linux-x86_64

    a. *nano /home/user/.bashrc* and add line at the end of the .bashrc file*:*
        i. *export PATH="/home/$USER/cmake-3.7.2-Linux-x86_64/bin:$PATH"*

4.4.1 VTK setup

1. Launch CMake GUI from the terminal by typing cmake-gui and hitting return. The GUI should pop up as shown in Figure 12.

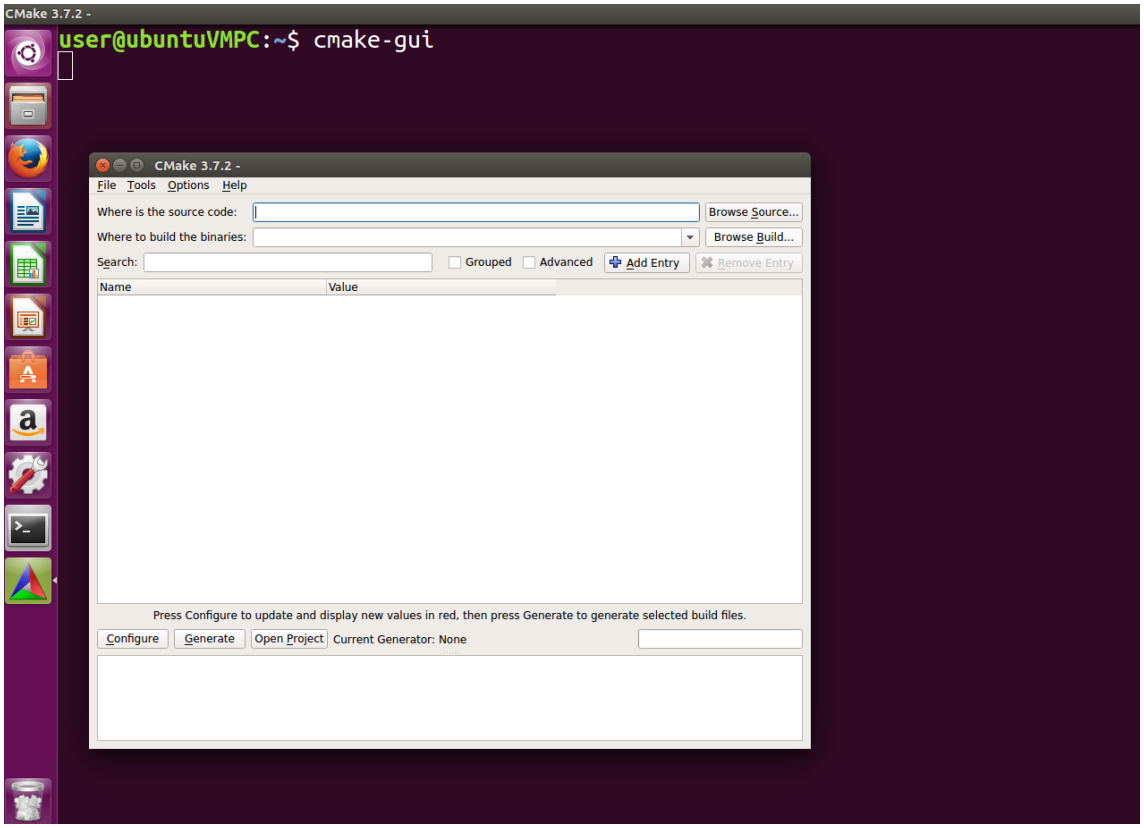

Figure 12. CMake GUI window

2. Source codes directory was chosen to have the following structure by separating source and build directories:

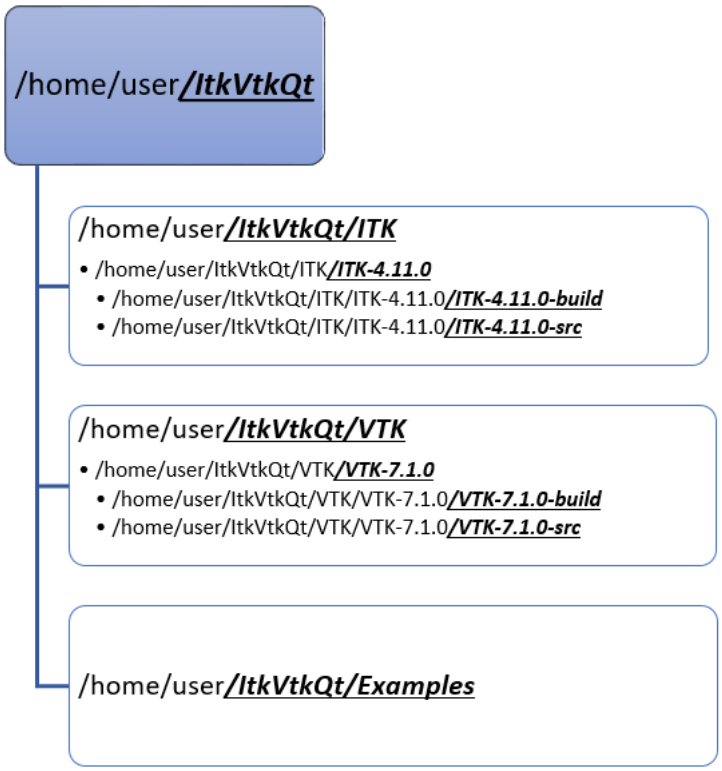

Figure 13. Source code directory structure

3. From CMake GUI, set the source and binary folders for VTK. Click configure and choose “Unix Makefiles” as generator for the project. Once configuration done, select appropriate options

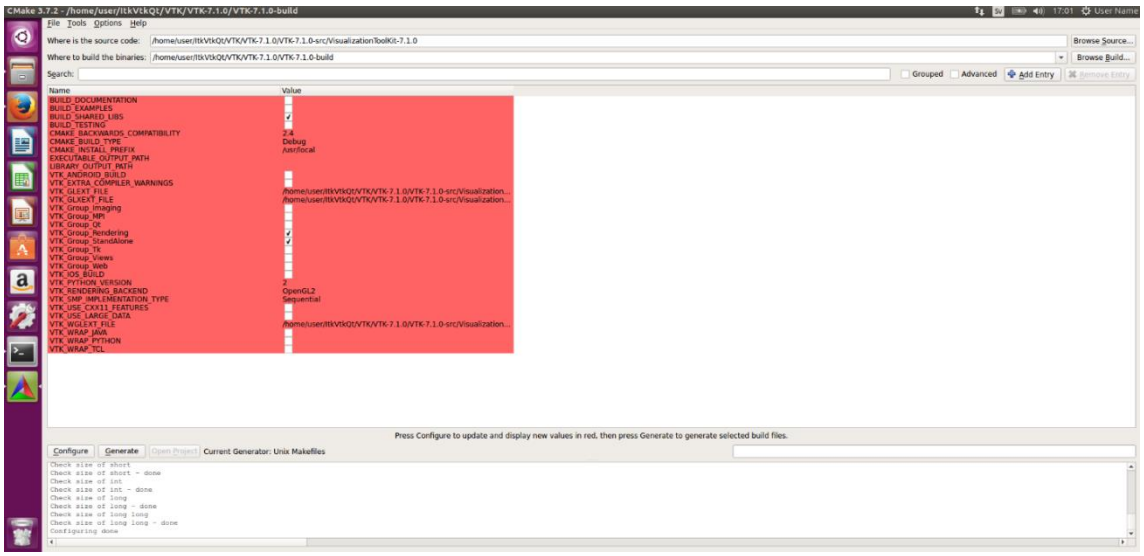

Figure 14. CMake initial configuration options for VTK

4. Click advanced button option. Necessary CMake configuration options for VTK are as follows:

    a. Check VTK_Group_Qt
    b. Check BUILD_SHARED_LIBS
    c. Uncheck BUILD_TESTING
    d. Uncheck BUILD_EXAMPLES
    e. Change CMAKE_BUILD_TYPE to Release
    f. Check VTK_ALL_NEW_OBJECT_FACTORY
    g. Check VTK_Group_Imaging
    h. Check VTK_MAKE_INSTANTIATORS
    i. Press Configure button again. There will be error with Qt version 4 and qmake path
    j. Change VTK_QT_VERSION to 5
    k. Set QT_QMAKE_EXECUTABLE to /home/user/Qt5.8.0/5.8/gcc_64/bin/qmake
    l. Press Configure button again. There will be error relating to Qt5 director or CMake prefix path
    m. Set QT5_DIR to /home/user/Qt5.8.0/5.8/gcc_64/lib/cmake/Qt5
    n. Add a variable CMAKE_PREFIX_PATH of type PATH with value /home/user/Qt5.8.0/5.8/gcc_64/lib/cmake
    o. Click Configure button again. It will highlight in red Qt/VTK options again but no error this time. Click Configure button on more time. This time there should be no more highlighted options/error.
    p. Now click the Generate button

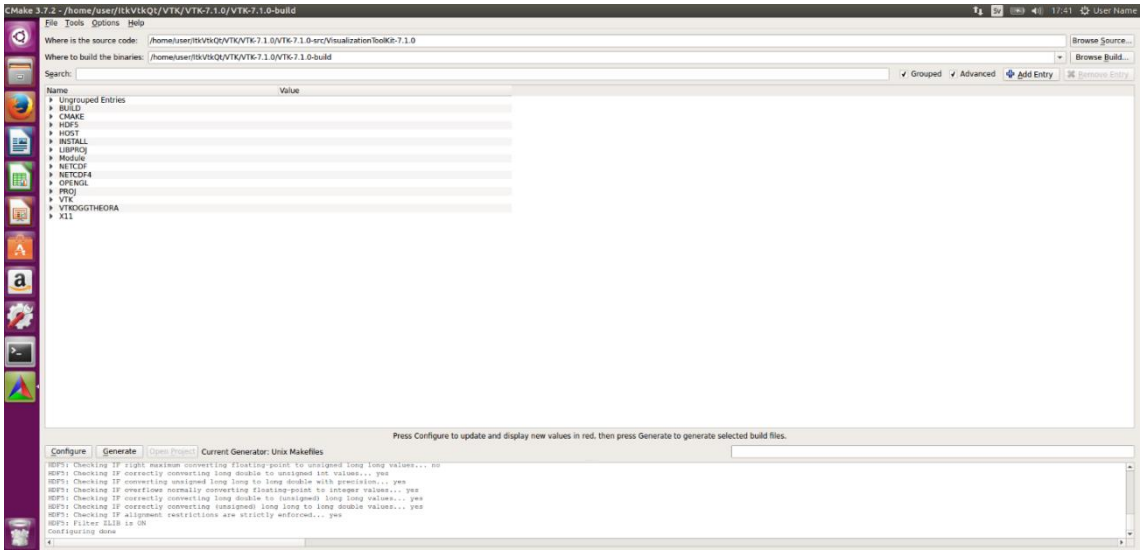

Figure 15. CMake GUI, final VTK configuration done

5. Once generating done, go to terminal and change path to /home/user/ItkVtkQt/VTK/VTK-7.1.0/VTK-7.1.0-build. Type make -j<# of processor cores> and return

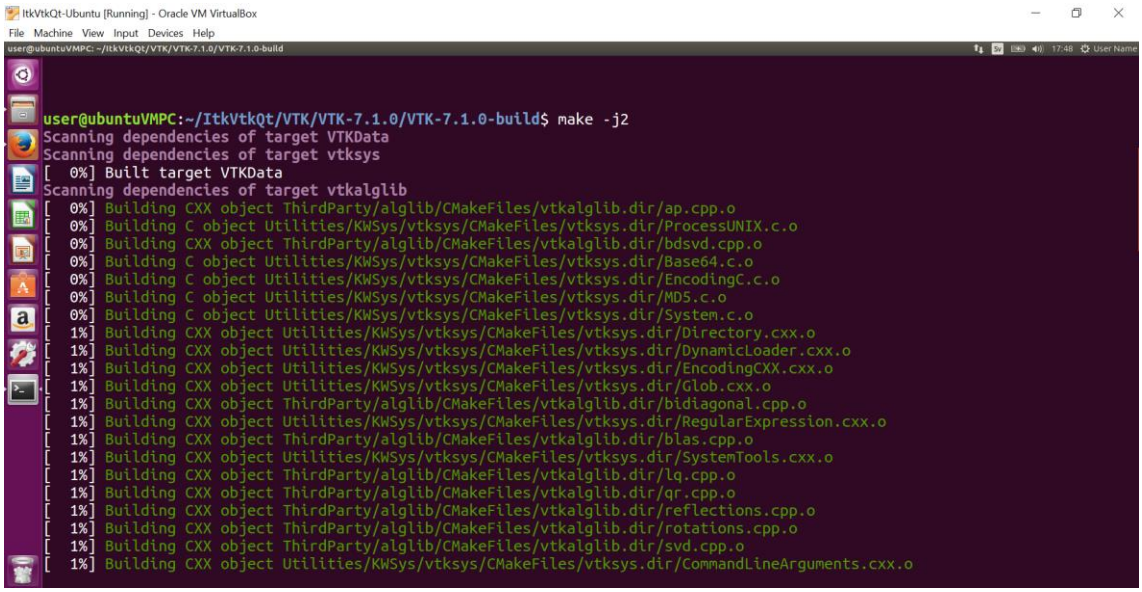

Figure 16. VTK build in progress after make -j2

6. After done building. Type sudo make install and enter password
    a. Note VTK install directory will be CMAKE_INSTALL_PREFIX which is /usr/local (files will be copied to bin, include and lib)

7. To be able to see QVTKWidget in Qt Creator/Designer, copy the file /home/user/ItkVtkQt/VTK/VTK-7.1.0/VTK-7.1.0-build/lib/libQVTKWidgetPlugin.so to /home/user/Qt5.8.0/Tools/QtCreator/lib/Qt/plugins/designer

4.4.2 ITK setup

1. To build and install ITK, launch CMake GUI again from terminal. Set ITK source and build directory, click configure and choose “Unix Makefiles” as generator for the project. Apply the following configuration options:
    a. Uncheck BUILD_EXAMPLES
    b. Check Module_ITKVtkGlue
    c. Uncheck BUILD_TESTING
    d. Check BUILD_SHARED_LIBS
    e. Set CMAKE_BUILD_TYPE as Release
    f. Note: by default, CMAKE_INSTALL_PREFIX is /usr/local

g. Set ITK_COMPUTER_MEMORY_SIZE as appropriate (RAM in GB). Default is 1
h. Check ITK_USE_64BITS_IDS
i. Uncheck ITK_DOXYGEN_HTML
j. Click Configure again. Some populated Qt options will be highlighted in red. Click configure once again. This time there should no highlighted options
k. Click Generate
l. Once generating done, go to terminal and change path to /home/user/ItkVtkQt/ITK/ITK-4.11.0/ITK-4.11.0-build. Type make -j<# of processor cores> and return
m. After done building. Type sudo make install and enter password

4.4.3 Generating project files and building applications using CMake and Qt

The sample application is in /home/user/ItkVtkQt/Examples/MyItkVtkQt/src. A CMake file is included "CMakeLists.txt" which is necessary to generate the project files/dependencies. The example source directory contains the following files:

- CMakeLists.txt
- CTDicomLungs3DVis.cxx
- CTDicomLungs3DVis.h
- CTDicomLungs3DVisMain.cxx
- CTDicomLungs3DVisUI.ui

The following steps are necessary in order to create native project files in CMake:

1. Open Ubuntu virtual machine and launch CMake GUI from the terminal by typing cmake-gui
2. Specify the source (/home/user/ItkVtkQt/Examples/MyItkVtkQt/src) and build directory for the project
    a. Click Configure and select "Unix Makefiles" as generator. CMake will populate some entries and highlight them in red. Set the option CMAKE_BUILD_TYPE to Release
    b. Click Configure again and there will be no more highlighted options in red
    c. Click Generate

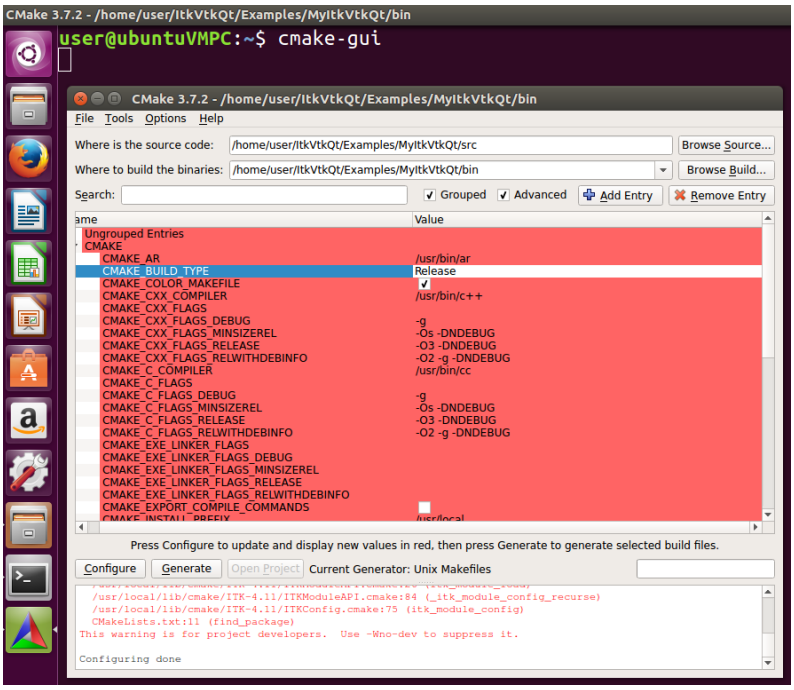

Figure 17. CMake project configuration options, setting CMAKE_BUILD_TYPE to Release

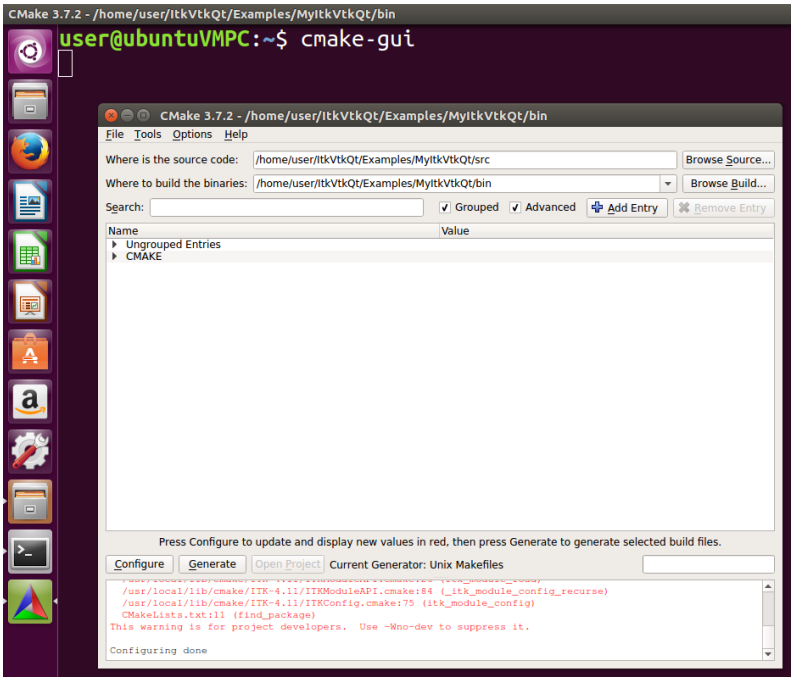

Figure 18. CMake configuration done, no more options highlighted in red

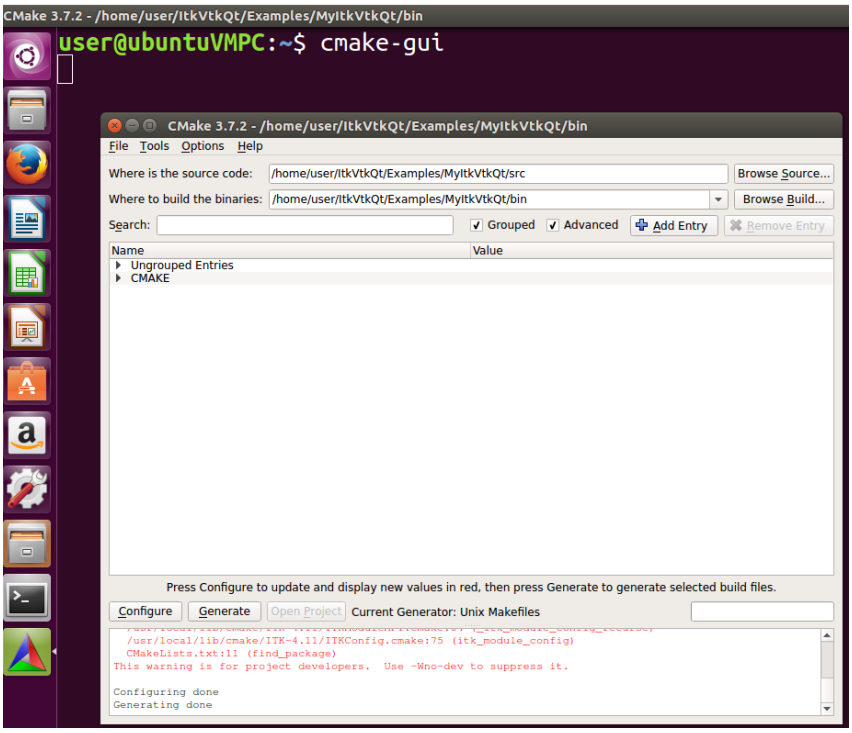

Figure 19. Application project files successfully generated

After successfully generating the project files, project can now be imported into Qt Creator. The following steps are necessary to import the project into Qt Creator:

1. Open Qt creator from Ubuntu Linux virtual machine
2. Go to menu and click “Open File or Project”
3. Select the file /home/user/ItkVtkQt/Examples/MyItkVtkQt/src/CMakeLists.txt and open it

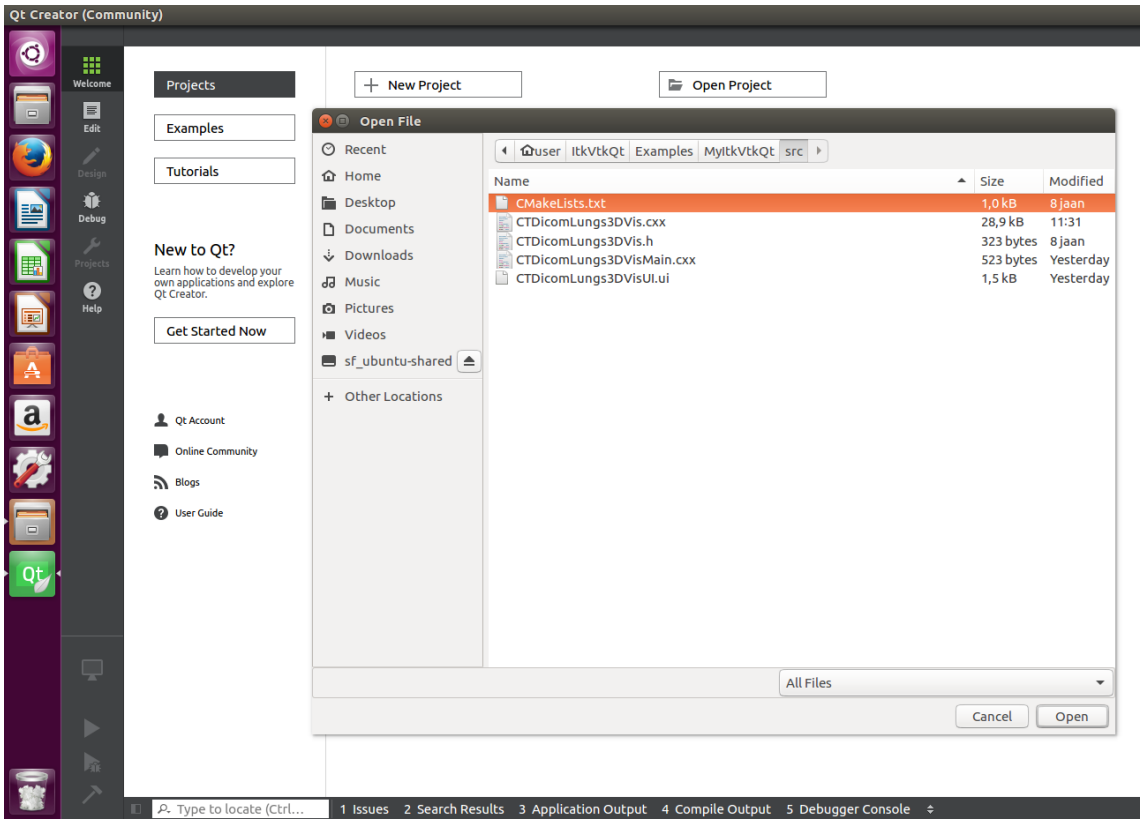

Figure 20. Opening project in Qt Creator IDE

4. After opening the “CMakeLists.txt”, Qt Creator displays its own project configuration page

    a. Click on “Details” to expand. Unselect the kits Default, Debug, Release with Debug Inforation and Minimum Size Release
    b. Only the kit Release should be selected. Click Browse to change the directory to /home/user/ItkVtkQt/Examples/MyItkVtkQt/bin
    c. Click Configure Project
    d. Qt Creator will show a prompt with changes, click “Apply”
    e. Project is ready and you can select “Build All” from the “Build” menu
    f. Click “Run” from the “Build” menu

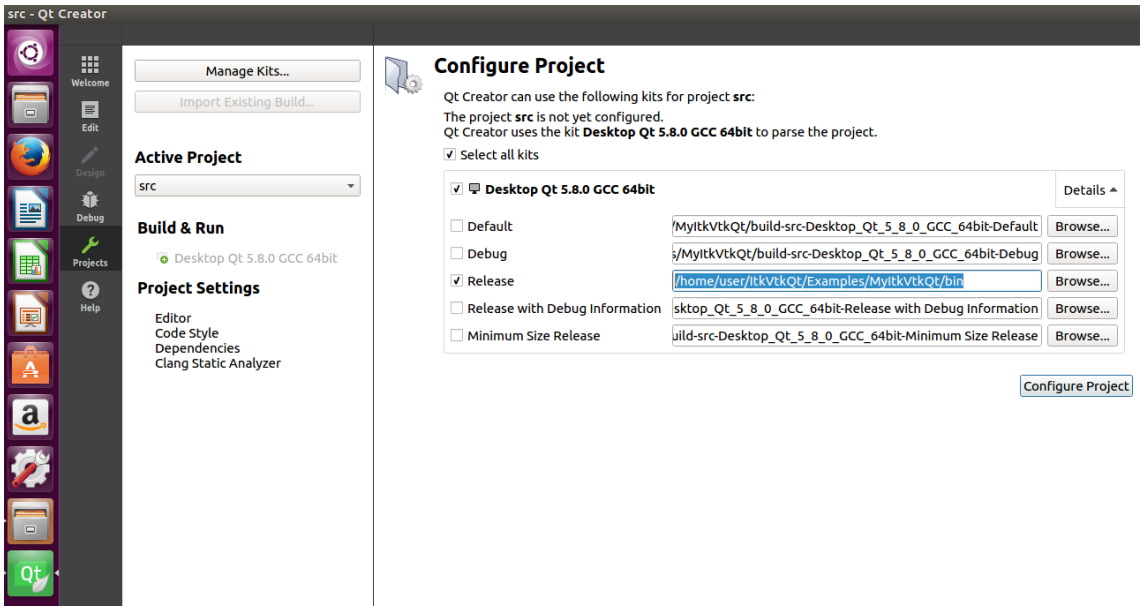

Figure 21. Qt creator configuration, selecting only the "Release" kit

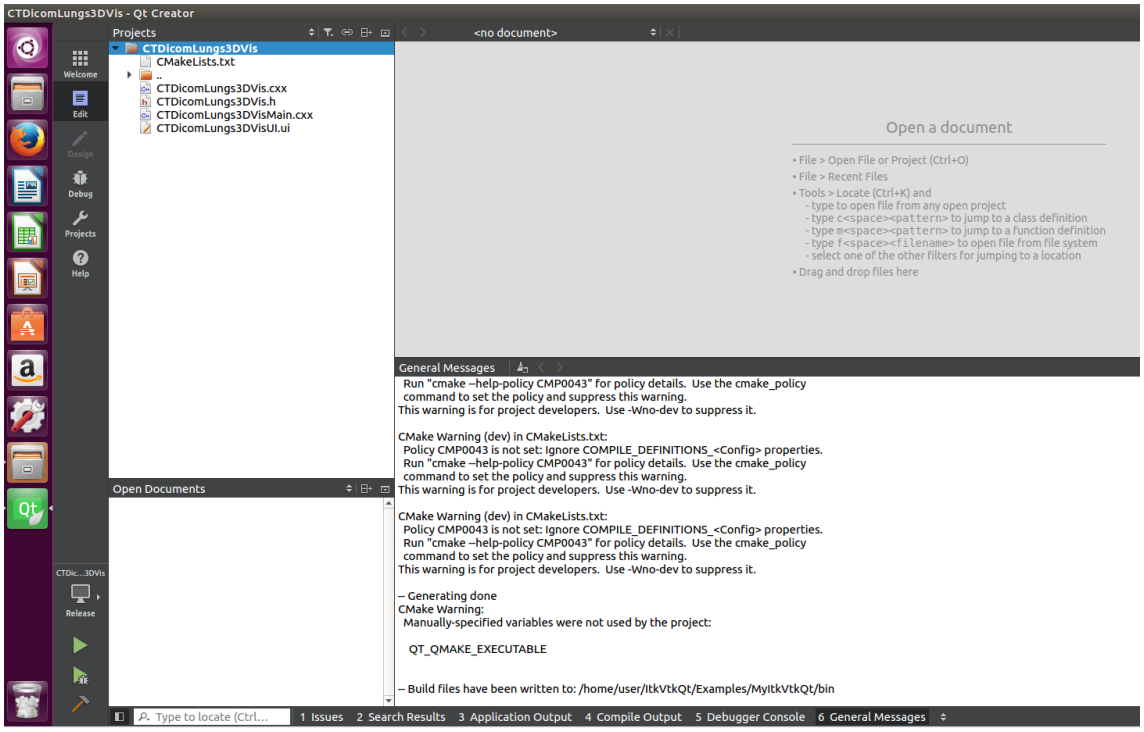

Figure 22. Qt project successfully configured and ready for build/run

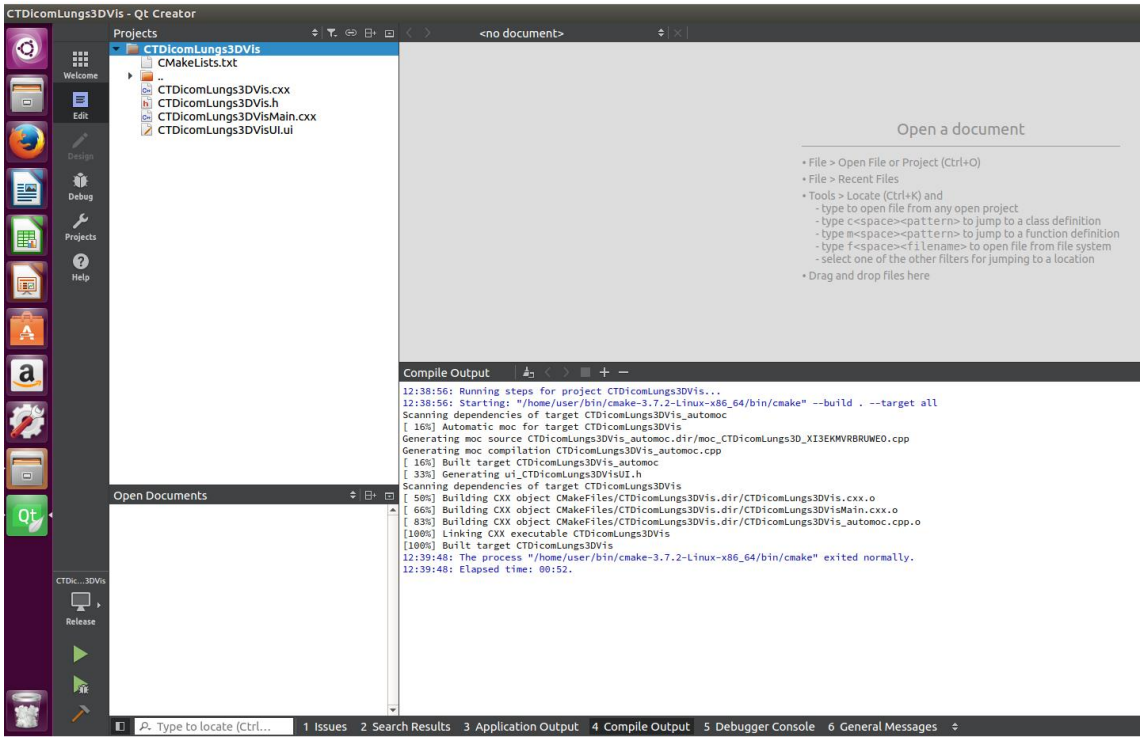

Figure 23. Project compilation/build output

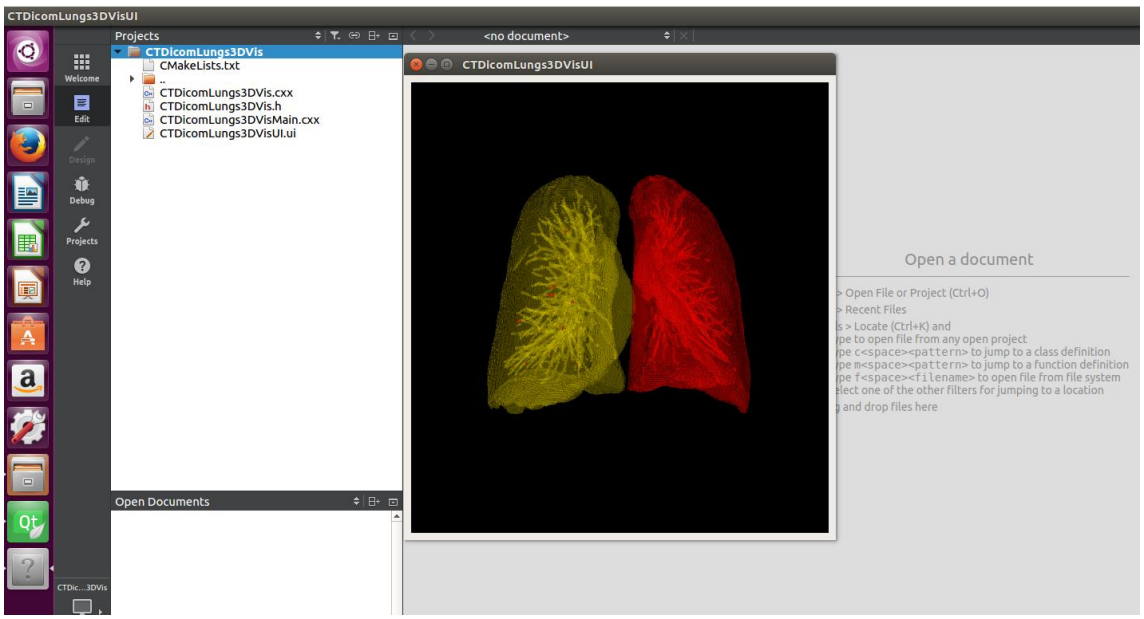

Figure 24. Sample application output

4.4.4 Python wrapping of C++ interface for ITK/VTK and PyQt setup

Ubuntu 16.04LTS comes with Python installed. In addition, the following setup is necessary to enable Python to work with ITK/VTK/Qt:

1. Install pip by typing sudo apt install python3-pip from terminal
   a. Upgrade to newer version by typing pip install --upgrade pip

2. Install SciPy stack
   a. From terminal, type pip3 install --user numpy scipy matplotlib ipython jupyter pandas sympy nose

3. Install PyQt by entering sudo pip3 install pyqt5 from terminal

4. For VTK, the following options necessary from CMake GUI

   a. Check VTK_WRAP_PYTHON
   b. Change VTK_PYTHON_VERSION to 3
   c. Run the command from terminal (pointing to /home/user/ItkVtkQt/VTK/VTK-7.1.0/VTK-7.1.0-build) make -j<# CPU cores> and afterwards sudo make install
   d. Run sample python vtk hello word application
   e. To .bashrc add the following paths:
      a. export PYTHONPATH=/usr/local/lib/python3.5/site-packages:/usr/local/lib/python3/dist-packages:$PYTHONPATH
      b. export LD_LIBRARY_PATH=/home/user/Qt5.8.0/5.8/gcc_64/lib:$LD_LIBRARY_PATH

   f. Sample Helloworld VTK application using Python wrapping [73] shown in Figure 25

5. For ITK, the following options necessary from CMake GUI

   a. Check ITK_WRAP_PYTHON
   b. Change PYTHON_EXECUTABLE to 3
   c. Wrapping requires to disable legacy code, check ITK_LEGACY_SILENT
   d. Press Configure, and then from highlighted options in red choose appropriate image types and dimensions to be wrapped e.g. ITK_WRAP_DOUBLE
   e. Press Configure again and there should be no more options highlighted in red
   f. Press Generate
   g. Run the command from terminal (/home/user/ItkVtkQt/ITK/ITK-4.11.0/ITK-4.11.0-build) make -j<# CPU cores> and afterwards sudo make install

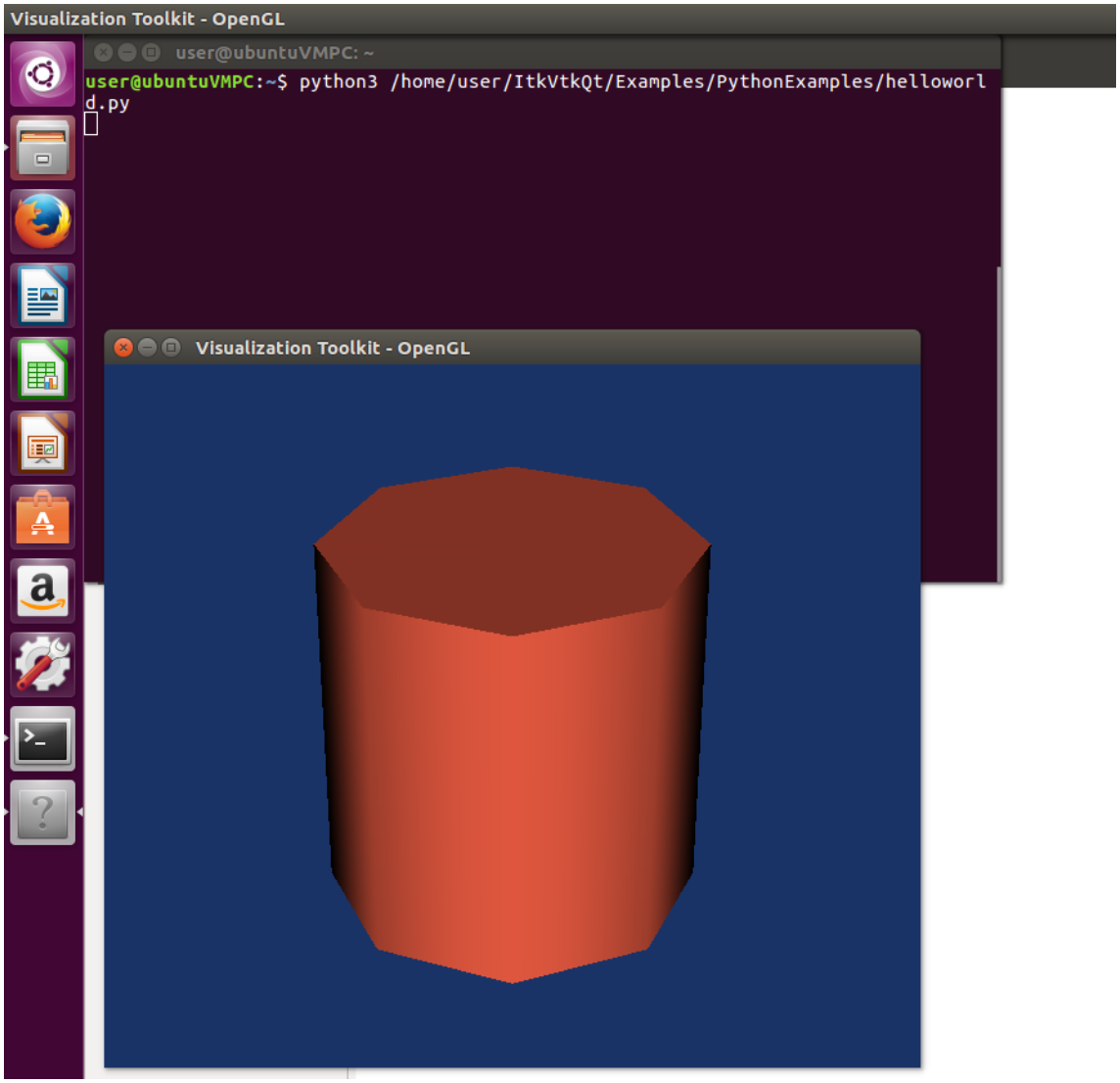

Figure 25. VTK sample helloworld application using Python wrapping

5 Implementation of 3D lung segmentation and visualization

The implemented lung segmentation starts by reading the DICOM file (".dcm") using ITK and then applying smoothing filter as a pre-processing step. The output of the smoothing filter is passed to the connected threshold filter for segmentation of the lungs. The effective Hounsfield value range for lungs seems to be between -250 to -980 which is not far from the range of -500 to -950 reported by the authors in [74]. The output from the connected threshold filter is binary. So in order to have image values in Hounsfield scale it is necessary to mask the output of connected threshold to the original input image. Mask image filter was used to achieve that. The output of the mask filter is passed on to rescale intensity filter to convert the image in to the range 1 to 255 for easier isosurface generation using the marching cube algorithm.

After finishing image processing using the ITK, it was necessary to pass the image to VTK for visualization. But internal image formats used by ITK and VTK are different and so the need for conversion. Pixel type is casted from float to unsigned short using the cast image filter before passing it to the ITK to VTK image filter. The output of the ITK to VTK image converter is used in generating VTK image volume data which is required for 3D surface extraction through marching cubes algorithm. Necessary isosurface value needs to be set as input to the marching cubes before passing the generated surface to VTK mapper. Output from the VTK mapper is then passed to VTK actor for setting opacity and colour for the generated 3D surface before passing it to the renderer. With the renderer ready, it can then be passed to Qt through the Qvtkwidget's renderer window and hence displayed on the screen. The entire source code can be found in Appendix. The following provides some key details regarding the implementation. C++ was used together with Qt Creator IDE for the main program code. The code in this thesis was inspired by the numerous examples from the ITK, VTK and Qt documentations and guides [55], [56], [75].

The algorithmic flow used for reading the DICOM series (CT scan input image), pre-processing the image, segmenting the lungs out of the whole image and visualizing it is shown in Figure 26.

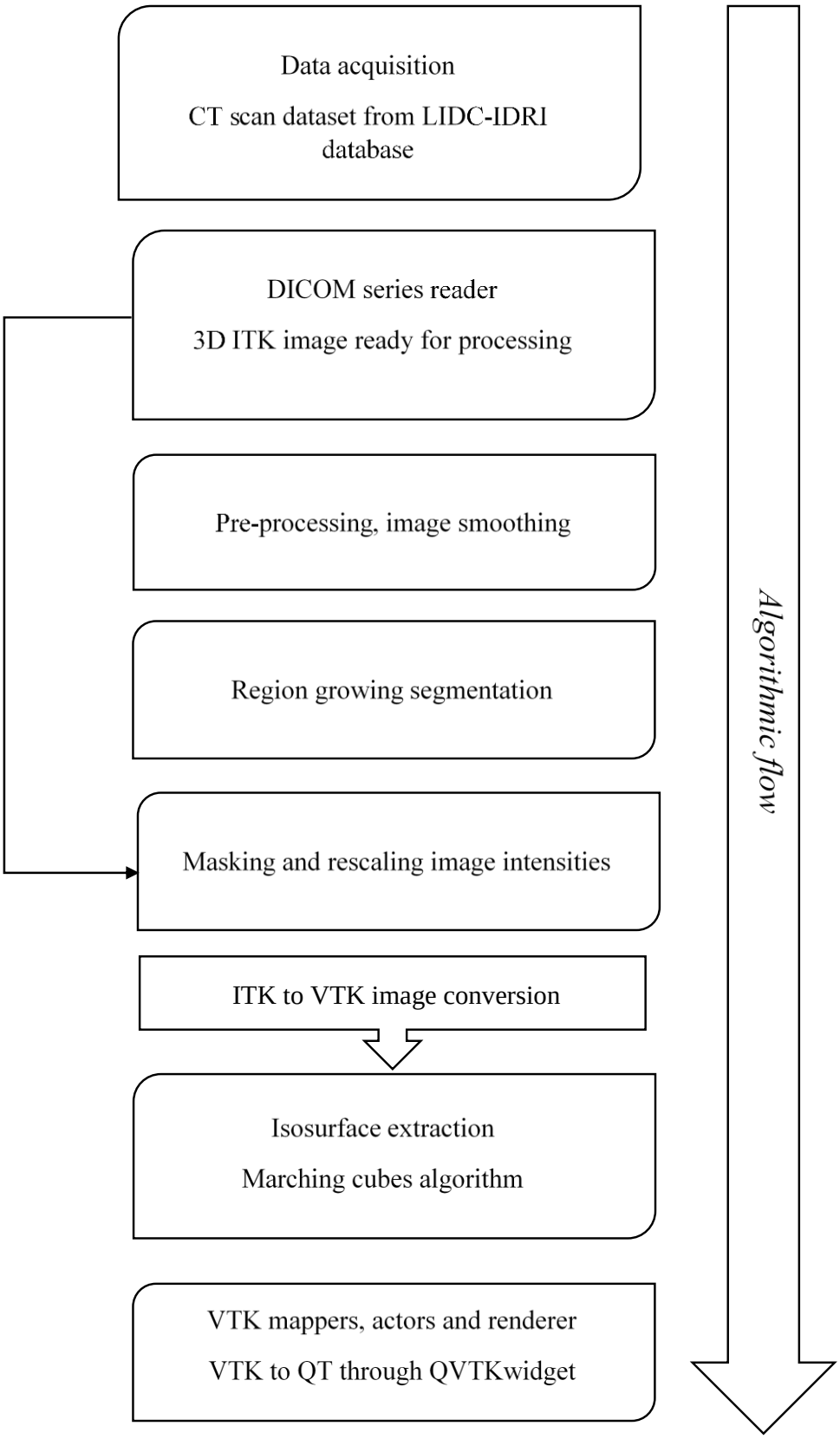

Figure 26. Algorithmic flow

5.1 Data Acquisition

As explained in chapter 2, CAD systems used for lung nodules detection composes of mainly the stages: data acquisition, pre-processing, lung segmentation, lung nodule detection and finally false positive reduction stage. The data acquisition step involves retrieving input images which could be through PACS. In this thesis, the data used come from the LIDC-IDRI database. For the generation of images in this chapter, the DICOM files "LIDC-IDRI-0029" from the database was used.

5.2 DICOM series reader

Reading the input CT scan DICOM files in ITK is handled by the Grassroots DICOM library (GDCM) library. GDCM is an open-source implementation of the DICOM standard mostly used by researchers to access clinical data. In order to read the DICOM series, GDCM IO object is passed to ITK series reader together with file name. Necessary header files are required to be included for using all relevant ITK, VTK and Qt classes. As such at the beginning of the implementation class "CTDicomLungs3DVis", all necessary headers were included as shown in the Figure 27 below:

```
#include "CTDicomLungs3DVis.h"
#include "vtkPolyDataMapper.h"
#include "vtkRenderer.h"
#include "vtkRenderWindow.h"
#include "vtkRenderWindowInteractor.h"
#include "vtkSmartPointer.h"
#include "vtkMarchingCubes.h"
#include "vtkActor.h"
#include "vtkProperty.h"
#include "itkImage.h"
#include "itkImageToVTKImageFilter.h"
#include "itkGDCMImageIO.h"
#include "itkGDCMSeriesFileNames.h"
#include "itkImageSeriesReader.h"
#include "gdcmUIDGenerator.h"
#include "itkConnectedThresholdImageFilter.h"
#include "itkCastImageFilter.h"
#include "itkCurvatureFlowImageFilter.h"
#include "itkMaskImageFilter.h"
#include "itkRescaleIntensityImageFilter.h"
#include "itkMaskImageFilter.h"
```

Figure 27. Header files included for used classes

The applicable headers for reading the DICOM series are "itkImageSeriesReader.h", "itkGDCMImageIO.h". "itkGDCMSeriesFileNames.h" and "gdcmUIDGenerator.h". As ITK uses templates, it is necessary to define object template types before instantiation and assigning to relevant smart pointers. ITK and VTK uses smart pointers to avoid the need for keeping track of memory that needs to be released when pointers are no longer referenced or in used anywhere in the program. Figure 28 shows the type definitions for the input DICOM file.

```
int ctrlSwitch = 4;
typedef   float           InPixelType;
typedef signed short    OutPixelType;
const unsigned int      Dimension = 3;
typedef itk::Image< InPixelType, Dimension >         InputImageType;
typedef itk::Image< OutPixelType, Dimension >         OutputImageType;
typedef itk::ImageSeriesReader< InputImageType >
        ReaderType;
ReaderType::Pointer reader = ReaderType::New();
typedef itk::GDCMImageIO
        ImageIOType;
ImageIOType::Pointer gdcmIO = ImageIOType::New();
typedef itk::GDCMSeriesFileNames
        InputNamesGeneratorType;
InputNamesGeneratorType::Pointer inputNames = InputNamesGeneratorType::New();
```

Figure 28. Type definitions for reading input DICOM series

The variable ctrlSwitch controls the behaviour of the program by taking values 1 to 4. As shown in Figure 27, The input pixel type is defined as float and output pixel defined as signed short which is the input as required by ITK to VTK image filter. For 3D image processing/segmentation and visualization, dimension of the input image is set to 3. Once the dimension and pixel type are ready, the ITK image types are created as *InputImageType* and *OutputImageType*. The *ImageSeriesReader* type is also defined as *ReaderType* using the input image type (passed to smart pointer *reader*). This is followed by the required *ImageIOType*, *InputNamesGeneratorType* and finally the *inputNames* as the smart pointer for reference the instantiated type. The input names will hold the files contained in the DICOM series which will be used by the ITK image series reader. The idea in ITK/VTK is that you create a class and pass the reference to a smart pointer which does the rest for you such as reference counting and deletion when out of scope.

Once types are defined, the input DICOM series is read as shown in Figure 29 below:

```
const ReaderType::FileNamesContainer & filenames =
            inputNames->GetInputFileNames();
reader->SetImageIO( gdcmIO );
reader->SetFileNames( filenames );
reader->Update();
```

Figure 29. Reading the DICOM series

The image series *reader* requires image IO type to be set through the method *SetImageIO* and the file names set through *SetFileNames*. Finally *update* method is called on the reader for actual execution. With this, the reader has the input image and can be read through its output port.

5.3 Pre-processing step

Once the input image is read, it necessary to apply some smoothing for noise reduction and making it more effective for the segmentation step. Fragment of code necessary for pre-processing the image is shown in Figure 30.

```
typedef itk::CurvatureFlowImageFilter< InputImageType, InputImageType >
            CurvatureFlowImageFilterType;
CurvatureFlowImageFilterType::Pointer smoothing =
            CurvatureFlowImageFilterType::New();
smoothing->SetInput( reader->GetOutput() );
smoothing->SetNumberOfIterations( 5);
smoothing->SetTimeStep( 0.01 );
```

Figure 30. Image pre-processing

For smoothing and denoising the image, curvature flow filter was used. The curvature flow image filter type is defined as having image type same as the input image type which is the output of the reader from section 5.1. This filter implements a curvature-driven denoising algorithm which is edge-preserving making it suitable prior to segmentation step. The *smoothing* filter requires two inputs which are the number of iterations and time step. The time step had to be finetuned and a value of 0.01 seems to give reasonable output. The input to the filter is set using *SetInput* and passing the output from the reader by calling *reader->GetOutput()*.

5.4 Region-growing segmentation

The ITK's connected threshold image filter is a region growing-based segmentation filter and uses the flood-fill iterator. It requires visiting neighbouring pixels of the image. A

criterion is specified as to whether a particular pixel will be in a region or not. This filter requires to be set the lower/upper threshold, replace value and the seed. The output from the connected threshold filter is a binary image and as such original image's Hounsfield values are lost which requires masking to get them back. The implementation code for the connected threshold filter is shown in Figure 31.

```
typedef itk::ConnectedThresholdImageFilter< InputImageType,
        InputImageType > ConnectedFilterType;
ConnectedFilterType::Pointer connectedThreshold1 =
ConnectedFilterType::New();
const InPixelType lowerThreshold1 = -990;
const InPixelType upperThreshold1 = -250;
InputImageType::IndexType  index;
index[0] = 152;
index[1] = 277;
index[2] = 152;
connectedThreshold1->SetInput( smoothing->GetOutput() );
connectedThreshold1->SetLower(  lowerThreshold1  );
connectedThreshold1->SetUpper(  upperThreshold1 );
connectedThreshold1->SetReplaceValue( 255 );
connectedThreshold1->SetSeed( index );
connectedThreshold1->Update();
```

Figure 31. Connected threshold filter

As with most classes in ITK, the filter type needs to be specified. Filter type was defined as *ConnectedFilterType* having two inputs both of type *InputImageType*. The second input servers as the filter's output type. The filter type is then passed to a smart pointer named *connectedThreshold1*. This filter requires and upper lower threshold limit to be set which should corresponds to the limits containing the lungs Hounsfield value. The lower limit was set to -990 and upper limit to -250. These values for the limits were arrived at after several adjustments and testing to find the range giving the best 3D segmentation output. It is necessary to have a seed value corresponding to a location in any of the lungs (left or right would suffice). Although additional seeds could be added but did not seem to make much difference after trying with many seeds for this particular connected threshold filter. The output from the curvature flow filter *smoothing->GetOutput()* is used as the input to connected threshold filter through the method *SetInput*. Similar, lower/upper threshold, replace value, and seed where set respectively using the methods *SetLower*, *SetUpper*, *SetReplaceValue* and *SetSeed*.

5.5 Masking and rescaling

ITK mask image filter uses the output of the connected threshold as mask in order to get the Hounsfield value of the segmented lungs from the original input CT image. By the used of rescale intensity filter, intensities are scaled to the range 0 to 255 to allow for better and faster 3D surface generation. The rescale intensity image filter takes three inputs including output from mask filter, output minimum and output maximum value. Figure 32 shows the portion of the code for masking and rescaling filters.

```
typedef itk::MaskImageFilter< InputImageType, InputImageType >
MaskFilterType1;
MaskFilterType1::Pointer maskFilter1 = MaskFilterType1::New();
typedef itk::RescaleIntensityImageFilter< InputImageType, InputImageType >
RescaleFilterType1;
RescaleFilterType1::Pointer rescaleFilter1 = RescaleFilterType1::New();
maskFilter1->SetInput(reader->GetOutput());
maskFilter1->SetMaskImage(connectedThreshold1->GetOutput());
maskFilter1->Update();
    switch (ctrlSwitch) {
    case 1:
        rescaleFilter1->SetInput(reader->GetOutput());  // 1 ==> Full CT
input image
        break;
    case 2:
        rescaleFilter1->SetInput(reader->GetOutput());  // 2 ==> Full CT
input image without cover
        break;
    case 3:
        rescaleFilter1->SetInput(maskFilter1->GetOutput()); // 3  ==>
segmented lungs (opaque)
        break;
    case 4:
        rescaleFilter1->SetInput(maskFilter1->GetOutput()); // 4  ==>
segmented lungs (transparent)
        break;
    default:
        break;
    }
rescaleFilter1->SetOutputMinimum(0);
rescaleFilter1->SetOutputMaximum(255);
rescaleFilter1->Update();
```

Figure 32. Masking and rescaling filters

Both the type for masking filter *MaskFilterType1* and rescale intensity filter *RescaleFilterType1* takes two input of image types *InputImageType*. Subsequently passed to smart pointers *maskFilter1* and *rescIaleFilter1*. The mask filter takes its input from the DICOM series reader and the mask input image from connected threshold filter since output of the segmentation is binary. The behaviour of the rescale intensity filter (and the program as a whole) is determined by the value of the control switch variable *ctrlSwitch* which takes values ranging from 1 to 4. The value "1" is used in displaying the original input image and the rescale intensity filter in this case takes its input image from the output of the DICOM series reader. The value "2" is used in displaying modified version of the input image in which the cover is removed to show the body of the patient in the CT scan image and rescale intensity filter in this case also takes input from the output of the DICOM series reader. When the value of the variable *ctrlSwitch* is set to "3", the result from the lung segmentation is displayed with opacity 1 and rescale intensity filter takes it input from the mask filter. Similarly, the value "3" displays the result from lung segmentation but having an opacity of 0.2 and rescale intensity filter takes input from mask filter.

5.6 ITK to VTK conversion and 3D visualization via Qt

With the image processing and segmentation part done in ITK, the image is passed using ITK to VTK image filter as bridge because the formats for volume data representation in VTK and ITK are different. For 3D visualization, VTK image data is generated from the output of ITK to VTK image filter which is required by the marching cubes for isosurface generation. The marching cubes requires VTK volume data with an isovalue(s). Isovalues could be automatically generated (evenly spaced). The code fragment for ITK to VTK conversion and 3D surface extraction (marching cubes) is shown in Figure 33. The cast image filter is used in casting float image type to short as the ITK to VTK image filter requires short. Therefore, the cast filter takes two inputs of type *InputImageType* and *OutputImageType*. The input from the cast filter comes from rescale intensity filter and its output goes to ITK to VTK filter. The ITK to VTK image filter takes the output image from cast filter and converts the image type into VTK compatible image thus enabling the generation of VTK image data *volume1*. The VTK marching cubes takes the volume data together with appropriate isovalues and extract the surface which is finally displayed.

```
typedef itk::CastImageFilter< InputImageType, OutputImageType >
CastingFilterType;
CastingFilterType::Pointer caster1 = CastingFilterType::New();
typedef itk::ImageToVTKImageFilter<OutputImageType>       ConnectorType;
ConnectorType::Pointer connector1 = ConnectorType::New();
vtkSmartPointer<vtkImageData> volume1 = vtkSmartPointer<vtkImageData>::New();
vtkSmartPointer<vtkMarchingCubes> surface1 =
vtkSmartPointer<vtkMarchingCubes>::New();
caster1->SetInput( rescaleFilter1->GetOutput() );
connector1->SetInput(caster1->GetOutput());
connector1->Update();
volume1->DeepCopy(connector1->GetOutput());
surface1->SetInputData(volume1);
surface1->ComputeNormalsOn();
surface1->SetValue(0,180);
```

Figure 33. ITK to VTK bridge and marching cubes

5.7 Segmentation Output

This sections shows sample screenshots from the output of the application. Before segmenting the lungs from the rest of the image, the CT input image looks as shown in Figure 34 and 35. Figure 36 and 37 shows the segmented lungs from the rest of the body (CT scan).

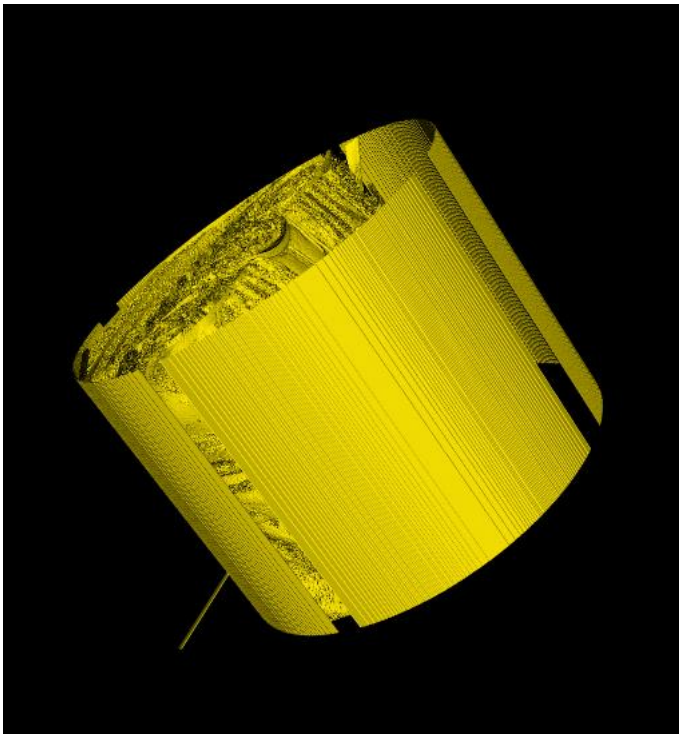

Figure 34. Input CT image before pre-processing

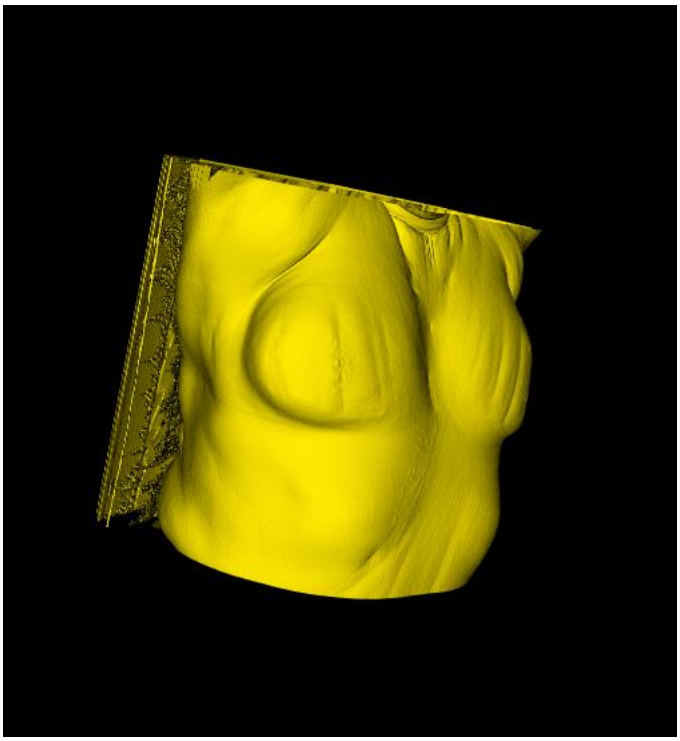

Figure 35. Input CT image after removing the covering

Figure 36. Segmented lungs from input CT with opacity 1

Figure 37. Segmented lungs from input CT with opacity 0.2 (transparent)

5.8 Challenges and lessons learnt

Certainly, medical image processing and analysis is a vast field. Moreover, due to the inter-disciplinary nature of the field, it requires bits and pieces of knowledge from other fields such as physics, digital signal processing, software programming and engineering, some part of medicine such as anatomy, perhaps advanced mathematics and so on. This makes it quite challenging to undertake any considerable research if coming from a background without strong foundation (as the case with the author) especially in biomedical engineering and image analysis. So many new concepts had to learnt through self-study and within very short period of time. As such, this thesis serves only as a stepping stone or beginning which is more or less like initial exploration and map construction with much details to be further explored later on.

Considering that this thesis focused on using open source alternatives such as ITK and VTK, it would have been much better starting out with easier environmental setup such as MATLAB instead. Despite the indispensable capabilities that using full-fledged ITK (C++) brings to the table, it simply does not seem a viable candidate for prototyping due

to the compile/build cycle which sometimes even requires starting from project file (re)generation using CMake. Prototyping seems much better done using interpreted languages such as Python. And once a working prototype is ready, then it can be converted into a full and efficient application using the complete ITK/C++ and visualization through VTK/Qt.

5.9 Alternative approach

As discussed in chapter 1 under literature review, the main steps that are involved in medical image segmentation and nodule detection are: data acquisition, pre-processing, lung segmentation, nodule detection and false positive reduction. In this thesis, curvature flow image filter was used in the pre-processing step in order to denoise the input image while preserving edges for better segmentation. As an alternative approach, other techniques could be explored such those in [12] - [15] and compared against the one used in this thesis. More importantly, semi-automated region growing-based segmentation algorithm was used in this thesis for which an automated approach could be a better alternative. One of the alternative automated segmentation approach could be to use the optimal thresholding technique and create appropriate masks afterwards to extract the lungs (and airways also) which could then be followed by some morphological operation in case of holes in the resulting segmentation. With the automated segmentation, it could be taken a step further in order to detect candidate lung nodules and perhaps look into false positive reduction.

6 Summary, conclusion and future work

With the increasing significance of lung cancer leading to many deaths worldwide, the need to have an accurate assisted diagnostic tool for used by radiologists cannot be underestimated. Lung nodules appearing as spot on the lungs are usually accidently encountered during patients' chest x-ray. As not all nodules are potentially malignant and cancerous, several CAD techniques are being developed to address the possibility to automatically detect and filter out true nodules from false candidate nodules. The difficulty in long nodules segmentation lies in their irregularities in terms of shape, size and location. Ground glass density nodule (GGN) could be part solid nodule (PSN) solid or non-solid, nodules could appear as fully circumscribed in the lungs, juxtapleural or even juxtavascular with possibly varying textures. Assisted diagnostic tools need to have high sensitivity with the ability to efficiently detect lung nodules together with high true positive rates. Moreover, computer assisted or aided diagnostic tools necessarily need to have high speed processing rate coupled with high level of automation requiring minimal intervention. These systems are also required to have low cost, maintenance and training requirements.

With all these challenges on desk, most of the researches published in the literature tend to follow similar general strategies and steps which begins with acquiring data coming from different modalities. The acquired data undergoes some form of pre-processing in order to make the segmentation more effective. Commonly used pre-processing includes smoothing and Gaussian filtering, some form of histogram equalization, linear interpolation and also some morphological operations. After pre-processing comes the lung segmentation steps which employs approaches broadly such as thresholding, shape-based and edge detection-based. Once the lungs are segmented from the rest of the body, further processing methods such as multi-level thresholding, artificial neural network or other enhanced approaches are used in detecting candidate nodules. Sometimes this is followed by the even further advanced techniques for false positive reduction.

One of the open-source framework used by researches for implementing medical image processing algorithms for lung nodules segmentation and detection is the Insight Toolkit (ITK) which is cross-platform. Although a limited and simplified interface to ITK exists, often early researches find it difficult to get up and running with full templated/generic

C++ based ITK. In this thesis project, the author having experienced such, and developed a bundled virtual environment from ground up and ready to be used right away without any additional setup pain. This will allow especially inexperienced early stage researchers to focus mainly on spending more time working with real segmentation and detection problems rather than the technical aspects regarding building ITK from source. Moreover, the environment is integrated with the Visualization Toolkit (VTK) and Qt to make visualization (especially 3D) seamlessly and easier. In addition to the bundled environment, the author also implemented as a stepping stone into the medical image analysis, a program using semi-automated connected threshold image segmentation (region-based) to successfully segment CT image from the LIDC-IDRI database.

Certainly, there are improvements that can be done in the future including much deeper review of the literature especially fully automating the segmentation step and also implementing the other two steps of nodule detection and false positive reduction. Furthermore, experimenting with much larger CT scan datasets and thorough comparison with established results from literature could also be done in the future.

References

[1] R. L. Siegel, K. D. Miller, and A. Jemal, "*Cancer statistics*", 2016. CA: A Cancer Journal for Clinicians, 66: 7–30.

[2] S. G. Armato, G. McLennan, L. Bidaut, M. F. McNitt-Gray, C. R. Meyer, A. P. Reeves, L. P. Clarke, "*The Lung Image Database Consortium (LIDC) and Image Database Resource Initiative (IDRI): A Completed Reference Database of Lung Nodules on CT Scans*", 2011, NCBI, Medical Physics, 38(2), 915–931.

[3] I. R. Valente, P. C. Cortez, E. C. Neto, J. M. Soares, V. H. de Albuquerque, and J. M. Tavares, "*Automatic 3D pulmonary nodule detection in CT images: A survey*", Computer Methods and Programs in Biomedicine, vol 124, February 2016, pp 91–107.

[4] I. Sluimer, A. Schilham, M. Prokop, and B. van Ginneken, "*Computer analysis of computed tomography scans of the lung: a survey*", IEEE Trans Med Imaging, 2006, vol 25, pp 385-405

[5] A. A. Farag, H. E. El Munim, and J. H. Graham, "*A novel approach for lung nodules segmentation in chest CT using level sets*", IEEE Trans Image Process, 2013, vol 22, pp 5202-13

[6] B. van Ginneken, C. M. Schaefer-Prokop, and M. Prokop, "*Computer-aided diagnosis: how to move from the laboratory to the clinic*", 2011, RSNA, Radiology vol 261, no.3, pp 719–732

[7] W. -J. Choi and T.-S. Choi, "*Automated pulmonary nodule detection based on three-dimensional shape-based feature descriptor*", 2014, Computer Methods and Programs in Biomedicine, vol 113 (1), pp 7–54

[8] A. M. Santos, A. O. de Carvalho Filho, A. C. Silva, A. C. de Paiva, R. A. Nunes, and M. Gattass, "*Automatic detection of small lung nodules in 3D CT data using Gaussian mixture models, Tsallis entropy and SVM*", 2014, Engineering Applications of Artificial Intelligence, 36, pp 27–39

[9] P. Badura and E. Pietka, "*Soft computing approach to 3D lung nodule segmentation in CT*", 2014, Computers in Biology and Medicine, 53, pp 230–243

[10] A. El-Baz, A. Elnakib, M. Abou El-Ghar, G. Gimel'farb, and R. Falk, A. Farag, "*Automatic Detection of 2D and 3D Lung Nodules in Chest Spiral CT Scans*", International Journal of Biomedical Imaging, 2013

[11] Q. Wang, W. Kang, C. Wu, and B. Wang, "*Computer-aided detection of lung nodules by SVM based on 3D matrix patterns*", 2013, NCBI, Clinical Imaging, 37(1), pp 62–69

[12] D. Cascio, R. Magro, F. Fauci, and M. Iacomi, G. Raso, "*Automatic detection of lung nodules in CT datasets based on stable 3D mass-spring models*", 2012, Computers in Biology and Medicine, 42(11), pp 1098–1109

[13] B. Chen, T. Kitasaka, H. Honma, H. Takabatake, M. Mori, H. Natori, and K. Mori, "*Automatic segmentation of pulmonary blood vessels and nodules based on local intensity structure analysis and surface propagation in 3D chest CT images*", 2012, International Journal of Computer Assisted Radiology and Surgery, 7(3), pp 465–482

[14] S. Soltaninejad, M. Keshani, and F. Tajeripour, "*Lung nodule detection by KNN classifier and active contour modelling and 3D visualization*", 2012, IEEE, The 16th CSI International Symposium on Artificial Intelligence and Signal Processing (AISP 2012), Iran, pp. 440–445

[15] W. Suiyuan and W. Junfeng, "*Pulmonary Nodules 3D Detection on Serial CT Scans*", 2012, IEEE, Third Global Congress on Intelligent Systems, Wuhan, China, pp. 257–260

[16] A. Riccardi, T. S. Petkov, G. Ferri, M. Masotti, and R. Campanini, "*Computer-aided detection of lung nodules via 3D fast radial transform, scale space representation, and Zernike MIP classification*", 2011, NCBI, Medical Physics, 38(4), pp 1962–1971

[17] Y. Liu, J. Yang, D. Zhao, and J. Liu, "*A Study of Pulmonary Nodule Detection in Three-Dimensional Thoracic CT Scans*", 2010, IEEE, Second International Conference on Computer Modeling and Simulation, vol 1, Sanya, China, pp. 481–484

[18] S. Taghavi Namin, H. Abrishami Moghaddam, R. Jafari, M. EsmaeilZadeh, and M. Gity, "*Automated detection and classification of pulmonary nodules in 3D thoracic CT images*", 2010, IEEE International Conference on Systems, Man and Cybernetics, Istanbul, Turkey, pp. 3774–3779

[19] S. Matsumoto, Y. Ohno, H. Yamagata, D. Takenaka, and K. Sugimura, "*Computer-aided detection of lung nodules on multidetector row computed tomography using three-dimensional analysis of nodule candidates and their surroundings*", 2008, NCBI, Radiation Medicine, 26(9), pp 562– 569

[20] S. Ozekes, O. Osman, "*Computerized Lung Nodule Detection Using 3D Feature Extraction and Learning Based Algorithms*", 2010, Springer, Journal of Medical Systems, 34(2), pp 185–194

[21] S. Ozekes, O. Osman, and O. N. Ucan, "*Nodule detection in a lung region that's segmented with using genetic cellular neural networks and 3D template matching with fuzzy rule based thresholding*", 2008, Korean Journal of Radiology, 9(1), pp 1–9

[22] M. Yang, S. Periaswamy, and Y. Wu, "*False Positive Reduction in Lung GGO Nodule Detection with 3D Volume Shape Descriptor*", 2007, IEEE International Conference on Acoustics, Speech and Signal Processing ICASSP '07, vol 1, Honolulu, Hawaii, USA

[23] Z. Ge, B. Sahiner, H. -P. Chan, L. M. Hadjiiski, P. N. Cascade, N. Bogot, E. A. Kazerooni, J. Wei, and C. Zhou, "*Computer-aided detection of lung nodules: False positive reduction using a 3D gradient field method and 3D ellipsoid fitting*", 2005, NCBI, Medical Physics, 32(8), pp 2443–2454

[24] S. Matsumoto, Y. Ohno, H. Yamagata, H. Asahina, K. Komatsu, and K. Sugimura, "*Diminution index: A novel 3D feature for pulmonary nodule detection*", 2005, Proceedings of computer assisted radiology and surgery CARS, Amsterdam, Elsevier, pp 1093–1098

[25] T. Hara, M. Hirose, X. Zhou, H. Fujita, T. Kiryu, R. Yokoyama, and H. Hoshi, "*Nodule detection in 3D chest CT images using 2nd order autocorrelation features*", 2005, Engineering in Medicine and Biology 27th Annual Conference, vol 6, Shanghai, China, pp. 6247–6249

[26] Y. Mekada, T. Kusanagi, Y. Hayase, K. Mori, J. Hasegawa, J. Toriwaki, M. Mori, and H. Natori, "*Detection of small nodules from 3D chest X-ray CT images based on shape features*", 2003, Elsevier, International Congress Series , vol 1256, pp 971–976

[27] J. Debmeshki, M. Valdivieso, M. Roddie, and J. Costello, "*Shape based region growing using derivatives of 3D medical images: application to automatic detection of pulmonary nodules*", 2003, IEEE, 3rd International Symposium on Image and Signal Processing and Analysis, Rome, Italy, vol 2, pp. 1118–1123

[28] S. G. Armato III, M. L. Giger, J. T. Blackburn, K. Doi, and H. MacMahon, "*Three-dimensional approach to lung nodule detection in helical CT*", 1999, Proc. SPIE 3661, Medical Imaging 1999: Image Processing, San Diego, California, USA, pp. 553–559

[29] M. F. McNitt-Gray, S. G. Armato, C. R. Meyer, A. P. Reeves, G. McLennan, R. C. Pais, J. Freymann, M. S. Brown, R. M. Engelmann, P. H. Bland, G. E. Laderach, C. Piker, J. Guo, Z. Towfic, D. P. Qing, D. F. Yankelevitz, D. R. Aberle, E. J. van Beek, H. MacMahon, E. A. Kazerooni, B. Y. Croft, and L. P. Clarke, "*The Lung Image Database Consortium (LIDC) data collection process for nodule detection and annotation*", 2007, NCBI, Academic Radiology, 14(12), pp 1464 – 1474

[30] S. Ashwin, J. Ramesh, S. A. Kumar, and K. Gunavathi, "*Efficient and reliable lung nodule detection using a neural network based computer aided diagnosis system*", 2012, IEEE International Conference on Emerging Trends in Electrical Engineering and Energy Management, Chennai, Tamil Nadu, India, pp. 135–142

[31] Y. Liu, J. Yang, D. Zhao, and J. Liu, "*A method of pulmonary nodule detection utilizing multiple support vector machines*", 2010, IEEE International Conference on Computer Application and System Modeling (ICCASM 2010), Taiyuan, China

[32] T. Messay, R. C. Hardie, and S. K. Rogers, "*A new computationally efficient CAD system for pulmonary nodule detection in CT imagery*", 201, Elsevier, Medical Image Analysis, 14(3), pp 390–406

[33] H. Shao, L. Cao, and Y. Liu, "*A detection approach for solitary pulmonary nodules based on CT images*", 2012, IEEE 2nd International Conference on Computer Science and Network Technology (ICCSNT), Changchun, China, pp 1253–1257

[34] X. Ye, X. Lin, J. Dehmeshki, G. Slabaugh, and G. Beddoe, "*Shape-based computer-aided detection of lung nodules in thoracic CT images*", 2009, IEEE Transactions on Bio-medical Engineering, vol 56 (issue 7) , pp 1810–1820

[35] A. Teramoto, and H. Fujita, "*Fast lung nodule detection in chest CT images using cylindrical nodule-enhancement filter*", 2013, International Journal of Computer Assisted Radiology and Surgery vol 8, pp 193–205

[36] H. Arimura, T. Magome, Y. Yamashita, and D. Yamamoto, "*ComputerAided Diagnosis Systems for Brain Diseases in Magnetic Resonance Images*", 2009, MDPI Algorithms, vol 2(3), pp 925–952

[37] Y. Lee, T. Hara, H. Fujita, S. Itoh, and T. Ishigaki, "*Automated detection of pulmonary nodules in helical CT images based on an improved template-matching technique*", 2001, IEEE Transactions on Medical Imaging, vol 20(7), pp 595–604

[38] Q. Li, and K. Doi, "*New selective nodule enhancement filter and its application for significant improvement of nodule detection on computed tomography*", 2004, Proc. SPIE 5370, Medical Imaging 2004: Image Processing, San Diego, California, USA, pp. 1–9

[39] K. Awai, K. Murao, A. Ozawa, M. Komi, H. Hayakawa, S. Hori, and Y. Nishimura, "*Pulmonary nodules at chest CT: effect of computeraided diagnosis on radiologist detection performance*", 2004, NCBI, Radiology, vol 230(2), pp 347–352

[40] M. Tanino, H. Takizawa, S. Yamamoto, T. Matsumoto, Y. Tateno, and T. Iinuma, "*A detection method of ground glass opacities in chest x-ray CT images using automatic clustering techniques*", Proc. SPIE 5032, Medical Imaging 2003: Image Processing, San Diego, California, USA, pp. 1728–1737

[41] S. G. Armato, M. L. Giger, C. J. Moran, J. T. Blackburn, K. Doi, and H. MacMahon, "*Computerized detection of pulmonary nodules on CT scans*", 1999, NCBI, Radiographics vol 19(5), pp 1303–1311

[42] S. Saita, T. Oda, M. Kubo, Y. Kawata, N. Niki, M. Sasagawa, H. Ohmatsu, R. Kakinuma, M. Kaneko, M. Kusumoto, K. Eguchi, H. Nishiyama, K. Mori, and N. Moriyama, "*Nodule detection algorithm based on multislice CT images for lung cancer screening*", Proc. SPIE 5370, Medical Imaging 2004: Image Processing, San Diego, California, USA, pp. 1083–1090

[43] M. Vijayalaxmi, and H. A. Girijamma, "*Solitary pulmonary nodules classification based on tumor size and volume of nodules*", 2016, 2nd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT), Bangalore, India

[44] A. A. Farag, H. E. Abd El Munim, J. H. Graham, and A. A. Farag, "*A Novel Approach for Lung Nodules Segmentation in Chest CT Using Level Sets*", 2013, IEEE Transactions on Image Processing, vol 22(12)

[45] F. Nery, J. S. Silva, and N. C. Ferreira, "*3D automatic lung segmentation in low-dose CT*", 2012, IEEE 2nd Portuguese Meeting in Bioengineering (ENBENG)

[46] R. Yan-hua, and S. Xi-wen, "*A 3D Segmentation Method of Lung Parenchyma Based on CT Image Sequences*", 2010, International Conference on Information, Networking and Automation (ICINA), Kunming, China

[47] J. Heuberger, A. Geissbuhler, and H. Muller, "*Lung CT segmentation for image retrieval using the Insight Toolkit (ITK)*", 2005, Medical Imaging and Telemedicine (MIT), China

[48] S. Hu, E. A. Hoffman, and J. M. Reinhardt, "*Automatic lung segmentation for accurate quantitation of volumetric X–ray CT images*", 2001, IEEE Transactions on Medical Imaging, vol 20(6), pp 490–498

[49] B. Zhao, G. Gamsu, M. S. Ginsberg, L. Jiang, and L. H. Schwartz, "*Automatic detection of small lung nodules on CT utilizing a local density maximum algorithm*", 2003, Journal of Applied Clinical Medical Physics, vol 4(3), pp 248–260

[50] A. Gupta, O. Martens, Y. Le Moullec, and T. Saar, "*Methods for increased sensitivity and scope in automatic segmentation and detection of lung nodules in CT images*", 2015, IEEE International Symposium on Signal Processing and Information Technology (ISSPIT), Abu Dhabi, United Arab Emirates

[51] J. T. Bushberg, J. A. Seibert, E. M. Leidholdt, and J. M. Boone, "*The Essential Physics of the Medical Imaging*", 3rd ed, 2012, Philadelphia: Lippincott Williams & Wilkins

[52] N. B. Smith and A. Webb, "*Introduction to Medical Imaging Physics, Engineering and Clinical Applications*", 2011, Cambridge University Press

[53] B. Preim, and D. Bartz, "*Visualization in Medicine: Theory, Algorithms, and Applications*", 1st edition, 2007, The Morgan Kaufmann Series in Computer Graphics

[54] O. S. Pianykh, "*Digital Imaging and Communications in Medicine (DICOM): A Practical Introduction and Survival Guide*", 2012, Springer

[55] H. J. Johnson, M. M. McCormick, and L. Ibanez, "*The ITK Software Guide*", 2017, The Insight Software Consortium. Available at: https://itk.org/ItkSoftwareGuide.pdf

[56] Kitware, Inc. "*The VTK User's Guide*", 2010. Available at: https://www.kitware.com/products/books/VTKUsersGuide.pdf

[57] Ubuntu 16.04.2 LTS, Available at: https://www.ubuntu.com/download/desktop/contribute?version=16.04.2&architecture=amd64

[58] Virtual box binary (VirtualBox-5.1.16-113841-Win), Available at: http://download.virtualbox.org/virtualbox/5.1.16/VirtualBox-5.1.16-113841-Win.exe

[59] CMake binary (cmake-3.7.2-Linux-x86_64.sh), Available at: https://cmake.org/files/v3.7/cmake-3.7.2-Linux-x86_64.sh

[60] Qt binary (qt-opensource-linux-x64-5.8.0.run), Available at: http://download.qt.io/official_releases/qt/5.8/5.8.0/qt-opensource-linux-x64-5.8.0.run

[61] VTK 7.1.0 source code, Available at: http://www.vtk.org/files/release/7.1/VTK-7.1.0.tar.gz

[62] ITK 4.11.0 source code, Available at: https://sourceforge.net/projects/itk/files/itk/4.11/InsightToolkit-4.11.0.tar.gz/download

[63] Qt Documentation for Linux, Available at: http://doc.qt.io/qt-5/linux.html

[64] Qt Installation Guide for Ubuntu Linux, Available at: http://wiki.qt.io/Install_Qt_5_on_Ubuntu

[65] VTK Wiki for building on Linux, Available at: http://www.vtk.org/Wiki/VTK/Building/Linux

[66] ITK Wiki for building on Linux, Available at: https://itk.org/Wiki/ITK/Getting_Started/Build/Linux

[67] ITK Wiki for Configuring and Building on Ubuntu, Available at: https://itk.org/Wiki/ITK_Configuring_and_Building_for_Ubuntu_Linux

[68] Building ITK on Windows, Available at: http://blog.chasank.com/building-qt-vtk-and-itk-on-windows-7-x64/

[69] PyQt installation guide, Availabl at: http://pyqt.sourceforge.net/Docs/PyQt5/installation.html

[70] ITK Python Wrapping, Available at: https://itk.org/Wiki/ITK/Python_Wrapping

[71] VTK Python Wrapping, Available at: http://www.vtk.org/Wiki/VTK/Python_Wrapping_FAQ

[72] ITK Configuring and Building on Ubuntu Linux, Available at: https://itk.org/Wiki/ITK_Configuring_and_Building_for_Ubuntu_Linux

[73] VTK Python helloword example, Available at: http://www.vtk.org/Wiki/VTK/Examples/Python/Cylinder

[74] M. Wu, J. Chang, A. A. Chiang, J. Lu, H. Hsu, W. Hsu, and C. Yang, "*Use of quantitative CT to predict postoperative lung function in patients with lung cancer*", 1994, NCBI, Radiology, vol 191, pp 257-262

[75] Qt window multiple inheritance with VTK, Available at: http://www.vtk.org/Wiki/VTK/Examples/Cxx/Qt/RenderWindowUIMultipleInheritance

Appendix 1 – Program header file (class CTDicomLungs3DVis)

CTDicomLungs3DVis.h

```
#ifndef CTDicomLungs3DVis_H
#define CTDicomLungs3DVis_H

#include "ui_CTDicomLungs3DVisUI.h"

#include <QMainWindow>

class CTDicomLungs3DVis : public QMainWindow, private
Ui::CTDicomLungs3DVisUI
{
    Q_OBJECT
public:

    CTDicomLungs3DVis();

public slots:

    virtual void slotExit();

};

#endif
```

Appendix 2 – Program main entry point

CTDicomLungs3DVisMain.cxx

```
#include <QApplication>
#include "CTDicomLungs3DVis.h"

int main(int argc, char** argv)
{

    QApplication app(argc, argv);

    CTDicomLungs3DVis MyCTDicomLungs3DVis;
    MyCTDicomLungs3DVis.show();


    return app.exec();
}
```

Appendix 3 – Program implementation (class CTDicomLungs3DVis)

CTDicomLungs3DVis.cxx

```
#include "CTDicomLungs3DVis.h"
#include "vtkPolyDataMapper.h"
#include "vtkRenderer.h"
#include "vtkRenderWindow.h"
#include "vtkRenderWindowInteractor.h"
#include "vtkSmartPointer.h"
#include "vtkMarchingCubes.h"
#include "vtkActor.h"
#include "vtkProperty.h"
#include "itkImage.h"
#include "itkImageToVTKImageFilter.h"
#include "itkGDCMImageIO.h"
#include "itkGDCMSeriesFileNames.h"
#include "itkImageSeriesReader.h"
#include "gdcmUIDGenerator.h"
#include "itkConnectedThresholdImageFilter.h"
#include "itkCastImageFilter.h"
#include "itkCurvatureFlowImageFilter.h"
#include "itkMaskImageFilter.h"
#include "itkRescaleIntensityImageFilter.h"
#include "itkMaskImageFilter.h"



// Constructor, initialises UI. ITK read and processing. VTK display through
Qt
CTDicomLungs3DVis::CTDicomLungs3DVis()
{
    this->setupUi(this);

    /* Variable ctrlSwitch controls the programs output: takes values 1 to 4

        1 ==> Displays full CT input image
        2 ==> Displays full CT input image without cover
        3 ==> Displays segmented lungs (opaque)
        4 ==> Displays segmented lungs (transparent)

        Default value is 4
    */

    int ctrlSwitch = 4;


    // template types definitions
```

```
    typedef   float           InPixelType;
    typedef signed short    OutPixelType;

    const unsigned int      Dimension = 3;

    typedef itk::Image< InPixelType, Dimension >         InputImageType;
    typedef itk::Image< OutPixelType, Dimension >         OutputImageType;

    typedef itk::CastImageFilter< InputImageType, OutputImageType >
CastingFilterType;
    CastingFilterType::Pointer caster1 = CastingFilterType::New();

    typedef itk::CurvatureFlowImageFilter< InputImageType, InputImageType >
            CurvatureFlowImageFilterType;
    CurvatureFlowImageFilterType::Pointer smoothing =
            CurvatureFlowImageFilterType::New();

    typedef itk::ConnectedThresholdImageFilter< InputImageType,
            InputImageType > ConnectedFilterType;
    ConnectedFilterType::Pointer connectedThreshold1 =
ConnectedFilterType::New();

    typedef itk::ImageSeriesReader< InputImageType >
            ReaderType;
    ReaderType::Pointer reader = ReaderType::New();

    typedef itk::GDCMImageIO
            ImageIOType;
    ImageIOType::Pointer gdcmIO = ImageIOType::New();

    typedef itk::GDCMSeriesFileNames
            InputNamesGeneratorType;
    InputNamesGeneratorType::Pointer inputNames =
InputNamesGeneratorType::New();

    typedef itk::MaskImageFilter< InputImageType, InputImageType >
MaskFilterType1;
    MaskFilterType1::Pointer maskFilter1 = MaskFilterType1::New();

    typedef itk::RescaleIntensityImageFilter< InputImageType, InputImageType
> RescaleFilterType1;
    RescaleFilterType1::Pointer rescaleFilter1 = RescaleFilterType1::New();

    //ITK to VTK connector
    typedef itk::ImageToVTKImageFilter<OutputImageType>       ConnectorType;
    ConnectorType::Pointer connector1 = ConnectorType::New();

    //VTK volume data
    vtkSmartPointer<vtkImageData> volume1 =
            vtkSmartPointer<vtkImageData>::New();
```

```
    //Marching cube surface
    vtkSmartPointer<vtkMarchingCubes> surface1 =
            vtkSmartPointer<vtkMarchingCubes>::New();

    vtkSmartPointer<vtkPolyDataMapper> mapper1 =
            vtkSmartPointer<vtkPolyDataMapper>::New();

    vtkSmartPointer<vtkActor> actor1 =
            vtkSmartPointer<vtkActor>::New();

    vtkSmartPointer<vtkRenderer> renderer =
vtkSmartPointer<vtkRenderer>::New();

    //end of template types definitions



    //ITK read input dicom image
    inputNames->SetInputDirectory( "C:\\MyKitware\\Examples\\ITK-VTK-
Qt\\InsightToolkit-4.10.1__AND__VTK-7.1.0_Qt\\MyItkVtkQ_tsz\\dicom-
images\\no-xml" );

    const ReaderType::FileNamesContainer & filenames =
            inputNames->GetInputFileNames();

    reader->SetImageIO( gdcmIO );
    reader->SetFileNames( filenames );
    reader->Update();


    //START smoothing filter + Connected threshold
    smoothing->SetInput( reader->GetOutput() );
    smoothing->SetNumberOfIterations( 5);
    smoothing->SetTimeStep( 0.01 );

    const InPixelType lowerThreshold1 = -990;
    const InPixelType upperThreshold1 = -250;

    InputImageType::IndexType  index;
    index[0] = 152;
    index[1] = 277;
    index[2] = 152;

    connectedThreshold1->SetInput( smoothing->GetOutput() );
    connectedThreshold1->SetLower(  lowerThreshold1  );
    connectedThreshold1->SetUpper(  upperThreshold1 );
    connectedThreshold1->SetReplaceValue( 255 );
    connectedThreshold1->SetSeed( index );
    connectedThreshold1->Update();

    //END smoothing filter + Connected threshold
```

```
    //BEGIN mask filtering
    maskFilter1->SetInput(reader->GetOutput());
    //maskFilter->SetInput(smoothing->GetOutput());
    maskFilter1->SetMaskImage(connectedThreshold1->GetOutput());
    maskFilter1->Update();

    //END mask filtering


    //BEGIN Rescale intensity filter
    //begin control switch
    switch (ctrlSwitch) {
    case 1:
        rescaleFilter1->SetInput(reader->GetOutput());  // 1 ==> Full CT
input image
        break;

    case 2:
        rescaleFilter1->SetInput(reader->GetOutput());  // 2 ==> Full CT
input image without cover
        break;

    case 3:
        rescaleFilter1->SetInput(maskFilter1->GetOutput()); // 3  ==>
segmented lungs (opaque)
        break;

    case 4:
        rescaleFilter1->SetInput(maskFilter1->GetOutput()); // 4  ==>
segmented lungs (transparent)
        break;

    default:

        break;
    }

    //end control switch

    //convert input to gray level image - rescale intensity to 0 - 255
    rescaleFilter1->SetOutputMinimum(0s);
    rescaleFilter1->SetOutputMaximum(255);
    rescaleFilter1->Update();

    //END Rescale intensity filter


    //input to VTK switches
    //caster1->SetInput( connectedThreshold1->GetOutput() );
```

```
    //caster1->SetInput( smoothing->GetOutput() );
    caster1->SetInput( rescaleFilter1->GetOutput() );


    //ITK to VTK
    connector1->SetInput(caster1->GetOutput());
    connector1->Update();


    //VTK display
    volume1->DeepCopy(connector1->GetOutput());

    surface1->SetInputData(volume1);
    surface1->ComputeNormalsOn();


    switch (ctrlSwitch) {
    case 1:
        surface1->GenerateValues(9, 0, 255);  // 1 ==> Full CT input image
        break;

    case 2:
        surface1->GenerateValues(5, 100, 255);  // 2 ==> Full CT input image
without cover
        break;

    case 3:
        surface1->SetValue(0,180);   // 3  ==> segmented lungs (opaque)
        break;

    case 4:
        surface1->SetValue(0,180);   // 4  ==> segmented lungs (transparent)
        break;

    default:
        break;
    }


    mapper1->SetInputConnection(surface1->GetOutputPort());
    mapper1->ScalarVisibilityOff();


    //set actor 1
    actor1->SetMapper(mapper1);

    switch (ctrlSwitch) {
    case 1:
        actor1->GetProperty()->SetOpacity(1); // 1 ==> Full CT input image
        break;
```

```
    case 2:
        actor1->GetProperty()->SetOpacity(1); // 2 ==> Full CT input image
without cover
        break;

    case 3:
        actor1->GetProperty()->SetOpacity(1); // 3  ==> segmented lungs
(opaque)
        break;

    case 4:
        actor1->GetProperty()->SetOpacity(.2);  // 4  ==> segmented lungs
(transparent)
        break;

    default:
        break;
    }

    actor1->GetProperty()->SetColor(1, .91, .0078);


    //Add actors to renderer
    renderer->AddActor(actor1);


    // VTK/Qt integration
    this->qvtkWidget->GetRenderWindow()->AddRenderer(renderer);

    // Set up Qt action signals and slots
    connect(this->actionExit, SIGNAL(triggered()), this, SLOT(slotExit()));

};

void CTDicomLungs3DVis::slotExit()
{
    qApp->exit();
}
```

Appendix 4 – Qt project generator file (CMake list file)

CMakeList.txt

```
cmake_minimum_required(VERSION 2.8)

if(POLICY CMP0020)
  cmake_policy(SET CMP0020 NEW)
endif()

PROJECT(CTDicomLungs3DVis)

find_package(ITK REQUIRED)
include(${ITK_USE_FILE})

find_package(VTK REQUIRED)
include(${VTK_USE_FILE})

if(${VTK_VERSION} VERSION_GREATER "6" AND VTK_QT_VERSION
VERSION_GREATER "4")
  set(CMAKE_AUTOMOC ON)
  find_package(Qt5Widgets REQUIRED QUIET)
else()
  find_package(Qt4 REQUIRED)
  include(${QT_USE_FILE})
endif()

include_directories(${CMAKE_CURRENT_SOURCE_DIR} ${CMAKE_CURRENT_BINARY_DIR})

file(GLOB UI_FILES *.ui)
file(GLOB QT_WRAP *.h)
file(GLOB CXX_FILES *.cxx)
  qt5_wrap_ui(UISrcs ${UI_FILES} )

  add_executable(CTDicomLungs3DVis MACOSX_BUNDLE
    ${CXX_FILES} ${UISrcs} ${QT_WRAP})
  qt5_use_modules(CTDicomLungs3DVis Core Gui)
  target_link_libraries(CTDicomLungs3DVis ${VTK_LIBRARIES} ${ITK_LIBRARIES})
```

Appendix 5 – Qt UI file

CTDicomLungs3DVisUI.ui, inspired by [73]

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>CTDicomLungs3DVisUI</class>
 <widget class="QMainWindow" name="CTDicomLungs3DVisUI">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>541</width>
    <height>583</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>CTDicomLungs3DVisUI</string>
  </property>
  <widget class="QWidget" name="centralwidget">
   <layout class="QVBoxLayout" name="verticalLayout">
    <item>
     <widget class="QVTKWidget" name="qvtkWidget" native="true"/>
    </item>
   </layout>
  </widget>
  <action name="actionOpenFile">
   <property name="enabled">
    <bool>true</bool>
   </property>
   <property name="text">
    <string>Open File...</string>
   </property>
  </action>
  <action name="actionExit">
   <property name="text">
    <string>Exit</string>
   </property>
  </action>
  <action name="actionPrint">
   <property name="text">
    <string>Print</string>
   </property>
  </action>
  <action name="actionHelp">
   <property name="text">
    <string>Help</string>
```

```
   </property>
  </action>
  <action name="actionSave">
   <property name="text">
    <string>Save</string>
   </property>
  </action>
 </widget>
 <customwidgets>
  <customwidget>
   <class>QVTKWidget</class>
   <extends>QWidget</extends>
   <header>QVTKWidget.h</header>
  </customwidget>
 </customwidgets>
 <resources/>
 <connections/>
</ui>
```

Appendix 6 – Python wrapper sample VTK application

Python wrapper sample VTK hello world application [75]

```
#!/usr/bin/env python
# This simple example shows how to do basic rendering and pipeline
# creation.
import vtk
# The colors module defines various useful colors.
from vtk.util.colors import tomato

# This creates a polygonal cylinder model with eight circumferential
# facets.
cylinder = vtk.vtkCylinderSource()
cylinder.SetResolution(8)

# The mapper is responsible for pushing the geometry into the graphics
# library. It may also do color mapping, if scalars or other
# attributes are defined.
cylinderMapper = vtk.vtkPolyDataMapper()
cylinderMapper.SetInputConnection(cylinder.GetOutputPort())

# The actor is a grouping mechanism: besides the geometry (mapper), it
# also has a property, transformation matrix, and/or texture map.
# Here we set its color and rotate it -22.5 degrees.
cylinderActor = vtk.vtkActor()
cylinderActor.SetMapper(cylinderMapper)
cylinderActor.GetProperty().SetColor(tomato)
cylinderActor.RotateX(30.0)
cylinderActor.RotateY(-45.0)

# Create the graphics structure. The renderer renders into the render
# window. The render window interactor captures mouse events and will
# perform appropriate camera or actor manipulation depending on the
# nature of the events.
ren = vtk.vtkRenderer()
renWin = vtk.vtkRenderWindow()
renWin.AddRenderer(ren)
iren = vtk.vtkRenderWindowInteractor()
iren.SetRenderWindow(renWin)

# Add the actors to the renderer, set the background and size
ren.AddActor(cylinderActor)
ren.SetBackground(0.1, 0.2, 0.4)
renWin.SetSize(200, 200)

# This allows the interactor to initalize itself. It has to be
# called before an event loop.
iren.Initialize()

# We'll zoom in a little by accessing the camera and invoking a "Zoom"
# method on it.
ren.ResetCamera()
ren.GetActiveCamera().Zoom(1.5)
renWin.Render()

# Start the event loop.
iren.Start()
```