การตัดแยกและการประมาณเนื้อร้ายเริ่มต้นในปอดโดยการใช้ค่าขีดแบ่งบนความหนาเฉพาะที่

นายอัครพันธุ์ เจนธีรพงศ์

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตร์มหาบัณฑิต
สาขาวิชาวิทยาการคอมพิวเตอร์และสารสนเทศ ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์
คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2555

LUNG NODULE SEGMENTATION AND ESTIMATION USING THRESHOLDING ON LOCAL THICKNESS

Mr. Akaraphan Janetheerapong

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science Program in Computer Science and Information Technology
Department of Mathematics and Computer Science
Faculty of Science
Chulalongkorn University
Academic Year 2012

Thesis Title	Lung Nodule Segmentation And Estimation Using Thresholding on Local Thickness
By	Mr. Akaraphan Janetheerapong
Field of Study	Computer Science and Information
Thesis Advisor	Assistant Professor Rajalida Lipikorn, Ph.D.
Thesis Co-advisor	Assistant Professor Nagul Cooharojananone, Ph.D.

Accepted by the Faculty of Science, Chulalongkorn University in Partial Fulfillment of the Requirements for the Master's Degree

……………………………………………….. Dean of the Faculty of Science
(Professor Supot Hannongbua, Dr. rer. nat.)

THESIS COMMITTEE

………………………………………………… Chairman
(Assistant Professor Pattarasinee Bhattarakosol, Ph.D.)

……………………………………………….... Thesis Advisor
(Assistant Professor Rajalida Lipikorn, Ph.D.)

………………………………………………… Thesis Co-advisor
(Assistant Professor Nagul Cooharojananone, Ph.D.)

………………………………………………… External Examiner
(Colonel Anan Wattanathum, Ph.D.)

อัครพันธุ์ เจนธีรพงศ์ : การตัดแยกและการประมาณเนื้อร้ายเริ่มต้นในปอดโดยการใช้ค่าขีดแบ่งบนความหนาเฉพาะที่. (LUNG NODULE SEGMENTATION AND ESTIMATION USING THRESHOLDING ON LOCAL THICKNESS) อ. ที่ปรึกษาวิทยานิพนธ์หลัก : ผศ.ดร. รัชลิดา ลิปิกรณ์,อ. ที่ปรึกษาวิทยานิพนธ์ร่วม : ผศ. ดร. นกุล คูหะโรจนานนท์, จำนวนหน้า 63 หน้า.

เสนอวิธีการที่ง่ายแต่ได้ผลสำหรับการตัดแยกและการประมาณเนื้อร้ายเริ่มต้นในปอดโดยการใช้ค่าเทรสโฮลด์บนความหนาเฉพาะที่ โดยการใช้ขั้นตอนวิธีประมวณความหนาเฉพาะที่บนจุดที่เราตีกรอบบนภาพรังสีส่วนตัดอาศัยคอมพิวเตอร์ ระบบที่นำเสนอสามารถเอาความหนาของบริเวณเฉพาะมาประมวลต่อ โดยใช้วิธีการคำนวณค่าขีดแบ่งแบบอัตโนมัตบนความหนาเฉพาะที่ ระบบจะสามารถกำจัดบริเวณ 3 มิติที่มีความหนาต่ำ อาทิ เส้นเลือด และในขณะเดียวกัน ระบบสามารถที่จะถนอมเก็บบริเวณ 3 มิติที่มีความหนาสูงกว่า เช่น เนื้อร้ายเริ่มต้น เป็นต้น การใช้ขั้นตอนวิธีการนี้สามารถทำให้ประมาณเนื้อร้ายเริ่มต้นได้ เพราะส่วนที่เป็นหนามและเส้นเลือดที่ติดกับเนื้อร้ายเริ่มต้นถูกกำจัดออก เพื่อที่จะประมวณว่า วิธีที่เรานำเสนอใช้ได้ผลและวัดได้จริง ผู้ทำวิจัยได้ทำการวัดขนาดกลุ่มข้อมูลเนื้อร้ายเริ่มต้นโดยการเปรียบเทียบระหว่าง โปรแกรม Philips Extended Brilliance Workspace กับวิธีการที่นำเสนอ และเห็นว่ามีความสัมพันธ์กัน 98.9%

ภาควิชา คณิตศาสตร์และวิทยาการคอมพิวเตอร์ ลายมือชื่อนิสิต...
สาขาวิชาวิทยาการคอมพิวเตอร์และสารสนเทศ ลายมือชื่อ อ.ที่ปรึกษาวิทยานิพนธ์หลัก........
ปีการศึกษา 2555 ลายมือชื่อ อ.ที่ปรึกษาวิทยานิพนธ์ร่วม........

5373615723 : MAJOR COMPUTER SCIENCE AND INFORMATION

KEYWORDS : LUNG NODULE SEGMENTATION / LOCAL THICKNESS / THRESHOLDING

AKARAPHAN JANETHEERAPONG : THREE DIMENSIONAL LUNG NODULE SEGMENTATION AND ESTIMATION USING THRESHOLDING ON LOCAL THICKNESS. ADVISOR : ASST. PROF. RAJALIDA LIPIKORN, Ph.D., CO-ADVISOR : ASST. PROF. NAGUL COOHAROJANANONE, Ph.D., 63 pp.

A simple-yet-effective method, utilizing local thickness and auto-thresholding for manual lung nodule segmentation and estimation is presented. By utilizing local thickness algorithm, we can achieve the field of local thickness for the respective region of interest, which can be used for further assessment in thresholding. The auto-thresholding technique, IsoData, is used on local thickness field for segmenting lung nodule, which can remove blood vessels or features that are difficult to be removed by conventional mean and must be excluded in order to ensure correct segmentation and measurement. By using correlation comparison on the same lung nodule dataset between our proposed method and Philips Extended Brilliance Workspace, we can see that the correlation is 98.9%.

Department : Mathematics and Computer	Student's Signature ....
Field of Study : Computer Science	Advisor's Signature ....
Academic Year : 2012	Co-advisor's Signature ....

Acknowledgements

I would like to thank my advisor, Assistant Professor Rajalida Lipikorn, and Assistant Professor Nagul Cooharojananone at the department of Mathematics in Chulalongkorn University. They dedicated themselves in assisting my research greatly. When I do not have the resource or knowledge, they are always there to assist and provide respective subjects that I lack. Without them, creating such a thesis would be next to impossible.

Moreover, I would like to thank Col. Dr. Anan Wattanathum for providing the data set for this research. Without such resources, this research would not be possible, as we will not have anything to compare ourselves with at all. Therefore, I highly appreciated the generosity.

All in all, I would like to once again thank you honorable people mentioned above, for assisting me in this research.

Table of Contents

Chapter Page

List of Tables

List of Figures

CHAPTER I

Introduction

Biomedical research is always a field of interest to many researchers. The progress in our research can increase life expectancy, survivability, health condition, etc. And as technologies progress further, we have better tools and techniques at our disposal to further increase effectiveness of healthcare. For example, in the past, without proper tools for inspecting inside the patient's body, it is impossible to diagnose a disease, such as internal cancer. But with technologies such as X-ray, Ultrasound, or CT scan, it is possible to inspect inside the patient and diagnose the disease more efficiently.

For example, in the case of Lung Cancer, if it is not detected at an early stage, it could spread to other organs and cause havoc damages, which eventually lead to death of a patient. However, using CT Scans, we can detect these lung cancers at their young states and deal with them before things get out of hand. According to [1], the chance of survival for patients, with the beginning stages of malignant nodules, may be as high as 80%, but for patients with advanced stages of malignant nodules, the survival rate is as low as 5% or less. Therefore, if we can detect these lung nodules during their early stages, we can perform medical operation to remove these nodules. Thus, we can ensure higher survivability of the patients.

However, there are some cases that are more difficult than other cases. Manual nodule detection requires the radiologists to skim manually through hundreds of slides. This is a time consuming and tedious task and it is prone to error. Without a doubt, nodule detection in a larger size is easy to spot, but this already implies that the cancer in patient might be in a more advanced stage. If we can detect such nodules when they are at earlier stages, we can eliminate the threat before things get out of hand. Nevertheless, nodules at such stages are very small in diameter (Less than three millimeters). These nodules may present themselves only in a few CT slices out of hundreds and occupy only a few voxels. These are very difficult to manually detect by human.

Many automated approaches [2-5] are suggested, and there are still false positives and false negatives within the automated results. Note that we are not looking for 100% perfect solution here. Automated solutions could suggest a radiologist the locations of a number of small nodules, but in the end, a radiologist will have to ensure that the nodule is valid and not false positive. Moreover, he will still have to skim through the slices for checking nodules that might have been missed by the automated solution. All in all, the CT scan still has to be thoroughly and intensively checked to ensure the right medical procedure.

Another issue that has to be considered is segmentation. Several methods such as in [3] suggest calculating shape index from the isotropic volume, for detecting spherical shapes. Although many nodules are spherical, not all nodules are spherical. Thus, undersegmentation or oversegmentation might occur in this case.

1.1 Objectives

We would like to propose a simple-yet-effective and user-friendly method for manual lung nodule segmentation, which allows non-spherical nodule to be taken into account to reduce undersegmentation, as well as to remove blood vessels that are attached to the nodule, to reduce oversegmentation.

1.2 Scope of the Work

- We will use only DICOM of Philips and Toshiba software.
- We will use Fiji software for this purpose.

1.3 Problem Formulation

In this research, we required CT Scans of patients with lung nodules, however, there are many types of lung nodules. For instance, juxtapluera nodules are nodules that stick to the lung wall. Nodules themselves, most of the time, tend to be spherical, but not all nodules are spherical. A few nodules have irregular shapes or have features spreading all over the place. One paper [13] suggested 4 types of nodules. However, another paper [14] has shown more complex shaped nodules. Their shapes scatter around the lung in a complex manner; one of them looks like a spike ball. In older researches, their algorithms are based on the assumption that nodules tend to be spherical. This tends to be true for most of the time, but if the nodule is not spherical at all, then the assumption would be false in the respective case. That is their limitation. Theoretically, the more nodules we can investigate, the more we can deal with. We believed that segmentation of many typical spherical nodules have been taken care of in past research already. What remains should be these special cases, and we believed our algorithm should be able to take care of these nodules.

Another issue to be addressed is, in many commercial software, should the radiologist feels the software segments the nodule incorrectly, he can adjust the lasso selection on each frame accordingly. If a nodule spans on for 100 frames, this can be a very tedious task and each different person performing this same task would achieve different result, due to human bias.

Therefore, we aim to address the following issues:

- How to segment non-spherical nodules in such a way that attached blood vessels and thorns could be removed.
- Coming up with a method that allows better usability to select a nodule.

1.4 Expected Outcomes

- A user-friendly method that can solve aforementioned problems
- Results from the method should be similar to those of Philips, which we hold as a golden standard.

CHAPTER II

Theoretical Background

This chapter elaborates some technical details that are mandatory to understand later parts of this thesis.

2.1 Image Processing

Image Processing is a technique in Computer Science to manipulate or process images to achieve specific objectives. It is a huge field and is still considered as an active ongoing research area.

Briefly explaining, an image in a computer is nothing more than a sequence of numbers. Depending on the format of the image type, the sequence of numbers of image type could be different from one to another. However, in this thesis, we are using DICOM files. In these DICOM files, we have grayscale image data. Suppose a 2D image has a resolution of 512 x 512 pixels, we can treat such image to be a 512 width x 512 height matrix. Each element in the matrix is a pixel for the image. For 8 bit image, a pixel can be a number from 0 (black) to 255 (white). For other variations, the value could be 0-65535, 0.0-1.0, etc. This all depends on formatting definition, but all in all, these values signify the blackest value to the whitest value. Higher dimensional images from 3D, 4D, 5D and so on, also exist. For this thesis, since we are using CT Scan and DICOM files, we will be using 3D images.

2.2 Biomedical Image

Biomedical images in this thesis refer to DICOM (Digital Imaging and Communications in Medicine) which is a standard for handling, transmitting, printing, and storing information in medical imaging. The standard is widely used in many commercial software, as the file formatting can be shared and understood not by just one software, but many other software, as the standard enforces that the respective biomedical images should be similar in the file format. This is why DICOM files could be opened by a multitude of various imaging programs. Nevertheless, DICOM standard is loose. Should developers of any entity feel additional header information of DICOM files should be added or stored elsewhere differently for any reason, it is technically possible to do so. Yet, doing so may result in fewer programs parsing the file incorrectly. Some software developers may prefer that certain information must be found in certain places. While this assumption is true for most of the time, if some entities decided that certain mandatory values are to be stored elsewhere instead of the usual places, a program other than these entities' own software may incorrectly parse the file and show incorrect result. Therefore, when handling different DICOM files from software from many different companies, it is wise to understand this tiny bit difference between their implementation. However, we will not discuss the difference in their implementation here, as it is not our objective.

2.3 Methods used in the past

We have come across several methods. S. G. Armato III et al [2] has suggested for analysis for nine features of nodules. M. Antonelli et al [3] has suggested for distinguishing nodules from blood vessels and other pulmonary elements via shape analysis. Y. Liu et al [4] has suggested using support vector machine to perform voxel analysis. From [2] and [3], we can see that they have been trying to segment the nodules mathematically. From [4], a supervised learning model is used.

Generally, for automated segmentation, the following procedure needs to be done:

- Preprocessing: on the CT Scan, the background needs to be removed. Background refers to the field outside the patient’s body. This implies the bed of the CT Scan machine or any other objects are to be removed.
- Lung Field segmentation: the flesh surrounding the lung is to be removed, leaving only the lung field intact.
- Error Correction From Lung Field Segmentation: many papers use of Rolling Ball algorithm to reinsert the attached to the wall juxtaplueral nodules. These nodules have the same intensity levels as the lung wall. Thus, when lung wall is removed, these nodules are removed along as well. Therefore, many papers have suggested to reinsert them back using Rolling Ball algorithm. Imagine in each frame of the CT slice, a ball is rolled around each lung field, and when there is a hole in the shape, the hole is filled (to reinsert the possible nodule candidate). The benefit is, the nodules are inserted back. However, the disadvantage with this method is that the nodule could be any size. It could be so big that it leaves a big hole after lung wall removal that the ball rolled pass through the section, as if there is nothing there. Shapes of the lung are also varied from person to person, thus, it is inaccurate to utilize curvy shape of the lung.

These are the general steps for automated system from many papers. After that, suggested methods of each respective paper are applied afterward.

How much “area” should be considered a nodule and how much “area” should be considered non-malignant tissues? The only way to know which nodule is malignant tissue and which is not requires the patient’s tissue to be removed and inspected in the laboratory. There has been a statistical research concerning a method to distinguish malignant and non-malignant nodules mentioned in [1]. It won’t be helpful to have more accurate result, as studying the patient’s nodule physically is the only way to have the best call on malignancy. Therefore, we could make do with estimation. If the size is so huge, then the doctor should just perform the operation on the patient. If the size is not so huge, then the doctor could estimate it first, and then perform the next procedure with respect to the method given in [1]. This is one more disadvantage of the automated method, that is, in the end, we cannot have a clear say whether a nodule is malignant or not. Nevertheless, automated methods are to be used for detection, as skimming through CT slices for very small nodules are difficult and tedious. The benefit of the automated method should be used for detection, yet it is preferable to have another skim through on the CT slices for false positives, false negatives and what not.

2.4 Commercial software

In this research, we use Philips’s software [5] as the golden standard for correlation test. Additionally, we also use Toshiba’s software [16] for comparison with our software as well. Both Philips and Toshiba have its own one-click segmentation methods, which they will try to measure the size of the specified nodule automatically. Should the user feel that there is a need to adjust the automated lasso selection, these software allow the user to do so frame by frame.

CHAPTER III

Proposed Method and Algorithmic Design and implementation

We will go further in-depth with the proposed method and algorithmic design in this chapter. Our method is semi-automatic, which means that a user-interaction is required. We required 4 interactions in our method, which will be covered later.

3.1 Computed Tomography Image

First, the image is to be loaded. Normally, this should be CT scanned image, which usually comes as DICOM format. As mentioned before, there are several implementations of DICOM files, and they might need to be taken care of differently. We require two things from DICOM file(s): image data and some parameters.

For image data, usually DICOM format header has a specific place for storing such image data. Several imaging software, such as IrfanView , could open a few type of DICOM files. However, as mentioned before, some DICOM implementation is different from the norm, and some regular software couldn’t open it due to different implementation. We will not cover how to open different type of DICOM files, and the bottomline is all the image data for a target patient is to be acquired and loaded into the RAM sequentially, meaning we should have the first slice to the last slice ordered correctly.

For the mandatory parameters, we will require the data from the DICOM header file. An example of a partial part of a DICOM file is shown on the next page:

```
0002,0002  Media Storage SOP Class UID: 1.2.840.10008.5.1.4.1.1.2

0002,0003  Media Storage SOP Inst UID:
1.2.392.200036.9116.2.5.1.37.2417493711.1343795704.895026

0002,0010  Transfer Syntax UID: 1.2.840.10008.1.2.1

0002,0012  Implementation Class UID: 1.2.392.200036.9116.2.6.1.100

0002,0013  Implementation Version Name: TM_CT_CMW_V3.00

0008,0005  Specific Character Set: ISO_IR 100

0008,0008  Image Type: ORIGINAL\PRIMARY\AXIAL

0008,0016  SOP Class UID: 1.2.840.10008.5.1.4.1.1.2

0008,0018  SOP Instance UID: 1.2.392.200036.9116.2.5.1.37.2417493711.1343795704.895026

0008,0020  Study Date: 20120801

0008,0021  Series Date: 20120801

0008,0022  Acquisition Date: 20120801

0008,0023  Image Date: 20120801

0008,0030  Study Time: 133047.000

0008,0031  Series Time: 133340.379

0008,0032  Acquisition Time: 133403.250

0008,0033  Image Time: 133405.193
```

As you can see, in the header of DICOM, there are numbers in front of each information. These numbers and the information could be studied in the DICOM standard website.

For the required parameters, we required “distance between slice” and “pixel spacing”, which is available in the following header:

```
0018,0088 distance between slice

0028,0030 pixel spacing
```

Again, due to different implementation of DICOM, sometimes, such header may not exist, or worse, it may exist, but the value is left as a default unusable value. One of the DICOM file types that we have encountered, have used the following original header:

```
7005,1013  ---: 1

7005,1016  ---: 1.2.392.200036.9116.2.5.1.37.2417493711.1343795678.307076

7005,1017  ---: 408

7005,1018  ---: 340

7005,101A  ---: 1

7005,101B  ---: FC07

7005,101F  ---: 20120801133438305675
```

```
7005,1022  ---: 0.80

7005,1023  ---: 1.388

7005,1024  ---: 20120411

7005,1030  ---: CT

7005,1040  ---: 234.0

7005,1041  ---: 0.5x80

7005,1043  ---: 0.00000\0.00000\-1.00000

7005,1063  ---: 5.9
```

As you can see, there are only numbers and no human readable information is given. Some freeware DICOM reader might be able to interpret these values, fortunately, and it is a must to do a bit of homework to figure which header is our required parameter. Again, we do not know how many implementation are out there, so we can only share some of the few we have encountered. But if the file is DICOM, the required values should be there somewhere in the header.

Once we have those parameters, we will be able to translate a voxel into a real world millimeter cube.

3.2 Preprocessing

Before we will perform any operation on the image, we should perform a bilateral filtering [6] to smooth the image and at the same time to preserve the edges in an image. This will allow us to have a “cleaner” input.

3.3 Human interaction

The user is required to give the program 4 things:

- The first slice of the nodule

- The last slice of the nodule

- The rough ellipse-shaped location of the nodule

- The pinpoint location of the nodule

The procedure may seem redundant at first, but in order to ensure that the region of interest is really what we are looking for, those four procedures above are mandatory.

Here are some explanations for each parameter above respectively:

1-2. We want only the slices that contain the lung nodule; therefore we get the respective substack here.

3. The ellipse is an area of interest which will be used for cropping on all slices. The result is cylindrical area crop on the remaining CT slice. Of course, it is possible to use other 2D shapes, which extend into 3D. The reason that we use ellipse selection is for a better ease of use, since nodule’s shape could be spherical, egg-shaped, or spiky urchin-shaped. These nodules

nonetheless will stay in one region of interest, and using ellipsical shaped which extend into cyrindrical shaped should provide a reasonable ease of use for the user. We will only segment one nodule at a time. If there are two nodules extending and connecting to each other, we will take of them one by one.

4. The fourth step may seem redundant, but for some rare cases such as nodule that glides along blood vessels, this will help us remove all the other blood vessels and possibly other lung lesions remaining from step 2, and retain only this single nodule of interest. The objects that will be removed are those that are not attached to the nodule. Other attached objects that are attached to the nodule will be taken care of later.

3.4 Segmentation

The segmentation step is as follow:

3.4.1 Auto-thresholding

Auto-thresholding will be performed on an image utilizing IsoData auto-thresholding technique [7]. This will binarize an image, which allows a more accurate result in the region growing section.

3.4.2 Region Growing

Region Growing algorithm will be performed starting from the specified point, using the human interaction from step 3.3. Any other objects outside the acquired region are to be removed. What remains within the user-specified ellipse should now have only nodule of interest and remnants of other attached objects of the nodule.

3.4.3 Local Thickness

There are more tasks needed to be done before volume estimation. For instance, a nodule may have regular blood vessels attached to it. These vessels are not a part of a nodule and must be excluded in volume estimation. Also, as a nodule gets bigger, it tends to develop extruding thorn-like or bump-like features, which, according to our radiologist, should be excluded from volume estimation as well. Local Thickness [5] is used to solve this issue, and is originally defined as:

$$\tau(\underline{p})=2\cdot\max(\{\mathrm{r} \mid \underline{p} \in \mathrm{sph}(\underline{x},\mathrm{r}) \subseteq \Omega,\ \underline{x} \in \Omega\}) \tag{1}$$

Local Thickness allows us to differentiate between areas that are thicker and less thick by giving these area different values of intensity with respect to "local thickness". In order to validate and clarify Local Thickness, we showed example cases from [9]:

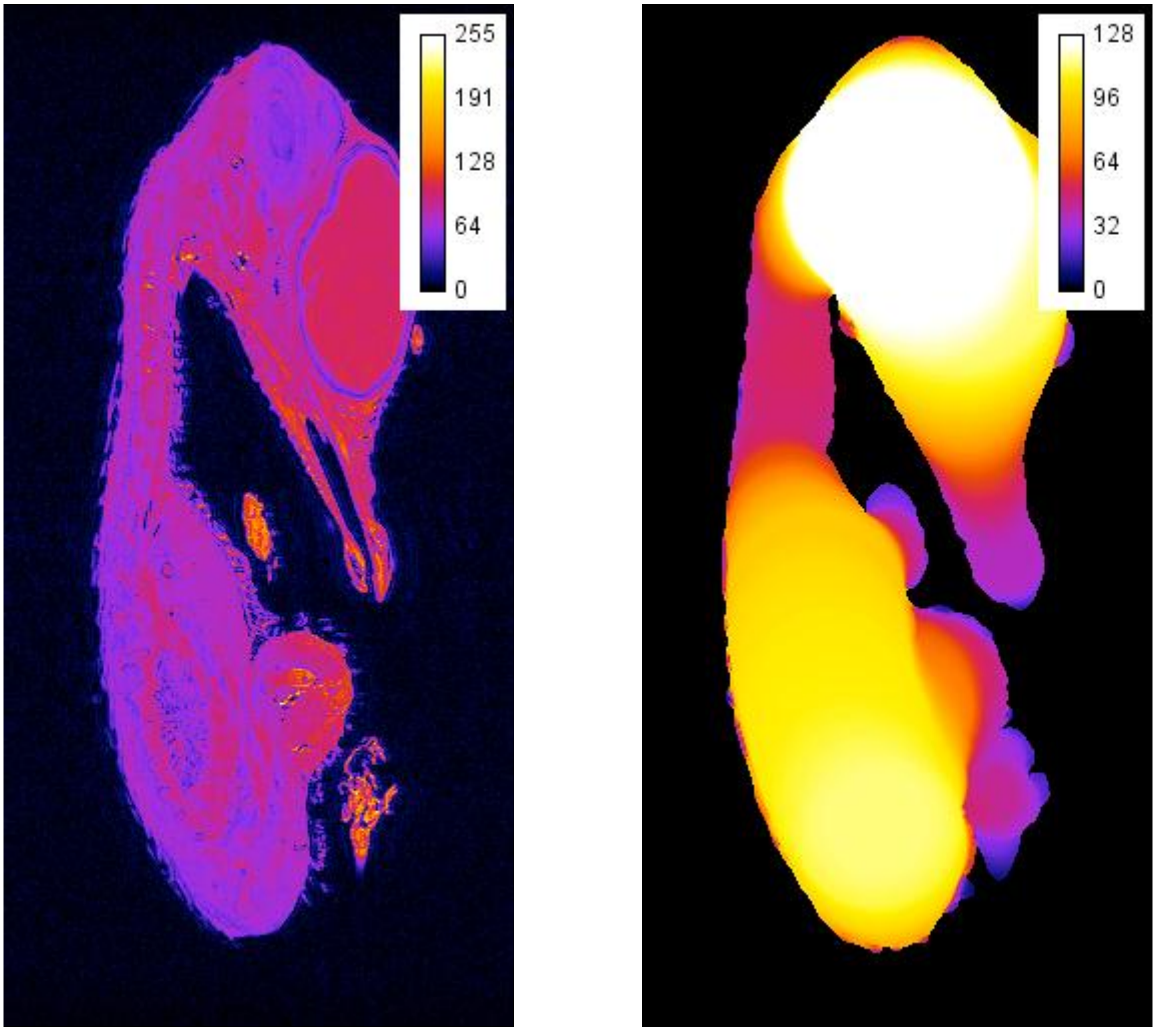


Figure 1: “Local Thickness” algorithm on MRI Quail Embryo. An example image of MRI Quail Embryo on the left that has “local thickness” algorithm applied on the right.

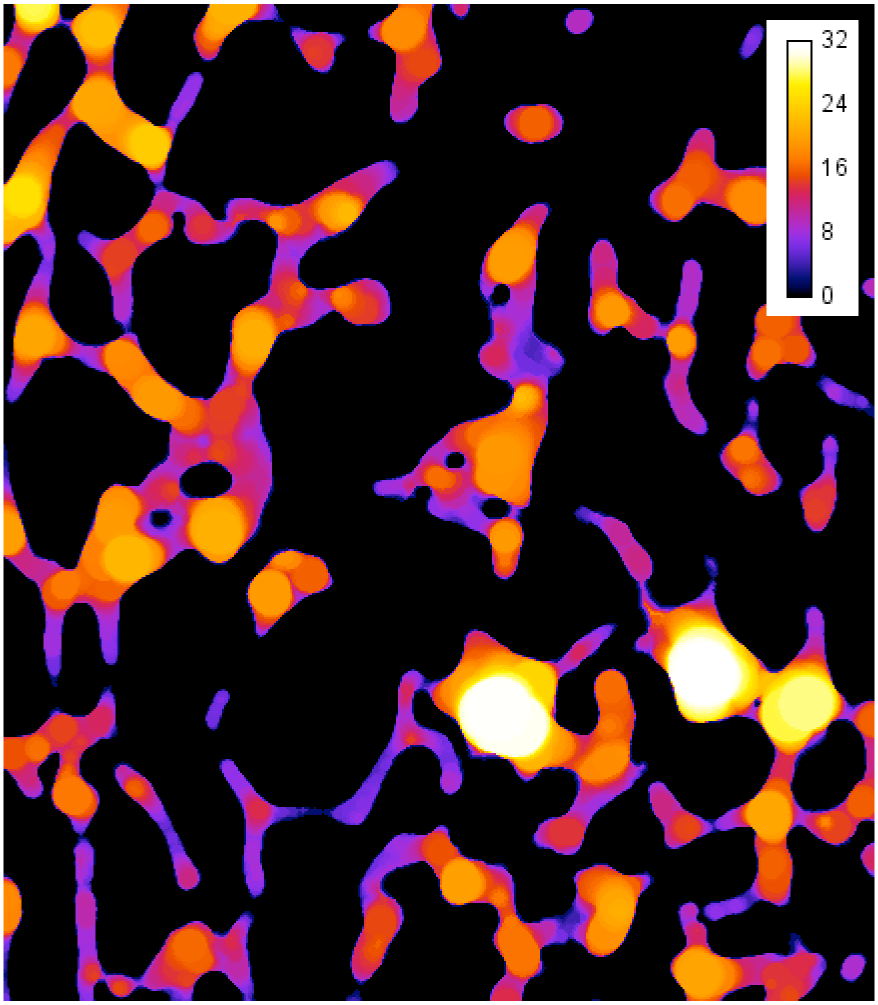


Figure 2: “Local Thickness” algorithm applied on Trabecular Rabbit Femur. We can see the femur’s structure’s thickness with “local thickness” algorithm.

Fig. 1 and Fig. 2 show how the Local Thickness algorithm behaves given the CT slices of several studies. The application proves useful to get local thickness information from three dimension data. Therefore, we would like to apply this method on to lung nodule segmentation to achieve the same goal.

Elaborating Local Thickness formula, from Fig. 3 and (1), let $\tau(\underline{p})$ be the Local Thickness function. Let Ω be a structure or object of interest, in which $\Omega \subset \mathbb{R}^3$, $\underline{x}$ be the center and r be the radius of a local largest sphere $\mathrm{sph}(\underline{x},\mathrm{r})$. Any particular point $\underline{p}$ will have local thickness value that is equal to the ratio size of all spheres of the local largest sphere, which have been fit into the object in an image, and is a superset of point $\underline{p}$.

$\tau(\underline{p})$ takes point $\underline{p}$ as an input and finds the local thickness intensity of this point $\underline{p}$. The local thickness intensity could be defined as the value of the diameter of the largest sphere that fits inside the given input object at the point $\underline{p}$. From here, we iteratively take each voxel of Ω, and process all of their local thickness intensity to get local thickness field.

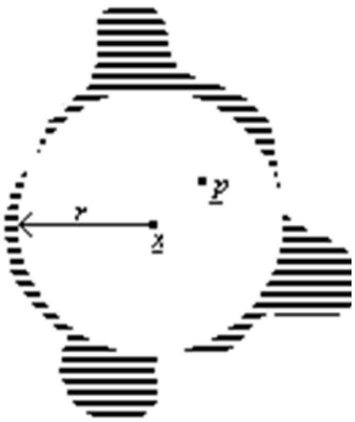


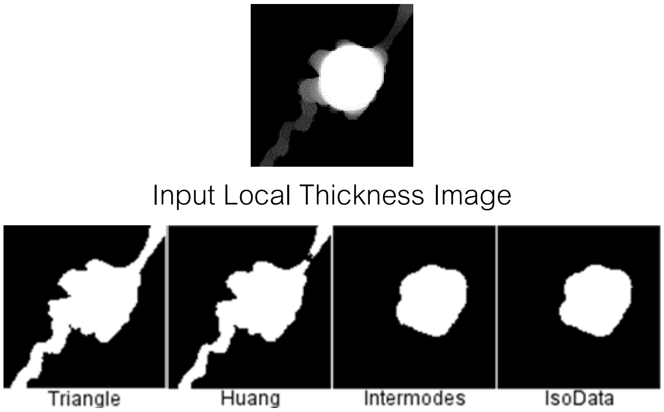
Figure 3: Local Thickness on the striped structure Ω. Only one sphere is shown here for simplicity. In the real definition, a sphere will exist in every voxel of Ω.

We will use an example naïve implementation of local thickness acquisition for the sake of simpler explanation. Let us think of local thickness like this: first, an input image IMG will be an image which contains a number of objects. An object in this case refers to a group of white voxels which are connected together in an image. All of the white voxels in the image will have a sphere created with diameter of 1. After that, all of the spheres will increment their diameters in each iteration. Any sphere will stop expanding if the next incremental diameter contains a black voxel. After all of the spheres have stopped expanding, an empty matrix LT with the same dimension as IMG is created. LT will be used for holding local thickness intensity. For each voxel v of LT, iterate through all of the spheres and check which sphere contains the voxel v and which sphere has the largest diameter value. This voxel v will hold the value of the sphere which has the highest diameter in this case. After that, the highest intensity value from LT will be stored as MAXINT. All of the voxels will then be divided by MAXINT and multiply with 255. All of the voxels in LT will now hold the desired result, a field of local thickness intensity, where 255 is the maximum intensity.

Several implementations for local thickness exist to improve the algorithm's performance as the original definition of local thickness is too slow for any application. We used R. P. Dougherty's implementation [9] as our base and add an extra step to the algorithm in the next step.

3.4.4 IsoData Thresholding on Local Thickness

Usually, a nodule has bigger volume than blood vessels and is expected to be the brightest feature in this user-specified region image. Other parts such as blood vessels should have lower intensity. In order to eliminate these features, we add an extra step into Local Thickness algorithm, which is to perform an auto-thresholding technique. We have tried a few auto-threshold techniques [7,10-12] (Fig. 4) and we see that the IsoData is the most stable for our method. (shown in Fig. 5 on top right) This way, all of the lower intensity features will be removed, and what remains should be the nodule of interest. (shown in Fig. 5 on bottom left)



Input Local Thickness Image

Figure 4: Comparison on a few auto-thresholding techniques

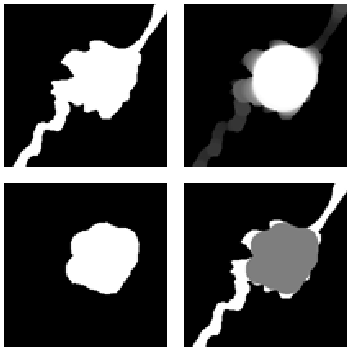


Figure 5 Nodule segmentation simulation: clockwise, starting from top left: the original binarized image, local thickness, auto-threshold on local thickness, and finalized segmentation.

3.5 Estimation

After thresholding on local thickness has been performed, we can now estimate the nodule simply by counting the remaining voxels in the process.

3.6 Conversion to millimeter cube

To convert, simply multiply the number of counted voxels with the 3 mandatory values mentioned earlier in step 3.1:

(pixelSpacingWidth * pixelSpacingHeight * distBetweenSlice) * totalCountedVoxel)

This will convert voxel to millimeter cube.

3.7 Program Design and Flowchart

Since we used Fiji as our base, we have to write a script and execute it from Fiji's interface. Fiji can execute Python, Java, etc. We decided to use Java as Fiji is based on ImageJ, which is written in Java. Because all the primary classes are already provided by Fiji, we simply built on top of Fiji. Fiji can run and edit these external routine in its editor.

There are templates provided within Fiji itself, however, we will not use their template; we create our own class and methods. The first method that will run when we hit "Run" on Fiji editor is the constructor of our class. Our class extends PlugInFrame of Fiji, therefore, it is a frame, and we can make it appear by adding it to the WindowManager.

We have the following essential methods in our class:

```
public masterNoduleSegmentFiji()

public void mouseClicked(MouseEvent e)

public void actionPerformed(ActionEvent e)

private void segmentNodule()

private Rectangle extraCrop(Rectangle bound)
```

The explanation for these methods are as follow:

`public masterNoduleSegmentFiji()` is the constructor of the class and is the main entrance for our script. First, it performs a check whether the user has his (DICOM image) file open or not. If so, the program proceeds by checking for the header data in the opened DICOM file. Differences in header data has been discussed in section 3.1. These data are crucial in that they are used for the correct calculation of estimated size. We need information of slice distance, etc, so we obtained them here. Furthermore, we will also register (add) “this” frame to the windowManager, making it visible on the screen. Various buttons and labels are also initialized and added to “this” frame, for displaying user interface. Listeners are also added.

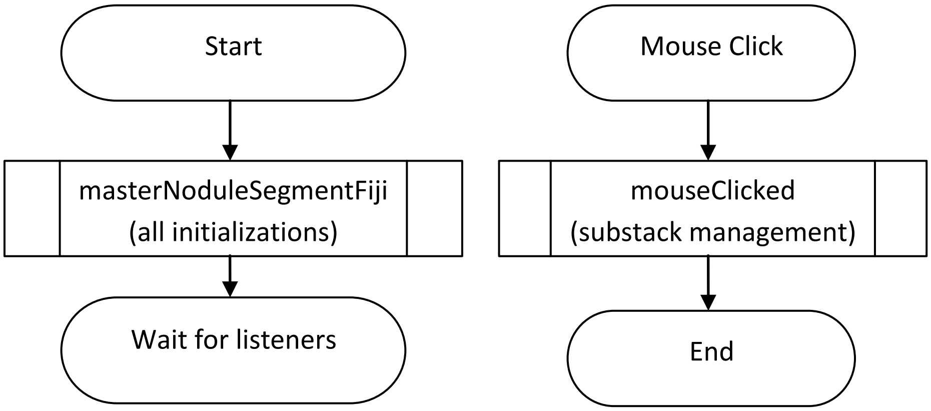
`public void mouseClicked(MouseEvent e)` contains the code for manipulating and checking the user’s input during the slice selection phase. When the user clicks on a slice, the program will remember the slice’s index as the “first” index. After that, the program will request for the “last” index, which the user will also supply. The program will remember the user’s next selected slice as the “last” index. Then, the program will proceed on to the next phase. In fact, this method is the key component during this substack selection, in which the program will use the “first” and “last” information for getting the substack.

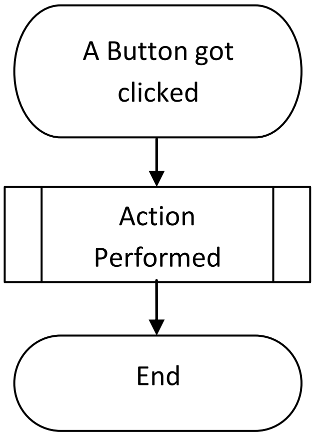
`public void actionPerformed(ActionEvent e)` is used for receiving “button click” from various buttons. The buttons that will invoke this method are the “start” button and the “cancel” button. The “start” button is used during ellipse selection and the final pinpoint selection, while “cancel” button basically closes down the frame and exit the script. The code regarding the later part of the algorithm such as bilateral filtering and local thickness are all located and/or originated from this method.

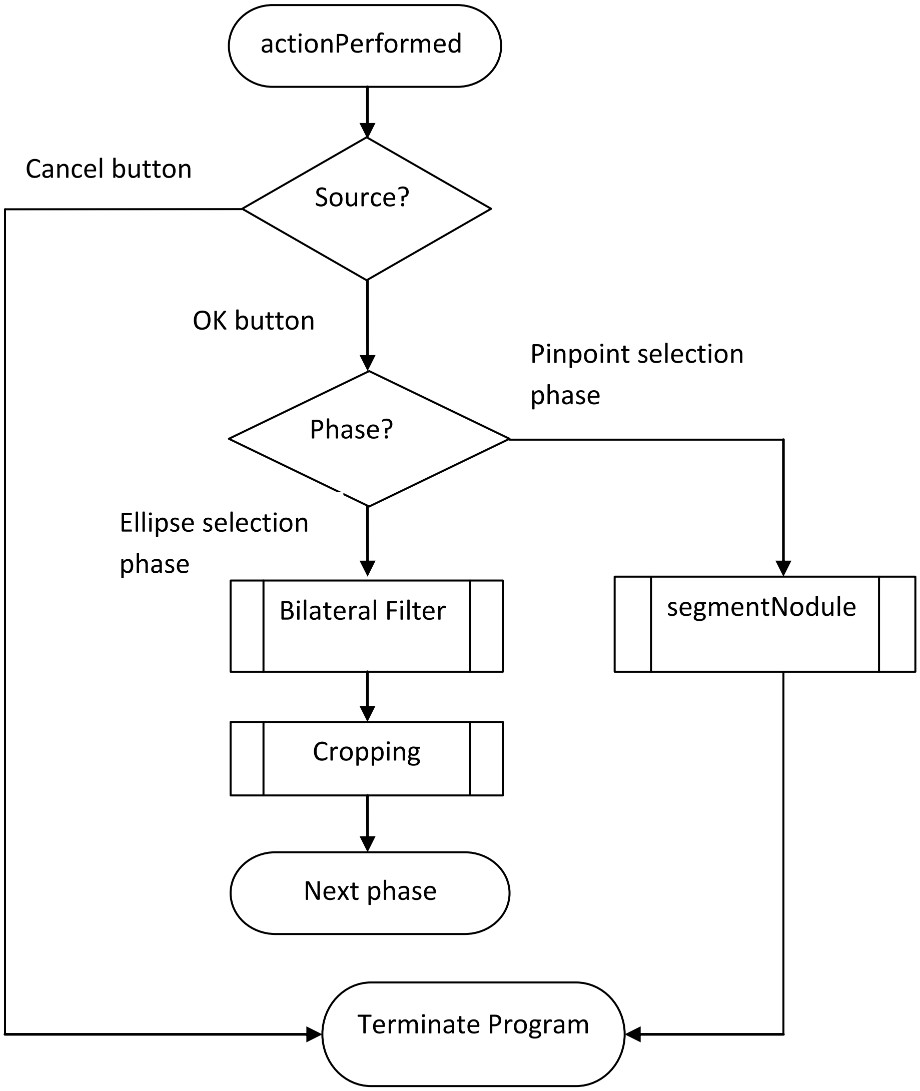
`private void segmentNodule()` is invoked near the end of actionPerformed method during the final pinpoint selection phase. As its name suggested, the method “segments” nodule after all the major algorithms have been applied. Up to this point, we should have all the required values to segment the nodule. At this point, the user just has to wait for all the remaining algorithms such as Level Sets and Local Thickness to finish. Once done, the program will report the estimated size of the specified nodule in the console.

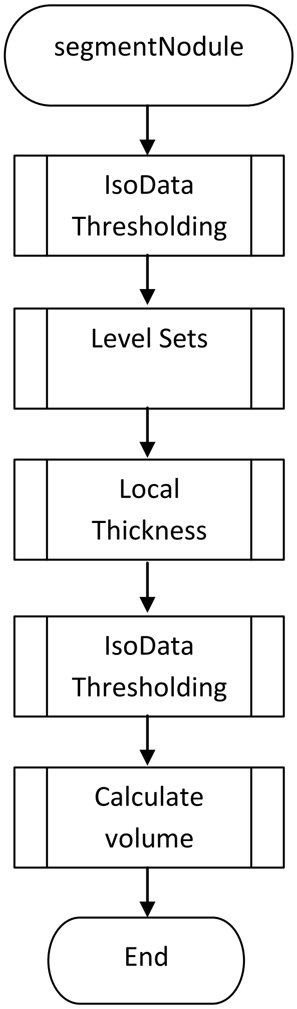
`private Rectangle extraCrop(Rectangle bound)` is a method for extra auto-cropping during user’s ellipse selection. Due to the fact that different people may create an ellipse selection differently, this may create a bias on the final result. To remedy this, the method “extra crop” is used for “tightening” selection down to the area that there are no rows of background color voxels remain in the selection.

To demonstrate the big picture for the explanation above, the flowchart is provided:




A Button got clicked
Action Performed
End


actionPerformed
Cancel button
Source?
OK button
Pinpoint selection phase
Phase?
Ellipse selection phase
Bilateral Filter
segmentNodule
Cropping
Next phase
Terminate Program


segmentNodule
IsoData Thresholding
Level Sets
Local Thickness
IsoData Thresholding
Calculate volume
End

3.8 Implementation

The source code is simply a macro written in Java language, and will run from Fiji Macro Interface. It is served as a macro that provides a simple user-interface, and will do all the tasks from the previous section. We use Fiji library, which is rich in image manipulation, and contain many imaging algorithms [15]. The source code is available at Appendix B.

Chapter IV

Experimental Results

We have proposed our method, and we must test our method in order to make sure that it does work. In order to validate our method, we came up with 2 methods. The first method is correlation test and the second method is using lungman (or “chest phantom”) with artificial nodules.

4.1 Correlation Test

In order to validate that our proposed method works and could segment the specified nodules correctly, one of the validation methods was correlation test.

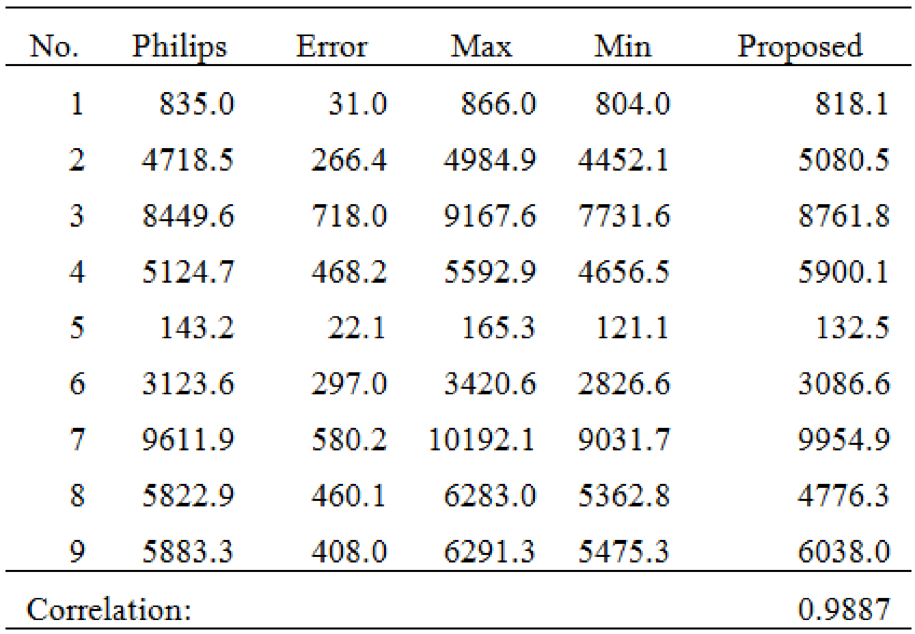
Correlation test, in this case, basically imply whether two methods that are trying to segment and measure a nodule, got similar result or not. We used our method and Philips software for correlation. In our software, the procedure is described in previous chapter. In the Philips Extended Brilliance Workspace software, it has Lung Nodule Assessment ™ method. It will automatically, tried to converge to a volume of the nodule. However, should the user feel the segmentation is incorrect, he or she can perform manual lasso selection, and the software will re-measure the volume accordingly. We will use this manual method along with the automatic one-click segmentation on the Philips side.

The data set contains nine lung nodules for evaluation. In this set, there are regular spherical nodules and also irregular shaped nodules as well. Some of them might have several thorn-like extruding features as well. Nonetheless, these nodules are segmented satisfactorily by both methods, from the visual inspection of our radiologist.

Philips Extended Brilliance Workspace gives out the main volumetric result along with ± error value, which can be used to calculate possible range of minimum and maximum volume of a nodule. In Table 1, the second column shows the volumetric result calculated by Philips, followed with the errors in the third column, the maximum volume (Max column, calculated by adding the error to the main volumetric value), and the minimum volume (Min column, calculated by subtracting the error from the main volumetric value).

We then calculate the correlation between the main volumetric values of Philips and our proposed method's results, and the correlation is 0.9887. This number shows a promising correlation result between the two methods, which employ different techniques to achieve the same goal.

TABLE I.
VOLUME IN mm^3 FOR NODULE ESTIMATION BETWEEN THE TWO METHODS



No.	Philips	Error	Max	Min	Proposed
1	835.0	31.0	866.0	804.0	818.1
2	4718.5	266.4	4984.9	4452.1	5080.5
3	8449.6	718.0	9167.6	7731.6	8761.8
4	5124.7	468.2	5592.9	4656.5	5900.1
5	143.2	22.1	165.3	121.1	132.5
6	3123.6	297.0	3420.6	2826.6	3086.6
7	9611.9	580.2	10192.1	9031.7	9954.9
8	5822.9	460.1	6283.0	5362.8	4776.3
9	5883.3	408.0	6291.3	5475.3	6038.0
Correlation:					0.9887

However, there are some problems with correlation test. We take it for granted that Philips is the gold standard, but in reality, we do not know the internal implementation of the Philips software at all. In this correlation test between two methods, Philips is considered as a black box. Moreover, we never know whether the segmentation is correct or not. Both methods simply output some numbers, using data from the CT scan, but we do not know whether these numbers are closed to the real nodules or not. We do not know whether whose method holds the correct answer, with the correlation test.

Can we take the real nodules out of living patients? The answer to this question lies in a gray area, but most of the time, complication from the patients' side will rise. Patients' information, by medical practice, is confidential, and must not be disclosed, unless the patient agrees to disclose such information. This will be complicated. Nevertheless, if we can measure the real nodules of the patients, this will give us the values from real world measurement that we can used against the programs.

4.2 Lungman and Artificial Nodule

Fortunately, a better method which can solve the complication of ethical medical practice is available. That is, we used Lungman (or "Chest Phantom"), which is an artificial chest, used for biomedical research, and could be used in the CT Scan machine. Furthermore, Lungman features artificial blood vessels, so we will get CT slices of lungs that are very similar to human's lungs.

Lungman set also comes with artificial lung nodules, but these nodules are spherical, and we know that spherical nodules can be easily segmented. We would prefer to use something more random shaped to reflect the fact that nodules could be any shape, and to test our method.

4.3 Material for Artificial Lung Nodule

We have been suggested by a radiologist to use bubblegum as artificial lung nodules. It has been claimed that the material of the bubblegum or rather, the gumbase gives a more similar texture to the lung nodule. Yet chewed bubblegums could be sticky, so we need to cover them with scotch tape, or else it could damage the artificial blood vessels of the lungman. Moreover, we also tried using water-based clay, which is easier to mold and they become not sticky after we left them to dry for a day. Images below show the results of the materials in CT Scan of the lungman.

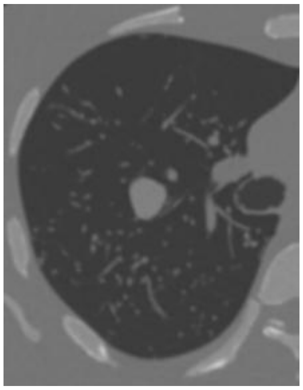


Figure 6 – Gumbase in CT Scan

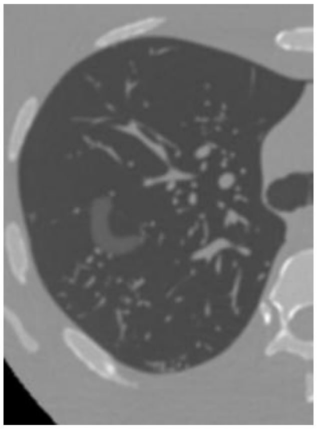


Figure 7 – Water-based clay in CT Scan

Figure 6 shows the gumbase. It is more similar to real lung nodule – the intensity and texture are similar to those of artificial blood vessels. The shape of the bubblegum could also be considered as quite close to the nodule. For figure 7, the water-based clay shows that it has a weaker intensity in the image, and because of this, it could not be used. We have tried Toshiba software to measure the clay, but the segmentation does not cover the whole shape; it only covers the inner part which is only about 30%. Therefore, we should not use the water-based clay, as it is not similar to real nodules at all.

4.4 Measurement and Analysis for each object

Because it is impossible to measure the real volume of the nodule inside of a patient, we suggest using Lungman, filled with bubblegums and perform CT scan on it. Lungman is an artificial lung in a 30 kilogram artificial body, imitating human's body exactly. Lungman is created solely for the educational purpose. This way, we don't have to deal with real patients.

Moreover, we could try to put in some objects, such as bubblegums or water-based clays, into the lungman. Of course, we could measure the volume of these objects in the real world, with Middle School science. By using water and a graduated cylinder, one could measure volume of an object by using water displacement method, originally discovered by the ancient Greek scholar Archimedes.

We have tested with several types of materials:

- Bubblegums
- Water-based clay
- Spherical marble
- Standard rectangular eraser

For preparation, we chewed several bubblegums and tried to create "artificial" nodules out of them. We got about 3-4 sizes of artificial nodules. The same goes with water-based clays. We can change their shape around, and after that, leave them for one day for solidification. The good thing about water-based clay is that they are not sticky after they solidified, unlike regular bubblegum. Therefore, when we put them in lungman, they should not be sticky with those artificial blood vessels. Therefore, preserving delicate artificial blood vessels. This is our main reason to go for water-based clays in the first place.

We have tried with several graduate cylinders, however we don’t have graduate cylinder that allows us to insert bigger objects, while giving more accurate and exact reading. In other words, smaller cylinders can give more accurate reading, but the hole is too small to fit our smallest bubblegum. On the other hand, we can slide bubblegums through the bigger cylinder, but these give us lesser accurate reading. Therefore, we have to make do with what we got.

We have measured our objects with water displacement method and also using our algorithm on the lungman CT slices, as shown in Table II below. Note that, this time, we are requesting Chulalongkorn hospital to perform CT scan on the lungman, so the software, in this case, is Toshiba software.

Table 2: measurement of all objects in millimeters cube

Object	Water Displacement	Toshiba software	Our algorithm
Bubblegum 1	0.5	0.6638	0.63
Bubblegum 2	1	1.514	1.459
Bubblegum 3	1	2.182	2.233
Bubblegum 4	2	3.018	2.941
Spherical Marble	8	7709.8	8998.87
Rectangular Eraser	7	7004.8	7433.99

We thought we could go for more bubblegums, but in order to have an easily distinguishable bubblegum from one another, we prefer small size to big size bubblegum. This is our initial plan, so we bought several bubblegums and chew them up. Unfortunately, we have discovered that we need to chew quite a number of bubblegums in order to get the sizes we want. After chewing the bubblegum, all the sugar and components are removed via saliva, and what remains is gum-based and possibly some more air. The remain is smaller than the original size, of course. Totally, it took 10 bubblegums to create 4 “artificial” nodules with the size we want.

For the eraser and marble, they have intensity higher than the nodules, but for testing purpose, they should be acceptable, since the overall intensity is higher than everything.

4.5 Shape Analysis

To further strengthen our experiment, we created more bubblegums, but this time, we mold the shapes differently to see how our algorithm impacts the segmented shape.

The bubblegums do not have to be big, as the bigger nodule tends to be malignant, and doctors are likely to perform surgery right away. Therefore, we will focus on smaller nodules for shape analysis.

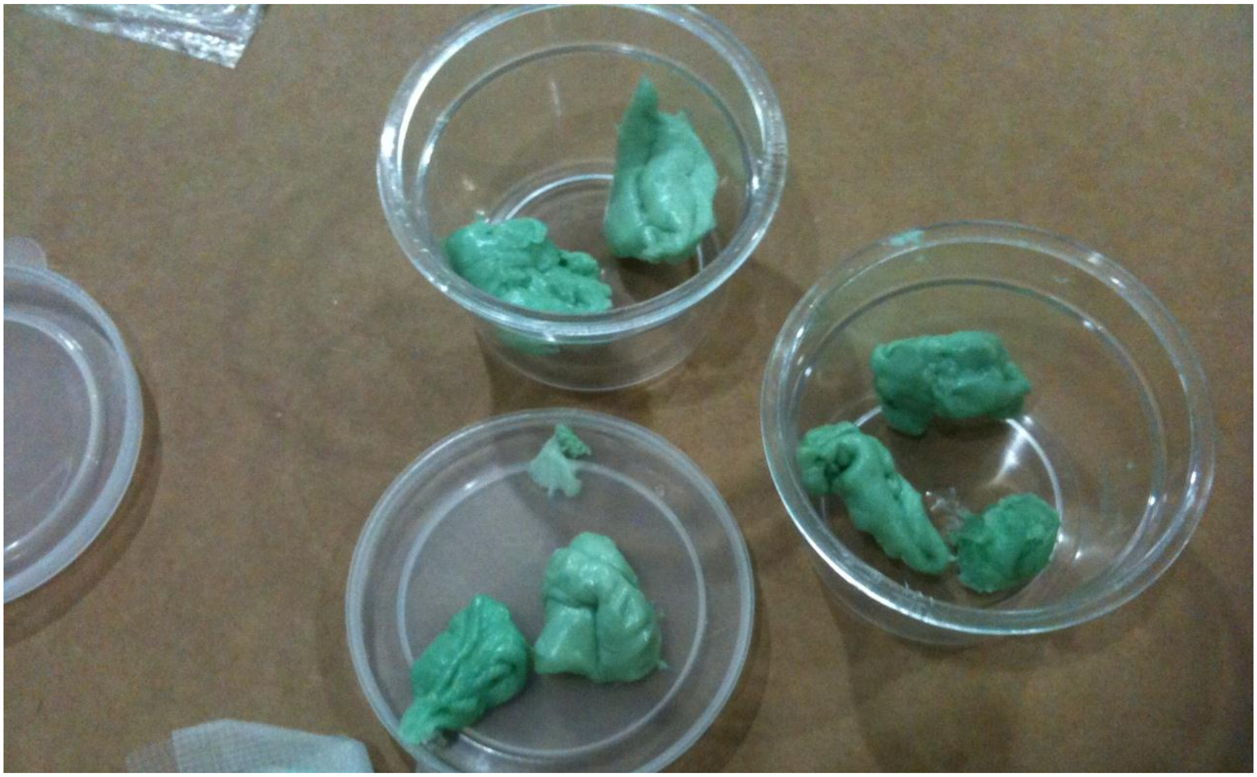


Figure 8: Bubblegums used for shape analysis. We wrapped our bubblegums with Transpore tape.

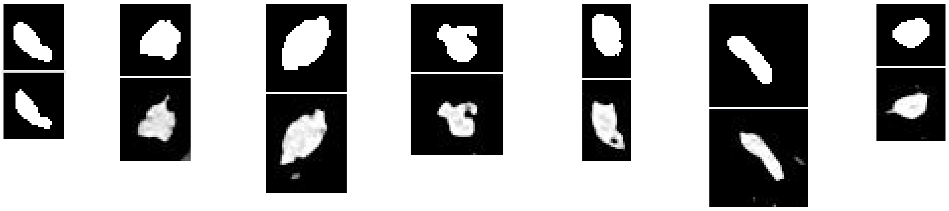


Figure 9: Algorithm's result for bubblegums. Each image on the top is the result of our proposed algorithm. The images on the bottom are the original bubblegum images in Lungman.

On figure 9, we can see the result of our proposed algorithm. We can see that the algorithm removed spikes and extruded features out such as the second and seventh image. On the other hand, we also see on the fifth image that the hole and the surrounding area have been removed entirely. This is acceptable because we want to measure only the denser part. The bubblegum of the fifth image is intended to have a "tunnel" through a part of it and the area around the tunnel should not be taken into account for estimation.

Moreover, we also measured these additional bubblegums and compared with our method in millimeter cube on Table 3.

Table 3: Measurements on additional bubblegums in millimeter cube.

Bubblegum No.	Toshiba's volume	Our method	Bubblegum No.	Toshiba's volume	Our method
1	1497.5	1448.391	5	2083.4	2138.448
2	1411.8	1459.517	6	1514.4	1493.771
3	1339.9	1406.569	7	812.3	760.1925
4	1440.3	1483.579			

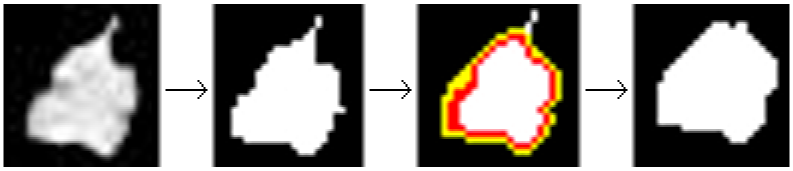


Figure 10: Example of Thresholding on Local Thickness on a bubblegum

Furthermore, we also show an instance on how our method behaves on one of our bubblegums in figure 10. Note that this bubblegum is the second bubblegum on figure 9 and is the second bubblegum on table 3. To clarify better, we upscale our images. The leftmost image shows the original nodule in the CT scan. The second image shows the image that has been thresholded. The third image shows local thickness at work. The white area shows the area containing the thickest zones of the image, while red and yellow show lesser thickness respectively. Once the image has been finalized with the final procedure, we will have the fourth image. We see that the spike on the top has been removed due to thresholding on local thickness. However, some areas have panned out, such as on the top left and top right of the bubblegum. Yet, from the measurement in table 3, we see that this bubblegum got 1411.8 and 1459.517millimeters cube from the two methods respectively. They are very close, although our method has a little bit higher measurement. In order to remedy this, we could set threshold to be more aggressive to exclude all yellow areas on the third image in figure 10.

On a miscellaneous side, for the eraser that we put into Lungman since the beginning, when we tried to segment it with our program, we see that the shape becomes like a Chinese tofu. The long cube has its 8 sharp edges reduced to dull edges. Yet, the eraser is used for material testing in CT scan and there are no real nodules that have rectangular shaped.

All in all, from the tables, we see that the method, more or less, is similar to Toshiba software. This proves another possibility in that our method is also applicable to perform estimation on nodule segmentation.

CHAPTER V

Conclusion and Discussion

From the results, we can see that Thresholding on Local Thickness is another possibility to perform a segmentation and estimation on lung nodule.

In Table I, we see that the results between Philips and our proposed method are fluctuating. This might be because of user-bias during manual selection. The manual selection for Philips Extended Brilliance Workspace could be user-biased during lasso selection adjustment in each frame. The same goes for the proposed method, where cylindrical selection could be user-biased. For example, different users may select the region of different nodules in a different way, thus giving small difference in the final result. This is why the values in the correlation table might jump around between the two methods. These semi-automatic methods after all require user's input and judgment could differ from radiologist to radiologist.

In conclusion, we propose a new method for lung nodule segmentation using local thickness and auto-thresholding. Blood vessels that are attached to the nodules and extruding feature will be removed by IsoData thresholding technique on the local thickness field. This semi-automatic method can reduce oversegmentation and gives us a cleaner segment of the nodule, which makes volume estimation to be more accurate without blood vessels and extruding features. With 0.9887 correlation result, we have validated that our proposed method is another possibility of semi-automatic nodule segmentation method that could work better for irregular shaped nodules.

Moreover, with the result from Table II, we see a similar result from Toshiba software and our method, which further added credibility on correlation in this experiment.

References

[1] D. Ost, A. M. Fein, S. H. Feinsilver, “The Solitary Pulmonary Nodule”. New England Journal of Medicine, 348, 2003.

[2] S. G. Armato III, M. L. Giger, C. J. Moran, J. T. Blackburn, K. Doi, H. MacMahon, “Computerized detection of pulmonary nodules on CT scans”, RadioGraphics, 19, 1999.

[3] M. Antonelli, G. Frosini, B. Lazzerini, F. Marcelloni, “Lung Nodule Detection in CT Scans”, World Academy of Science, Engineering and Technology, 1, 2005.

[4] Y. Liu, J. Yang, D. Zhao, “Computer Aided Detection of Lung Nodules Based on Voxel Analysis utilizing Support Vector Machines”, International Conference on Future BioMedical Information Engineering, 2009.

[5] Brilliance Workspace. [Online]. Available: http://www.healthcare.philips.com/in_en/products/ct/products/ct_brilliance_workspace/

[6] C. Tomasi, R. Manduchi, "Bilateral Filtering for Gray and Color Images", Computer Vision, 1998. Sixth International Conference on, 1998.

[7] T. W. Ridler, S. Calvard, "Picture thresholding using an iterative selection method", IEEE Transactions on Systems, Man and Cybernetics 8, 1978.

[8] T. Hildebrand, P. Rüesgsegger, “A new method for the modelindependent assessment of thickness in three-dimensional images”, J. of Microscopy, 185, 1996.

[9] R.P. Dougherty, K-H Kunzelmann, "Computing Local Thickness of 3D Structures with ImageJ," Microscopy & Microanalysis 2007, Ft. Lauderdale, FL, Aug. 5-9, 2007.

[10] L-K Huang, M-J J. Wang, "Image thresholding by minimizing the measure of fuzziness", Pattern Recognition 28, 1995.

[11] J.M.S. Prewitt, M.L. Mendelsohn, "The analysis of cell images", Annals of the New York Academy of Sciences 128, 1966.

[12] G.W. Zack, W.E. Rogers, S.A. Latt (1977), "Automatic measurement of sister chromatid

exchange frequency", J. Histochem. Cytochem. 25, 1977.

[13] W.J. Kostis, A.P. Reeves, D.F. Yankelevitz, C.I. Henschke (2003), “Three-Dimensional Segmentation and Growth-Rate Estimation of Small Pulmoary Nodules in Helical CT Images”, IEEE Transactions on Medical Imaging, 2003.

[14] Artit C. Jirapatnakul, Anthony P. Reeves, Alberto M. Biancardi, David F. Yankelevitz, Claudia I. Henschke (2009), “Semi-Automated Measurement of Pulmonary Nodule Growth without Explicit Segmentation”, IEEE International Symposium on Biomedical Imaging, 2009.

[15] Johannes Schindelin, Ignacio Arganda-Carreras, Erwin Frise, Verena Kaynig, Mark Longair, Tobias Pietzsch, Stephan Preibisch, Curtis Rueden, Stephan Saalfeld, Benjamin Schmid, Jean-Yves Tinevez, Daniel James White, Volker Hartenstein, Kevin Eliceiri, Pavel Tomancak and Albert Cardona (2012), Fiji: an open-source platform for biological-image analysis, Nature Methods 9(7): 676-682.

APPENDICES

Appendix A

THREE DIMENSIONAL LUNG NODULE SEGMENTATION AND ESTIMATION USING THRESHOLDING ON LOCAL THICKNESS

Akaraphan Janetheerapong, Nagul Cooharojananone,
Rajalida Lipikorn
Department of Mathematics and Computer Science
Chulalongkorn University
Bangkok, Thailand
akaraphanjack@gmail.com

Anan Wattanathum
Department of Pulmonary Medicine
Phramongkutklao Hospital
Bangkok, Thailand

***Abstract*— A fast alternative user-selection procedure to semi-automatic Philips Extended Brilliance Workspace's Lung Nodule Assessment^{TM} is presented. By using modified Local Thickness algorithm which utilizes 3D Region Growing algorithm and IsoData auto-thresholding algorithm, the user can perform fewer steps to semi-automatically segment and estimate a respective nodule more accurately, especially on irregular shape nodules. Attached blood vessels and extruding thorn-like features of the selected nodule are also automatically excluded in segmentation with the proposed method. By using correlation comparison on the same lung nodule dataset between our proposed method and Philips Extended Brilliance Workspace's Lung Nodule Assessment^{TM}'s semi-automatic method, we can see that the correlation is 0.9887.**

Keywords- Lung Nodule Segmentation; Image Processing; Local Thickness; Thresholding

I. INTRODUCTION

Biomedical researches are always a field of interest to many researchers. The progress in this research can increase life expectancy, survivability, health condition, etc. And as technologies progress further, we have better tools and techniques at our disposal to further increase effectiveness of healthcare. For example, in the past, without proper tools for inspecting inside the patient's body, it is impossible to diagnose a disease, such as internal cancer. But with technologies such as X-ray, Ultrasound, or CT scan, it is possible to inspect inside the patient and diagnose the disease more efficiently.

For example, in the case of lung cancer, if it is not detected at an early stage, it could spread to other organs and cause damages, which eventually lead to death of a patient. However, using CT Scans, we can detect these lung cancers at their young stages and deal with them before things get out of hand. According to [1], the chance of survival for patients, with the early stages of malignant nodules, may be as high as 80%, but for patients with advanced stages of malignant nodules, the survival rate is as low as 5% or less. Therefore, if we can detect these lung nodules during their early stages, we can perform medical operation to remove these nodules. Thus, we can ensure higher survivability of the patients.

However, there are some cases that are more difficult than other cases. Manual nodule detection requires the radiologists to skim manually through hundreds of slides. This is a time consuming and tedious task and it is prone to error. Without a doubt, nodule detection in a larger size is easy to spot, but this already implies that the cancer in patient might be in a more advanced stage. If we can detect such nodules when they are at earlier stages, we can treat the patience before things get out of hand. Nevertheless, nodules at such stages are very small in diameter (Less than three millimeters). These nodules may present themselves only in a few CT slices out of hundreds and occupy only a few voxels wide areas. These are very difficult to manually detect by human.

Many proposed methods [2-4] have tried to solve these problems, however, we see that most of the time, they treat nodules as spherical, which is not always true.

In biomedical software, they help the doctors to diagnose these nodules with greater efficiency. For instance, Philips Extended Brilliance Workspace [5] has a semi-automatic Lung Nodule Assessment^{TM} (LNA) method for nodule volume measurement. The doctors can single-click on any nodule for instant segmentation and measurement. He can select option of including or excluding exclude any part after the automatic assessment. However, he has to inspect each frame and adjust the lasso selection of the software's assessment accordingly. But adjusting the lasso selection in each frame could be time-consuming.

Therefore, a fast alternative user-selection method to semi-automatic Lung Nodule Assessment^{TM} in the procedure above, that can segment irregular shaped nodules more accurately is presented.

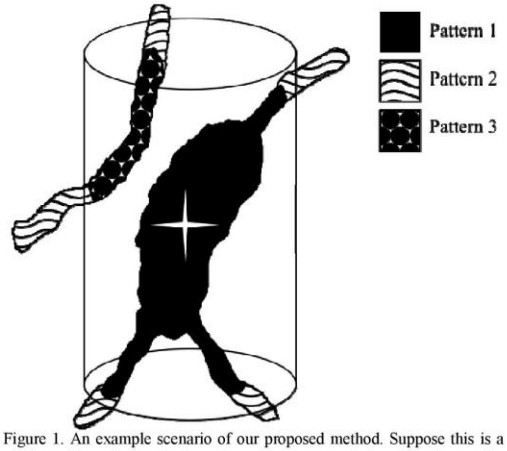


Figure 1. An example scenario of our proposed method. Suppose this is a portion of lung in 3D, the huge lump represents nodule, while the smaller vessels represent blood vessels. By specifying region of interest in 3D via user's parameters, this can create a cylindrical selection to crop all other pulmonary features (Pattern 2) out. By specifying the nodule in the region (Star Point), features not connected to the nodule (Pattern 3) will be removed, leaving the nodule with blood vessels attached. (Pattern 1)

II. PROPOSED METHOD

Lung nodule segmentation and estimation consists of the following seven steps:

A. Pre-Processing

Before performing segmentation, it is mandatory to apply the preprocessing step, which is 3D bilateral filter [6] to smooth an image and at the same time to preserve the edges in an image. This will allow us to have a "cleaner" input.

B. Specification

Our proposed method requires the following parameters from a user for any given CT slices:

1. A user specifies the starting and ending slices of nodule of interest.
2. A user circles the region of nodule.
3. A user specifies the nodule in the region using a single point.

These parameters are crucial in this proposed method. When a user specifies the starting and ending slices, we use these specified indices to retrieve the substack which contains only the nodule of interest. Furthermore, by specifying the region of nodule, we now have an impression of a cylindrical volume of interest. This information can be used to crop the substack. All of the images in the substack will be cropped to this specified location, leaving only the volume of interest in the substack. Finally, by specifying the nodule in this cylindrical volume of interest, this seed point can be used later on to indicate the valid volume of interest. Although the last step may seem redundant, this last step is actually beneficial to indicate some of the more irregular shaped nodules. By indicating the body of nodule, this last step can be used to exclude all other pulmonary tissues except the nodule. Fig. 1 illustrates the whole picture of this section. After that, segmentation is performed.

C. Binarization

Auto-thresholding will be performed on an image utilizing IsoData auto-thresholding technique [7]. This will binarize an image and remove all of the lower intensity voxels, which allows a more accurate result in the region growing section.

D. 3D Region Growing

3D Region Growing algorithm will be performed from a starting point specified by the user from specification step 3. Any other objects outside the acquired region are to be removed. What remains within the user-specified circle should now have only nodule of interest.

E. Local Thickness

There are more tasks needed to be done before volume estimation. For instance, a nodule may have regular blood vessels attached to it. These vessels are not a part of a nodule and must be excluded in volume estimation. Also, as a nodule gets bigger, it tends to develop extruding thorn-like or bump-like features, which, according to our radiologist, should be excluded from volume estimation as well. Local Thickness [8] is used to solve this issue, and is originally defined as:

$$\tau(\underline{p})=2\cdot\max(\{\mathrm{r} \mid \underline{p} \in \mathrm{sph}(\underline{x},\mathrm{r}) \subseteq \Omega, \underline{x} \in \Omega\}) \qquad (1)$$

From Fig. 2 and (1), let $\tau(\underline{p})$ be the Local Thickness function. Let Ω be a structure or object of interest, in which $\Omega \subset \mathbb{R}^3$, $\underline{x}$ be the center and r be the radius of a local largest sphere $\mathrm{sph}(\underline{x},\mathrm{r})$. Any particular point $\underline{p}$ will have local thickness intensity that is equal to the ratio size of all spheres of the local largest sphere, which have been fit into the object in an image, and is a superset of point $\underline{p}$.

The local thickness intensity could be defined as the value of the diameter of the largest sphere that fits inside the given input object at the point $\underline{p}$. From here, we iteratively take each voxel of Ω, and process all of their local thickness intensity to

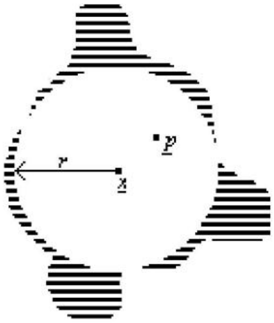


Figure 2. Local Thickness on the striped structure Ω. Only one sphere is shown here for simplicity. In the real definition, a sphere will exist in every voxel of Ω.

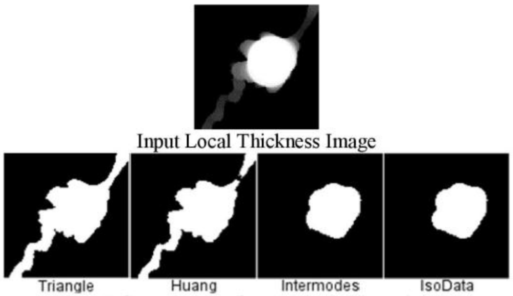


Figure 3. Comparison on a few auto-thresholding techniques

get local thickness field.

We will use an example naïve implementation of local thickness acquisition for the sake of simpler explanation. Let us think of local thickness like this: first, an input image *IMG* will be an image which contains a number of objects. An object in this case refers to a group of white voxels which are connected together in an image. All of the white voxels in the image will have a sphere created with diameter of 1. After that, all of the spheres will increment their diameters in each iteration. Any sphere will stop expanding if the next incremental diameter contains a black voxel. After all of the spheres have stopped expanding, an empty matrix *LT* with the same dimension as *IMG* is created. *LT* will be used for holding local thickness intensity. For each voxel *v* of *LT*, iterate through all of the spheres and check which sphere contains the voxel *v* and which sphere has the largest diameter value. This voxel *v* will hold the value of the sphere which has the highest diameter in this case. After that, the highest intensity value from *LT* will be stored as *MAXINT*. All of the voxels will then be divided by *MAXINT* and multiply with 255. All of the voxels in *LT* will now hold the desired result, a field of local thickness intensity, where 255 is the maximum intensity. This is the result we want.

Several implementations for local thickness exist to improve the algorithm's performance as the original definition of local thickness is too slow for any application. We used R. P. Dougherty's implementation [9] as our base and add an extra step to the algorithm in the next step.

F. IsoData Thresholding on Local Thickness

Usually, a nodule has bigger volume than blood vessels and is expected to be the brightest feature in this user-specified region image. Other parts such as blood vessels should have lower intensity. In order to eliminate these features, we add an extra step into Local Thickness algorithm, which is to perform an auto-thresholding technique. We have tried a few auto-threshold techniques [7, 10-12] and we see that the IsoData is the most stable for our method. (Fig. 3) This way, all of the lower intensity features will be removed, and what remains should be the nodule of interest. (Fig. 4)

G. Estimation

After thresholding on local thickness has been performed, we can now estimate the nodule simply by counting the remaining voxels in the process.

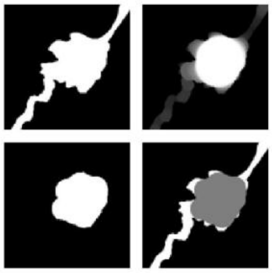


Figure 4. Nodule segmentation simulation: clockwise, starting from top left: the original binarized image, local thickness, auto-threshold on local thickness, and finalized segmentation.

III. CORRELATION TEST

In order to validate that our proposed method works and could segment the specified nodules correctly, just like manually lasso select per frame in Philips Extended Brilliance Workspace, we performed the experiment on a set of lung nodules using two methods and performed a correlation test. We compare the result of Philips Extended Brilliance Workspace's Lung Nodule Assessment^{TM} with our method. The correlation results are shown in Table 1.

For Philips Extended Brilliance Workspace, our procedure is to use Lung Nodule Assessment^{TM} first, and if our radiologist thinks some parts should be include or exclude, additional manual lasso selection on the nodule will be performed.

The data set contains nine lung nodules for evaluation. In this set, there are regular spherical nodules and also irregular shaped nodules as well. Some of them might have several thorn-like extruding features as well. Nonetheless, these nodules are segmented satisfactorily by both methods, from the visual inspection of our radiologist.

Philips Extended Brilliance Workspace gives out the main volumetric result along with ± error value, which can be used to calculate possible range of minimum and maximum volume of a nodule. In Table 1, the second column shows the volumetric result calculated by Philips, followed with the errors in the third column, the maximum volume (Max column,

TABLE I.
VOLUME IN mm^3 FOR NODULE ESTIMATION BETWEEN THE TWO METHODS

No.	Philips	Error	Max	Min	Proposed
1	835.0	31.0	866.0	804.0	818.1
2	4718.5	266.4	4984.9	4452.1	5080.5
3	8449.6	718.0	9167.6	7731.6	8761.8
4	5124.7	468.2	5592.9	4656.5	5900.1
5	143.2	22.1	165.3	121.1	132.5
6	3123.6	297.0	3420.6	2826.6	3086.6
7	9611.9	580.2	10192.1	9031.7	9954.9
8	5822.9	460.1	6283.0	5362.8	4776.3
9	5883.3	408.0	6291.3	5475.3	6038.0
Correlation:					0.9887

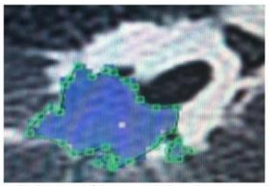


Figure 5. An example image of lung nodule where one-click segmentation may not work, given the irregular shape nodule. The blue area indicates automatic assessment of the Philips software.

calculated by adding the error to the main volumetric value), and the minimum volume (Min column, calculated by subtracting the error from the main volumetric value).

We then calculate the correlation between the main volumetric values of Philips and our proposed method's results, and the correlation is 0.9887. This number shows a promising correlation result between the two methods, which employ different techniques to achieve the same goal.

IV. BENEFIT OF THE PROPOSED METHOD

The difference between the two methods is as follow: Philips Extended Brilliance Workspace's Lung Nodule Assessment^{TM} has one-click nodule selection and the user can adjust the nodule's region in each CT slice accordingly. Our proposed method has substack selection, cylindrical selection and nodule pinpoint selection.

Several conventional methods in lung nodule segmentation treat lung nodule as mostly spherical, but this is not always true. If a nodule is not spherical, there is a higher chance that the segmentation might fail. Therefore, we would prefer substack and cylindrical selection over one-click nodule selection for irregular shaped nodules. This way, we can define the boundary of an irregular shaped nodule. An example of an irregular shaped nodule is in Fig. 5.

Philips Extended Brilliance Workspace allows a user to adjust the region of the lung nodule, should the software segments the nodule unsatisfactorily, from the radiologist's perspective. If the software segments the nodule correctly, one-click nodule selection is sufficient, but again, irregular shaped nodules could be problematic. For irregular shaped nodules, a user might have to go through all of the frames of nodules and fix the lasso selection from the software's assessment accordingly. The proposed method can achieve the objective of this manual step's user-iterative procedure almost completely, as thresholding on local thickness results in automatic selection of the thicker area in 3D.

V. DISCUSSION AND CONCLUSION

In Table 1, we see that the results between Philips and our proposed method are fluctuating. This might be because of user-bias during manual selection. The manual selection for Philips Extended Brilliance Workspace could be user-biased during lasso selection adjustment in each frame. The same goes for the proposed method, where cylindrical selection could be user-biased. For example, different users may select the region of different nodules in a different way, thus giving small difference in the final result. This is why the values in the correlation table might jump around between the two methods. These semi-automatic methods after all require user's input and judgment could differ from radiologist to radiologist.

In conclusion, we propose a new method for lung nodule segmentation using local thickness and auto-thresholding. Blood vessels that are attached to the nodules and extruding feature will be removed by IsoData thresholding technique on the local thickness field. This semi-automatic method can reduce oversegmentation and gives us a cleaner segment of the nodule, which makes volume estimation to be more accurate without blood vessels and extruding features. With 0.9887 correlation result, we have validated that our proposed method is another possibility of semi-automatic nodule segmentation method that could work better for irregular shaped nodules.

ACKNOWLEDGMENT

Akaraphan J. would like to thank Phramongkut hospital for providing data, tools and assistance in this research.

REFERENCES

[1] D. Ost, A. M. Fein, S. H. Feinsilver, "The Solitary Pulmonary Nodule". New England Journal of Medicine, 348, 2003.

[2] S. G. Armato III, M. L. Giger, C. J. Moran, J. T. Blackburn, K. Doi, H. MacMahon, "Computerized detection of pulmonary nodules on CT scans", RadioGraphics, 19, 1999.

[3] M. Antonelli, G. Frosini, B. Lazzerini, F. Marcelloni, "Lung Nodule Detection in CT Scans", World Academy of Science, Engineering and Technology, 1, 2005.

[4] Y. Liu, J. Yang, D. Zhao, "Computer Aided Detection ofLung Nodules Based on Voxel Analysis utilizing Support Vector Machines", International Conference on Future BioMedical Information Engineering, 2009.

[5] Brilliance Workspace. [Online]. Available: http://www.healthcare.philips.com/in_en/products/ct/products/ct_brilliance_workspace/

[6] C. Tomasi, R. Manduchi, "Bilateral Filtering for Gray and Color Images", Computer Vision, 1998. Sixth International Conference on, 1998.

[7] T. W. Ridler, S. Calvard, "Picture thresholding using an iterative selection method", IEEE Transactions on Systems, Man and Cybernetics 8, 1978.

[8] T. Hildebrand, P. Rüesgsegger, "A new method for the model-independent assessment of thickness in three-dimensional images", J. of Microscopy, 185, 1996.

[9] R.P. Dougherty, K-H Kunzelmann, "Computing Local Thickness of 3D Structures with ImageJ," Microscopy & Microanalysis 2007, Ft. Lauderdale, FL, Aug. 5-9, 2007.

[10] L-K Huang, M-J J. Wang, "Image thresholding by minimizing the measure of fuzziness", Pattern Recognition 28, 1995.

[11] J.M.S. Prewitt, M.L. Mendelsohn, "The analysis of cell images", Annals of the New York Academy of Sciences 128, 1966.

[12] G.W. Zack, W.E. Rogers, S.A. Latt (1977), "Automatic measurement of sister chromatid exchange frequency", J. Histochem. Cytochem. 25, 1977.

Appendix B

SOURCE CODE

```
import ij.IJ;
import ij.plugin.PlugIn;
import ij.gui.GenericDialog;
import java.util.Hashtable;
import java.util.Collections;
import java.util.ArrayList;
import java.util.Iterator;
import java.awt.TextField;
import java.awt.event.TextListener;
import java.awt.event.TextEvent;
import java.awt.Color;
import java.awt.*;
import java.awt.event.*;
import java.lang.Runnable;
import ij.*;
import ij.plugin.frame.PlugInFrame;
import ij.gui.*;
import java.awt.*;
import java.awt.event.*;

import ij.*;
import ij.process.*;
import ij.gui.*;
import java.awt.*;
import ij.plugin.*;
import ij.plugin.filter.Info;
import process3d.Flood_Fill;

public class masterNoduleSegmentFiji extends PlugInFrame
implements KeyListener, ActionListener, MouseListener {

      String text;
      private int phase;
      private static final int PHASE_CROP_FRONT = 0;
      private static final int PHASE_CROP_BACK = 1;
      private static final int PHASE_SELECT = 2;
      private static final int PHASE_SEED = 3;

      private double sliceThickness = -1; //in mm
      private double distBetweenSlice = -1;
      private double pixelSpacingWidth = -1, pixelSpacingHeight = -1;


      private int first;
      private int last;
      private int inc;
      String title;
      Label label;

      ImagePlus currentImp;

      ImageCanvas originalCanvas, keptCanvas, seedCanvas;
      String keptCanvasName;
      Button okay, startButton;

                                    OvalRoi oroi;
```

```
                              Polygon polygon;
                              Rectangle bound;

      private String segName = "NODULE SEGMENTATION", seedName =
"SEED SELECTION";

    /** The constructor, preparing the window (panel) */
    public masterNoduleSegmentFiji() {
        super("User Action Required:");

            if (WindowManager.getCurrentWindow() == null){
                  IJ.showMessage("No windows are open! Open the file
first!");
            }else{

                  Info info = new Info();
                  ImagePlus img = WindowManager.getCurrentImage();
                  ImageProcessor ip = img.getProcessor();
                  String s = info.getImageInfo(img ,ip);
                  //IJ.log(s);

                  String[] ss = s.split("\n");
                  //IJ.log(ss[0]);

                  for (int i = 0; i < ss.length; i++){
                        String[] sss = ss[i].split("\\s");
                        //IJ.log(ss[i]);

                        if (sss.length == 0)continue;

                        if (sliceThickness < 0)
                        if (sss[0].equals("0018,0050")){
                              sliceThickness =
Double.parseDouble(sss[sss.length-1]);
                              IJ.log("Slice thickness: " +
sliceThickness);
                        }

                        if (distBetweenSlice < 0)
                        if (sss[0].equals("0018,0088")){
                              distBetweenSlice =
Double.parseDouble(sss[sss.length-1]);
                              IJ.log("distance between slice: " +
distBetweenSlice);
                        }else if (sss[0].equals("Voxel")){
                              String[] ssss = sss[2].split("x");

                              distBetweenSlice =
Double.parseDouble(ssss[2]);

                              IJ.log("Via Voxel: distance between
slice: " + distBetweenSlice);

                        }

                        if (pixelSpacingWidth < 0)
                        if (sss[0].equals("0028,0030")){
                              String[] ssss = sss[sss.length-
1].split("\\\\");

                              pixelSpacingWidth =
Double.parseDouble(ssss[0]);
                              pixelSpacingHeight =
Double.parseDouble(ssss[1]);

                              IJ.log("pixel spacing: w:" +
pixelSpacingWidth + " h:" + pixelSpacingHeight);

                        }else if (sss[0].equals("Voxel")){
                              String[] ssss = sss[2].split("x");

                              pixelSpacingWidth =
Double.parseDouble(ssss[0]);
                              pixelSpacingHeight =
Double.parseDouble(ssss[1]);

                              IJ.log("Via voxel: pixel spacing: w:" +
pixelSpacingWidth + " h:" + pixelSpacingHeight);
                        }
                  }

                  phase = PHASE_CROP_FRONT;
                  WindowManager.addWindow(this);
                  okay = new Button("Cancel");
                  okay.addActionListener(this);
                  okay.addKeyListener(this);

                  originalCanvas =
WindowManager.getCurrentWindow().getCanvas();
                  originalCanvas.addMouseListener(this);

                  GridBagLayout gridbag = new GridBagLayout();//set
up the layout
                  GridBagConstraints c = new GridBagConstraints();
                  setLayout(gridbag);
                  c.insets = new Insets(4,4,2,4);     //(top, left,
bottom, right)
                  c.gridx = 0; c.gridy = 0; c.anchor =
GridBagConstraints.WEST;
                  c.gridx = 0; c.gridy = 1;

                  label = new Label("STEP 1/4: Select the first slide
of the nodule by clicking on respective slide");
                  IJ.setTool("point");

                  add(label, c);
                  c.gridx = 0; c.gridy = 2; c.anchor =
GridBagConstraints.EAST;
                  add(okay, c);
                  c.gridx = 1; c.gridy = 2; c.anchor =
GridBagConstraints.EAST;
                  startButton = new Button("Start");
                  add(startButton, c);
                  startButton.addActionListener(this);
                  startButton.setEnabled(false);
                  this.pack();
                  GUI.center(this);
                  setResizable(false);
                  show();

                  synchronized(this) {                //wait for OK
                        try {wait();}
                        catch(InterruptedException e) {return;}
                  }
            }
    }

    public void close() {
        synchronized(this) { notify(); }

            if (originalCanvas != null)
                  originalCanvas.removeMouseListener(this);
            if (keptCanvas != null)
                  keptCanvas.removeMouseListener(this);
            if (seedCanvas != null)
                  seedCanvas.removeMouseListener(this);

        super.close();
    }

     private String keepSlices(ImageStack stack, int first, int last,
int inc) {
            if (last>stack.getSize())
            last = stack.getSize();

            int count = 0;
            ImageProcessor ip;
            ImageStack newstack = new ImageStack(stack.getWidth(),
stack.getHeight()) ;

            for (int i=first; i<=last; i+=inc)
                  {
                  if ((i-count)>stack.getSize())
                        break;
                  ip = stack.getProcessor(i);
                  newstack.addSlice("slice:" + i, ip);
                  count++;
                  }

            String newName = title+ " " + segName;
            new ImagePlus(newName, newstack).show();
            return newName;
      }

public void mousePressed(MouseEvent e) {}

    public void mouseReleased(MouseEvent e) {}

    public void mouseEntered(MouseEvent e) {}

    public void mouseExited(MouseEvent e) {}

      public static synchronized void FloodFillFromBorder(ImagePlus
imp, int color){
```

```
            //1. how many images? If 1-2 images in stack, flood fill
from all sides.
            if (imp.getStackSize() > 0 && imp.getStackSize() <= 2){

                  IJ.log("stack size is <= 2");
            }else if (imp.getStackSize() > 2){

                  ImageStack stack = imp.getStack();

                  //there are 6 angles to take care of: top, left,
front, back, right, bottom

                  //top: the first image in the stack
                  int x,y,z;
                  z = 0;
                  IJ.log("FloodFillFromBorder top");
                  for (x = 0; x < imp.getWidth(); ++x){
                        for (y = 0; y < imp.getHeight(); ++y){

                              if (stack.getVoxel(x,y,z) == 255){
                                    Flood_Fill.fill(imp, x, y, z,
128);

                              }
                        }
                  }

                  // left: the left plane
                  x = 0;
                  IJ.log("FloodFillFromBorder left");
                  for (y = 0; y < imp.getHeight(); ++y){
                        for (z = 0; z < imp.getStackSize(); ++z){
                              //IJ.log("left y:" + y + " z:" + z);
                              if (stack.getVoxel(x,y,z) == 255)
                                    Flood_Fill.fill(imp, x, y, z,
128);
                        }
                  }

                  // front
                  y = 0;
                  IJ.log("FloodFillFromBorder front");
                  for (x = 0; x < imp.getWidth(); ++x){
                        for (z = 0; z < imp.getStackSize(); ++z){
                              //IJ.log("front x:" + x + " z:" + z);
                              if (stack.getVoxel(x,y,z) == 255)
                                    Flood_Fill.fill(imp, x, y, z,
128);
                        }
                  }

                  // back
                  y = imp.getHeight() - 1;
                  IJ.log("FloodFillFromBorder back");
                  for (x = 0; x < imp.getWidth(); ++x){
                        for (z = 0; z < imp.getStackSize(); ++z){
                              //IJ.log("back x:" + x + " z:" + z);
                              if (stack.getVoxel(x,y,z) == 255)
```

```
                                    Flood_Fill.fill(imp, x, y, z,
128);
                        }
                  }

                  // right: the right plane
                  x = imp.getWidth() - 1;
                  IJ.log("FloodFillFromBorder right");
                  for (y = 0; y < imp.getHeight(); ++y){
                        for (z = 0; z < imp.getStackSize(); ++z){
                              //IJ.log("right y:" + y + " z:" + z);
                              if (stack.getVoxel(x,y,z) == 255)
                                    Flood_Fill.fill(imp, x, y, z,
128);
                        }
                  }

                  //bottom: the last image in stack
                  z = imp.getStackSize() - 1;
                  IJ.log("FloodFillFromBorder bottom");
                  for (x = 0; x < imp.getWidth(); ++x){
                        for (y = 0; y < imp.getHeight(); ++y){
                              //IJ.log("bottom x:" + x + " y:" + y);
                              if (stack.getVoxel(x,y,z) == 255)
                                    Flood_Fill.fill(imp, x, y, z,
128);
                        }
                  }
            }
      }

    public void mouseClicked(MouseEvent e) {
            if (phase == PHASE_CROP_FRONT){
                  if (IJ.getToolName() == "point"){
                        first =
WindowManager.getCurrentImage().getSlice();
                        ImagePlus imp =
WindowManager.getCurrentImage();

                        ImageStack stack = imp.getStack();

                        title=imp.getTitle();

                        IJ.log("first " + first + " size: " +
stack.getSize());

                        phase = PHASE_CROP_BACK;
                        label.setText("STEP 2/4: Select the last
slide of the nodule by clicking on respective slide");
                        show();
                  }
            }else if (phase == PHASE_CROP_BACK){
                  if (IJ.getToolName() == "point"){
                        last =
WindowManager.getCurrentImage().getSlice();

                        if (last < first){
                              IJ.showMessage("Please select a slice
number greater than " + first);
```

```
                        }else{

                              ImagePlus imp = 
WindowManager.getCurrentImage();

                              ImageStack stack = imp.getStack();

                              title=imp.getTitle();

                              keptCanvasName = keepSlices(stack, 
first, last, 1);

                              keptCanvas = 
WindowManager.getCurrentWindow().getCanvas();

      originalCanvas.removeMouseListener(this);
                              keptCanvas.addMouseListener(this);


                              phase = PHASE_SELECT;
                              startButton.setEnabled(true);
                              IJ.setTool("oval");
                              label.setText("STEP 3/4: Select a 
circle area on the nodule. When you are ready, press Start.");
                              show();
                        }
                  }
            }
    }

      int seedX, seedY, seedSlice;
      int originLeft, originRight, originUp, originDown;
      int neoLeft, neoRight, neoUp, neoDown, neoWidth, neoHeight;

      private Rectangle extraCrop(Rectangle bound){
            originLeft = bound.x;
            originRight = bound.x + bound.width;
            originUp = bound.y;
            originDown = bound.y + bound.height;

            neoLeft = neoRight = neoUp = neoDown = 0;//for 
adding,reducing values in bounds.

            ImagePlus impp = WindowManager.getCurrentImage();
            ImageStack stackk = impp.getStack();

            IJ.log("testPlane");

            int x,y,z;
            double testPlane[][] = new 
double[impp.getWidth()][impp.getHeight()];
            for (z = 0; z < impp.getStackSize(); ++z){
                  for (x = originLeft; x < originRight; ++x){
                        for (y = originUp; y < originDown; ++y){
                              double v = stackk.getVoxel(x,y,z);

                              if (testPlane[x][y] < v){
                                    testPlane[x][y] = v;
                              }
                        }
                  }
            }

            IJ.log("left");

            //check test plane for boundary
            //left
            for (x = 0; x < impp.getWidth(); ++x){
                  for (y = 0; y < impp.getHeight(); ++y){
                        if (testPlane[x][y] > 0){
                              neoLeft = x;
                              x = impp.getWidth();
                              y = impp.getHeight();
                        }
                  }
            }

            IJ.log("right");

            //right
            for (x = impp.getWidth() - 1; x >= 0; --x){
                  for (y = impp.getHeight() - 1; y >= 0; --y){
                        if (testPlane[x][y] > 0){
                              neoRight = x;
                              x = -1;
                              y = -1;
                        }
                  }
            }

            IJ.log("up");

            //up
            for (y = 0; y < impp.getHeight(); ++y){
                  for (x = 0; x < impp.getWidth(); ++x){
                        if (testPlane[x][y] > 0){
                              neoUp = y;
                              x = impp.getWidth();
                              y = impp.getHeight();
                        }
                  }
            }

            IJ.log("down");

            //down
            for (y = impp.getHeight()-1; y >= 0; --y){
                  for (x = impp.getWidth()-1; x >= 0; --x){
                        if (testPlane[x][y] > 0){
                              neoDown = y;
                              x = -1;
                              y = -1;
                        }
                  }
            }

            IJ.log("neoLeftRightUpDown " + neoLeft + " " + neoRight +
" " + neoUp + " " +neoDown);
```

```
            neoWidth = neoRight - neoLeft;
            neoHeight = neoDown - neoUp;

            IJ.log("!!! " + bound.x + " " + bound.y + " " +
bound.width + " " + bound.height);

            bound.x = neoLeft;
            bound.y = neoUp;
            bound.width = neoRight - neoLeft;
            bound.height = neoDown - neoUp;

            IJ.log("!!! " + bound.x + " " + bound.y + " " +
bound.width + " " + bound.height);

            return bound;
      }

      public void actionPerformed(ActionEvent e) {
            if (e.getSource() == okay) close();
            else if (e.getSource() == startButton){

                  if (keptCanvas ==
WindowManager.getCurrentWindow().getCanvas() || seedCanvas ==
WindowManager.getCurrentWindow().getCanvas()){
                        if (phase == PHASE_SELECT && keptCanvas ==
WindowManager.getCurrentWindow().getCanvas()){
                              if (keptCanvas ==
WindowManager.getCurrentWindow().getCanvas()){

                                    Roi roi =
WindowManager.getCurrentImage().getRoi();

                                    if (roi.getType() == 1){

                                          //apply bilateral filter

      IJ.run(WindowManager.getCurrentImage(), "8-bit", "");

      IJ.run(WindowManager.getCurrentImage(), "Bilateral Filter",
"spatial=3 range=50");

                                          oroi = (OvalRoi)roi;


                                          polygon =
oroi.getPolygon();
                                          bound =
polygon.getBounds();


      ////////////////////////CONTROLS EXTRA CROP
                                          int performExtraCrop = 1;
                                          if (performExtraCrop > 0){
                                                IJ.log("!");
                                                bound =
extraCrop(bound);
                                                IJ.log("@");
                                          }
                                          ////////////////////////

                                          StackProcessor sp = new
StackProcessor(WindowManager.getCurrentImage().getStack(), null);

                                          //crop

      WindowManager.getCurrentImage().setStack(sp.crop(bound.x,bound.
y,bound.width,bound.height));

                                          phase = PHASE_SEED;
                                          IJ.setTool("point");
                                          label.setText("STEP 4/4:
Click on the nodule, then press start.");
                                          show();

                                          seedCanvas =
WindowManager.getCurrentWindow().getCanvas();

      keptCanvas.removeMouseListener(this);

      seedCanvas.addMouseListener(this);
                                    }else{
                                          IJ.showMessage("Please use
oval selection.");
                                    }

                              }
                        }else if (phase == PHASE_SEED && seedCanvas
== WindowManager.getCurrentWindow().getCanvas()){

                              if (IJ.getToolName() == "point"){
                                    Roi roi =
WindowManager.getCurrentImage().getRoi();
                                    PointRoi proi = (PointRoi)roi;
                                    Polygon p = proi.getPolygon();

                                    seedX = p.xpoints[0];
                                    seedY = p.ypoints[0];

                                    seedSlice =
WindowManager.getCurrentImage().getSlice();


                                    segmentNodule();

                                    close();
                              }else{
                                    IJ.showMessage("Please use point
selection");
                              }
                        }else{
                              IJ.showMessage("Incorrect window
selected");
                        }
                  }else{
                        IJ.showMessage("The currently selected window
is incorrect. Please select the provided window under the name " +
keptCanvasName + ".");
```

```
                  }
            }
     }

      private void segmentNodule(){
            int x2 = bound.x + bound.width, y2 = bound.y + 
bound.height;
            int cx = (bound.x + x2) / 2, cy = (bound.y + y2) / 2;

            ImagePlus imp = WindowManager.getCurrentImage();
            int[] list;

            //clear all the things outside the oval.
            WindowManager.getCurrentImage().setRoi(new 
OvalRoi(0,0,bound.width,bound.height) ,false);
            IJ.run(imp, "Clear Outside", "stack");
            IJ.run("Make Inverse", "");
            IJ.run(imp, "Invert", "stack");
            IJ.run("Make Inverse", "");
            IJ.run(imp, "Auto Threshold", "method=Default white stack 
use_stack_histogram");
            IJ.run(imp, "Select None", "");

            IJ.setTool("point");
            imp.setRoi(new PointRoi(seedX, seedY));
            imp.setSlice(seedSlice);

            IJ.run(imp, "Level Sets", "method=[Active Contours] 
use_fast_marching use_level_sets grey_value_threshold=50 
distance_threshold=0.50 advection=2.20 propagation=1 curvature=1 
grayscale=30 convergence=0.0050 region=outside");

            list = WindowManager.getIDList();
            imp = WindowManager.getImage(list[list.length-1]);

            IJ.run(imp, "Invert", "stack");
            IJ.run(imp, "Local Thickness (complete process)", 
"threshold=128");

            // for debug purpose
            boolean stopB4ThresCheck = false;
            if (stopB4ThresCheck)
            return;

            list = WindowManager.getIDList();

            imp = WindowManager.getImage(list[list.length-1]);

            IJ.run(imp, "8-bit", "");
            IJ.run(imp, "Auto Threshold", "method=Default white stack 
use_stack_histogram");

            ImageStack stack = imp.getStack();

            int totalWhiteVoxel = 0;
            for(int x = 0; x < stack.getWidth(); ++x){
                  for(int y = 0; y < stack.getHeight(); ++y){
                        for(int z = 0; z < stack.getSize(); ++z){
                              if (stack.getVoxel(x,y,z) > 0){
                                    ++totalWhiteVoxel;
                              }
                        }
                  }
            }

            IJ.log("Total white voxel: " + totalWhiteVoxel);

           //these are for setting values manually for debug purpose.
            //sliceThickness
            //distBetweenSlice = 0.799;
            //pixelSpacingWidth = 0.64;
            //pixelSpacingHeight = 0.64;
            IJ.log("Convert to mm^3 (MAX): " + ((pixelSpacingWidth *
pixelSpacingHeight * distBetweenSlice) * totalWhiteVoxel));

            double voxelCountWithZSpacingPerspective = 0;

            for(int x = 0; x < stack.getWidth(); ++x){
                  for(int y = 0; y < stack.getHeight(); ++y){
                        for(int z = 0; z < stack.getSize(); ++z){
                              if (stack.getVoxel(x,y,z) > 0){

                                    //check prev slice
                                    if (z > 0){
                                          if (stack.getVoxel(x,y,z-1)
> 0)

      voxelCountWithZSpacingPerspective += 0.5;
                                    }
                                    //check next slice
                                    if (z < stack.getSize()){
                                          if (stack.getVoxel(x,y,z+1)
> 0)

      voxelCountWithZSpacingPerspective += 0.5;
                                    }
                              }
                        }
                  }
            }

            IJ.log("voxelCountWithZSpacingPerspective: " +
voxelCountWithZSpacingPerspective);

            IJ.log("Convert to mm^3 (3DSpacing): " +
((pixelSpacingWidth * pixelSpacingHeight * distBetweenSlice) *
voxelCountWithZSpacingPerspective));

            int maxArea = 0;

            for(int z = 0; z < stack.getSize(); ++z){
                  int tempMaxArea = 0;
                  for(int x = 0; x < stack.getWidth(); ++x){
                        for(int y = 0; y < stack.getHeight(); ++y){


                              if (stack.getVoxel(x,y,z) > 0){
                                    ++tempMaxArea;
                              }
                        }
                  }

                  if (tempMaxArea > maxArea){
                        maxArea = tempMaxArea;
                  }
            }

            IJ.log("Max Area: " + (maxArea * pixelSpacingWidth *
pixelSpacingHeight) + " mm^2");
      }

      public void keyPressed(KeyEvent e) {}
      public void keyTyped(KeyEvent e) {}
      public void keyReleased(KeyEvent e) {}
}
```

Appendix C

INSTRUCTION / HOW TO USE

To use the macro, one needs to download and install Fiji first. After that, execute Fiji, and go to File > Open, and open the java macro file. The Java source code will be loaded, and Macro Interface will show up with the source code. The next step is to load the DICOM of user's choice with Fiji. (Using File > Import > Image Sequence, if required) After that, run the macro with "Run" button from the Macro interface.

The user has to follow these steps:

1. Once the program is "running", you can use the scroll wheel on the mouse to scroll between the slices of the CT images. You then look for the slices that have a particular nodule. Find the slices that the nodule spans on. For example, if the nodule is on slice 100-123. You will scroll to slice 100, click anywhere on slice 100, then scroll to slice 123, and click anywhere on it.

   Now, you will see that a new window will pop up with a lesser number of slices from the original slice. These slices should be the only slices that have your nodule. For instance, from previous step, you will have slice 100-123.

2. Now, try drag your mouse around (hold left click and release after moving to a new destination) the new window. You will see an ellipse being drawn, just like in those painting programs. This ellipse is your selection. The selection will still stay intact, should you scroll the slices around. This is so that you can see how your selection goes with other slices.

3. You must perform ellipse selection on the nodule, ensure that it covers the whole nodule, not only on one slice, but every slice. Just circle an area around the nodule, scroll around and see if it fits. If everything seems right, press "start" on the "User Action Required" box.

You should see a new window, but the image should be smaller, containing only the area of your selection.

4. In the new window, click on the nodule, presumably on the center. (You can also scroll around and click on the slice that has the largest area of the nodule, if you prefer.) Before you press “start”, make sure that you are on the correct slice. The point you click on served as a x,y coordinate, while the current slice you are viewing will serve as a z-coordinate. You want to have these coordinates on the nodule. Once you have the point of the center of nodule and the correct slice, press “start”.

   Once you press “start”, please wait for a while. Leave it be. At this time, do not click on any Fiji’s windows, as it may confuse a few of the underlying algorithms, and the result might be incorrect.

   Once the program is finished, you will have yet another window popping up, showing you the remaining area, which should be the nodule segmentation volume. The information will also be shown in the log.

Biography

Mr. Akaraphan Janetheerapong was born in 1987. He obtained his degree in Computer Science from the Mahidol University International College, Nakhonpathom, Thailand, in 2010.