Mathematics 412: Topology

Topological Data Analysis Project

Part 1: Bibliography

TopoCAT : Exploring the Potential Role of TDA in Detecting Lung Cancer

Brody Kellish, Anirudh Jonnavithula, Nick von Turkovich

Thank you to the Duke Mathematics Department for the TDA Tools used in this report

April 4, 2016

Contents

I Background **2**

1 The Problem **3**

1.1 Idea and Inspiration 3

1.2 Hypothesis 4

1.3 Data 4

2 The Approach **5**

2.1 Progress So Far 5

2.1.1 Visualizing CT Scans in Osirix Lite 5

2.1.2 Filtering Individual Slides of CT Scans by Pixel Intensity 6

2.1.3 Filtering Individual Slides of CT Scans by Generating Point Cloud of Image and Embedding Cloud in $\mathbb{R}^2$ 9

2.1.4 Quantifying Topological Features of an Entire Scan from Individual Slides 12

2.2 Future Directions 12

2.2.1 Exploring Different Filtrations and Point Cloud Constructions . . . 12

2.2.2 Quantifying Topological Features of an Entire Scan from Individual Slides (cont.) 12

2.2.3 Data Storage and Handling 12

2.2.4 Training and Machine Learning 13

Appendices **14**

A Appendix **15**

A.1 testScript.m 15

A.2 generateS.m 17

A.3 testScript2.m 19

A.4 testScript3.m 23

A.5 testScript4.m 25

Part I

Background

Chapter 1

The Problem

1.1 Idea and Inspiration

Radiology is one of the most important fields of medicine in terms of early detection and diagnosis of insidious, invasive, and ultimately life-threatening disorders and diseases, especially various forms of cancer. However, the analysis of various types of films (CT, MRI, X-ray, etc.) is a largely subjective process, which draws heavily from the radiologist's history, experience, and own natural biases and tendencies.

Modern medical imaging produces large amounts of high-quality image data. Large datasets that combine this image data with official diagnoses are publicly available. A cursory visual inspection of these scans showed obvious visual differences between healthy and ill patients. This delineation suggests that a basic classification algorithm might have some success in analytically determining patient diagnoses.

However, such binary classification is not inherently useful on its own - there is a very clear notion of *severity*, that is directly linked to the proliferation of cancerous tissues. This relationship suggests a classic regression problem, in which the characteristics of various forms of medical imagery may be used as as a predictor for patient prognosis.

We observe that cancerous tissue in the lung forms *nodules*, which grow and form as the cancer spreads. These nodules are typically small, connected, clusters of distinct tissue, with an underlying structure that appears to generally persist between patients. We hope to apply topological analysis to this image data in order to extract relevant features from these tissue structures, which will be used to develop, train, and test classification and/or regression algorithms.

1.2 Hypothesis

The extraction of topological features from CT scans of patients with and without lung cancer will allow us to train a reliable classification and regression algorithm to accurately and reliably predict:

- Whether or not this patient has lung cancer (generally).
- Patient prognosis based on the proliferation, connectivity, and growth of cancerous tissue.

1.3 Data

Chapter 2

The Approach

2.1 Progress So Far

Some preliminary data processing and feature extraction using the TDA toolkit has yielded some interesting, early results. At the same time, it has uncovered some dilemmas including: how to best down-sample an image to preserve important information while making processing the image practical, what is the best way to generate a point cloud from a down-sampled image, what is the best way to summarize the persistence diagrams of each individual slide in a CT scan and the CT scan as a whole.

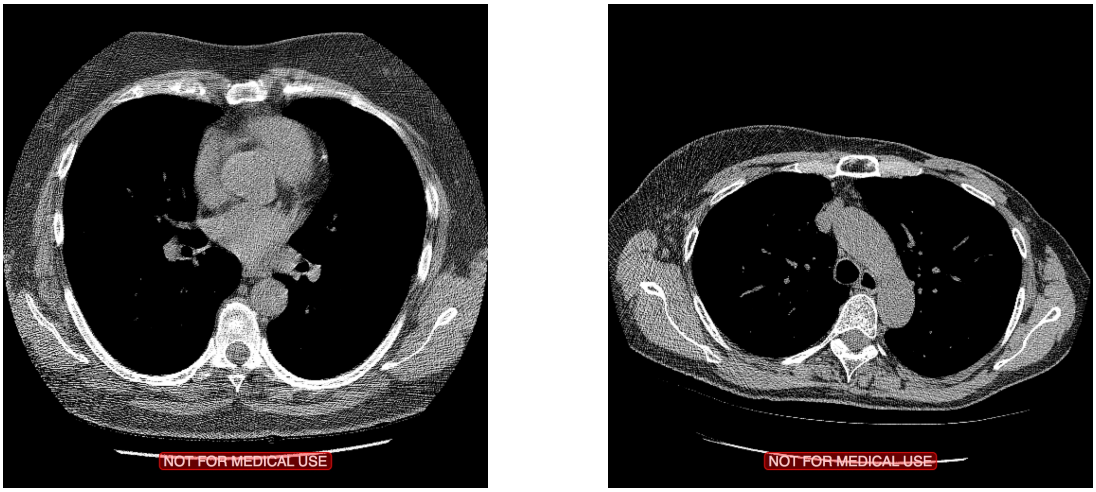
2.1.1 Visualizing CT Scans in Osirix Lite

Osirix Lite is the free version of a popular software that is used to visualize and keep track of medical imagery, specifically scans composed of individual images taken over some height function (like CT scans).

Referring to the annotations spreadsheet that accompanies the LIDC-IDRI, a couple subjects were used to get a sense of the visual differences between patients with a large number of nodules and those with relatively "clear" lungs. The two CT scans selected were ID's 321 (25 nodules) and 306 (0 nodules).

Below are cross sections of the lung that illustrate the difference between images of lungs with virtually no growths or nodules and lungs with a large amount of nodules:

While these cross sections only illustrate the presence of nodules for a particular "height"



(a) A Slide from the CT Scan of Patient 306 (b) A Slide from the CT Scan of Patient 321

Figure 2.1: A figure with two subfigures

within the lungs of the two patients, visualizing the images of these cross sections helped to foster an intuition about the topological features of the lungs or images of the lungs. Just from these images, one can see that the number of unconnected components in the image of the patient with a large number of nodules is greater than that of the patient with clear lungs.

2.1.2 Filtering Individual Slides of CT Scans by Pixel Intensity

The first approach to generating persistence diagrams was to triangulate an image based on pixel intensity (see generateS.m in Appendix) values and use that distance matrix to filter the image (testScript.m in Appendix). The findings did not seem to be point to any strong difference between cloudy and clear lungs using this filtration (more work needs to be done on this route to see if it yields anything interesting).

Below are the pre-processed images of the lungs before analysis with rca1mfscm.m.

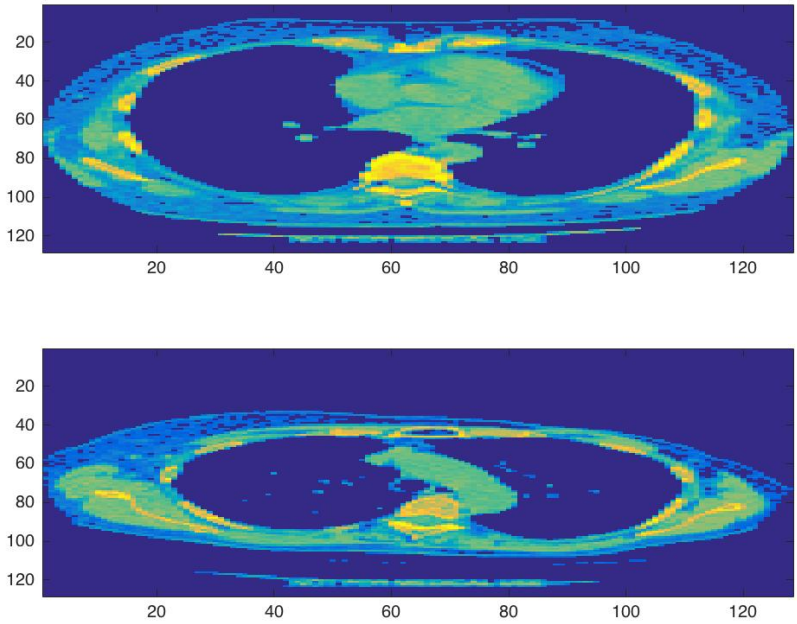


Figure 2.2: Post Image Processing: Clear Lung above Cloudy Lung

Below are the 0-dimensional and 1-dimensional persistence diagrams for the filtration of the image.

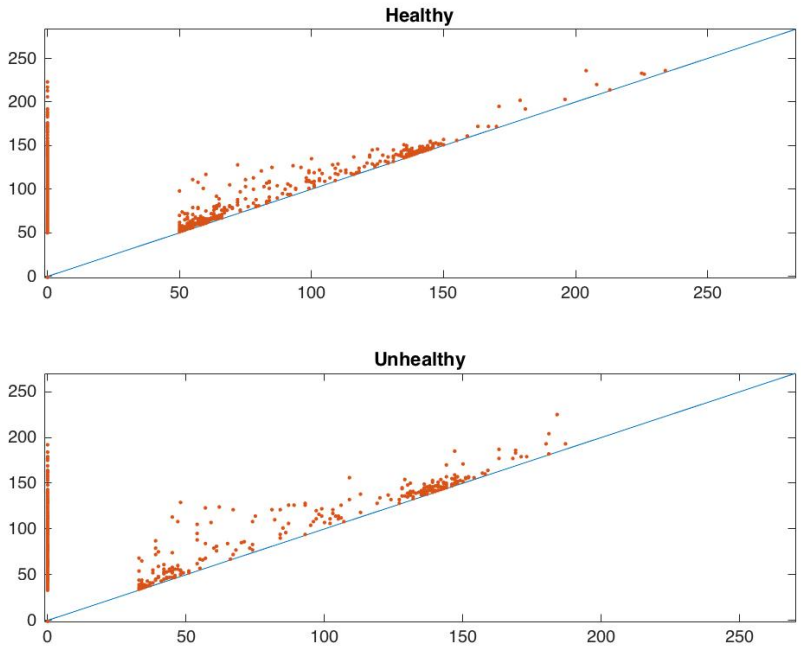


Figure 2.3: Dgm_0

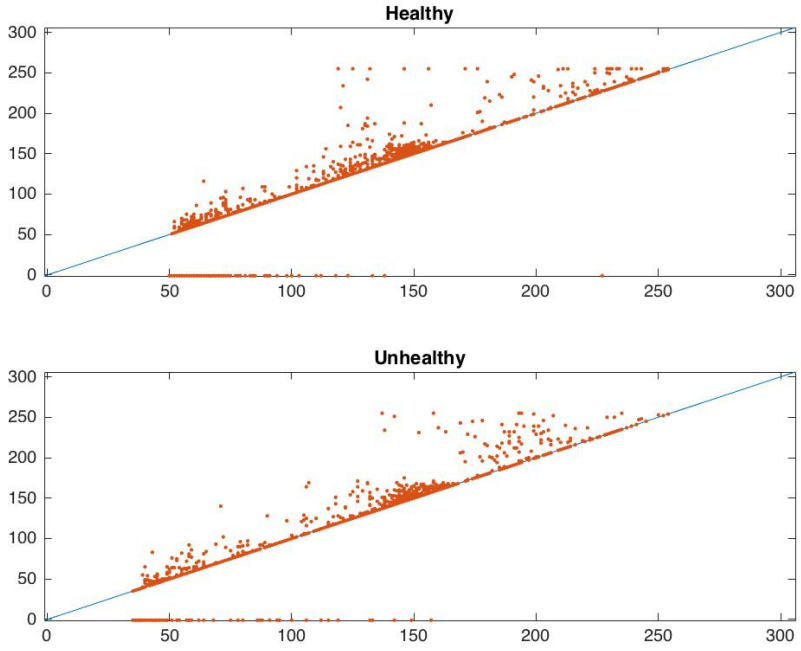


Figure 2.4: Dgm_1

2.1.3 Filtering Individual Slides of CT Scans by Generating Point Cloud of Image and Embedding Cloud in $\mathbb{R}^2$

This approach involved flooring the background of the image and generating points out of the significant pixels (i.e. making the bright pixels in the image vertices). This point cloud was embedded in $\mathbb{R}^2$ and rca1dm.m was used to compute the 0-dimensional and 1-dimensional persistence diagrams (see testScript2.m in Appendix). Below are images detailing the steps of the process:

Below are the original images in grayscale:

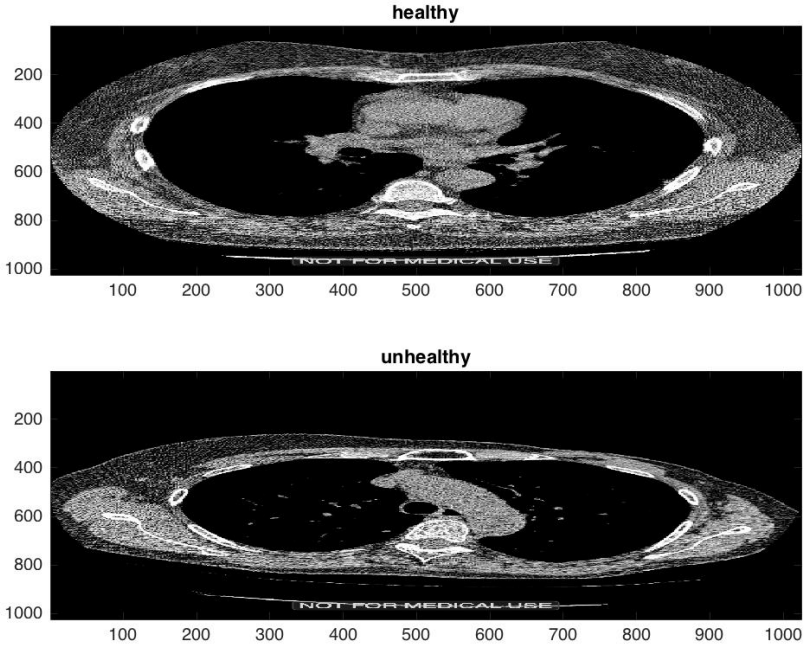


Figure 2.5: Grayscale Images of the Lungs at a Cross-section for Patients 306 and 321

Below are the same images but have been down-sampled by a factor of 0.0625 to make the size of the point cloud manageable:

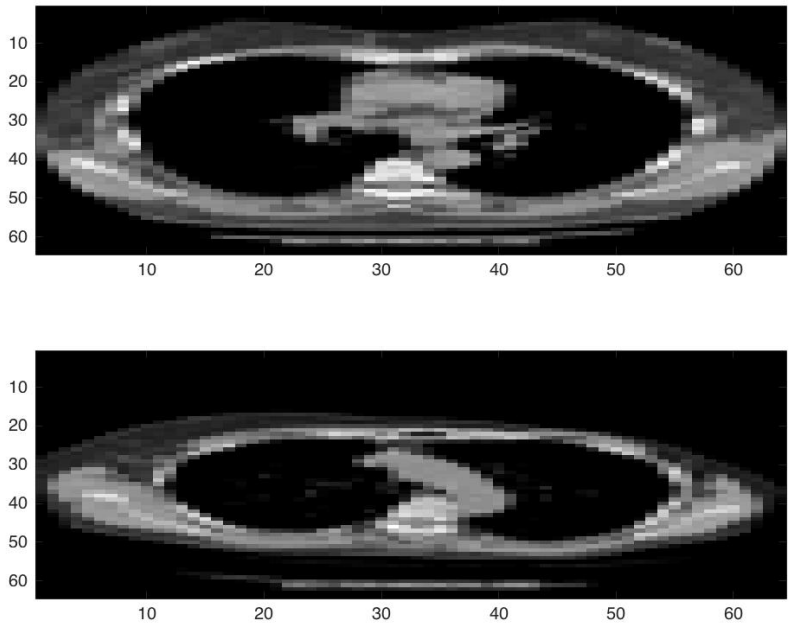


Figure 2.6: Grayscale, Down-sampled Images

Below are the persistence diagrams after running rca1dm.m:

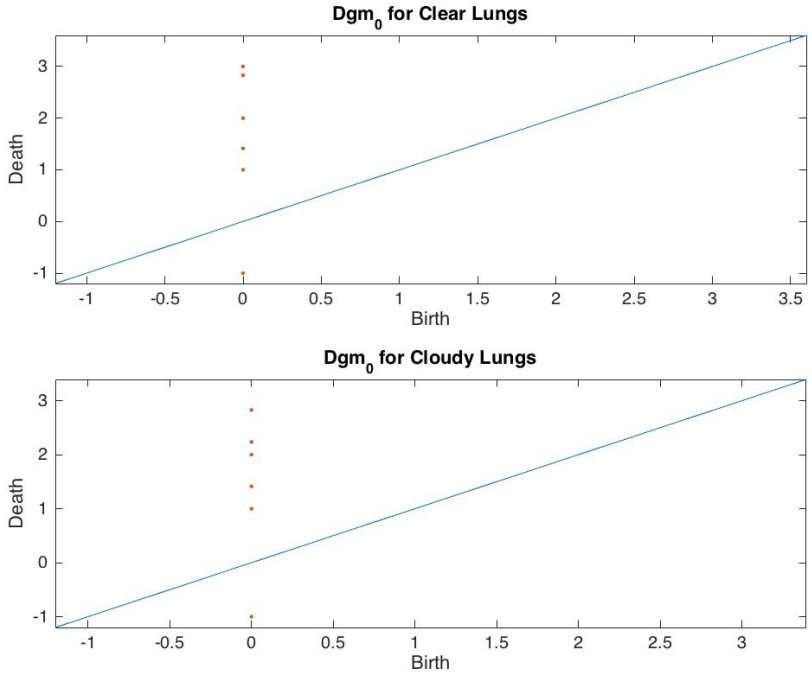


Figure 2.7: 0-Dimensional Persistence Diagram

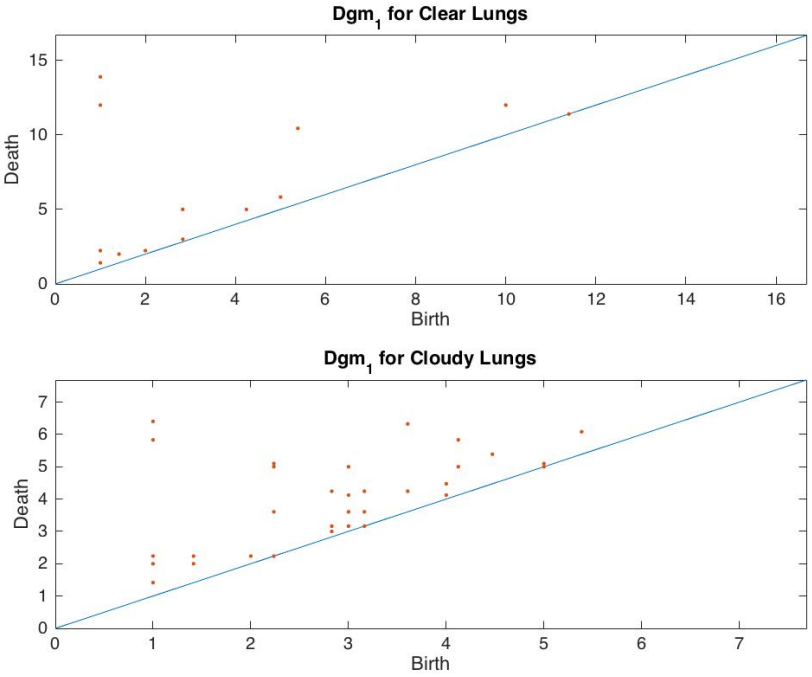


Figure 2.8: 1-Dimensional Persistence Diagram

As can be seen in the persistence plots, there seem to be some differences (in this very

specific case) between lungs with nodules and lungs without nodules in terms of topological features of their CT scans. This is a very promising start and while we are looking into a few different filtrations to try.

2.1.4 Quantifying Topological Features of an Entire Scan from Individual Slides

Some headway has been made in exploring ways to summarize the information of N slides for a particular CT scan. In the Appendix, testScript3.m and testScript4.m are early attempts at computing persistence using the method of the previous section and tiling up metrics on those values into a vector. For example, by looking at some persistence diagrams of different slides by using the method of 2.1.3, one can see that the "clear" lungs have one or two very persistent 1-cycles and a couple very insignificant 1-cycles. However, for slides of "cloudy" lungs, there is a more even distribution. Therefore, testScript3.m and testScript4.m are concatenating the standard deviation of the 1-dimensional persistence intervals and comparing them for healthy and cloudy lungs. More ways of using the data from each slide will be implemented to see what can maximize the difference between CT scans of patients with clear and cloudy lungs.

2.2 Future Directions

2.2.1 Exploring Different Filtrations and Point Cloud Constructions

2.2.2 Quantifying Topological Features of an Entire Scan from Individual Slides (cont.)

More work needs to be done in exploring how the topological features of one image's point cloud can be combined with the other images in a CT scan to create a vector of information that represents one CT scan (see 2.1.4).

2.2.3 Data Storage and Handling

One tedious but important obstacle is how we are going to:

1. Store the 125 GB of CT scan data
2. Divide the data into different blocks for processing
3. Keeping track of each CT scan / how they will be labeled

2.2.4 Training and Machine Learning

The final goal of the project is to train a classifier to identify generally whether a patient has nodes in the lungs or does not as well as a regression model that predicts the degree of tumor proliferation in the lungs based on some metric calculated on the CT scan using our TDA techniques.

Appendices

Appendix A

Appendix

A.1 testScript.m

```
%% Test Script to Load and Examine CT Scan Jpegs
% Nicholas von Turkovich

pathHealthy = '/Users/nbv3/Desktop/Math_Projects/topoCAT/ctScans/306/Ct_Lung
filenameHealthy = 'IM-0001-0060.jpg';

pathUnhealthy = '/Users/nbv3/Desktop/Math_Projects/topoCAT/ctScans/321/Ct_Che
filenameUnhealthy = 'IM-0001-0060.jpg';

image = rgb2gray(imread([pathHealthy filenameHealthy]));
image2 = rgb2gray(imread([pathUnhealthy filenameUnhealthy]));

figure(1)
subplot(2,1,1)
imagesc(image);
title('healthy');
colormap gray

subplot(2,1,2)
imagesc(image2);
title('unhealthy');
colormap gray
```

```
% Need to condense the image

smallerImage = imresize(image,0.125);
smallerImage2 = imresize(image2,0.125);

figure(2)
subplot(2,1,1)
mesh(smallerImage);

subplot(2,1,2)
mesh(smallerImage2);

% processing for image 1

croppedImage = double(smallerImage);

indicesToFloor = find(croppedImage < mean(croppedImage(:)));
croppedImage(indicesToFloor) = 0;

% processing for image 2

croppedImage2 = double(smallerImage2);

indicesToFloor2 = find(croppedImage2 < mean(croppedImage2(:)));
croppedImage2(indicesToFloor2) = 0;

figure(3)
subplot(2,1,1)
imagesc(croppedImage);

subplot(2,1,2)
imagesc(croppedImage2);



% Cropped image ready in uni-valued intensity grayscale
start = tic;
```

```
S1 = generateS(croppedImage);
S2 = generateS(croppedImage2);
%
stop = toc(start)

% compute persistence diagrams

[I, J] = rca1mfscm(S1, 270);
[I2, J2] = rca1mfscm(S2, 270);
figure(4)
subplot(2,1,1)
plotpersistencediagram(I);
title('Healthy');

subplot(2,1,2)
plotpersistencediagram(I2);
title('Unhealthy');

figure(5)
subplot(2,1,1)
plotpersistencediagram(J);
title('Healthy');

subplot(2,1,2)
plotpersistencediagram(J2);
title('Unhealthy');


% figure(4)
% I = sortbypersistence(I)
% plot(I(:,3));
```

A.2 generateS.m

```
function [ sMat ] = generateS( mat )

% This function performs triangulation and generates S matrix
```

```
dim = size(mat);
dim = dim(1);
numPixels = dim^2;

% below is the vertex row array for the S matrix, initialized
vertices = zeros(numPixels, 3);

% filling the vertices array with each pixel of mat
for i = 1:numPixels
    vertexRow = [i-1, i-1, mat(i)];
    vertices(i,:) = vertexRow;
end

% now have all the vertex rows that contain their filtration values
% (intenisty vals)

% find the edge rows

edges = [];
edgeRelations = [1 dim (dim + 1)];



% iterate through all points

for i = 1:numPixels

    % generate all the possible edge pairs (not lower left and upper right
    % diagonals)

    for j = 1:length(edgeRelations)
        neighborIndex = i + edgeRelations(j);
        if neighborIndex < 1 || neighborIndex > numPixels || (mod(i, dim) ==
            continue;
        end
        newEntry = [i-1, neighborIndex-1, max(mat(i), mat(neighborIndex))];
        edges = [edges; newEntry];
```

```
    end

end

sMat = vertcat(vertices, edges);



end
```

A.3 testScript2.m

```
%% Test2
% Nicholas von Turkovich

pathHealthy = '/Users/nbv3/Desktop/Math_Projects/topoCAT/ctScans/306/Ct_Lung
filenameHealthy = '/IM-0001-0050.jpg';

pathUnhealthy = '/Users/nbv3/Desktop/Math_Projects/topoCAT/ctScans/321/Ct_Che
filenameUnhealthy = '/IM-0001-0060.jpg';

image = rgb2gray(imread([pathHealthy filenameHealthy]));
image2 = rgb2gray(imread([pathUnhealthy filenameUnhealthy]));

figure(1)
subplot(2,1,1)
imagesc(image);
title('healthy');
colormap gray
subplot(2,1,2)
imagesc(image2);
title('unhealthy');
colormap gray

% Need to condense the image

smallerImage = imresize(image,0.0625);
smallerImage2 = imresize(image2,0.0625);
```

```
figure(2)
subplot(2,1,1)
mesh(smallerImage);

subplot(2,1,2)
mesh(smallerImage2);

% processing for image 1
% xcropAmount = 10;
% xstartCrop = xcropAmount;
% xendCrop = length(smallerImage) - xcropAmount;
%
%% processing for image 2
% xcropAmount = 5;
% xstartCrop = xcropAmount;
% xendCrop = length(smallerImage2) - xcropAmount;
%
% ycropAmount = 5;
% ystartCrop = ycropAmount;
% yendCrop = length(smallerImage2) - ycropAmount;
%
% croppedImage2 = double(smallerImage2(xstartCrop:xendCrop, ystartCrop:yendCr
% ycropAmount = 1;
% ystartCrop = ycropAmount;
% yendCrop = length(smallerImage) - ycropAmount;
%
% croppedImage = double(smallerImage(xstartCrop:xendCrop, ystartCrop:yendCrop

croppedImage = double(smallerImage);
croppedImage2 = double(smallerImage2);

figure(3)
subplot(2,1,1)
imagesc(croppedImage);
colormap gray;
```

```
subplot(2,1,2)
imagesc(croppedImage2);
colormap gray;

m = mean2(croppedImage);
m2 = mean2(croppedImage2);
dev = std(croppedImage);
dev2 = std(croppedImage2);

% indicesToFloor = find(croppedImage < m);
% croppedImage(indicesToFloor) = 0;
% indicesToFloor2 = find(croppedImage2 < m2);
% croppedImage2(indicesToFloor2) = 0;

figure(4)
subplot(2,1,1)
imagesc(imadjust(croppedImage));
colormap gray;

subplot(2,1,2)
imagesc(imadjust(croppedImage2));
colormap gray;


% compute topology features on a cloud composed of the nonzero 2D vertices

[row col] = find(croppedImage);
[row2 col2] = find(croppedImage2);

figure(5)
subplot(2,1,1);
plot(row, col, 'k.');

subplot(2,1,2);
plot(row2, col2, 'k.');
```

```
cloud = [row col];
cloud2 = [row2 col2];

distances = pdist(cloud);
distances2 = pdist(cloud2);

dmat = squareform(distances);
dmat2 = squareform(distances2);

distanceBound = max(distances);
distanceBound2 = max(distances2);

init;
[I, J] = rca1dm(dmat,distanceBound*0.25);
[I2, J2] = rca1dm(dmat2, distanceBound2*0.25);

figure(5)
subplot(2,1,1)
plotpersistencediagram(J);
title('Dgm_{0} for Clear Lungs');
xlabel('Birth');
ylabel('Death');

subplot(2,1,2)
plotpersistencediagram(J2);
title('Dgm_{0} for Cloudy Lungs');
xlabel('Birth');
ylabel('Death');

figure(6)
subplot(2,1,1)
plotpersistencediagram(I);
title('Dgm_{1} for Clear Lungs');
xlabel('Birth');
ylabel('Death');

subplot(2,1,2)
```

```
plotpersistencediagram(I2);
title('Dgm_{1} for Cloudy Lungs');
xlabel('Birth');
ylabel('Death');
```

A.4 testScript3.m

```
%% Test3: Clear Lungs
% Nicholas von Turkovich

imageNum = 123;

pers0 = [];
pers1 = [];
pathHealthy = '/Users/nbv3/Desktop/Math_Projects/topoCAT/ctScans/306/Ct_Lung
```

```
 for i = 1:10:imageNum

    i

    string = '';

    if i < 10
        string = sprintf('000%d', i);
    end

    if i >= 10 && i < 100
        string = sprintf('00%d',i);
    end

    if i >= 100
        string = sprintf('0%d',i);
    end
```

```
    filenameHealthy = strcat('/IM-0001-', string , '.jpg');

    image = rgb2gray(imread([pathHealthy filenameHealthy]));

    % Need to condense the image

    croppedImage = imresize(image,0.0625);

%
%       xcropAmount = 5;
%       xstartCrop = xcropAmount;
%       xendCrop = length(smallerImage) - xcropAmount;
%
%       ycropAmount = 5;
%       ystartCrop = ycropAmount;
%       yendCrop = length(smallerImage) - ycropAmount;
%
%       croppedImage = double(smallerImage(xstartCrop:xendCrop, ystartCrop:yend


%       m = mean2(croppedImage);
%
%       indicesToFloor = find(croppedImage < m);
%       croppedImage(indicesToFloor) = 0;

%       figure(1)
%       imagesc(croppedImage);


    % compute topology features on a cloud composed of the nonzero 2D vertice

    [row col] = find(croppedImage);

    cloud = [row col];

    distances = pdist(cloud);
```

```
    dmat = squareform(distances);

    distanceBound = max(distances);

    init;
    [I, J] = rca1dm(dmat,distanceBound*0.2);

    pers0 = horzcat(pers0, std(J(:,2)));
    pers1 = horzcat(pers1, std(I(:,2)));


end
```

A.5 testScript4.m

```
%% Test4: Cloudy Lungs
% Nicholas von Turkovich

imageNum = 131;

p0 = [];
p1 = [];
pathHealthy = '/Users/nbv3/Desktop/Math_Projects/topoCAT/ctScans/84/Unnamed__




for i = 1:10:imageNum

    i

    string = '';

    if i < 10
        string = sprintf('000%d', i);
    end

    if i >= 10 && i < 100
```

```
        string = sprintf('00%d',i);
    end

    if i >= 100
        string = sprintf('0%d',i);
    end


    filenameHealthy = strcat('/IM-0001-', string , '.jpg');

    image = rgb2gray(imread([pathHealthy filenameHealthy]));

    % Need to condense the image

    croppedImage = imresize(image,0.0625);

%
%       xcropAmount = 5;
%       xstartCrop = xcropAmount;
%       xendCrop = length(smallerImage) - xcropAmount;
%
%       ycropAmount = 5;
%       ystartCrop = ycropAmount;
%       yendCrop = length(smallerImage) - ycropAmount;
%
%       croppedImage = double(smallerImage(xstartCrop:xendCrop, ystartCrop:yend

%
%      m = mean2(croppedImage);
%
%       indicesToFloor = find(croppedImage < m);
%       croppedImage(indicesToFloor) = 0;
%
%       figure(1)
%       imagesc(croppedImage);
```

```
    % compute topology features on a cloud composed of the nonzero 2D vertice

    [row col] = find(croppedImage);

    cloud = [row col];

    distances = pdist(cloud);

    dmat = squareform(distances);

    distanceBound = max(distances);

    init;
    [I, J] = rca1dm(dmat,distanceBound*0.2);

    p0 = horzcat(p0, std(J(:,2)));
    p1 = horzcat(p1, std(I(:,2)));


end
```