Applying Persistent Homology to Cell Structure Data

By Armaun Sanayei armaun@stanford.edu
Stanford University
Department of Computer Science

1 Introduction

Two of the major problems of analyzing and classifying cell image data are:

- **Varying Environment/Circumstance:** Cell images often taken by a microscope have different amounts of **noise, magnification, brightness, contrast, reflectivity,** etc. that make it hard for a single algorithm or technique to be applied to more than one dataset. For example, when you train a neural network on one set of images from a microscope but then move to a microscope with a different magnification, your neural network will not work as well.
- **Low Amount of Data:** Often for machine learning, you need large amounts of data; however, most image datasets are not that large (you have to be lucky to find a good dataset that is large). Most often datasets are in the range of 15-30 images.

This calls for an algorithm that is resilient to changes in magnification, resolution, etc. as well as one that can work on a low amount of data.

One such technique is computational topological methods. Topological measurements, by their very nature, are robust against changes in magnification, resolution, and noise (to some extent). Topology is interested in the general shape of objects. For example, if they have holes, if they are tubular, if they curve back into each other. These features do not depend on magnification and resolution (unless they become lost in the noise). Moreover, topological measurements do not need much data. Since every type of object has its unique topological "signature" it does not take much data to find out the unique topological signature for a set of geometrically related objects. In fact, even one example of an object with a strong topological signature that we expect to see repeated in other images can be enough for classifcation.

The goal of this project is to use this insight to apply topological algorithms to cell data. As conjectured above, we were able to find that topological algorithms perform very well for classification and segmentation of cell images. Although it may sound like an easy process of simply applying a topological algorithm, the main difficulty is getting the data in the right form to be able to measure these topological characteristics. This is the bulk of the work, and all the algorithms we use for this pre-processing are those that we learned in **CS279**. In many ways, the algorithms from CS279 are the gateway to the topological methods.

2 Background

2.1 Persistent Homology

In this section we will provide a quick background on persistent homology. A more thorough introduction to the method can be found in [1,2,3,4]. In this quick introduction, we will not discuss the mathematics, only the practical aspects of how the algorithm works.

Formally, persistent homology is a topological technique that calculates which homology classes stay present when the resolution of some geometric object changes. Intuitively, persistent homology keeps track of structures of different scale and different dimension and tracks how long different geometric structures stay in the data while we reduce the resolution, until all geometric properties are lost.

Persistent homology works on point-cloud data. It was initially developed to understand the "shape", geometric, and topological characteristics of complex high-dimensional data. Given a point cloud, the algorithm makes a ball $B_r(x_i)$ around each x_i point. The radius of the ball is then varied from $r = 0 \longrightarrow$ (mfv) max filtration value. As r varies for each x_i, different $x_i's$ begin to fall in to the balls of other $x_j's$. In other words, at some point when $r_1 > 0$, we have that $x_j \in B_{r'}(x_i)$ where $i \neq j$. These sets of $\{x_i\}$ of intersecting

points can be recorded, which we call a simplex. Simplexes are of some dimension, depending on the dimensionality of the data and number of points. We can then record when this simplex is "born" and when it "dies" due to a loss of resolution. Of course, this happens for many different sets of points at different radii values, and we record all of these (Birth radius, Death radius) tuples. This leads to a birth-death diagram, which plots the birth time on one axis and the death time on another axis, giving you a scatter plot (intutively all values will be above the diagonal, since death occurs after birth of the structure). Without going into the mathematics, it turns out that these diagrams tell us a lot about the internal topology and geometry of our data (For further reading, read Betti numbers). We will be using these plots for the rest of our analysis. There are many useful images in the references in the bibliography and on the web that may help with understanding how this algorithm works.

2.2 Persistent Homology in Structural Biology

Unfortunately due to the abstract nature and novelty of persistent homology, it has not been widely applied to structural biology data (both in cell image analysis and protein folding). This project hopes to show how useful persistent homology can be in structural biology, specifically in cell-image analysis.

The first thing to note is that many of the unique shapes that appear in cell data have unique topological signatures. For example, consider this figure from a paper on persistent homology that presents the birth-death diagrams for various geometric structures:

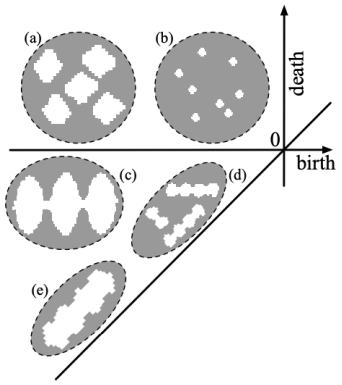

Figure 1. [Figure From Obayashi et al. Persistence Diagrams with Linear Machine Learning Models] (a)-(e) express typical geometric structures in the 0th persistence diagram. (a). Large islands (b) Small islands (c) Large islands with narrow bridges (d) Narrow bands (e) Broad bands

This figure appeared in a paper that has no connection to biological image data, yet a lot of the shapes they analyzed above are exactly the type of structures present in cell image data and that one would like to discern. Moreover, as shown in the image, each structure occupies its own unique region in the birht-death diagram, so this can be use for classification.

Although this project is not the first time persistent homology has been applied to biology data, the methods below are ones I uniquely came up with to analyze the specific datasets that I chose. For the datasets I chose, I believe this is the first time that persistence homology was applied. In general, there are not many examples of where persistent homology has been applied to microscope data; this is because persistent homology is made for non-image data. Therefore, a lot of this project is using methods in CS 279 to extract out meaninful data from the images to input into the topology algorithms.

In terms of novelty of the material, the pipeline (not the algorithms within it) and the classification algorithm is something I came up with for this project and has (to my knowledge) not been done before. Moreover, the segmentation technique/pipeline discussed in the Results section has not been implemented before (to my knowledge).

3 Methods

In the methods section, we will discuss our persistent homology pipeline. Many components of the pipeline were methods learned from CS279. We will then discuss how to use this pipeline for image classification. Additionally, for the figures in this section, we will be using the example of an image of C. Elegans. This example will be thoroughly investigated in the Results/Analysis section as a case study.

3.1 Calculating Persistent Homology of Cell Image Data

Persistent homology is not traditionally applied to image data; it is usually applied to point cloud data where the dimension of the point cloud represents different features observed. Therefore, there needs to be a lot of preprocessing done on the cell image data in order for it to be analyzed by persistence homology. Moreover, persistence homology requires that the structure and topology of the data stay intact while going through these preprocessing steps, so we had to give careful consideration in each step to make sure that an important part of the topology or geometry of the data was not lost with the application of each algorithm.

All the preprocessing steps involve algorithms we learned throughout **CS279** to help digest the noisy image data and turn it into a point cloud. In order of application, we used the following algorithms **PCA, Low Pass Filter, High Pass Filter, Otsu's Algorithm, Morphological Binary Image Thinning Algorithm,** and **Skeleton Modeling Algorithm**. In each paragraph below, we will discuss the specifics of each algorithm as well as why we chose to perform them.

PCA: As discussed in class Principal Component Analysis is a great algorithm for identifying key dimensions of the data that have high variance. In this vein, it can be also used for noise reduction, since it follows that dimensions that capture less variance are most likely noise. Therefore, we use PCA as a first pass removal of noise from the image. Note that PCA was an important step because noise can lead to unwanted geometry in the data (i.e. creating closure to a circle when there is not necessarily a closure).

Low Pass Filter: We specifically use a median 3×3 window low pass filter to further reduce noise as a second pass.

High Pass Filter: We used a Gaussian High Pass filter with a $\sigma = 10$. This σ, however should be changed when dealing with different datasets. For the analysis here, it turned out similar σ values worked well, but when dealing with arbitrary cell data, a choice of σ must be chosen at the very beginning specifically for a dataset. It will usually depend on things like resolution of the image, magnification of the microscope, and brightness/contrast.

Otsu's Algorithm: We used Otsu's Algorithm and a filling algorithm to create a binary mask out of the boundary of the cells that we got from the high pass filter. This temporarily changes the topology of our data (since it fills in volumes that were not filled in), but in the next few steps we will correct for this.

Morphological Image Thinning: This algorithm is specifically used for binary images as we have in this step. It takes a binary image and creates an outline of the filled in volumes that appear in the image and returns only the boundary in a binary image. It has been shown in [5] that this preserves the topology of the structures (which we had before Otsu's Algorithm).

Skeleton Modeling Algorithm: This is a final optional pre-processing step that we performed. It simply takes the high-resolution boundaries from morphological image thinning and turns them into a lower-resolution skeleton that preserves the "skeleton" of the boundary. This is better for the algorithm since such high resolution point clouds are not necessary, it is only necessary for a discernable geometric structure to be present (within the radius resolution we choose). Too many points will cause the algorithm to not be efficient.

After the preprocessing is done, we now have binary images that outline the skeleton of the boundaries of the cells. To get point cloud data from this, we simply assume each cell represents a point with a square plane of $[0, 1] \times [0, 1]$ and we get a point cloud representation of the boundary of the cells.

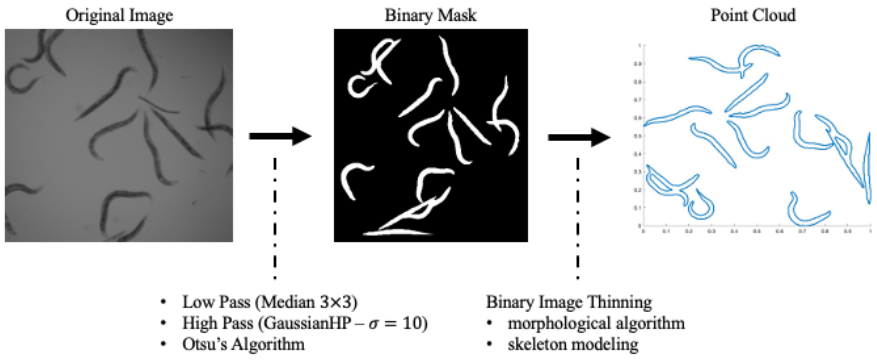

Figure 2. Pipeline for Pre-Processing Image

Now that we have the point cloud data, we perform the **Vietoris-Rips** Algorithm on the data, limiting ourselves to simplexes of 2 dimensions and a filtration value of 1. The choice of simplexes of 2 dimensions was due to the inherent dimensionality and complexity of the shapes present in the cell data. In other words, mostly circles, ovals, tubes, etc. were present in the data, and 2 dimensional simplexes are enough to capture these unique structures. We chose a filtration value of 1 due to the fact that we mapped the point cloud to the $[0,1] \times [0,1]$ space, so we did not expect to find any structures beyond a radius of 1.

After performing Vietoris-Rips, we distilled the output to Betti Plots/Barcode Plots. That look as follows:

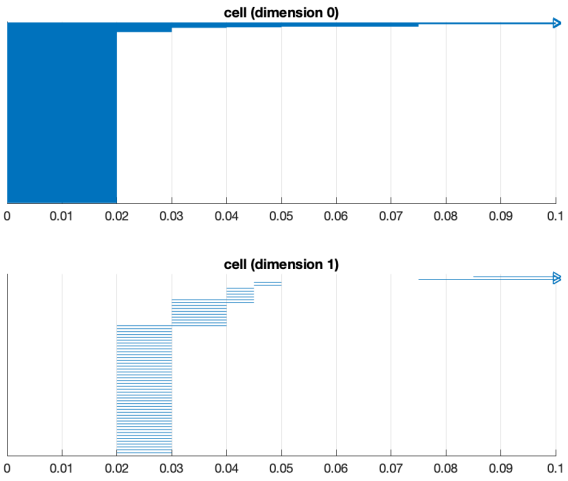

Figure 3. Barcode/Betti Plot of C. Elegans

We further distilled this into a birth-death diagram of structures. We found that these birth-death diagrams gave a unique signature for various cell shapes, cell structures, etc. Multiple birth-death diagrams can be seen in the results section.

This unique signature for each structure is what inspired us to use the birth-death diagram for classification. Moreover, the beauty is that this signature can be acquired when run on only one example and is very consistent across similar structures, even taking into account changes in magnification and noise (since topological structures are resilient to such shifts).

3.2 Summary of Persistent Homology Pipeline

The pipeline can be summarized in six steps:

1. **Noise-Reduction**: Use PCA and a low-pass filter to remove noise
2. **Binary Mask:** Use a high pass filter to get the boundary of the cell, then use Otsu's algorithm to fill in the cell in the binary mask
3. **Cell Boundary as Skeleton**: Use binary image techniques to extract a skeleton of cell boundaries
4. **Covert Skeleton to Point Cloud:** Convert the binary image to a point cloud using a simple map from pixel to $[0,1] \times [0,1]$ space
5. **Compute Persistent Homology:** Run Vietoris-Rips algorithm and get structure/geometry data
6. **Visualize and Classify Results:** Visualize and classify the results using Betti plots as well as birth-death diagrams to get a unique signature for each type of structure.

3.3 Using Persistent Homology for Classification

When we run the above pipeline on an image we get a unique "signature" in the birth-death diagram. Given its uniqueness to each structure. It is possible to use these diagrams as a classification method. We will show an example of how it can be used for classification in the Results section, but here we quickly outline the steps required to turn our pipeline into a classification algorithm.

Given a set of labeled images with some binary label (i.e. 0 - has phenotype, 1 - does not have phenotype), we run this pipeline on all the images. We then add together the birth-death diagrams of all the 0 labeled images and the birth-death diagrams of all the 1 labeled images. Adding these images together and normalizing gives us an overall birth-death diagram (note that summing the "diagrams" is a valid operation since birth-death diagrams are histograms and when we are adding two birth-death diagrams, it is as if we are adding two probability distributions and normalizing). Now we have two overall birth-death diagrams representing the accumulated birth-death diagrams of our two sets.

The classification step works as follows. When given a new image to classify. We run it through the pipeline and get its birth-death diagram. Then we compute a histogram intersection metric between the two birth-death diagrams generated during training. These comparisons give us two scores for similarity, and we assign the new image to the the higher score. Intuitively, we are assigning the image to the label that has a similar "signature" in the birth-death diagram.

This is a new method created for this project. It proved to work surprisingly well since the birth-death plots are very well preserved. The biggest benefit of this classification scheme is that these training sets do not need to be large. In fact, a training set of size 1 for each label is enough to run this classification scheme with good accuracy. In the future examples, we use only 15 images per set. This is a big benefit in the case of biological data, since we often do not have many labeled images.

4 Results and Analysis

4.1 Results

In this section, we will apply this pipeline for classification in two datasets. In the first dataset, we do a simple classification of whether or not C. Elegans bacteria are live or dead. In the second dataset, we do a more complex classification, where we analyze the signatures of various cells from the cell image library data bank and use those signatures to segment an electron micro-graph of the Acinar cell of the salivary gland of a tick (which has remarkable variance in topology and makes it a good candidate for our algorithms).

4.1.1 C. Elegans Live/Dead Assay Classification

In this section, we used the C. Elegans Infection live/dead image set version 1 provided Fred Ausubel and available from the Broad Bioimage Benchmark Collection [6]. The dataset has images of bacteria and has classified them either as "mostly alive" which is positive or "mostly dead" which is negative.

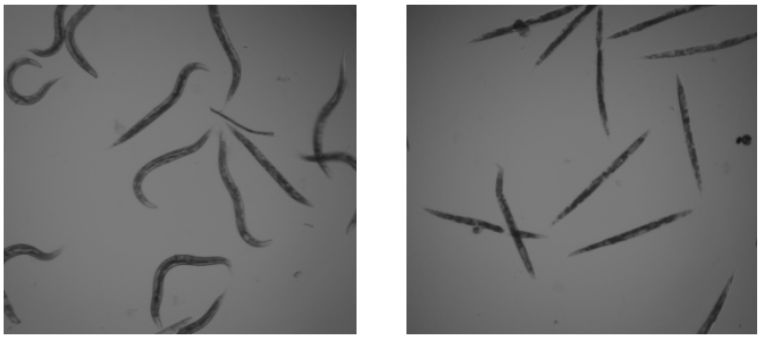

Figure 4. C. Elegans (a) Mostly Live - Positive (b) Mostly Dead - Negative

The goal is to be able to extract a binary mask of the image and then classify them as alive or dead. Our classification scheme was able to do this with 97% accuracy, only misclassifying one image.

The reason we thought that this would be a great dataset for persistent homology was the topological and geometric characteristics of the two sets being classified were different. As seen in the figure, the main characteristic that made the bacteria look alive was their "curved" nature while dead bacteria were

straight . The simplicial complexes in the Vietoris-Rips algorithm are able to pick up on these difference in geometry and thus lead to a different birth-death plot. For example, running two sample images from the training set through our pipeline, we get the following unique birth-death plots:

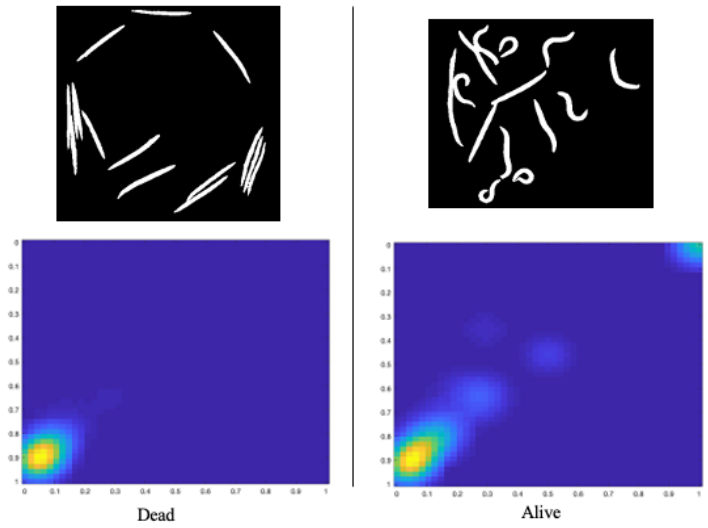

Figure 5. Birth - Death Plots of Alive and Dead Image

Although the bottom left of the two plots is similar (this is because of the tubular structure that is being captured), the key difference (that is present in all the "live" images), is the unique signature in the top right corner of the birth-death diagram, which represents the presence of a global curvature structure; something born late is in the top right corner, and things that are born late represent larger geometric structures, in this case curvature along the entire bacteria.

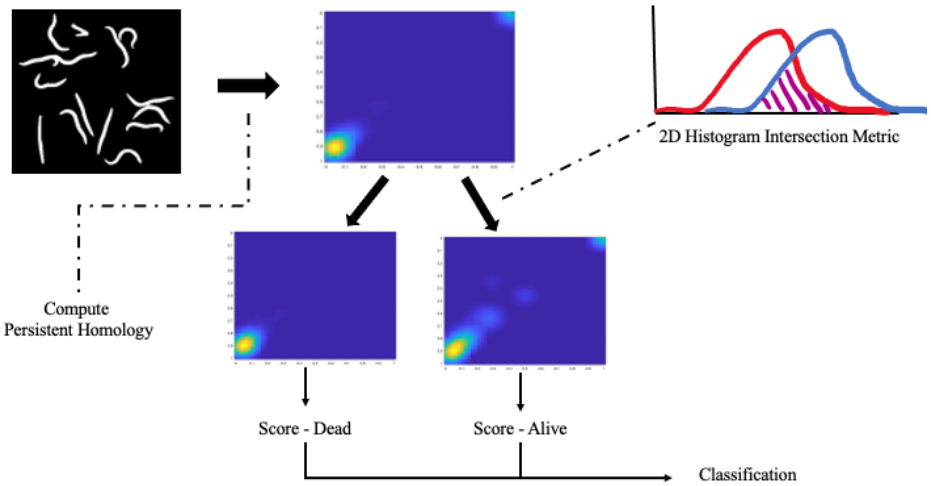

Figure 6. Classification Pipeline using Persistent Homology

Now putting these in our classification pipeline, we get the 97% accuracy. The scores of some 4 of the 30 images were near 0.5 (i.e. 0.5 is the threshold for our probabilistic score). This is because of the fact that many of the images actually had some mix of dead and alive bacteria. In these cases, the algorithm was not strong enough in either category. As a future addition to this algorithm, we would like to find a way to separate the individual bacteria from each other and then perform the classification on each bacteria individually. This will lead to no ambiguity on the classification sets.

4.1.2 *Analysis of Cells from Cell Image Library Data-bank*

To see the unique topological signatures of various types of cells (with varying topologies), I ran the pipeline on the images in the Cell Image Library [http://www.cellimagelibrary.org/] form the Center for Research in Biological Systems (CRBS). Here we will show two representative examples and then we will use our classification pipeline to segment a third, more complicated image.

Transmission Electron Micrograph (TEM) of Rough Endoplasmic Reticulum:

Here we ran this tube-like structure through our pipeline. We got the following birth-death curve.

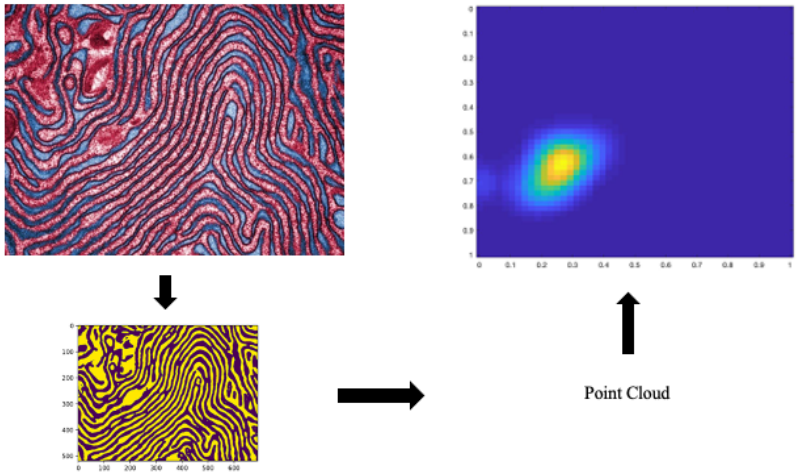

Figure 7. Running RER if Acinar Cell Image through Pipeline

Electron Micrograph of Plasma Membrane of Epithelial Cells:

Here we ran this very clean circle structure through our pipeline. We got the following birth-death curve:

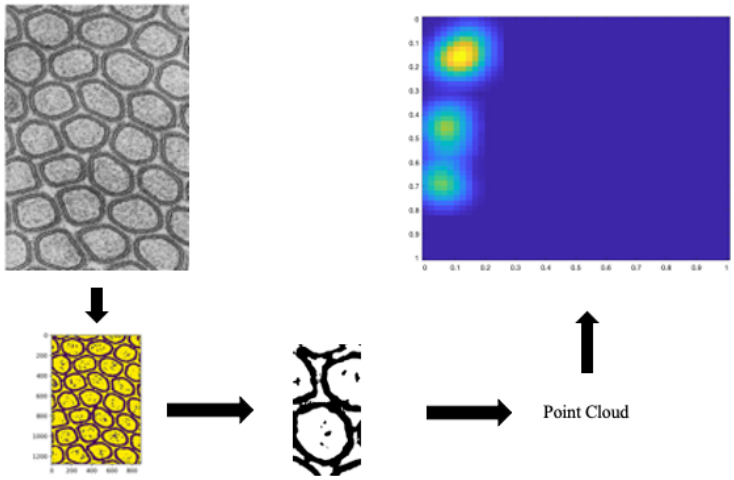

Figure 8. Running Epithelial Cells through Pipeline

Segmentation of TEM of the Salivary Gland of the Tick:

This structure is specifically noted by biologist for the dramatic difference in nuclear size seen in the different cell types. This dramatic variation in circle-like structures is exactly the type of topological data that our algorithm can pick out and use to segment.

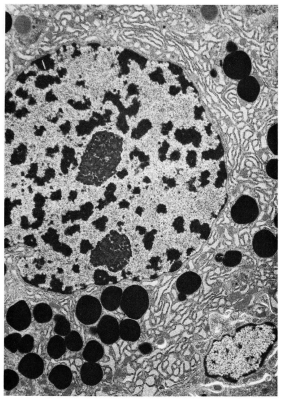

Figure 9. TEM of the Acinar Cell of Salivary Gland of the Tick

As illustrated in the following diagram, we segmented the image into tiles and then classified each tile according to its birth-death curve (using our classification algorithm). Then according to which group the classification algorithm placed the tile, we painted the grey image either red, blue, or green (depending on which classification it got). Note that the classification usually does only two groups. In this case, we were able to identify three groups/labels since one label (green) was for the case that there were no significant topological features.

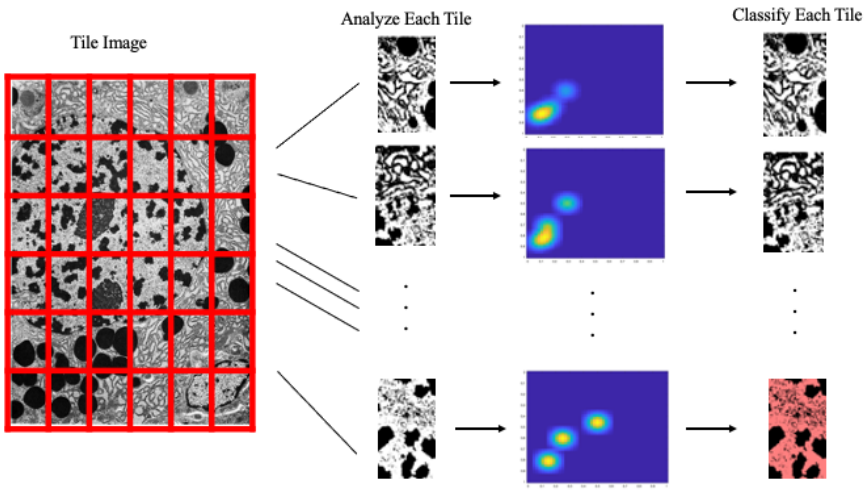

Figure 10. Image Tiling and Classification Pipeline

After classification and coloring, we stitched the tiles back together to get.

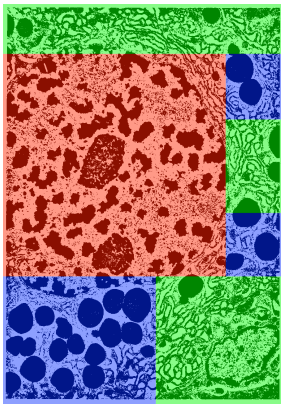

Figure 11. Stitched Segmentation of Tiled Image

As seen , it is remarkable at how well the segmentation performed.

4.2 Analysis

4.2.1 Major Design Decisions

The major design decisions involved how to pre-process the data into a point cloud, the parameters to run the topological algorithm, and finally how to go about classifying data and come up with a robust pipeline to implement classification. Each of these involved alot of adjustment. In the pre-processing step, there was a lot of adjusting involved with finding the best filters to perserve topological structure and adjusting the filter parameters. In the case of the topological algorithm, there was some important adjusting with the parameters of the topological algorithm (i.e. considering too many geometric structures would lead to a very long run time, while considering too few geometric structures would not lead to a unique plot). Finally, the classification took a while to perfect (especially the intersection metric and scoring metric for the different labels, which had to be developed from scratch for this project).

4.2.2 Challenges

The main challenge I faced was converting the data to point cloud data in a robust and repeatable manner. This involved using a lot of methods from CS279 and was the bulk of the work. The other half of the work was coming up with the classification pipeline, which took a while to implement in pipeline. Moreover, I had to learn how to use the JavaPlex package, which is quite a complicated package due to its dependency of Java structures (more discussion in the appendix).

4.2.3 Overall Usefulness Implementation

Although this implementation has proven to be very good (as show in the results), it should be noted that I chose datasets that I knew had good topological data. For example, I purposely went through the datasets on the cell image library and the Broad Bioimage Benchmark Collection to find datasets that had really clear topological structures. In other words, even though the algorithm is promising, I think the choice of datset is just as important. The data your looking at must have clear geometry.

4.2.4 Major Takeaways

The major takeaway is that the the algorithms used in CS279 form the bedrock of any future analysis I want to do with cell images. As seen here, without the algorithms from CS279 I would not have been able to apply any of the topological methods to the data.

4.2.5 Next Steps

There are many next steps I have planned (and even one that I implemented but did not include, due to lack of space). In relation to the methods above, one next step that I envision is using a geometry "recognition" algorithm that can crop out any important features of the cell image so that each geometric object can be analyzed on its own. For example, in the case of C. Elegans it would be nice to analyze each bacteria on its own. Or in the case of the Acinar cell, it would be nice to look at the geometry of each nucleus on its own and use that for a higher-resolution segmentation.

Another possible avenue of exploration that I have started to implement is using persistent homology to study protein structure. Protein structures are inherently 3D point cloud data and might have interesting topological structures that can be investigated (especially the dynamics of proteins might end up being some multi-dimensional shape).

A Appendix A: JavaPlex and How To Use the Code

All the codes in this project were written from scratch (i.e. no starter code). The only possible case of starter code is *two lines* when I read in the image (I copied the tiff image reading portion from our last PSET, but this is relatively minor in the entire scale of the project).

The code for this implementation is in two languages, python and Matlab. Python is the primary language in which the pre-processing, image tiling/stitching, etc. is done. Matlab is used for calculating the persistent homology. Specifically, I used Matlab because of the capability to run JavaPlex, a Stanford-made package for topological data analysis. JavaPlex has a lot of capabilities, and Vietoris-Rips algorithm is one of them.

Although I was not really familiar with JavaPlex, I learned how to use JavaPlex for this project. It turns out that it is quite a complicated package to work with since you are using Java objects inside Matlab. This leads to messy types and data formats. To run my code, however, the user will not need to know how to use JavaPlex. They will only need to install the latest version of JavaPlex from JavaPlex | Javaplex (appliedtopology.github.io). Once downloaded, the user must run the "load_javaplex.m" file in the library when an instance of Matlab has opened.

Afterward, navigate to the folder with your files and you can run all the algorithms. Here is a quick breakdown of the codes and there functions:

image_pre_processing.py - this is an interactive python script I made from scratch that takes in an image name. It then displays the image with all the pre-processing done to it (i.e. PCA, low-pass, high-pass, Otsu's Algorithm, filling ,etc.). The remarkable part of this script is that it allows the user (presumably a biologist) to change the parameters of the filters (i.e. size of window for low pass filter, standard deviation of Gaussian high pass filter, etc.) in real time while seeing the effects of the changes on the image. The user can change the parameters via sliders under the image to get the perfect combination they need to have their image preprocessed and remove noise. Here is an example of the interactive environment:

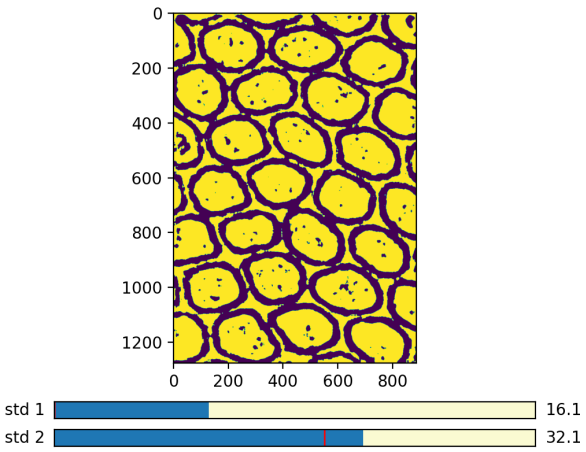

Figure 12. In this example, th user has control over the standard deviation of the first low pass Gaussian filter (std1) as well as the the second Gaussian high pass filter (std2). The choice of filter can be adjusted based on the user preference. Changing the std is as simple as clicking on the slider and dragging the blue bar.

Once you have selected your choice of standard deviation and filters and close the figure, the script saves the binary mask and slices into tiles and saves those if desired (i.e. in the case of segmentation).

This binary mask is then uploaded into one of the many possible Matlab files. Since there are many, here are brief explanations of many of them:

- **single_image_ph.m** - Computes the persistent homology graph (birth-death diagram) of a single image and also computes the Betti/Barcode plot of the image. Displays the two images with the option of saving them.

- **group_image_ph.m** - Computes the cumulative persistent homology histogram of an entire group of images (i.e. all have the same label) by summing together the birth-death histograms. Displays histogram as a heat map. You are simply required to put in the directory of the file with all the images in the group (all files in the directory, must have the same label).
- **image_tiles_ph.m** - Computes persistent homology for each individual tile in an image and save the birth-death plot for each tile in a directory. Requires user to input name of directory where image tiles are stored.
- **classification_group_ph.m** - Computes the cumulative persistent homology histogram for two groups individually. Very similar to group_image_ph.m, but computes two groups simultaneously. This is useful for classification pipeline ,rather than just looking at the plots.
- **classify_images_ph.m** - After running classification_group_ph.m, run this script on the data you wish to classify. It uses the histograms generated from the previous script (and saved into the same folder) to run classification on a specific set of images you specify. It outputs a matrix, called "overall" that has the score0, score1, ratio, unclassified_flag. The scores represent which group it belongs to, the higher the score, the more likely it belongs to that group. The unclassified flag is in the case that the persistent homology was not able to extract any two dimensional features. In which case, we just say it falls into a third group of "unknown".

Note that all of these files run the morphological algorithm to convert the binary mask into a point cloud, since it is easier to do this image analysis in Matlab (point cloud data is difficult to transfer and format).

BIBLIOGRAPHY

1. Carlsson, Gunnar. ``Topology and data.'' Bulletin of the American Mathematical Society 46.2 (2009): 255-308.
2. Edelsbrunner, Herbert, and John Harer. Computational topology: an introduction. American Mathematical Soc., 2010.
3. Obayashi, Ippei, Yasuaki Hiraoka, and Masao Kimura. ``Persistence diagrams with linear machine learning models." *Journal of Applied and Computational Topology* 1.3 (2018): 421-449.
4. Adams, Henry, et al. ``Persistence images: A stable vector representation of persistent homology." *Journal of Machine Learning Research* 18 (2017).
5. Haralick, Robert M., and Linda G. Shapiro, *Computer and Robot Vision*, Vol. 1, Addison-Wesley, 1992.
6. Ljosa, Vebjorn, Katherine L. Sokolnicki, and Anne E. Carpenter. ``Annotated high-throughput microscopy image sets for validation." *Nature methods* 9.7 (2012): 637-637.
7. doi:10.7295/W9CIL42804, CIL:42804, acinar cell. CIL. Dataset (cellimagelibrary.org)
8. doi:10.7295/W9CIL12063, CIL:12063, Felis catus, epithelial cell, enterocyte. CIL. Dataset (cellimagelibrary.org)
9. doi:10.7295/W9CIL10970, CIL:10970, Rhipicephalus appendiculatus, acinar cell of salivary gland. CIL. Dataset (cellimagelibrary.org)