CLUSTERING OF FEATURES OBTAINED FROM PATHOLOGY DATA

Project Report

CSE 595 – DATA ANALYTICS & SOFTWARE STACKS

Akhilesh Chaganti

Gautham Reddy Kunta

Nafees Ahmed Abdul

Raghavendra Suvvari

Vamsi Krishna Atluri

Contents

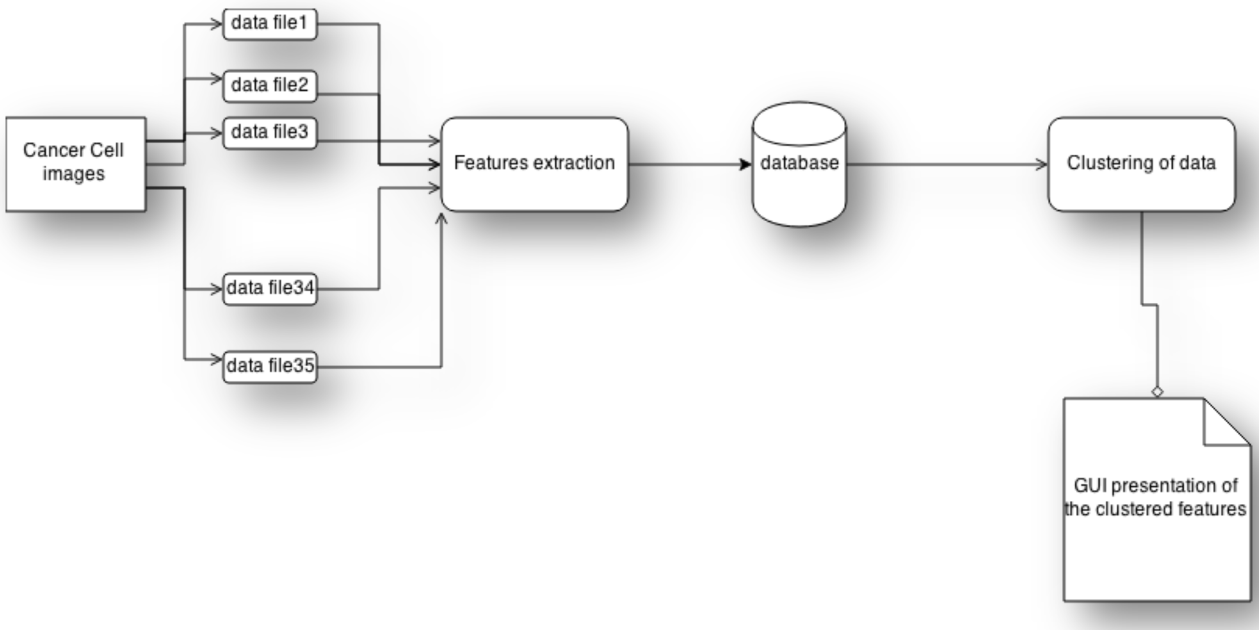
Motivation

In pathological analysis identification of infected cells is a real challenge especially cancerous cells in case of cancer detection. The main reason is that there are huge numbers of cells that are to be considered during analysis and it is quite difficult to distinguish malicious cells from normal cells. So identifying these malicious cells by their unique properties/features becomes the significant part of any tumor analysis.

Our aim was to create an application that enable pathologists input their image data and allow them to use various clustering algorithms to cluster various cells. We need to generate different features from input data, which can be used by pathologists to study and distinguish various nuclei depending on these features. Also our goal was to provide interactive UI for the visualization of the output.

Architecture

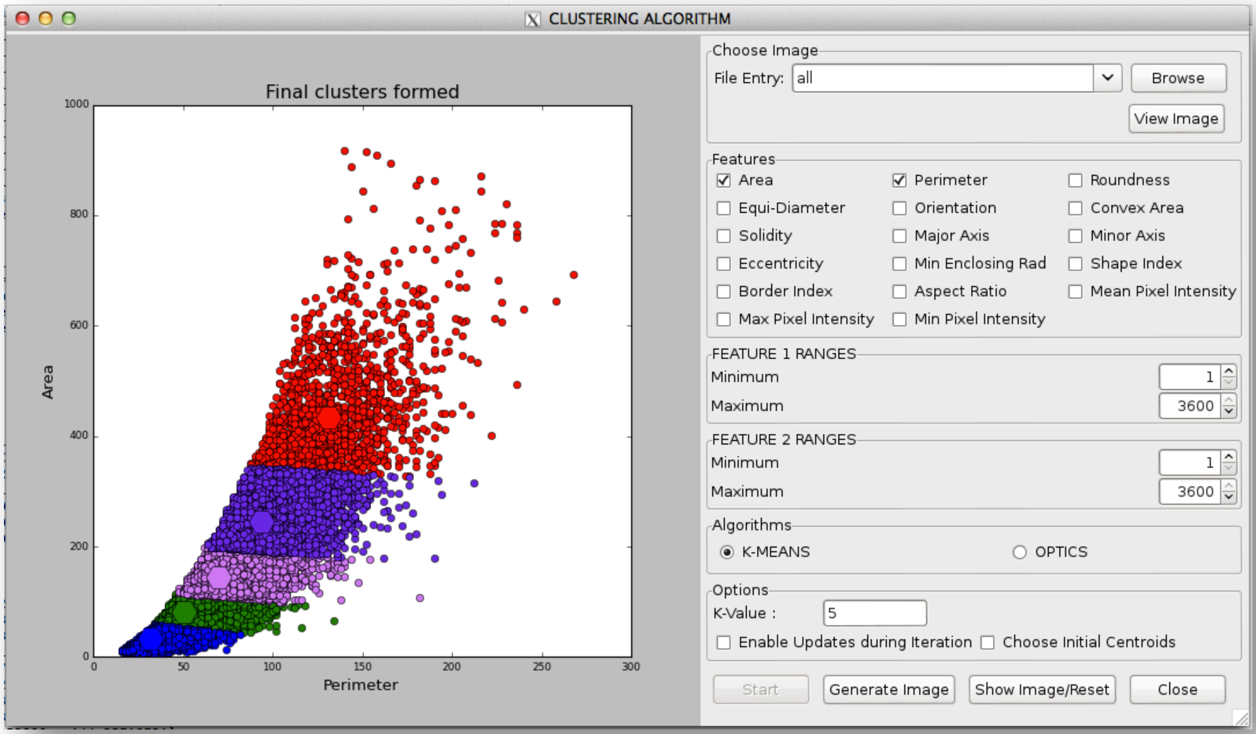
We used Python as the programing language for the development of entire application. Python-OpenCV, which is an external package, is used for feature generation. The obtained features were stored in a database, which in turn used in clustering algorithms, and while plotting our clusters. Other libraries like wx-python, matplotlib are used for UI development. Existing k-means implementation that is defined in Sklearn package was used as primary clustering technique. The GUI was implemented as a multi-threaded process to make the application more responsive. We have used two clustering algorithms to cluster the data – K means algorithm and Optics algorithm.



Use Cases

1. User can select two features and then select:
    - Type of clustering - KMeans or Optics
    - Type of visualization - Iterative or just final output

   A scatterplot showing the resulting clusters will be generated.



Choose Image
File Entry: all

Features

☑ Area	☑ Perimeter	☐ Roundness
☐ Equi-Diameter	☐ Orientation	☐ Convex Area
☐ Solidity	☐ Major Axis	☐ Minor Axis
☐ Eccentricity	☐ Min Enclosing Rad	☐ Shape Index
☐ Border Index	☐ Aspect Ratio	☐ Mean Pixel Intensity
☐ Max Pixel Intensity	☐ Min Pixel Intensity	

FEATURE 1 RANGES
Minimum 1
Maximum 3600

FEATURE 2 RANGES
Minimum 1
Maximum 3600

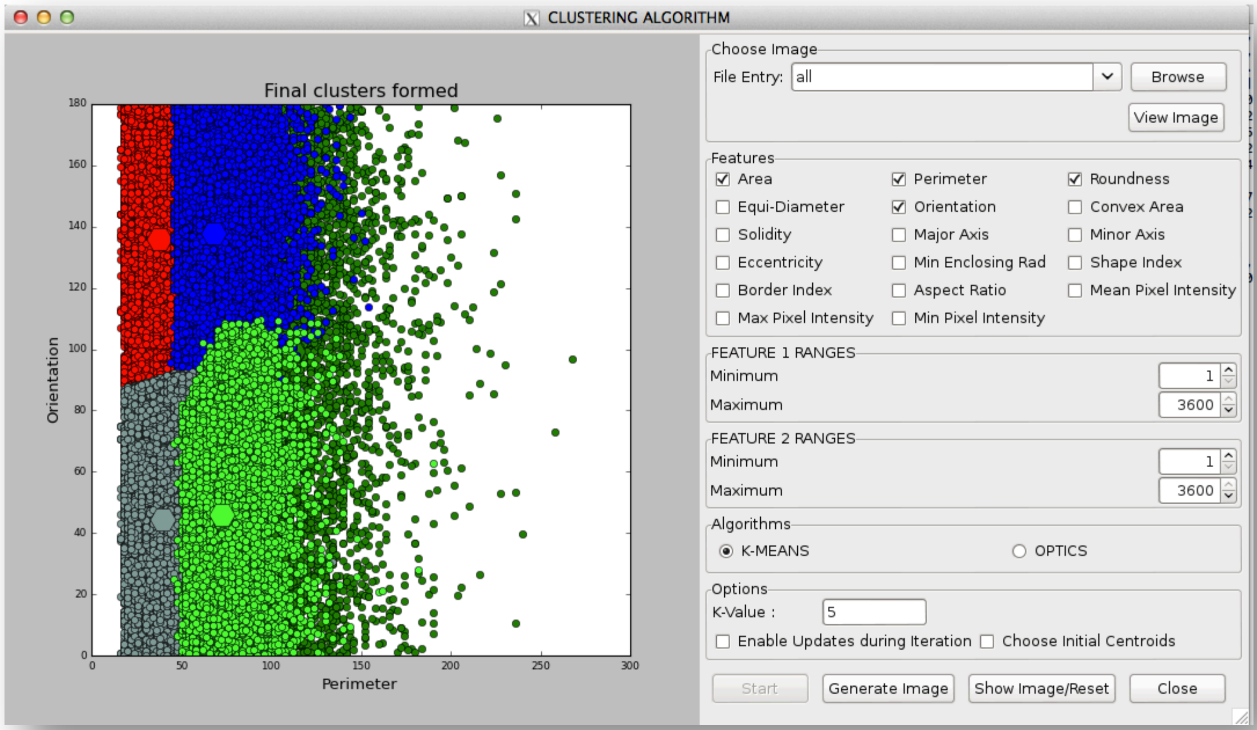
Algorithms
[✓ K-MEANS] OPTICS

Options
K-Value : 5
☐ Enable Updates during Iteration ☐ Choose Initial Centroids

2. User can select more than two features and then select:
    - Type of clustering - KMeans or Optics
    - Type of visualization - Iterative or just final output

   A scatterplot showing the resulting clusters for just two features will be generated and an output file for all ‘n’ features will be generated.

   The below screen shot shows the final clusters.



Choose Image

File Entry: all

Browse

View Image

Features

[✓ Area] [✓ Perimeter] [✓ Roundness]

Equi-Diameter [✓ Orientation] Convex Area

Solidity Major Axis Minor Axis

Eccentricity Min Enclosing Rad Shape Index

Border Index Aspect Ratio Mean Pixel Intensity

Max Pixel Intensity Min Pixel Intensity

FEATURE 1 RANGES

Minimum 1

Maximum 3600

FEATURE 2 RANGES

Minimum 1

Maximum 3600

Algorithms

[✓ K-MEANS] OPTICS

Options

K-Value : 5

Enable Updates during Iteration Choose Initial Centroids

Start Generate Image Show Image/Reset Close

Features

Contours can be explained simply as a curve joining all the continuous points (along the boundary), having same color or intensity. The contours are a useful tool for shape analysis and object detection and recognition.

- Before finding contours we have to apply threshold or canny edge detection.
- findContours function modifies the source image. So if you want source image even after finding contours, already store it to some other variables.
- In OpenCV, finding contours is like finding white object from black background. So object to be found should be white and background should be black.

Features Based on geometry

By using the given coordinates of the polygons (nuclei) we have calculated the below features

1. **Area**: Contour area is the Area bounded by the contour region. Contour area is given by the function cv2.contourArea() or from moments.
2. **Perimeter**: It is also called arc length. It can be found out using cv2.arcLength() function. Second argument specify whether shape is a closed contour (if passed True), or just a curve.
3. **Roundness**: It gives the roundness of each polygon (nuclei).
4. **Equi-Diameter**: Equivalent Diameter is the diameter of the circle whose area is same as the contour area.
5. **Convex Hull**: It is the smallest convex polygon that can be drawn enclosing the nucleus.
6. **Solidity**: Solidity is the ratio of contour area to its convex hull area i.e, it signifies how many pixels are in the convex hull.
7. **Major axis**: It is the length of major axis of the ellipse formed over the contour.
8. **Minor axis**: It is the length of major axis of the ellipse formed over the contour.
9. **Eccentricity**: It is the eccentricity of the ellipse formed.

10. **Border Index**: The Border Index feature describes how jagged an image object is. The more jagged , the higher its border index.
11. **Shape Index** (Smoothness): The Shape index describes the smoothness of an image object border. The smoother the border of an image object is, the lower its shape index.
12. **Orientation of the enclosing ellipse**: Orientation is the angle at which object is directed.
13. **Aspect ratio of bounding rectangle**: The aspect ratio of an image describes the proportional relationship between its width and
14. **Minimum enclosing circle**: It is a circle which completely covers the object with minimum area.

Features Based on Image

And by using the given images we have calculated the below features for the images

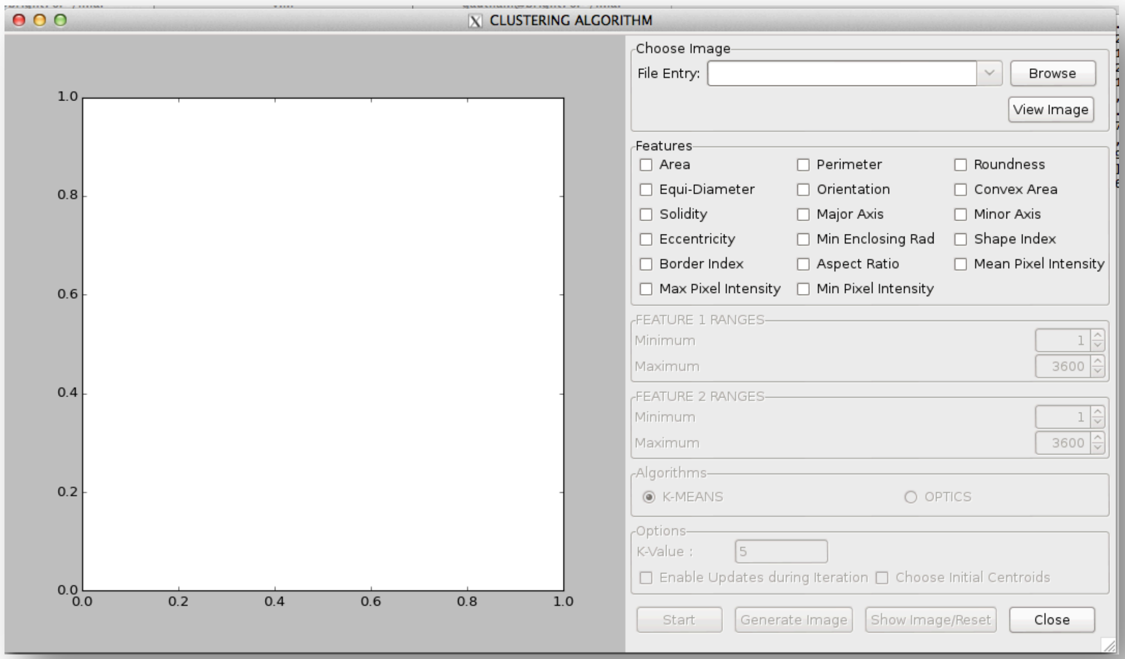
1. **Mean Pixel Intensity** : The mean pixel intensity value of all the pixels in the nucleus.
2. **Max Pixel Intensity** : The value of the pixel with the maximum layer intensity value in the nucleus.
3. **Min Pixel Intensity**: The value of the pixel with the minimum layer intensity value in the image object.

Dataset

We used an image of cancer cell, which had 35 tiles, and each one is a tif file. We identified the polygons and extracted the regions for which the features were calculated. These features were stored in a SQlite database. These data were used for calculating the clusters by using the cluster algorithms and presenting to the user.

Visualization

Below is the screenshot of the application when it is launched. We have the options to select the file and features along with the algorithm to use for clustering on the right hand side and we display the clusters on the left hand side.



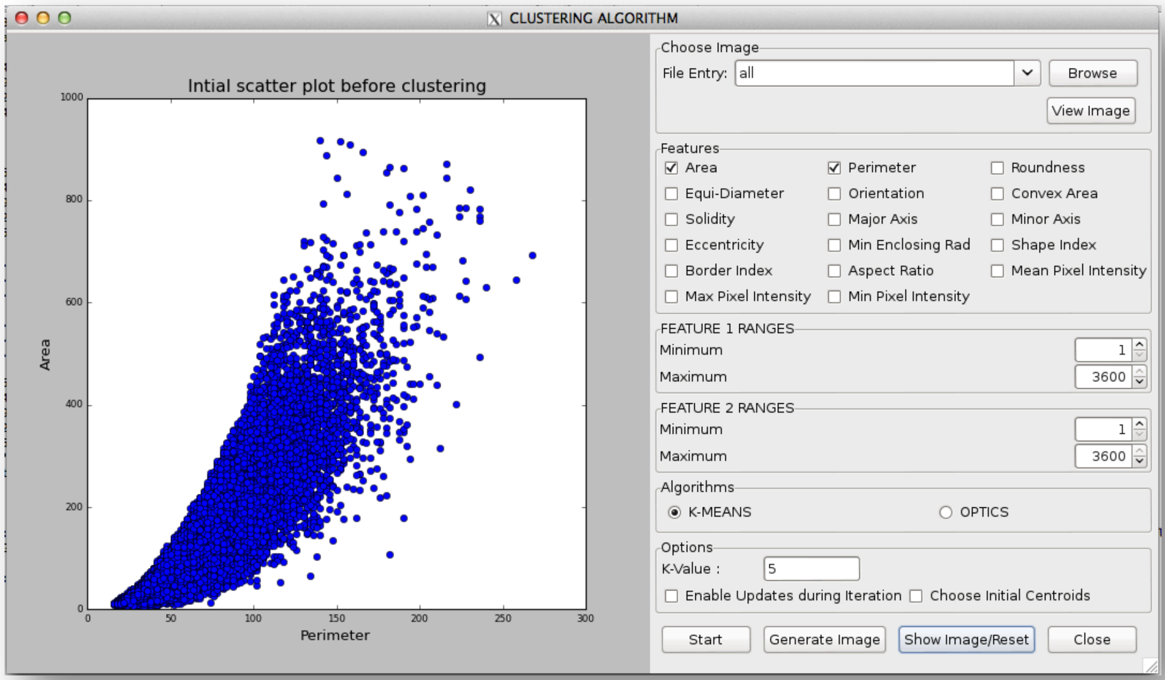
FEATURE 1 RANGES: Minimum 1, Maximum 3600

FEATURE 2 RANGES: Minimum 1, Maximum 3600

Algorithms: [✓ K-MEANS] OPTICS

K-Value : 5

When we give "all" as file name instead of selecting a specific tile of the image, we form the clusters over the data from all the tiles of the image.



Choose Image
File Entry: all

Features
[✓ Area] [✓ Perimeter] Roundness
Equi-Diameter Orientation Convex Area
Solidity Major Axis Minor Axis
Eccentricity Min Enclosing Rad Shape Index
Border Index Aspect Ratio Mean Pixel Intensity
Max Pixel Intensity Min Pixel Intensity

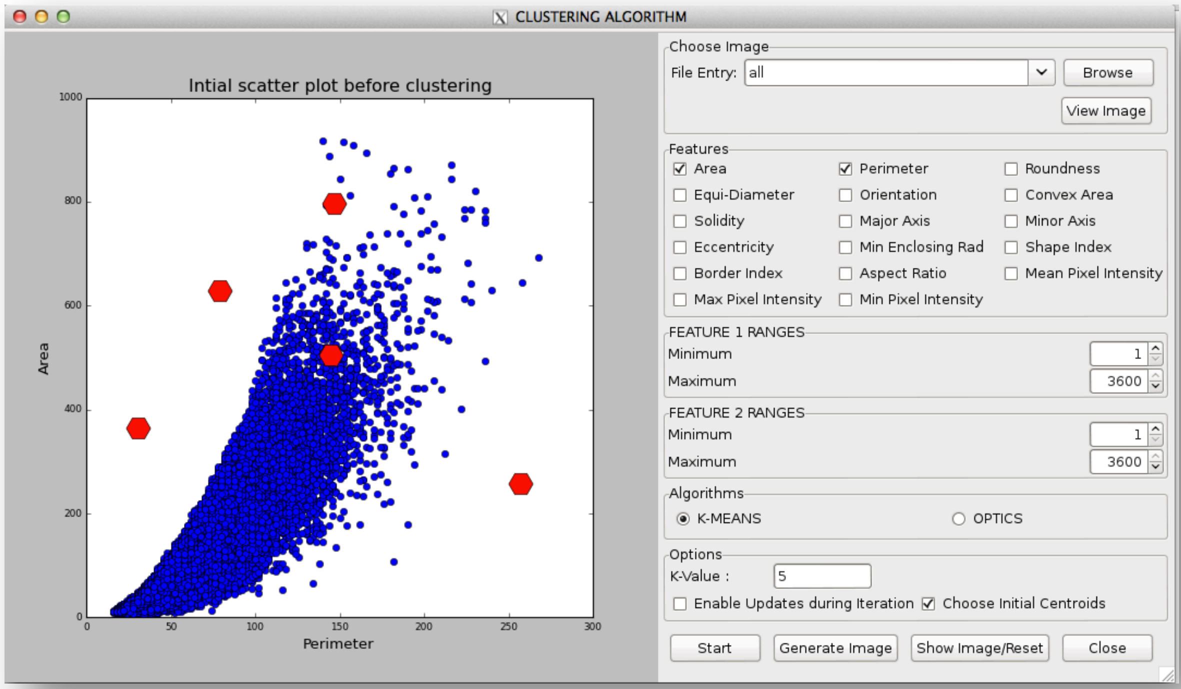
FEATURE 1 RANGES
Minimum 1
Maximum 3600

FEATURE 2 RANGES
Minimum 1
Maximum 3600

Algorithms
[✓ K-MEANS] OPTICS

Options
K-Value : 5
Enable Updates during Iteration Choose Initial Centroids

The user can select the centroids over the plotted nodes for the clustering as below.



Choose Image
File Entry: all

Features
[✓ Area] [✓ Perimeter] Roundness
Equi-Diameter Orientation Convex Area
Solidity Major Axis Minor Axis
Eccentricity Min Enclosing Rad Shape Index
Border Index Aspect Ratio Mean Pixel Intensity
Max Pixel Intensity Min Pixel Intensity

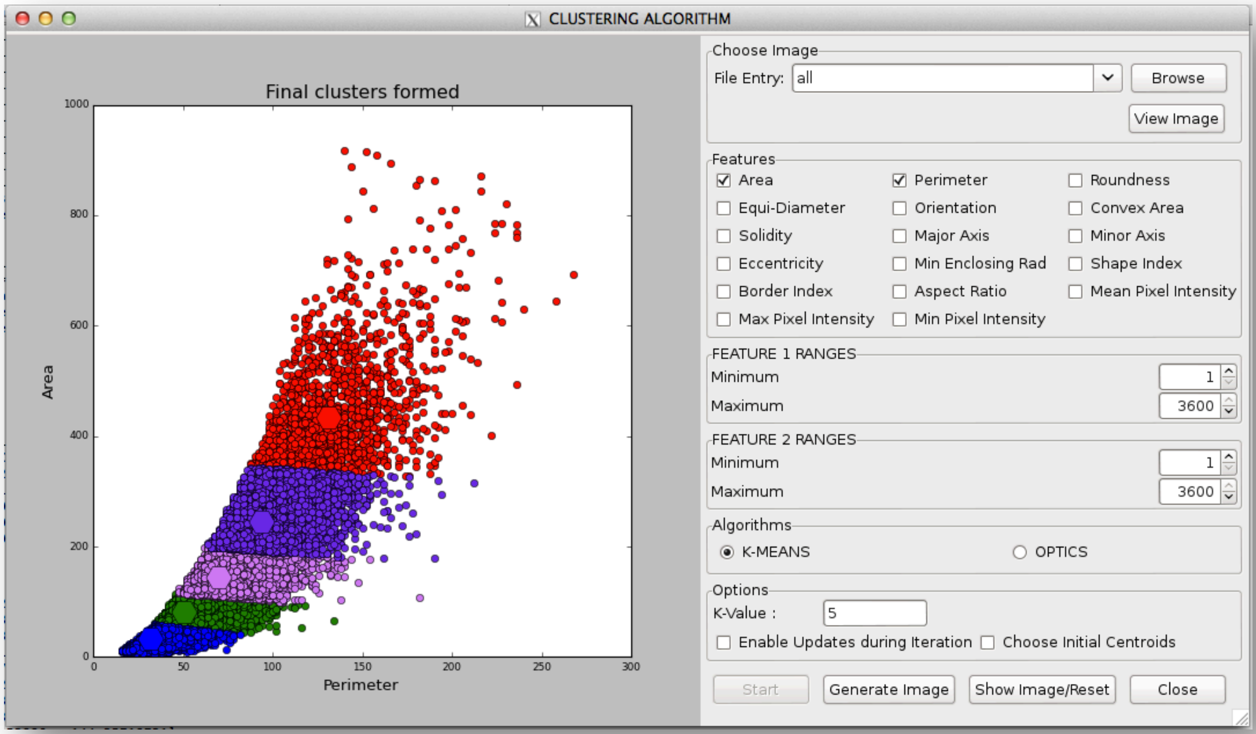
FEATURE 1 RANGES
Minimum 1
Maximum 3600

FEATURE 2 RANGES
Minimum 1
Maximum 3600

Algorithms
[✓ K-MEANS] OPTICS

Options
K-Value : 5
Enable Updates during Iteration [✓ Choose Initial Centroids]

Below is the screenshot of the final clusters formed over the data.



Choose Image

File Entry: all

Features

[✓] Area [✓] Perimeter ☐ Roundness
☐ Equi-Diameter ☐ Orientation ☐ Convex Area
☐ Solidity ☐ Major Axis ☐ Minor Axis
☐ Eccentricity ☐ Min Enclosing Rad ☐ Shape Index
☐ Border Index ☐ Aspect Ratio ☐ Mean Pixel Intensity
☐ Max Pixel Intensity ☐ Min Pixel Intensity

FEATURE 1 RANGES

Minimum	1
Maximum	3600

FEATURE 2 RANGES

Minimum	1
Maximum	3600

Algorithms

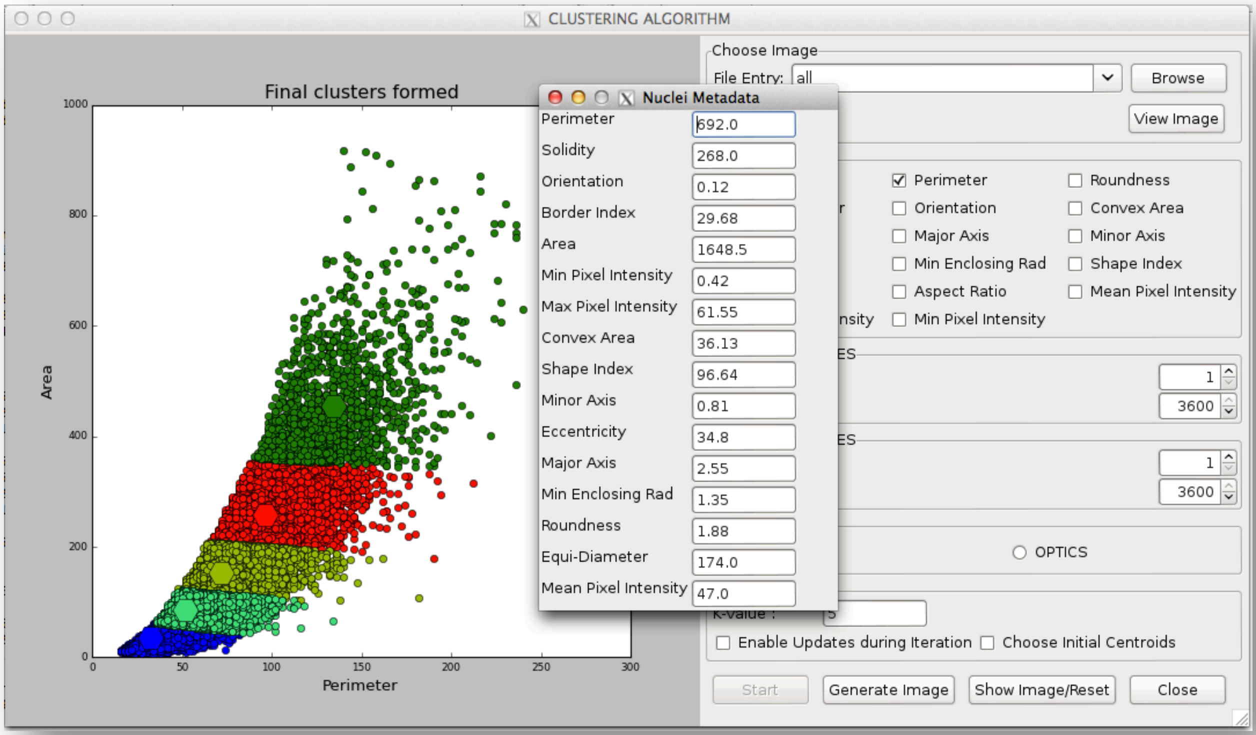
[✓ K-MEANS] OPTICS

Options

K-Value : 5

☐ Enable Updates during Iteration ☐ Choose Initial Centroids

When a nucleus is clicked on the scatter plot its Meta information is displayed to the user.



Choose Image

File Entry: all

Nuclei Metadata

Perimeter	692.0
Solidity	268.0
Orientation	0.12
Border Index	29.68
Area	1648.5
Min Pixel Intensity	0.42
Max Pixel Intensity	61.55
Convex Area	36.13
Shape Index	96.64
Minor Axis	0.81
Eccentricity	34.8
Major Axis	2.55
Min Enclosing Rad	1.35
Roundness	1.88
Equi-Diameter	174.0
Mean Pixel Intensity	47.0

☑ Perimeter ☐ Roundness
☐ Orientation ☐ Convex Area
☐ Major Axis ☐ Minor Axis
☐ Min Enclosing Rad ☐ Shape Index
☐ Aspect Ratio ☐ Mean Pixel Intensity
☐ Min Pixel Intensity

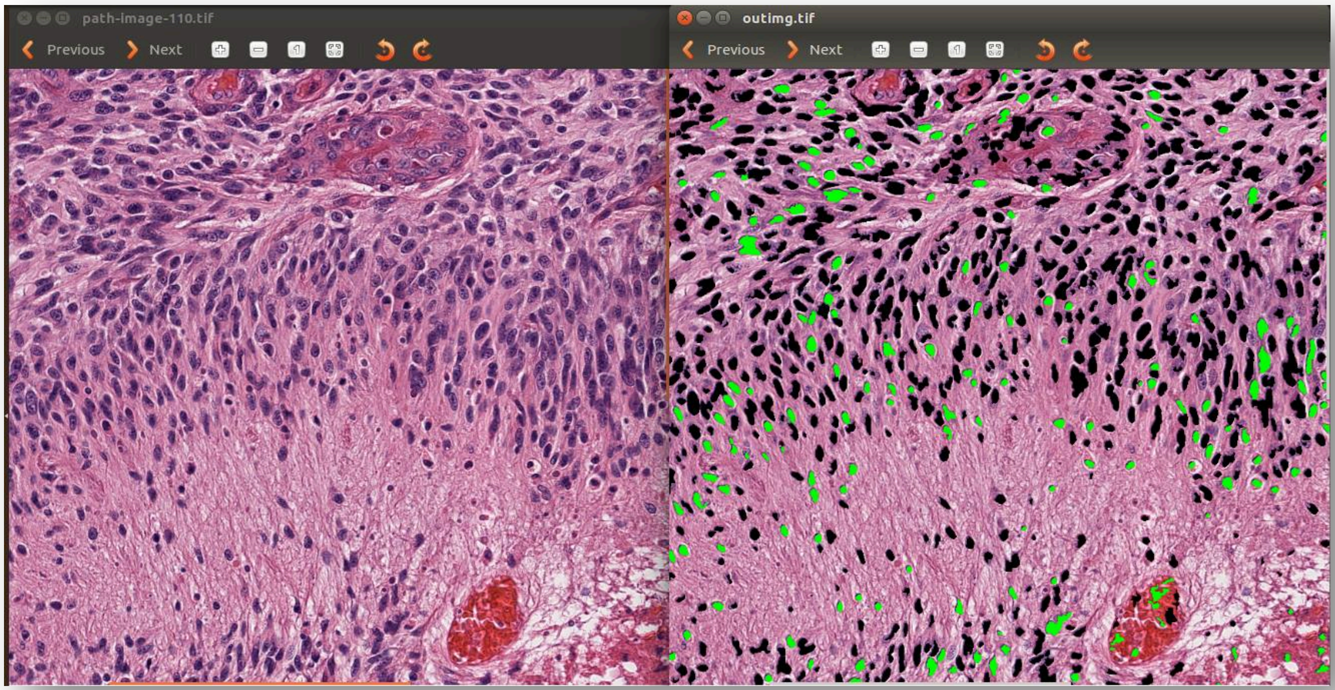
1 / 3600

1 / 3600

K-value : 5

☐ Enable Updates during Iteration ☐ Choose Initial Centroids

Below is the obtained image after clustering on the mean pixel intensity.



Challenges

1) Identifying features and extracting:
   The initial step of the project was to identify the geometric and image features and extract them from the data provided. We needed to identify the features which would help most in studying the nuclei.
2) Creating interactive GUI:
   a. We have designed an interactive GUI to initiate the clustering process and provided different feature like showing the metadata of the nuclei in clusters, allowing users to pick initial centroids for clustering.
   b. Providing the updates during the clustering process was challenging
3) Porting files to the cluster:
   Initially the application was single threaded. Since cluster had more resources ie. more cpu cores and more memory, we had to make the clustering algorithm multi-threaded, there by utilizing all the resources on the cluster.