AUTOMATIC CLASSIFICATION OF CELLS DURING NEURODEVELOPMENT

Sudha Gopalakrishnan Brain Centre, IIT Madras
A Siddharth Reddy[1]
[1] Intern, SGBC, arvindsiddharthreddy@gmail.com

Abstract

The research commenced with the implementation of a watershed-based cell detection algorithm designed to identify cells within specific tiles of JP2 images, focusing on the extraction of regions of interest (ROIs) and refining these selections based on a defined threshold area. This foundational step enabled the generation of data for more complex analyses. The feature vectors extracted from these ROIs were subsequently enhanced with additional attributes. To streamline the clustering process, Principal Component Analysis (PCA) was applied, reducing the feature space to two dimensions. Following this, KMeans clustering and Graph Cut Segmentation were employed to automatically cluster the vectors and identify distinct cell classes. Average feature values for each cluster or segment were then calculated and used to generate synthetic cell images.

The scope of the research was broadened by populating a comprehensive training dataset, involving feature extraction from multiple tiles across a significant portion of the fetal brain image. The relationship between the number of segments (cell classes) and changes in the number of training data vectors was systematically analyzed to identify a point of stability in segment count. Once this stability was established, average feature values for each cluster were recalculated, leading to the creation of a new set of synthetic cell images. An algorithm was developed to extract visible features from these synthetic images, forming the basis for training a semi-supervised model. This model is crucial for validating the accuracy of cell classification and ensuring the biological relevance of the identified cell classes. Additionally, visualization techniques were employed to observe changes across clusters, enhancing the understanding of cell differentiation patterns during neurodevelopment.

1 Introduction

The primary objective of this research is to develop an automated classifier that accurately classifies cells during neurodevelopment within the fetal brain. This process begins with the detection of cells and the extraction of regions of interest (ROIs) using a watershed algorithm, which lays the groundwork for further analysis. The feature vectors from these ROIs are then enhanced with additional attributes. Principal Component Analysis (PCA) is applied to reduce the feature space, facilitating easier clustering. KMeans clustering and Graph Cut Segmentation are employed to identify distinct cell classes based on these refined feature vectors.

This research also focuses on generating synthetic cell images derived from the average feature values of each cluster. A comprehensive training dataset is developed by extracting features from multiple tiles, enabling a robust analysis of segment count stability as the number of training data vectors changes. Finally, an algorithm is created to extract visible features from these synthetic images, which are then used to train a semi-supervised model. Visualization approaches are also integrated to monitor changes across clusters, providing insights into cell differentiation patterns during neurodevelopment. This model is essential for ensuring the biological relevance and accuracy of cell classification, ultimately contributing to a deeper understanding of neurodevelopment.

2 Motivation

The motivation behind this research is to enhance the classification of cells during fetal brain development, a crucial part of neurodevelopmental studies. Accurate classification can reveal important details about cell differentiation and growth patterns, aiding in the understanding of brain disorders and brain abnormalities. Traditional methods are time-consuming and error-prone, highlighting the need for automated techniques. While neuron classification has been attempted [1] in adult brains, this is, to our knowledge, the first time such cell classification has been attempted on a developing human brain.

3 Cell Detection

3.1 Tile Extraction and Preprocessing

3.1.1 Tools Used:

- Python Libraries: **rasterio, cv2 (OpenCV), numpy, matplotlib.**

3.1.2 Steps:

- **Tile Extraction** - Extracted 512x512 pixel tile (Fig. 1) from the original JP2 image using rasterio. Then, converted extracted tile to a format suitable for OpenCV (BGR format).

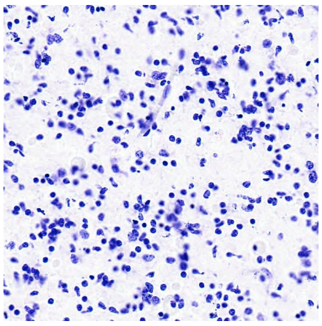


Fig. 1: Extracted 512x512 tile

- **Blue Channel Extraction** - Extracted blue channel from the image tile, as extracting blue channel enhances contrast and improves accuracy of edge detection (Fig. 2).

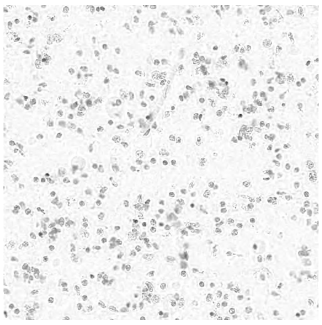


Fig. 2: Blue channel

- **Gaussian Blur** - Applied Gaussian blur to blue channel to reduce noise and smoothen the image. Experimentation revealed that a 9x9 mask was optimal for detecting the maximum number of cells (Fig. 3).

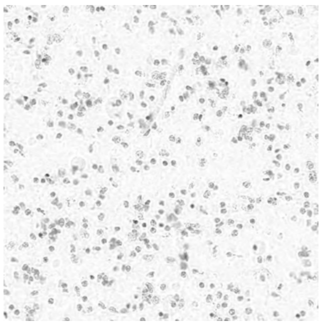


Fig. 3: With 9x9 gaussian blur

- **Edge Detection** - Applied Canny Edge Detection on blurred blue channel to detect edges more accurately (Fig. 4).

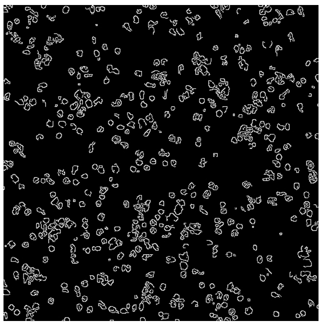


Fig. 4: Canny edge detection

3.2 Image Segmentation

3.2.1 Steps:

- Identified sure background and foreground regions for the Watershed algorithm (Fig. 5).
  1. **Sure Background**: Regions that are definitely back ground.
  2. **Distance Transform**: Calculated distances to the nearest zero pixel.
  3. **Sure Foreground**: Thresholded the distance tansform.
  4. **Unknown Area**: Subtracted sure foreground from sure background.

3.3 Watershed Algorithm

3.3.1 Steps:

- **Labeling** - Labeled the connected components in the sure foreground. Adjusted the label so that the background is not zero. Then, marked the unknown regions with zero.
- **Watershed Application** - Applied the Watershed Algorithm to segment the image into different regions (Fig. 6).

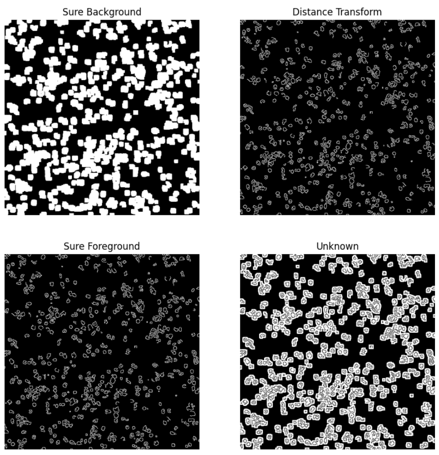


Fig. 5

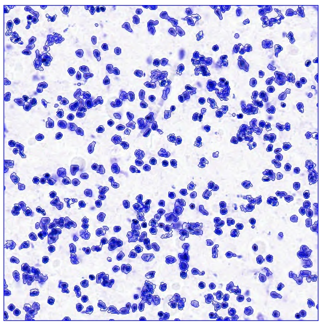


Fig. 6: Marker labeling

4 Results

4.1 ROI Extraction and Display

- Extracted and saved all ROI's.
- Displayed all ROIs in a subplot(Fig. 7).
- Centroids are calculated and displayed for each cell detected in the tile(Fig. 8).
- Total 185 ROIs found (For 512x512 tile).

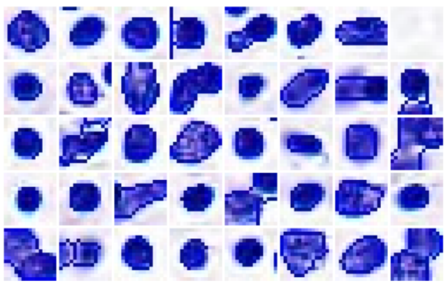


Fig. 7: Detected ROIs (A few...)

4.2 Choosing Threshold

A threshold area of 150 sq units was fixed, upon observing all the ROIs that were extracted so far. Therefore, all those ROIs with area less than 150

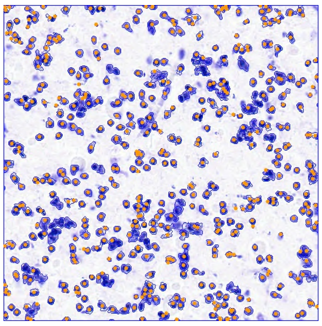


Fig. 8: Tile with centroid markings

units will be extracted. These ROIs will then be used for feature extraction which will also be used as training data for the cell auto-classifier. 180 ROIs below threshold (For the 512x512 tile used here).Example of an extracted ROI is shown in Fig. 9.

Fig. 9: Extracted ROI

5 Summary - Cell Detection

The cell detection algorithm performed so far performs the following tasks:

- Reads and preprocesses a JP2 image.
- Extracts the blue channel and applies Gaussian blur.
- Performs Canny Edge Detection.
- Identifies sure background and foreground areas.
- Applies the watershed algorithm to segment the image.
- Extracts and saves regions of interest (ROIs). Also displays them in subplots.
- Draws contours (for detected objects) and centroids on the original image.
- Extracts specific ROIs based on a fixed threshold.

6 Cell Feature Extraction

Now, for each of the extracted ROIs (the ones below the threshold area), a feature vector is prepared. This vector contains information about various features of a cell such as:

- Aspect Ratio
- Extent
- Solidity
- Contrast
- Local Homogeneity
- Global Homogeneity
- Perimeter
- Minor Axis
- Major Axis
- Equivalent Diameter
- Circularity
- Convexity

6.1 Definitions

6.1.1 Aspect Ratio

- Ratio of the width to the height of the bounding rectangle of the contour.

6.1.2 Extent

- Ratio of the contour area to the area of its bounding rectangle. It measures how much of the bounding rectangle is filled by the contour.

6.1.3 Solidity

- Ratio of the contour area to the area of it's convex hull. The convex hull is the smallest convex shape that can enclose the contour.

6.1.4 Contrast

- Calculated as the square of the variance of the greyscale pixel values within the contour. Indicates the level of detail or texture.

6.1.5 Local Homogeneity

- Measures the similarity of pixel values within a region (Inside the cell). Calculated using the gray-level co-occurrence matrix (GLCM) and derived from this matrix using the **greycoprops(glcm, 'homogeneity')** function.

6.1.6 Global Homogeneity

- Represents the overall uniformity of pixel values in an image. Lower entropy values indicate higher homogeneity.

6.1.7 Perimeter

- Total length of the contour of the cell. It is calculated using the **cv2.arcLength** function, which takes the contour and a flag indicating whether the contour is closed (True) or open (False).

6.1.8 Minor Axis

- Shorter side of the bounding rectangle around the contour. It is derived from the dimensions of the bounding rectangle (width w and height h).

6.1.9 Major Axis

- Longer side of the bounding rectangle around the contour. It is derived from the dimensions of the bounding rectangle (width w and height h).

6.1.10 Equivalent Diameter

- Diameter of a circle that has the same area as the contour.

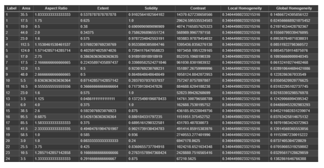
6.1.11 Circularity

- Measure of how close the shape of the contour is to a perfect circle. A perfect circle has a circularity value of 1, while other shapes have values less than 1.

6.1.12 Convexity

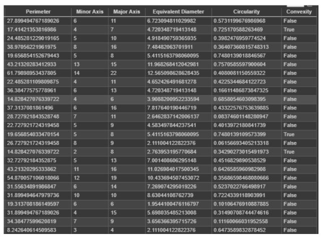
- Binary measure that indicates whether the contour is convex or not. Determined using the **cv2.isContourConvex** function, which returns True if the contour is convex and False otherwise.

6.2 Results



Label	Area	Aspect Ratio	Extent	Solidity	Contrast	Local Homogeneity	Global Homogeneity
1	35.5	1.8333333333333333	0.5378787878787878	0.9102564102564102	14370.627238650566	0.34044980233215316	0.14866724305856122
1	17.5	1.75	0.625	1.0	29624.595535534503	0.34044980233215316	0.024566668921075452
1	19.0	0.5	0.38	0.6909090909090909	4074.7165857625323	0.34044980233215316	0.21974534428792367
2	44.0	2.0	0.34375	0.7586206896551724	568989.9967797158	0.34044980233215316	0.15560799339476095
3	23.0	1.6	0.575	0.9787234042553191	103883.97879454032	0.34044980233215316	0.09538764071838031
4	112.5	1.1538461538461537	0.5769230769230769	0.9533898305084746	1395436.0356276138	0.34044980233215316	0.08511922798360572
5	124.0	1.5714285714285714	0.4025974025974026	0.7294117647058823	1073450.1951229165	0.34044980233215316	0.08545759114875076
6	17.0	2.75	0.38636363636363635	0.918918918918919	852315.1968190339	0.34044980233215316	0.10936660000040345
7	17.5	2.1666666666666665	0.22435897435897437	0.3398058252427l846	961030.8301983932	0.34044980233215316	0.06132410274402468
8	12.0	1.5	0.5	0.9230769230769231	151897.26759999996	0.34044980233215316	0.028910644694421086
9	48.0	2.6666666666666665	0.5	0.8648648648648649	1058124.8842072953	0.34044980233215316	0.1228286367033549
10	5.5	0.6363636363636364	0.07142857142857142	0.2037037037037037	757247.0757897807	0.34044980233215316	0.03505620920776625
11	16.5	0.5555555555555556	0.36666666666666664	0.717391304347826	884688.6204188238	0.34044980233215316	0.03182295102737745
12	23.0	1.6	0.575	1.0	52823.99426268699	0.34044980233215316	0.02183393298576978
13	3.5	1.125	0.04861111111111111	0.13725490196078433	14761.300796508789	0.34044980233215316	0.12420066994395342
14	6.0	4.0	0.375	1.0	162688.7530195752	0.34044980233215316	0.044869453202983293
15	38.5	2.6	0.5923076923076923	0.875	436185.05228049756	0.34044980233215316	0.045211602831239074
16	95.5	0.6875	0.5426136363636364	0.880184331797235	1151051.375452752	0.34044980233215316	0.03763425814675132
17	85.5	1.5833333333333333	0.375	0.6895161290322581	431703.407830073	0.34044980233215316	0.10701983424372963
18	41.5	2.3333333333333335	0.49404761904761907	0.9021739130434783	491414.85915383976	0.34044980233215316	0.12014560365553056
19	58.5	1.0	0.585	0.936	2740553.277401996	0.34044980233215316	0.11153967230015222
20	3.0	1.0	0.08333333333333333	0.24	6947178.0625	0.34044980233215316	0.0737405304728029
22	25.5	3.75	0.425	0.8360655737704918	1024218.6521634348	0.34044980233215316	0.07059085319259802
23	10.5	1.2857142857142858	0.16666666666666666	0.27631578947368424	2428880.7516565416	0.34044980233215316	0.03295617128629859
24	3.5	1.3333333333333333	0.2916666666666667	0.875	67210.5625	0.34044980233215316	0.1382861646766388

Fig. 10: Extracted features



Perimeter	Minor Axis	Major Axis	Equivalent Diameter	Circularity	Convexity
27.899494767189026	6	11	6.723094811029982	0.5731199676986968	False
17.41421353816986	4	7	4.720348719413148	0.725170588263469	True
24.485281229019165	5	10	4.918490759365935	0.3982476959774524	False
38.97056221961975	8	16	7.48482063701911	0.36407360815748313	False
19.656854152679443	5	8	5.4115163798060095	0.7480139018846567	False
43.21320283412933	13	15	11.968268412042981	0.7570585597900604	False
61.79898953437805	14	22	12.565098628628435	0.4080081150559322	False
22.485281109809875	4	11	4.652426491681278	0.4225344684122723	False
36.38477575778961	6	13	4.720348719413148	0.16611486873847325	False
14.828427076339722	4	6	3.9088200952233594	0.6858054603098395	False
37.3137081861496	6	16	7.817640190446719	0.43322576753639885	False
28.727921843528748	7	11	2.6462837142006137	0.08374601148280947	False
22.727921724319458	5	9	4.583497844237541	0.4013972180841739	False
19.656854033470154	5	8	5.4115163798060095	0.7480139109573399	True
26.727921724319458	8	9	2.111004122822376	0.06156693405213318	False
14.828427076339722	2	8	2.763953195770684	0.34290273015491973	True
32.72792184352875	5	13	7.001408606295148	0.4516829890538529	False
43.21320295333862	11	16	11.026984017500345	0.6426585960982908	False
54.870057106018066	12	19	10.433694507453072	0.35686598468080666	False
31.55634891986847	6	14	7.269074295019226	0.5237022766498917	False
31.899494647979736	10	10	8.63044108762739	0.7224339118903991	False
19.313708186149597	6	6	1.9544100476116797	0.10106476910887885	False
31.899494767189026	4	15	5.698035485213008	0.31490708744474616	False
34.38477599620819	7	9	3.656366395715726	0.11160066031952558	False
8.242640614509583	3	4	2.111004122822376	0.6473589832878452	False

Fig. 11: Extracted features (Contd.)

7 KMeans Algorithm

The KMeans [2] algorithm is a clustering technique used to partition a dataset into K distinct, non-overlapping subsets (clusters). The algorithm iterates to find cluster centroids that minimize the within-cluster variance

7.1 Algorithm

7.1.1 Initialization

- Select K initial centroids randomly from the dataset.

7.1.2 Assignment Step

- Assign each data point to the nearest centroid, creating K clusters.

7.1.3 Update Step

- Calculate new centroids as the mean of all data points assigned to each cluster.

7.1.4 Repeat

- Repeat the assignment and update steps until the centroids no longer change or a maximum number of iterations is reached.

7.2 Implementation

7.2.1 PCA for Dimentionality Reduction

- Before applying KMeans, PCA was used to reduce the feature space to 2 dimensions for easier clustering.

7.2.2 KMeans Clustering

- We initialized the KMeans algorithm with 6 clusters.

- The the KMeans algorithm was applied to the PCA - reduced data and each data point was assigned to one of the 6 clusters. The cluster assignment was added as a new column 'Cluster' to the DataFrame.

7.2.3 Calculate Average Features

- The features used for clustering are then averaged within each cluster to obtain a representative feature vector for each cluster.

7.3 Results

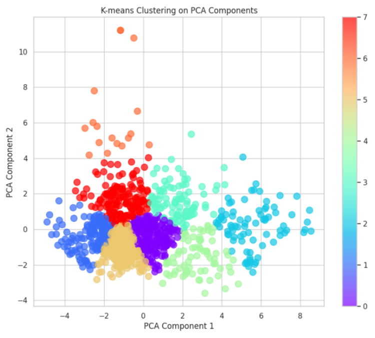
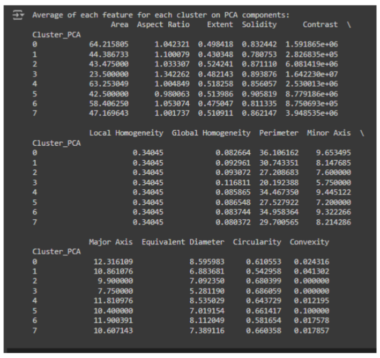


Fig. 12: KMeans clustering on PCA components



```
Average of each feature for each cluster on PCA components:
                 Area  Aspect Ratio    Extent  Solidity      Contrast  \
Cluster_PCA
0           64.215805      1.042321  0.498418  0.832442  1.591865e+06
1           44.386733      1.100079  0.430348  0.780753  2.826835e+05
2           43.475000      1.033307  0.524241  0.871110  6.081419e+06
3           23.500000      1.342262  0.482143  0.893876  1.642230e+07
4           63.253049      1.004849  0.518258  0.856057  2.530013e+06
5           42.500000      0.980063  0.513986  0.905819  8.779186e+06
6           58.406250      1.053074  0.475047  0.811335  8.750693e+05
7           47.169643      1.001737  0.510911  0.862147  3.948535e+06

            Local Homogeneity  Global Homogeneity  Perimeter  Minor Axis  \
Cluster_PCA
0                     0.34045            0.082664  36.106162    9.653495
1                     0.34045            0.092961  30.743351    8.147685
2                     0.34045            0.093072  27.208683    7.600000
3                     0.34045            0.116811  20.192388    5.750000
4                     0.34045            0.085865  34.467350    9.445122
5                     0.34045            0.086548  27.527922    7.200000
6                     0.34045            0.083744  34.958364    9.322266
7                     0.34045            0.080372  29.700565    8.214286

            Major Axis  Equivalent Diameter  Circularity  Convexity
Cluster_PCA
0            12.316109             8.595983     0.610553   0.024316
1            10.861076             6.883681     0.542958   0.041302
2             9.900000             7.092350     0.680399   0.000000
3             7.750000             5.281190     0.686059   0.000000
4            11.810976             8.535029     0.643729   0.012195
5            10.400000             7.019154     0.661417   0.100000
6            11.900391             8.112049     0.581654   0.017578
7            10.607143             7.389116     0.660358   0.017857
```

Fig. 13: Average of each feature in each cluster

8 Graph Cut Segmentation

The graph cut segmentation [3] algorithm first constructs a graph where each pixel (or data point) is a node, and edges represent the similarity between neighboring pixels. It then applies label propagation to detect communities within this graph, effectively grouping similar pixels into segments. Finally, it assigns a unique label to each segment, ensuring the total number of segments does not exceed a specified limit. Overall,this function leverages graph-based methods and community detection to segment data effectively. It provides a flexible and efficient approach to identifying distinct regions or clusters in a dataset.

8.1 Algorithm

8.1.1 Graph Construction

- Construct a k-nearest neighbors graph from the input data, where nodes represent pixels and edges represent similarity between neighbors.

8.1.2 Community Detection

- Apply the label propagation algorithm to detect communities (segments) within the graph.

8.1.3 Segment Limitation

- Ensure the number of detected segments does not exceed the specified maximum number.

8.1.4 Label Assignment

- Assign a unique label to each segment based on the detected communities.

8.1.5 Output Labels

- Return the array of segment labels for each pixel in the input data.

8.2 Results

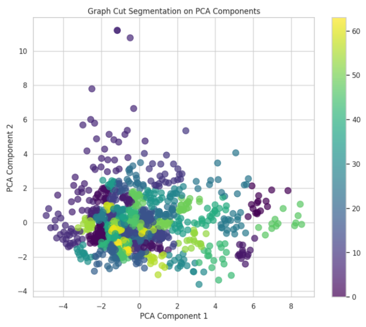
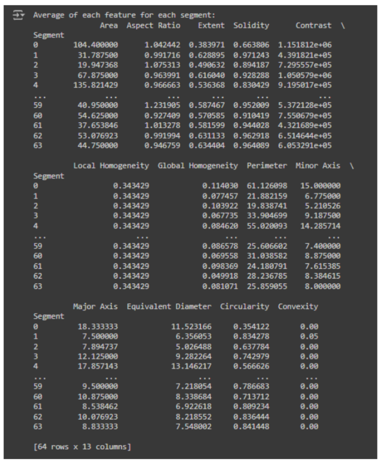


Fig. 14: Graph Cut Segmentation on PCA components



```
Average of each feature for each segment:
               Area  Aspect Ratio    Extent  Solidity      Contrast  \
Segment
0        104.400000      1.042442  0.383971  0.663806  1.151812e+06
1         31.787500      0.991716  0.628895  0.971243  4.391821e+05
2         19.947368      1.075313  0.490632  0.894187  7.295557e+05
3         67.875000      0.963991  0.616040  0.928288  1.050579e+06
4        135.821429      0.966663  0.536368  0.830429  9.195017e+05
...             ...           ...       ...       ...           ...
59        40.950000      1.231905  0.587467  0.952009  5.372128e+05
60        54.625000      0.927409  0.570585  0.910419  7.550679e+05
61        37.653846      1.013278  0.581599  0.944028  4.321689e+05
62        53.076923      0.991994  0.631133  0.962918  6.514644e+05
63        44.750000      0.946759  0.634404  0.964089  6.053291e+05

         Local Homogeneity  Global Homogeneity  Perimeter  Minor Axis  \
Segment
0                 0.343429            0.114030  61.126098   15.000000
1                 0.343429            0.077457  21.882159    6.775000
2                 0.343429            0.103922  19.838741    5.210526
3                 0.343429            0.067735  33.904699    9.187500
4                 0.343429            0.084620  55.020093   14.285714
...                    ...                 ...        ...         ...
59                0.343429            0.086578  25.606602    7.400000
60                0.343429            0.069558  31.038582    8.875000
61                0.343429            0.098369  24.180791    7.615385
62                0.343429            0.049918  28.236785    8.384615
63                0.343429            0.081071  25.859055    8.000000

         Major Axis  Equivalent Diameter  Circularity  Convexity
Segment
0         18.333333            11.523166     0.354122       0.00
1          7.500000             6.356053     0.834278       0.05
2          7.894737             5.026488     0.637784       0.00
3         12.125000             9.282264     0.742979       0.00
4         17.857143            13.146217     0.566626       0.00
...             ...                  ...          ...        ...
59         9.500000             7.218054     0.786683       0.00
60        10.875000             8.338684     0.713712       0.00
61         8.538462             6.922618     0.809234       0.00
62        10.076923             8.218552     0.836444       0.00
63         8.833333             7.548002     0.841448       0.00

[64 rows x 13 columns]
```

Fig. 15: Average of each feature in each segment

9 Synthetic Generation of Cell Images

9.1 Algorithm

The algorithm for generating synthetic cell images based on average features consists of several key steps. These steps are designed to create representative images that encapsulate the characteristic properties of clusters identified in the PCA - transformed feature space. Here's a detailed explanation:

9.1.1 Image Initialization

- The algorithm starts by creating a blank image (img) and a mask (mask) of a specified size (e.g., 20x20 pixels). These arrays are initialized to zeros, which represents the background.

9.1.2 Drawing the Cell Shape

- The center of the image is calculated to position the synthetic cell in the middle of the image.

- The major and minor axis lengths of the cell are derived from the average feature values of the cluster. These axes determine the shape and size of the ellipse that represents the cell.

- The **cv2.ellipse** function is used to draw the ellipse on the mask, which fills the cell shape with a white color (255). This mask is used to delineate the cell region from the background.

9.1.3 Assigning Gray Values

- The pixel values within the cell area (as defined by the mask) are set to a mid-gray value (128). This uniform color represents the baseline intensity of the cell.

9.1.4 Adding Texture (Homogeneity)

- To simulate local homogeneity within the cell, Gaussian noise is added to the cell area. The standard deviation of this noise is proportional to the Local Homogeneity feature, allowing the texture to reflect the variability observed in real cells.

- The **cv2.addWeighted** function blends the noise with the base cell intensity to create a realistic texture while ensuring that the background remains black.

9.1.5 Generating and Displaying Images

- For each cluster, a synthetic image is generated using the average features.
- The synthetic image is saved to disk and displayed in a grid layout using Matplotlib for visual inspection.

9.2 Results

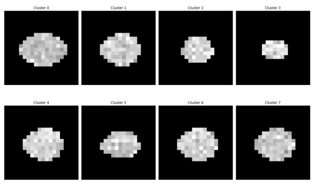


Fig. 16: Based on KMeans cluster data

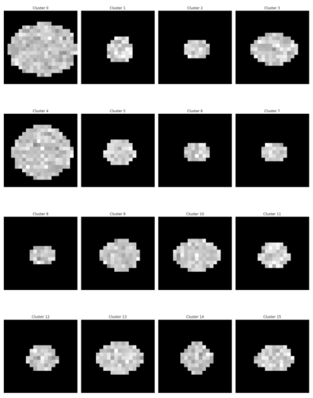


Fig. 17: Based on Graph Cut segment data

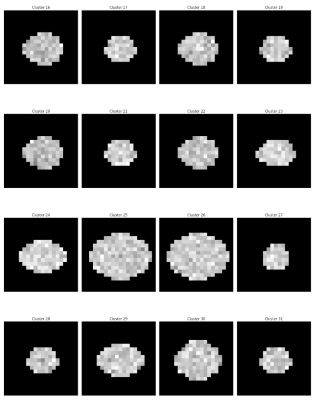


Fig. 18: Based on Graph Cut segment data (Contd.)

10 Number of Cell Classes

The objective here was to identify the number of cell classes present in the fetal brain image, which depends on the number of segments detected. However, due to variations in the number of segments with changes in the training data size, accurately estimating the number of cell classes is challenging. By plotting the changes in training data size against the detected number of segments, a discernible trend emerged, allowing for an approximate determination of the number of cell classes.

No. of Vectors	Segments
1093	71
1289	77
2798	116
5166	108
7994	100
13106	105
23536	105
54297	108
83332	105
100451	105

Table 1: Number of Vectors vs. Segments

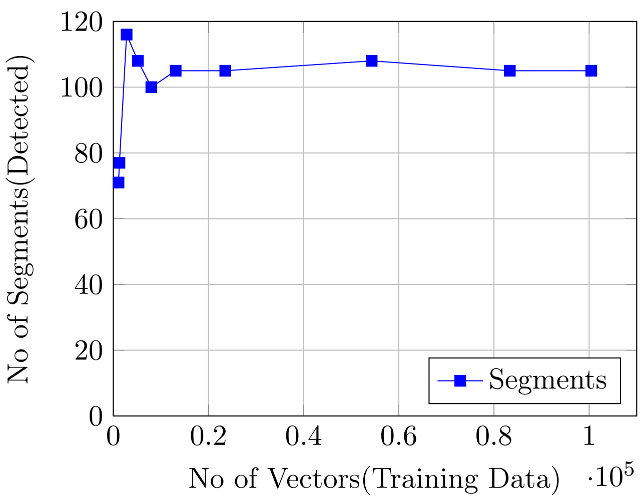


Fig. 19: Number of Vectors vs. Segments

Based on the graphical analysis and tabular representation, it can be concluded that the number of cell classes in the fetal brain image falls within the range of 100 to 108, with 105 being the most prominent estimate.

11 Distinctive Features of Cell Classes

We aim to explore the distinguishing characteristics of segments or cell classes by extracting specific visible features. The features chosen are as follows:

- **Area**: The number of pixels in the region

- **Perimeter**: The length of the boundary of the region.
- **Aspect Ratio**: The ratio of the major axis length to the minor axis length of the region. It describes the elongation of the region.
- **Extent**: The ratio of the area of the region to the area of the bounding box. It describes how much of the bounding box is filled by the region.
- **Solidity**: The ratio of the area of the region to the area of the convex hull. It describes how convex the region is.
- **Mean Intensity**: The average intensity of the pixels in the region.
- **Std Intensity**: The standard deviation of the intensity values of the pixels in the region.
- **Contrast**: A measure of the local variations in the gray-level co-occurrence matrix (GLCM).
- **Dissimilarity**: A measure of how different the elements of the GLCM are from each other
- **Homogeneity**: A measure of how similar the elements of the GLCM are to each other.
- **ASM (Angular Second Moment)**: Also known as energy, it measures the uniformity of the GLCM.
- **Energy**: The sum of squared elements in the GLCM, which is another measure of texture uniformity.

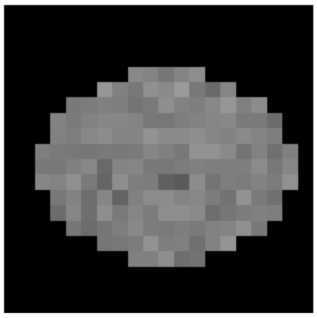
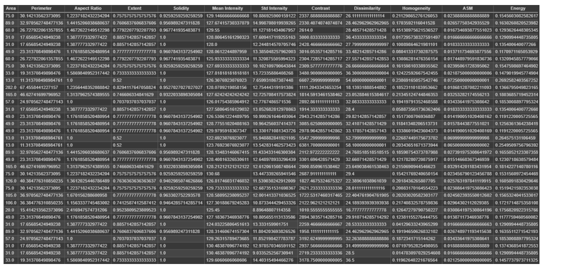


Fig. 20: Synthetically generated cell

Such a feature extraction was performed on a single synthetically generated image (Fig. 20) and the results were as follows:

- Area : 165
- Perimeter : 46.62741699796952
- Aspect Ratio : 1.3137962574389555
- Extent : 0.746606334841629
- Solidity : 0.9322033898305084
- Mean Intensity : 127.56363636363636
- Std Intensity : 11.018601852179772
- Contrast : 1710.3509615384617
- Dissimilarity : 20.418269230769234
- Homogeneity : 0.24878671916240513
- ASM : 0.03290264423076923
- Energy : 0.18139086603837835

We then applied this feature extraction algorithm to all the 105 synthetically generated images. Upon completion, a CSV file, as shown below, was compiled (Fig. 21).



Area	Perimeter	Aspect Ratio	Extent	Solidity	Mean Intensity	Std Intensity	Contrast	Dissimilarity	Homogeneity	ASM	Energy
75.0	30.14213562373095	1.2237182432234204	0.7575757575757576	0.9259259259259259	129.14666666666668	10.886925900159122	2337.8888888888887	26.111111111111114	0.21129865276126053	0.02388888888888889	0.1545603082582617
89.0	32.970562748477136	1.4415206038680637	0.7606837606837606	0.956989247311828	127.67415730337079	14.998708019939265	2330.4074074074074	28.462962962962965	0.1703592116041528	0.02657750342935529	0.1630260820523982
60.0	26.727922061357855	1.4672622149512298	0.7792207792207793	0.967741935483871	129.55	11.127181434067957	2614.0	28.48571428571428	0.15138975621536527	0.016734693877551023	0.1293626448305345
31.0	17.65685424949238	1.387777332977422	0.8857142857142857	1.0	128.8064516129032 3	17.609417119255163	3000.833333333333	36.43333333333334	0.017295738327591497	0.016666666666666666	0.1290994448735805
31.0	17.65685424949238	1.387777332977422	0.8857142857142857	1.0	128.0	12.244814570705746	2428.4666666666662	27.799999999999997	0.10089864621981101	0.01833333333333333	0.13540064077266
49.0	23.31370849898476	1.1761858520480954	0.7777777777777778	0.9607843137254902	128.0612244897959	13.385048257962003	3016.0535714285716	33.48214285714286	0.08041331730287575	0.013711734693877556	0.1170971165053929
60.0	26.727922061357855	1.4672622149512298	0.7792207792207793	0.967741935483871	125.93333333333334	11.320875505994323	2304.7285714285717	27.557142857142853	0.13086281476356154	0.017448979591836736	0.1320945857779066
75.0	30.14213562373095	1.2237182432234204	0.7575757575757576	0.9259259259259259	127.69333333333333	10.102109790643844	2309.5777777777776	26.666666666666664	0.16159810338935562	0.02395061728395062	0.1547598697464902
33.0	19.31370849898476	1.5869840952317442	0.7333333333333333	1.0	127.81818181818181	13.72335886408268	3480.9000000000005	36.300000000000004	0.12422592667542455	0.021875000000000006	0.1479019945774904
13.0	11.313708498984761	1.0	0.52	1.0	126.3076923076923	7.659931867387448	6607.299999999999	54.60000000000001	0.23869165057542746	0.07250000000000001	0.2692582403567252
282.0	67.4558441227157	1.2356448352088842	0.8294117647058824	0.9527027027027027	128.0709219858156	12.75444319191386	1111.2043343653254	18.13931888544892	0.18523110183953662	0.018681287082211093	0.1366795049823165
165.0	46.62741699796952	1.3137962574389555	0.746606334841629	0.9322033898305084	127.62424242424242	12.725708417573824	1814.5913461538462	23.052884615384617	0.21345124744846353	0.0325328217455621 3	0.1803685719452314
57.0	24.970562748477143	1.0	0.7037037037037037	1.0	126.01754385964912	12.7767486571536	2892.8611111111113	32.08333333333333	0.19419791352468588	0.0343364197530864 2	0.1853008897795324
31.0	17.65685424949238	1.387777332977422	0.8857142857142857	1.0	127.5806451612903 2	13.052682812970863	1914.333333333333	28.4	0.058873561736362406	0.018333333333333333	0.1354006400772660
49.0	23.31370849898476	1.1761858520480954	0.7777777777777778	0.9607843137254902	126.53061224489795	10.999261646493064	2943.214285714286	29.82142857142857	0.1517300706936887	0.014190051020408162	0.1191220005725565
49.0	23.31370849898476	1.1761858520480954	0.7777777777777778	0.9607843137254902	128.7755102040816 3	10.964256037414371	3085.6250000000005	32.41071428571429	0.11841340206513731	0.01578443877551021	0.1256361364238419
49.0	23.31370849898476	1.1761858520480954	0.7777777777777778	0.9607843137254902	129.9795918367347	13.330171081343726	2978.9642857142862	33.17857142857143	0.13386019423604373	0.014190051020408169	0.1191220005725565
13.0	11.313708498984761	1.0	0.52	1.0	122.6923076923077	15.948882841921105	5547.299999999998	52.79999999999999	0.22687449175673702	0.06999999999999998	0.264575131106459
13.0	11.313708498984761	1.0	0.52	1.0	123.7692307692307 7	13.542831462573423	6381.700000000001	58.10000000000001	0.20343651671373944	0.06500000000000002	0.2549509756796392
89.0	32.970562748477136	1.4415206038680637	0.7606837606837606	0.956989247311828	128.13483146067415	11.433433146300384	2112.972222222222	24.768518518518515	0.18590754373787988	0.027391975308641972	0.1655052123307359
49.0	23.31370849898476	1.1761858520480954	0.7777777777777778	0.9607843137254902	128.4081632653061 1	12.648978933296439	3301.696428571429	32.66071428571429	0.12178280720875917	0.01514668367346939	0.1230718638579484
165.0	46.62741699796952	1.3137962574389555	0.746606334841629	0.9322033898305084	128.21212121212122	12.612061508748644	2000.850961538462	23.649038461538463	0.2150690944546623	0.03291420118343195 4	0.1814227140780116
75.0	30.14213562373095	1.2237182432234204	0.7575757575757576	0.9259259259259259	130.68	13.447339265941546	2687.911111111111	29.4	0.15421769240650154	0.023456790123456788	0.1531560972454469
126.0	40.384776310850235	1.3612025446786489	0.7636363636363637	0.9402985074626866	126.81746031746032	11.539030242912009	1827.4675324675327	22.38961038961039	0.2018436265807705	0.025763197841119915	0.1605091830429646
75.0	30.14213562373095	1.2237182432234204	0.7575757575757576	0.9259259259259259	129.73333333333332	12.687351531698367	2621.2333333333336	28.011111111111116	0.20083170164554223	0.023086419753086423	0.1519421592353038
105.0	36.970562748477136	1.670156288600958	0.7777777777777778	0.963302752293578	128.58095238095237	12.001413371036525	1722.5317460317465	22.404761904761905	0.20203020582303177	0.024502393550012602	0.1565324041533017
106.0	36.384776310850235	1.1563337745483002	0.7412587412587412	0.9464285714285714	127.30188679245283	10.873344429453326	2122.962121212121	24.189393939393938	0.21740832578759836	0.02964302112029385	0.1721714875358108
20.0	15.414213562373096	2.416947574731206	0.9523809523809523	1.0	125.45	8.89648807114358	1010.5555555555555	16.77777777777778	0.12647279790750227	0.030864197530864196	0.1756820922315766
49.0	23.31370849898476	1.1761858520480954	0.7777777777777778	0.9607843137254902	127.18367346938776	10.805655113133586	2894.3035714285716	29.910714285714292	0.12381155276444755	0.01387117346938776	0.1177759460560082
31.0	17.65685424949238	1.387777332977422	0.8857142857142857	1.0	124.03225806451613	13.333159901751	2259.4666666666667	28.53333333333333	0.04129633243965299	0.016666666666666663	0.1290994448735805
89.0	32.970562748477136	1.4415206038680637	0.7606837606837606	0.956989247311828	128.31460674157304	11.804203693826526	1958.1111111111115	24.462962962962965	0.19194650626832102	0.02674897119341563 8	0.1635511271542193
57.0	24.970562748477143	1.0	0.7037037037037037	1.0	129.26315789473685	11.852198427783787	3192.6249999999995	32.263888888888886	0.1872341715144262	0.03433641975308641	0.1853008897795324
31.0	17.65685424949238	1.387777332977422	0.8857142857142857	1.0	130.48387096774192	12.978570346591152	2937.566666666666	31.499999999999996	0.07197952825498055	0.018888888888888889	0.137436854187255 3
31.0	17.65685424949238	1.387777332977422	0.8857142857142857	1.0	130.48387096774192	9.033525256730941	2719.2333333333336	28.5	0.014783097029254608	0.016666666666666666	0.1290994448735805
33.0	19.31370849898476	1.5869840952317442	0.7333333333333333	1.0	129.6060606060606	14.40315494466276	3178.7500000000005	36.5	0.11962640221676564	0.021250000000000005	0.1457737973711325

Fig. 21: Extracted features from synthetic cells

Our next step involves refining these parameters to calculate the significant features from the un-biased clusters, followed by re-clustering using those significant parameters. This is a semi-supervised approach aimed at ensuring that the clusters obtained are valid and that the model is a real world applicable model.

12 What makes a cluster unique?

12.1 Visualization of Feature Variations Across Clusters

To analyze the variations in feature values across different clusters, we employed a bar plot visualization technique. This approach allowed us to compare how individual features change across various clusters, providing insights into the underlying patterns within the dataset.

- **Data Preparation**: The feature data was first reshaped from a wide to a long format, which made it easier to plot. Each feature was associated with its corresponding cluster, enabling a clear comparison across clusters.

- **Normalization**: To ensure that all feature values were represented on a positive axis, we normalized the data by shifting the entire dataset upwards. Specifically, we added a constant value to each feature, which was calculated as the absolute value of the minimum feature value in the dataset, plus a small epsilon to avoid numerical issues. This normalization step ensured that all plotted values were positive, providing a clearer and more interpretable visualization.

- **Visualization**: We generated a bar plot with features on the x-axis and their corresponding normalized values on the y-axis. Different clusters were represented by distinct colors (hues), enabling a visual comparison of how each feature varied across clusters. The plot was enhanced by rotating the x-axis labels for better readability and setting an appropriate figure size for clarity (Fig. 22).

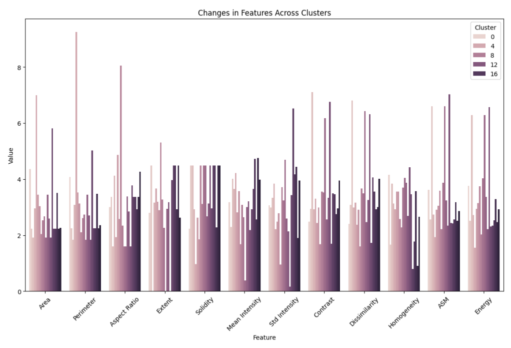


Fig. 22: Changes in features across clusters

This visualization helped to highlight key differences and similarities in feature values across clusters, serving as a valuable tool in the analysis of the data's structure and the effectiveness of the clustering algorithm.

12.2 Generation of Synthetic Image Equations

In our analysis, we sought to quantify the relationship between feature vectors and the corresponding synthetic images. To achieve this, we developed a method to generate mathematical equations that represent synthetic images as a linear combination of their feature vectors. This approach allowed us to express the contribution of each feature to the overall synthetic image in a clear and interpretable form.

- **Equation Generation**: For each cluster center, an equation was generated that defines the synthetic image associated with that cluster. The feature vector of the cluster center was used as the basis for this equation. Specifically, each feature in the vector was multiplied by its corresponding value, and these terms were summed to produce the final equation. The general form of the equation is as follows:

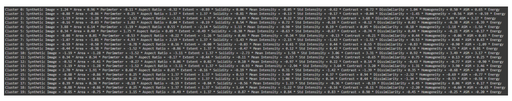
$$\text{Synthetic Image} = \sum_{i=1}^{n} (\text{Value}_i \times \text{Feature}_i)$$

  Where $Value_i$ represents the numerical value of the i-th feature, and $Feature_i$ is the name of the feature. Each equation thus provides a linear combination of features that defines the synthetic image for a specific cluster.

- **Implementation**: The process was automated using a custom Python function, which constructs the equation string by iterating over the feature vector of each cluster center. For each feature, the function formats the corresponding value to two decimal places and appends it to the equation string. The final equation is then stored in a list, with each equation associated with its respective cluster.

12.3 Results

The generated equations (Fig. 23) were displayed as shown below, each corresponding to a different cluster. These equations provide a straightforward representation of how the features combine to form the synthetic images, offering insights into the structure and characteristics of each cluster.



```
Cluster 0: Synthetic Image = 1.24 * Area + 0.96 * Perimeter + -0.11 * Aspect Ratio + -0.32 * Extent + -0.89 * Solidity + 0.06 * Mean Intensity + -0.05 * Std Intensity + -0.62 * Contrast + -0.72 * Dissimilarity + 1.04 * Homogeneity + 0.50 * ASM + 0.65 * Energy
Cluster 1: Synthetic Image = -0.88 * Area + -0.86 * Perimeter + 0.25 * Aspect Ratio + 1.37 * Extent + 1.37 * Solidity + -0.81 * Mean Intensity + -0.13 * Std Intensity + -0.17 * Contrast + -0.04 * Dissimilarity + -1.45 * Homogeneity + -0.56 * ASM + -0.59 * Energy
Cluster 2: Synthetic Image = -1.19 * Area + -1.28 * Perimeter + -1.52 * Aspect Ratio + -3.11 * Extent + 1.37 * Solidity + 0.89 * Mean Intensity + 0.22 * Std Intensity + 3.99 * Contrast + 3.68 * Dissimilarity + 0.73 * Homogeneity + 3.49 * ASM + 3.17 * Energy
Cluster 3: Synthetic Image = -0.16 * Area + -0.03 * Perimeter + 1.02 * Aspect Ratio + 0.04 * Extent + -0.19 * Solidity + 0.54 * Mean Intensity + 0.72 * Std Intensity + -0.18 * Contrast + -0.12 * Dissimilarity + 0.02 * Homogeneity + -0.38 * ASM + -0.39 * Energy
Cluster 4: Synthetic Image = 3.88 * Area + 6.14 * Perimeter + -1.19 * Aspect Ratio + 0.55 * Extent + -2.14 * Solidity + 1.11 * Mean Intensity + -0.90 * Std Intensity + 0.19 * Contrast + 0.05 * Dissimilarity + -0.18 * Homogeneity + -1.18 * ASM + -1.56 * Energy
Cluster 5: Synthetic Image = 0.34 * Area + 0.40 * Perimeter + 1.75 * Aspect Ratio + 0.09 * Extent + -0.49 * Solidity + -0.30 * Mean Intensity + -0.63 * Std Intensity + -0.67 * Contrast + -0.74 * Dissimilarity + 0.44 * Homogeneity + -0.21 * ASM + -0.17 * Energy
Cluster 6: Synthetic Image = -0.08 * Area + 0.01 * Perimeter + -0.53 * Aspect Ratio + -0.22 * Extent + -1.26 * Solidity + 0.46 * Mean Intensity + -0.34 * Std Intensity + -0.13 * Contrast + -0.21 * Dissimilarity + 0.44 * Homogeneity + -0.06 * ASM + 0.03 * Energy
Cluster 7: Synthetic Image = -1.07 * Area + -1.01 * Perimeter + 4.93 * Aspect Ratio + 2.19 * Extent + 1.37 * Solidity + -1.43 * Mean Intensity + -2.15 * Std Intensity + -1.43 * Contrast + -1.51 * Dissimilarity + -0.53 * Homogeneity + 0.48 * ASM + 0.63 * Energy
Cluster 8: Synthetic Image = -0.59 * Area + -0.50 * Perimeter + -0.78 * Aspect Ratio + 0.16 * Extent + -0.00 * Solidity + -0.03 * Mean Intensity + 0.61 * Std Intensity + 0.44 * Contrast + 0.55 * Dissimilarity + -0.83 * Homogeneity + -0.90 * ASM + -1.09 * Energy
Cluster 9: Synthetic Image = -0.44 * Area + -0.38 * Perimeter + -1.52 * Aspect Ratio + -0.86 * Extent + 1.37 * Solidity + -0.47 * Mean Intensity + 0.12 * Std Intensity + 0.42 * Contrast + 0.38 * Dissimilarity + 0.58 * Homogeneity + 0.75 * ASM + 0.91 * Energy
Cluster 10: Synthetic Image = -1.19 * Area + -1.28 * Perimeter + -1.52 * Aspect Ratio + -3.11 * Extent + 1.37 * Solidity + -2.73 * Mean Intensity + 1.58 * Std Intensity + 3.06 * Contrast + 3.31 * Dissimilarity + 0.92 * Homogeneity + 3.49 * ASM + 3.17 * Energy
Cluster 11: Synthetic Image = 0.34 * Area + 0.34 * Perimeter + 0.26 * Aspect Ratio + -0.17 * Extent + -0.44 * Solidity + -0.11 * Mean Intensity + -0.53 * Std Intensity + -0.56 * Contrast + -0.65 * Dissimilarity + 0.75 * Homogeneity + 0.13 * ASM + 0.25 * Energy
Cluster 12: Synthetic Image = -0.52 * Area + -0.41 * Perimeter + -0.27 * Aspect Ratio + 0.06 * Extent + 0.02 * Solidity + 0.10 * Mean Intensity + -0.97 * Std Intensity + 0.22 * Contrast + 0.14 * Dissimilarity + -0.43 * Homogeneity + -0.77 * ASM + -0.90 * Energy
Cluster 13: Synthetic Image = -1.19 * Area + -1.28 * Perimeter + -1.52 * Aspect Ratio + -3.11 * Extent + 1.37 * Solidity + -0.93 * Mean Intensity + -2.94 * Std Intensity + 3.64 * Contrast + 3.20 * Dissimilarity + 1.31 * Homogeneity + 3.90 * ASM + 3.46 * Energy
Cluster 14: Synthetic Image = 2.70 * Area + 1.91 * Perimeter + 0.66 * Aspect Ratio + 0.85 * Extent + -0.16 * Solidity + -0.19 * Mean Intensity + 0.31 * Std Intensity + -1.42 * Contrast + -1.39 * Dissimilarity + 0.35 * Homogeneity + -0.68 * ASM + -0.80 * Energy
Cluster 15: Synthetic Image = -0.88 * Area + -0.86 * Perimeter + 0.25 * Aspect Ratio + 1.37 * Extent + 1.37 * Solidity + 0.53 * Mean Intensity + 3.40 * Std Intensity + 0.37 * Contrast + 0.94 * Dissimilarity + -2.32 * Homogeneity + -0.69 * ASM + -0.77 * Energy
Cluster 16: Synthetic Image = -0.88 * Area + -0.86 * Perimeter + 0.25 * Aspect Ratio + 1.37 * Extent + 1.37 * Solidity + 1.61 * Mean Intensity + 1.06 * Std Intensity + 0.34 * Contrast + 0.44 * Dissimilarity + -1.34 * Homogeneity + -0.55 * ASM + -0.58 * Energy
Cluster 17: Synthetic Image = 0.39 * Area + 0.36 * Perimeter + -0.18 * Aspect Ratio + -0.19 * Extent + -0.84 * Solidity + -0.56 * Mean Intensity + 1.33 * Std Intensity + -0.36 * Contrast + -0.19 * Dissimilarity + 0.45 * Homogeneity + 0.06 * ASM + 0.17 * Energy
Cluster 18: Synthetic Image = -0.88 * Area + -0.86 * Perimeter + 0.25 * Aspect Ratio + 1.37 * Extent + 1.37 * Solidity + 1.64 * Mean Intensity + -1.22 * Std Intensity + -0.16 * Contrast + -0.11 * Dissimilarity + -2.20 * Homogeneity + -0.60 * ASM + -0.65 * Energy
Cluster 19: Synthetic Image = -0.85 * Area + -0.75 * Perimeter + 1.15 * Aspect Ratio + -0.49 * Extent + 1.37 * Solidity + 0.87 * Mean Intensity + 0.84 * Std Intensity + 0.83 * Contrast + 0.90 * Dissimilarity + -0.45 * Homogeneity + -0.25 * ASM + -0.20 * Energy
```

Fig. 23: Generated equations

This approach of representing synthetic images as equations is particularly useful for understanding the contributions of individual features and for comparing the relative importance of features across different clusters.

13 Comparison with Baseline

The baseline study [4] on neuron classification primarily focuses on adult human brain tissues, employing a combination of manual and semi-automated techniques [5] to distinguish between neurons, glia, and endothelial cells. While effective, this approach is limited by its reliance on static histological markers and manual intervention, which can introduce variability and restrict scalability.

In contrast, our work presents a fully automated pipeline for cell classification, specifically designed for application to developing brain tissue—an area where, to our knowledge, no such classification has been attempted before. The dynamic nature of cellular development in the fetal brain presents unique challenges that our model addresses by integrating advanced image processing techniques and machine learning algorithms.

Our model automates the entire classification process, eliminating the need for manual intervention and allowing for the efficient processing of large datasets. This scalability is particularly crucial in the context of fetal brain research, where the cell populations are not only diverse but also rapidly evolving. Furthermore, our approach significantly advances classification accuracy by incorporating features such as Gaussian Blur, watershed algorithm, and PCA-based clustering. These enhancements enable a more nuanced distinction between cell types, even within the complex environment of the developing brain.

This work represents a novel contribution to the field, extending the scope of neuron classification from static adult tissues to the dynamic and formative stages of brain development. By applying our automated model to fetal brain tissue, we provide new insights into neurodevelopment, paving the way for future research into early brain formation and associated developmental disorders.

In summary, our model not only builds upon the foundation laid by previous studies but also expands the frontier of neurodevelopmental research, offering a robust and scalable solution for the automated classification of cells in the developing human brain.

14 Future Works

In the next phase of this project, I plan to enhance the robustness of the model by applying additional layers of synthetic image generation and feature extraction. This iterative approach will enable further refinement of the dataset, allowing the model to better capture the complexities of fetal brain cell classification. By generating more diverse synthetic images and extracting increasingly detailed features, the model's ability to generalize across various cell types and developmental stages will be significantly improved.

Furthermore, I aim to explore the incorporation of DALL-E, a cutting-edge generative model, into the workflow. DALL-E's ability to generate high-quality, detailed images from textual descriptions could be leveraged to create synthetic images that represent various cell types and conditions more accurately. This could serve as an additional source of training data, particularly for underrepresented cell types or rare developmental stages. By integrating DALL-E-generated images with the existing dataset, the model could benefit from a more diverse and enriched training set, potentially leading to improved classification accuracy and generalizability.

These enhancements will not only strengthen the current model but also open new avenues for research into the development of more sophisticated tools for neurodevelopmental studies. By continually refining the dataset and leveraging advanced generative models, the project will move closer to achieving highly accurate and reliable cell classification in developing human brains.

15 Codes

All the codes can be found in the following links:

- Part-1:Watershed Algorithm, ROI Extraction & Feature Extraction
- Part-2: Feature Extraction from Single Cell
- Part-3: Graph Cut Segmentation, KMeans Clustering & Synthetic Image Generation
- Part-4: Visible Feature Extraction & Cluster Visualization Techniques

References

[1] Ofek Ophir, Orit Shefi and Ofir Lindenbaum, "Classifying Neuronal Cell Types Based on Shared Electrophysiological Information from Humans and Mice," 2024.

[2] Abiodun M. Ikotun, Absalom E. Ezugwu, Laith Abualigah, Belal Abuhaija, and Jia Heming, " K-means clustering algorithms: A comprehensive review, variants analysis, and advances in the era of big data," 2023.

[3] Chen Yu-ke, Wu Xiao-ming, Cai Ken and Ou Shan-xin, "CT Image Segmentation based on Clustering and Graph-Cuts," 2011.

[4] Miguel Á. García-Cabezas, Yohan J. John, Helen Barbas and Basilis Zikopoulos, "Distinction of Neurons, Glia and Endothelial

Cells in the Cerebral Cortex: An Algorithm Based on Cytological Features," 2016.

[5] Agata KOŁODZIEJCZYK, Magdalena ŁADNIAK and Adam PIORKOWSKI, "Constructing Software for analysis of Neuron, Glial and Endotheliaal Cell Numbers and Density in Histological NISSL-Stained Rodent Brain Tissue," 2014.