# esraaj_lab1

January 16, 2026

MLPR Lab 1 By Ersaaj Sarkar Gupta U20240040

Report attached at the end

## 1 Introduction

[1]:
```
# ----- Install Libraries ----- #
!pip install opencv-python
!pip install numpy
!pip install matplotlib
```

```
Requirement already satisfied: opencv-python in
/home/esraaj/jupyterenv/lib/python3.13/site-packages (4.12.0.88)
Requirement already satisfied: numpy<2.3.0,>=2 in
/home/esraaj/jupyterenv/lib/python3.13/site-packages (from opencv-python)
(2.2.6)
Requirement already satisfied: numpy in
/home/esraaj/jupyterenv/lib/python3.13/site-packages (2.2.6)
Requirement already satisfied: matplotlib in
/home/esraaj/jupyterenv/lib/python3.13/site-packages (3.10.5)
Requirement already satisfied: contourpy>=1.0.1 in
/home/esraaj/jupyterenv/lib/python3.13/site-packages (from matplotlib) (1.3.3)
Requirement already satisfied: cycler>=0.10 in
/home/esraaj/jupyterenv/lib/python3.13/site-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/home/esraaj/jupyterenv/lib/python3.13/site-packages (from matplotlib) (4.59.1)
Requirement already satisfied: kiwisolver>=1.3.1 in
/home/esraaj/jupyterenv/lib/python3.13/site-packages (from matplotlib) (1.4.9)
Requirement already satisfied: numpy>=1.23 in
/home/esraaj/jupyterenv/lib/python3.13/site-packages (from matplotlib) (2.2.6)
Requirement already satisfied: packaging>=20.0 in
/home/esraaj/jupyterenv/lib/python3.13/site-packages (from matplotlib) (25.0)
Requirement already satisfied: pillow>=8 in
/home/esraaj/jupyterenv/lib/python3.13/site-packages (from matplotlib) (11.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/home/esraaj/jupyterenv/lib/python3.13/site-packages (from matplotlib) (3.2.3)
Requirement already satisfied: python-dateutil>=2.7 in
/home/esraaj/jupyterenv/lib/python3.13/site-packages (from matplotlib)
```

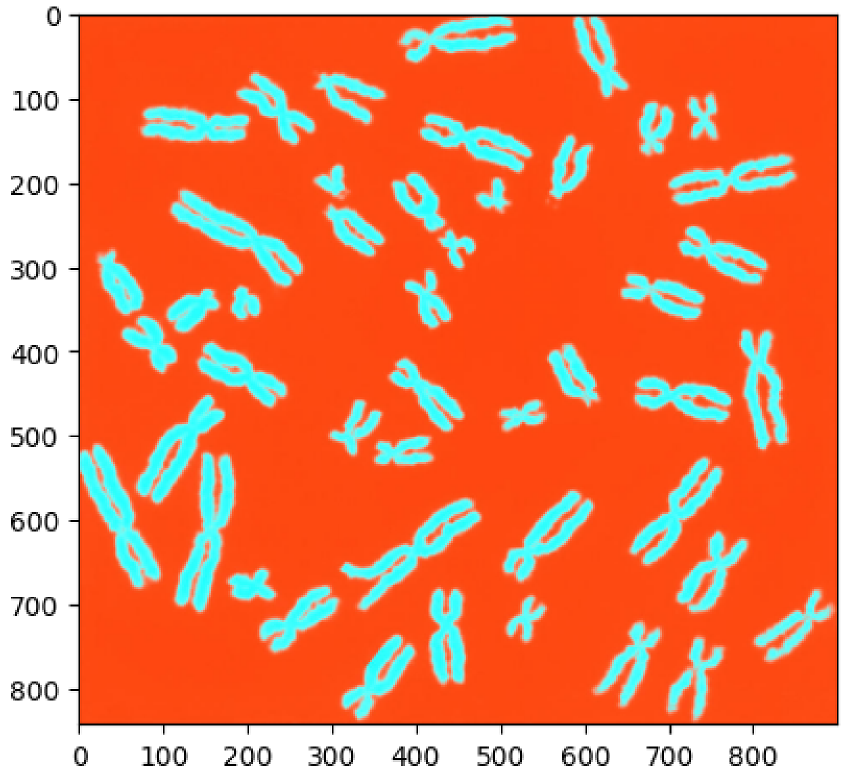
```
(2.9.0.post0)
Requirement already satisfied: six>=1.5 in
/home/esraaj/jupyterenv/lib/python3.13/site-packages (from python-
dateutil>=2.7->matplotlib) (1.17.0)
```

[2]:
```python
# ---- Imports ---- #

import cv2
import numpy as np
import matplotlib.pyplot as plt

from pathlib import Path
```

[ ]:
```python
# ---- Read Image ---- #

image_path : Path = Path("chromosomes.jpg")
image = cv2.imread(image_path) # <--- This is a stupid error, imread expects a␣
 ↪str for a path (non fatal error)

print(f"Image imported as {type(image)}")
# Display image
plt.imshow(image)
```

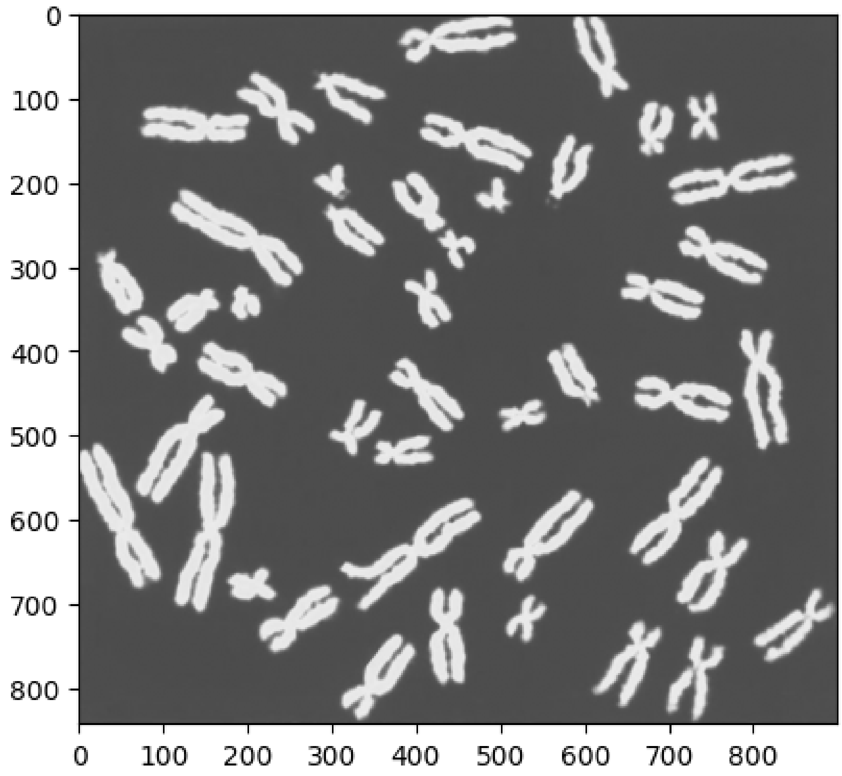
```
Image imported as <class 'numpy.ndarray'>
```

[ ]:
```
<matplotlib.image.AxesImage at 0x7cb8328eba10>
```




[4]:
```
# ---- Convert Image to Greyscale ---- #

grayscale_image : np.ndarray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# Display image
plt.imshow(grayscale_image, cmap='gray', vmin=0, vmax=255)
```

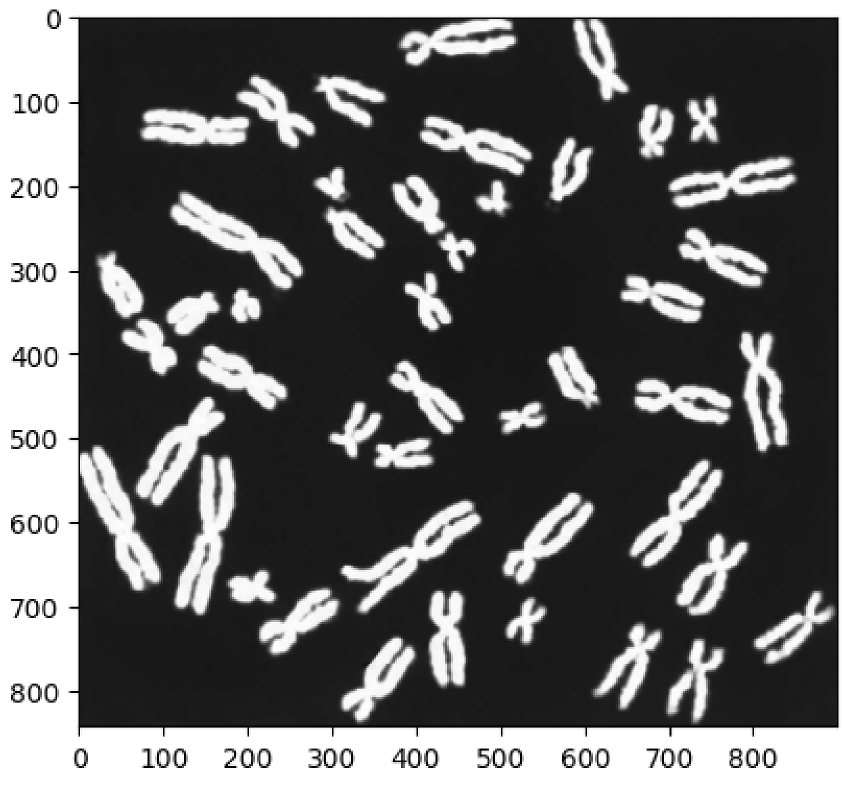
[4]: <matplotlib.image.AxesImage at 0x7cb8327e5810>




```
[ ]: # ---- Morphological Opening for Background Removal ---- #

# Define kernel
kernel = cv2.getStructuringElement(
    cv2.MORPH_RECT, # Select shape
    (5,5), # Select k-size
    anchor=None
)

"""
Notes for Reference:
    Erosion:
        A pixel stays white only if the kernel fits fully in the white
        region.

        White blobs shrink, thin lines break and small white noise disappears.

    Dialtion:
        A pixel becomes white if any part of the kernel touches white.
```

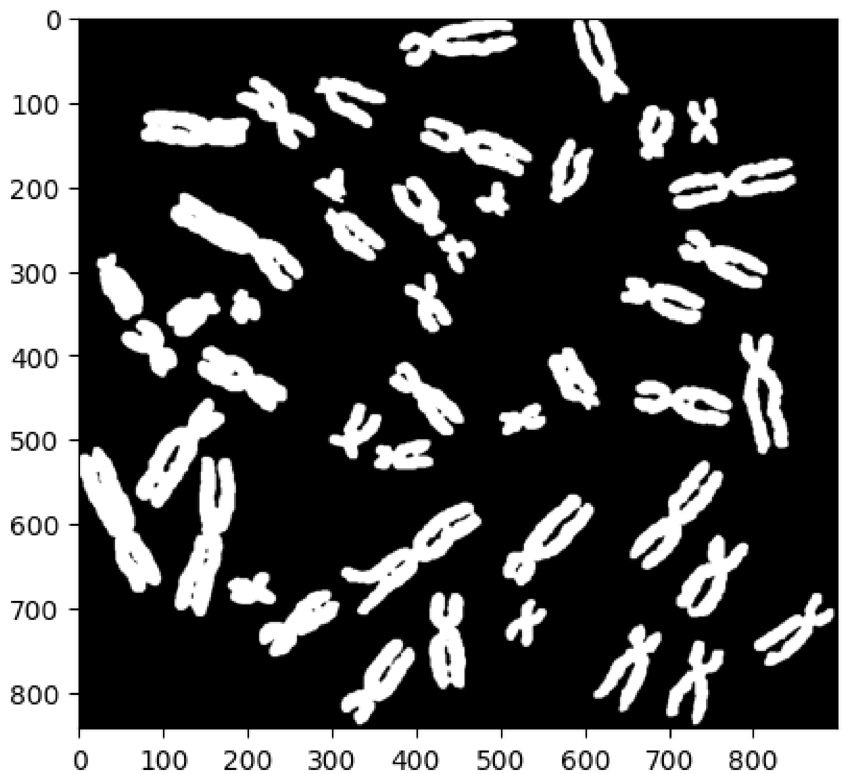
```
        White blobs grow, gaps between white regions may connect and holes get␣
↪smaller
        (since white expands into them).

    Opening: Erosion, followed by dialation
        Removes all small, white noise. Large shapes remain mostly intact and␣
↪boundaries
        are smooothened.

    Closing: Dialation, followed by erosion.
"""

# Morphological operation -- Opening
image_open = cv2.morphologyEx(grayscale_image, cv2.MORPH_OPEN, kernel)

# Display
plt.imshow(image_open, cmap="gray")
```

[ ]: <matplotlib.image.AxesImage at 0x7cb83286b110>



[ ]:
```
# ---- Image Thresholding ---- #

_, thresh_image = cv2.threshold(
    image_open,
    127, # Threshold
    255, # Maxval
    cv2.THRESH_BINARY, # Threshold type
)

# Display
plt.imshow(thresh_image, cmap='gray')
```

[ ]: <matplotlib.image.AxesImage at 0x7cb83248bc50>




[ ]:
```
#  ---- Find Contours ---- #

contours, hierarchy = cv2.findContours(
    thresh_image,
    cv2.RETR_EXTERNAL, # Outermost contours, ignore holes (Contour Mode)
    cv2.CHAIN_APPROX_SIMPLE, # Stores corner pixels, uses less space (Contour␣
↪method)
```

```python
)

print(f"Number of contours extracted: {len(contours)}")

if len(contours) < 5 :
    raise Exception("Too few contours!")

# Store features for each chromosome
features : list = list([])

# Loop through all the contours
for i,c in enumerate(contours):
    area = cv2.contourArea(c)
    perimeter = cv2.arcLength(c, True)
    circularity = (4 * np.pi * area) /  (perimeter**2)

    x,y,w,h = cv2.boundingRect(c) # Bounding box

    features.append(dict({
        'id': i,
        'x': x, 'y': y,
        'area': area,
        'perimeter': perimeter,
        'circularity': circularity,
        'width': w,
        'height': h,
        'aspect_ratio' : h / w # Shape
    }))


# ---- Place Features into a Dataframe ---- #

import pandas as pd

df : pd.DataFrame = pd.DataFrame(features)
print(df)

# Does anyone even read these comments?
```

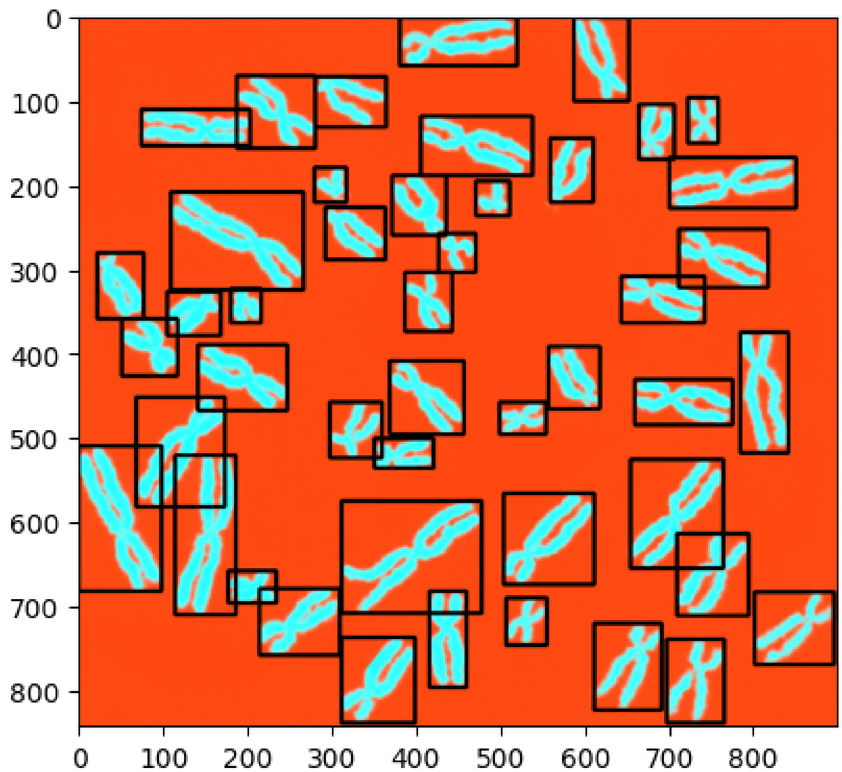
```
Number of contours extracted: 46
   id    x    y    area   perimeter  circularity  width  height  aspect_ratio
0   0  696  739  2488.5  412.634556     0.183661     68      99      1.455882
1   1  311  737  3492.5  385.404108     0.295470     87     101      1.160920
2   2  610  720  2818.0  437.102592     0.185346     80     103      1.287500
3   3  506  690  1163.0  214.166520     0.318630     48      56      1.166667
4   4  800  683  2588.5  441.144223     0.167146     94      86      0.914894
5   5  415  682  3674.5  370.007140     0.337278     44     114      2.590909
6   6  214  679  3274.0  315.563488     0.413157     95      79      0.831579
```

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 7 | 7 | 177 | 658 | 1431.5 | 183.095453 | 0.536594 | 57 | 38 | 0.666667 |
| 8 | 8 | 708 | 614 | 3639.5 | 341.462984 | 0.392251 | 85 | 97 | 1.141176 |
| 9 | 9 | 311 | 575 | 5932.0 | 543.126977 | 0.252702 | 166 | 133 | 0.801205 |
| 10 | 10 | 503 | 566 | 4392.0 | 358.818322 | 0.428670 | 107 | 108 | 1.009346 |
| 11 | 11 | 653 | 526 | 4255.5 | 603.796025 | 0.146683 | 110 | 129 | 1.172727 |
| 12 | 12 | 114 | 521 | 6248.0 | 605.161468 | 0.214392 | 72 | 189 | 2.625000 |
| 13 | 13 | 0 | 510 | 6332.0 | 464.617312 | 0.368604 | 98 | 172 | 1.755102 |
| 14 | 14 | 350 | 501 | 1713.0 | 235.681239 | 0.387541 | 70 | 35 | 0.500000 |
| 15 | 15 | 498 | 458 | 1161.5 | 178.124891 | 0.460023 | 55 | 38 | 0.690909 |
| 16 | 16 | 297 | 458 | 1631.0 | 266.936072 | 0.287640 | 62 | 66 | 1.064516 |
| 17 | 17 | 68 | 452 | 4996.0 | 383.788885 | 0.426233 | 105 | 130 | 1.238095 |
| 18 | 18 | 658 | 431 | 3343.5 | 466.232535 | 0.193289 | 116 | 54 | 0.465517 |
| 19 | 19 | 368 | 409 | 2835.0 | 315.019332 | 0.358995 | 88 | 87 | 0.988636 |
| 20 | 20 | 556 | 392 | 2369.0 | 229.137083 | 0.567001 | 61 | 74 | 1.213115 |
| 21 | 21 | 141 | 390 | 3613.5 | 325.806129 | 0.427779 | 106 | 78 | 0.735849 |
| 22 | 22 | 783 | 375 | 3995.0 | 592.617312 | 0.142948 | 57 | 143 | 2.508772 |
| 23 | 23 | 51 | 359 | 2115.5 | 246.551296 | 0.437329 | 66 | 68 | 1.030303 |
| 24 | 24 | 105 | 327 | 1902.5 | 189.923879 | 0.662789 | 63 | 52 | 0.825397 |
| 25 | 25 | 180 | 323 | 967.0 | 133.053823 | 0.686407 | 36 | 41 | 1.138889 |
| 26 | 26 | 642 | 309 | 2945.5 | 292.693432 | 0.432059 | 99 | 55 | 0.555556 |
| 27 | 27 | 386 | 304 | 1718.5 | 271.036578 | 0.293970 | 56 | 70 | 1.250000 |
| 28 | 28 | 22 | 281 | 2331.5 | 222.894442 | 0.589721 | 55 | 78 | 1.418182 |
| 29 | 29 | 426 | 258 | 914.0 | 180.852812 | 0.351160 | 44 | 46 | 1.045455 |
| 30 | 30 | 710 | 253 | 3033.5 | 431.060963 | 0.205152 | 106 | 69 | 0.650943 |
| 31 | 31 | 292 | 227 | 2247.5 | 237.178714 | 0.502063 | 71 | 62 | 0.873239 |
| 32 | 32 | 109 | 209 | 5706.5 | 495.771639 | 0.291754 | 157 | 116 | 0.738854 |
| 33 | 33 | 470 | 196 | 830.5 | 143.982755 | 0.503418 | 40 | 41 | 1.025000 |
| 34 | 34 | 371 | 189 | 2387.5 | 227.379723 | 0.580296 | 64 | 71 | 1.109375 |
| 35 | 35 | 279 | 180 | 838.0 | 142.225395 | 0.520595 | 38 | 41 | 1.078947 |
| 36 | 36 | 699 | 168 | 4013.0 | 634.901582 | 0.125103 | 150 | 60 | 0.400000 |
| 37 | 37 | 558 | 145 | 1967.0 | 316.735062 | 0.246389 | 51 | 76 | 1.490196 |
| 38 | 38 | 405 | 119 | 3977.5 | 440.232535 | 0.257903 | 132 | 70 | 0.530303 |
| 39 | 39 | 74 | 111 | 4130.5 | 332.066015 | 0.470721 | 129 | 43 | 0.333333 |
| 40 | 40 | 663 | 105 | 1886.5 | 194.610172 | 0.625944 | 42 | 65 | 1.547619 |
| 41 | 41 | 721 | 97 | 1189.5 | 226.409160 | 0.291599 | 36 | 54 | 1.500000 |
| 42 | 42 | 280 | 72 | 2201.5 | 338.492421 | 0.241452 | 84 | 60 | 0.714286 |
| 43 | 43 | 187 | 71 | 2968.0 | 341.705624 | 0.319425 | 93 | 86 | 0.924731 |
| 44 | 44 | 586 | 2 | 2679.5 | 401.261972 | 0.209126 | 65 | 99 | 1.523077 |
| 45 | 45 | 380 | 2 | 3846.5 | 567.546244 | 0.150063 | 139 | 57 | 0.410072 |

[ ]:
```
# ----- Print Boxes ----- #

# Draw bounding boxes
out = image.copy() # Copy image

for c in contours:
    x, y, w, h = cv2.boundingRect(c)
```

```python
    cv2.rectangle(
        out,
        (x,y), # Coordinates
        (x+w, y +h), # Bounds
        (0, 0, 0), # Box colour (BGR)
        3 # Box thickness
    )

# Display
plt.imshow(out)
```

[ ]: <matplotlib.image.AxesImage at 0x7cb7f7511f90>




[ ]:
```python
#  ----- Standardization ---- #

from sklearn.preprocessing import StandardScaler

feature_coloums : list = [
    'width', 'height', 'aspect_ratio', 'area', 'perimeter', 'circularity'
]

scaler = StandardScaler() # Define scaler
```

```python
df_standard = df.copy()
df_standard[feature_coloums] = scaler.
 ↪fit_transform(df_standard[feature_coloums])

print(df_standard)
```

```
    id    x    y      area  perimeter  circularity     width    height  \
0    0  696  739 -0.299402   0.530183    -1.170675 -0.403718  0.538851
1    1  311  737  0.402025   0.329132    -0.421609  0.166314  0.596162
2    2  610  720 -0.069203   0.710838    -1.159382 -0.043698  0.653473
3    3  506  690 -1.225441  -0.935169    -0.266451 -1.003752 -0.693342
4    4  800  683 -0.229539   0.740678    -1.281314  0.376325  0.166327
5    5  415  682  0.529177   0.215451    -0.141518 -1.123758  0.968686
6    6  214  679  0.249374  -0.186523     0.366833  0.406327 -0.034262
7    7  177  658 -1.037858  -1.164576     1.193802 -0.733737 -1.209144
8    8  708  614  0.504725   0.004701     0.226776  0.106310  0.481540
9    9  311  575  2.106341   1.493649    -0.708135  2.536446  1.513143
10  10  503  566  1.030446   0.132841     0.470762  0.766347  0.796752
11  11  653  526  0.935083   1.941587    -1.418407  0.856352  1.398520
12  12  114  521  2.327110   1.951669    -0.964792 -0.283711  3.117860
13  13    0  510  2.385795   0.913987     0.068351  0.496332  2.630714
14  14  350  501 -0.841192  -0.776319     0.195217 -0.343715 -1.295111
15  15  498  458 -1.226489  -1.201275     0.680815 -0.793740 -1.209144
16  16  297  458 -0.898480  -0.545555    -0.474067 -0.583728 -0.406786
17  17   68  452  1.452421   0.317206     0.454439  0.706344  1.427176
18  18  658  431  0.297929   0.925913    -1.106174  1.036362 -0.750654
19  19  368  409 -0.057326  -0.190541     0.003976  0.196315  0.194983
20  20  556  392 -0.382889  -0.824637     1.397514 -0.613730 -0.177540
21  21  141  390  0.486560  -0.110899     0.464794  0.736345 -0.062918
22  22  783  375  0.753089   1.859051    -1.443430 -0.733737  1.799700
23  23   51  359 -0.559993  -0.696062     0.528774 -0.463722 -0.349474
24  24  105  327 -0.708801  -1.114160     2.039243 -0.553727 -0.807965
25  25  180  323 -1.362373  -1.534049     2.197470 -1.363772 -1.123177
26  26  642  309  0.019873  -0.355380     0.493467  0.526334 -0.721998
27  27  386  304 -0.837350  -0.515279    -0.431656 -0.763738 -0.292163
28  28   22  281 -0.409088  -0.870728     1.549725 -0.793740 -0.062918
29  29  426  258 -1.399400  -1.181134    -0.048512 -1.123758 -0.979899
30  30  710  253  0.081353   0.666230    -1.026692  0.736345 -0.320819
31  31  292  227 -0.467773  -0.765263     0.962459 -0.313713 -0.521408
32  32  109  209  1.948800   1.144010    -0.446507  2.266431  1.025997
33  33  470  196 -1.457736  -1.453357     0.971535 -1.243765 -1.123177
34  34  371  189 -0.369964  -0.837612     1.486583 -0.523725 -0.263507
35  35  279  180 -1.452497  -1.466333     1.086611 -1.303768 -1.123177
36  36  699  168  0.765664   2.171249    -1.562986  2.056419 -0.578720
37  37  558  145 -0.663740  -0.177873    -0.750425 -0.913747 -0.120229
38  38  405  119  0.740863   0.733947    -0.673292  1.516389 -0.292163
39  39   74  111  0.847754  -0.064680     0.752484  1.426384 -1.065866
40  40  663  105 -0.719980  -1.079559     1.792400 -1.183762 -0.435441
```

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 41 | 41 | 721 | 97 | -1.206927 | -0.844778 | -0.447542 | -1.363772 | -0.750654 |
| 42 | 42 | 280 | 72 | -0.499910 | -0.017232 | -0.783504 | 0.076309 | -0.578720 |
| 43 | 43 | 187 | 71 | 0.035592 | 0.006492 | -0.261120 | 0.346324 | 0.166327 |
| 44 | 44 | 586 | 2 | -0.165963 | 0.446215 | -1.000072 | -0.493723 | 0.538851 |
| 45 | 45 | 380 | 2 | 0.649342 | 1.673944 | -1.395764 | 1.726401 | -0.664687 |

| | aspect_ratio |
|---|---|
| 0 | 0.709955 |
| 1 | 0.139011 |
| 2 | 0.384027 |
| 3 | 0.150136 |
| 4 | -0.337208 |
| 5 | 2.906967 |
| 6 | -0.498476 |
| 7 | -0.817688 |
| 8 | 0.100796 |
| 9 | -0.557269 |
| 10 | -0.154382 |
| 11 | 0.161867 |
| 12 | 2.972955 |
| 13 | 1.289139 |
| 14 | -1.140296 |
| 15 | -0.770763 |
| 16 | -0.047592 |
| 17 | 0.288396 |
| 18 | -1.207042 |
| 19 | -0.194468 |
| 20 | 0.240043 |
| 21 | -0.683775 |
| 22 | 2.747979 |
| 23 | -0.113816 |
| 24 | -0.510442 |
| 25 | 0.096368 |
| 26 | -1.032760 |
| 27 | 0.311440 |
| 28 | 0.636981 |
| 29 | -0.084488 |
| 30 | -0.848123 |
| 31 | -0.417836 |
| 32 | -0.677960 |
| 33 | -0.124081 |
| 34 | 0.039239 |
| 35 | -0.019658 |
| 36 | -1.333861 |
| 37 | 0.776375 |
| 38 | -1.081640 |
| 39 | -1.462904 |
| 40 | 0.887525 |

```
41      0.795352
42     -0.725514
43     -0.318166
44      0.840020
45     -1.314365
```

[ ]:
```python
# ---- Normalization ---- #

from sklearn.preprocessing import MinMaxScaler

feature_coloums : list = [
    'width', 'height', 'aspect_ratio', 'area', 'perimeter', 'circularity'
]

normalizer = MinMaxScaler() # Define scaler (normal)
df_normal = df.copy()

df_normal[feature_coloums] = normalizer.
 ↪fit_transform(df_normal[feature_coloums])
print(df_normal)
```

```
    id    x    y      area  perimeter  circularity     width    height  \
0    0  696  739  0.301372   0.557103     0.104325  0.246154  0.415584
1    1  311  737  0.483868   0.502842     0.303521  0.392308  0.428571
2    2  610  720  0.361265   0.605859     0.107328  0.338462  0.441558
3    3  506  690  0.060438   0.161628     0.344781  0.092308  0.136364
4    4  800  683  0.319549   0.613912     0.074904  0.446154  0.331169
5    5  415  682  0.516950   0.472162     0.378004  0.061538  0.512987
6    6  214  679  0.444152   0.363675     0.513188  0.453846  0.285714
7    7  177  658  0.109243   0.099715     0.733099  0.161538  0.019481
8    8  708  614  0.510588   0.415284     0.475943  0.376923  0.402597
9    9  311  575  0.927293   0.817127     0.227326  1.000000  0.636364
10  10  503  566  0.647369   0.449867     0.540825  0.546154  0.474026
11  11  653  526  0.622557   0.938018     0.038447  0.569231  0.610390
12  12  114  521  0.984731   0.940739     0.159075  0.276923  1.000000
13  13    0  510  1.000000   0.660685     0.433814  0.476923  0.889610
14  14  350  501  0.160411   0.204499     0.467550  0.261538  0.000000
15  15  498  458  0.060165   0.089810     0.596683  0.146154  0.019481
16  16  297  458  0.145506   0.266779     0.289571  0.200000  0.201299
17  17   68  452  0.757157   0.499624     0.536484  0.530769  0.616883
18  18  658  431  0.456785   0.663904     0.121478  0.615385  0.123377
19  19  368  409  0.364355   0.362591     0.416695  0.400000  0.337662
20  20  556  392  0.279651   0.191459     0.787271  0.192308  0.253247
21  21  141  390  0.505862   0.384085     0.539238  0.538462  0.279221
22  22  783  375  0.575207   0.915743     0.031793  0.161538  0.701299
23  23   51  359  0.233573   0.226159     0.556252  0.230769  0.214286
24  24  105  327  0.194856   0.113321     0.957923  0.207692  0.110390
25  25  180  323  0.024811   0.000000     1.000000  0.000000  0.038961
```

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 26 | 26 | 642 | 309 | 0.384441 | 0.318104 | 0.546863 | 0.484615 | 0.129870 |
| 27 | 27 | 386 | 304 | 0.161411 | 0.274949 | 0.300849 | 0.153846 | 0.227273 |
| 28 | 28 | 22 | 281 | 0.272835 | 0.179020 | 0.827748 | 0.146154 | 0.279221 |
| 29 | 29 | 426 | 258 | 0.015178 | 0.095246 | 0.402737 | 0.061538 | 0.071429 |
| 30 | 30 | 710 | 253 | 0.400436 | 0.593820 | 0.142614 | 0.538462 | 0.220779 |
| 31 | 31 | 292 | 227 | 0.257566 | 0.207483 | 0.671579 | 0.269231 | 0.175325 |
| 32 | 32 | 109 | 209 | 0.886304 | 0.722765 | 0.296900 | 0.930769 | 0.525974 |
| 33 | 33 | 470 | 196 | 0.000000 | 0.021777 | 0.673993 | 0.030769 | 0.038961 |
| 34 | 34 | 371 | 189 | 0.283014 | 0.187957 | 0.810957 | 0.215385 | 0.233766 |
| 35 | 35 | 279 | 180 | 0.001363 | 0.018276 | 0.704595 | 0.015385 | 0.038961 |
| 36 | 36 | 699 | 168 | 0.578479 | 1.000000 | 0.000000 | 0.876923 | 0.162338 |
| 37 | 37 | 558 | 145 | 0.206580 | 0.366010 | 0.216080 | 0.115385 | 0.266234 |
| 38 | 38 | 405 | 119 | 0.572026 | 0.612095 | 0.236592 | 0.738462 | 0.227273 |
| 39 | 39 | 74 | 111 | 0.599836 | 0.396559 | 0.615742 | 0.715385 | 0.051948 |
| 40 | 40 | 663 | 105 | 0.191948 | 0.122659 | 0.892282 | 0.046154 | 0.194805 |
| 41 | 41 | 721 | 97 | 0.065255 | 0.186023 | 0.296625 | 0.000000 | 0.123377 |
| 42 | 42 | 280 | 72 | 0.249205 | 0.409364 | 0.207284 | 0.369231 | 0.162338 |
| 43 | 43 | 187 | 71 | 0.388530 | 0.415767 | 0.346199 | 0.438462 | 0.331169 |
| 44 | 44 | 586 | 2 | 0.336090 | 0.534441 | 0.149693 | 0.223077 | 0.415584 |
| 45 | 45 | 380 | 2 | 0.548214 | 0.865785 | 0.044468 | 0.792308 | 0.142857 |

| | aspect_ratio |
|---|---|
| 0 | 0.489840 |
| 1 | 0.361129 |
| 2 | 0.416364 |
| 3 | 0.363636 |
| 4 | 0.253772 |
| 5 | 0.985124 |
| 6 | 0.217416 |
| 7 | 0.145455 |
| 8 | 0.352513 |
| 9 | 0.204162 |
| 10 | 0.294987 |
| 11 | 0.366281 |
| 12 | 1.000000 |
| 13 | 0.620408 |
| 14 | 0.072727 |
| 15 | 0.156033 |
| 16 | 0.319062 |
| 17 | 0.394805 |
| 18 | 0.057680 |
| 19 | 0.285950 |
| 20 | 0.383905 |
| 21 | 0.175643 |
| 22 | 0.949282 |
| 23 | 0.304132 |
| 24 | 0.214719 |
| 25 | 0.351515 |

```
26      0.096970
27      0.400000
28      0.473388
29      0.310744
30      0.138593
31      0.235595
32      0.176954
33      0.301818
34      0.338636
35      0.325359
36      0.029091
37      0.504813
38      0.085950
39      0.000000
40      0.529870
41      0.509091
42      0.166234
43      0.258065
44      0.519161
45      0.033486
```

# 2 Report

### 2.0.1 Question 1

Contour detection can be used to find objects in an image. It does this by tracing the boundaries. Here, this is done on a thresholded image to eliminate noise. Each detected body is returned as a contour.

### 2.0.2 Question 2

Standardization of data is useful for ML methods to ensure all data is weighted equally. Most ML algorithms are sensitive to feature scaling. Standardization ensures all continuous features exist within the same scale. Otherwise features with larger scales will dominate the ML model.

### 2.0.3 Question 3

There are multiple ways in which one may handle a missing value in a dataset.

- One may extrapolate the missing value using linear regression against a correlated feature.
- One may interpolate the missing value from the rest of the filled data.
- Mean imputation may be used to fill in missing values.
- The datapoint with the missing value may be dropped.
- If most of the datapoints in the coloumn (feature) are missing, the feature may be dropped in its entirety.

### 2.0.4 Question 4

Normalization allows features to be scaled to a bound range, usually between 0 and 1. This allows all features to live in the same range, equalising the contribution of each feature in scale sensitive ML algorithms. Normalization removes dimensionality from features.

### 2.0.5 Question 5

We may seperate overlapping chromosomes by applying morphological operatons (such as erosion) to break thin connections. One may also use rotated boxes instead of axis-aligned boxes. Merged contours may be split using shape cues.