DAFTAR PUSTAKA

[1] Y. F. Fung, H. Lee, and M. F. Ercan, “Image processing application in toll collection,” Lect. Notes Eng. Comput. Sci., no. April 2015, pp. 584–588, 2006.

[2] M. Hutter and N. Brewer, “Matching 2-D ellipses to 3-D circles with application to vehicle pose identification,” 2009 24th Int. Conf. Image Vis. Comput. New Zealand, IVCNZ 2009 - Conf. Proc., pp. 153–158, 2009, doi: 10.1109/IVCNZ.2009.5378421.

[3] A. Bhujbal and D. Mane, “A survey on deep learning approaches for vehicle and number plate detection,” Int. J. Sci. Technol. Res., vol. 8, no. 12, pp. 1378–1383, 2019.

[4] A. O. Djekoune, K. Messaoudi, and K. Amara, “Incremental circle hough transform: An improved method for circle detection,” Optik (Stuttg)., vol. 133, pp. 17–31, 2017, doi: 10.1016/j.ijleo.2016.12.064.

[5] V. A. Gunawan and L. S. A. Putra, “Comparison of American Sign Language Use Identification using Multi-Class SVM Classification, Backpropagation Neural Network, K - Nearest Neighbor and Naive Bayes,” Teknik, vol. 42, no. 2, pp. 137–148, 2021, doi: 10.14710/teknik.v42i2.36929.

[6] J. Jumadi and D. Sartika, “Pengolahan Citra Digital Untuk Identifikasi Objek Menggunakan Metode Hierarchical Agglomerative Clustering,” vol. 10, no. 2, pp. 148–156, 2021.

[7] A. R. Putri, “Pengolahan Citra Dengan Menggunakan Web Cam Pada Kendaraan Bergerak Di Jalan Raya,” JIPI (Jurnal Ilm. Penelit. dan Pembelajaran Inform., vol. 1, no. 01, pp. 1–6, 2016, doi: 10.29100/jipi.v1i01.18.

[8] B. Sit, M. Iqbal Quraishi, and P. Student, "A Review Paper on Hough Transform and it's Applications in Image Processing," Int. J. Innov. Res. Sci. Eng. Technol. (An ISO, vol. 3297, p. 13, 2007, [Online]. Available: www.ijirset.com.

[9] V. Georgieva, P. Petrov, and A. Mihaylova, "GUI for Circular and Elliptic Objects Detection in Digital Images GUI für kreisförmigen und ellipsenförmigen Objektserkennung in den," vol. 1, pp. 7–10, 2017.

[10] R. M. Putra, R. D. Puriyanto, K. Uad, and J. Ring, "Sistem Deteksi dan Pelacakan Bola dengan Metode Hough circle Transform Menggunakan Kamera Omnidirectional pada Robot Sepak Bola Beroda," vol. 3, no. 3, pp. 176–184, 2021, doi: 10.12928/biste.v3i3.4786.

[11] R. A. Rizal, I. S. Girsang, and S. A. Prasetiyo, "Klasifikasi Wajah Menggunakan Support Vector Machine (SVM)," REMIK (Riset dan E-Jurnal Manaj. Inform. Komputer), vol. 3, no.2,p.1,2019,doi: 10.33395/remik.v3i2.10080.

[12] H. M and S. M.N, "A Review on Evaluation Metrics for Data Classification Evaluations," Int. J. Data Min. Knowl. Manag. Process, vol. 5, no. 2, pp. 01–11, 2015, doi: 10.5121/ijdkp.2015.5201.

[13] W. S. Maulsby, "Getting in News", in Mondry, 2008, pp. 132-133

[14] A. Z. Arifin, and A. N. Setiono, "Klasifikasi Dokumen Berita Kejadian Berbahasa Indonesia dengan Algoritma Single Pass Clustering", Institut Teknologi sepuluh November (ITS) Surabaya. http://mail.itssby.edu/~agusza/SITIAKlasifikasiEvent.pdf.

[15] V. A. Gunawan and L. S. A. Putra, "Comparison of American Sign Language Use Identification using Multi-Class SVM Classification,Backpropagation Neural Network, K - Nearest Neighbor and Naive Bayes," Teknik, vol. 42, no. 2, pp. 137–148, 2021, doi:10.14710/teknik.v42i2.36929.

[16] J. Jumadi and D. Sartika, “Pengolahan Citra Digital Untuk Identifikasi Objek Menggunakan Metode Hierarchical Agglomerative Clustering,”vol. 10, no. 2, pp. 148–156, 2021.

[17] A. R. Putri, “Pengolahan Citra Dengan Menggunakan Web Cam Pada Kendaraan Bergerak Di Jalan Raya,” JIPI (Jurnal Ilm. Penelit. Dan Pembelajaran Inform., vol. 1, no. 01, pp. 1–6, 2016, doi: 10.29100/jipi.v1i01.18.

[18] B. Sit, M. Iqbal Quraishi, and P. Student, “A Review Paper on Hough Transform and it’s Applications in Image Processing,” Int. J. Innov.Res. Sci. Eng. Technol. (An ISO, vol. 3297, p. 13, 2007, [Online]. Available: www.ijirset.com.

[19] V. Georgieva, P. Petrov, and A. Mihaylova, “GUI for Circular and Elliptic Objects Detection in Digital Images GUI für kreisförmigen und ellipsenförmigen Objektserkennung in den,” vol. 1, pp. 7–10, 2017.

[20] R. M. Putra, R. D. Puriyanto, K. Uad, and J. Ring, “Sistem Deteksi dan Pelacakan Bola dengan Metode Hough circle Transform Menggunakan Kamera Omnidirectional pada Robot Sepak Bola Beroda,” vol. 3, no. 3, pp. 176–184, 2021, doi: 10.12928/biste.v3i3.4786.

[21] R. A. Rizal, I. S. Girsang, and S. A. Prasetiyo, “Klasifikasi Wajah Menggunakan Support Vector Machine (SVM),” REMIK (Riset dan EJurnal Manaj. Inform. Komputer), vol. 3, no. 2, p. 1, 2019, doi: 10.33395/remik.v3i2.10080.

[22] H. M and S. M.N, “A Review on Evaluation Metrics for Data Classification Evaluations,” Int. J. Data Min. Knowl. Manag. Process, vol. 5,no. 2, pp. 01–11, 2015, doi: 10.5121/ijdkp.2015.5201.

[23] M. F. Rahman, M. I. Darmawidjadja, and D. Alamsah, “kalsifikasi untuk Diagnosa Diabetes Menggunakan Metode Bayesian Regularization Neural Network (RBNN),” Inform., vol. 11, no. 1, pp. 36-45, 2017.

LAMPIRAN A

```
#otsu 2
# import required module
import os
import matplotlib as plt
import matplotlib.pyplot as plt
import numpy as np
import cv2
from skimage import data
from skimage.filters import threshold_multiotsu
import csv
matplotlib.rcParams['font.size'] = 9
# Setting the font size for all plots.
def createData(imageName, e_lines, e_circles, otsuResult, show=True):
  # The input image.
  image = cv2.imread(imageName,0)

  # Applying multi-Otsu threshold for the default value, generating
  # three classes.
  # thresholds = threshold_otsu(image)
  ret,th =
cv2.threshold(image,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)
  ret, thresh1 = cv2.threshold(image, 120, 255, cv2.THRESH_BINARY +
                                            cv2.THRESH_OTSU)

  # print(ret)
  # print(th)
  ret0 = [ret]
  # Using the threshold values, we generate the three regions.
  regions = np.digitize(image, bins=ret0)
```

```
fig, ax = plt.subplots(nrows=2, ncols=2, figsize=(28,28))

# Plotting the original image.
ax[0][0].imshow(image, cmap='gray')
ax[0][0].set_title('Original')
ax[0][0].axis('off')

# Plotting the histogram and the two thresholds obtained from
# multi-Otsu.
ax[0][1].hist(image.ravel(), bins=255)
ax[0][1].set_title('Histogram')
for thresh in ret0:
    ax[0][1].axvline(thresh, color='r')

# Plotting the Multi Otsu result.
ax[1][0].imshow(thresh1,cmap='gray')
ax[1][0].set_title('Otsu result')
ax[1][0].axis('off')

ax[1][1].hist(thresh1.ravel(), bins=255)
ax[1][1].set_title('Histogram Otsu')
for thresh1 in ret0:
    ax[1][1].axvline(thresh1, color='r')

plt.subplots_adjust()
if show == False:
   plt.close()
else :
  plt.show()
fig.savefig(otsuResult, dpi=300)
```

```
#deteksi garis
RGB = cv2.imread(imageName)

# Convert the img to grayscale
gray = cv2.cvtColor(RGB, cv2.COLOR_BGR2GRAY)

# Apply edge detection method on the image
edges = cv2.Canny(gray, 50, 150, apertureSize=3)



# This returns an array of r and theta values
lines = cv2.HoughLines(edges, 1, np.pi/18, 200)
# print(lines)
# The below for loop runs till r and theta values
# are in the range of the 2d array
backtorgb = cv2.cvtColor(image,cv2.COLOR_GRAY2RGB)
lines_pos = []
if lines is not None:
  for r_theta in lines:
      arr = np.array(r_theta[0], dtype=np.float64)
      r, theta = arr
      # Stores the value of cos(theta) in a
      a = np.cos(theta)

      # Stores the value of sin(theta) in b
      b = np.sin(theta)

      # x0 stores the value rcos(theta)
      x0 = a*r
```

```
        # y0 stores the value rsin(theta)
        y0 = b*r

        # x1 stores the rounded off value of (rcos(theta)-1000sin(theta))
        x1 = int(x0 + 1000*(-b))

        # y1 stores the rounded off value of (rsin(theta)+1000cos(theta))
        y1 = int(y0 + 1000*(a))

        # x2 stores the rounded off value of (rcos(theta)+1000sin(theta))
        x2 = int(x0 - 1000*(-b))

        # y2 stores the rounded off value of (rsin(theta)-1000cos(theta))
        y2 = int(y0 - 1000*(a))
        distance = np.linalg.norm([x1 - x2, y1 - y2])
        lines_pos_data = [x1,y1,x2,y2,distance]
        lines_pos.append(lines_pos_data)

        # cv2.line draws a line in img from the point(x1,y1) to (x2,y2).
        # (0,0,255) denotes the colour of the line to be
        # drawn. In this case, it is red.

        cv2.line(backtorgb, (x1, y1), (x2, y2), (0, 0, 255), 2)

    # All the changes made in the input image are finally
    # written on a new image houghlines.jpg

    cv2.imwrite("linesDetected.jpg", backtorgb)  #not necessary at the moment
    # Ll=len(lines)
```

```
# mendeteksi lingkaran
# Blur using 3 * 3 kernel.
gray_blurred = cv2.blur(gray, (3, 3))

# Apply Hough transform on the blurred image.
detected_circles = cv2.HoughCircles(gray_blurred,
                  cv2.HOUGH_GRADIENT, 1, 20, param1 = 50,
              param2 = 30, minRadius = 1, maxRadius = 40)

# Draw circles that are detected.
if detected_circles is not None:

    # Convert the circle parameters a, b and r to integers.
    detected_circles = np.uint16(np.around(detected_circles))

  # for pt in detected_circles[0, :]:
    #    a, b, r = pt[0], pt[1], pt[2]

        # Draw the circumference of the circle.
    #   cv2.circle(img, (a, b), r, (0, 255, 0), 2)

        # Draw a small circle (of radius 1) to show the center.
      #  cv2.circle(img, (a, b), 1, (0, 0, 255), 3)
      # cv2.imshow("Detected Circle", img)
        #cv2.waitKey(0)



#Deteksi lingkaran

# Read image.
img = cv2.imread(imageName)
```

```
# Convert to grayscale.
g = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Blur using 3 * 3 kernel.
gb = cv2.blur(g, (3, 3))

# Apply Hough transform on the blurred image.
dc = cv2.HoughCircles(gb,
            cv2.HOUGH_GRADIENT, 1, 20, param1 = 50,
        param2 = 30, minRadius = 1, maxRadius = 40)
# print("Ini lingkaran", dc[0])
# print("Garis" , lines)

# export 'lines' variable to csv
# export 'dc' variable to csv

# data lingkaran
data_circles = [['x', 'y', 'r']]
for data in dc[0]:
  data_circles.append(data)



# name of csv file
filename = e_circles

# writing to csv file
with open(filename, 'w', newline=") as csvfile:
    # creating a csv writer object
    csvwriter = csv.writer(csvfile)
```

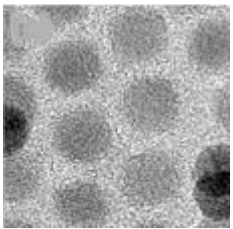
```
      # writing the data rows
      csvwriter.writerows(data_circles)

  # data garis
  data_lines = [['a', 'b','x1','y1','x2','y2','d']]
  if lines is not None:
    for i in range(len(lines)):
      dummy = [lines[i][0][0], lines[i][0][1]] + lines_pos[i]
      data_lines.append(dummy)
    # name of csv file
    filename = e_lines
    # writing to csv file
    with open(filename, 'w', newline='') as csvfile:
        # creating a csv writer object
        csvwriter = csv.writer(csvfile)
        # writing the data rows
        csvwriter.writerows(data_lines)
  return [ret, th, data_circles, data_lines]

def extractData(data, kategori):
  # Extract black and white value
  black = 0
  for subdata in data[1]:
    black += np.count_nonzero(subdata == 0)
  white = 0
  for subdata in data[1]:
    white += np.count_nonzero(subdata == 255)
  # Batas otsu
  batas_otsu = data[0]
  # print(black, white, batas_otsu)
  # Extract circles information
  jumlah_lingkaran =  len(data[2])-1
  # Sum all radius to get average
  sum_radius = round(sum([x[2] for x in data[2] if x[2] != 'r']), 2)
  average_radius = round(sum_radius/jumlah_lingkaran,2)
  # print(jumlah_lingkaran, sum_radius, average_radius)
  # Extract lines information
  jumlah_lines = len(data[3])-1
  # Sum all radius to get average
  sum_lines = round(sum([x[6] for x in data[3] if x[6] != 'd']), 2)
  try:
    average_lines = round(sum_lines/jumlah_lines,2)
  except:
    average_lines = 0
  # print(jumlah_lines, sum_lines, average_lines)
  return [black, white, batas_otsu, jumlah_lingkaran, average_radius,
jumlah_lines, average_lines, kategori]

#SVM
# assign directory
directory = 'cdots'
result_dir = "result_" + directory
# To export result, just upload your image to files folder and click run.
# iterate over files in
# that directory
hasil = []
for filename in os.listdir(directory):
    f = os.path.join(directory, filename)
    # checking if it is a file
    if os.path.isfile(f):
        f_temp = os.path.join(result_dir, os.path.splitext(filename)[0])
        # Set last parameter to True to show result, False to only save it. The value is
False by Default.
        data = createData(f, "{}-lines.csv".format(f_temp), "{}-
circles.csv".format(f_temp), "{}-otsu-result.jpg".format(f_temp), False)
        hasil_temp = extractData(data, directory)
        hasil.append(hasil_temp)
        print("{} done!".format(filename))

data_hasil = [['black',
'white','batas_otsu','n_lingkaran','average_radius','n_lines','average_line_length',
'kategori']]
for sub_data in hasil:
  data_hasil.append(sub_data)
with open('datasvm-cdots.csv', 'w', newline=") as csvfile:
    csvwriter = csv.writer(csvfile)
    csvwriter.writerows(data_hasil)

# Importing required libraries
from seaborn import load_dataset, pairplot
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
df = pd.read_csv('data.csv')
# df = df.drop(columns=['average_line_length'])
# Print the first 5 rows of the DataFrame
print(df.head())
#print(df.head())
# Dropping missing records
pairplot(df, hue='kategori')
```

```
plt.show()

X = df[['black', 'n_lingkaran']]
y = df.iloc[:, -1]

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=100)

# Building and training our model
clf = SVC(kernel='linear')
clf.fit(X_train, y_train)

import numpy as np
from seaborn import scatterplot
w = clf.coef_[0]
b = clf.intercept_[0]
x_visual = np.linspace(55,70) #Disesuaikan
y_visual = -(w[0] / w[1]) * x_visual - b / w[1]

scatterplot(data = X_train, x='black', y='n_lingkaran', hue=y_train) #Disesuaikan
plt.plot(x_visual, y_visual)
plt.show()
```

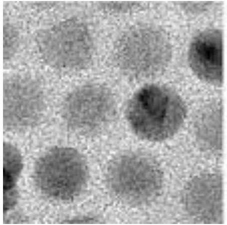
LAMPIRAN B

Kumpulan gambar TEM C-dots dan Non C-dots.

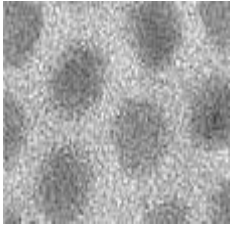
C dots 1



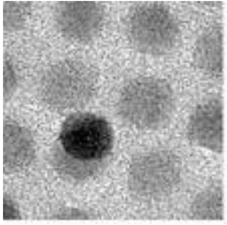
C dots 2



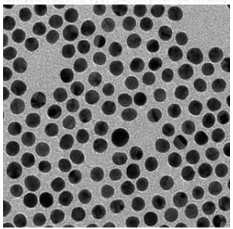
C dots 3



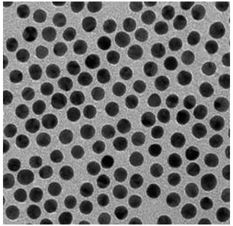
C dots 4



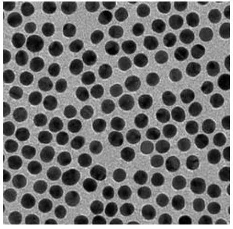
C dots 5



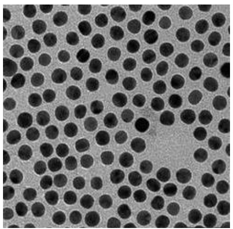
C dots 6



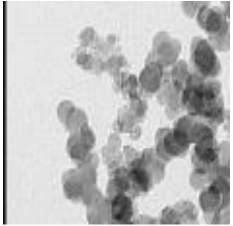
C dots 7



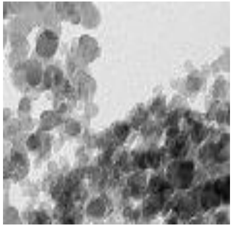
C dots 8



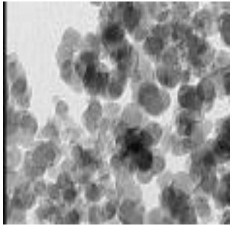
C dots 9



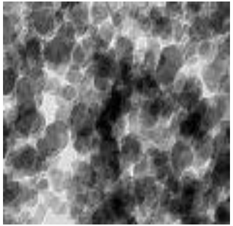
C dots 10



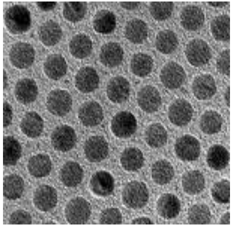
C dots 11



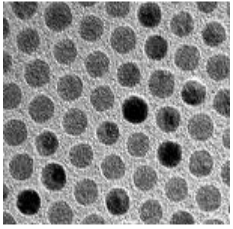
C dots 12



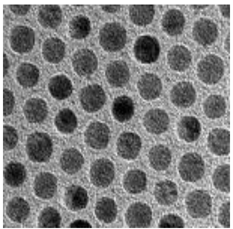
C dots 13



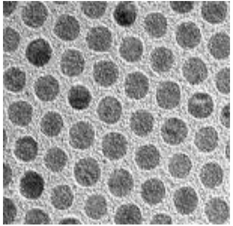
C dots 14



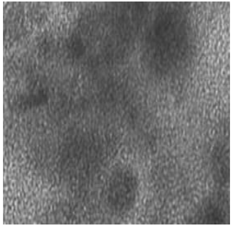
C dots 15



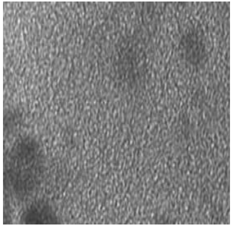
C dots 16



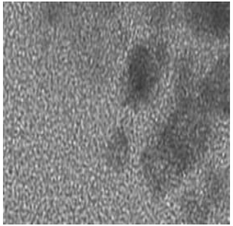
C dots 17



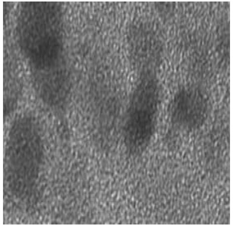
C dots 18



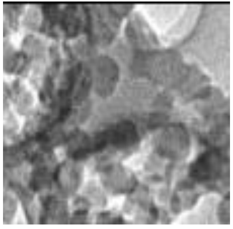
C dots 19



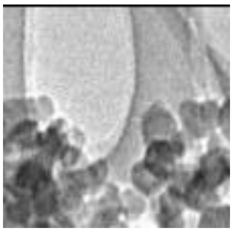
C dots 20



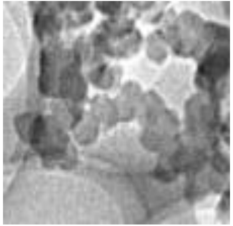
C dots 21



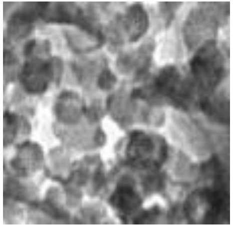
C dots 22



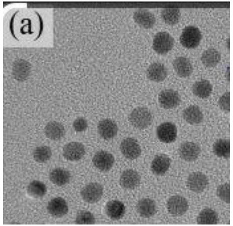
C dots 23



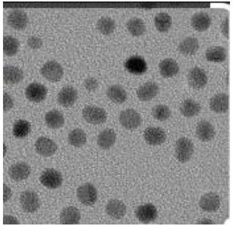
C dots 24



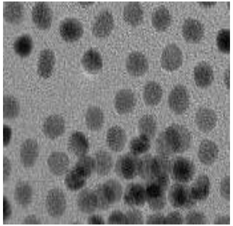
C dots 25



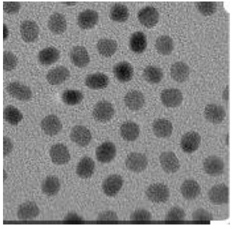
C dots 26

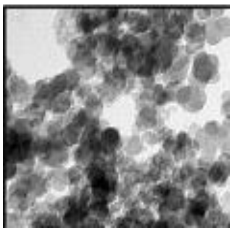
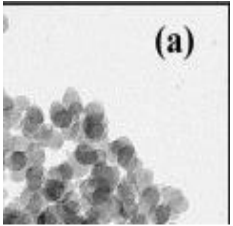
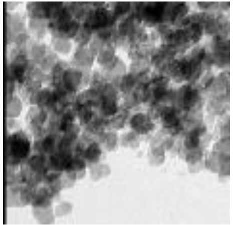
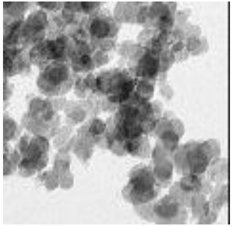
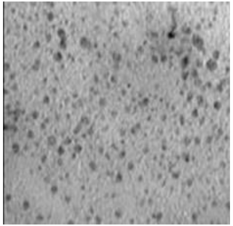
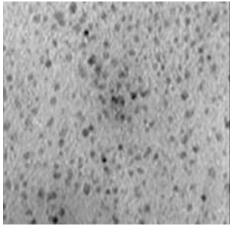
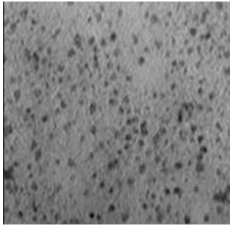


C dots 27

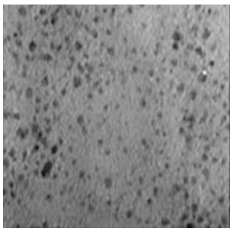


C dots 28

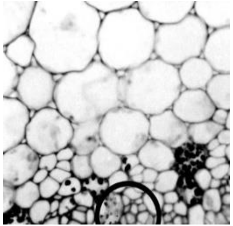


C dots 29	
C dots 30	
C dots 31	
C dots 32	
C dots 33	
C dots 34	
C dots 35	

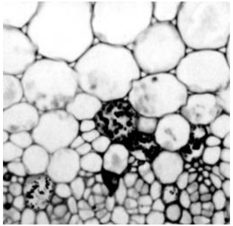
C dots 36



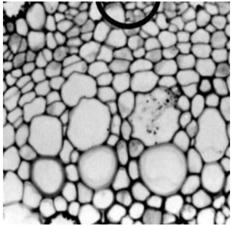
C dots 37



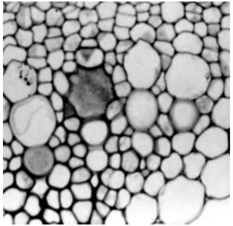
C dots 38



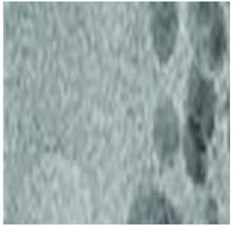
C dots 39



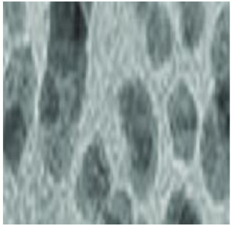
C dots 40



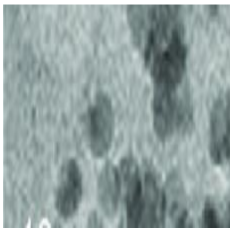
C dots 41



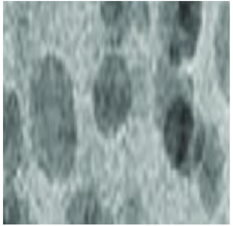
C dots 42



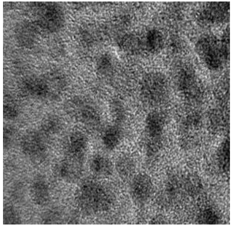
C dots 43



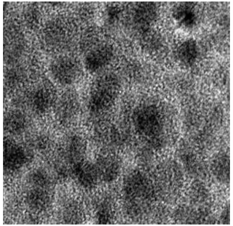
C dots 44



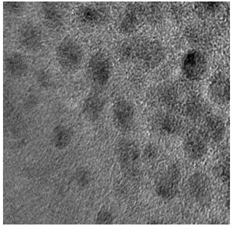
C dots 45



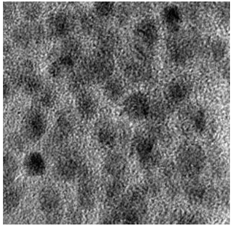
C dots 46



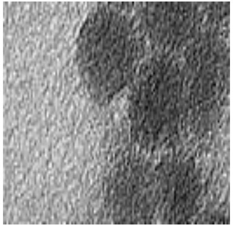
C dots 47



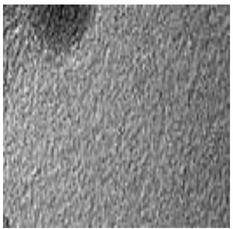
C dots 48



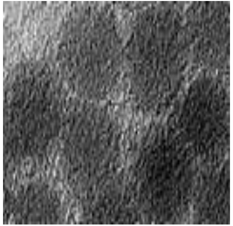
C dots 49



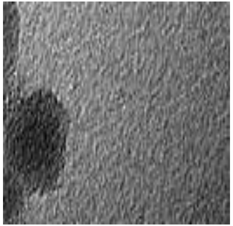
C dots 50



C dots 51



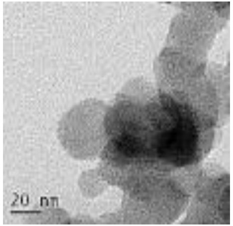
C dots 52



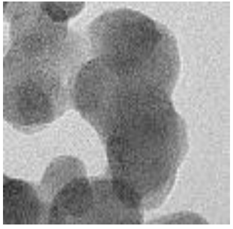
C dots 53

C dots 54

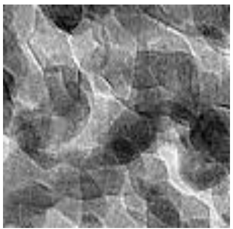
C dots 55



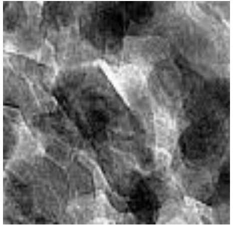
C dots 56



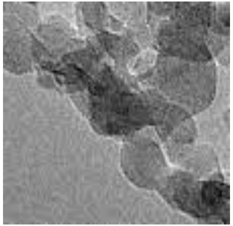
C dots 57



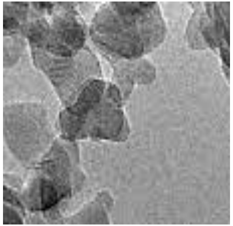
C dots 58



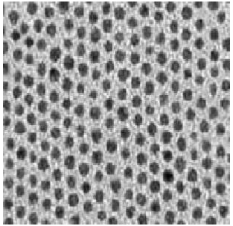
C dots 59



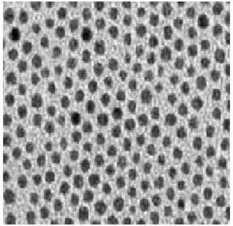
C dots 60



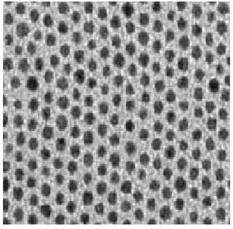
C dots 61



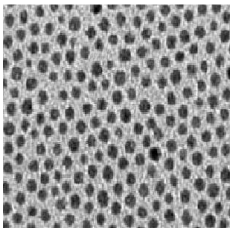
C dots 62



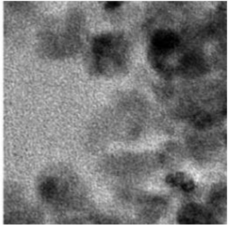
C dots 63



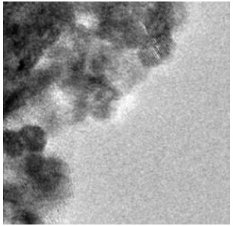
C dots 64



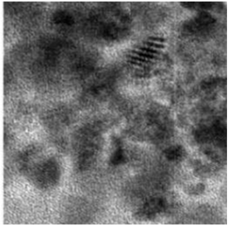
C dots 65



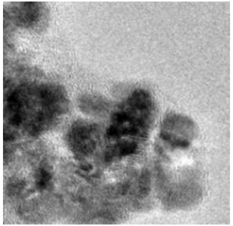
C dots 66



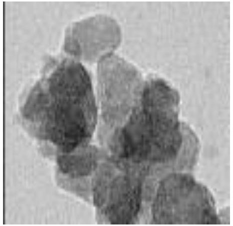
C dots 67



C dots 68

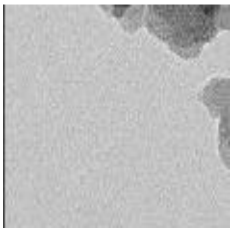


C dots 69

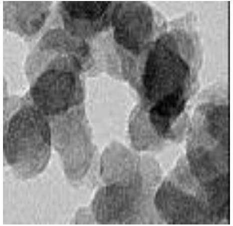


C dots 70

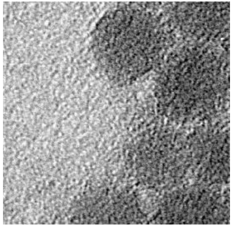
C dots 71



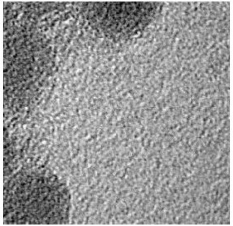
C dots 72



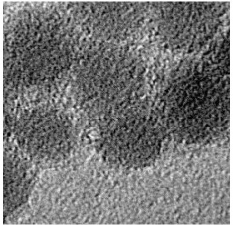
C dots 73



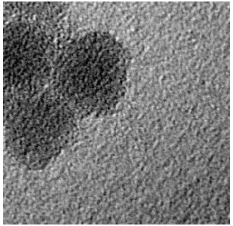
C dots 74



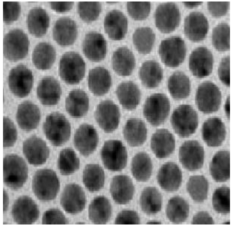
C dots 75



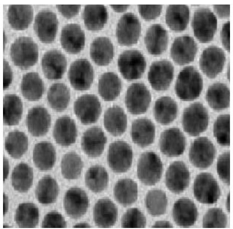
C dots 76



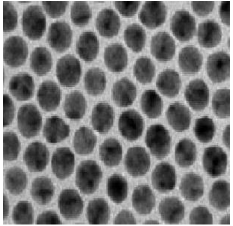
C dots 77



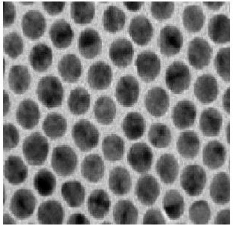
C dots 78



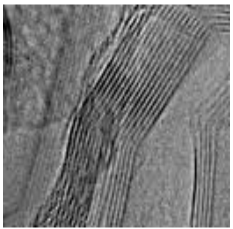
C dots 79



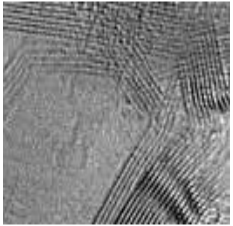
C dots 80



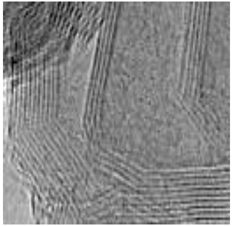
Non C dots 1



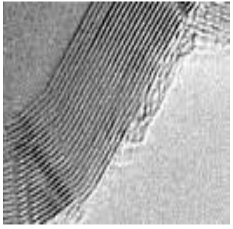
Non C dots 2



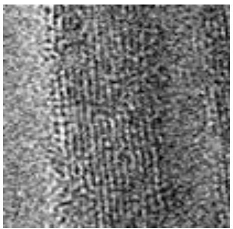
Non C dots 3



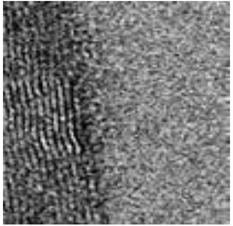
Non C dots 4



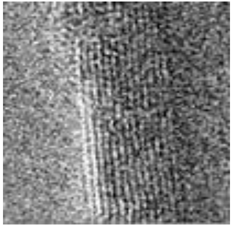
Non C dots 5



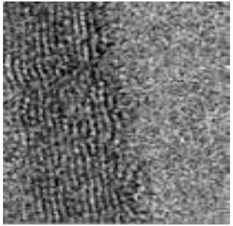
Non C dots 6



Non C dots 7



Non C dots 8

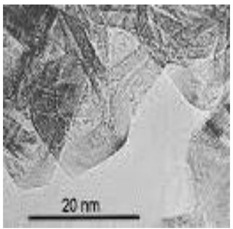


Non C dots 9

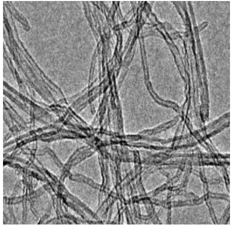
Non C dots 10

Non C dots 11

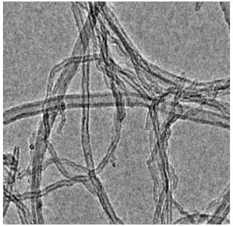
Non C dots 12



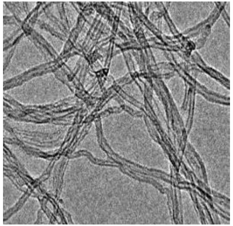
Non C dots 13



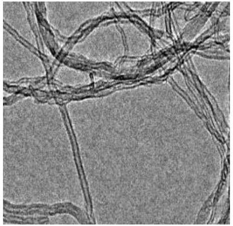
Non C dots 14



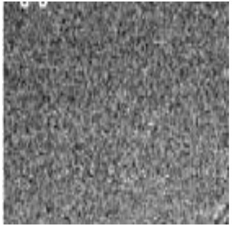
Non C dots 15



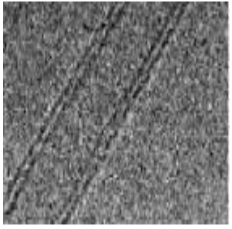
Non C dots 16



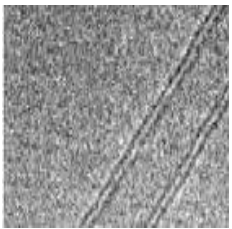
Non C dots 17



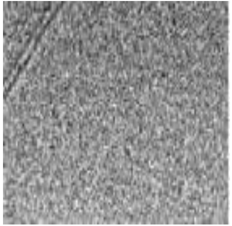
Non C dots 18



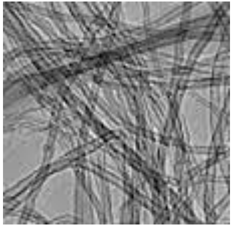
Non C dots 19



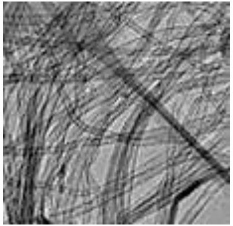
Non C dots 20



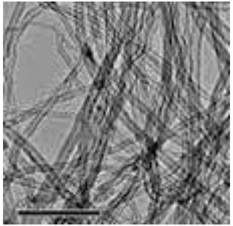
Non C dots 21



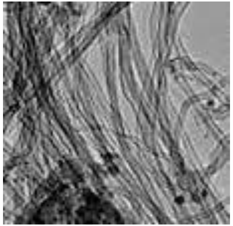
Non C dots 22



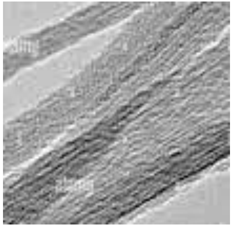
Non C dots 23



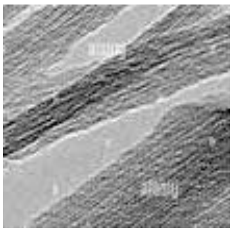
Non C dots 24



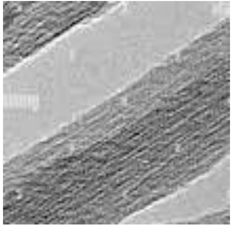
Non C dots 25



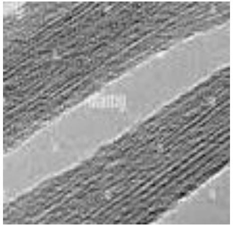
Non C dots 26



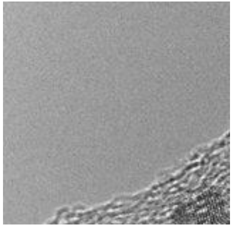
Non C dots 27



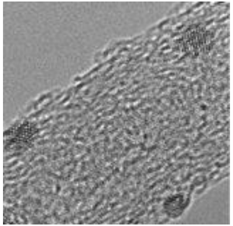
Non C dots 28



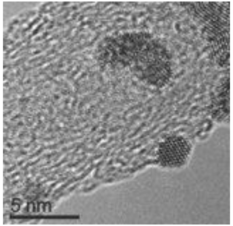
Non C dots 29



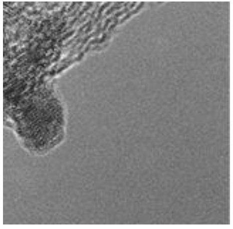
Non C dots 30



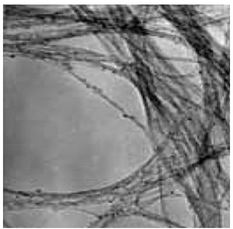
Non C dots 31



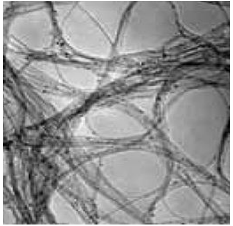
Non C dots 32



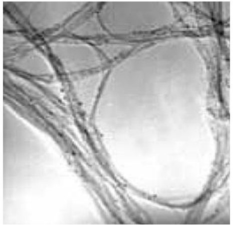
Non C dots 33



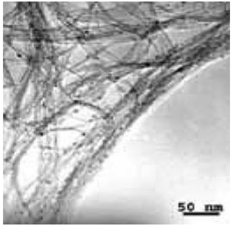
Non C dots 34



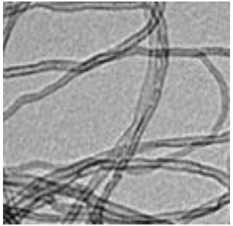
Non C dots 35



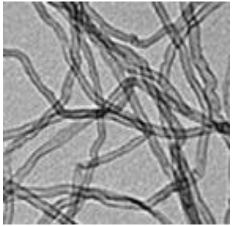
Non C dots 36



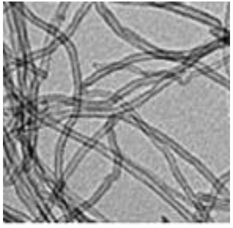
Non C dots 37



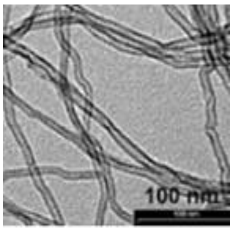
Non C dots 38



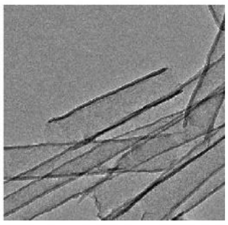
Non C dots 39



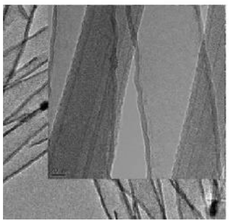
Non C dots 40



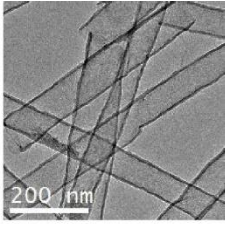
Non C dots 41



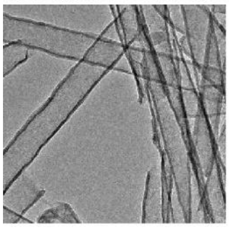
Non C dots 42



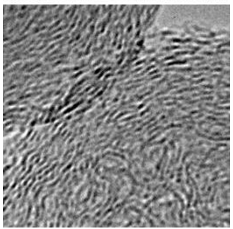
Non C dots 43



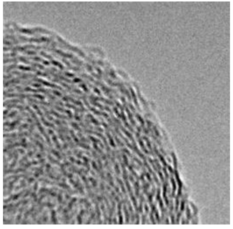
Non C dots 44



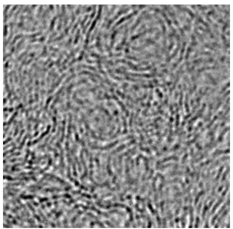
Non C dots 45



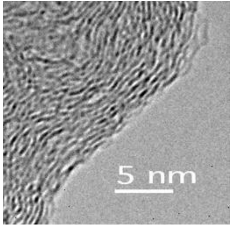
Non C dots 46



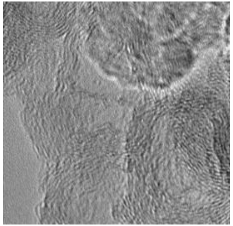
Non C dots 47



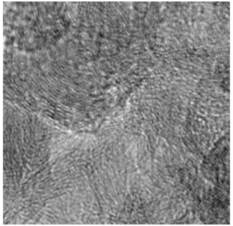
Non C dots 48



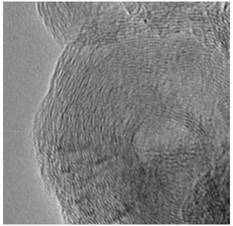
Non C dots 49



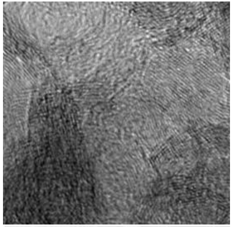
Non C dots 50



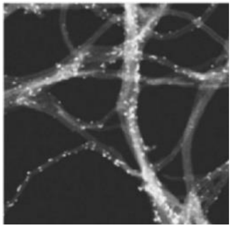
Non C dots 51



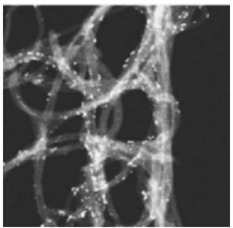
Non C dots 52



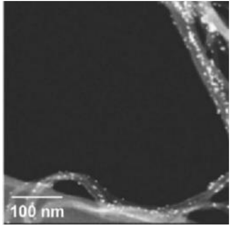
Non C dots 53



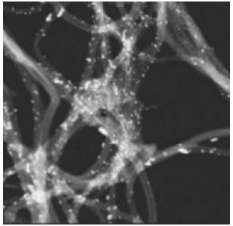
Non C dots 54



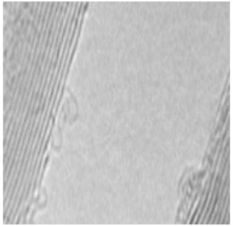
Non C dots 55



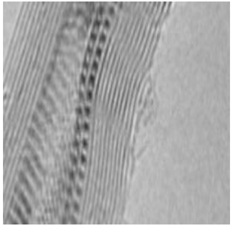
Non C dots 56



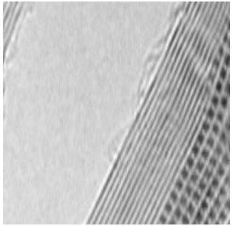
Non C dots 57



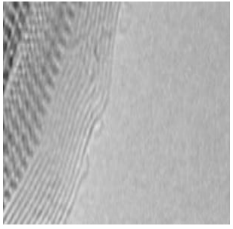
Non C dots 58



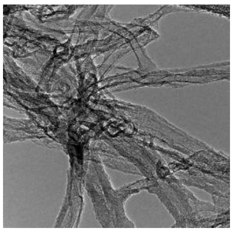
Non C dots 59



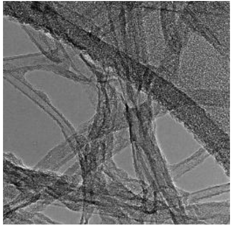
Non C dots 60



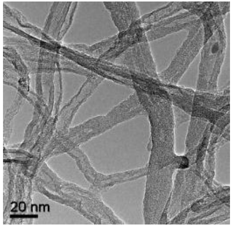
Non C dots 61



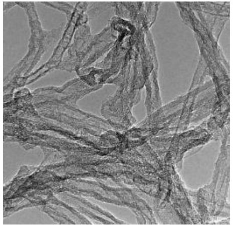
Non C dots 62



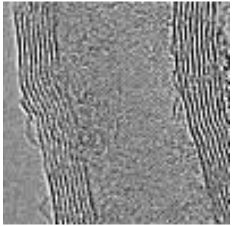
Non C dots 63



Non C dots 64

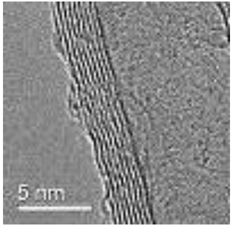


Non C dots 65

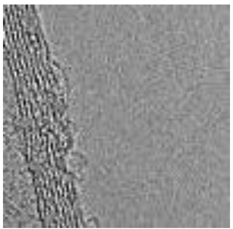


Non C dots 66

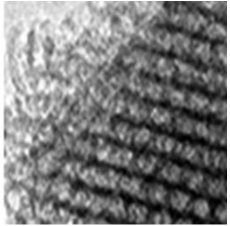
Non C dots 67



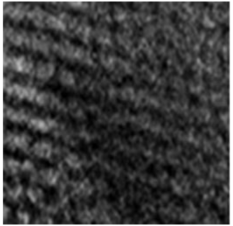
Non C dots 68



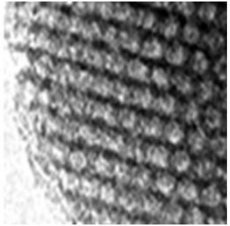
Non C dots 69



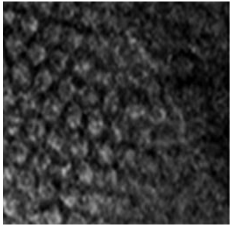
Non C dots 70



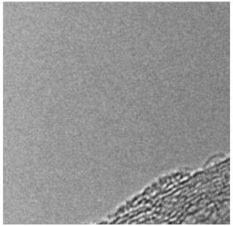
Non C dots 71



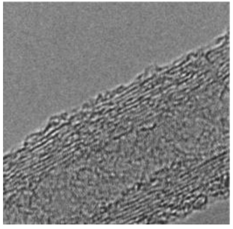
Non C dots 72



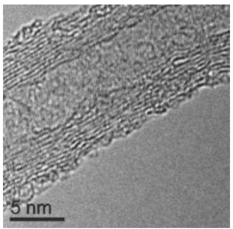
Non C dots 73



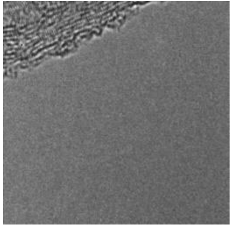
Non C dots 74



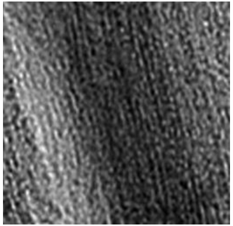
Non C dots 75



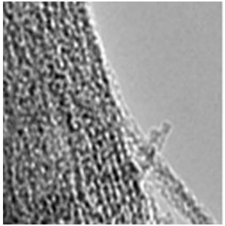
Non C dots 76



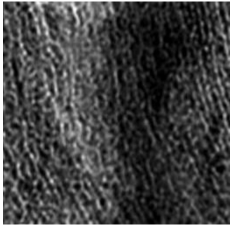
Non C dots 77



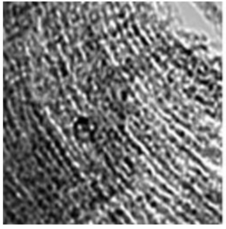
Non C dots 78



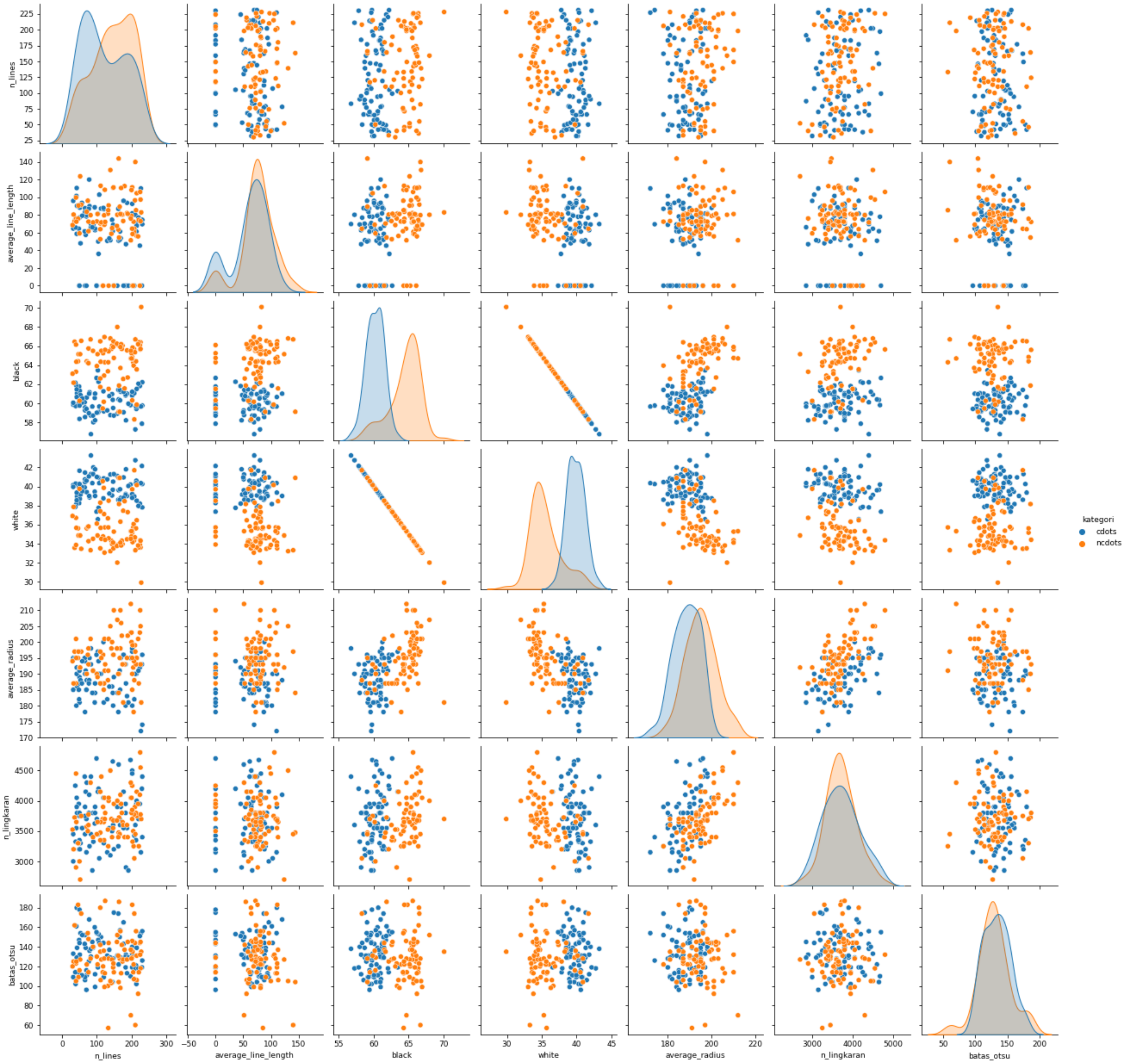
Non C dots 79



Non C dots 80



Lampiran C


n_lines
225
200
175
150
125
100
75
50
25
average_line_length
140
120
100
80
60
40
20
0
black
70
68
66
64
62
60
58
white
42
40
38
36
34
32
30
average_radius
210
205
200
195
190
185
180
175
170
n_lingkaran
4500
4000
3500
3000
batas_otsu
180
160
140
120
100
80
60
0
100
200
300
n_lines
−50
0
50
100
150
average_line_length
55
60
65
70
black
30
35
40
45
white
180
200
220
average_radius
3000
4000
5000
n_lingkaran
50
100
150
200
batas_otsu
kategori
cdots
ncdots