University of Moratuwa

ER4903: PLANT PERFORMANCE

ASSIGNMENT 1

Group No : D

Group Members:

ATHTHANAYAKE A.M.D.R. (170804J)
BEDDAGE B.C.D. (170805M)
JAYATHILAKA G.M.A.N.G.N.M.B. (170817C)
WEERASINGHE W.M.R.T. (170845J)

Department of Earth Resources Engineering

Contents

1.1 Introduction

Particle size distribution or PSD is a representation; a mathematical function,a visualization (a histogram) or a series of value; indicating relative number of particles according to their sizes. Among several particle size analysing techniques, image analysis is a sophisticated analytical approach that can reveal additional information on a sample. Image analysis' additional factors can help provide further insight into critical aspects that can affect a sample's dissolution, processing differences, flowability, and material handling concerns. (Kumara et al., 2012).

Image processing is the process of translating a physical image to a digital format and then conducting operations on it to improve it or extract relevant data.Images are viewed as two-dimensional signals by image processing systems that are processed using predetermined signal processing procedures (K H et al., 2021)

Different Particle Size Descriptors (PSD) can be described based on individual particle images in 2D that are obtained by image analysis. The equivalent circular area diameter is used to calculate the particle size descriptors to make comparisons with other approaches easier. Particle form factor definitions rely on the equivalent circular perimeter diameter, width W and length L (Li et al., 2005).

Because 2D imaging techniques have limitations for researching 3D shape and interior organization, techniques for non-invasive 3D imaging such as X-ray micro-computed tomography (CT) are becoming more popular, allowing researchers to investigate anatomical inner structures' 3D morphological characteristics without having to section them. The same information about the samples is not provided by X-ray CT. Differential X-ray absorption by the picture is the basis for X-ray tomography. Furthermore, X-ray CT may attain substantially better spatial resolutions than conventional CT. As a result, X-ray computed tomography (CT) has grown in popularity as a means of acquiring a thorough understanding of the 3D structures of plant tissues. The thickness, density, and sample's molecular structure are all factors that affect the contrast of X-ray CT images (Guntoro et al., 2019; Le et al., 2019)

1.2 Methodology

This part of report contain the methodology of image processing
Given data: X-ray Micro-CT images (Dicom) of a sample (sand and gravel)

Objectives:

1. Extract following data and visualize
   (a) Number of particles
   (b) Particle size,Sphericity (circularity in 2D) distribution
   (c) Average particle size

The image preprocessing, detail extraction (Particle Size Descriptors, Shape Factors) was done using python on Google Colab platform. Several python libraries, built-in modules, and tools were used.
(***Google Colab*** - an online Jupyter Notebooks environment from Google)

The Assignment was done with following stages.

1. Initiation of online Jupyter notebook
2. Image converting and preprocessing
3. Image processing and noise reduction
4. Detect particles and extract details
5. Calculate the parameters
6. Data Visualization

Summary of each stage of process.

Stage01: Initiation of online Jupyter notebook

First the Dicom image was renamed (as GroupD.dcm) using windows file Explorer and upload to the Google Drive (for sake of convenience, work with Colab). New Colab notebook opened, and import required python libraries, built-in modules, and tools.

Imported items are:

1. math
2. pydicom and its module (pydicom.data)
3. pandas (pd)
4. numpy (np)
5. Pillow (PIL) and its module (Image , ImageFilter)
6. skimage.exposure
7. OpenCV (cv2), since the cv2.imhow not working in colab - google.colab.patches import cv2_imshow
8. matplotlib and matplotlib.pyplot (plt)

after that the Google dives was mounted and give permission to Colab edit and read Google Drive content.

Stage02: Image converting and preprocessing

In this stage image was preprocessing done

1 Convert Dicom Image to JPG format

Dicom format image was converted to more convenience format, due to two reasons. First, only limited python libraries, or modules could handle .dcm files (ex:- pydicom) and Dicom format is not permeable for image processing and manipulation, therefore the Dicom image was converted to JPG format using pydicom and numpy operations. And the image was saved in Google Drive

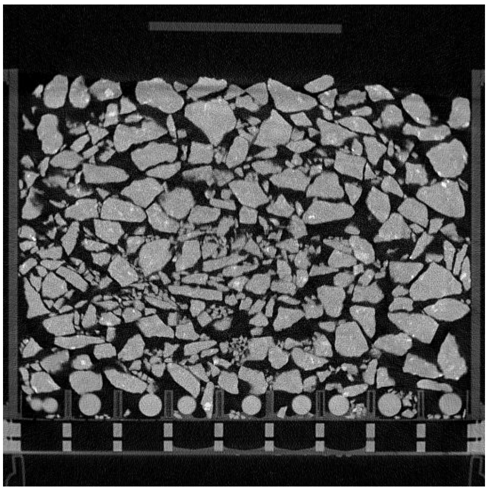


Image 01: Group D sample image (512×512,JPG)

2 Crop the image
The image; X-ray Micro-CT images of the sample is contain cross section of particles (sand and gravel), blank spaces and also some parts of containing chamber/ holder, therefore by cropping was done to capture all particles in the cropped window and get rid that unwanted parts as best as possible. The cropped image was saved in Google Drive

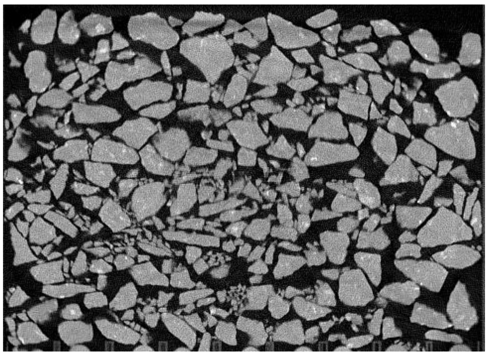


Image 02 : Cropped Image (485×350,JPG)

Stage03: Image processing and noise reduction

In this stage number of image processing techniques were used to reduce noise and also enhance target details of image. The tuning was done to find the balance of noise and detail in order to maximum object extraction.

1 Image gray-scale
The image was converted to Black and White using OpenCV, filter COLOR_BGR2GRAY.

2 Image Blurring
There is a two blurring techniques in OpenCV, GaussianBlur() and medianBlur(). Both are used in processing and the parameters are adjusted in tuning.
Gaussian Blur: Uses the Gaussian kernel. The height and width of the kernel should be a positive and an odd number.
Median Blur: The median of all the pixels of the image is calculated inside the kernel area. The central value is then replaced with the resultant median value. Median blurring is used when there are salt and pepper noise in the image.

Image 03 : Blurred Image (485×350,JPG)

3 Detect edges/ Optimal detector
The edges of image were detected and was used to take an idea of, how the particles are arranged and not used in direct processing procedure. Done with the Canny() method of

cv2 which implements the Canny edge detector. The Canny edge detector is also known as the optimal detector.

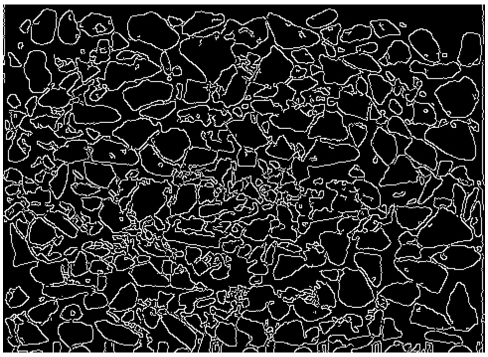


Image 04 : Detected Edges (485×350,JPG)

4 Invert color and adjust exposure
Invert the grayscale image using bitwise_not of OpenCV. This was done prior to exposer correction. The exposure adjustment was done as stretch the pixel values to required range in order to reduce noise but preserve details. The skimage rescale was used(skimage.exposure. rescale_intensity). Result image was generated as the in range (30,200) to out range (0,255).

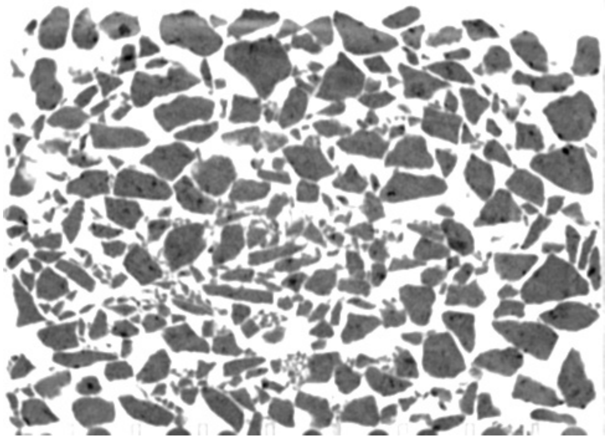


Image 05 : Adjusted/ re-scaled Image (485×350,JPG)

5 Noise reduction
The noise reduction was done with fastNlMeansDenoising(): Removes noise from a gray-scale image.

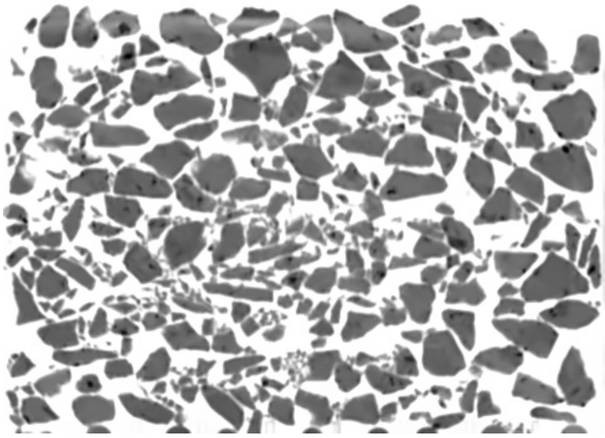


Image 06 : Processed Image (485×350,JPG)

Stage04: Detect particles and extract details

In this stage, the particles were detected and map their boundaries and locations.

1 extract contours
The Contours are the curves in an image that are joint together. The curves join the continuous points in an image. The purpose of contours is used to detect the objects. By using the findContours() of OpenCV, contours could extracted as array. Prior to execute find findContours(), it need to find threshold using threshold(). The detected contours could draw on the image using drawContours() method. The sensitivity of detection of contours is dependent on the image processing (how successfully reduce noise while preserving details) and the threshold. Therefore, the tuning was done in several times with adjusting parameters and processing sequence.
And the number of detected contours (excluding image frame contour) is assumed to be equal to the number of particles in the sample.

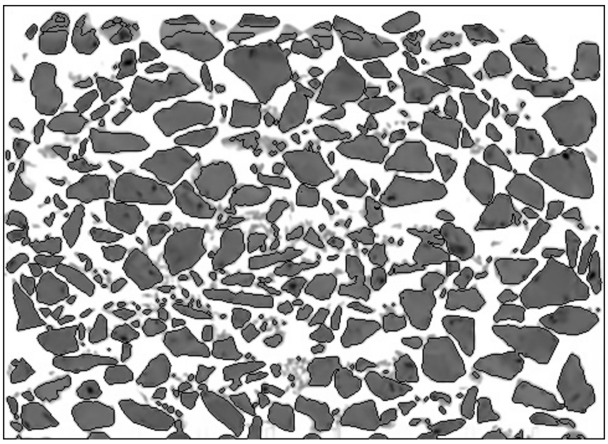


Image 07 : Detected Contours-thin black line (485×350,JPG)

2 Straight Bounding Rectangle
The bounding rectangle of particles (in this case around contours) was calculated using simple for loop and coordination manipulation and draw on image. This was done to find how accurate the particle detection was and as an aid for tuning process.

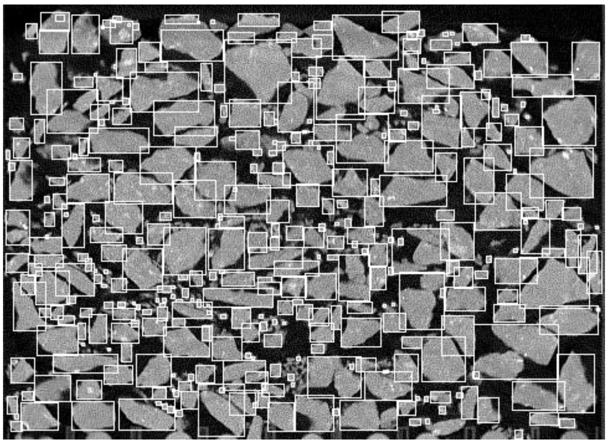


Image 08 : Straight Bounding Rectangle on particles (485×350,JPG)

3 Calculate geometric properties of contours
The area and the perimeter of contours were calculated using OpenCV modules; contourArea() and arcLength() respectively. The data was recorded as lists. These data was used for calculations of Particle Size Descriptors and Sphericity.

4 Rotated Bounding Rectangle
Bounding rectangle is drawn with minimum area, so it considers the rotation also. The function used is minAreaRect(). It returns a Box2D structure. Data extracted from this function was used to calculate the width and length of particles. And also the drawing rotated rectangle show the orientation of detected particles in more accurate way.

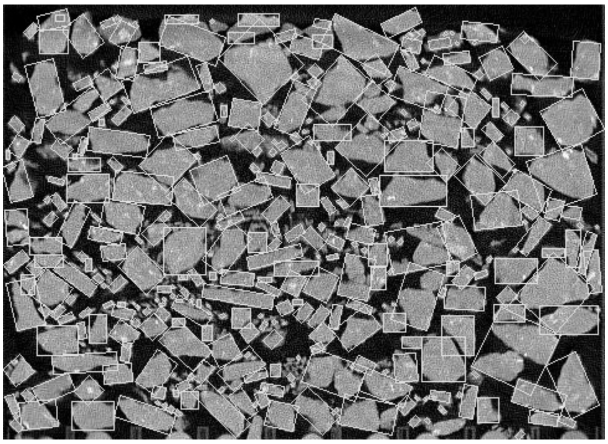


Image 09 : Rotated Bounding Rectangle on particles (485×350,JPG)

Stage05: Calculate the parameters

Following parameters were calculated using data extracted from previous stages.

1 Particle size parameters AKA article size descriptors

- i Equivalent Circular Perimeter Diameter
- ii Equivalent Circular Area Diameter
- iii Length
- iv Width
- v Minimum Enclosing Circle diameter

In estimating the particle size of particles, several particle size descriptors could use. First four parameters (Equivalent Circular Perimeter Diameter ,equivalent Circular Area Diameter , Length, and Width), were calculated using particle area, perimeter and Rotated Bounding Rectangle dimensions. The Minimum Enclosing Circle diameter is calculate using the OpenCV function minEnclosingCircle(). All details of each parameter are recorded in separate list. Note that the units if calculated paraments were converted from pixels from mm (using scale given from example, approximately; 1mm = 1.82px)

2 Sphericity
Calculated using perimeter P and the area A of particles. This data also recorded in a list.

3 Average particle size
Calculated using CSV file, excel functions

Stage05: Data Visualization

1. Export data
Data was exported as CSV using pandas (Display as DataFrame() and copy as CSV using Colab interactive tables)
2. Visualize data as histograms
Data was visualize using hist() function of Pillow with some modifications.

1.3 Algorithms Employed

Formulas used in calculations (Li ,2005)

Formulas

1 Sphericity
Sphericity express the deviation of an image shape from spherical, measured by the difference

between the image and the circle of diameter (A- area of Particle, P – Perimeter of particle)

$$S = \frac{4\pi A}{P^2} \tag{1.1}$$

2 Equivalent Circular Perimeter Diameter
Diameter of circle, which having same projected area as the particle. Calculate suing A- Area of particle (2D). Most common particle size estimator in 2D analysis.

$$D_a = 2\sqrt{\frac{A}{\pi}} \tag{1.2}$$

3 equivalent Circular Area Diameter
The circle of diameter has the same perimeter as the particle image silhouette, which can be calculated using the particle perimeter P.

$$D_p = \frac{P}{\pi} \tag{1.3}$$

4 Minimum Enclosing Circle diameter
The minimum Enclosing Circle is obtained from the OpenCV function minEnclosingCircle() and obtain the radius of Minimum Enclosing Circle and from that, the Minimum Enclosing Circle diameter could easily calculated.

$$D_{En} = radius \times 2 \tag{1.4}$$

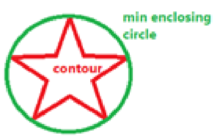


Image 10 : Minimum enclosing circle on contour (Retrieve from medium.com/analyticsvidhya)

Image processing algorithms

1. Image gray-scale
   Transformations within RGB space like adding/removing the alpha channel, reversing the channel order, conversion to/from 16-bit RGB color (R5:G6:B5 or R5:G5:B5), as well as conversion to/from grayscale using following transformation.

   $RGB[A]toGray : Y$ ¸ $0.299R + 0.587G + 0.114B$ (1.5)

   $Gray\ to\ RGB[A]$
   $: R$ ¸ Y, G ¸ Y, B ¸ Y, A ¸ $max(ChannelRange)$ (1.6)

2. Image Blurring
   (a) GaussianBlur()
   The density function of normal distribution is called Gaussian function. The two dimension format of Gaussian function is used to calculate weight of each pixel (Weight matrix) and from that each point multiplies its relevant weight value and obtain blur image. (two dimensional Gaussian function)

   $$G_{(x,y)} = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2} \tag{1.7}$$

   (b) medianBlur()
   The Median blur operation is similar to the other averaging methods. Here, the central element of the image is replaced by the median of all the pixels in the kernel area. This operation processes the edges while removing the noise.

3. edges/ Optimal detector
   A multi-stage algorithm, containing; Noise Reduction, Finding Intensity Gradient of the Image, Non-maximum Suppression and finally Hysteresis Thresholding. (https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html)

4. Invert color
   The bitwise NOT, or complement, is a unary operation that performs logical negation on each bit, forming the ones' complement of the given binary value. Bits that are 0 become 1, and those that are 1 become 0. In this case, the pixel values of image (grayscale) subjected to operation

   $$NOT\ x = 255 - x \tag{1.8}$$

5. Exposure (skimage.exposure.rescale_intensity)
   Return image after stretching or shrinking its intensity levels. The desired intensity range of the input and output, in_range and out_range respectively, are used to stretch or shrink the intensity range of the input image.

6. Noise reduction (fastNlMeansDenoising())
   Perform on grayscale images, denoising using Non-local Means Denoising algorithm (http://www.ipol.im/pub/algo/bcm_non_local_means_denoising) with several computational optimizations. Noise expected to be a gaussian white noise.

1.4 Results and Discussion

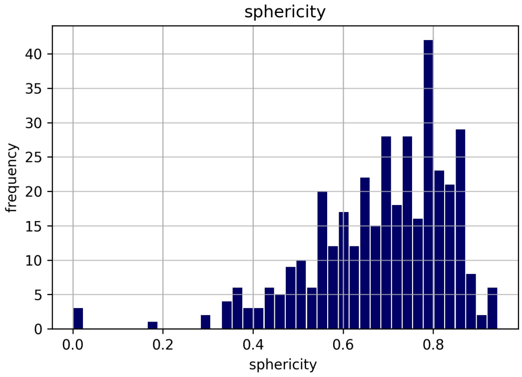
1. Number of particles

   Number of particles detect : 377

   From manual counting, average number of particles found was 210.5 and therefore, it can conclude that 377 may be represent acceptable estimation on number of particles found in sample.

2. Sphericity

   Sphericity is lies along the range of 0 to 1 and it indicate how spherical the particles are. The sphericity of particles are from 0 to 0.94 and average 0.686209 which implies, particles are non-spherical.

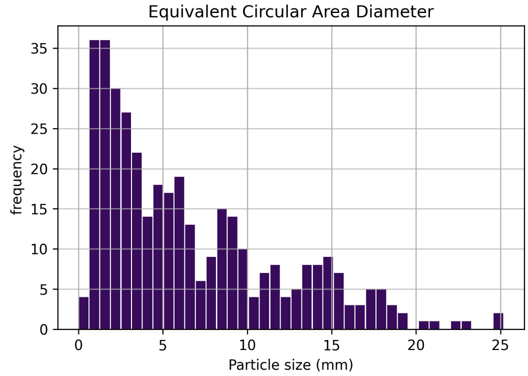


Graph 01 : Spericity of Particles (Histogram)

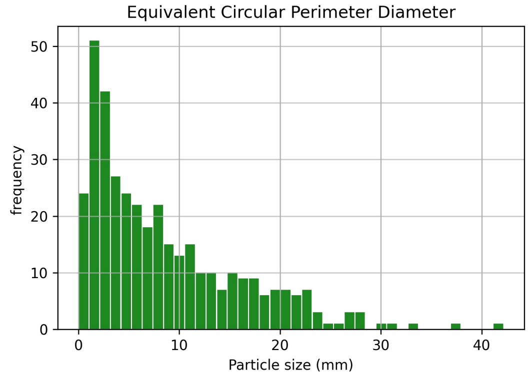
	Spericity
Mean	0.686209
Standard deviation	0.145516
Min	0
Max	0.943271

Table 01: Spericity of particles:

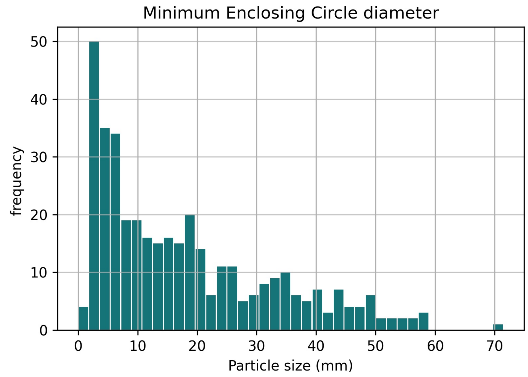
3. Particle size distributions



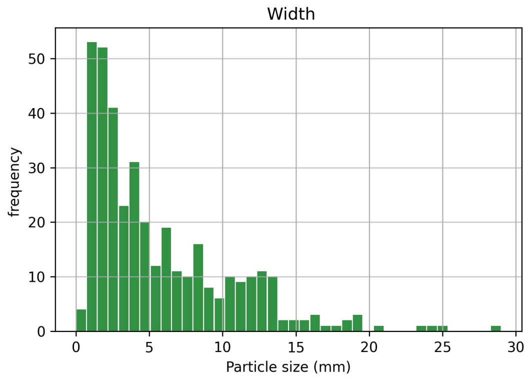
Graph 02 : Equivalent Circular Area Diameter of Particles (Histogram)



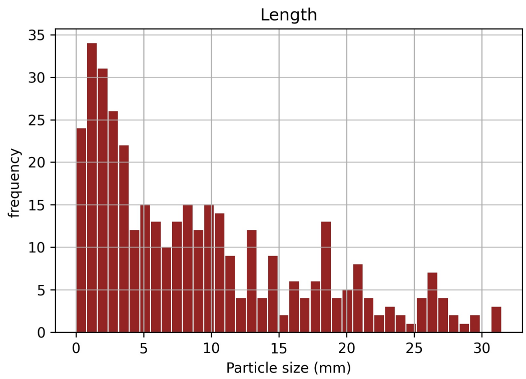
Graph 03 : Equivalent Circular Perimeter Diameter of Particles (Histogram)



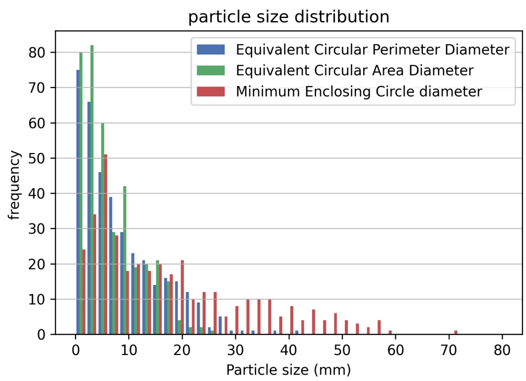
Graph 04 : Minimum Enclosing Circle diameter of Particles (Histogram)



Graph 05 : Width of Particles (Histogram)



Graph 06 : Length of Particles (Histogram)



Graph 07 : Comparison of each Parameter (multi-Histogram)

4. Average particle size

(in mm)	Mean	S.d	Range
EqP Dia	8.572	7.455	0 to 42.174
EqA Dia	6.652	5.316	0 to 25.184
Length	9.209	7.765	0 to 31.461
Width	5.629	7.765	0 to 28.995
Min Enc Dia	17.860	14.738	0.0002 to 71.47

Table02 : Particle Sizes,
EqP Dia = Equivalent Circular Perimeter Diameter(mm), EqA Dia =Equivalent Circular Area Diameter, Min Enc Dia =Minimum Enclosing Circle diameter, S.d = standard deviation

1.4.1 Discussion and Recommendations

the objectives og this assingment was to extract data (Particle Size Descriptors, Shape Factors) from given X-ray Micro-CT images (Dicom) of a (sand and gravel) sample. It was done with python on Google Colab platform.

There were several challengers, face during the assignment. First, the python image processing and reporting platform (LaTeX) were not familiarized and difficult to master it with limited time period. But due to good amount of documentation, web support and tutorials (Git hub, stackoverflow etc.) this challenge was easily overcome. Another challenge in image processing, reduce noise while preserving the details of image. This was done with tuning the processing parameters, more precisely a trail and error method.

Considering the advantages of this image processing method is, it can extract and calculate fast than traditional methods. And also possible to work with batch wise (number of images) and obtain result with very short time. And may be give excellent result for loosely packed particles.

Main disadvantages are, the method can't detect particles with coagulated or very closely packed particles. And tuning for optimizing particle detection is time consuming.

As a recommendation, this method could further be developed by using machine learning technique (Semantic Segmentation or instance segmentation) and with much deep neuron network. The process may robust and fast and may not required hard tuning.

Google Colab Notebook available at: https://drive.google.com/file/d/1A0RMjEiwgaiNG8PLmqtoaLxo-59oRsNR/view?usp=sharing

1.5 References

1 Buades,A., Coll,B,. and Morel,J-M, (2011), Non-Local Means Denoising, Image Processing On Line, 1 pp. 208–212. https://doi.org/10.5201/ipol.2011.bcm_nlm

2 Guan, Y., Zhou, F., Zhou, J. (2019). Research and Practice of Image Processing Based on Python. Journal of Physics: Conference Series, 1345(2). https://doi.org/10.1088/1742-6596/1345/2/022018

3 Guntoro, P. I., Ghorbani, Y., Koch, P. H., Rosenkranz, J. (2019). X-ray microcomputed tomography (µct) for mineral characterization: A review of data analysis methods. Minerals, 9(3), 20–26. https://doi.org/10.3390/min9030183

4 K H, M. A., B E, M., A S, M. C. (2021). Python - Based Image Processing. International Journal of Scientific Research and Management, 9(11), 635–638. https://doi.org/10.18535/ijsrm/v9i11.ec03

5 Kumara, G. H. A. J. J., Hayano, K., Ogiwara, K. (2012). 290-295-1261-Kumara-Sept-2012. 1999.

6 Kumari, R Rana. N (2015). Particle Size and Shape Analysis using Imagej with Customized Tools for Segmentation of Particles. International Journal of Engineering Research And, V4(11), 247–250. https://doi.org/10.17577/ijertv4is110211

7 Le, T. D. Q., Alvarado, C., Girousse, C., Legland, D., Chateigner-Boutin, A. L. (2019). Use of X-ray micro computed tomography imaging to analyze the morphology of wheat grain through its development. Plant Methods, 15(1), 1–19. https://doi.org/10.1186/s13007-019-0468-y

8 Li, M., Wilkinson, D., Patchigolla, K. (2005). Comparison of particle size distributions measured using different techniques. Particulate Science and Technology, 23(3), 265–284. https://doi.org/10.1080/02726350590955912

9 Particle Size and Shape Analysis. Retrieved 14 April 2022, from https://www.particletechlabs.com/analytical-testing/particle-size-distribution-analyses/particle-size-and-shape-analysis

1.6 Appendix

1.6.1 Colab Notebook

GroupD-PP_As1

April 15, 2022

1 Appendix 01 : Colab Notebook

Objectives: extract the following details,

- Number of particles
- Particle size, sphericity (circularity in 2D) distribution
- Average particle size List item

According to given example image,the unit conversion taken as 1mm = 1.82 px

1.1 Initiation of Notebook

Import stuff and mount with source

[34]:
```
pip install pydicom
```

```
Requirement already satisfied: pydicom in /usr/local/lib/python3.7/dist-packages
(2.3.0)
```

[35]:
```
pip install pillow
```

```
Requirement already satisfied: pillow in /usr/local/lib/python3.7/dist-packages
(7.1.2)
```

[36]:
```
#Upload Required python pacakages and libraries

import math
import pydicom
import pandas as pd
import PIL
from PIL import Image , ImageFilter

import pydicom.data
import numpy as np
import skimage.exposure

import cv2
from google.colab.patches import cv2_imshow
```

```python
import tensorflow as tf
import keras

import matplotlib.pyplot as plt
%matplotlib inline
```

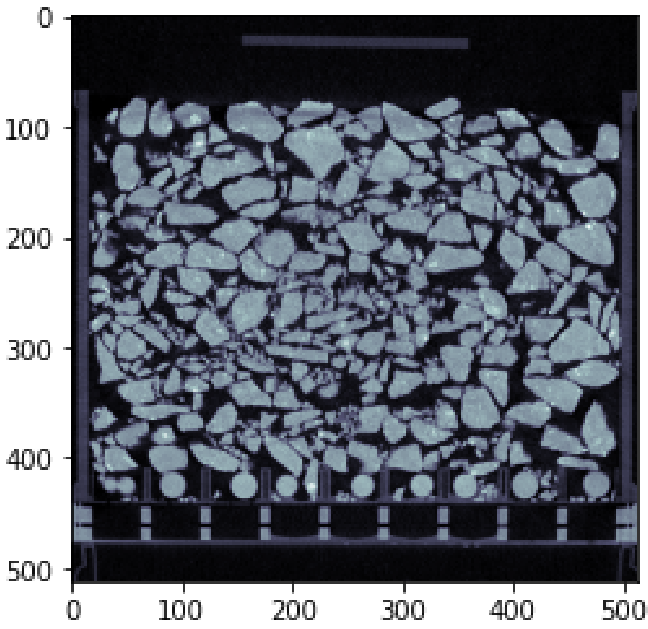
[37]:
```python
#maount with my Google Drive

from os import chdir
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

1.2 Dicom image convertion and pre-processing

[38]:
```python
#Open/ Dispaly the dcm file,
#ds = pydicom.dcmread(filename)
ds = pydicom.dcmread('/content/drive/MyDrive/PlantPerform_A01/Final/GroupD.dcm')


plt.imshow(ds.pixel_array, cmap=plt.cm.bone)  # set the color map and show the␣
 ↪dicom
#plt.show()
```

[38]:
```
<matplotlib.image.AxesImage at 0x7ff301711f10>
```



[39]:
```
#Convert dcm to jpg (done)
#Crope the Image (done)
#Extract number of particles
#Estimate parameters
#Avg PS est.
```

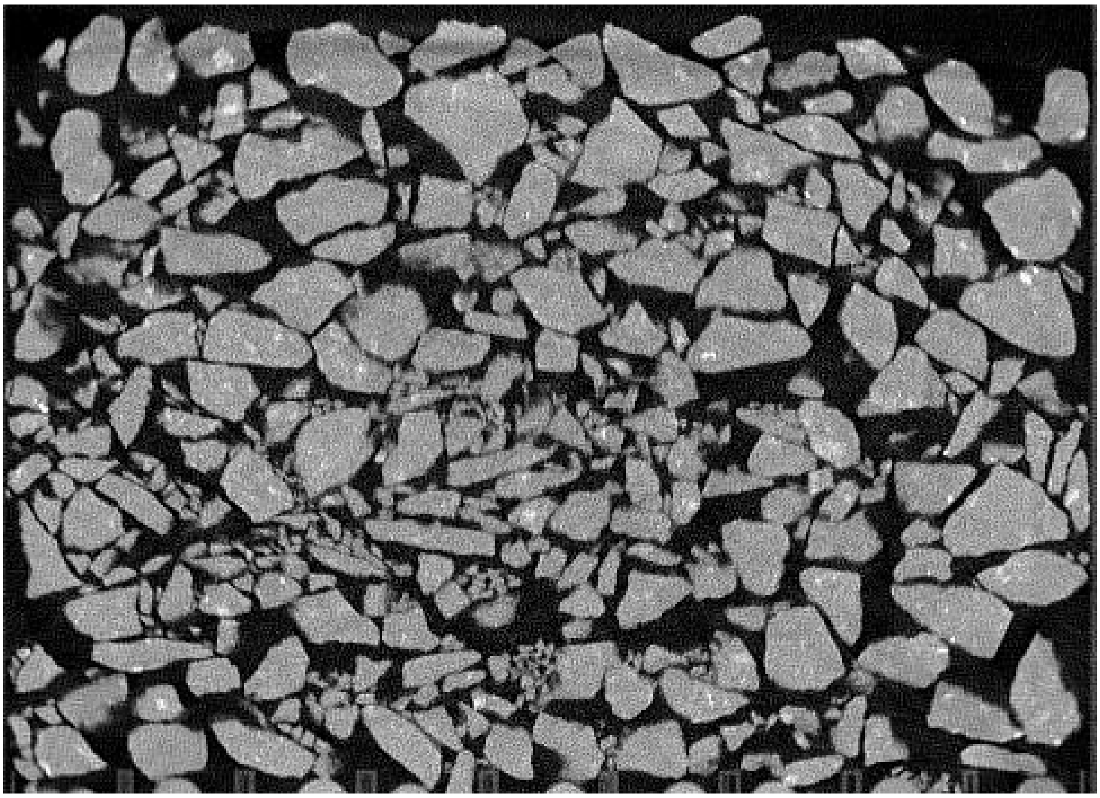
[40]:
```
#Convert dicom to jpg format (pydicom and np)

ds = pydicom.dcmread('/content/drive/MyDrive/PlantPerform_A01/Final/GroupD.dcm')

new_image = ds.pixel_array.astype(float)


scaled_image = (np.maximum(new_image, 0) / new_image.max())*255.0

scaled_image = np.uint8(scaled_image)
final_image = Image.fromarray(scaled_image)
```

[41]:
```
final_image.save('/content/drive/MyDrive/PlantPerform_A01/Final/GroupD.jpg')
```

[42]:
```
#Display the crated jpg image (512*512, using PIL)

#from IPython.display import Image
#Image(filename='/content/drive/MyDrive/PlantPerform_A01/Final/GroupD.jpg',␣
↪width=512,height=512)
```

[43]:
```python
#Crop the Jpg as all minerals include (Using OpenCV)


img = cv2.imread('/content/drive/MyDrive/PlantPerform_A01/Final/GroupD.jpg')
print(img.shape) # Print image shape
#cv2_imshow(img)

#print('Original image \n')

# Cropping an image
#y:y+h, x:x+w, the pixels from 70 to 420 (ydir) and 15 to 500 (xdir) is␣
↪extracting
cropped_image = img[70:420,15:500]
# Display cropped image
cv2_imshow(cropped_image)

print('Cropped image \n')

#Save cropped image

cv2.imwrite('/content/drive/MyDrive/PlantPerform_A01/Final/Crop_GroupD.jpg',␣
↪cropped_image)
```

```
(512, 512, 3)
```



```
Cropped image
```

[43]: True

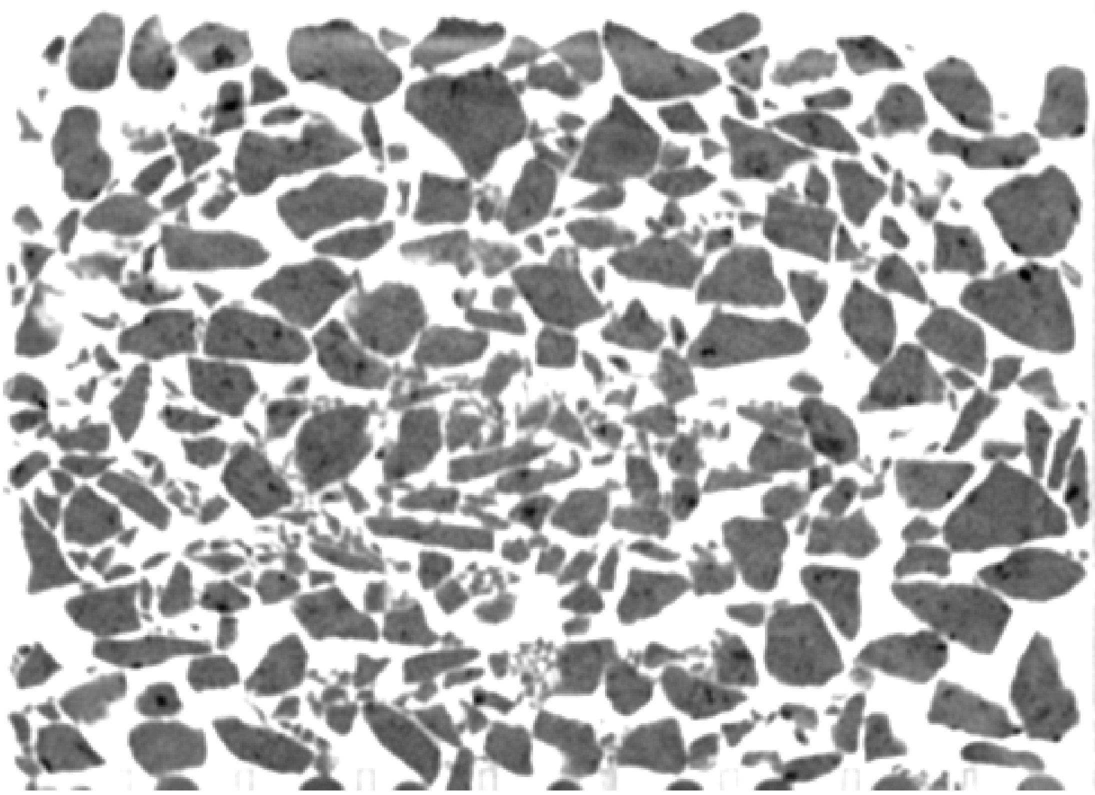
1.3 The JPG image Processing (grayscale, Blurring etc.)

[44]:
```python
#Extract number of particles

cp_img =cv2.imread('/content/drive/MyDrive/PlantPerform_A01/Final/Crop_GroupD.
↪jpg')


#Test 01 - procedure
#convert to grayscale, apply image burrer, extract edges etc. then find a␣
↪trained model to cpounting or make amodel to counting

#01 cov grascale

gray_img = cv2.cvtColor(cp_img, cv2.COLOR_BGR2GRAY)
#cv2_imshow(gray_img)

print('Gray Image \n')

#02 Blurring 1. Gaussing (not working) , 2. Median Blur

#blur_img = cv2.GaussianBlur(gray_img, (7,7), 0)
blur_img = cv2.medianBlur(gray_img,3)
#cv2_imshow(blur_img)

print('Blurred Image \n')

#03 Detect edges

edg_img = cv2.Canny(blur_img,100,200)
#cv2_imshow(edg_img)

print('Detected edges of Image \n')
```

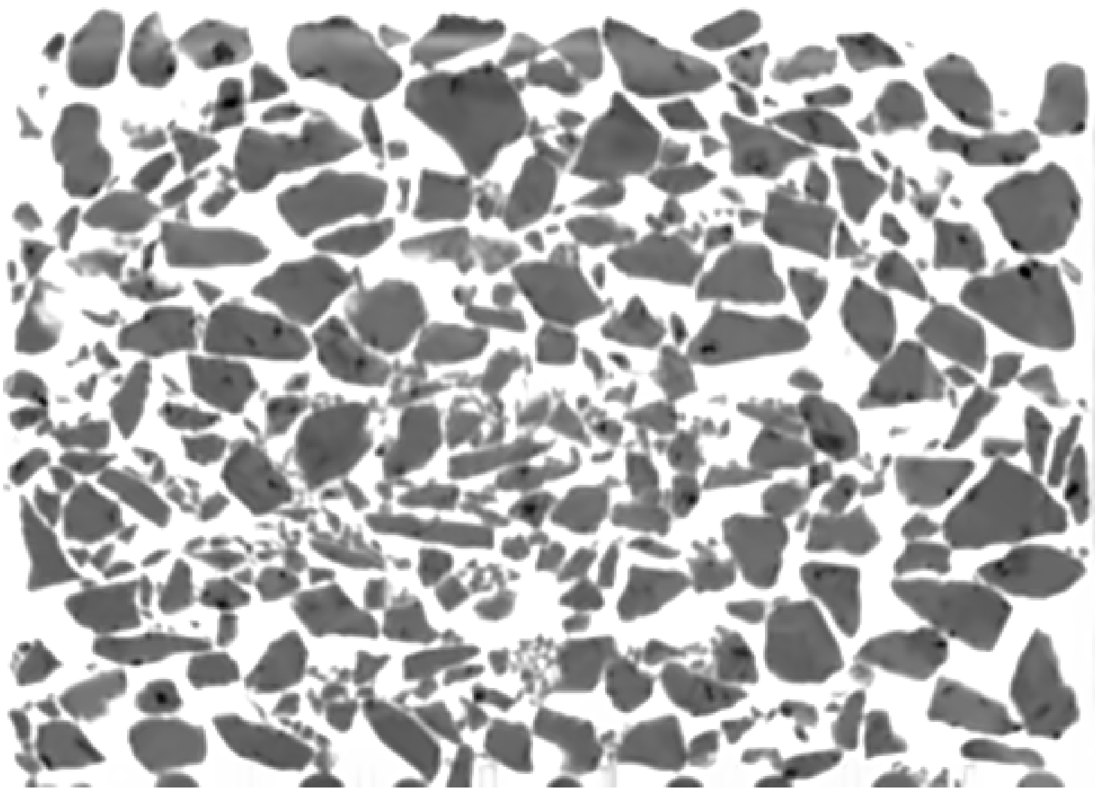
```
Gray Image

Blurred Image
```

Detected edges of Image

[45]:
```
# reduce noice and adjust he exposure

# blur threshold image
temp0 = cv2.GaussianBlur(gray_img, (0,0), sigmaX=1, sigmaY=1, borderType = cv2.
 ↪BORDER_DEFAULT)
temp1 = cv2.bitwise_not(temp0)
#result =blur

# stretch so that 160 -> 255 and 70 -> 0
# C = A*X+B # calc error//
# 255 = A*160+B
# 0 = A*225+B
# Thus A=2 and B=-127.5
#aa = a*2.0-255.0 does not work correctly, so use skimage
result = skimage.exposure.rescale_intensity(temp1, in_range=(30,200),␣
 ↪out_range=(0,255)).astype(np.uint8)
cv2_imshow(result)
```



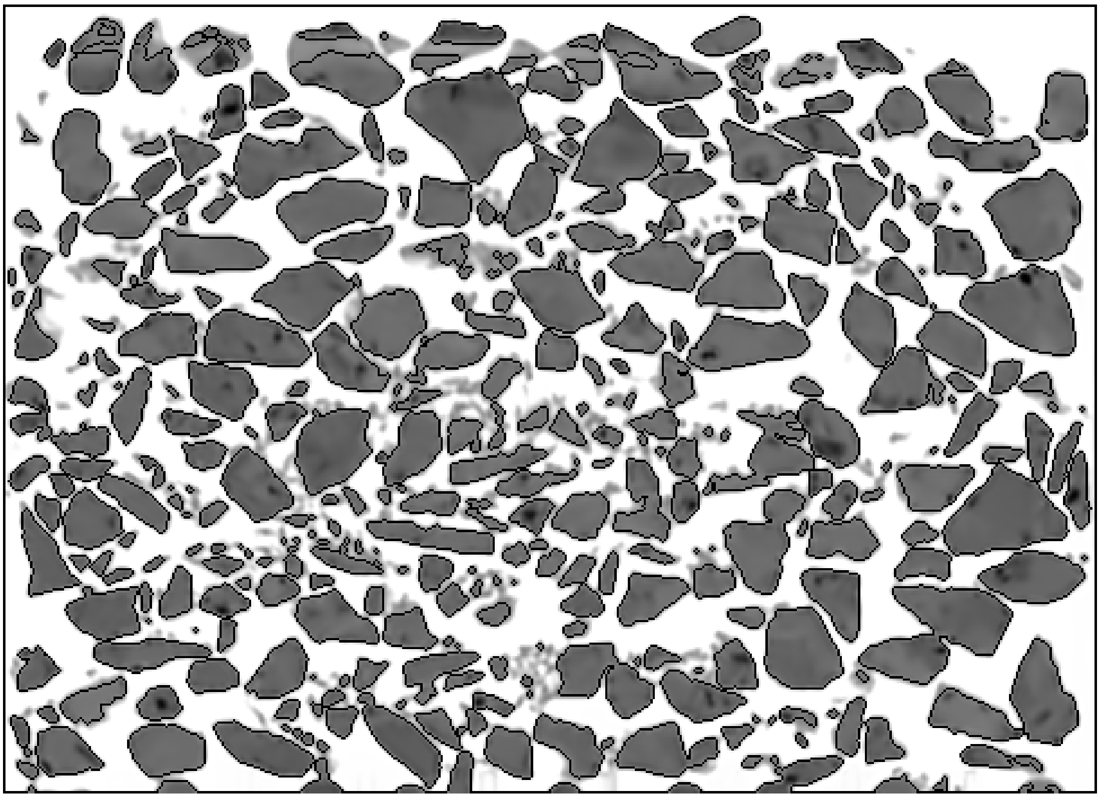
[46]:
```
#Reduce noice

Cor_img =result
#Cor_img = cv2.bitwise_not(result)
#Cor_img = cv2.bitwise_not(gray_img)
temp = cv2.fastNlMeansDenoising(Cor_img,2,8)
#inv_img= cv2.medianBlur(temp,3)
inv_img= temp

#cv2_imshow(result)
#cv2_imshow(Cor_img)
cv2_imshow(inv_img)
print('Preporcessed image')

#cv2.imwrite('/content/drive/MyDrive/PlantPerform_A01/Corr_edge_samp1.jpg',␣
↪Cor_img)
```



Preporcessed image

[47]:
```
#extract contures #RETR_LIST or #RETR_TREE

#cont_img1 = result.copy()
```

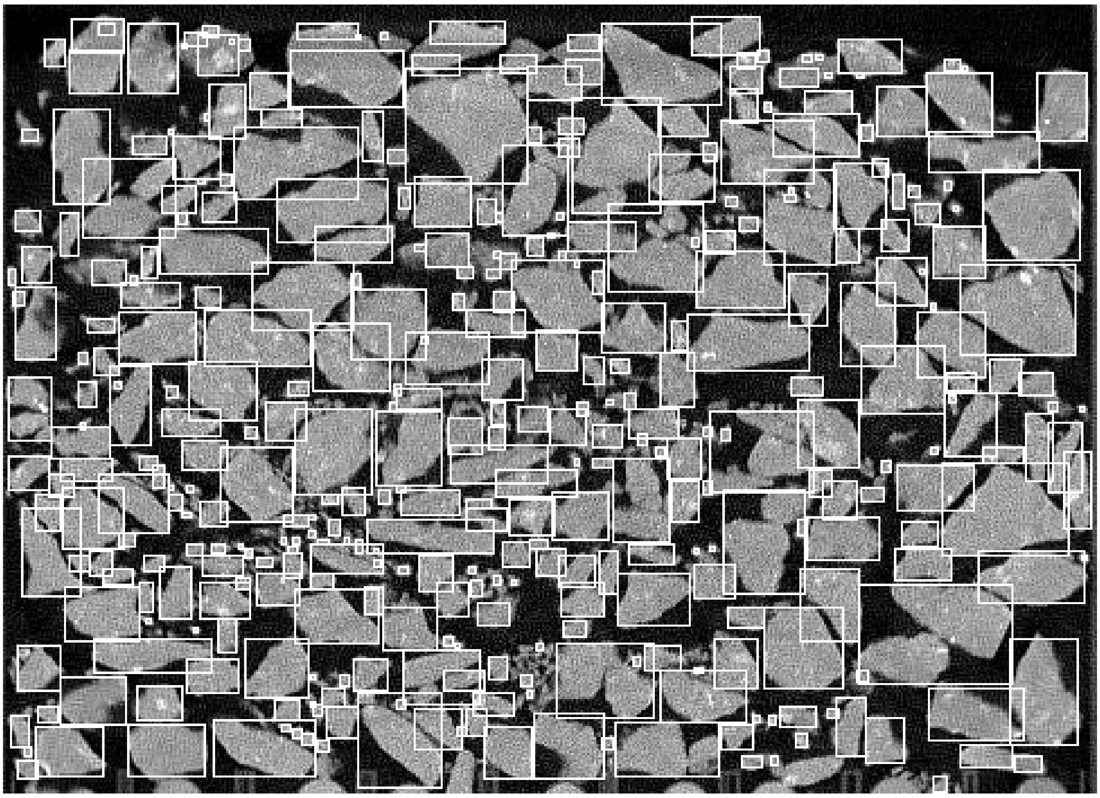
```python
#cont_img1 =edg_img.copy()
cont_img1 =inv_img.copy()

retval, thresh = cv2.threshold(cont_img1,127, 255, 0)
#retval, thresh = cv2.threshold(cont_img1,127,200,cv2.THRESH_BINARY)

img_contours0, _ = cv2.findContours(thresh, cv2.RETR_LIST, cv2.
↪CHAIN_APPROX_SIMPLE)
cv2.drawContours(cont_img1, img_contours0, -1, (0, 0, 0))

#remove the image boundry contour
img_contours0 = img_contours0[:-1]


cv2_imshow(cont_img1)
```



[48]:
```python
len(img_contours0)
```

[48]: 377

[49]:
```
#for loop to draw box on all contours

img_cc=gray_img.copy()
Square_list=[]

for i in range (0, len(img_contours0)):
  list1= img_contours0[i]

  max = np.amax(list1, axis=0)
  min = np.amin(list1, axis=0)
  maxL = max.tolist() + min.tolist()
  cc = (maxL[0][0],maxL[1][1])
  cp = (maxL[1][0],maxL[0][1])
  temp = [cc,cp]
  Square_list.append(temp)

  cv2.rectangle(img_cc,cc, cp, (255, 255, 251))
  #cv2.rectangle(img_cc,max,min, (255, 255, 251))
cv2_imshow(img_cc)
```



[50]:
```
#img_contours
print('Number of of contours found ' +str(len(img_contours0)) + '\n')

# lets assue that; countours  == a boundry of partical
#thefore no of particles

print('Number of particles  ' +str(len(img_contours0)))
```

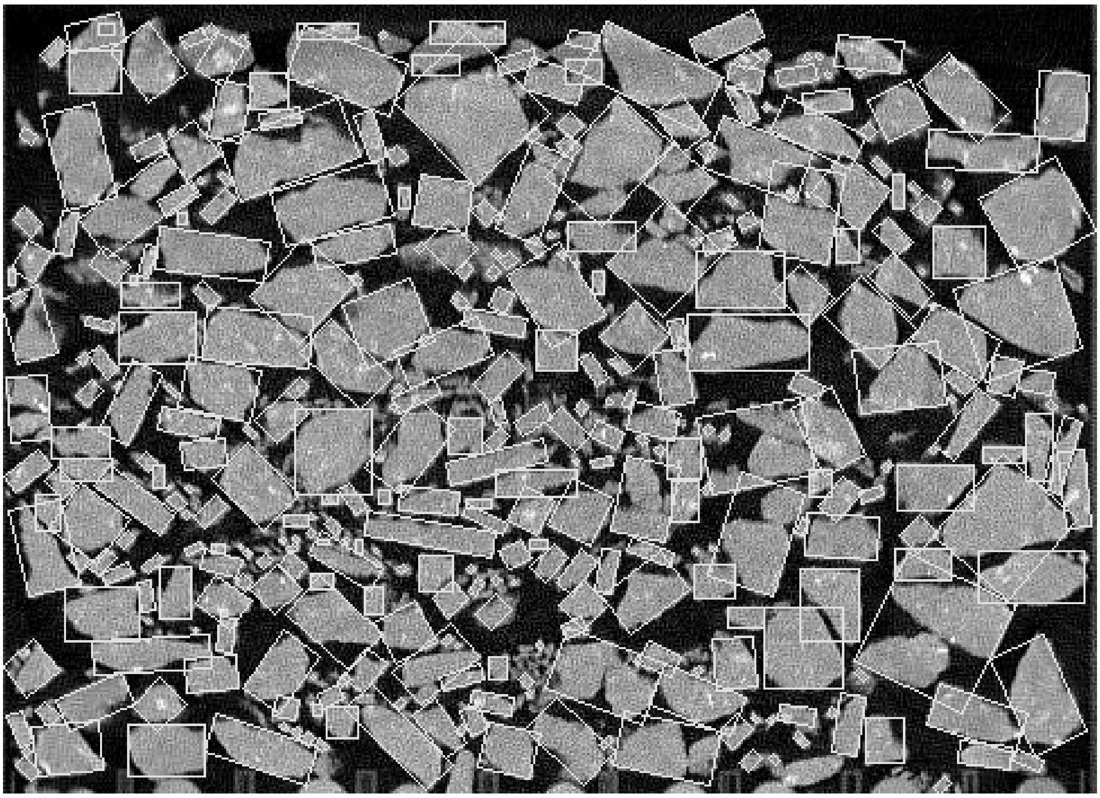
```
Number of of contours found 377

Number of particles  377
```

[51]:
```
#Calculate the area of contours

c_Area=[]

for i in range(len(img_contours0)):
  area = cv2.contourArea(img_contours0[i])
  c_Area.append(area)

#c_Area
```

[52]:
```
# Calculate Contour Perimeter

c_Peri=[]

for i in range(len(img_contours0)):
  perimeter = cv2.arcLength(img_contours0[i],True)
  c_Peri.append(perimeter)
#c_Peri
```

[53]:
```
# Find and draw rotated rectangle

Round_rec = gray_img.copy()

Rect_area =[]

for i in range(len(img_contours0)):
  rect = cv2.minAreaRect(img_contours0[i])
  box = cv2.boxPoints(rect)
  box = np.int0(box)
  cv2.drawContours(Round_rec,[box],0,(225,225,255),1)

  # obtain area of rotated rect.
  R_area = cv2.contourArea(box)
  Rect_area.append(R_area)
cv2_imshow(Round_rec)
```

```
print('Detected particles')
```



```
Detected particles
```

1.4 Following data were extrcted

1. No of contours/ particles detected = 377 2.The area of particles (using contour area), find in c_Area list 3.The perimeter of particles (using contour perimeter), find in c_Peri list 4.The area of bounding rectangle with minimum area around particles (using contour otated Rectangle and area), find in Rect_area list

1.5 Parameter Calclulation

[54]:
```
#Cacl circularity , ircularity (or isoperimetric quotient), a function of the␣
↪perimeter P and the area A: C = 4 A/P^2 (Sphericity S)
#no need for unit conversion

c_circ =[]

for i in range(len(img_contours0)):
```

```
  if c_Peri[i] ==0 :
     circularity = 'n/A'
  else:
    Sq_para =(c_Peri[i])*(c_Peri[i])
    circularity = (4*(np.pi)* c_Area[i])/(Sq_para)
  c_circ.append(circularity)
```

1.5.1 the particles particle size can measured in follwoing methods

Particle size parameters AKA article size descriptors

1. Equivalent Circular Perimeter Diameter (EqPD)
2. equivalent Circular Area Diameter (EqAD)
3. Length L and Width W
4. Minimum Enclosing Circle diameter

[55]:
```
# Calc Equivalent Circular Perimeter Diameter, EqPD = Perimeter/
# unit conversion 1mm =1.82px
EqPD =[]

for i in range(len(img_contours0)):
  Temp = (c_Peri[i]/(np.pi))
  EQP_Diameter = Temp/1.82
  EqPD.append(EQP_Diameter)
```

[56]:
```
# Calc Equivalent Circular Area Diameter, EqAD = 2*√(Area/ )
#unit conversion 1mm =1.82px
EqAD =[]

for i in range(len(img_contours0)):
  Temp= 2*(math.sqrt(c_Area[i]/(np.pi)))
  EQA_Diameter = Temp/1.82
  EqAD.append(EQA_Diameter)
```

[57]:
```
#Calc width and height Using bounding rectangle ,width and Length
# unit conversion 1mm =1.82px
Wid_Len =[]

for i in range(len(img_contours0)):
  rect = cv2.minAreaRect(img_contours0[i])
  Wid_Len.append(rect[1])

#Append width and length separate lists

Width =[]
Length =[]
```

```
for i in range(len(Wid_Len)):
  if Wid_Len[i][0] > Wid_Len[i][1] :
    W_pix = ((Wid_Len[i][1])/1.82)
    L_pix =((Wid_Len[i][0])/1.82)
    Width.append(W_pix)
    Length.append(L_pix)
  else:
    W_pix = ((Wid_Len[i][0])/1.82)
    L_pix =((Wid_Len[i][1])/1.82)
    Width.append(W_pix)
    Length.append(L_pix)
```

[58]:
```
#Minimum Enclosing Circle, circle which completely covers the object with␣
 ↪minimum area

MiEnD =[]
for i in range(len(img_contours0)):
  (x,y),radius = cv2.minEnclosingCircle(img_contours0[i])
  Min_enc_Dia = 2*radius
  MiEnD.append(Min_enc_Dia)
```

1.6 Following data were extrcted (377)

1. Equivalent Circular Perimeter Diameter (EqPD) record in list **EqPD**
2. Equivalent Circular Area Diameter (EqAD) record in list **EqAD**
3. Length L and Width W record in list **Wid_Len** (separately in lists of width and Length)
4. Minimum Enclosing Circle diameter record in list **MiEnD**
5. Sphericity of Particles are in list of **c_circ**

1.7 Visualizing data

[59]:
```
#Generate table contatining all extracted data

lst0 = c_circ
lst1 = EqPD
lst2 = EqAD
lst3 = Wid_Len
lst31 = Length
lst32 =Width
lst4 = MiEnD

percentile_list = pd.DataFrame(
    {'sphericity':lst0,   'Equivalent Circular Perimeter Diameter(mm)': lst1,
     'Equivalent Circular Area Diameter (mm)': lst2,
     'Length and Width (px)': lst3,'Length (mm)':lst31 ,'Width (mm)':lst32,
```

```
    'Minimum Enclosing Circle diameter (mm)':lst4
    })

#percentile_list
#The generated data copied to .csv
```

[60]:
```
#Histrogram saving fuction
def hisrogram_save(figname):
  fold = ('/content/drive/MyDrive/PlantPerform_A01/Final/Grap_')
  format = '.jpg'
  dir=fold+figname+format
  #print(dir)
  plt.savefig(dir,dpi=300)
```

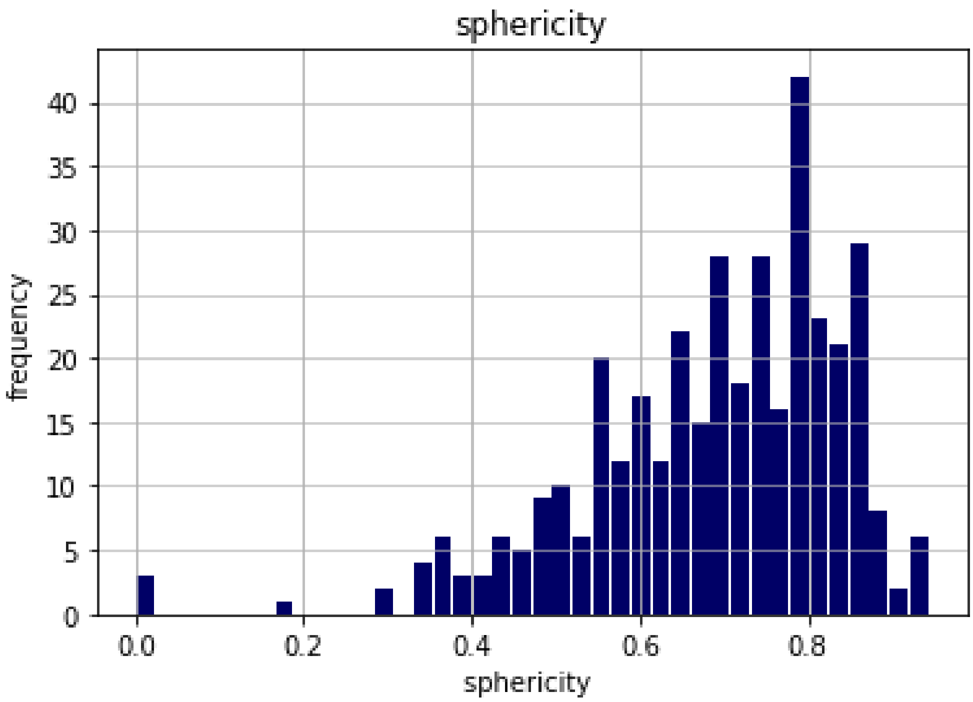
[61]:
```
#define fuction to display histrogrames
def Histro(datalist,colrs,title,xlab,ylab,figname):
  size, scale = 1000, 10
  commutes = pd.Series(datalist, dtype='float')

  #bin = np.linspace(0, 80, 30)
  bin=40

  commutes.plot.hist(grid=True, bins=bin, rwidth=0.9,color=colrs)
  plt.title(title)
  plt.xlabel(xlab)
  plt.ylabel(ylab)
  plt.grid(axis='y', alpha=0.75)
  hisrogram_save(figname)
  plt.show()
```

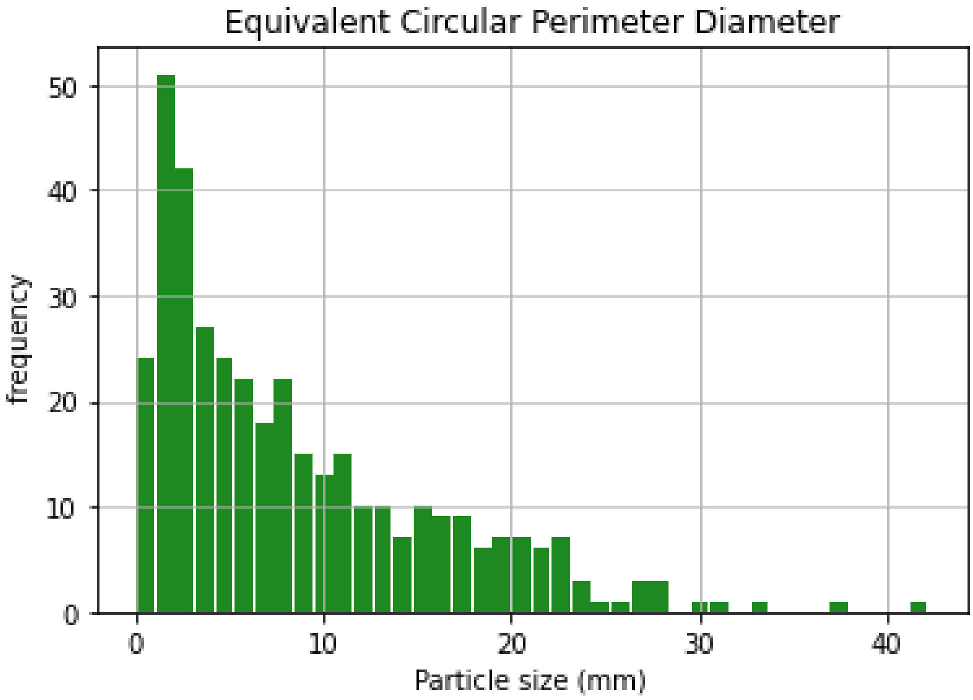
[61]:
```

```

[62]:
```
#remove 'n/A' in c_circ
c_circ_r = np.array(c_circ)
c_circ_r = np.where(c_circ_r == 'n/A', 0, c_circ_r)
c_circ_r= c_circ_r.tolist()
```

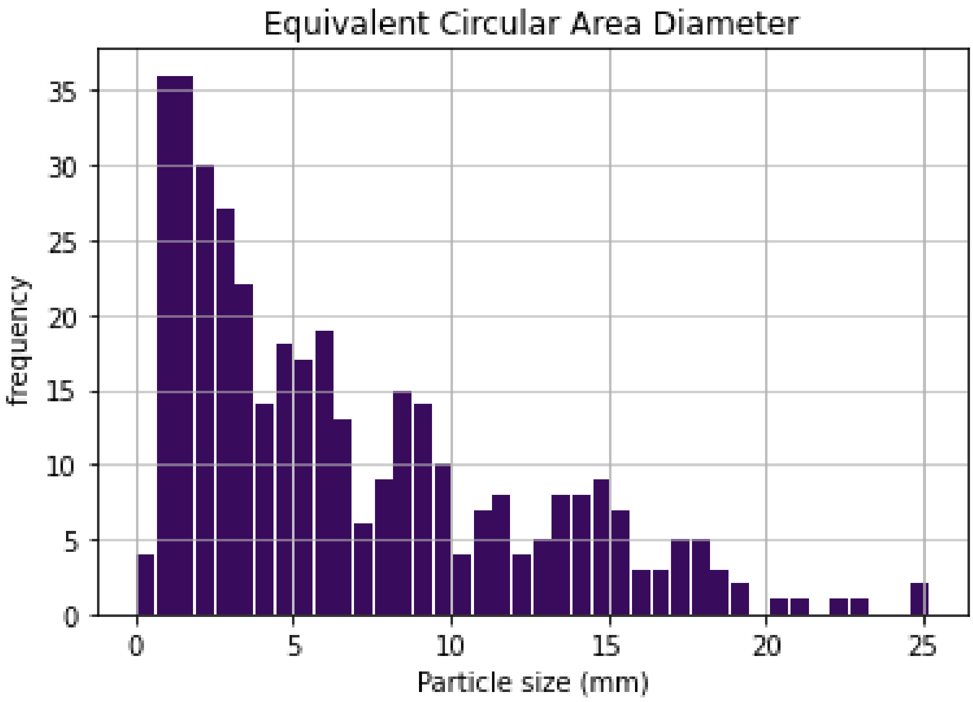
[63]:
```
#Visualize data

Data_list = [c_circ_r,EqPD ,EqAD ,Width, Length  , MiEnD]
Titl_list = ['sphericity','Equivalent Circular Perimeter Diameter', 'Equivalent␣
 ↪Circular Area Diameter',' Width','Length','Minimum Enclosing Circle␣
 ↪diameter']
colour_list = ['#000066','#1D8920','#390B5C','#319244','#942323','#157477']
xlab_list = ['sphericity ','Particle size (mm)','Particle size (mm)','Particle␣
 ↪size (mm)','Particle size (mm)','Particle size (mm)']
```

```
ylab_list = ['frequency','frequency ','frequency ','frequency ','frequency␣
↪','frequency ']
Graph = ['sphericity','EqP_Dia', 'EqA_Dia',' Width','Length','MinEnCir_Dia']

for i in range(0,6):
  #i=0
  Histro(Data_list[i],colour_list[i], Titl_list[i], xlab_list[i], ylab_list[i],␣
↪Graph[i])
```

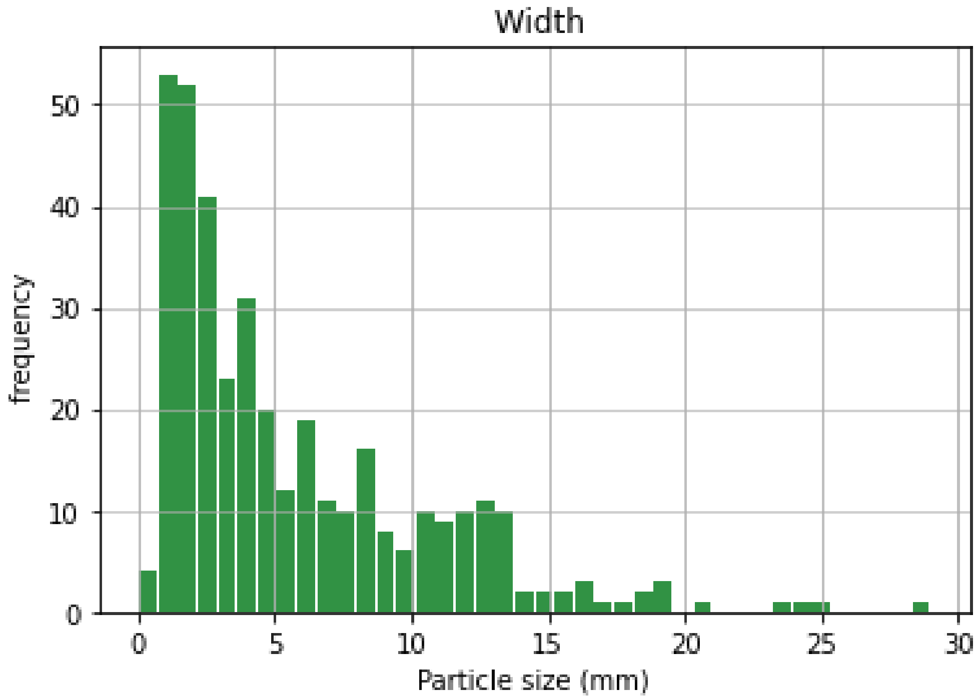


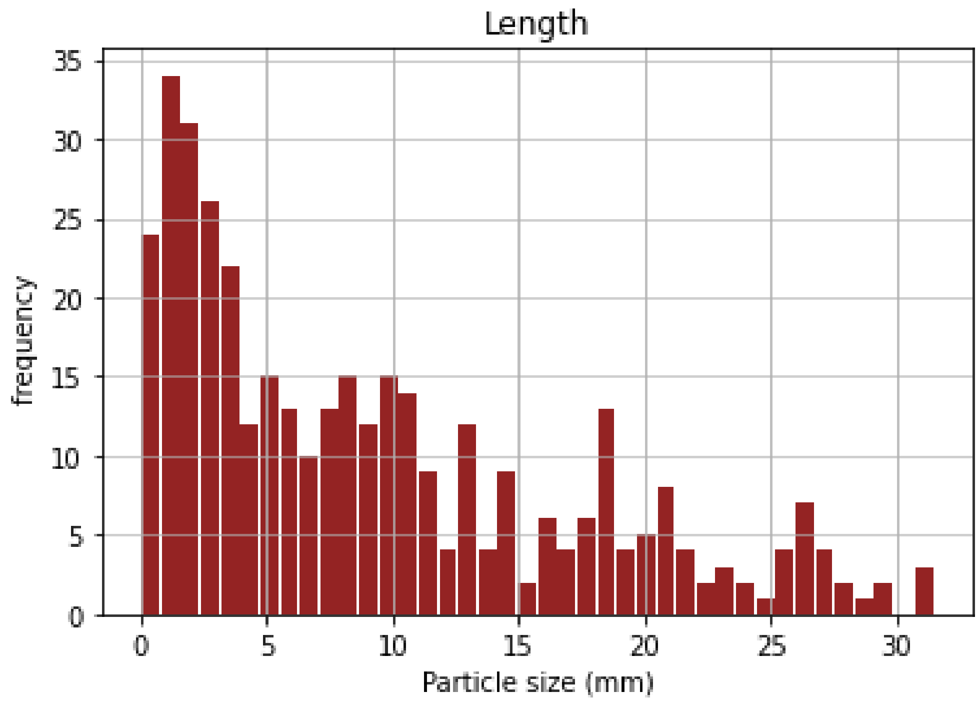


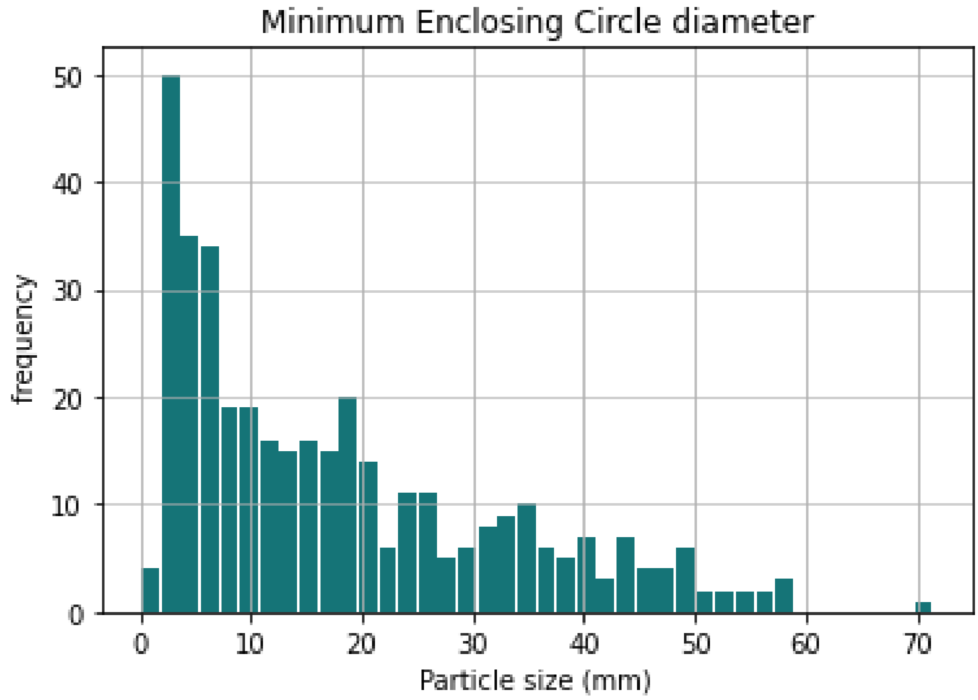
Equivalent Circular Perimeter Diameter
frequency
Particle size (mm)
0
10
20
30
40
50



Equivalent Circular Area Diameter
frequency
Particle size (mm)
0
5
10
15
20
25
30
35


Width
frequency
Particle size (mm)
0
10
20
30
40
50
0
5
10
15
20
25
30


Length
frequency
Particle size (mm)
0
5
10
15
20
25
30
35
0
5
10
15
20
25
30

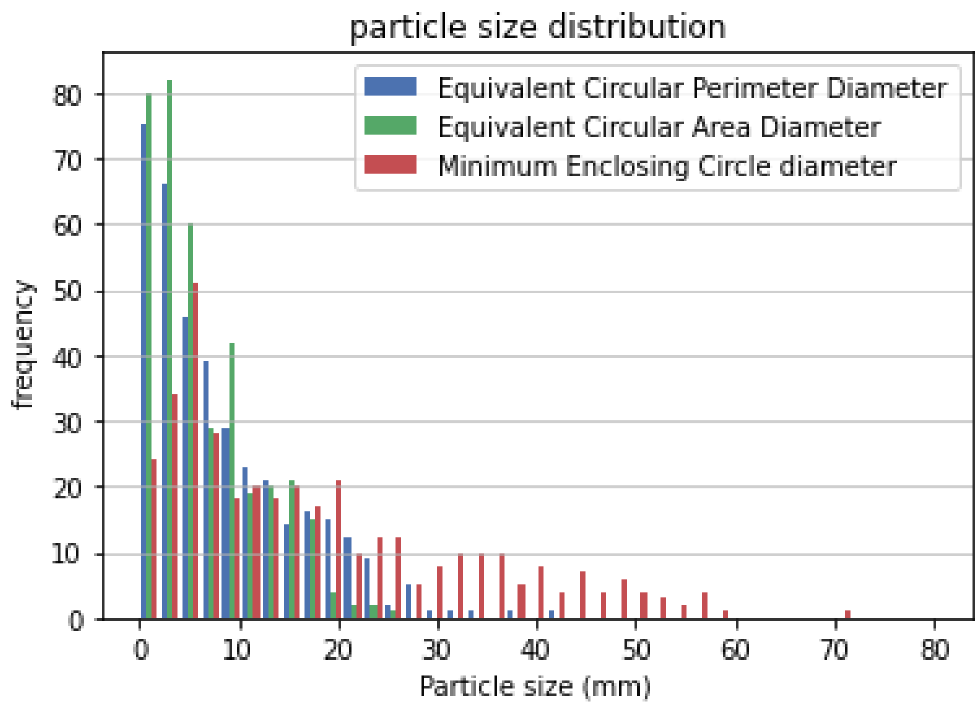


[63]:

```
```

[64]:

```
# Three datas in one hisrogram

plt.style.use('seaborn-deep')

x = EqPD
y = EqAD
z = MiEnD
bins = np.linspace(0, 80, 40)

plt.hist([x, y,z], bins, label=['Equivalent Circular Perimeter Diameter',␣
↪'Equivalent Circular Area Diameter','Minimum Enclosing Circle diameter'])
plt.title('particle size distribution')
plt.legend(loc='upper right')
plt.xlabel('Particle size (mm)')
plt.ylabel('frequency ')
plt.grid(axis='y', alpha=0.75)

plt.savefig('/content/drive/MyDrive/PlantPerform_A01/Final/PDS.jpg',dpi=300)
```



1.8 Saving code and images

[65]:
```
#Save images

image_Name =␣
↪['Blured_img','Detected_edges_img','Adjusted_exposure_img','Noise_reduced_PP_img','Detected
image_list =[blur_img,edg_img,result,inv_img,cont_img1,img_cc,Round_rec]

#for i in range(len(image_list)):
  #cv2_imshow(image_list[i])

  #fold = ('/content/drive/MyDrive/PlantPerform_A01/Final/fig_')
  #format = '.jpg'
  #dir=fold+image_Name[i]+format
  #cv2.imwrite(dir,image_list[i])
  #print(dir)
```

[ ]:
```
#Print the colab Notebook

!wget -nc https://raw.githubusercontent.com/brpy/colab-pdf/master/colab_pdf.py
from colab_pdf import colab_pdf
colab_pdf('GroupD-PP_As1.ipynb')
```

```
--2022-04-15 09:24:04--  https://raw.githubusercontent.com/brpy/colab-
pdf/master/colab_pdf.py
Resolving raw.githubusercontent.com (raw.githubusercontent.com)…
185.199.108.133, 185.199.109.133, 185.199.110.133, …
Connecting to raw.githubusercontent.com
(raw.githubusercontent.com)|185.199.108.133|:443… connected.
HTTP request sent, awaiting response… 200 OK
Length: 1864 (1.8K) [text/plain]
Saving to: 'colab_pdf.py'

colab_pdf.py        100%[===================>]   1.82K  --.-KB/s    in 0s

2022-04-15 09:24:04 (25.3 MB/s) - 'colab_pdf.py' saved [1864/1864]


WARNING: apt does not have a stable CLI interface. Use with caution in scripts.


WARNING: apt does not have a stable CLI interface. Use with caution in scripts.

Extracting templates from packages: 100%
```

1.6.2 Particle Size distribution -Table

Particle index	sphericity	Equivalent Circ	Equivalent Circ	Length and Width (px)	Length (mm)	Width (mm)	Minimum Enclosing Circle diameter (mm)	
0	0.608575563	3.727672563	2.908004082	7.071067810058594,4.949747085	3.885202093	2.719641256	7.615972996	
1	0.742845185	4.717031135	4.065536824	5.656853675842285,9.899494171	5.439282512	3.10816136	10.29583073	
2	0.497251577	5.663952925	3.993996865	5.149810314178467,12.99156761	7.138223962	2.829566107	13.00020027	Unit relationship
3	0.871464655	1.936280383	1.807561715	2.8284268379211426,3.53553390	1.942601047	1.55408068	4.123305321	1mm = 1.83px
4	0.716718881	4.077463626	3.451950173	8.485280990600586,4.949747085	4.662242303	2.719641256	9.48703289	
5	0.871464655	1.936280383	1.807561715	2.8284268379211426,3.53553390	1.942601047	1.55408068	4.123305321	
6	0.58299472	10.39856198	7.939727466	9.717975616455078,23.41148567	12.86345367	5.339547042	23.53740501	
7	0.857371637	2.678299291	2.479954213	3.535533905029297,4.949747085	2.719641256	1.942601047	5.440027237	
8	0.770504029	2.780750753	2.440897375	5.656853675842285,3.535533905	3.10816136	1.942601047	5.831151485	
9	0.838416234	3.028090354	2.772673101	3.5355336666107178,6.36396121	3.496681989	1.942600916	6.708403587	
10	0.924798548	2.883202193	2.772673101	4.949747085571289,4.949747085	2.719641256	2.719641256	5.590369701	
11	0.886759584	2.183620026	2.056269406	3.535533905029297,3.535533905	1.942601047	1.942601047	4.333512783	
12	0.795154212	13.13687591	11.71433877	20.499929428100586,23.2115631	12.7536061	11.26369749	27.45926094	
13	0.806145091	2.183620005	1.960575952	4.242640495300293,2.828426837	2.331121151	1.55408068	4.472335815	
14	0.751004039	15.26048138	13.22480782	29.453102111816406,21.9903240	16.18302314	12.08259562	32.2026825	
15	0.647128273	7.105554042	5.716012031	12.727922439575195,10.6066007	6.993363978	5.827802616	14.56042004	
16	0.800106387	1.833828942	1.640335528	2.1213202476501465,3.53553390	1.942601047	1.165560576	4.123305321	
17	0.795146376	17.21433962	15.35019541	34,23	18.68131868	12.63736264	35.44028854	
18	0.459725861	23.33053502	15.81882296	24.31839942932129,44.32224655	24.35288272	13.36175793	46.17378235	
19	0.584865567	20.07996391	15.35645441	17.39252281188965,47.43416213	26.06272645	9.556331215	47.43436432	
20	0.655474341	11.55038669	9.351351289	17,20	10.98901099	9.340659341	22.92165375	
21	0.800106387	1.833828942	1.640335528	3.535533905029297,2.121320247	1.942601047	1.165560576	4.123305321	
22	0.839954455	1.586489299	1.454002041	2.1213202476501465,2.82842683	1.55408068	1.165560576	3.162477732	
23	0.689539281	6.465986554	5.369258373	7.139510154724121,14.86422634	8.16715733	3.922807777	14.86626911	
24	0.574146264	17.69144096	13.40523646	20.153465270996094,36.1120910	19.84180828	11.07333257	36.24251556	
25	0.779838144	2.328508228	2.056269406	3.535533905029297,3.535533905	1.942601047	1.942601047	5.000199795	
26	0.73869009	6.013744009	5.168637493	8.497057914733887,12.07476615	6.634486901	4.66871314	12.44561481	
27	0.586853408	10.97811479	8.409933909	12.20000171661377,24.2000076	13.29670372	6.703297646	25.23905945	
28	0.49215395	7.805136189	5.475590643	7.222716808319092,16.72629165	9.190270141	3.968525719	17.08820724	
29	0.769512338	8.257378713	7.243525561	14,14	7.692307692	7.692307692	16.12471581	
30	0.790671711	4.572142933	4.065536824	6.363960266113281,9.192387580	5.050762407	3.496681465	9.849058151	
31	0.912863342	2.63586255	2.518405409	4,5	2.747252747	2.197802198	5.385364532	
32	0.78539819	0.989358572	0.876796221	1.4142134189605713,1.41421341	0.77704034	0.77704034	2.000200033	
33	0.806145076	2.183620026	1.960575952	4.242640495300293,2.828426837	2.331121151	1.55408068	4.472335815	
34	0.74213362	5.868855827	5.055853649	9.370742797851562,10.19273757	5.600405263	5.148759779	12.53016472	
35	0.574416429	11.88259986	9.005853937	26.475048065185547,11.0908975	14.54672971	6.093899758	26.47660446	
36	0.544473449	8.752058	6.458007431	8.548747062683105,18.74148559	10.29751956	4.697113771	18.97386551	
37	0.481160019	22.22842831	15.41890467	16.263456344604492,50.9116897	27.97345591	8.935965025	51.614151	
38	0.857371624	1.339149656	1.239977107	2.1213202476501465,2.12132024	1.165560576	1.165560576	3.000200033	
39	0.564516074	19.19305676	14.42056591	18.973663330078125,42.0582847	23.10894767	10.42508974	42.58103943	
40	0.748627916	5.066822198	4.383981103	10.733126640319824,6.26099014	5.89732233	3.440104474	10.77052975	
41	0.828291062	9.988756215	9.090816704	17.677669525146484,16.2634544	9.713005234	8.935963977	20.09995079	
42	0.51108489	15.15802985	10.83650607	31.382511138916016,15.0413227	17.24313799	8.264463026	31.38491058	
43	0.820213966	2.780750753	2.518405409	4.242640495300293,4.949747085	2.719641256	2.331121151	5.427273273	
44	0.898353492	2.53341111	2.401205342	4.949747085571289,3.535533905	2.719641256	1.942601047	5.385364532	
45	0.78539819	0.989358572	0.876796221	1.4142134189605713,1.41421341	0.77704034	0.77704034	2.000200033	
46	0.803785923	2.67829927	2.401205342	2.8284268379211426,5.65685367	3.10816136	1.55408068	5.831151485	
47	0.602765347	17.4865381	13.57619218	18.755231857299805,38.6246376	21.22232835	10.30507245	39.05144501	
48	0.494851845	7.907587651	5.562648417	18.9649600982666,6.417436122	10.42030775	3.526063804	18.97386551	
49	0.808518143	1.68894074	1.518655602	2.8284268379211426,2.82842683	1.55408068	1.55408068	4.000199795	
50	0.655078763	8.547155139	6.917805496	17.669090270996094,11.2672462	9.708291358	6.190794641	18.02795601	
51	0.81075781	11.34548375	10.21571076	23.28386878967285,14.72655105	12.7933345	8.091511569	23.3674984	
52	0.800106387	1.833828942	1.640335528	2.8284268379211426,2.82842683	1.55408068	1.55408068	4.000199795	
53	0.720794092	5.561501505	4.721692151	8,10	5.494505495	4.395604396	10.77052975	
54	0.375998062	18.69837741	11.46559836	33.801734924316406,19.2055320	18.57238183	10.55249015	35.24117661	
55	0.612037604	7.805136189	6.106179101	10.762669563293457,15.4332618	8.479814215	5.913554705	16.40570641	
56	0.555645534	12.70949223	9.473863297	19.9737548828125,18.18123817	10.97459059	9.989691305	22.39019775	
57	0.394099081	28.10758092	17.64518357	29.209136962890625,45.8802299	25.20891755	16.04897635	45.88047791	
58	0.839954455	1.586489299	1.454002041	2.1213202476501465,2.82842683	1.55408068	1.165560576	3.162477732	
59	0.78539819	0.989358572	0.876796221	1.4142134189605713,1.41421341	0.77704034	0.77704034	2.000200033	
60	0.652200967	22.63095288	18.27652225	26.325002670288086,47.3092498	25.99409334	14.46428718	48.2702713	
61	0.752691671	12.582182	10.91602492	18.010173797607422,22.7863502	12.51997266	9.895699889	24.37812996	
62	0.712147869	15.55025764	13.12268422	31.15436553955078,19.76209831	17.11778326	10.85829578	31.48223877	
63	0.414671493	20.91715335	13.46959912	52.03324508666992,14.82159137	28.5896951	8.143731526	52.07712555	
64	0.886759584	2.183620026	2.056269406	4.242640495300293,2.828426837	2.331121151	1.55408068	4.472335815	
65	0.78539819	0.989358572	0.876796221	1.4142134189605713,1.41421341	0.77704034	0.77704034	2.000200033	

66	0.745750625	4.921934017	4.250427378	10.39893913269043,6.313641548	5.71370282	3.469033818	10.44050694		
67	0.646507175	6.918229057	5.562648417	15.380306243896484,7.19401454	8.450717716	3.952755247	15.52437496		
68	0.78539819	0.989358572	0.876796221	1.4142134189605713,1.41421341	0.77704034	0.77704034	2.000200033		
69	0.837131828	21.04446358	19.2546127	35,36	19.78021978	19.23076923	40.79244614		
70	0.73797652	7.412908323	6.368100389	16,9	8.791208791	4.945054945	16.49262238		
71	0.700552215	12.78708488	10.70266183	21.13130760192871,21.33449172	11.7222482	11.61060857	25.13990211		
72	0.729883595	6.960665819	5.946721296	12.727922439575195,9.89949417	6.993363978	5.439282512	14.56042004		
73	0.650386648	18.14368347	14.63225526	22.978309631347656,37.8958930	20.82191928	12.62544485	39.39816284		
74	0.856529133	6.176210149	5.716012031	8,12	6.593406593	4.395604396	13.00020027		
75	0.60408411	18.80082889	14.61253966	33,24	18.13186813	13.18681319	35.45474243		
76	0.340594847	42.17380072	24.6128428	53.66563034057617,52.77120208	29.48661008	28.99516598	71.47047424		
77	0.719727286	9.681401872	8.213385965	15.55634880065918,15.55634880	8.547444396	8.547444396	19.37322617		
78	0.770089141	7.787558293	6.833950209	12.675652503967285,11.1392097	6.964644233	6.120444916	15.15907764		
79	0.735007123	10.6283237	9.111933612	16,19.200000762939453	10.54945097	8.791208791	22.20380402		
80	0.608177594	5.621516184	4.383981103	5.939697742462158,13.01076602	7.148772544	3.263570188	13.03860474		
81	0.78539819	0.989358572	0.876796221	1.4142134189605713,1.41421341	0.77704034	0.77704034	2.000200033		
82	0.707453375	3.377881479	2.841144476	6.933752536773682,4.43760156	3.809754141	2.438242619	7.280309677		
83	0.84327026	3.130541815	2.874768658	4.949747085571289,5.65685367	3.10816136	2.719641256	6.411989212		
84	0.871464655	1.936280383	1.807561715	2.8284268379211426,3.53553390	1.942601047	1.55408068	4.123305321		
85	0.682081933	15.96006352	13.18113745	33.50361633300781,21.3125095	18.4085804	11.71017008	34.17621231		
86	0.687331421	5.126836898	4.250427378	10,7	5.494505495	3.846153846	10.64726067		
87	0.804048388	9.391625446	8.421352698	16.26345443725586,15.55634880	8.935963977	8.547444396	19.23558426		
88	0.6671957	17.31679114	14.14470969	26,32	17.58241758	14.28571429	37.45349121		
89	0.601119056	5.766404387	4.470801039	13.576773643493652,5.50093412	7.459765738	3.022491277	13.60167122		
90	0.943271167	1.68894074	1.640335528	2.8284268379211426,2.82842683	1.55408068	1.55408068	3.162477732		
91	0.871464655	1.936280383	1.807561715	3.535533905029297,2.82842683	1.942601047	1.55408068	4.123305321		
92	0.707379011	10.9956927	9.248018269	14,23	12.63736264	7.692307692	24.04183006		
93	0.745827712	10.2536738	8.855207761	18,15	9.89010989	8.241758242	19.50544167		
94	0.839954455	1.586489299	1.454002041	2.1213202476501465,2.82842683	1.55408068	1.165560576	3.162477732		
95	0.7545746	3.028090375	2.630388662	5.656853675842285,4.24264095	3.10816136	2.331121151	7.071267605		
96	0.78539819	0.989358572	0.876796221	1.4142134189605713,1.41421341	0.77704034	0.77704034	2.000200033		
97	0.623508708	8.547155097	6.749053122	18.80000114440918,9.199999809	10.32967096	5.05494495	19.41668892		
98	0.778381948	9.349188726	8.248411224	15,16	8.791208791	8.241758242	18.13941956		
99	0.618709048	7.702684748	6.058782427	8.928435325622559,17.58215141	9.660522754	4.905733695	18.02795601		
100	0.644573078	4.264788611	3.423998699	9.192387580871582,4.24264095	5.050762407	2.331121151	9.849058151		
101	0.765837829	4.367240031	3.82186612	6.363961219787598,7.77817440	4.273722198	3.496681989	9.257768631		
102	0.782646482	5.31416182	4.701295977	7.77817440032959,9.89949417	5.439282512	4.273722198	11.40195465		
103	0.703035628	20.18241539	16.92239421	47,23	25.82417582	12.63736264	47.04273224		
104	0.60450415	8.112490532	6.307450432	19.39908218383789,7.56235361	10.65883636	4.155139347	19.41668892		
105	0.642598885	5.603938225	4.492243852	11.31370735168457,6.36396026	6.216322721	3.496681465	12.17727947		
106	0.638629061	12.43729382	9.939172612	25,14	13.73626374	7.692307692	26.24901009		
107	0.78539819	0.989358572	0.876796221	1.4142134189605713,1.41421341	0.77704034	0.77704034	2.000200033		
108	0.707414329	7.352893665	6.184366396	14.849242210388184,9.19238758	8.158924291	5.050762407	15.26453781		
109	0.78539819	0.989358572	0.876796221	1.4142134189605713,1.41421341	0.77704034	0.77704034	2.000200033		
110	0.871464655	1.936280383	1.807561715	2.8284268379211426,3.53553390	1.942601047	1.55408068	4.123305321		
111	0.78539819	0.989358572	0.876796221	1.4142134189605713,1.41421341	0.77704034	0.77704034	2.000200033		
112	0.768939401	1.936280383	1.69790858	3.535533905029297,2.82842683	1.942601047	1.55408068	4.123305321		
113	0.45755607	8.299815475	5.614235132	6.685031890869141,20.05509757	11.01928438	3.673094446	20.39627838		
114	0.343496605	14.27112272	8.364102865	11.35590934753418,31.78453063	17.46402782	6.239510631	33.24173737		
115	0.78539819	0.989358572	0.876796221	1.4142134189605713,1.41421341	0.77704034	0.77704034	2.000200033		
116	0.840858864	2.985653634	2.737795322	6,4	3.296703297	2.197802198	6.324755192		
117	0.627001734	3.275430018	2.593598197	7.602630138397217,3.13049507	4.177269307	1.720052237	7.615972996		
118	0.769074021	6.303520434	5.527989834	11.147406578063965,9.26092338	6.124948669	5.088419443	12.24347973		
119	0.778573022	3.582784361	3.161333732	8,4	4.395604396	2.197802198	8.062458038		
120	0.691563138	2.63586255	2.191990552	3,6	3.296703297	1.648351648	6.082962513		
121	0.739838837	2.038731823	1.753592441	5,2	2.747252747	1.098901099	5.000199795		
122	0.857371624	1.339149656	1.239977107	2.1213202476501465,2.12132024	1.165560576	1.165560576	3.000200033		
123	0.839954455	1.586489299	1.454002041	2.1213202476501465,2.82842683	1.55408068	1.165560576	3.162477732		
124	0.857371624	1.339149656	1.239977107	2.1213202476501465,2.12132024	1.165560576	1.165560576	3.000200033		
125	0	0.349791084	0	1,0	0.549450549	0	1.000200033		
126	0.805326872	3.522769682	3.161333732	7.071067810058594,4.24264095	3.885202093	2.331121151	7.615972996		
127	0.943271167	1.68894074	1.640335528	2.8284268379211426,2.82842683	1.55408068	1.55408068	3.162477732		
128	0.694271893	7.642670028	6.368100389	12.727920532226562,11.3137073	6.99336293	6.216322721	16.12471581		
129	0.647160889	3.130541836	2.518405409	7.155416965484619,3.13049507	3.931547783	1.720052237	7.280309677		
130	0.347573724	23.01589986	13.56911203	11.094342231750488,57.2279586	31.44393334	6.095792435	57.27148438		
131	0.808518153	3.377881459	3.037311204	4.242640495300293,7.07106781	3.885202093	2.331121151	7.615972996		
132	0.625404413	16.10495169	12.73620078	18.967212677001953,31.8273353	17.4875469	10.42154543	32.39668655		

133	0.857371624	1.339149656	1.239977107	2.1213202476501465,2.121320247	1.165560576	1.165560576	3.000200033		
134	0.673765115	4.777045814	3.921151904	11,5	6.043956044	2.747252747	11.04556084		
135	0.808518143	1.68894074	1.518655602	2.8284268379211426,2.828426837	1.55408068	1.55408068	4.000199795		
136	0.370500245	8.052475832	4.90143988	5.079641342163086,19.64127922	10.79191166	2.791011726	19.69791603		
137	0.474490531	19.58528465	13.49098511	23.536956787109375,38.6136207	21.21627514	12.93239384	41.76633453		
138	0.749354407	6.405971938	5.545346202	14.142135620117188,7.07106733	7.770404187	3.885201831	14.42240524		
139	0.681868127	10.031193	8.283288382	16.440589904785156,15.1636514	9.033291156	8.33167663	20.8808136		
140	0.521287208	4.819482576	3.479677126	11,4	6.043956044	2.197802198	11.04556084		
141	0.845834851	3.232993277	2.973360648	7,4	3.846153846	2.197802198	7.071267605		
142	0.775075873	3.625221102	3.191586418	4.949747085571289,7.07106781	3.885202093	2.719641256	7.45376873		
143	0.740740961	7.907587651	6.805768851	11,16	8.791208791	6.043956044	17.46444893		
144	0.767992292	13.5715406	11.89344259	23.255107879638672,21.0190372	12.7775318	11.54892156	27.3132		
145	0.478272679	26.21373728	18.12870278	47.769630432128906,30.2704734	26.24704969	16.63212829	47.811306		
146	0.583470719	11.59282345	8.855207761	10.795912742614746,25.8271446	14.1907388	5.931820188	26.47660446		
147	0.882610291	3.232993256	3.037311204	5,6	3.296703297	2.747252747	6.708403587		
148	0.606070023	4.469691471	3.479677126	10.435516357421875,4.42718839	5.733800196	2.432521097	10.44050694		
149	0.601841533	5.953729309	4.618810696	5.800000190734863,13.2000007	7.252747672	3.186813292	13.41660786		
150	0.791983534	15.3029181	13.6185958	24.70321273803711,26.83533287	14.74468839	13.57319381	29.4741478		
151	0.871464655	1.936280383	1.807561715	2.8284268379211426,3.53553390	1.942601047	1.55408068	4.123305321		
152	0.839954455	1.586489299	1.454002041	2.1213202476501465,2.82842683	1.55408068	1.165560576	3.162477732		
153	0.78539819	0.989358572	0.876796221	1.4142134189605713,1.41421341	0.77704034	0.77704034	2.000200033		
154	0.74556233	8.939382943	7.718795241	17.541053771972656,10.7550983	9.637941633	5.909394694	18.38497734		
155	0.720794092	2.780750753	2.360846075	5.656853675842285,3.53553390	3.10816136	1.942601047	6.324755192		
156	0.831276434	8.95696086	8.166451946	12,18	9.89010989	6.593406593	18.68174171		
157	0.544051954	15.63513133	11.53245373	12.089754104614258,36.3855133	19.99204028	6.642722036	36.40074921		
158	0.802461902	6.013743988	5.38712617	8.485280990600586,12.0208148	6.604843349	4.662242303	12.55181789		
159	0.766624288	6.773340855	5.930539712	10,12	6.593406593	5.494505495	13.89264393		
160	0.376406138	15.50782098	9.514350077	36,12	19.78021978	6.593406593	36.87837601		
161	0.703675358	16.61720891	13.93940548	34,20	18.68131868	10.98901099	34.6885376		
162	0.70901222	4.32480329	3.641608205	5,10	5.494505495	2.747252747	10.19823933		
163	0.685702635	27.72263388	22.95632681	47.86824035644531,43.931167602	26.30123097	24.13800418	56.23849487		
164	0.779838158	2.328508207	2.056269406	4.242640495300293,2.82842683	2.331121151	1.55408068	5.099219322		
165	0.353813171	22.8358558	13.58326864	33.524662017822266,24.8641204	18.42014397	13.66160466	39.92512894		
166	0.615757252	3.830124004	3.005506022	9.21745777130127,3.72620677947	5.064537237	2.047366362	9.219744682		
167	0.857371624	1.339149656	1.239977107	2.1213202476501465,2.121320247	1.165560576	1.165560576	3.000200033		
168	0.616675181	10.04877087	7.891165986	23,10	12.63736264	5.494505495	23.08699226		
169	0.717676774	3.872560766	3.280671056	5.656853675842285,7.07106781	3.885202093	3.10816136	8.246411324		
170	0.702798285	8.981819746	7.529734818	19.52172088623047,9.370426177	10.72622027	5.148585812	19.74365044		
171	0.497930993	13.83645809	9.763589502	11.741438865661621,33.9307670	18.6432786	6.451340036	34.01490021		
172	0.583663985	7.762699427	5.930539712	18,7	9.89010989	3.846153846	18.24848747		
173	0.839954455	1.586489299	1.454002041	2.8284268379211426,2.121320247	1.55408068	1.165560576	3.162477732		
174	0.886759584	2.183620026	2.056269406	3.535533905029297,3.53553390	1.942601047	1.942601047	4.333543301		
175	0.75868694	17.71629987	15.43136439	22.62741470336914,33.23401641	18.26044858	12.43264544	34.74752808		
176	0.374743447	18.71595541	11.45721401	44.635406494140625,10.7196674	24.52494862	5.889927162	44.64322662		
177	0.7100498	9.843868012	8.294881514	19.801239013671875,12.8359804	10.87980166	7.052736492	20.10691261		
178	0.759494067	9.741416551	8.489542902	15,18	9.89010989	8.241758242	18.71054268		
179	0.813058601	2.430959669	2.191990552	3.535533905029297,4.24264049	2.331121151	1.942601047	5.099219322		
180	0.813058601	2.430959669	2.191990552	4.242640495300293,3.53553390	2.331121151	1.942601047	5.099219322		
181	0.886759584	2.183620026	2.056269406	3.535533905029297,3.53553390	1.942601047	1.942601047	4.333512783		
182	0.673743319	4.717031094	3.871827275	9.899494171142578,5.65685367	5.439282512	3.10816136	10.77052975		
183	0.750200183	2.678299291	2.31978475	2.8284268379211426,5.65685367	3.10816136	1.55408068	5.831151485		
184	0.54817906	12.72707023	9.423010201	26,14	14.28571429	7.692307692	26.17637825		
185	0.730620566	6.405971854	5.475590643	8.049844741821289,12.9691934	7.125930472	4.422991616	13.34186459		
186	0.395737704	13.67399206	8.601992078	33.5201416015625,8.53814888	18.41766022	4.691290593	33.61566925		
187	0.857371624	1.339149656	1.239977107	2.1213202476501465,2.121320247	1.165560576	1.165560576	3.000200033		
188	0.519681654	11.49037199	8.283288382	15,16	8.791208791	8.241758242	19.22272491		
189	0.550754094	12.64219677	9.38212927	12,30	16.48351648	6.593406593	30.41401291		
190	0.187441512	37.80656071	16.36816749	47.06787872314453,34.7125549	25.86147183	19.07283238	49.40356064		
191	0.570075962	19.44039636	14.67815535	38.400001525878906,22	21.09890194	12.08791209	38.58776093		
192	0.605646842	10.6883384	8.318019303	13.096923828125,21.8282051086	11.99351929	7.196111993	22.22940445		
193	0.606070018	8.939382985	6.959354251	10.60660171508789,18.3847706	10.10152586	5.82780314	18.91509819		
194	0.691742759	22.46848688	18.68728191	34,38	20.87912088	18.68131868	44.65442657		
195	0.574869259	11.44793525	8.679839747	10.59027099609375,26.4756793	14.54707659	5.818830218	26.47660446		
196	0.78539819	0.989358572	0.876796221	1.4142134189605713,1.41421341	0.77704034	0.77704034	2.000200033		
197	0.650548421	6.960665819	5.614235132	15.808955192565918,7.48845243	8.686239117	4.114534305	15.81158829		
198	0.886759584	2.183620026	2.056269406	2.8284268379211426,4.24264049	2.331121151	1.55408068	4.472335815		
199	0.785893936	2.925638934	2.593598197	4.949747085571289,4.24264049	2.719641256	2.331121151	6.082962513		

200	0.535539552	4.717031135	3.451950173	5.219700813293457,10.43940162€	5.73593496	2.86796748	10.44050694		
201	0.800106387	1.833828942	1.640335528	2.8284268379211426,2.82842683	1.55408068	1.55408068	4.000199795		
202	0.731978148	9.988756215	8.545952445	19.091880798339844,14.1421356	10.49004439	7.770404187	19.82251549		
203	0.721405146	16.55719421	14.06294768	21.0190372467041,33.541019439	18.42913156	11.54892156	34.20545959		
204	0.857371624	1.339149656	1.239977107	2.1213202476501465,2.12132024	1.165560576	1.165560576	3.000200033		
205	0.839954455	1.586489299	1.454002041	2.1213202476501465,2.82842683	1.55408068	1.165560576	3.162477732		
206	0.495662932	14.04864194	9.890712194	32.01261901855469,11.31404685	17.58935111	6.216509264	32.09222412		
207	0.698659066	3.275430039	2.737795322	7.155416965484619,3.57770848	3.931547783	1.965773892	7.280309677		
208	0.443621631	8.607169818	5.73279918	20.571826934814453,7.60263204	11.30320161	4.177270355	20.8808136		
209	0.666230897	2.57584785	2.102483477	2.1213202476501465,5.65685367	3.10816136	1.165560576	5.831151485		
210	0.725530568	2.183620026	1.85996566	4.242640495300293,2.82842683	2.331121151	1.55408068	5.099219322		
211	0.776177417	4.222351849	3.71993132	8.485280990600586,5.65685367	4.662242303	3.10816136	9.48703289		
212	0.839954455	1.586489299	1.454002041	2.8284268379211426,2.12132024	1.55408068	1.165560576	3.162477732		
213	0.559236795	5.561501505	4.159009652	9.899494171142578,7.07106781	5.439282512	3.885202093	11.40195465		
214	0.716666996	6.363535113	5.38712617	7.602630138397217,13.86362075	7.617374043	4.177269307	14.31802177		
215	0.301473657	19.68773607	10.8098698	17.854778289794922,28.733737	15.7877681	9.810317742	29.73233795		
216	0.640130182	7.310456924	5.848960314	9.65981388092041,15.29470729	8.403685329	5.307590044	15.45258045		
217	0.704015808	5.911292547	4.959908427	7.071067810058594,12.02081489	6.604843349	3.885202093	13.60167122		
218	0.542488205	15.30291808	11.27118046	34.882659912109375,14.7580471	19.16629666	8.10881709	35.90284348		
219	0.742181939	16.76209709	14.44054362	23.536958694458008,32.2226394	17.75969201	12.93239489	33.75196838		
220	0.73474528	8.709621259	7.465650803	16.473756790161133,12.3553161	9.05151472	6.788635254	18.43928909		
221	0.790630876	3.975012206	3.534478562	4.949747085571289,8.48528099	4.662242303	2.719641256	8.944472313		
222	0.659089854	10.0736297	8.178210702	21.21320343017578,10.6066017	11.65560628	5.82780314	21.40113449		
223	0.616013827	5.911292568	4.639569501	5.696134567260742,13.83346843	7.600806834	3.129744268	13.89264393		
224	0.471951334	13.71642876	9.423010201	14.631509780883789,24.3310508	13.36870927	8.039291088	24.3483429		
225	0.778573026	2.53341111	2.235400519	4.949747085571289,3.53553390	2.719641256	1.942601047	5.385364532		
226	0.618217398	6.405971875	5.036810818	6.773190498352051,13.54638099	7.443066482	3.721533241	13.60167122		
227	0.469189375	22.67338968	15.53068232	38.07866287231445,28.2799186	20.92234224	15.53841685	39.61521149		
228	n/A	0	0	0,0	0	0	0.0002		
229	0.839954455	1.586489299	1.454002041	2.8284268379211426,2.12132024	1.55408068	1.165560576	3.162477732		
230	0.433748718	19.64529926	12.93831591	40.303367614746094,18.2585182	22.14470748	10.03215287	40.31148529		
231	0.81293312	11.0557074	9.96813578	18,18	9.89010989	9.89010989	21.25020027		
232	0.612524666	18.53591127	14.50693727	20.17112922668457,36.94898986	20.30164278	11.08303804	37.8475914		
233	0.693598916	5.766404366	4.802410684	6.667948246002197,13.33589744	7.327416179	3.663707827	13.34186459		
234	0.510665656	5.169273639	3.694008138	4.338609218597412,12.36503696	6.793976354	2.383851219	12.36951733		
235	0.579171744	24.13256869	18.36568934	54,25	29.67032967	13.73626374	54.58957291		
236	0.746911632	16.84697061	14.55983421	20.40833282470703,34.12821960	18.75176901	11.21336968	35.33695984		
237	0.691531598	17.66658214	14.69124332	34,23	18.68131868	12.63736264	36.40074921		
238	0.480529392	11.79772631	8.178210702	8.27971076965332,27.41192245	15.06149585	4.549291632	27.85697746		
239	0.687649142	21.80406047	18.08093273	23.62982749938965,47.53126525	26.11607981	12.9834217	47.85414124		
240	0.857371624	1.339149656	1.239977107	2.1213202476501465,2.12132024	1.165560576	1.165560576	3.000200033		
241	0.655595237	14.10865669	11.42361507	22.62741470336914,21.92031097	12.43264544	12.04412691	28.28447151		
242	0.842206409	3.377881459	3.099942767	4.949747085571289,6.36396121	3.496681989	2.719641256	7.280309677		
243	0.827134513	5.169273639	4.701295977	9.391485214233398,8.04984474	5.160156711	4.422991616	10.29583073		
244	0.804048377	3.130541836	2.807117566	6.363960266113281,4.24264049	3.496681465	2.331121151	6.708403587		
245	0.746666918	19.03059064	16.44431253	32.31098937988281,29.71125411	17.75329087	16.3248649	33.84972		
246	0.6384669	5.459050065	4.362006122	6.363960266113281,11.31370735	6.216322721	3.496681465	12.08324623		
247	0.390555422	15.85761202	9.910124798	18.68000030517578,30.7199993	16.8791205	10.26373643	32.47887039		
248	0.746547801	17.44410136	15.07222649	24.041629791259766,34.6482276	19.03748774	13.2096867	38.89749908		
249	0.4633704	11.43035731	7.780794592	26,11	14.28571429	6.043956044	26.1727047		
250	0.924798547	2.038731823	1.960575952	3,4	2.197802198	1.648351648	4.123305321		
251	0.660486692	11.43035735	9.289489411	16.970561981201172,21.2132034	11.65560628	9.324484605	24.11284637		
252	0.750200183	2.678299291	2.31978475	3.5355336666107178,4.94974708	2.719641256	1.942600916	6.082962513		
253	0.688461141	4.32480329	3.588443188	5,10	5.494505495	2.747252747	10.05007553		
254	0.78539819	0.989358572	0.876796221	1.4142134189605713,1.41421341	0.77704034	0.77704034	2.000200033		
255	0.651169252	2.925638955	2.360846075	4.949747085571289,4.24264049	2.719641256	2.331121151	6.082962513		
256	0.765837813	3.088105075	2.70246745	3,7	3.846153846	1.648351648	7.071267605		
257	0.68143043	3.522769682	2.908004082	6.363960266113281,4.94974708	3.496681465	2.719641256	7.097939491		
258	0.703392229	26.75085331	22.43553461	45.853023529052734,43.3232040	25.19396897	23.80395826	54.81311798		
259	0.691377708	20.80440472	17.29868059	35.69054412841797,30.40760612	19.61018908	16.70747589	45.54138947		
260	0.857371624	1.339149656	1.239977107	2.1213202476501465,2.12132024	1.165560576	1.165560576	3.000200033		
261	0.623763407	21.81134137	17.2263125	27.557655334472656,37.7312698	20.73146694	15.14156887	44.38488007		
262	0.664756171	7.352893706	5.995003994	10.30746841430664,14.52416133	7.980308428	5.663444184	15.55337524		
263	0.700455208	12.17237628	10.18745148	14.849241256713867,26.1629505	14.37524754	8.158923767	26.41060066		
264	0.878906801	3.770109283	3.534478562	6.363961219787598,5.65685367	3.496681989	3.10816136	7.325118542		
265	0.695444794	19.45797436	16.2266533	39,25	21.42857143	13.73626374	40.15311432		
266	0.800106387	1.833828942	1.640335528	2.1213202476501465,3.53553390	1.942601047	1.165560576	4.123305321		

267	0.943271167	1.68894074	1.640335528	2.8284268379211426,2.828426837	1.55408068	1.55408068	3.162477732		
268	0.558003302	6.01374403	4.492243852	14.55213737487793,5.335783958	7.995679876	2.931749428	14.56042004		
269	0.703073957	8.299815496	6.959354251	15.263373374938965,12.0049018	8.386468887	6.596099937	16.85133171		
270	0.806145076	2.183620026	1.960575952	4.242640495300293,2.828426837	2.331121151	1.55408068	5.099219322		
271	0.800106387	1.833828942	1.640335528	3.535533905029297,2.121320247	1.942601047	1.165560576	4.123305321		
272	0.711495966	4.222351849	3.561563085	5.656853675842285,8.485280990	4.662242303	3.10816136	8.713406563		
273	0.857371624	1.339149656	1.239977107	2.1213202476501465,2.121320247	1.165560576	1.165560576	3.000200033		
274	0.447340632	11.7801484	7.87897885	15.55634880065918,17.67766952	9.713005234	8.547444396	19.80293465		
275	0.737747846	6.713326197	5.766226864	14.230247497558594,8.221920967	7.818817306	4.517538993	14.42944622		
276	0.668026135	7.412908344	6.058782427	10,15	8.241758242	5.494505495	15.84669113		
277	0.551410369	21.12205621	15.68460761	18.7011661529541,48.636245727	26.72321194	10.27536602	48.79030228		
278	0.705861369	14.63849176	12.29861028	23,23	12.63736264	12.63736264	29.20636368		
279	0.55960304	14.82581675	11.0906924	35.49702072143555,13.924245834	19.50385754	7.650684524	35.7353363		
280	0.514219581	13.32420093	9.5546653	30,13	16.48351648	7.142857143	30.59431648		
281	0.792730635	7.352893644	6.546679876	9.192388534545898,15.55634880	8.547444396	5.050762931	16.12471581		
282	0.717649564	3.975012206	3.367399781	7.602630138397217,5.813776969	4.177269307	3.19438295	8.239199638		
283	0.839954455	1.586489299	1.454002041	2.8284268379211426,2.121320247	1.55408068	1.165560576	3.162477732		
284	0.871464655	1.936280383	1.807561715	3.535533905029297,2.828426837	1.942601047	1.55408068	4.123305321		
285	0.857538291	2.63586255	2.440897375	4,5	2.747252747	2.197802198	5.385364532		
286	0.54965167	8.257378713	6.121896447	19.563480377197266,7.56235361	10.74916504	4.155139347	19.64708328		
287	0.655022462	5.808841107	4.701295977	7.77817440032959,11.313707351	6.216322721	4.273722198	12.08324623		
288	0.78539819	0.989358572	0.876796221	1.4142134189605713,1.41421341	0.77704034	0.77704034	2.000200033		
289	0.757547712	5.808841127	5.055853649	9.899494171142578,9.192388534	5.439282512	5.050762931	11.21359158		
290	0.341475028	28.37977941	16.5839695	39.0567626953125,34.492736816	21.45975972	18.9520532	44.92724228		
291	0.834946792	5.211710421	4.76222244	7.77817440032959,9.192388534	5.050762931	4.273722198	11.18054008		
292	0.813058615	2.430959648	2.191990552	2.8284268379211426,4.949747085	2.719641256	1.55408068	5.385364532		
293	0.725530568	2.183620026	1.85996566	3.535533905029297,3.535533905	1.942601047	1.942601047	5.000199795		
294	0.61492967	7.43776723	5.83250751	15.55634880065918,7.071067810	8.547444396	3.885202093	16.55314636		
295	0.857371624	1.339149656	1.239977107	2.1213202476501465,2.121320247	1.165560576	1.165560576	3.000200033		
296	0.605013573	4.179915087	3.251247403	5,9	4.945054945	2.747252747	9.27648735		
297	0.696614445	2.678299312	2.235400519	2.8284268379211426,5.656853675	3.10816136	1.55408068	5.831151485		
298	0.666663013	7.497781825	6.121896447	17.19570541381836,7.488452911	9.448189788	4.114534567	17.20485115		
299	0.871464655	1.936280383	1.807561715	3.535533905029297,2.828426837	1.942601047	1.55408068	4.123305321		
300	0.64490173	22.67338968	18.20804102	49.51962661743164,24.80770111	27.20858605	13.63060501	49.81987381		
301	0.857371624	1.339149656	1.239977107	2.1213202476501465,2.121320247	1.165560576	1.165560576	3.000200033		
302	0.733526084	15.19318573	13.01237671	21.86506462097168,24.16665077	13.27837955	12.01377177	29.15496063		
303	0.541716483	6.21864689	4.577010645	5,15	8.241758242	2.747252747	15.03349686		
304	0.696614455	2.678299291	2.235400519	2.8284268379211426,5.656853675	3.10816136	1.55408068	5.831151485		
305	0.785190356	24.05497618	21.31534712	38.94856262207031,38.07440185	21.40030913	20.92000102	44.00055695		
306	0.416187641	26.56352838	17.13682456	40.72776794433594,31.82048797	22.37789447	17.4837846	43.6448555		
307	0.68118969	14.91069027	12.3064214	22.48902130126953,27.17423629	14.93089906	12.35660511	31.12845421		
308	0.690660475	4.017448968	3.338740538	4.242640495300293,8.485280990	4.662242303	2.331121151	8.944472313		
309	0.369684697	24.34475242	14.80202383	48.786476135253906,21.66604614	26.80575612	11.90442096	48.82641983		
310	0.43856484	17.40166452	11.52411802	21.390993118286133,25.09069252	13.78609479	11.75329292	29.08878326		
311	0.735999074	4.119900409	3.534478562	6.363960266113281,7.071067810	3.885202093	3.496681465	8.246411324		
312	0.48385624	21.41911351	14.89908733	41.59575653076172,18.94092369	22.85481128	10.40710093	41.87330246		
313	0.548349069	4.469691471	3.30983315	6.363960266113281,6.363960266	3.496681465	3.496681465	8.135735512		
314	0.875477102	4.614579632	4.317720649	7.071067810058594,7.778174400	4.273722198	3.885202093	9.219744682		
315	0.662003601	11.32790589	9.216792457	19.000003814697266,18.80000114	10.43956254	10.32967096	23.25154495		
316	0.554102406	6.610874735	4.921006584	14.443709373474121,6.827935695	7.936104051	3.75161302	15.03349686		
317	0.428381525	21.49670614	14.06977932	16.5423641204834,49.191761016	27.02844012	9.089211055	49.25463867		
318	0.78539819	0.989358572	0.876796221	1.4142134189605713,1.41421341	0.77704034	0.77704034	2.000200033		
319	0.640076871	3.377881459	2.70246745	4.599999904632568,6.600000381	3.626373836	2.527472475	7.159110546		
320	0.772150891	3.42031822	3.005506022	5.656853675842285,4.949747085	3.10816136	2.719641256	7.071267605		
321	0.483958918	27.18551793	18.91219164	56.0156364440918,27.939138412	30.77782222	15.35117495	56.51568222		
322	0.808518153	3.377881459	3.037311204	5.656853675842285,5.656853675	3.10816136	3.10816136	7.071267605		
323	0.557488707	20.59222107	15.37521612	24.29814910888672,43.311943054	23.79777091	13.35063138	43.35906601		
324	0.539012092	0.597130727	0.43839811	0.7071067094802856,1.41421341	0.77704034	0.38852017	1.414413571		
325	0.823490715	5.066822177	4.597958171	8.485280990600586,8.485280990	4.662242303	4.662242303	11.04556084		
326	0.570550101	16.59963093	12.53849252	16.54690170288086,39.35479354	21.62351294	9.091704232	39.84991455		
327	0.857371624	1.339149656	1.239977107	2.1213202476501465,2.121320247	1.165560576	1.165560576	3.000200033		
328	0.553535643	9.204300545	6.847997398	7.589465618133545,22.45217132	12.33635787	4.170036054	22.5612278		
329	0.602282136	7.557796567	5.865366967	18.238800048828125,7.06018066	10.02131871	3.879220145	18.24848747		
330	0.697385473	20.50734757	17.12560569	23.667875289916992,42.90756607	23.57558575	13.00432708	42.95366287		
331	0.731646609	10.44099866	8.930848493	14.600000381469727,20.79999923	11.42857101	8.021978232	23.76992798		
332	0.871464655	1.936280383	1.807561715	2.8284268379211426,3.535533905	1.942601047	1.55408068	4.123305321		
333	0.454048169	30.11115693	20.28982993	48.070838928222656,33.39559173	26.41254886	18.34922623	52.2572937		

1.6.2 Particle Size distribution - Table

334	0.599157104	9.409203426	7.283216409	21.259645462036133,9.13500308	11.68112388	5.019232467	21.37775803		
335	0.586315904	7.702684748	5.898043359	8.049844741821289,17.44132995	9.583148328	4.422991616	17.49305534		
336	0.799492389	12.62461872	11.28821929	21.0190372467041,20.57182502	11.54892156	11.30320056	23.65353012		
337	0.64801865	12.33484236	9.929499449	25.298980712890625,14.9048271	13.90053885	8.189465449	25.94244385		
338	0.720941829	16.63478678	14.12431356	29.451459884643555,22.0885925	16.18212082	12.1365893	32.31215286		
339	0.672642567	17.19676176	14.10388793	19.798988342285156,37.4766578	20.59157026	10.87856502	37.64325714		
340	0.857371624	1.339149656	1.239977107	2.1213202476501465,2.12132024	1.165560576	1.165560576	3.000200033		
341	0.857371624	1.339149656	1.239977107	2.1213202476501465,2.12132024	1.165560576	1.165560576	3.000200033		
342	0.736688254	9.596528411	8.236752686	17,16	9.340659341	8.791208791	19.41668892		
343	0.50714026	7.412908344	5.279012309	17.2624454498291,7.5131893157	9.484860137	4.128125998	17.26287651		
344	0.64964935	31.24540362	25.18405409	45.76881408691406,51.0775032	28.0645622	25.14770005	57.9865303		
345	0.78539819	0.989358572	0.876796221	1.4142134189605713,1.41421341	0.77704034	0.77704034	2.000200033		
346	0.546288767	8.197364035	6.058782427	10.412663459777832,15.4020643	8.462672705	5.721243659	16.12471581		
347	0.616820046	11.98505128	9.412806614	13.792698860168457,24.9992656	13.73586026	7.578405967	25.00020027		
348	n/A	0	0	0,0	0	0	0.0002		
349	0.808518153	3.377881459	3.037311204	5.656853675842285,5.65685367	3.10816136	3.10816136	7.071267605		
350	0.636078542	8.462281616	6.749053122	16,11	8.791208791	6.043956044	17.6007061		
351	0.857371624	1.339149656	1.239977107	2.1213202476501465,2.12132024	1.165560576	1.165560576	3.000200033		
352	0.800106387	1.833828942	1.640335528	3.535533905029297,2.121320247	1.942601047	1.165560576	4.123305321		
353	0.546498374	7.165568701	5.297184522	17.04930877685547,6.70478487	9.367752075	3.683947731	17.08820724		
354	0.501601377	9.264315203	6.561342111	21,9	11.53846154	4.945054945	21.37775803		
355	0.811767614	4.32480329	3.896567637	7.77817440032959,7.0710678100	4.273722198	3.885202093	8.630964279		
356	0.857371637	2.678299291	2.479954213	4.242640495300293,4.24264049	2.331121151	2.331121151	5.37501812		
357	0.610620806	22.01624425	17.20398416	24.092885971069336,49.0401268	26.94512462	13.23784943	50.35891342		
358	0.828640632	12.94955098	11.78793744	23,19	12.63736264	10.43956044	26.41988945		
359	0.78539819	0.989358572	0.876796221	1.4142134189605713,1.41421341	0.77704034	0.77704034	2.000200033		
360	0.60187308	13.11929799	10.17801429	13.962528228759766,29.0934696	15.98542287	7.671718807	30.08341789		
361	0.80378591	2.678299291	2.401205342	3.535533905029297,4.94974708	2.719641256	1.942601047	5.440038204		
362	0.78539819	0.989358572	0.876796221	1.4142134189605713,1.41421341	0.77704034	0.77704034	2.000200033		
363	0.767287001	5.706389728	4.998507518	11.31370735168457,7.07106781	6.216322721	3.885202093	12.64931107		
364	0.514855997	6.465986554	4.639569501	6.88709020614624,14.13195037	7.7648079	3.784115498	14.17839336		
365	0.78539819	0.989358572	0.876796221	1.4142134189605713,1.41421341	0.77704034	0.77704034	2.000200033		
366	0.544340496	10.87566333	8.024002901	19.195070266723633,14.4781923	10.5467419	7.95505073	19.67558479		
367	0.943271167	1.68894074	1.640335528	2.8284268379211426,2.82842683	1.55408068	1.55408068	3.162477732		
368	0.661957718	4.572142912	3.71993132	10.027548789978027,5.1994695	5.509642192	2.85685141	10.19823933		
369	0.806145086	3.275430018	2.940863928	5.656853675842285,4.94974756	3.10816136	2.719641518	7.071267605		
370	0.29301767	33.42174278	18.09155919	26.554777145385742,57.2587356	31.46084377	14.59053689	57.27323151		
371	0.425213987	10.4585767	6.819874087	27,7	14.83516484	3.846153846	27.00020027		
372	0.401257446	18.92813916	11.99000799	21.615310668945312,28.5998477	15.71420208	11.87654432	31.57550621		
373	0.512110307	13.79402137	9.871261413	33,10	18.13186813	5.494505495	33.13628006		
374	0.539012092	4.179915087	3.068786772	7,5	3.846153846	2.747252747	7.9651227		
375	0.545179446	11.32790579	8.364102865	24.220911026000977,12.7689428	13.30819287	7.015902655	24.35179138		
376	0.659471644	13.57154058	11.0211576	31.010307312011172,13.2701768	17.03863039	7.291305982	31.32111931		
Mean	0.686209	8.57238	6.652214		9.208779	5.629091	17.85953		
S.d	0.145516	7.454936	5.316048		7.765233	4.883726	14.73769		
Min	0	0	0		0	0	0.0002		
Max	0.943271	42.1738	25.18405		31.46084	28.99517	71.47047		