Particle size measurment with distribution

IMAGE FORMAT CONVERSION FROM TIF TO JPG AND THEN CROPPING

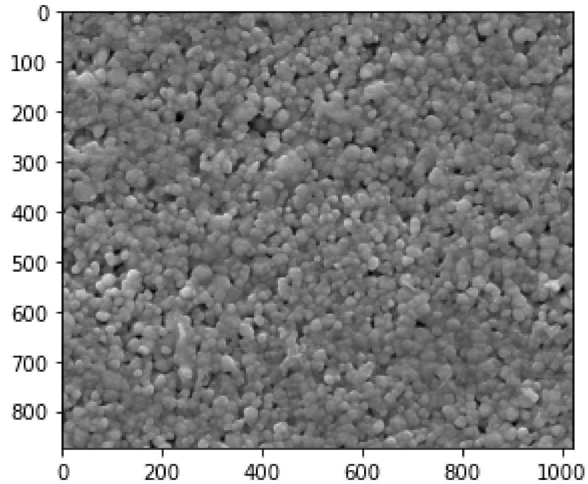
In [2]:
```
from PIL import Image

image_path='/Users/shivashankar/Downloads/grain_image/8/DS-115_003'

# Load the TIFF image
tiff_image = Image.open(image_path+'.tif')  # Replace 'input.tif' with your TI

# Save it as a JPEG image
tiff_image.save(image_path+'.jpg', 'JPEG')  # 'output.jpg' is the desired outp
```

In [3]:
```
# Importing Image class from PIL module
from PIL import Image

# Opens a image in RGB mode
im = Image.open(image_path+'.jpg')

# Size of the image in pixels (size of original image)
# (This is not mandatory)
width, height = im.size

# Setting the points for cropped image
left = 0
top = 0
right = width
bottom = height-70

# Cropped image of above dimension
# (It will not change original image)
im1 = im.crop((left, top, right, bottom))


# Shows the image in image viewer
im1.save(image_path+'.jpg')
```

THE ORIGINAL IMAGE

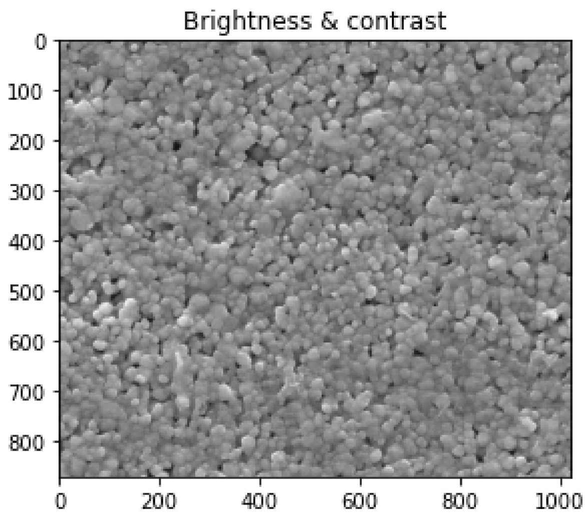
In [4]:
```
import cv2
import numpy as np
import matplotlib.pyplot as plt


image = cv2.imread(image_path+'.jpg',1)

plt.imshow(image)
#plt.savefig("/Users/shivashankar/Downloads/original.jpg")
```

Out[4]:
```
<matplotlib.image.AxesImage at 0x11658e2b0>
```



THE IMAGE AFTER BRIGHTNESS ADJUSTED

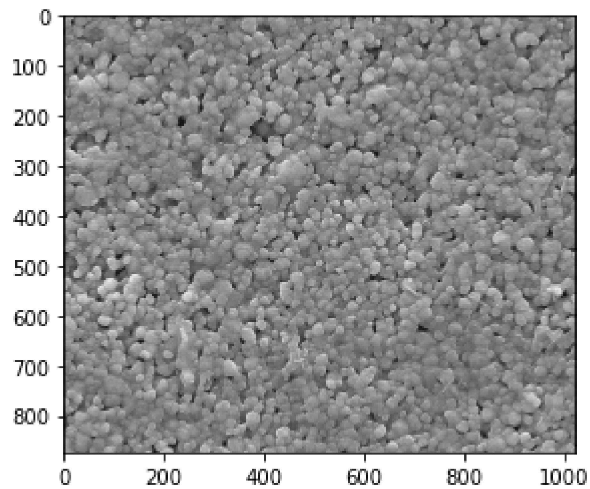
In [5]:
```
# Adjust the brightness and contrast
# Adjusts the brightness by adding 10 to each pixel value
brightness = 20
# Adjusts the contrast by scaling the pixel values by 2.3
contrast = 1

image2 = cv2.addWeighted(image, contrast, np.zeros(image.shape, image.dtype),


plt.title("Brightness & contrast")
plt.imshow(image2)
plt.show()
```



THE IMAGE AFTER SHARPENING

In [6]:
```
# Create the sharpening kernel
kernel = np.array([[0, -1, 0], [-1, 5, -1], [0, -1, 0]])

# Sharpen the image
image = cv2.filter2D(image2, -1, kernel)

plt.imshow(image)
#plt.savefig("/Users/shivashankar/Downloads/sharpened.jpg")
```

Out[6]:
```
<matplotlib.image.AxesImage at 0x1166bf580>
```



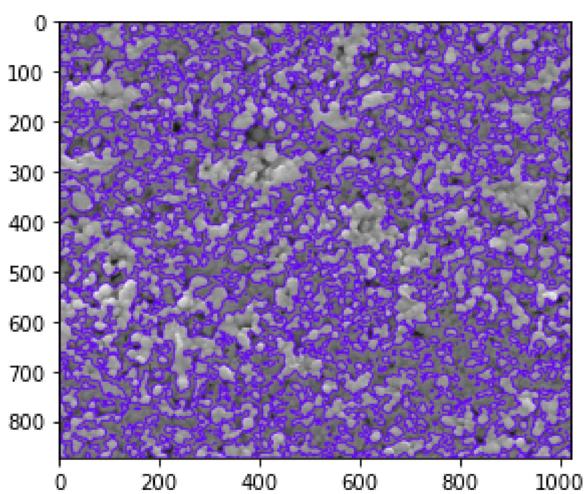
THE IMAGE AFTER BLURRING, THRESHOLDING TO SEPARATE PARTICLES, DRAW CONTOURS AROUND PARTICLES

In [7]:

```
# Convert the cropped image to grayscale
gray_cropped = cv2.cvtColor(image2, cv2.COLOR_BGR2GRAY)

# Apply Gaussian blur to reduce noise
blurred = cv2.GaussianBlur(gray_cropped, (5, 5), 0)

# Apply thresholding to segment particles
threshold_value, thresholded = cv2.threshold(blurred, 0, 255, cv2.THRESH_BINAR

# Find contours of particles
contours, threshold_value = cv2.findContours(thresholded, cv2.RETR_EXTERNAL, c

# Calculate particle diameters and collect them in a list
particle_diameters = []
for contour in contours:
    (x, y), radius = cv2.minEnclosingCircle(contour)
    diameter = radius * 2
    particle_diameters.append(diameter)

# Draw particle boundaries
cv2.drawContours(image2, contours, -1, (100, 0, 255), 2)  # Green color, thick

average_diameter = np.mean(particle_diameters)
print("Average Diameter of Particles:", average_diameter,"(pixels)")



# Display the cropped image with particle boundaries
plt.imshow(image2)
```

```
Average Diameter of Particles: 22.864645661164303 (pixels)
```

Out[7]: <matplotlib.image.AxesImage at 0x11674d700>



In [8]:
```
for i in range(len(particle_diameters)):
    particle_diameters[i]=round(particle_diameters[i],2)
```

In [9]:
```
particle_diameters.sort()
print(particle_diameters)
```

```
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.
0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
1.0, 1.41, 1.41, 1.41, 1.41, 1.41, 1.41, 1.41, 1.41, 1.41, 1.41, 1.41, 1.41,
1.41, 1.41, 1.41, 1.41, 1.41, 1.41, 1.41, 1.41, 1.41, 1.41, 1.41, 1.41, 1.41,
1.41, 1.41, 1.41, 1.41, 1.41, 1.41, 1.41, 1.41, 1.41, 1.41, 1.41, 1.41, 1.41,
2.0, 2.0, 2.0, 2.0, 2.0, 2.0, 2.0, 2.0, 2.0, 2.0, 2.0, 2.0, 2.24, 2.24, 2.24,
2.24, 2.24, 2.24, 2.24, 2.24, 2.24, 2.24, 2.24, 2.24, 2.24, 2.24, 2.24, 2.24,
2.24, 2.24, 2.24, 2.24, 2.24, 2.24, 2.24, 2.24, 2.24, 2.24, 2.24, 2.36, 2.36,
2.36, 2.5, 2.5, 2.83, 2.83, 2.83, 2.83, 2.83, 2.83, 2.83, 2.83, 2.83, 2.83,
2.83, 2.83, 2.83, 2.83, 2.83, 3.0, 3.0, 3.0, 3.0, 3.0, 3.0, 3.0, 3.0, 3.0, 3.
16, 3.16, 3.16, 3.16, 3.16, 3.16, 3.16, 3.16, 3.16, 3.16, 3.16, 3.16, 3.16,
3.16, 3.16, 3.16, 3.16, 3.16, 3.16, 3.16, 3.16, 3.16, 3.33, 3.54, 3.61, 3.61,
3.61, 3.61, 3.61, 3.61, 3.61, 3.61, 3.61, 3.61, 3.61, 3.61, 3.61, 3.61, 3.61,
3.64, 3.64, 3.64, 3.68, 3.68, 3.68, 3.8, 4.0, 4.0, 4.12, 4.12, 4.12, 4.12, 4.
12, 4.12, 4.12, 4.12, 4.12, 4.12, 4.24, 4.24, 4.24, 4.27, 4.47, 4.47, 4.47,
4.47, 4.47, 4.47, 4.47, 4.47, 4.47, 4.47, 4.47, 4.47, 4.47, 4.47, 4.47, 4.47,
4.47, 4.47, 5.0, 5.0, 5.0, 5.0, 5.0, 5.05, 5.05, 5.05, 5.05, 5.1, 5.1, 5.1,
5.1, 5.1, 5.1, 5.15, 5.2, 5.39, 5.39, 5.39, 5.39, 5.39, 5.39, 5.39, 5.39, 5.3
9, 5.39, 5.39, 5.41, 5.49, 5.53, 5.66, 5.66, 5.66, 5.68, 5.83, 5.83, 5.83, 5.
83, 5.83, 5.83, 5.83, 5.83, 5.86, 5.96, 6.0, 6.0, 6.08, 6.08, 6.08, 6.09, 6.1
5, 6.18, 6.32, 6.32, 6.32, 6.32, 6.32, 6.32, 6.32, 6.36, 6.4, 6.4, 6.4, 6.4,
6.4, 6.44, 6.44, 6.71, 6.71, 6.71, 6.71, 6.71, 6.71, 6.71, 6.82, 6.82, 7.07,
7.07, 7.07, 7.07, 7.07, 7.07, 7.07, 7.07, 7.21, 7.21, 7.21, 7.28, 7.28, 7.28,
7.34, 7.45, 7.62, 7.62, 7.62, 7.62, 7.81, 7.84, 7.85, 8.06, 8.06, 8.06, 8.06,
8.06, 8.13, 8.25, 8.25, 8.25, 8.25, 8.25, 8.26, 8.29, 8.31, 8.5, 8.54, 8.54,
8.54, 8.54, 8.6, 8.61, 8.69, 8.94, 8.94, 8.94, 8.94, 8.94, 8.94, 8.94, 8.94,
9.06, 9.06, 9.06, 9.22, 9.22, 9.22, 9.22, 9.22, 9.22, 9.22, 9.43, 9.43, 9.43,
9.43, 9.46, 9.49, 9.49, 9.56, 9.72, 9.85, 10.0, 10.0, 10.0, 10.05, 10.09, 10.
09, 10.1, 10.2, 10.33, 10.44, 10.5, 10.63, 10.63, 10.77, 10.81, 10.82, 11.0,
11.09, 11.1, 11.18, 11.18, 11.18, 11.24, 11.25, 11.25, 11.4, 11.4, 11.4, 11.
4, 11.4, 11.58, 11.66, 11.68, 11.7, 11.75, 11.78, 11.97, 12.04, 12.08, 12.09,
12.21, 12.51, 12.53, 12.53, 12.65, 12.67, 12.69, 12.7, 12.81, 12.81, 12.82, 1
2.9, 12.94, 13.0, 13.04, 13.04, 13.04, 13.04, 13.07, 13.08, 13.15, 13.15, 13.
42, 13.6, 13.84, 13.89, 13.89, 13.91, 13.98, 14.1, 14.14, 14.18, 14.18, 14.3
2, 14.32, 14.32, 14.42, 14.43, 14.43, 14.47, 14.77, 14.87, 14.87, 14.87, 14.9
8, 15.0, 15.0, 15.0, 15.23, 15.26, 15.29, 15.3, 15.31, 15.48, 15.49, 15.55, 1
5.55, 15.67, 15.81, 15.81, 16.12, 16.12, 16.12, 16.12, 16.17, 16.17, 16.18, 1
6.19, 16.28, 16.28, 16.33, 16.55, 16.64, 16.76, 16.8, 16.98, 17.04, 17.13, 1
7.26, 17.26, 17.3, 17.49, 17.52, 17.52, 17.54, 17.72, 17.89, 17.91, 18.03, 1
8.03, 18.03, 18.03, 18.03, 18.06, 18.25, 18.25, 18.41, 18.68, 18.69, 18.87, 1
8.87, 18.97, 19.11, 19.28, 19.42, 19.6, 19.7, 19.72, 19.85, 19.85, 19.85, 20.
05, 20.12, 20.25, 20.25, 20.4, 20.47, 20.47, 20.62, 20.88, 20.88, 21.19, 21.
2, 21.3, 21.35, 21.38, 21.4, 21.47, 21.47, 21.59, 21.61, 21.68, 21.82, 21.84,
21.93, 21.95, 22.02, 22.26, 22.31, 22.47, 22.47, 22.47, 22.58, 22.65, 22.81,
23.09, 23.26, 23.62, 23.82, 24.14, 24.26, 24.35, 24.48, 24.74, 24.84, 24.84,
24.85, 24.92, 24.97, 25.01, 25.2, 25.24, 25.37, 26.08, 26.1, 26.17, 26.4, 26.
57, 27.09, 27.2, 27.29, 27.31, 27.31, 27.66, 27.68, 28.24, 28.47, 28.48, 28.6
7, 29.07, 29.07, 29.07, 29.08, 29.12, 29.15, 29.27, 29.57, 30.41, 30.48, 30.6
5, 30.87, 30.89, 31.15, 31.39, 32.14, 32.32, 32.45, 32.7, 32.8, 33.04, 33.29,
33.53, 33.62, 33.7, 34.13, 34.23, 34.41, 34.67, 34.85, 35.28, 35.85, 36.31, 3
6.31, 36.36, 36.4, 36.77, 36.9, 37.64, 38.28, 38.83, 39.2, 39.81, 40.79, 41.0
3, 41.23, 41.37, 41.59, 42.22, 42.29, 42.57, 43.01, 43.08, 44.05, 44.43, 44.8
```

```
5, 45.49, 45.88, 46.33, 46.39, 47.51, 49.04, 49.05, 49.24, 50.0, 50.1, 50.29,
50.54, 51.66, 51.67, 52.35, 53.25, 53.46, 53.46, 53.85, 56.4, 57.77, 58.46, 5
9.67, 59.67, 60.0, 60.78, 61.13, 61.45, 62.51, 66.37, 66.49, 66.71, 67.57, 6
7.62, 68.0, 68.37, 68.68, 69.38, 70.52, 70.83, 76.73, 78.43, 78.49, 78.6, 79.
08, 79.13, 81.61, 82.08, 84.15, 84.65, 84.86, 86.47, 86.77, 89.28, 90.38, 91.
6, 91.83, 92.82, 92.92, 93.61, 95.89, 96.4, 96.8, 96.88, 97.74, 101.83, 101.9
7, 104.8, 108.58, 109.02, 112.72, 115.13, 115.74, 122.67, 124.34, 126.87, 12
7.03, 129.99, 131.49, 136.96, 142.64, 148.19, 151.16, 154.49, 156.12, 158.51,
161.06, 167.62, 175.88, 190.38, 194.88, 195.21, 202.81, 207.99, 230.79, 237.5
7, 261.02, 274.61, 275.77, 278.15, 315.99, 358.93, 519.17]
```

In [10]:
```
temp=particle_diameters
```

In [11]:
```
print(len(temp))
```

```
823
```

In [12]:
```
temp_copy = temp[:]
for i in range(len(temp_copy)):
    if temp_copy[i] == 0.0 or temp_copy[i] >= 90 or temp_copy[i]<20:
        temp.remove(temp_copy[i])
```

In [13]:
```
print(len(particle_diameters))
```

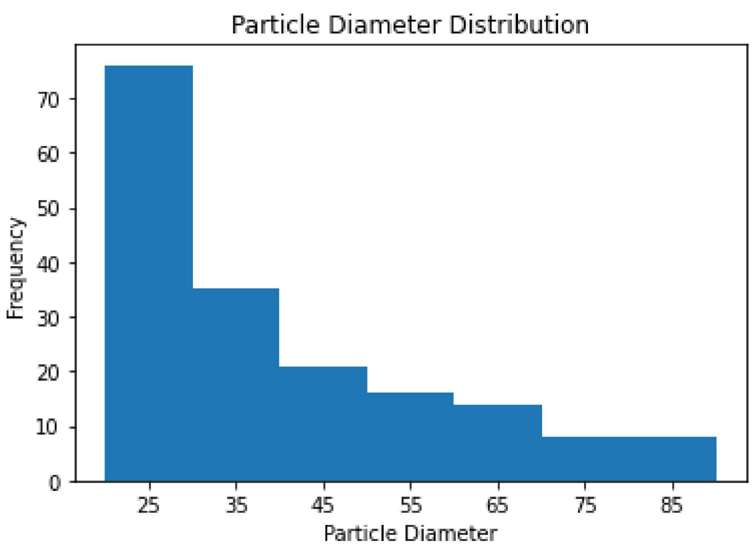
```
178
```

MEAN SIZE OF THE PARTICLE after excluding outliers and cluserters

In [16]:
```
print(np.mean(particle_diameters))
```

```
39.56837078651685
```

In [14]:
```
import matplotlib.pyplot as plt
import numpy as np


# Define the bins for particle diameter ranges
bins = [20, 30, 40, 50,60,70,80,90]  # Adjust these bins according to your dat

# Create a histogram of the particle diameters
hist, bins = np.histogram(particle_diameters, bins=bins)

# Calculate the midpoints of each bin for labeling the x-axis
bin_midpoints = [(bins[i] + bins[i + 1]) / 2 for i in range(len(bins) - 1)]

# Create the bar chart
plt.bar(bin_midpoints, hist, width=10, align='center')

# Label the axes
plt.xlabel('Particle Diameter')
plt.ylabel('Frequency')

# Set the x-axis ticks to be the midpoints of the bins
plt.xticks(bin_midpoints)

# Add a title to the chart
plt.title('Particle Diameter Distribution')

# Show the chart
plt.show()
```



In [145]:
```
# import matplotlib.pyplot as plt
# import numpy as np



# # Calculate the number of bins using the Freedman-Diaconis rule
# data_range = max(particle_diameters) - min(particle_diameters)
# print(data_range)
# bin_width = 2 * np.percentile(particle_diameters, 75) / (len(particle_diamet
# print(bin_width)
# num_bins = int(data_range / bin_width)
# print(num_bins)

# # Create the histogram
# plt.hist(particle_diameters, bins=num_bins, edgecolor='k', alpha=0.75)

# # Label the axes
# plt.xlabel('Particle Diameter')
# plt.ylabel('Frequency')

# # Add a title to the chart
# plt.title('Particle Diameter Distribution')

# # Show the chart
# plt.show()
```

In [ ]: