Load environment

In [3]:
```
import sys
sys.path.append('/Users/qingjunwang/Documents/code/python')
import main_check_focus_exp as main
import matplotlib.pyplot as plt
import os
import cv2
import numpy as np
import re
import io
import os
```

Operation area

Single exposure image creation and focus find

In [2]:
```
#modified equation
def overexposure_file_new(folder):
    files = os.listdir(folder)
    overexposed_files = []
    for file in files:
        file_path = os.path.join(folder, file)
        img = cv2.imread(file_path, cv2.IMREAD_UNCHANGED)  # Read the image
        if img is None:
            print(f"Failed to load image at {file_path}")
            continue
        if len(img.shape) == 2:  # Grayscale image
            # Check for overexposure in the grayscale image
            if np.sum(img > 254)>1000:
                overexposed_files.append(file)
                print(f"{file} is overexposed in grayscale.")
        elif len(img.shape) == 3 and img.shape[2] == 3:  # Color image
            blue, green, red = cv2.split(img)
            if np.sum(red > 254)>1000 or np.sum(green > 254)>1000 or np.sum(
                overexposed_files.append(file)
                print(f"{file} is overexposed in color.")
    overexposed_files = np.array(overexposed_files)
    print(overexposed_files)
    return overexposed_files
```

In [2]:
```
folder=[
        # '/Users/qingjunwang/Documents/image files/speckle 2 study/yuhang/Y
        # '/Users/qingjunwang/Documents/image files/speckle 2 study/yuhang/Y
        # '/Users/qingjunwang/Documents/image files/speckle 2 study/yuhang/Y
        # '/Users/qingjunwang/Documents/image files/speckle 2 study/yuhang/Y
    # '/Users/qingjunwang/Documents/image files/speckle 2 study/yuhang/8um_0
    #     '/Users/qingjunwang/Documents/image files/Spectrum_contrast/Green-
```

```
    # '/Users/qingjunwang/Documents/image files/Spectrum_contrast/Green-P20-
    #     '/Users/qingjunwang/Documents/image files/Spectrum_contrast/Green-
    #     '/Users/qingjunwang/Documents/image files/Spectrum_contrast/Green-
    #     '/Users/qingjunwang/Documents/image files/Spectrum_contrast/Green-
    #     '/Users/qingjunwang/Documents/image files/Spectrum_contrast/Green-
    # '/Users/qingjunwang/Documents/image files/speckle 2 study/yuhang/FL2 2
    # '/Users/qingjunwang/Documents/image files/speckle 2 study/yuhang/20um_
    # '/Users/qingjunwang/Documents/image files/speckle 2 study/yuhang/20um_
    # '/Users/qingjunwang/Documents/image files/speckle 2 study/yuhang/14um_
    #     '/Users/qingjunwang/Documents/image files/speckle 2 study/yuhang/1
    #     '/Users/qingjunwang/Documents/image files/speckle 2 study/yuhang/8
    #         '/Users/qingjunwang/Documents/image files/speckle 2 study/yuha
        # '/Users/qingjunwang/Documents/image files/speckle 2 study/yuhang/8
    # '/Users/qingjunwang/Documents/image files/speckle 2 study/EPIC-FL1/NAP
    '/Users/qingjunwang/Documents/image files/FL3_YSSI_P12_rSM'

]
for item in folder:
    single_folder = item+'_single_image_folder_less1000hotspot'
    exposure_time_array=main.exposure_time_array(item)
    # overexposure_file=main.overexposure_file(item)
    overexposure_file=overexposure_file_new(item)
    # Example usage
    main.single_image_folder(item, single_folder, exposure_time_array, overe
```

```
exposure time: [160000 108863  74070  50397  34290  23331  15874  10801   73
49   5000]
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[2], line 29
     27 exposure_time_array=main.exposure_time_array(item)
     28 # overexposure_file=main.overexposure_file(item)
---> 29 overexposure_file=overexposure_file_new(item)
     30 # Example usage
     31 main.single_image_folder(item, single_folder, exposure_time_array, o
verexposure_file)

NameError: name 'overexposure_file_new' is not defined
```

In [5]:
```
infocus=[]
```

In [18]:
```
infocus1=main.find_the_infocus_image(single_folder)
```

```
Image with highest contrast: EPIC-FL1_D0C0_NAP-20um-P20-0.25r_Green__X0.000_
Y0.000_Z34.927_Exp898.png
Contrast: 1.8691804317047733
```

In [53]:
```
infocus.append(infocus1)
```

In [58]:
```
import pandas as pd

df = pd.DataFrame(infocus, columns=['Filename'])
filename = '/Users/qingjunwang/Documents/image files/speckle 2 study/infocus
df.to_csv(filename, index=False)
print(f"File saved: {filename}")
```

```
File saved: /Users/qingjunwang/Documents/image files/speckle 2 study/infocus
_filenames.csv
```

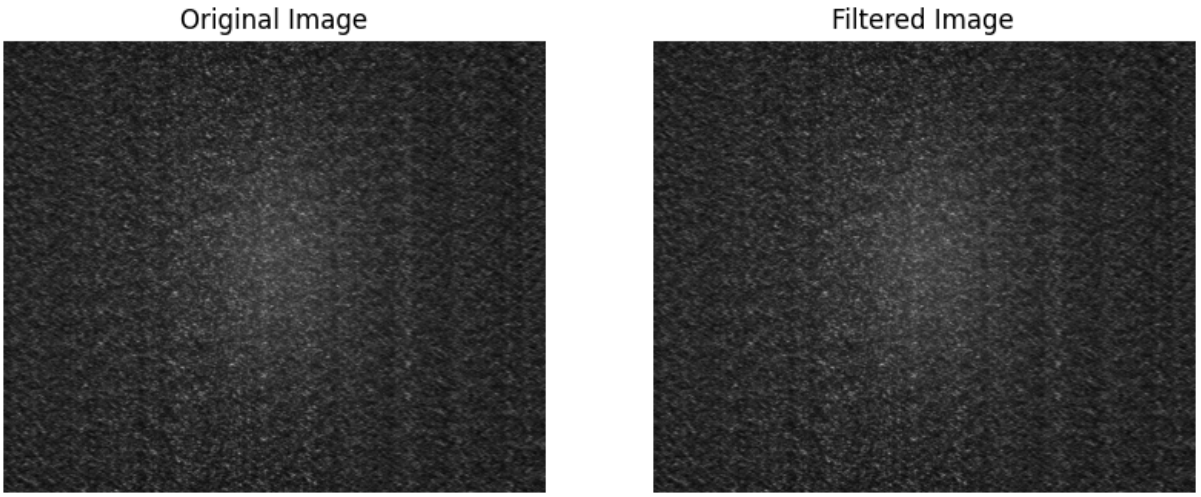
In [55]:
```
z_options_array=main.z_plane(folder)
```

```
[34.741 34.691 34.641 34.591 34.541 34.491 34.441 34.391 34.341 34.291
 34.241 34.191 34.141 34.091 34.041 33.991 33.941 33.891 33.841 33.791
 33.741 33.691 33.641 33.591 33.541 33.491 33.441 33.391 33.341 33.291
 33.241 33.191 33.141 33.091 33.041 32.991 32.941 32.891 32.841 32.791
 32.741 32.691 32.641 32.591 32.541 32.491 32.441 32.391 32.341 32.291
 32.241 32.191 32.141 32.091 32.041 31.991 31.941 31.891 31.841 31.791
 31.741 31.691 31.641 31.591 31.541 31.491 31.441 31.391 31.341 31.291
 31.241 31.191 31.141 31.091 31.041 30.991 30.941 30.891 30.841 30.791
 30.741 30.691 30.641 30.591 30.541 30.491 30.441 30.391 30.341 30.291
 30.241 30.191 30.141 30.091 30.041 29.991 29.941 29.891 29.841 29.791
 29.741]
num of z planes: 101
```

Filter out low frequency information- function completed and included in the contrast function

In [1]:
```
image_path = "/Users/qingjunwang/Documents/image files/speckle 2 study/EPIC-

import numpy as np
import cv2
from scipy.ndimage import convolve
import matplotlib.pyplot as plt

def get_2d_gaussian(size, sigma):
    """Generate a 2D Gaussian kernel."""
    kernel = cv2.getGaussianKernel(ksize=size, sigma=sigma)
    gaussian_kernel = np.outer(kernel, kernel.transpose())
    return gaussian_kernel

def get_filtered(image, low_pass=0, high_pass=0):
    """Apply low-pass and high-pass filters to the image."""
    if low_pass == 0:
        im_filtered_low = image
    else:
        size = min(max(int(3 * low_pass), 4), image.shape[1])
        gaussian_kernel = get_2d_gaussian(size, low_pass)
        im_filtered_low = convolve(image, gaussian_kernel, mode='reflect')
    if high_pass == 0:
        im_filtered = im_filtered_low
    else:
        size = min(int(2 * high_pass), im_filtered_low.shape[1])
        gaussian_kernel = get_2d_gaussian(size, high_pass)
        im_filtered_low_hp = convolve(im_filtered_low, gaussian_kernel, mode
        im_filtered = im_filtered_low / (im_filtered_low_hp + 1e-5)  # Addir
    return im_filtered

# Example usage
image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
if image is None:
    raise ValueError("Image not found or path is incorrect")
```

```
low_pass = 0  # Example value for low-pass filter
high_pass = 0  # Example value for high-pass filter #50 is good

filtered_image = get_filtered(image, low_pass, high_pass)

# Display the results using matplotlib
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.imshow(image, cmap='gray')
plt.title('Original Image')
plt.axis('off')

plt.subplot(1, 2, 2)
plt.imshow(filtered_image, cmap='gray')
plt.title('Filtered Image')
plt.axis('off')

plt.show()
```



In [62]:
```
## kernel_size: This parameter defines the size of the Gaussian kernel. It m
```

Contrast v.s. z for each PIC design

Check different color

In [47]:
```
z_data = {}
contrast_data = {}
```

In [48]:
```
# Define folders for each dataset
folders = {
    #Brian FL1
    '20um_0.25r_FL1': '/Users/qingjunwang/Documents/image files/speckle 2 st
    '20um_0.50r_FL1': '/Users/qingjunwang/Documents/image files/speckle 2 st
    '14um_0.25r_FL1': '/Users/qingjunwang/Documents/image files/speckle 2 st
    '14um_0.50r_FL1': '/Users/qingjunwang/Documents/image files/speckle 2 st
    '8um_0.25r_FL1': '/Users/qingjunwang/Documents/image files/speckle 2 stu
```

```
    '8um_0.50r_FL1': '/Users/qingjunwang/Documents/image files/speckle 2 stu

    #Yuhang FL1
    # '20um_0.25r_FL1': '/Users/qingjunwang/Documents/image files/speckle 2
    # '20um_0.50r_FL1':'/Users/qingjunwang/Documents/image files/speckle 2 s
    #  '14um_0r_FL1':    '/Users/qingjunwang/Documents/image files/speckle 2
    # '14um_0.25r_FL1': '/Users/qingjunwang/Documents/image files/speckle 2
    # '14um_0.50r_FL1': '/Users/qingjunwang/Documents/image files/speckle 2
    # '8um_0.25r_FL1': '/Users/qingjunwang/Documents/image files/speckle 2 s
    # # '8um_0.50r_FL1': '/Users/qingjunwang/Documents/image files/speckle 2
    # '8um_r0': '/Users/qingjunwang/Documents/image files/speckle 2 study/yu

    # '20um20um_non_interleaving_p_FL2': '/Users/qingjunwang/Documents/image

    # '80um_p_FL2': '/Users/qingjunwang/Documents/image files/speckle 2 stud
    # '80um_p_FL2_red': '/Users/qingjunwang/Documents/image files/speckle 2
    # '80um_p_FL2_blue': '/Users/qingjunwang/Documents/image files/speckle 2

    # '20um20um_MMSS_non_interleaving_p_FL2': '/Users/qingjunwang/Documents/
    # '20um20um_MMSS_non_interleaving_p_FL2_red': '/Users/qingjunwang/Docume
    # '20um20um_MMSS_non_interleaving_p_FL2_blue': '/Users/qingjunwang/Docum

    # '20um_p_FL2': '/Users/qingjunwang/Documents/image files/speckle 2 stud
    # '20um_p_FL2': '/Users/qingjunwang/Documents/image files/speckle 2 stud

    # '20um_0.05r_FL2':'/Users/qingjunwang/Documents/image files/speckle 2 s

    # '20um_0.05r_FL2_red': '/Users/qingjunwang/Documents/image files/speckl
    # '20um_0.05r_FL2_blue': '/Users/qingjunwang/Documents/image files/speck


    # '20um_p_FL2': '/Users/qingjunwang/Documents/image files/speckle 2 stud
    # '20um_p_FL2_red': '/Users/qingjunwang/Documents/image files/speckle 2
    # '20um_p_FL2_blue': '/Users/qingjunwang/Documents/image files/speckle 2
   # '20um_p_FL2_blue':  '/Users/qingjunwang/Documents/image files/speckle 2
}
# Dictionary to store z and contrast data

# Calculate z and contrast for each dataset
for label, folder in folders.items():
    z, contrast = main.z_contrast(folder,"green")
    z_data[label] = z
    contrast_data[label] = contrast
    print(f"folder contrast={label} finished")
```

```
len(z)=101
len(contrast)=101
folder contrast=20um_0.25r_FL1 finished
len(z)=101
len(contrast)=101
folder contrast=20um_0.50r_FL1 finished
len(z)=101
len(contrast)=101
folder contrast=14um_0.25r_FL1 finished
len(z)=101
len(contrast)=101
folder contrast=14um_0.50r_FL1 finished
len(z)=101
len(contrast)=101
folder contrast=8um_0.25r_FL1 finished
len(z)=101
len(contrast)=101
folder contrast=8um_0.50r_FL1 finished
```

In [50]:
```
contrast_data.keys()
```

Out[50]:
```
dict_keys(['20um_0.25r_FL1', '20um_0.50r_FL1', '14um_0.25r_FL1', '14um_0.50
r_FL1', '8um_0.25r_FL1', '8um_0.50r_FL1', '8um_r0', '20um_0.05r_FL2', '20um
_0.05r_FL2 redo', '20um_p_FL2'])
```

In [11]:
```
z_data['14um_0r_FL1']
```

Out[11]:
```
[32.775,
 33.475,
 29.875,
 31.025,
 34.425,
 30.525,
 33.375,
 32.075,
 29.975,
 29.775,
 32.875,
 34.325,
 31.125,
 31.625,
 29.675,
 30.325,
 32.175,
 33.275,
 31.825,
 30.225,
 30.425,
 31.725,
 33.575,
 32.675,
 31.925,
 31.525,
 31.325,
 34.225,
 32.475,
 33.775,
 34.025,
 33.875,
 29.475,
 31.425,
 33.075,
 32.375,
 32.975,
 30.825,
 32.275,
 33.175,
 33.975,
 30.025,
 34.125,
 30.625,
 29.575,
 30.925,
 33.675,
 32.575,
 30.725,
 30.125,
 31.225,
 32.225,
 33.125,
 31.575,
 30.675,
 29.525,
```

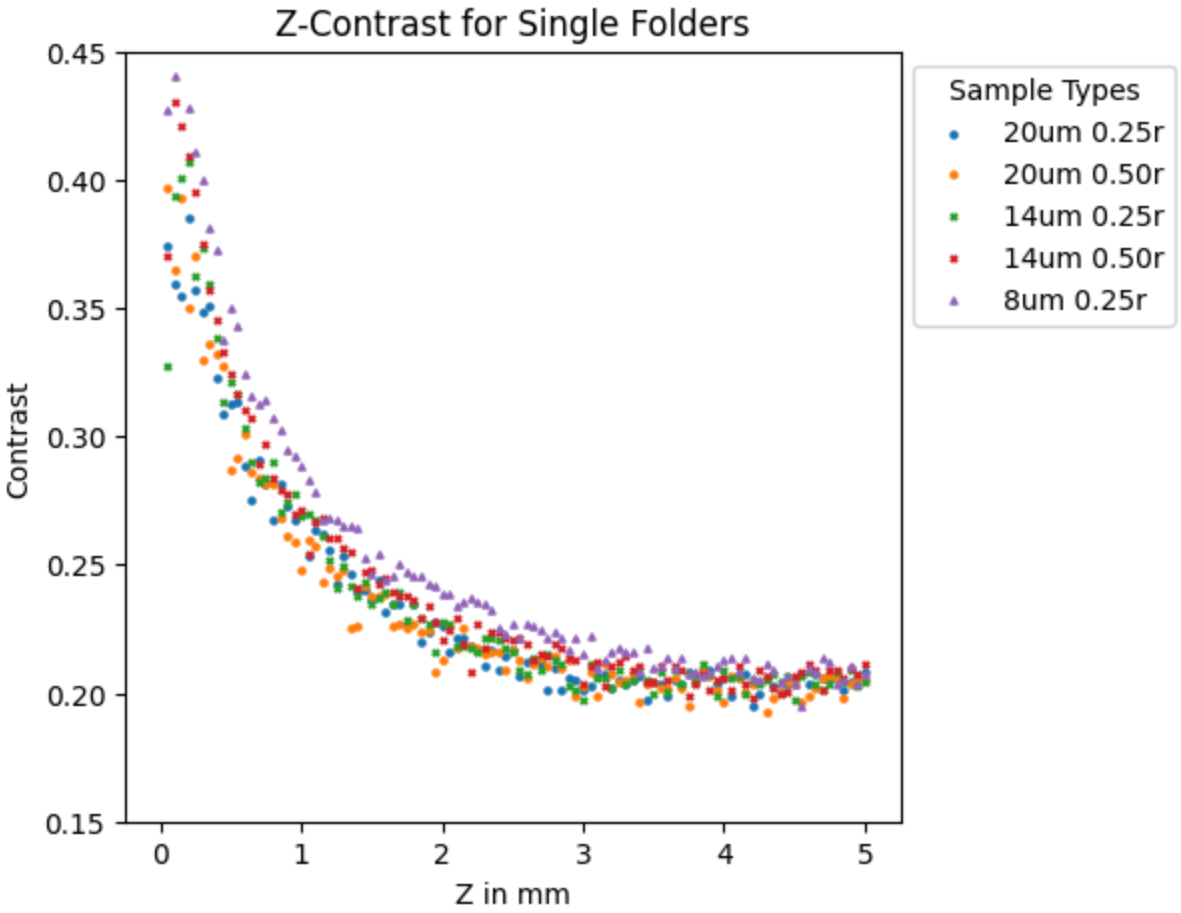
```
 33.925,
 34.175,
 33.625,
 33.725,
 32.525,
 30.175,
 31.275,
 30.775,
 31.475,
 31.375,
 30.075,
 32.425,
 30.875,
 34.075,
 33.825,
 33.025,
 30.975,
 32.325,
 34.275,
 32.925,
 30.375,
 32.125,
 33.225,
 31.875,
 30.475,
 30.275,
 31.175,
 29.925,
 33.525,
 32.625,
 29.825,
 32.725,
 33.425,
 31.675,
 30.575,
 29.625,
 31.075,
 34.475,
 33.325,
 31.975,
 32.025,
 32.825,
 34.375,
 29.725,
 31.775]
```

In [49]:

```
# Plotting the data
# plt.scatter(z_data['20um_0.05r'], contrast_data['20um_0.05r'], s=10, label
plt.figure(figsize=(5, 5))  # Set the figure size

focus_values = {
    '20um_0.25r_FL1': 34.927,
    '20um_0.50r_FL1': 34.741,
    '14um_0.25r_FL1': 34.885,
    '14um_0.50r_FL1': 34.699,
    '8um_0.25r_FL1': 34.838,
```

```
    '8um_0.50r_FL1': 34.652,

    #yuhang redo
    # '20um_0.25r_FL1': 35.537,
    # '20um_0.50r_FL1': 40.740,

    # '14um_0.25r_FL1': 35.695,
    # '14um_0.50r_FL1': 35.555,
    # '14um_0r_FL1': 34.475,

    # '8um_0.25r_FL1': 35.508,
    # '8um_0.50r_FL1': 5.1,
    # '8um_r0':35.775,

    '20um_0.05r_FL2':35.606,
    # '20um_0.05r_FL2 redo':35.606,

    '80um_p_FL2': 35.477,
    # '80um_p_FL2_red': 33.986,
    # '80um_p_FL2_blue':  35.627,
    # '20um20um_non_interleaving_p_FL2': 35.588,
    # '20um20um_MMSS_non_interleaving_p_FL2': 35.588,
    # '20um20um_MMSS_non_interleaving_p_FL2_red': 35.606,
    # '20um20um_MMSS_non_interleaving_p_FL2_blue': 35.586,
    # '20um_0.05r_FL2':35.606
    # '20um_0.05r_FL2_red':35.627,
    # '20um_0.05r_FL2_blue':35.606,,
    '20um_p_FL2': 34.990,
    # '20um_p_FL2_red': 35.486,
    # '20um_p_FL2_blue': 35.436
}
corrected_z = {}
for key in z_data:
    if key in focus_values:
        # Subtract the scalar focus value from each element in the list
        corrected_z[key] = [focus_values[key] -z  for z in z_data[key]]
    else:
        print(f"Key {key} not found in focus_values")

plt.scatter(corrected_z['20um_0.25r_FL1'], contrast_data['20um_0.25r_FL1'],
plt.scatter(corrected_z['20um_0.50r_FL1'], contrast_data['20um_0.50r_FL1'],
plt.scatter(corrected_z['14um_0.25r_FL1'], contrast_data['14um_0.25r_FL1'],
plt.scatter(corrected_z['14um_0.50r_FL1'], contrast_data['14um_0.50r_FL1'],
plt.scatter(corrected_z['8um_0.25r_FL1'], contrast_data['8um_0.25r_FL1'], s=
# plt.scatter(corrected_z['8um_0.50r_FL1'], contrast_data['8um_0.50r_FL1'],
# plt.scatter(corrected_z['20um_0.05r_FL2'], contrast_data['20um_0.05r_FL2']
# plt.scatter(corrected_z['20um_0.05r_FL2 redo'], contrast_data['20um_0.05r_


# plt.scatter(corrected_z['20um20um_non_interleaving_p_FL2'], contrast_data[

# plt.scatter(corrected_z['80um_p_FL2'], contrast_data['80um_p_FL2'], s=15,
# plt.scatter(corrected_z['80um_p_FL2_red'], contrast_data['80um_p_FL2_red']
# plt.scatter(corrected_z['80um_p_FL2_blue'], contrast_data['80um_p_FL2_blue

# plt.scatter(corrected_z['20um20um_MMSS_non_interleaving_p_FL2'], contrast_
```

```
# plt.scatter(corrected_z['20um20um_MMSS_non_interleaving_p_FL2_red'], contr
# plt.scatter(corrected_z['20um20um_MMSS_non_interleaving_p_FL2_blue'], cont
# plt.scatter(corrected_z['20um_0.05r_FL2'], contrast_data['20um_0.05r_FL2']
# plt.scatter(corrected_z['20um_p_FL2'], contrast_data['20um_p_FL2'], s=15,
# plt.scatter(corrected_z['20um_p_FL2_red'], contrast_data['20um_p_FL2_red']
# plt.scatter(corrected_z['20um_p_FL2_blue'], contrast_data['20um_p_FL2_blue

# plt.scatter(corrected_z['20um_0.05r_FL2'], contrast_data['20um_0.05r_FL2']
# plt.scatter(corrected_z['20um_0.05r_FL2_red'], contrast_data['20um_0.05r_F
# plt.scatter(corrected_z['20um_0.05r_FL2_blue'], contrast_data['20um_0.05r_

# plt.scatter(corrected_z['8um_0.25r_FL1'], contrast_data['8um_0.25r_FL1'],
# plt.scatter(corrected_z['8um_0.25r_FL1 Yuhang redo'], contrast_data['8um_0


plt.ylim(0.15, 0.45)
#plt.xlim(0,5)
plt.title('Z-Contrast for Single Folders', fontsize=12)
plt.xlabel('Z in mm')
plt.ylabel('Contrast')
plt.legend(title='Sample Types', loc='upper left', bbox_to_anchor=(1, 1))
plt.show()
```



In [15]:
```
# Define the keys for which you want to find the value at x=1.5
keys = [
    '20um_0.25r_FL1', '20um_0.50r_FL1', '14um_0.25r_FL1','14um_0.50r_FL1',
    '8um_0.25r_FL1', '8um_0.50r_FL1', '20um_0.05r_FL2 redo','20um_p_FL2','8u
]
```

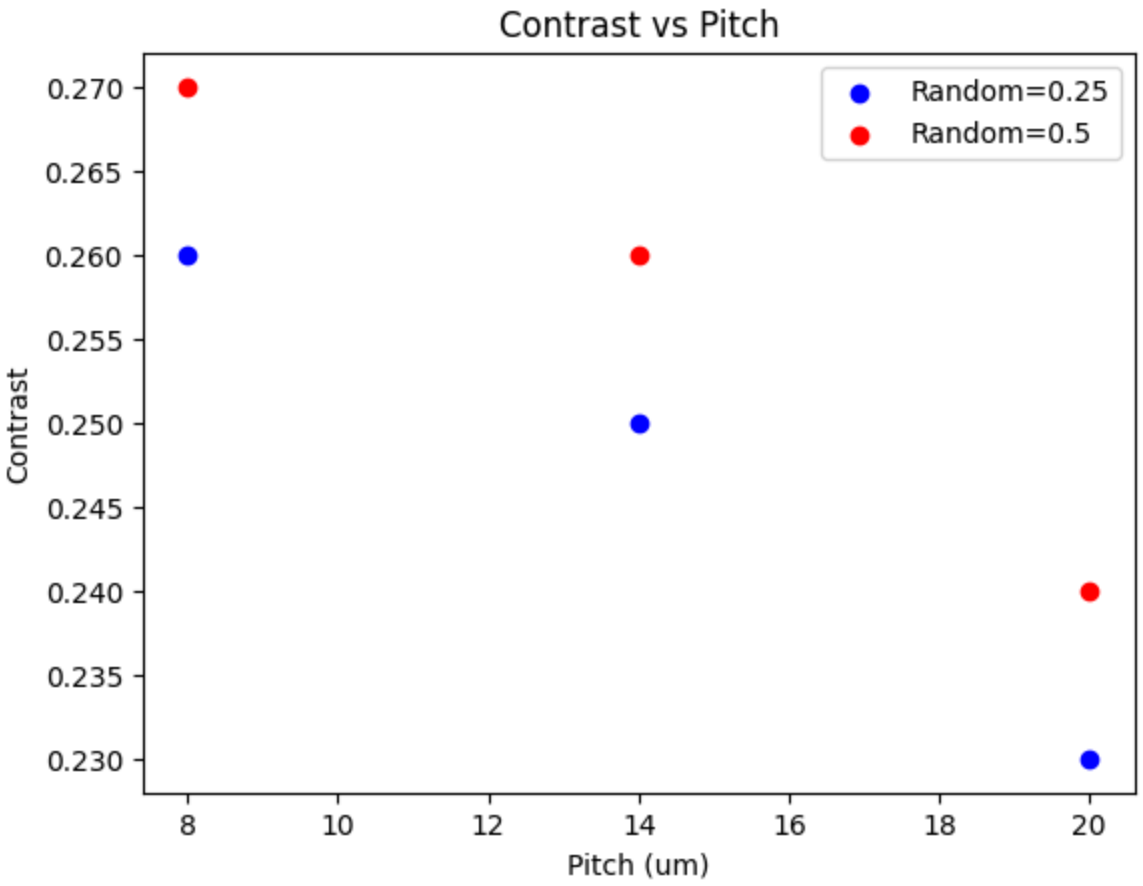
```
x=1.5
def find_value_at_x(data_x, data_y, x):
    if data_x.size > 0:  # Check if the array is not empty
        idx = (np.abs(data_x - x)).argmin()
        return data_y[idx]
    else:
        return None  # Return None if the data_x array is empty

for key in keys:
    if key in corrected_z and key in contrast_data:
        data_x = np.array(corrected_z[key])
        data_y = np.array(contrast_data[key])
        if data_x.size > 0 and data_y.size > 0:  # Ensure both arrays are no
            value_at_x = find_value_at_x(data_x, data_y,x)
            print(f"Value at x={x} for {key}: {value_at_x}")
        else:
            print(f"No data available for {key}.")
    else:
        print(f"Data for {key} not found.")
```

```
Value at x=1.5 for 20um_0.25r_FL1: 0.2317300268224123
Value at x=1.5 for 20um_0.50r_FL1: 0.23608728333942258
Value at x=1.5 for 14um_0.25r_FL1: 0.2484791822536564
Value at x=1.5 for 14um_0.50r_FL1: 0.2551239905612006
Value at x=1.5 for 8um_0.25r_FL1: 0.2623315205456727
Data for 8um_0.50r_FL1 not found.
Data for 20um_0.05r_FL2 redo not found.
Value at x=1.5 for 20um_p_FL2: 0.2250239404333581
Value at x=1.5 for 8um_r0: 0.2161391413792149
Value at x=1.5 for 14um_0r_FL1: 0.7846858356466142
Value at x=1.5 for 80um_p_FL2: 0.2164208517922098
```

In [17]:

```
import matplotlib.pyplot as plt

# Define the data for the first curve (random=0.25)
pitch_1 = [8, 14, 20]
contrast_1 = [0.26, 0.25, 0.23]

# Define the data for the second curve (random=0.5)
pitch_2 = [8, 14, 20]
contrast_2 = [0.27, 0.26, 0.24]

# Create a figure and axis object
fig, ax = plt.subplots()

# Plot the first curve with a specific color
ax.scatter(pitch_1, contrast_1, label='Random=0.25', color='blue')

# Plot the second curve with a different color
ax.scatter(pitch_2, contrast_2, label='Random=0.5', color='red')

# Set the title and labels
ax.set_title('Contrast vs Pitch')
ax.set_xlabel('Pitch (um)')
ax.set_ylabel('Contrast')
```

```
# Show the legend
plt.legend()

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



In [19]:
```
import pandas as pd

# Assuming z_data and contrast_data are structured as follows:
# z_data = {'20um_0.25r': [...], '20um_0.50r': [...], ...}
# contrast_data = {'20um_0.25r': [...], '20um_0.50r': [...], ...}

# Create a DataFrame from the dictionaries
data = []
for key in z_data:
    for z, contrast in zip(corrected_z[key], contrast_data[key]):
        data.append({'Sample Type': key, 'Z in mm': z, 'Contrast': contrast}

df = pd.DataFrame(data)

# Save the DataFrame to a CSV file
df.to_csv('/Users/qingjunwang/Documents/image files/speckle 2 study/z_contra

print("Data saved to 'z_contrast_data.csv'.")
```

```
Data saved to 'z_contrast_data.csv'.
```

PIC design z=4mm, contrast-shaking amplitude

select samples of z=4mm

In [20]:
```
# downselect z
import matplotlib.pyplot as plt
# Define folders for each dataset
folders_random = {
    '20um_0.25r_FL1': '/Users/qingjunwang/Documents/image files/speckle 2 st
    '20um_0.50r_FL1': '/Users/qingjunwang/Documents/image files/speckle 2 st
    '14um_0.25r_FL1': '/Users/qingjunwang/Documents/image files/speckle 2 st
    '14um_0.50r_FL1': '/Users/qingjunwang/Documents/image files/speckle 2 st
    '8um_0.25r_FL1': '/Users/qingjunwang/Documents/image files/speckle 2 stu
    '8um_0.50r_FL1': '/Users/qingjunwang/Documents/image files/speckle 2 stu
    # '20um_0.05r_FL2': '/Users/qingjunwang/Documents/image files/speckle 2
}
folders_period={
    '80um_p_FL2': '/Users/qingjunwang/Documents/image files/speckle 2 study/
    '20um20um_non_interleaving_p_FL2': '/Users/qingjunwang/Documents/image f
    '20um20um_MMSS_non_interleaving_p_FL2': '/Users/qingjunwang/Documents/im
    '20um_p_FL2': '/Users/qingjunwang/Documents/image files/speckle 2 study/
}
# Dictionary to store closest images and their Z-values
closest_images_random = {}
closest_z_values_random = {}
closest_images_period = {}
closest_z_values_period = {}
targe_z=4
# Find closest images for each dataset
for label, folder in folders_random.items():
    closest_images_random[label], closest_z_values_random[label] = main.finc
    print(f"Closest image for {label}, filepath: {closest_images_random[labe
for label, folder in folders_period.items():
    closest_images_period[label], closest_z_values_period[label] = main.finc
    print(f"Closest image for {label}, filepath: {closest_images_period[labe
```

```
Closest image for 20um_0.25r_FL1, filepath: /Users/qingjunwang/Documents/ima
ge files/speckle 2 study/EPIC-FL1/NAP_20um/EPIC-FL1_D0C0_NAP-20um-0.25R_sing
le_image_folder/EPIC-FL1_D0C0_NAP-20um-P20-0.25r_Green__X0.000_Y0.000_Z30.92
7_Exp100000.png  with Z-value: 30.927
Closest image for 20um_0.50r_FL1, filepath: /Users/qingjunwang/Documents/ima
ge files/speckle 2 study/EPIC-FL1/NAP_20um/EPIC-FL1_D0C0_NAP-20um-0.50R_sing
le_image_folder/EPIC-FL1_D0C0_NAP-20um-P20-0.50r_Green__X0.000_Y0.000_Z30.74
1_Exp100000.png  with Z-value: 30.741
Closest image for 14um_0.25r_FL1, filepath: /Users/qingjunwang/Documents/ima
ge files/speckle 2 study/EPIC-FL1/NAP_14um/EPIC-FL1_D0C0_NAP-14um-0.25R_sing
le_image_folder/EPIC-FL1_D0C0_NAP-14um-P20-0.25r_Green__X0.000_Y0.000_Z30.88
5_Exp100000.png  with Z-value: 30.885
Closest image for 14um_0.50r_FL1, filepath: /Users/qingjunwang/Documents/ima
ge files/speckle 2 study/EPIC-FL1/NAP_14um/EPIC-FL1_D0C0_NAP-14um-0.50R_sing
le_image_folder/EPIC-FL1_D0C0_NAP-14um-P20-0.50r_Green__X0.000_Y0.000_Z30.69
9_Exp30784.png  with Z-value: 30.699
Closest image for 8um_0.25r_FL1, filepath: /Users/qingjunwang/Documents/imag
e files/speckle 2 study/EPIC-FL1/NAP_8um/EPIC-FL1_D0C0_NAP-8um-0.25R_single_
image_folder/EPIC-FL1_D0C0_NAP-8um-P20-0.25r_Green__X0.000_Y0.000_Z30.838_Ex
p32812.png  with Z-value: 30.838
Closest image for 8um_0.50r_FL1, filepath: /Users/qingjunwang/Documents/imag
e files/speckle 2 study/EPIC-FL1/NAP_8um/EPIC-FL1_D0C0_NAP-8um-0.50R_single_
image_folder/EPIC-FL1_D0C0_NAP-8um-P20-0.50r_Green__X0.000_Y0.000_Z30.652_Ex
p120000.png  with Z-value: 30.652
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[20], line 30
     28     print(f"Closest image for {label}, filepath: {closest_images_ran
dom[label]}  with Z-value: {closest_z_values_random[label]}")
     29 for label, folder in folders_period.items():
---> 30     closest_images_period[label], closest_z_values_period[label] = m
ain.find_closest_z_image(folder,focus_values[label]-targe_z)
     31     print(f"Closest image for {label}, filepath: {closest_images_per
iod[label]}  with Z-value: {closest_z_values_period[label]}")

KeyError: '80um_p_FL2'
```

In [2]:
```
# print(closest_images_random)
closest_images_random={'20um_0.25r_FL1': '/Users/qingjunwang/Documents/image
```

In [61]:
```
# closest_images_random
import cv2
# List to store the loaded images
images = {}
# Loop through the image paths and load each image
for label, image_path in closest_images_random.items():
    img = cv2.imread(image_path)
    if img is not None:
        images[label]=img
    else:
        print(f"Failed to load image at {image_path}")
print(f"Loaded {len(images)} images.")
print(images)
```

```
Loaded 7 images.
{'20um_0.25r_FL1': array([[[20, 43, 10],
        [20, 49, 10],
        [19, 46,  9],
        ...,
        [19, 37, 10],
        [18, 38,  9],
        [17, 35,  9]],

       [[20, 38,  9],
        [20, 40,  9],
        [19, 41,  9],
        ...,
        [19, 41,  9],
        [18, 39,  9],
        [17, 40,  9]],

       [[16, 33,  7],
        [16, 33,  8],
        [16, 35,  9],
        ...,
        [19, 44,  9],
        [19, 44,  8],
        [20, 45,  8]],

       ...,

       [[21, 48,  9],
        [21, 48, 10],
        [24, 44, 10],
        ...,
        [15, 33,  8],
        [16, 35,  8],
        [16, 36,  8]],

       [[23, 53,  8],
        [23, 53,  9],
        [23, 50, 10],
        ...,
        [17, 32,  8],
        [17, 35,  7],
        [17, 33,  7]],

       [[24, 58,  8],
        [24, 54,  9],
        [23, 50, 10],
        ...,
        [18, 36,  8],
        [18, 40,  7],
        [18, 37,  7]]], dtype=uint8), '20um_0.50r_FL1': array([[[20, 49, 1
0],
        [20, 54,  9],
        [19, 46,  9],
        ...,
        [31, 61, 13],
        [28, 60, 11],
```

```
        [25, 57, 11]],

       [[20, 44, 11],
        [20, 46, 10],
        [19, 43,  9],
        ...,
        [31, 63, 12],
        [28, 60, 11],
        [25, 61, 11]],

       [[23, 45, 11],
        [23, 41, 10],
        [22, 44, 10],
        ...,
        [29, 67, 11],
        [27, 65, 11],
        [26, 68, 11]],

       ...,

       [[32, 66, 11],
        [32, 66, 13],
        [31, 70, 14],
        ...,
        [29, 69, 13],
        [28, 69, 13],
        [28, 65, 13]],

       [[33, 65, 12],
        [33, 64, 13],
        [32, 69, 14],
        ...,
        [31, 75, 13],
        [30, 72, 13],
        [29, 67, 13]],

       [[34, 66, 12],
        [34, 66, 13],
        [33, 69, 14],
        ...,
        [34, 76, 13],
        [31, 77, 13],
        [29, 72, 13]]], dtype=uint8), '14um_0.25r_FL1': array([[[ 38,  89,
16],
        [ 38,  91,  15],
        [ 40,  97,  15],
        ...,
        [ 42,  91,  16],
        [ 37,  82,  15],
        [ 32,  68,  15]],

       [[ 38,  86,  16],
        [ 38,  93,  16],
        [ 40, 101,  15],
        ...,
        [ 42,  94,  16],
```

```
         [ 37,  84,  16],
         [ 32,  76,  16]],

        [[ 39,  88,  16],
         [ 39,  93,  16],
         [ 41,  96,  16],
         ...,
         [ 40,  96,  17],
         [ 35,  81,  17],
         [ 29,  70,  17]],

        ...,

        [[ 39, 126,  19],
         [ 39, 108,  17],
         [ 35,  91,  15],
         ...,
         [ 37,  90,  15],
         [ 36,  98,  15],
         [ 35,  89,  15]],

        [[ 37, 105,  18],
         [ 37, 100,  17],
         [ 36,  89,  15],
         ...,
         [ 39,  95,  17],
         [ 38,  93,  16],
         [ 38,  91,  16]],

        [[ 34,  93,  18],
         [ 34,  95,  17],
         [ 36,  85,  15],
         ...,
         [ 41,  92,  17],
         [ 40,  89,  16],
         [ 40,  90,  16]]], dtype=uint8), '14um_0.50r_FL1': array([[[25, 50,
10],
         [25, 51, 10],
         [23, 52, 11],
         ...,
         [29, 71, 13],
         [28, 61, 14],
         [27, 62, 14]],

        [[25, 49, 10],
         [25, 51, 10],
         [23, 52, 10],
         ...,
         [29, 62, 12],
         [28, 56, 11],
         [27, 56, 11]],

        [[23, 51, 10],
         [23, 51, 10],
         [22, 52,  9],
         ...,
```

```
 [23, 53, 10],
 [22, 50,  9],
 [21, 50,  9]],

...,

[[23, 58, 11],
 [23, 56, 10],
 [24, 57,  9],
 ...,
 [25, 48, 10],
 [23, 47, 10],
 [20, 45, 10]],

[[22, 50, 10],
 [22, 47,  9],
 [23, 48,  8],
 ...,
 [23, 43,  9],
 [21, 42,  9],
 [19, 40,  9]],

[[21, 50, 10],
 [21, 47,  9],
 [21, 45,  8],
 ...,
 [21, 42,  9],
 [19, 40,  9],
 [17, 40,  9]]], dtype=uint8), '8um_0.25r_FL1': array([[[39, 94, 15],
 [39, 95, 15],
 [37, 93, 15],
 ...,
 [29, 76, 14],
 [30, 78, 15],
 [31, 83, 15]],

[[39, 92, 15],
 [39, 92, 15],
 [37, 94, 15],
 ...,
 [29, 74, 13],
 [30, 77, 14],
 [31, 80, 14]],

[[37, 83, 14],
 [37, 85, 15],
 [38, 85, 15],
 ...,
 [27, 70, 12],
 [29, 73, 13],
 [30, 82, 13]],

...,

[[28, 59, 12],
 [28, 58, 12],
```

```
 [24, 54, 11],
 ...,
 [25, 57, 12],
 [24, 48, 12],
 [23, 47, 12]],

[[29, 62, 12],
 [29, 59, 12],
 [27, 57, 11],
 ...,
 [27, 53, 11],
 [24, 46, 10],
 [21, 36, 10]],

[[30, 72, 12],
 [30, 63, 12],
 [29, 63, 11],
 ...,
 [29, 54, 11],
 [24, 49, 10],
 [19, 42, 10]]], dtype=uint8), '8um_0.50r_FL1': array([[[20, 45, 11],
 [20, 43, 10],
 [22, 46, 10],
 ...,
 [24, 54, 11],
 [24, 54, 11],
 [24, 52, 11]],

[[20, 47, 11],
 [20, 47, 10],
 [22, 51,  9],
 ...,
 [24, 54, 11],
 [24, 55, 11],
 [24, 53, 11]],

[[18, 46, 10],
 [18, 49, 10],
 [20, 48,  9],
 ...,
 [23, 52, 11],
 [24, 52, 11],
 [25, 52, 11]],

...,

[[23, 50, 11],
 [23, 53, 11],
 [22, 53, 11],
 ...,
 [19, 47,  9],
 [20, 51,  9],
 [22, 50,  9]],

[[25, 55, 12],
 [25, 56, 12],
```

```
                [27, 61, 12],
                ...,
                [22, 47,  9],
                [21, 48,  9],
                [20, 46,  9]],

               [[26, 58, 12],
                [26, 61, 12],
                [31, 72, 12],
                ...,
                [24, 48,  9],
                [21, 47,  9],
                [18, 47,  9]]], dtype=uint8), '20um_0.05r_FL2': array([[[19, 48,
9],
                [19, 45,  9],
                [21, 50,  9],
                ...,
                [29, 64, 11],
                [26, 54, 11],
                [23, 49, 11]],

               [[19, 52,  9],
                [19, 49,  9],
                [21, 52,  9],
                ...,
                [29, 62, 11],
                [26, 51, 11],
                [23, 49, 11]],

               [[19, 48,  9],
                [19, 47,  8],
                [21, 51,  8],
                ...,
                [28, 60, 11],
                [25, 52, 10],
                [22, 45, 10]],

               ...,

               [[19, 40, 10],
                [19, 39,  9],
                [19, 34,  8],
                ...,
                [19, 41,  8],
                [19, 33,  7],
                [19, 35,  7]],

               [[25, 49, 12],
                [25, 46, 10],
                [23, 45,  8],
                ...,
                [18, 38,  8],
                [20, 36,  8],
                [21, 34,  8]],

               [[31, 65, 12],
```

```
        [31, 52, 10],
        [27, 51,  8],
        ...,
        [17, 39,  8],
        [20, 37,  8],
        [23, 36,  8]]], dtype=uint8)}
```

shaking simulation

In [62]:
```
#image dimension x 21*20um=420um in dimension counted the pic pitch number
```

In [77]:
```
images['20um_0.25r_FL1'].shape
```

Out[77]:
```
(2056, 2464, 3)
```

In [50]:
```
2056/2464*420
```

Out[50]:
```
350.45454545454544
```

In [ ]:
```
2464
```

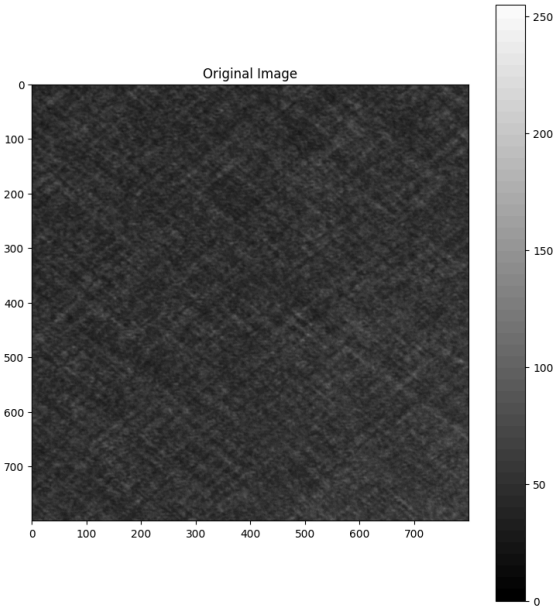
In [3]:
```
700/2464*420
```

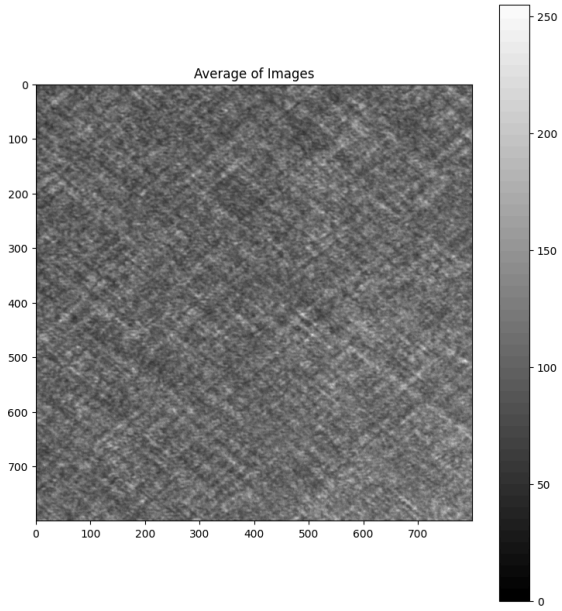
Out[3]:
```
119.31818181818183
```

In [26]:
```
import cv2
import matplotlib.pyplot as plt
import numpy as np
from scipy.ndimage import shift

vmin = 0
vmax = 255

amplitude_arr = [0,20,40,60,80,100]
print(f'amplitude array={amplitude_arr}')
image_pixel_per_um = 2464 / 420
img_save_path = '/Users/qingjunwang/Documents/image files/speckle 2 study/cr
contrast={}

def shaking_scan_amplitude(dic_images_key_path, amplitude_arr, image_pixel_p
    radius_arr = np.ceil([amp * image_pixel_per_um for amp in amplitude_arr]
    sampling_arr = [np.ceil(amp/30*100) for amp in amplitude_arr] #1.8um per
    for pic_design, path in dic_images_key_path.items():
        img = cv2.imread(path, cv2.IMREAD_UNCHANGED)
        if img is None:
            print(f"Failed to load image at {path}")
            continue
        if len(img.shape) == 2:
            img1 = img.astype(np.float32)
        elif len(img.shape) == 3:
            color_map = {'red': 2, 'green': 1, 'blue': 0}
            if color in color_map:
                img1 = img[:, :, color_map[color]].astype(np.float32)
            else:
```

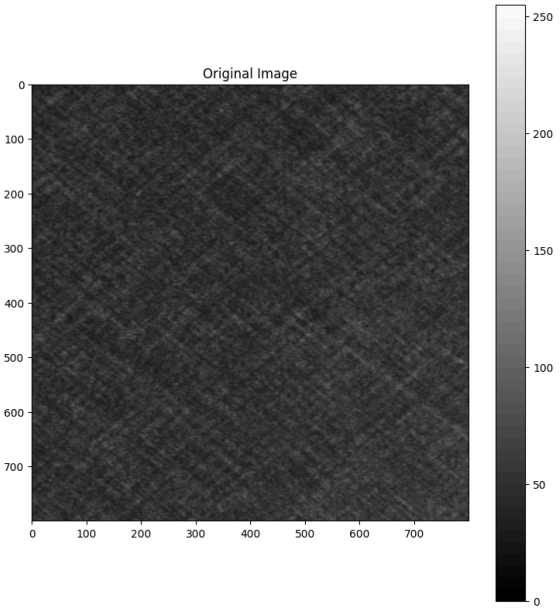
```
                print(f"Invalid color specified: {color}")
                continue
        else:
            print(f"Unsupported image format at {path}")
            continue
        img1=main.square_crop_img_upperleft(img1, 800)
        for j in range(len(amplitude_arr)):
            if amplitude_arr[j]==0:
                avg_img=img1
            else:
                angle = np.linspace(0, 2 * np.pi, int(sampling_arr[j]))
                sum_img = np.zeros_like(img1)
                for a in angle:
                    x_shift = radius_arr[j] * np.cos(a)
                    y_shift = radius_arr[j] * np.sin(a)
                    shifted_img = shift(img1, [y_shift, x_shift], mode="wrap
                    sum_img += shifted_img
                avg_img = sum_img / len(angle)
            max_value = avg_img.max()
            scale = 255 / max_value
            scaled_img = avg_img * scale
            contrast[(pic_design,amplitude_arr[j])]=scaled_img.std()/scaled_

            fig, axs = plt.subplots(nrows=1, ncols=2, figsize=(20, 10))
            im1 = axs[0].imshow(img1, cmap="gray", vmin=vmin, vmax=vmax)
            axs[0].set_title("Original Image")
            fig.colorbar(im1, ax=axs[0])
            axs[0].grid(False)
            im2 = axs[1].imshow(scaled_img, cmap="gray", vmin=vmin, vmax=vma
            axs[1].set_title("Average of Images")
            fig.colorbar(im2, ax=axs[1])
            axs[1].grid(False)
            plt.show()
            filename=f'{img_save_path}{pic_design}_{int(sampling_arr[j])}poi
            directory = os.path.dirname(filename)
            if not os.path.exists(directory):
                os.makedirs(directory)

            success =cv2.imwrite(f'{img_save_path}{pic_design}_{int(samplinc
            if success:
                print(f"Image successfully saved ")
            else:
                print(f"Failed to save image to ")
contrast=shaking_scan_amplitude(closest_images_random, amplitude_arr, image_
```

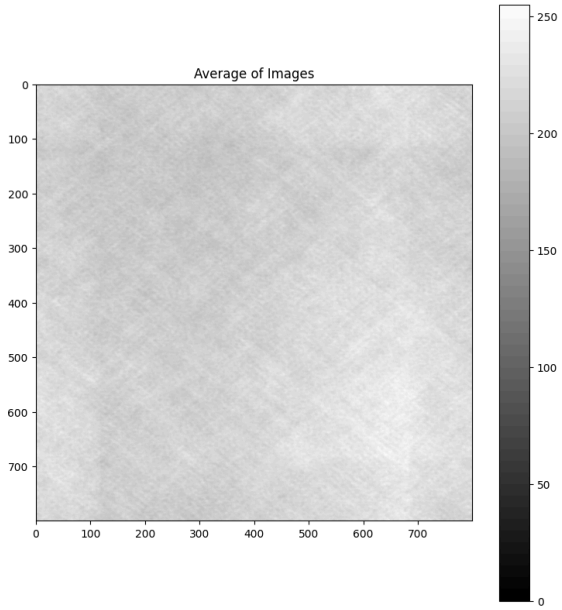
```
amplitude array=[0, 20, 40, 60, 80, 100]
```



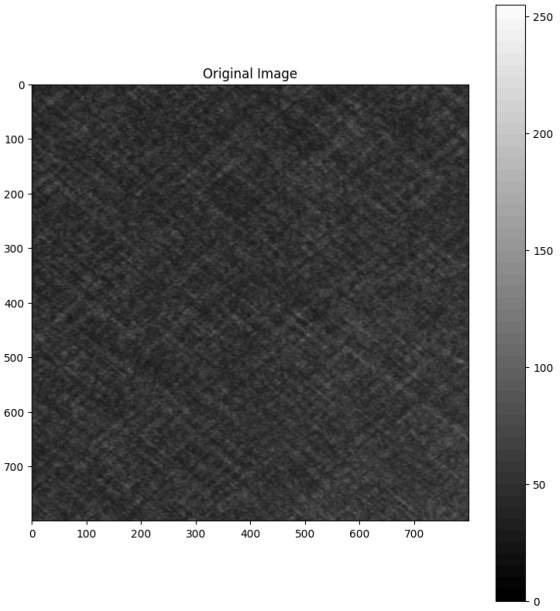


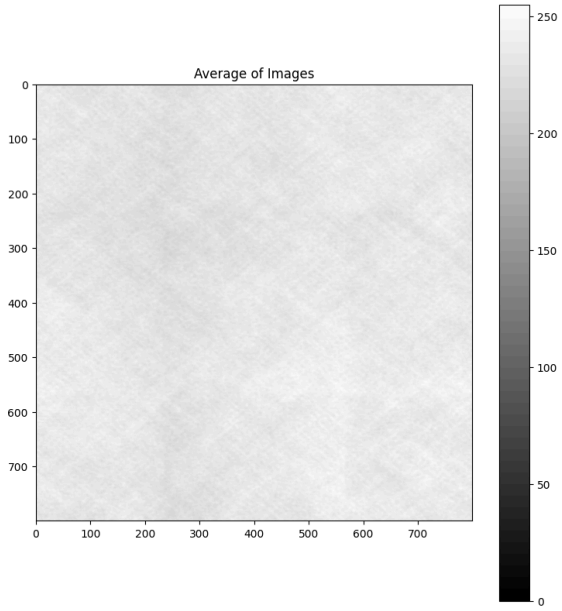

```
Image successfully saved
```



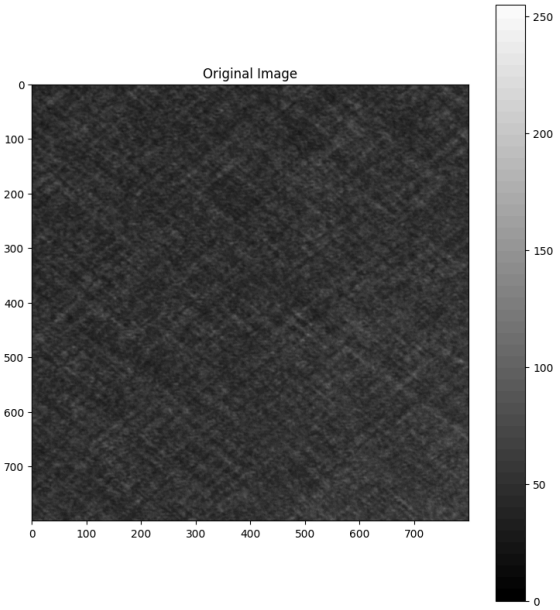


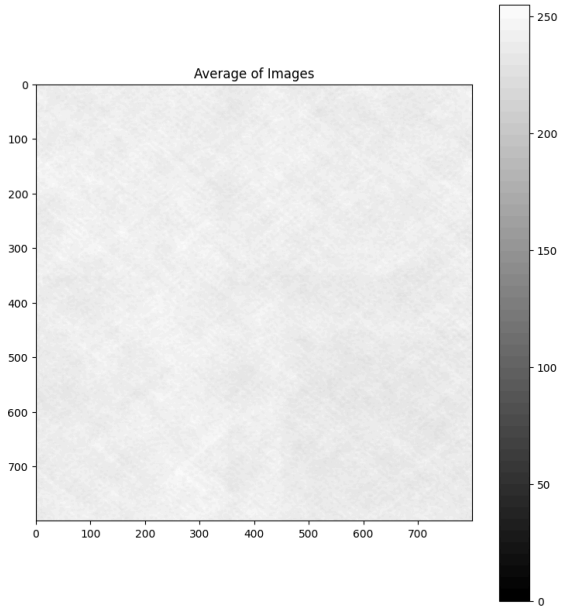

```
Image successfully saved
```



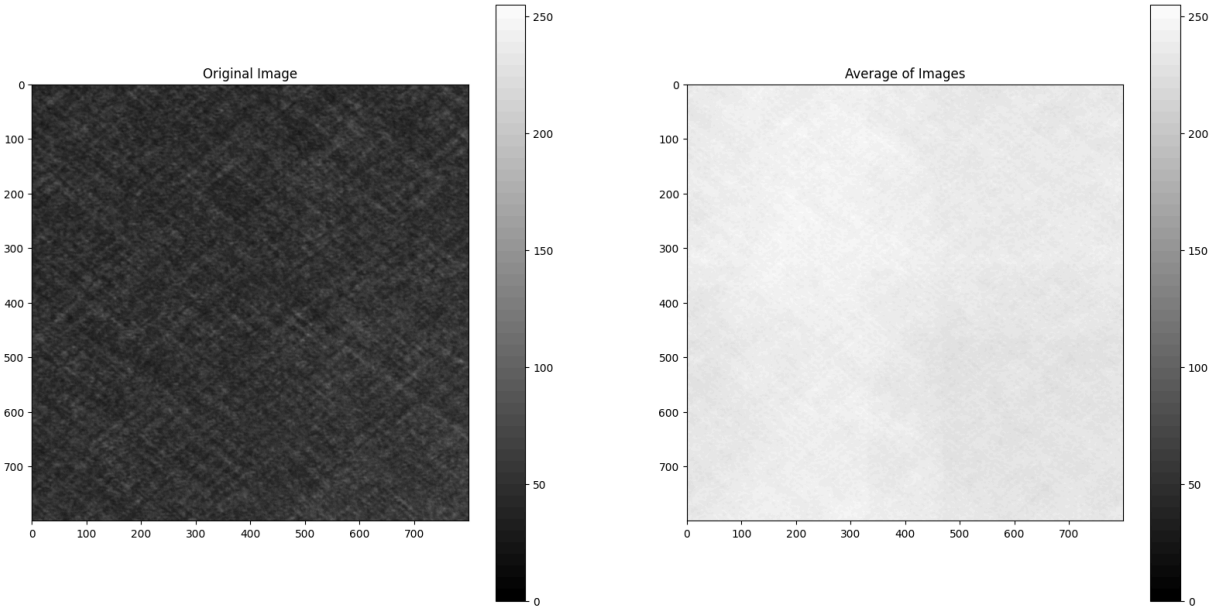



```
Image successfully saved
```

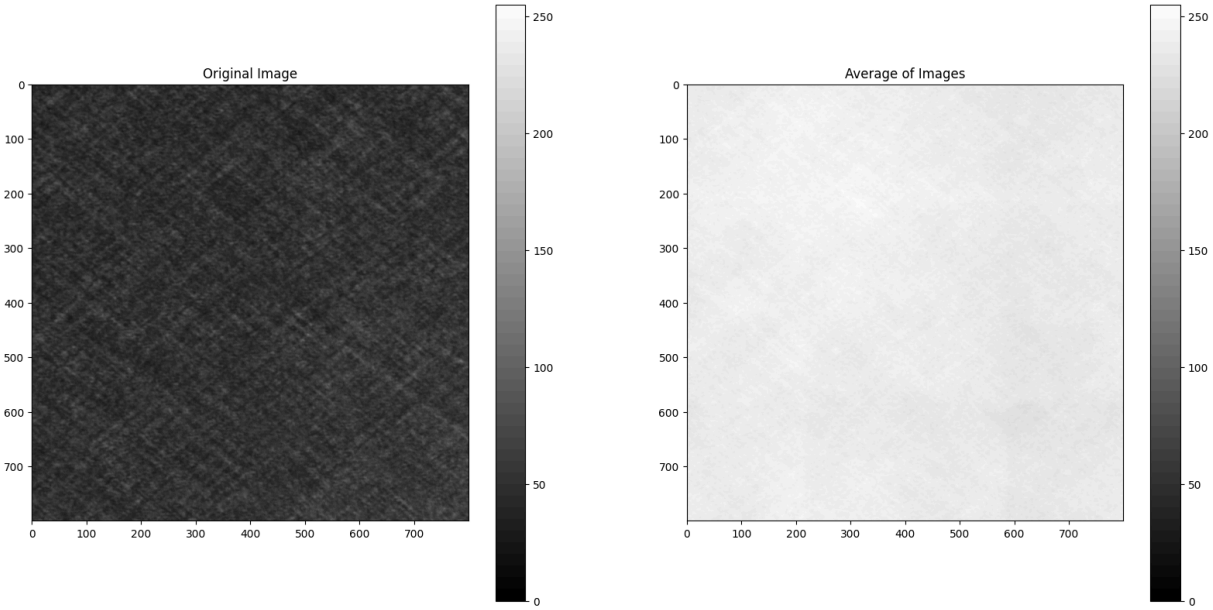




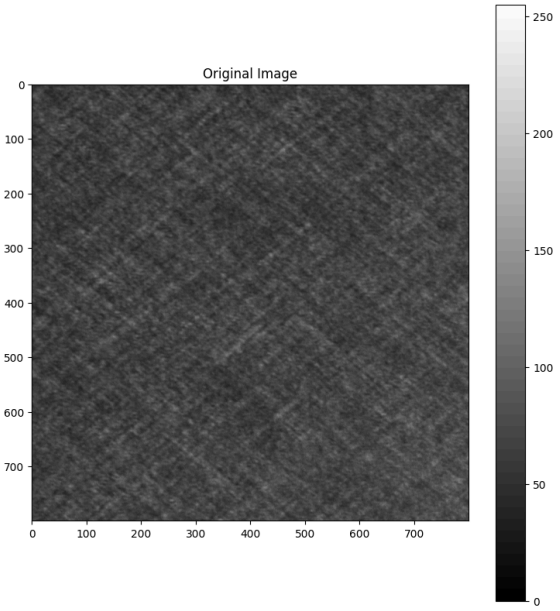

```
Image successfully saved
```

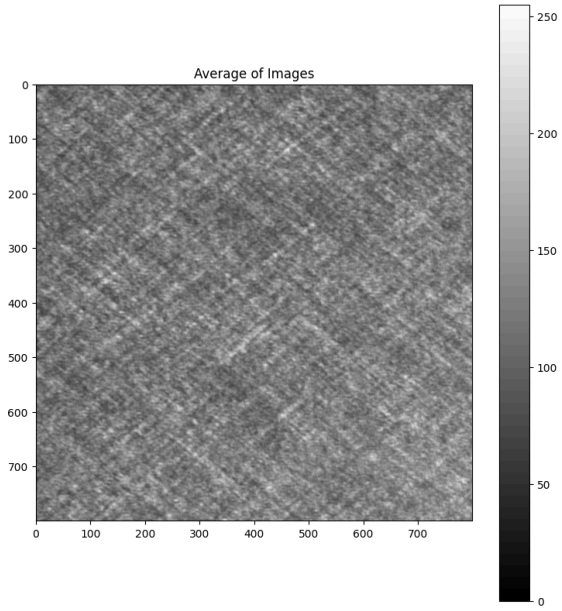



```
Image successfully saved
```

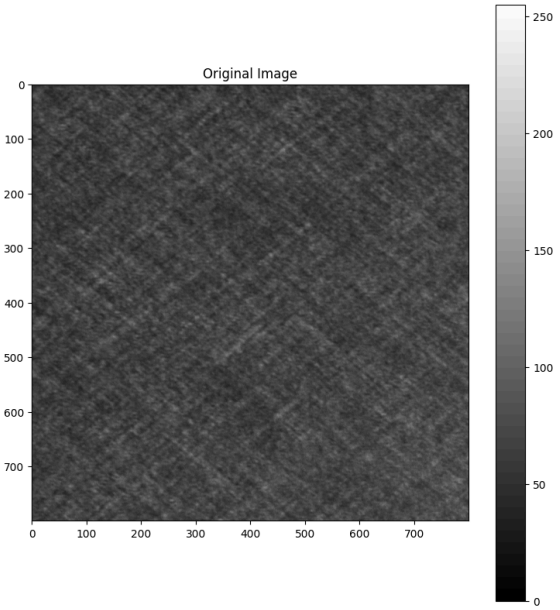


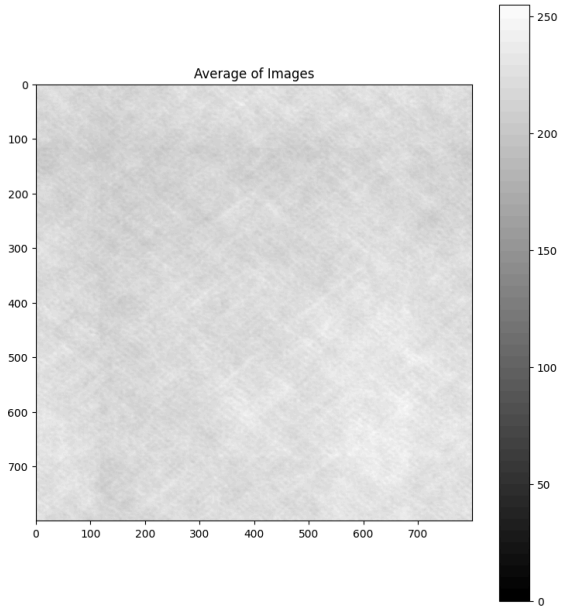

```
Image successfully saved
```



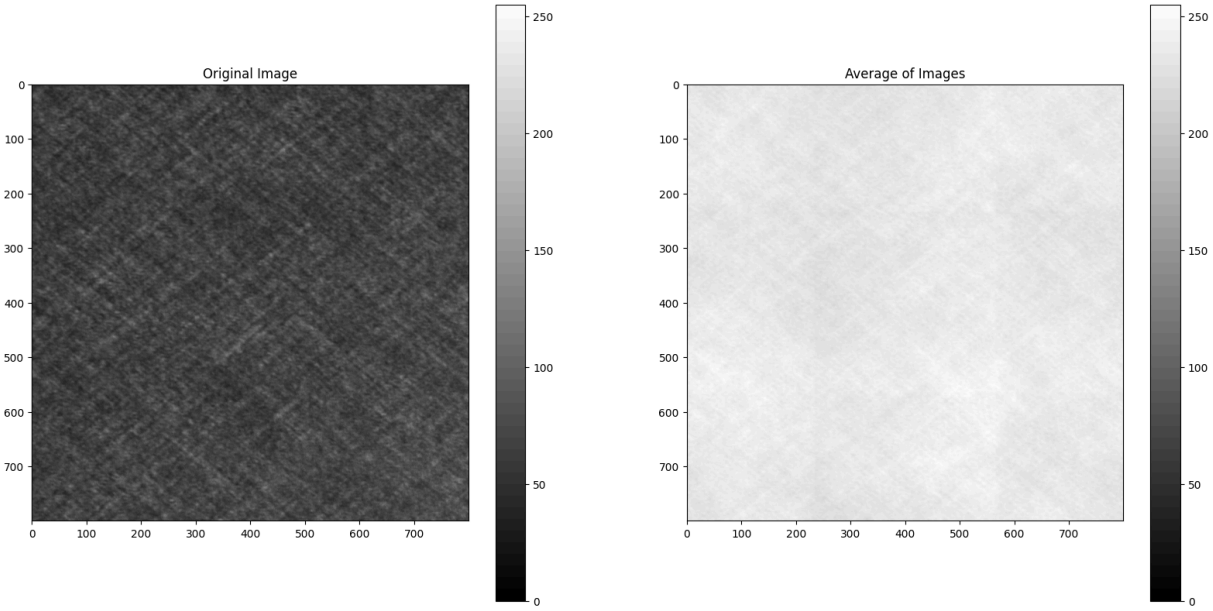



```
Image successfully saved
```

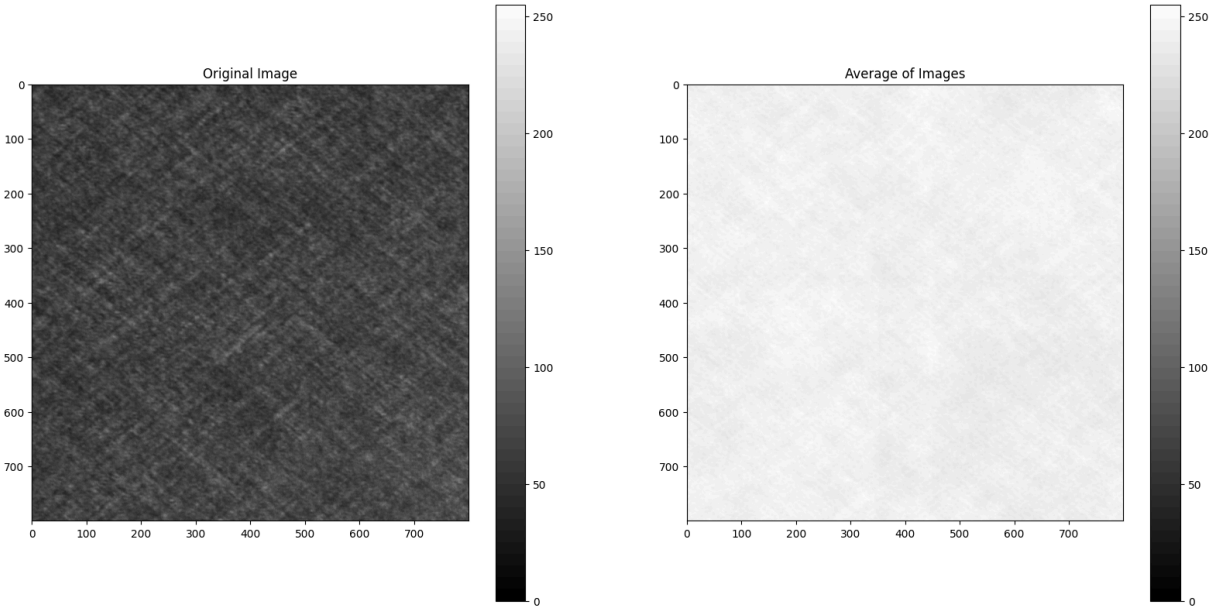




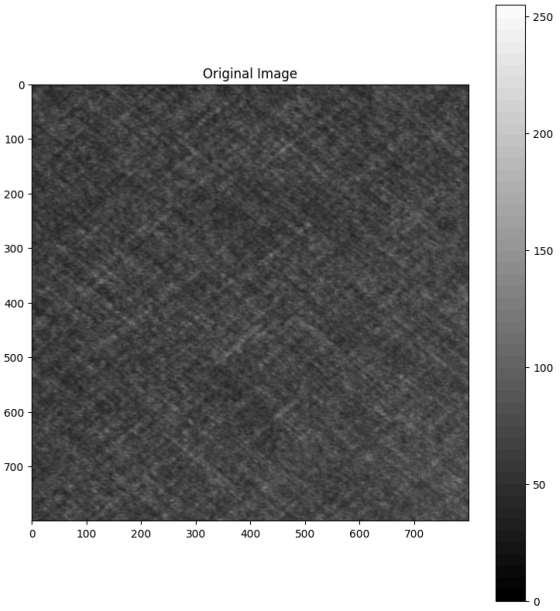

```
Image successfully saved
```

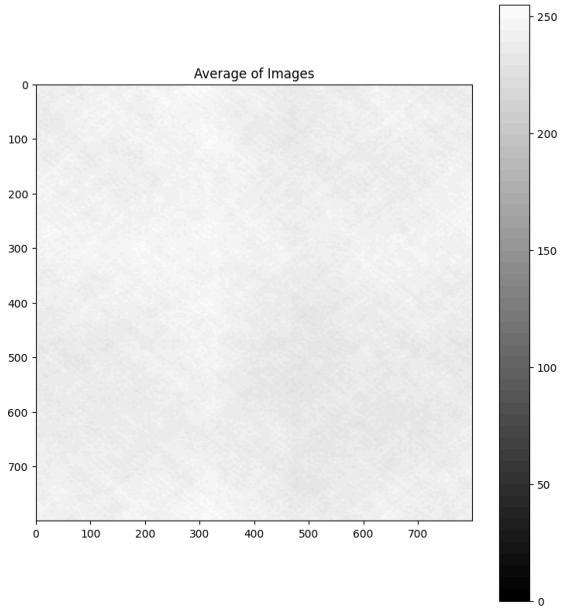



```
Image successfully saved
```

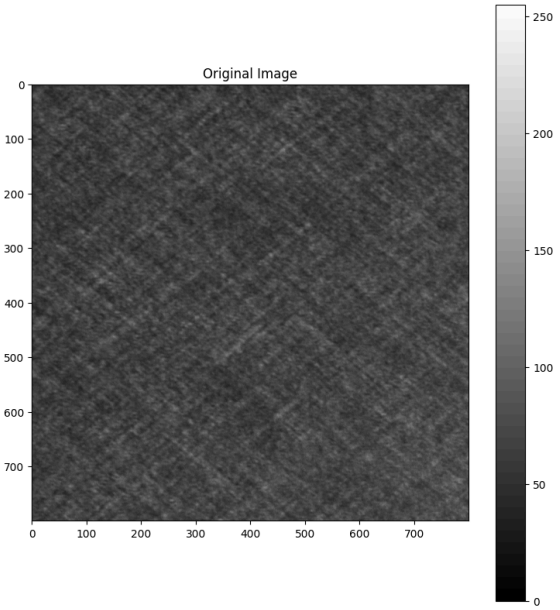


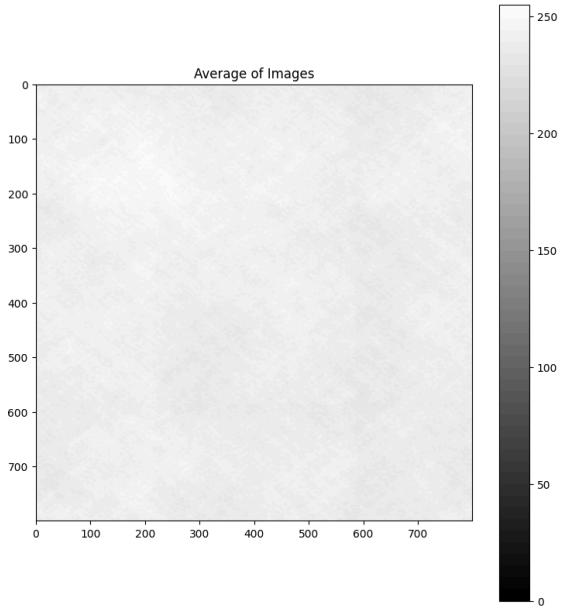

```
Image successfully saved
```



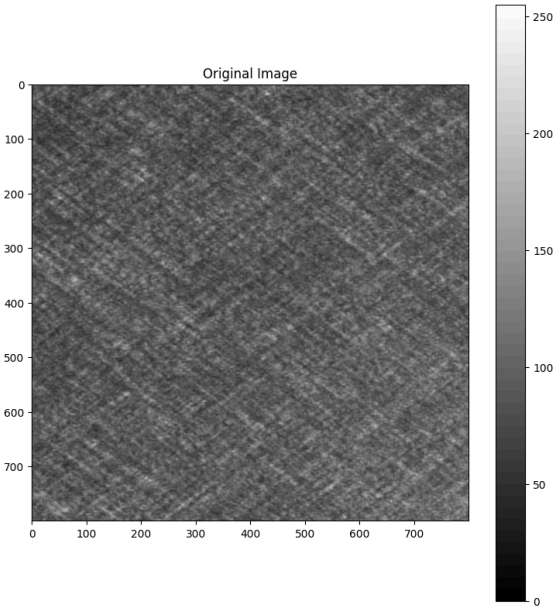


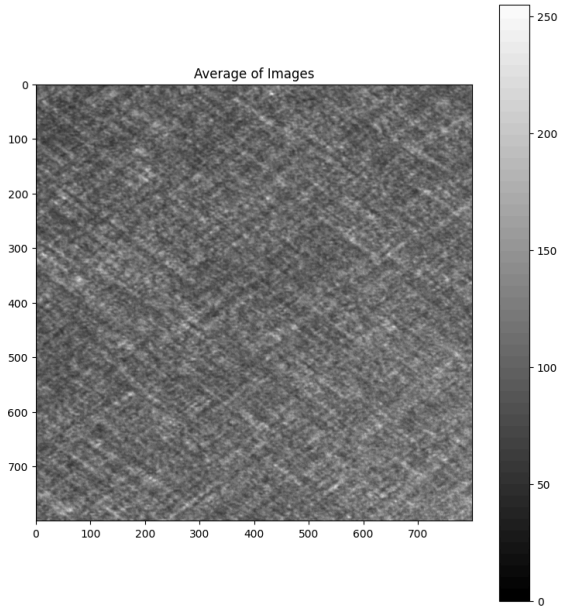

```
Image successfully saved
```



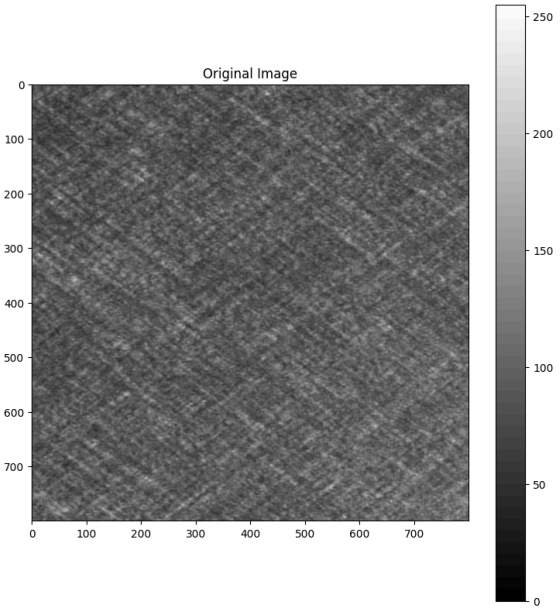


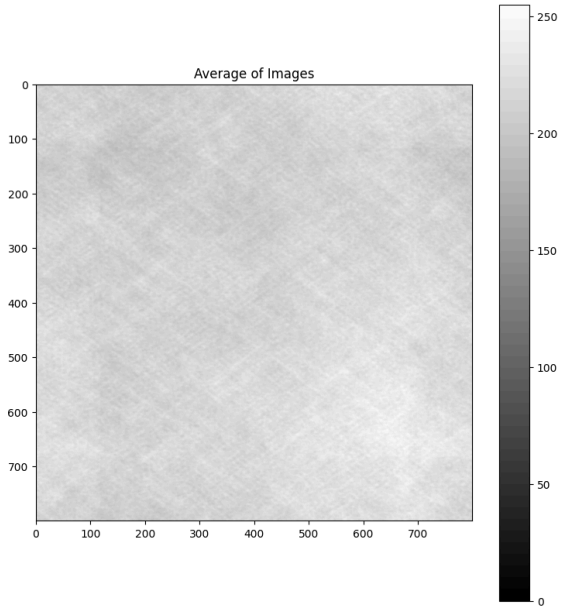

```
Image successfully saved
```



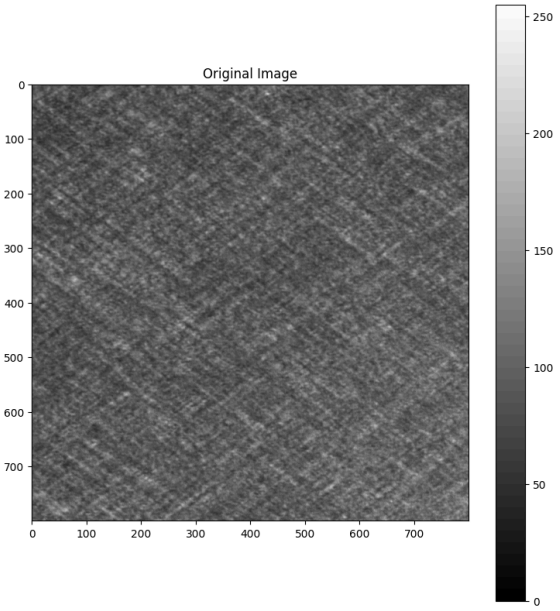


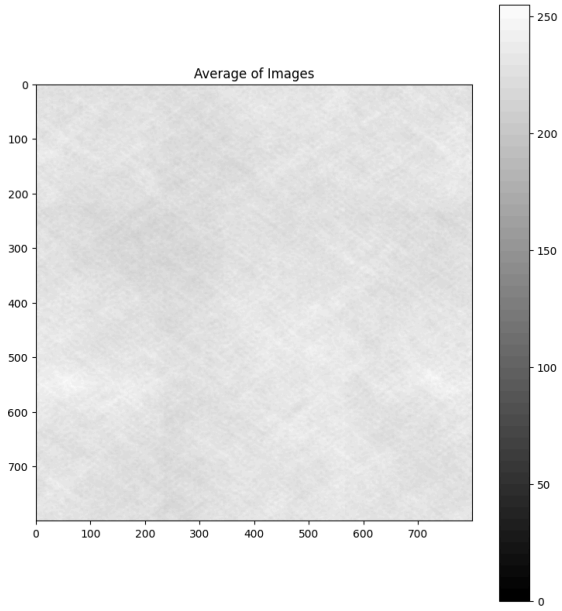

```
Image successfully saved
```



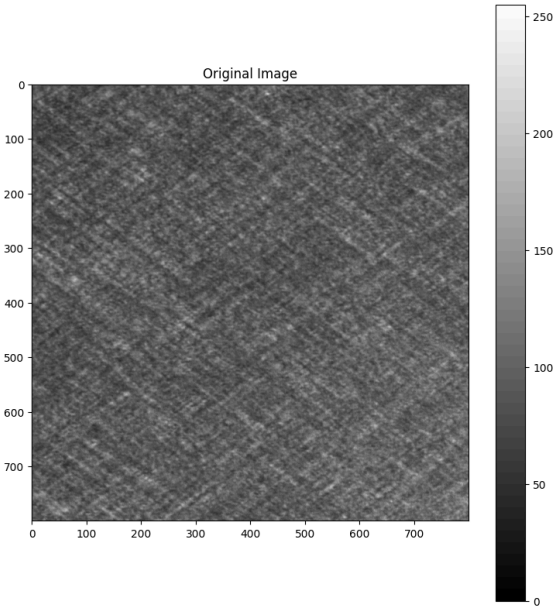


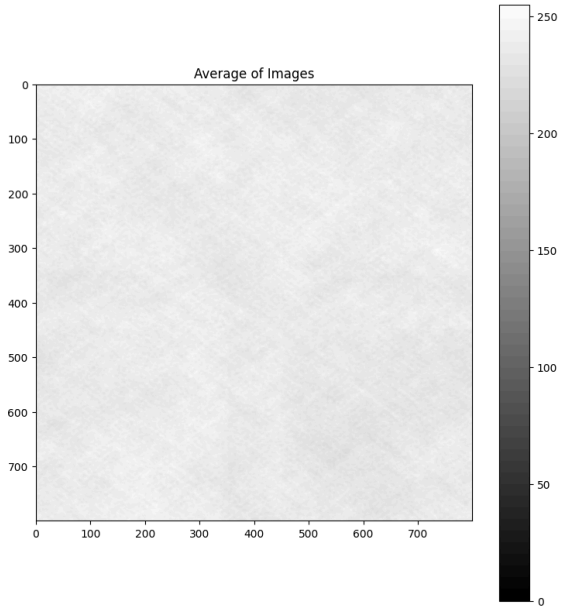

```
Image successfully saved
```



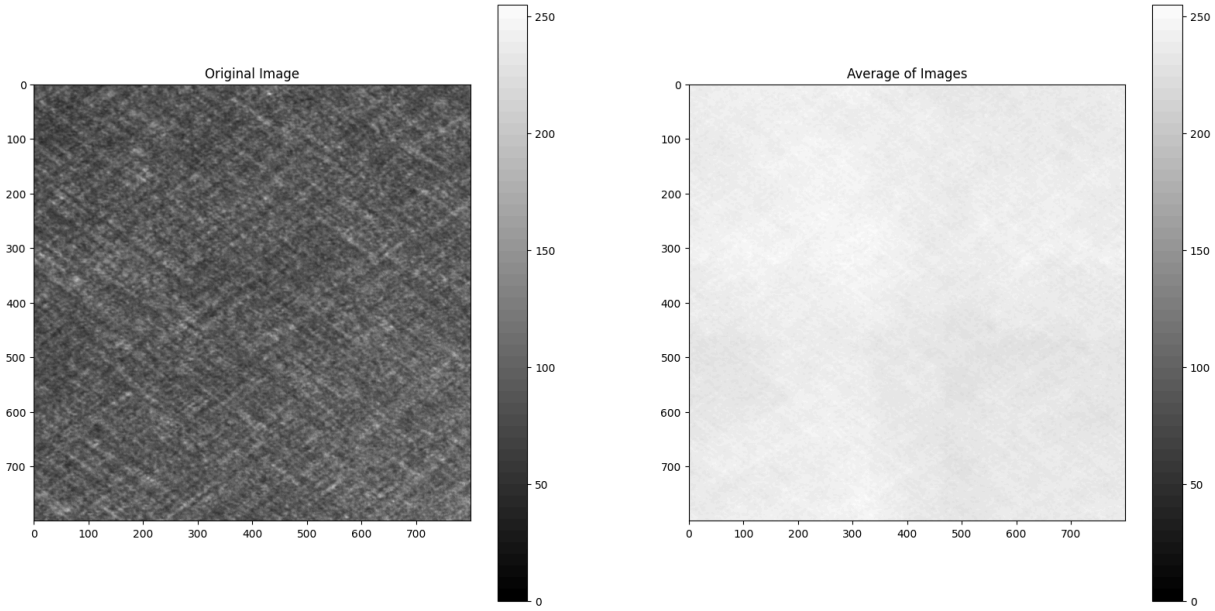



```
Image successfully saved
```

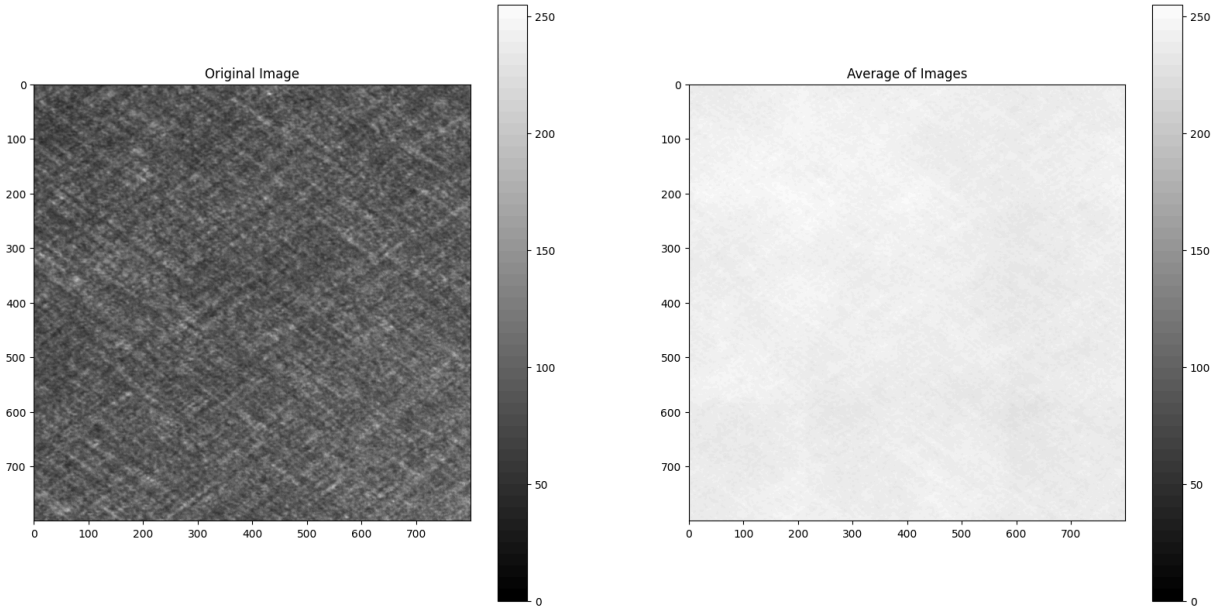




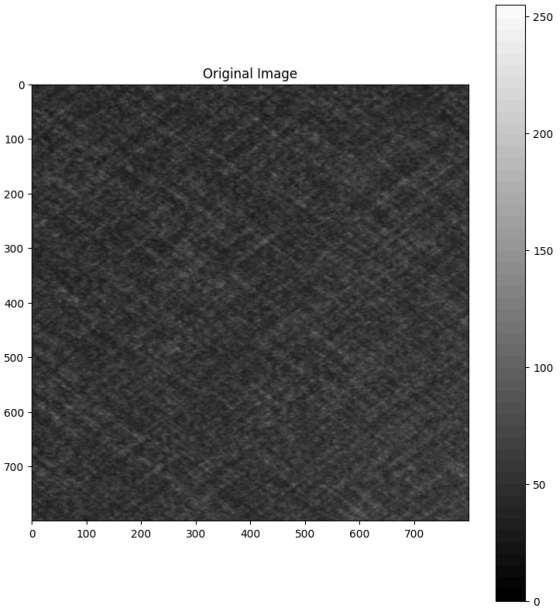

```
Image successfully saved
```



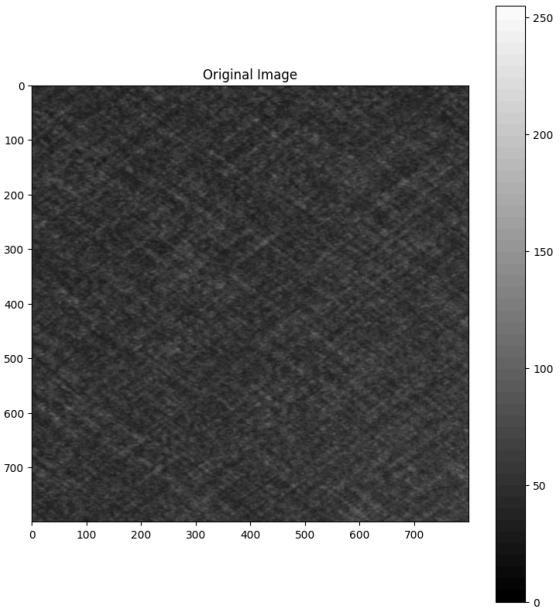

```
Image successfully saved
```



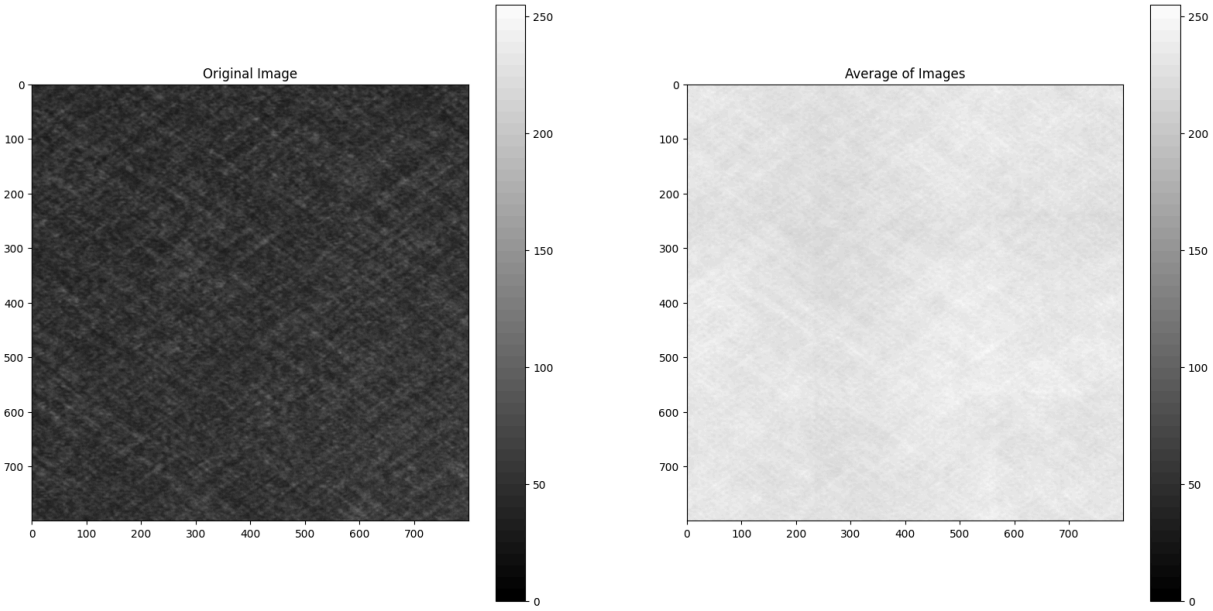

```
Image successfully saved
```



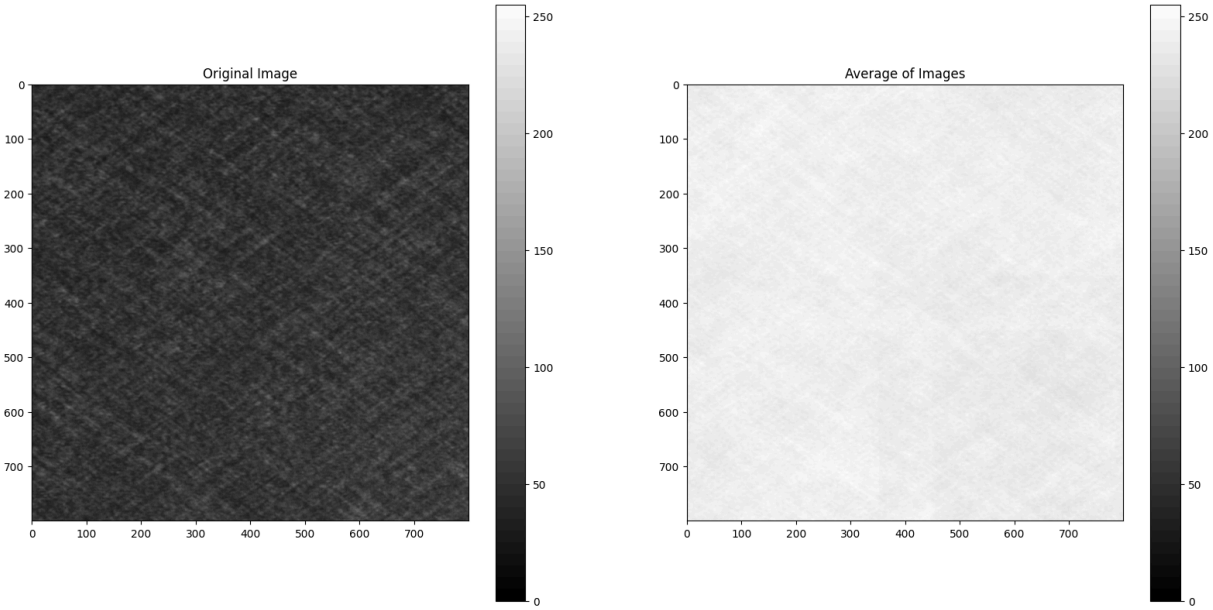

```
Image successfully saved
```



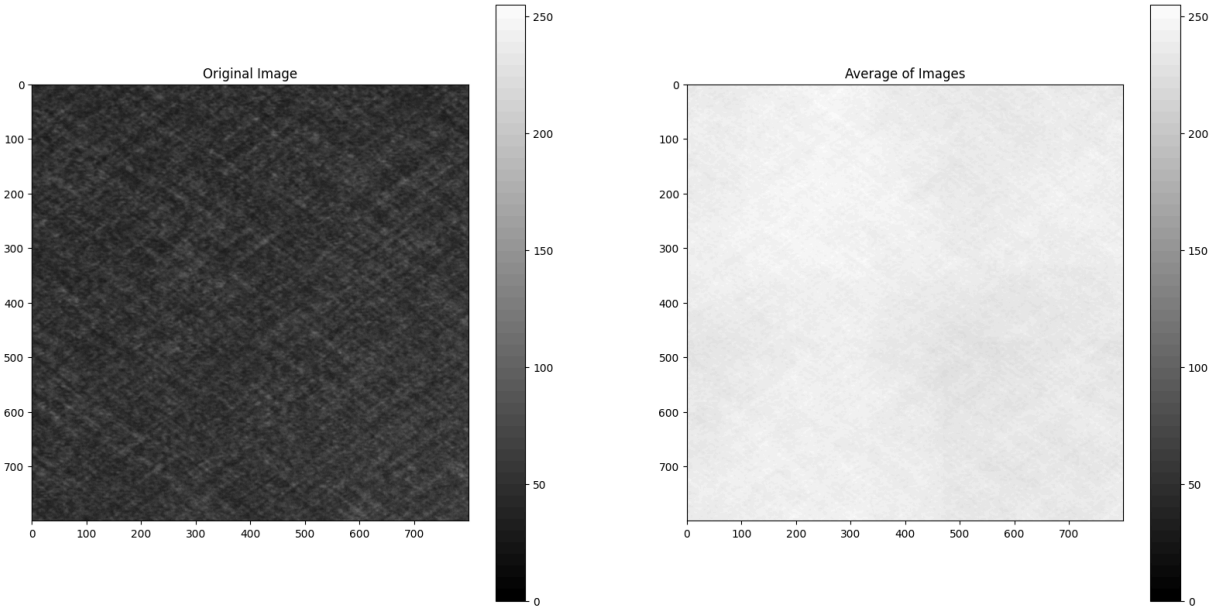

```
Image successfully saved
```



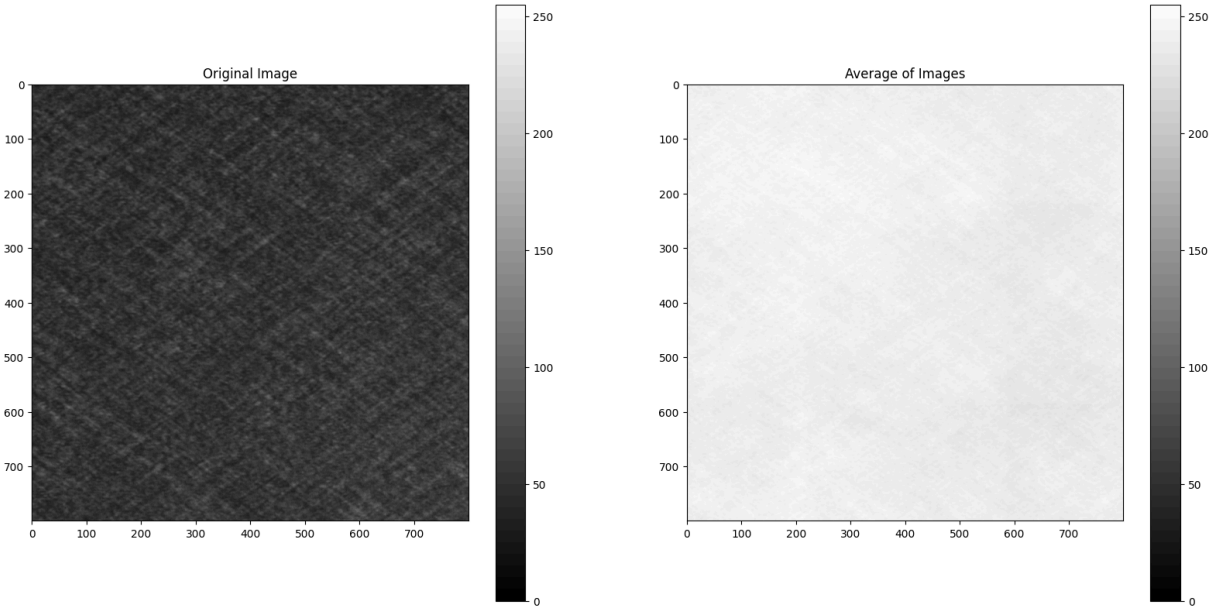

```
Image successfully saved
```



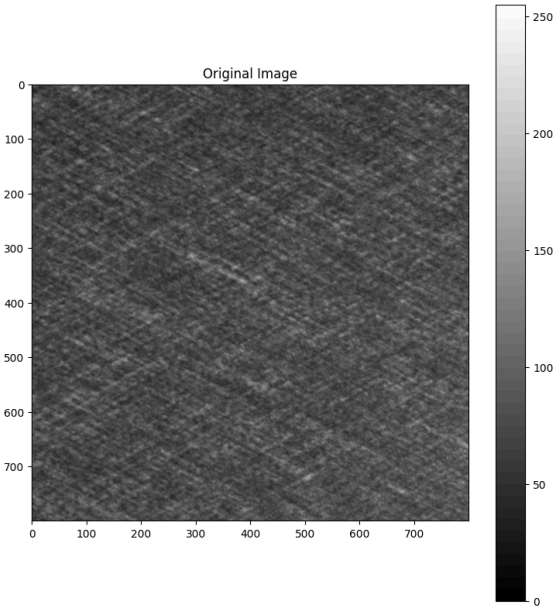

```
Image successfully saved
```



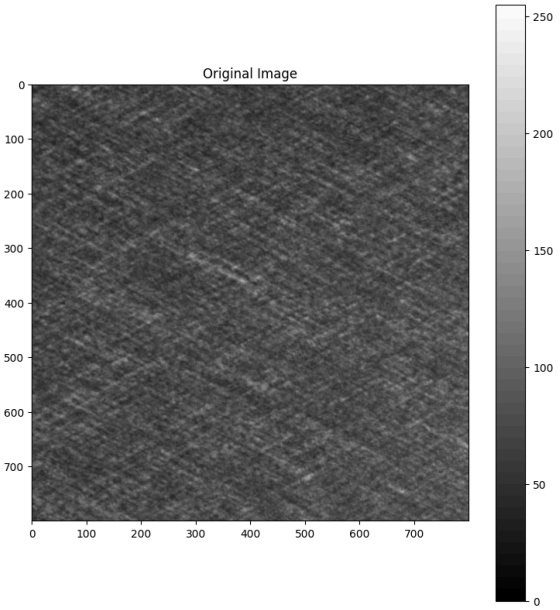

```
Image successfully saved
```



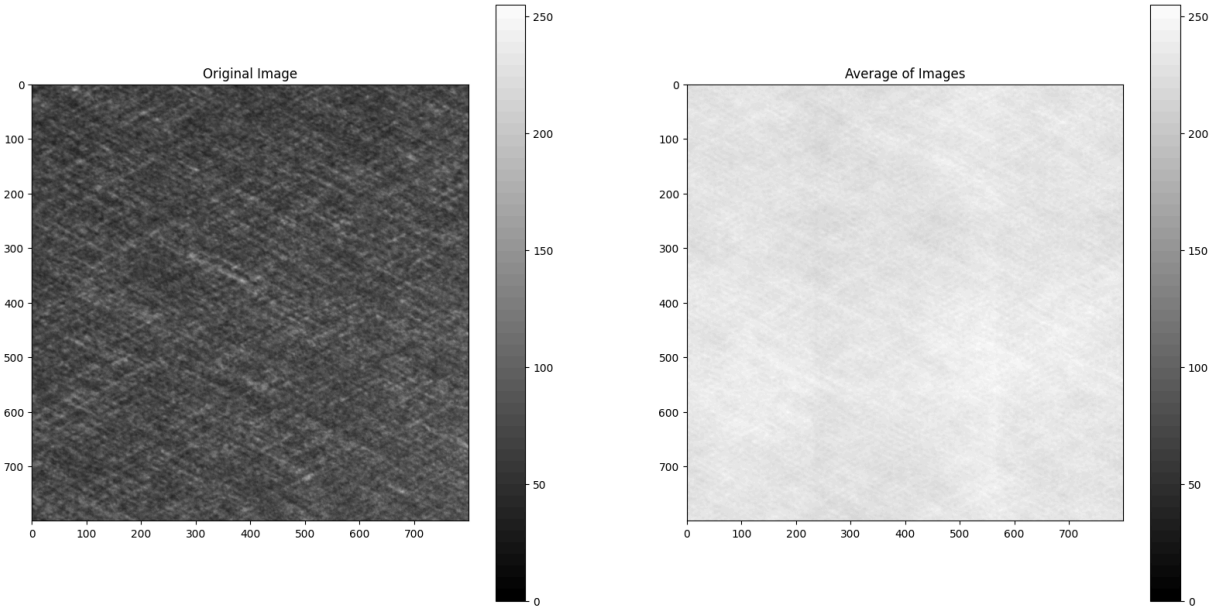

```
Image successfully saved
```



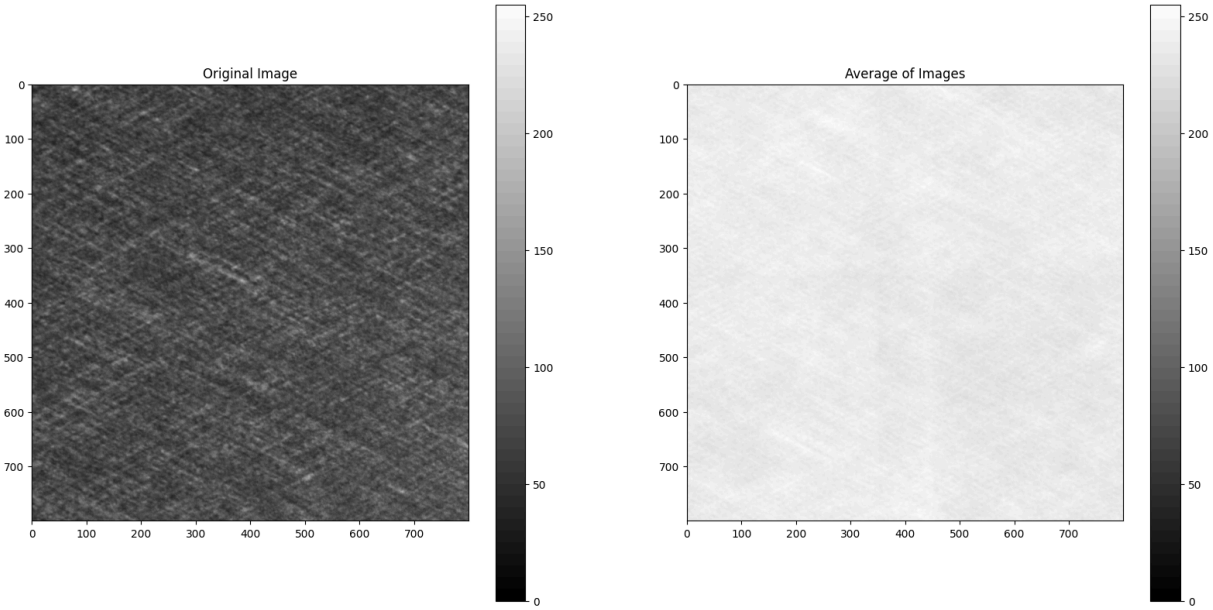

```
Image successfully saved
```



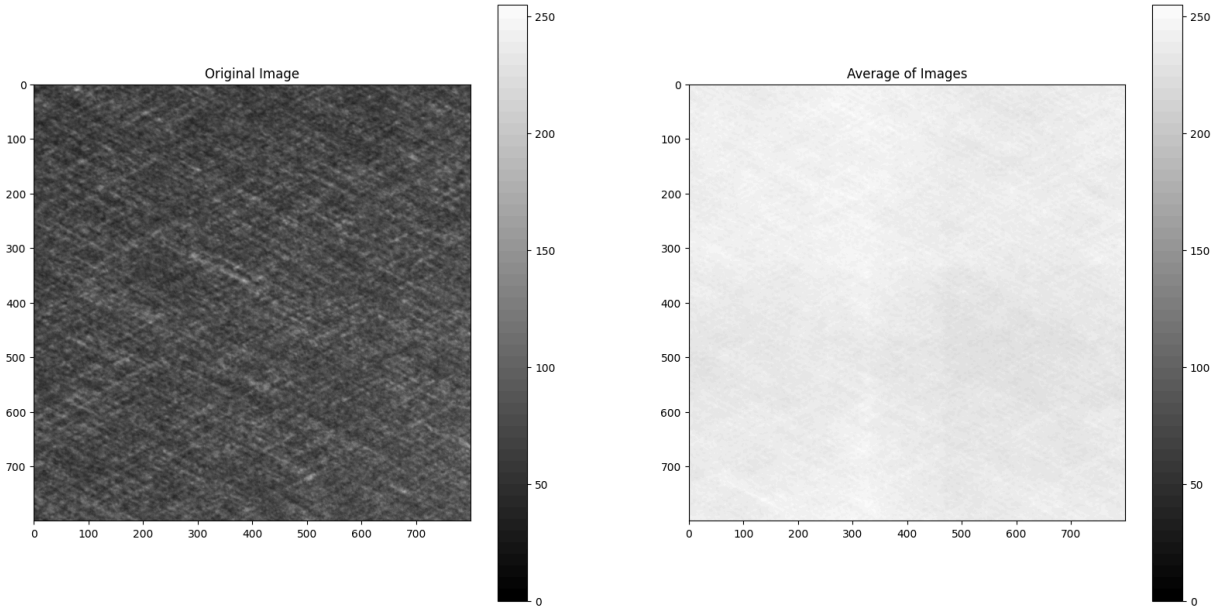

```
Image successfully saved
```



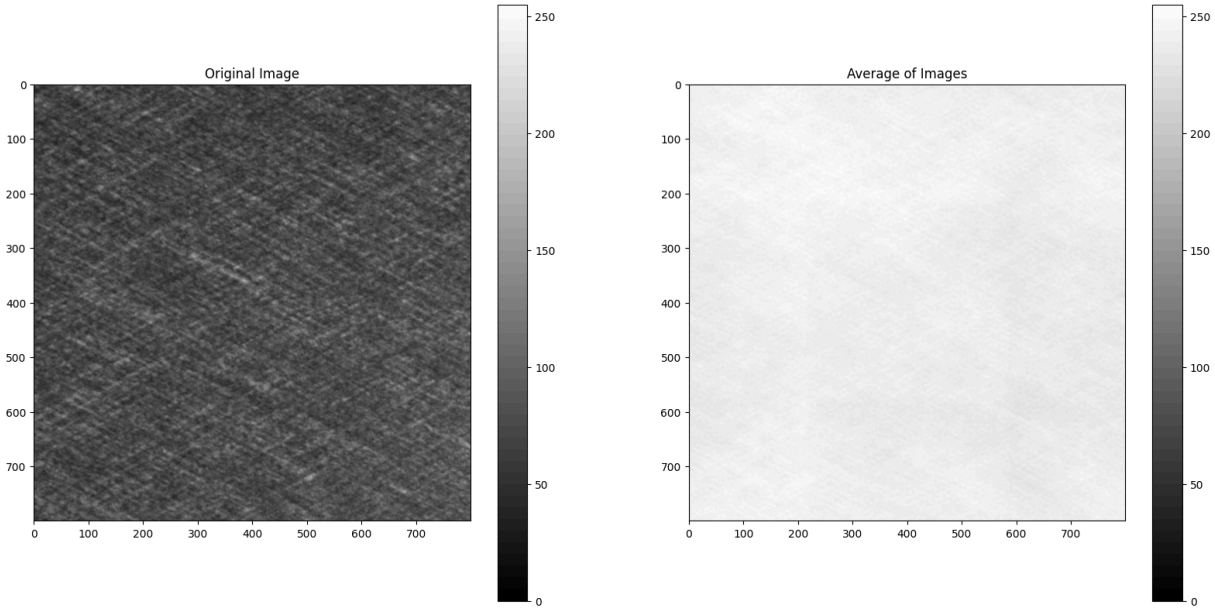

```
Image successfully saved
```



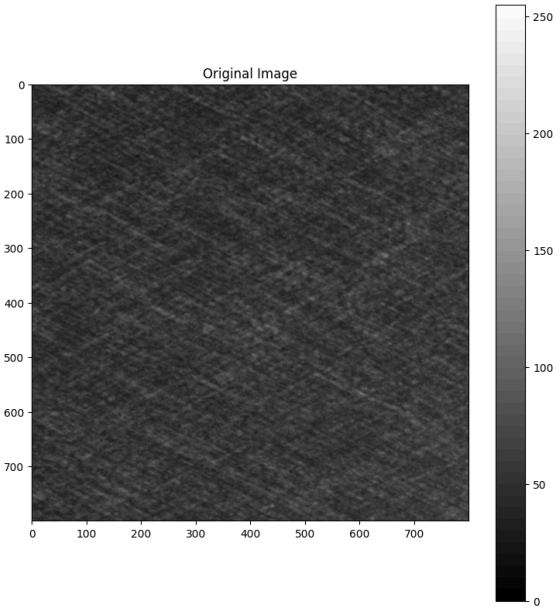

```
Image successfully saved
```

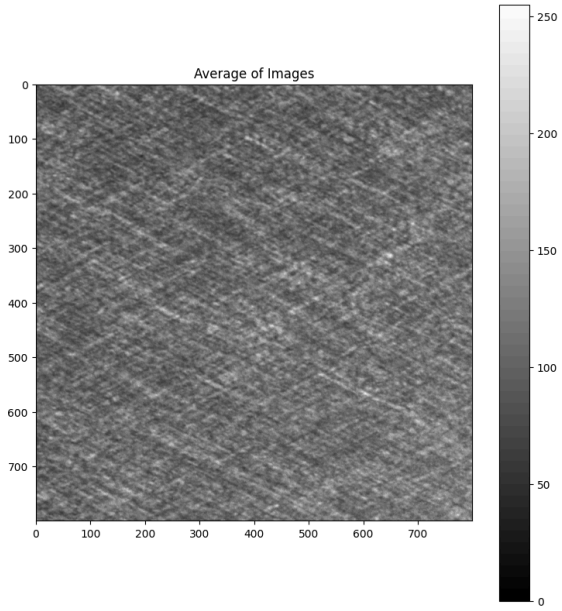



```
Image successfully saved
```

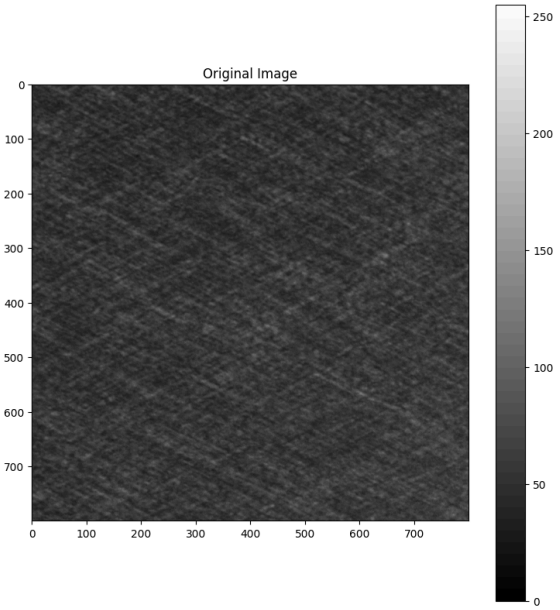


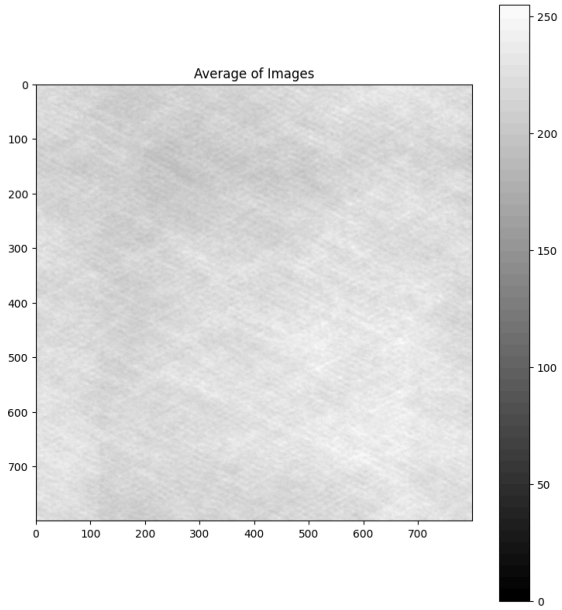

```
Image successfully saved
```



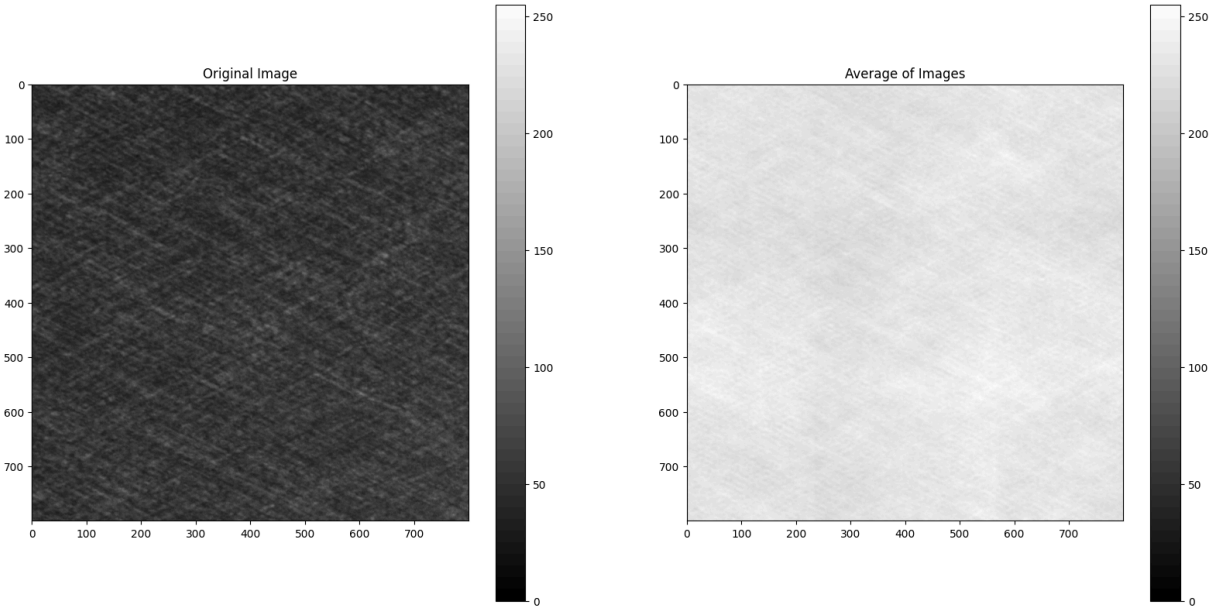



```
Image successfully saved
```

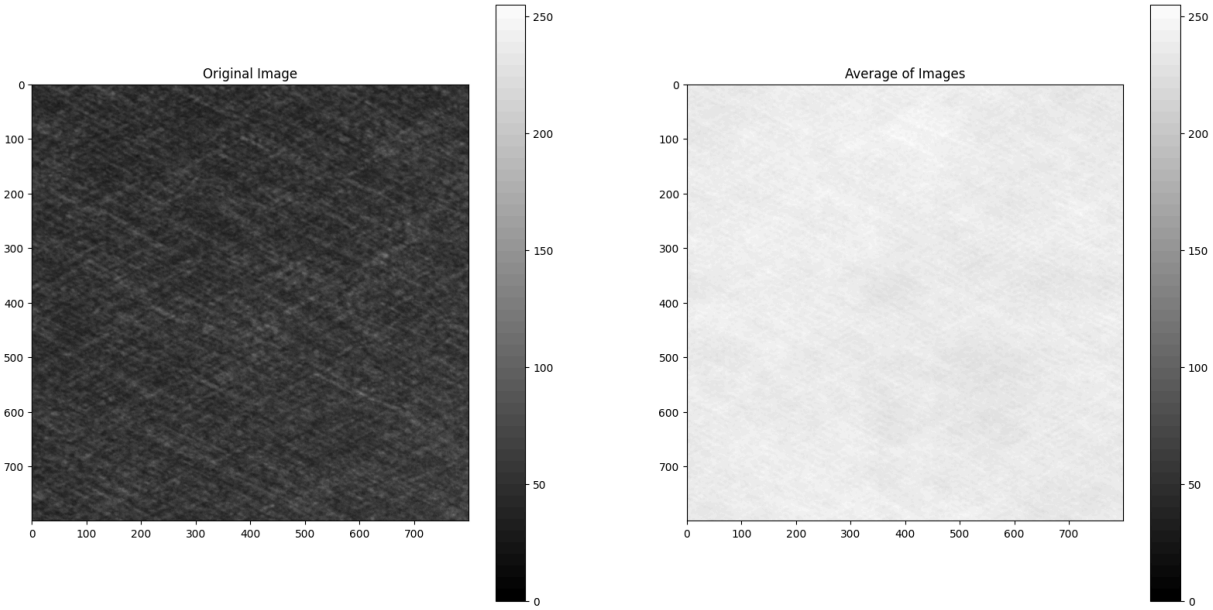




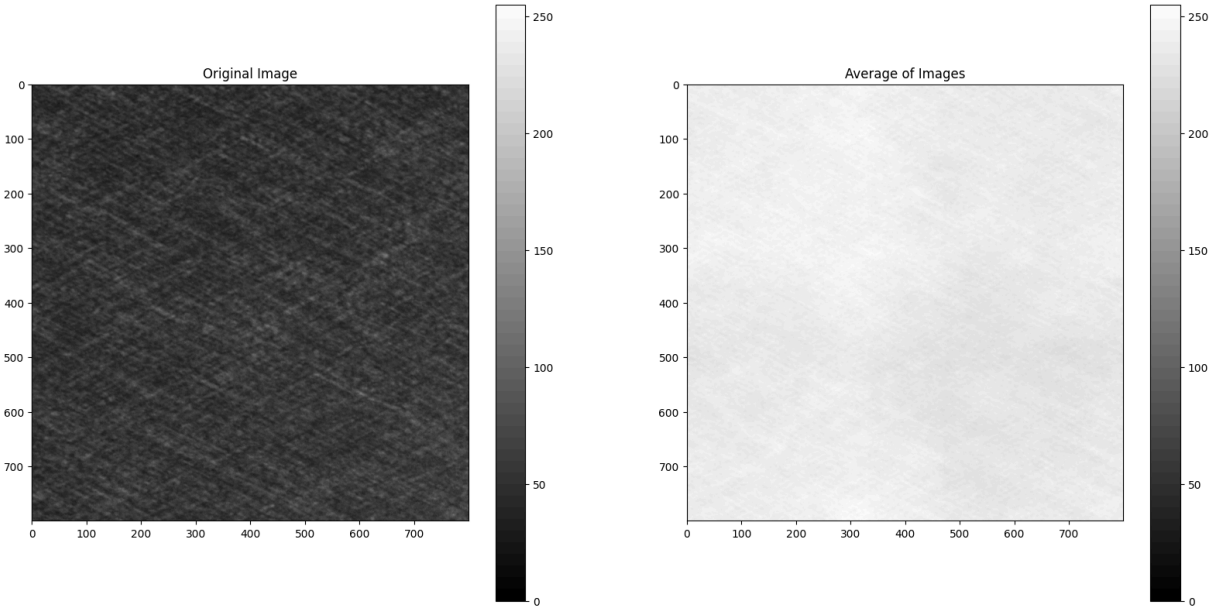

```
Image successfully saved
```



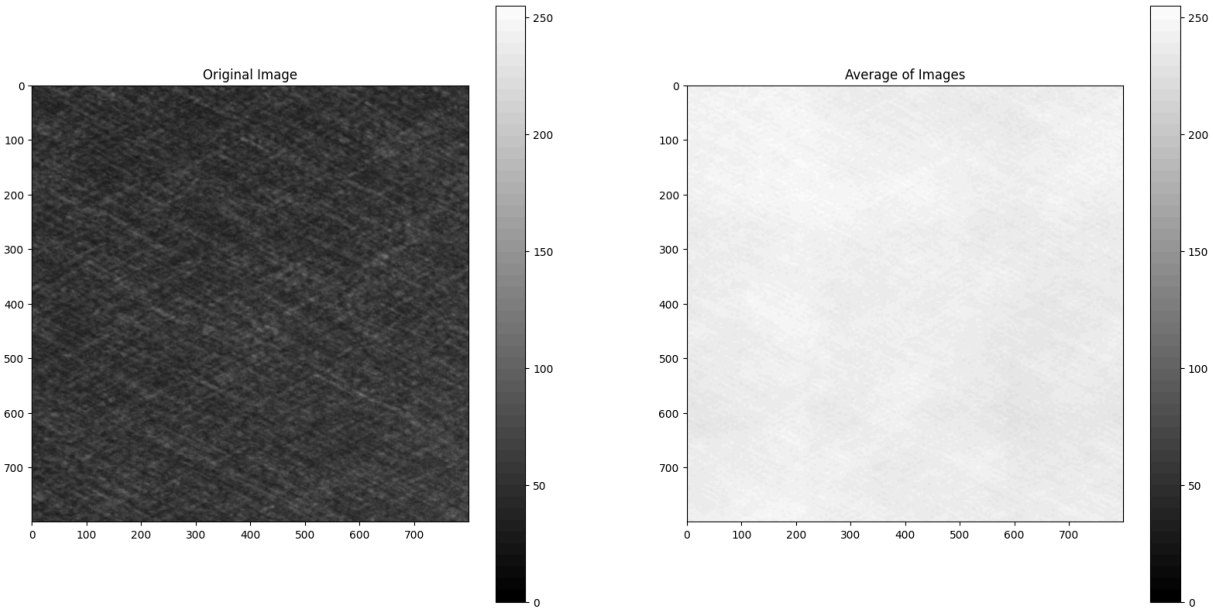

```
Image successfully saved
```




```
Image successfully saved
```




```
Image successfully saved
```

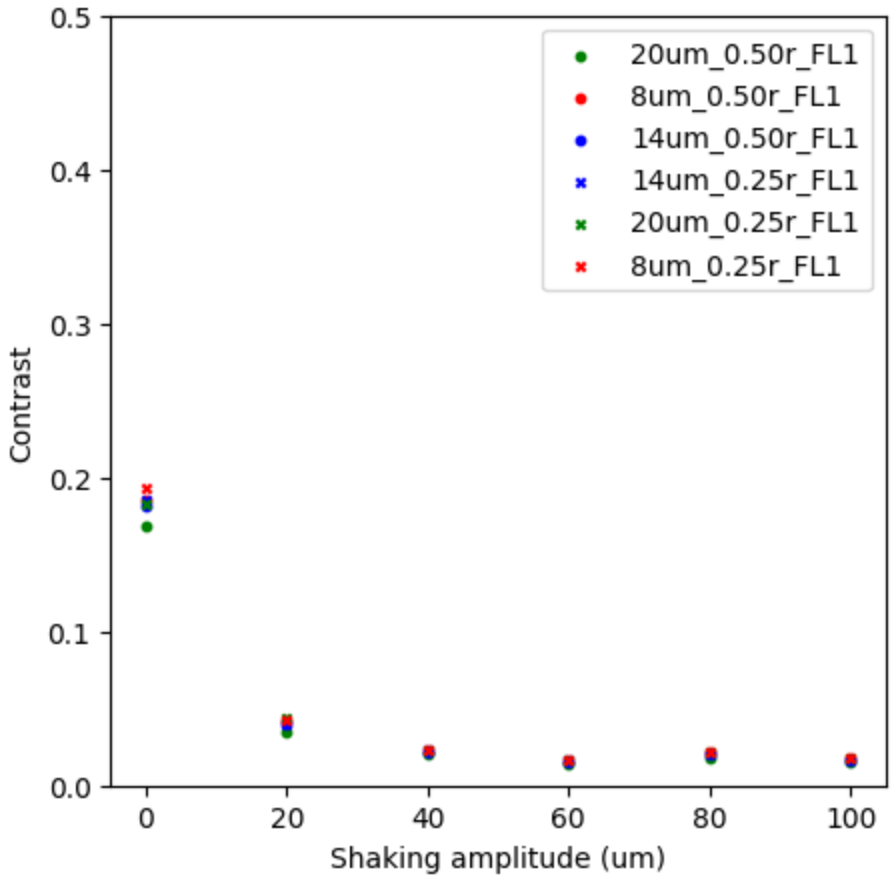



```
Image successfully saved
```

In [39]:
```python
import os
import matplotlib.pyplot as plt
# Example folder path
folder_path = '/Users/qingjunwang/Documents/image files/speckle 2 study/crop

data = {}

for filename in os.listdir(folder_path):
    if filename.endswith(".png"):
        parts = filename.split('_')
        label = '_'.join(parts[:3])
        x_value = float(parts[4].replace('um', ''))
        y_value = float(parts[-1].split('contrast')[1].replace('.png', ''))

        if label not in data:
            data[label] = {'x': [], 'y': []}
```

```
        data[label]['x'].append(x_value)
        data[label]['y'].append(y_value)

print(data)
fig, ax = plt.subplots(figsize=(5, 5))
colors = plt.cm.viridis(np.linspace(0, 1, len(data)))
markers = ['o', 's', '^', 'p', '*', '+', 'x', 'D', 'H', 'v']
size=10

# for (label, values), color, marker in zip(data.items(), colors, markers):
#     if label=='8um_0.50r_FL1':
#         ax.scatter(values['x'], (values['y']), label=label, color="red", m
#     if label=='8um_0.25r_FL1':
#         ax.scatter(values['x'], (values['y']), label=label, color="red", m
#     if label=='20um_0.50r_FL1':
#         ax.scatter(values['x'], (values['y']), label=label, color="green",
#     if label=='20um_0.25r_FL1':
#         ax.scatter(values['x'], (values['y']), label=label, color="green",
#     if label=='14um_0.50r_FL1':
#         ax.scatter(values['x'], (values['y']), label=label, color="blue",
#     if label=='14um_0.25r_FL1':
#         ax.scatter(values['x'], (values['y']), label=label, color="blue",
# ax.set_xlabel('Shaking amplitude (um)')
# ax.set_ylabel('Contrast')
# ax.set_ylim(0., 0.5)

# ax.legend(title='PIC deisigns')
# plt.show()

# Plotting loop
for (label, values) in data.items():
    if label == '8um_0.50r_FL1':
        ax.scatter(values['x'], values['y'], label=label, color="red", marke
    elif label == '8um_0.25r_FL1':
        ax.scatter(values['x'], values['y'], label=label, color="red", marke
    elif label == '20um_0.50r_FL1':
        ax.scatter(values['x'], values['y'], label=label, color="green", mar
    elif label == '20um_0.25r_FL1':
        ax.scatter(values['x'], values['y'], label=label, color="green", mar
    elif label == '14um_0.50r_FL1':
        ax.scatter(values['x'], values['y'], label=label, color="blue", mark
    elif label == '14um_0.25r_FL1':
        ax.scatter(values['x'], values['y'], label=label, color="blue", mark
# Setting labels and limits
ax.set_xlabel('Shaking amplitude (um)')
ax.set_ylabel('Contrast')
ax.set_ylim(0., 0.5)
# Display the plot
plt.legend()
plt.show()
```

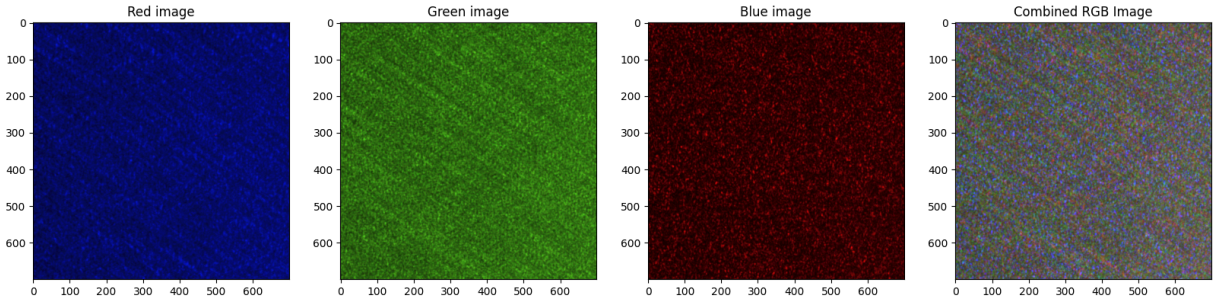
{'20um_0.50r_FL1': {'x': [80.0, 60.0, 40.0, 20.0, 0.0, 100.0], 'y': [0.01799
5508387684822, 0.014514430426061153, 0.02069002576172352, 0.0340883620083332
06, 0.16856133937835693, 0.014782347716391087]}, '8um_0.50r_FL1': {'x': [0.
0, 40.0, 80.0, 60.0, 100.0, 20.0], 'y': [0.1854531168937683, 0.0232482496649
02687, 0.02159913070499897, 0.016546787694096565, 0.017510542646050453, 0.04
275994375348091]}, '14um_0.50r_FL1': {'x': [40.0, 60.0, 80.0, 100.0, 20.0,
0.0], 'y': [0.022044111043214798, 0.015316733159124851, 0.02046981826424598
7, 0.016125893220305443, 0.03963811323046684, 0.18200072646141052]}, '14um_
0.25r_FL1': {'x': [40.0, 100.0, 0.0, 60.0, 20.0, 80.0], 'y': [0.023288616910
57682, 0.016575690358877182, 0.18543119728565216, 0.016213173046708107, 0.03
943619132041931, 0.02031904645264148 7]}, '20um_0.25r_FL1': {'x': [0.0, 100.
0, 40.0, 80.0, 20.0, 60.0], 'y': [0.18326689302921295, 0.017638321965932846,
0.022777119651436806, 0.022160161286592484, 0.04443974792957306, 0.016229918
226599693]}, '8um_0.25r_FL1': {'x': [100.0, 80.0, 0.0, 60.0, 20.0, 40.0],
'y': [0.017499219626188278, 0.021689338609576225, 0.193216010928154, 0.01679
857075214386, 0.043016910552978516, 0.02362787164747715]}}



In [1]:
```
#FL2 20um r sample
```

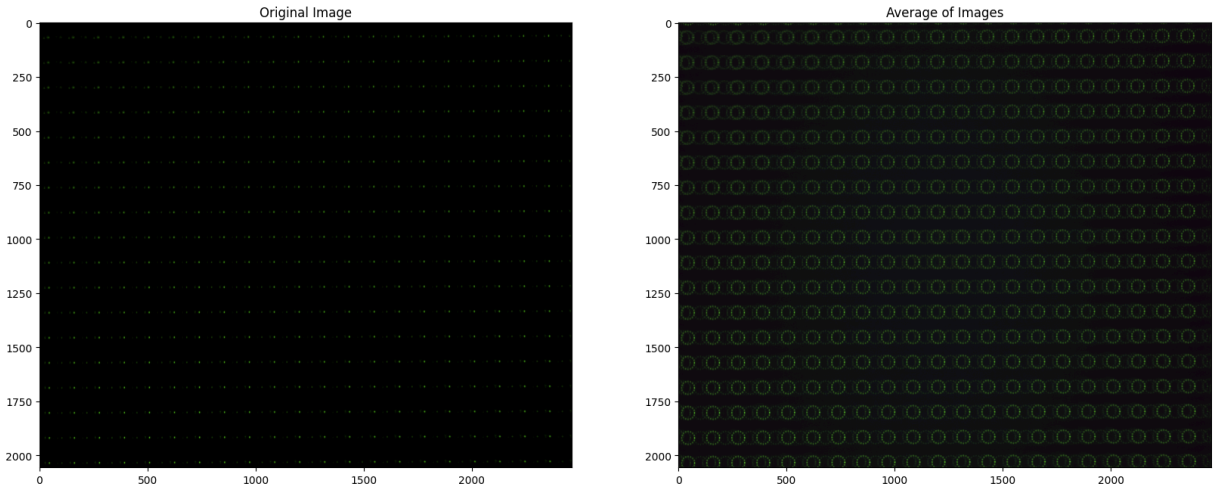
In [4]:
```
# #same z rgb image
#     '20um_0.05r_FL2':35.606,
#     '20um_0.05r_FL2_red':35.627,
#     '20um_0.05r_FL2_blue':35.606,
# 35.627-4=31.627
# 35.606-4=31.606
```

In [2]:
```
folder='/Users/qingjunwang/Documents/image files/speckle 2 study/yuhang/RGB
image_save_path='/Users/qingjunwang/Documents/image files/speckle 2 study/yu

main.RGB_combine_into_white(
    folder+'EPIC_FL2_D12C5_YSSI_P20_rSD_Red_45mA_25degrees_X0.000_Y0.000_Z31
    folder+'EPIC_FL2_D12C5_YSSI_P20_rSD_Green_50mA_25degrees_X0.000_Y0.000_Z
    folder+'EPIC_FL2_D12C5_YSSI_P20_rSD_Blue_30mA_25degrees_X0.000_Y0.000_Z3
    700,
    image_save_path
)
```




```
scaled image saved
```

In [25]:
```
contrast={}
dic_images_key_path={
'red':'/Users/qingjunwang/Documents/image files/speckle 2 study/yuhang/RGB c
'green':'/Users/qingjunwang/Documents/image files/speckle 2 study/yuhang/RGE
'blue':'/Users/qingjunwang/Documents/image files/speckle 2 study/yuhang/RGB
'combined_white':'/Users/qingjunwang/Documents/image files/speckle 2 study/y
}
dic_images_key_path={
    'test': '/Users/qingjunwang/Documents/image files/speckle 2 study/yuhang

}
image_pixel_per_um = 2464 / 420
amplitude_arr=[5]
select_color_chan=0
color='green'
img_save_path='/Users/qingjunwang/Documents/image files/speckle 2 study/yuha
contrast=main.shaking_scan_amplitude(dic_images_key_path, amplitude_arr, ima
```



```
Image successfully saved to /Users/qingjunwang/Documents/image files/speckle
2 study/yuhang/RGB combination for shaking check/shaking_simulation/test_17p
oints_5um_contrast0.4824529330303748.png
Image saved: /Users/qingjunwang/Documents/image files/speckle 2 study/yuhan
g/RGB combination for shaking check/shaking_simulation/test_17points_5um_con
trast0.4824529330303748.png
```

In [16]:
```python
import os
import re
import cv2

# Load images from the specified folder path
folder_path = '/Users/qingjunwang/Documents/image files/speckle 2 study/yuha
image_files = [f for f in os.listdir(folder_path) if f.endswith('.png')]  #
images = {}

def parse_filename(filename):
    if 'blue' in filename:
        image_type = 'blue'
    elif 'red' in filename:
        image_type = 'red'
    elif 'green'in filename:
        image_type = 'green'
    else:
        image_type = 'white'
    match = re.match(r".*_(\d+)um_.*", filename)
    if match:
        amplitude = match.group(1)
        return amplitude, image_type
    return None, None

for image_file in image_files:
    amplitude, image_type = parse_filename(image_file)
    if amplitude and image_type:
        image_path = os.path.join(folder_path, image_file)
        img = cv2.imread(image_path)
        if img is not None:
            images[(amplitude, image_type)] = img
        else:
            print(f"Failed to load image: {image_path}")
    else:
        print(f"Filename does not match expected pattern: {image_file}")
```

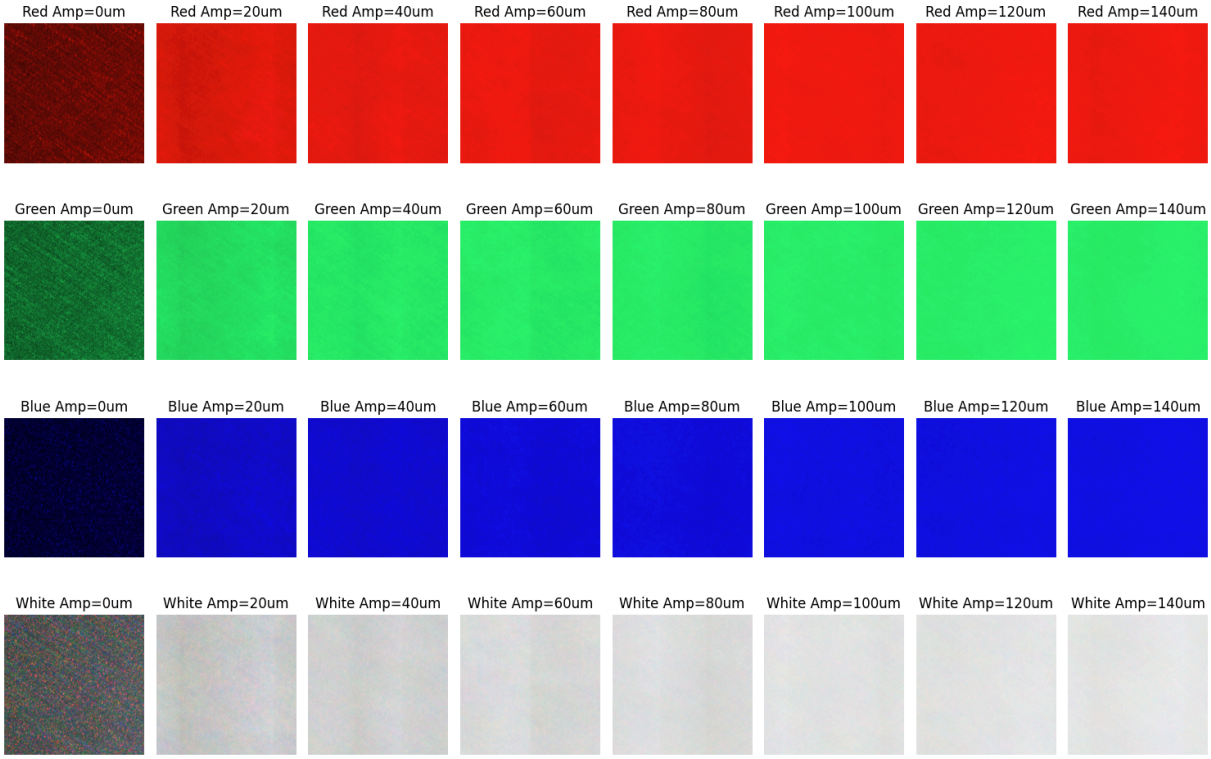
In [12]:
```python
len(images.keys())
```

Out[12]: 32

In [13]:
```python
images.keys()
```

Out[13]:
```
dict_keys([('0', 'blue'), ('0', 'white'), ('60', 'green'), ('120', 'gree
n'), ('40', 'red'), ('80', 'green'), ('40', 'green'), ('20', 'blue'), ('6
0', 'white'), ('80', 'white'), ('80', 'red'), ('0', 'red'), ('120', 'red'),
('100', 'red'), ('60', 'red'), ('140', 'white'), ('40', 'blue'), ('100', 'g
reen'), ('20', 'green'), ('80', 'blue'), ('60', 'blue'), ('20', 'white'),
('140', 'blue'), ('100', 'blue'), ('0', 'green'), ('140', 'green'), ('40',
'white'), ('100', 'white'), ('120', 'white'), ('140', 'red'), ('120', 'blu
e'), ('20', 'red')])
```

In [22]:
```
import matplotlib.pyplot as plt
import numpy as np
import cv2  # Assuming images are loaded using OpenCV


color_order = [ 'red', 'green','blue','white']

fig, axs = plt.subplots(len(color_order), 8, figsize=(15, 10))  # Adjust the

# Loop through each color and plot in a row
for i, color in enumerate(color_order):
    # Filter images by color and sort by amplitude
    filtered_images = {k: v for k, v in images.items() if k[1] == color}
    sorted_images = sorted(filtered_images.items(), key=lambda x: int(x[0][0

    # Create a subplot for each image
    for j, ((amplitude, _), img) in enumerate(sorted_images):
        ax = plt.subplot(len(color_order), len(sorted_images), i * len(sorte
        ax.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))  # Convert BGR to RG
        ax.axis('off')  # Hide axes
        ax.set_title(f"{color.capitalize()} Amp={amplitude}um")

# Adjust layout
plt.tight_layout()
plt.show()
```



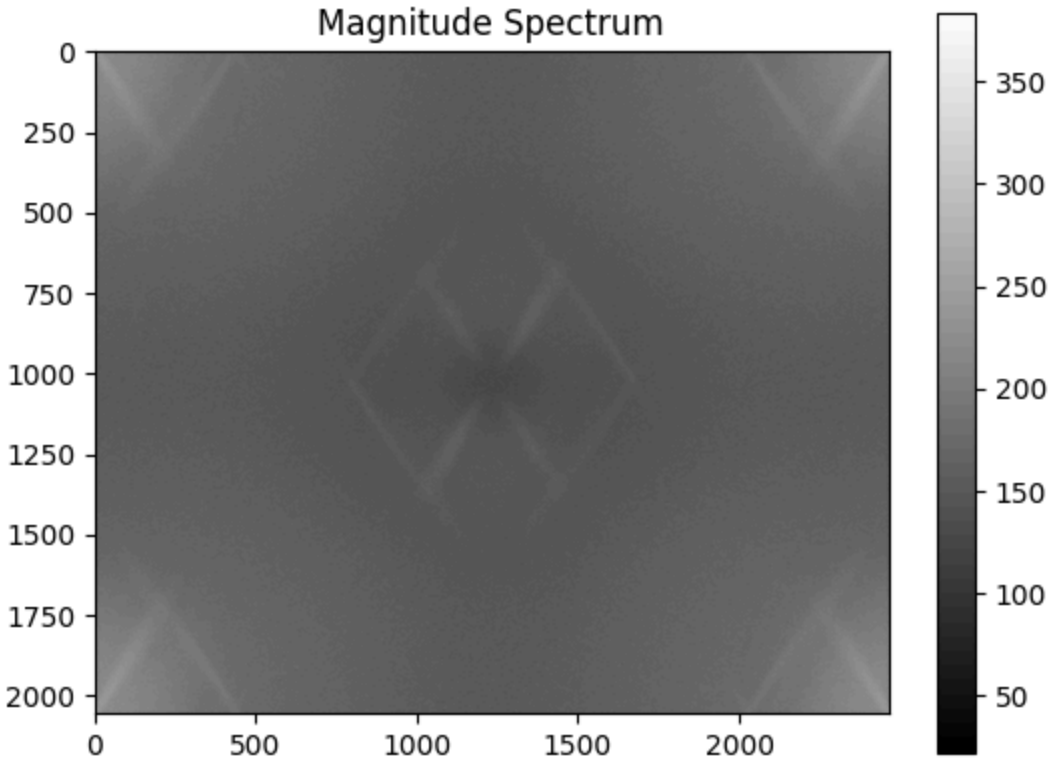
per PIC design, z=31mm, contrast-frequency

In [8]:
```
# downselect z
import matplotlib.pyplot as plt
# Define folders for each dataset
folders = {
    '20um_0.25r': '/Users/qingjunwang/Documents/image files/speckle 2 study/
    '20um_0.50r': '/Users/qingjunwang/Documents/image files/speckle 2 study/
    '14um_0.25r': '/Users/qingjunwang/Documents/image files/speckle 2 study/
    '14um_0.50r': '/Users/qingjunwang/Documents/image files/speckle 2 study/
    '8um_0.25r': '/Users/qingjunwang/Documents/image files/speckle 2 study/E
    '8um_0.50r': '/Users/qingjunwang/Documents/image files/speckle 2 study/E
}
# Dictionary to store closest images and their Z-values
closest_images = {}
closest_z_values = {}
targe_z=31
# Find closest images for each dataset
for label, folder in folders.items():
    closest_image, closest_z_value = main.find_closest_z_image(folder,targe_
    closest_images[label] = closest_image
    closest_z_values[label] = closest_z_value
    print(f"Closest image for {label}: {closest_image} with Z-value: {closes
```

```
Closest image for 20um_0.25r: /Users/qingjunwang/Documents/image files/speck
le 2 study/EPIC-FL1/NAP_20um/EPIC-FL1_D0C0_NAP-20um-0.25R_single_image_folde
r/EPIC-FL1_D0C0_NAP-20um-P20-0.25r_Green__X0.000_Y0.000_Z30.977_Exp100000.pn
g with Z-value: 30.977
Closest image for 20um_0.50r: /Users/qingjunwang/Documents/image files/speck
le 2 study/EPIC-FL1/NAP_20um/EPIC-FL1_D0C0_NAP-20um-0.50R_single_image_folde
r/EPIC-FL1_D0C0_NAP-20um-P20-0.50r_Green__X0.000_Y0.000_Z30.991_Exp100000.pn
g with Z-value: 30.991
Closest image for 14um_0.25r: /Users/qingjunwang/Documents/image files/speck
le 2 study/EPIC-FL1/NAP_14um/EPIC-FL1_D0C0_NAP-14um-0.25R_single_image_folde
r/EPIC-FL1_D0C0_NAP-14um-P20-0.25r_Green__X0.000_Y0.000_Z30.985_Exp100000.pn
g with Z-value: 30.985
Closest image for 14um_0.50r: /Users/qingjunwang/Documents/image files/speck
le 2 study/EPIC-FL1/NAP_14um/EPIC-FL1_D0C0_NAP-14um-0.50R_single_image_folde
r/EPIC-FL1_D0C0_NAP-14um-P20-0.50r_Green__X0.000_Y0.000_Z30.999_Exp30784.png
with Z-value: 30.999
Closest image for 8um_0.25r: /Users/qingjunwang/Documents/image files/speckl
e 2 study/EPIC-FL1/NAP_8um/EPIC-FL1_D0C0_NAP-8um-0.25R_single_image_folder/E
PIC-FL1_D0C0_NAP-8um-P20-0.25r_Green__X0.000_Y0.000_Z30.988_Exp32812.png wit
h Z-value: 30.988
Closest image for 8um_0.50r: /Users/qingjunwang/Documents/image files/speckl
e 2 study/EPIC-FL1/NAP_8um/EPIC-FL1_D0C0_NAP-8um-0.50R_single_image_folder/E
PIC-FL1_D0C0_NAP-8um-P20-0.50r_Green__X0.000_Y0.000_Z31.002_Exp120000.png wi
th Z-value: 31.002
```

In [9]:

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt
from scipy.fftpack import fftshift, fft2

def fft_image(image, shift_back=False):
    f = np.fft.fft2(image)
    #f = fftshift(fft2(image)) #julian
    magnitude_spectrum = 20 * np.log(np.abs(f) + 1)
    #magnitude_spectrum = np.abs(f)**2
    # magnitude_spectrum = np.abs(f)
    return magnitude_spectrum

# Load image
file_path = "/Users/qingjunwang/Documents/image files/speckle 2 study/EPIC-F
image = cv2.imread(file_path, cv2.IMREAD_GRAYSCALE)

magnitude_spectrum = fft_image(image)
radius=3500
# Display the magnitude spectrum
plt.imshow(magnitude_spectrum, cmap='gray')
#plt.xlim(-0, 450)
#plt.ylim(-0, 450)

plt.title('Magnitude Spectrum')
plt.colorbar()
plt.show()
```



In [10]:
```
max_radius=250
ring_width=1
import os
import cv2
base_folder = '/Users/qingjunwang/Documents/image files/speckle 2 study/'
img_ave_num = {}
for label, filename in closest_images.items():
    file_path = os.path.join(base_folder, filename)
    img = cv2.imread(file_path, cv2.IMREAD_GRAYSCALE)
    img_ave_num[label]=img.mean()
print('scale calculation finished')
max_value = max(img_ave_num.values())
scale_parameters = {label: max_value / value for label, value in img_ave_num

print(f"The maximum value is {max_value}")
print("Scale parameters for each label to reach the maximum value:")
for label, scale in scale_parameters.items():
    print(f"{label}: {scale}")
```

```
scale calculation finished
The maximum value is 69.30972778437516
Scale parameters for each label to reach the maximum value:
20um_0.25r: 1.8296536373395482
20um_0.50r: 1.303431610047318
14um_0.25r: 1.0
14um_0.50r: 1.6801692846222747
8um_0.25r: 1.2131190766736943
8um_0.50r: 1.6419212769010516
```

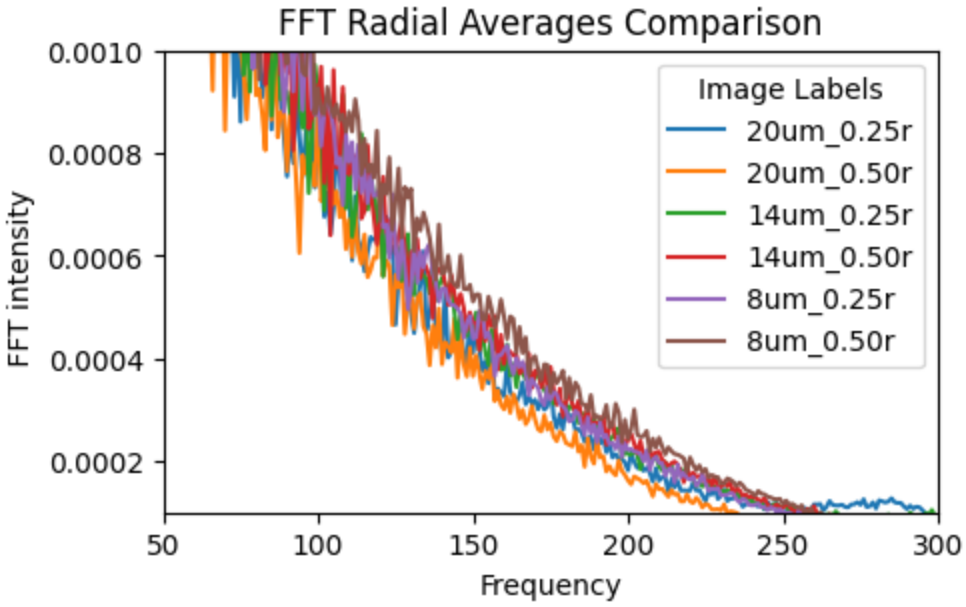
In [11]:
```
max_radius=250
ring_width=1
import os
```

```
import cv2
base_folder = '/Users/qingjunwang/Documents/image files/speckle 2 study/'
fft_results = {}
img_ave_num = {}
for label, filename in closest_images.items():
    file_path = os.path.join(base_folder, filename)
    img = cv2.imread(file_path, cv2.IMREAD_GRAYSCALE)
    img=img*scale_parameters[label]
    if img is None:
        print(f"Failed to load image {file_path}")
        continue
    print(img.shape)
    radial_average = main.fft2Dring(img, max_radius, ring_width)
    fft_results[label] = radial_average
print('2D fft finished')
```

```
(2056, 2464)
halfDim=1233
(2056, 2464)
halfDim=1233
(2056, 2464)
halfDim=1233
(2056, 2464)
halfDim=1233
(2056, 2464)
halfDim=1233
(2056, 2464)
halfDim=1233
2D fft finished
```

In [18]:

```
import matplotlib.pyplot as plt
# Define the radius range based on the max_radius and ring_width used in the
radius_range = range(max_radius)  # Adjust this if max_radius or ring_width
# Create a plot for the FFT radial averages
plt.figure(figsize=(5, 3))  # Set the figure size
for label, radial_average in fft_results.items():
    # plt.plot(np.arange(1233), np.log(radial_average), label=label)
    plt.plot(np.arange(1233), radial_average, label=label)
# Adding plot title and labels
plt.title('FFT Radial Averages Comparison')
plt.xlabel('Frequency')
plt.ylabel(' FFT intensity')
# plt.ylim(-10,0)
plt.xlim(50,300)
plt.ylim(10**-4,10**-3)
# plt.xlim(0,350)

#plt.xlim(0,400)

# Adding a legend to identify each line
plt.legend(title='Image Labels')
# Show the plot
plt.show()
```



In [13]:
```
import pandas as pd

# Assuming z_data and contrast_data are dictionaries with matching keys and
# z_data = {'20um_0.25r': [z1, z2, ...], '20um_0.50r': [z1, z2, ...], ...}
# contrast_data = {'20um_0.25r': [c1, c2, ...], '20um_0.50r': [c1, c2, ...],

# Create a list to hold all the data in a structured format
data = []

# Iterate over the items in z_data and contrast_data
for sample_type in z_data:
    for z, contrast in zip(z_data[sample_type], contrast_data[sample_type]):
        data.append({
            'Sample Type': sample_type,
            'Z in mm': z,
            'Contrast': contrast
        })

# Convert the list of dictionaries into a DataFrame
df = pd.DataFrame(data)

# Save the DataFrame to a CSV file
df.to_csv('contrast_data.csv', index=False)

print("Data saved to '/Users/qingjunwang/Documents/image files/speckle 2 stu
```

```
Data saved to '/Users/qingjunwang/Documents/image files/speckle 2 study/freq
uency_contrast_data_31mm.csv'.
```

Check simulation result of different PIC design

In [1]:
```
import os
import re

# Define the directory path
```

```
directory = '/Users/qingjunwang/Downloads/1000um'
directory = "/Users/qingjunwang/Documents/image files/PIC deisgn- large pitc
# Create an empty dictionary to store the image paths with labels
image_dict = {}

# Loop through all files in the directory
for filename in os.listdir(directory):
    # Check if the file is a PNG image
    if filename.endswith('.png'):
        # Extract the information from the filename using regular expressior
        match = re.search(r'-(\d+.\d+random_\d+um)', filename)
        if match:
            label = match.group(1)  # Extract the label from the match
            image_path = os.path.join(directory, filename)  # Get the full i
            image_dict[label] = image_path  # Add the image path to the dict

# Print the resulting dictionary
print(image_dict)
```

{'0.25random_120um': '/Users/qingjunwang/Documents/image files/PIC deisgn- large pitch simulation/sum_electric_field=1_note_1800um_image_size-0.25random_120um_pitch_1plane_0umdistance_1000umdistance0_0intensityarray_contrast0.09818.png', '0.05random_140um': '/Users/qingjunwang/Documents/image files/PIC deisgn- large pitch simulation/sum_electric_field=1_note_2100um_image_size-0.05random_140um_pitch_1plane_0umdistance_1000umdistance0_0intensityarray_contrast0.10946.png', '0.05random_120um': '/Users/qingjunwang/Documents/image files/PIC deisgn- large pitch simulation/sum_electric_field=1_note_1800um_image_size-0.05random_120um_pitch_1plane_0umdistance_1000umdistance0_0intensityarray_contrast0.08276.png', '0.1random_40um': '/Users/qingjunwang/Documents/image files/PIC deisgn- large pitch simulation/sum_electric_field=1_note_600um_image_size-0.1random_40um_pitch_1plane_0umdistance_1000umdistance0_0intensityarray_contrast0.29058.png', '0.01random_140um': '/Users/qingjunwang/Documents/image files/PIC deisgn- large pitch simulation/sum_electric_field=1_note_2100um_image_size-0.01random_140um_pitch_1plane_0umdistance_1000umdistance0_0intensityarray_contrast0.10955.png', '0.01random_40um': '/Users/qingjunwang/Documents/image files/PIC deisgn- large pitch simulation/sum_electric_field=1_note_600um_image_size-0.01random_40um_pitch_1plane_0umdistance_1000umdistance0_0intensityarray_contrast0.47446.png', '0.25random_60um': '/Users/qingjunwang/Documents/image files/PIC deisgn- large pitch simulation/sum_electric_field=1_note_900um_image_size-0.25random_60um_pitch_1plane_0umdistance_1000umdistance0_0intensityarray_contrast0.19216.png', '0.1random_60um': '/Users/qingjunwang/Documents/image files/PIC deisgn- large pitch simulation/sum_electric_field=1_note_900um_image_size-0.1random_60um_pitch_1plane_0umdistance_1000umdistance0_0intensityarray_contrast0.19384.png', '0.05random_80um': '/Users/qingjunwang/Documents/image files/PIC deisgn- large pitch simulation/sum_electric_field=1_note_1200um_image_size-0.05random_80um_pitch_1plane_0umdistance_1000umdistance0_0intensityarray_contrast0.15525.png', '0.25random_100um': '/Users/qingjunwang/Documents/image files/PIC deisgn- large pitch simulation/sum_electric_field=1_note_1500um_image_size-0.25random_100um_pitch_1plane_0umdistance_1000umdistance0_0intensityarray_contrast0.12851.png', '0.25random_80um': '/Users/qingjunwang/Documents/image files/PIC deisgn- large pitch simulation/sum_electric_field=1_note_1200um_image_size-0.25random_80um_pitch_1plane_0umdistance_1000umdistance0_0intensityarray_contrast0.15562.png', '0.25random_40um': '/Users/qingjunwang/Documents/image files/PIC deisgn- large pitch simulation/sum_electric_field=1_note_600um_image_size-0.25random_40um_pitch_1plane_0umdistance_1000umdistance0_0intensityarray_contrast0.23405.png', '0.1random_120um': '/Users/qingjunwang/Documents/image files/PIC deisgn- large pitch simulation/sum_electric_field=1_note_1800um_image_size-0.1random_120um_pitch_1plane_0umdistance_1000umdistance0_0intensityarray_contrast0.08417.png', '0.1random_100um': '/Users/qingjunwang/Documents/image files/PIC deisgn- large pitch simulation/sum_electric_field=1_note_1500um_image_size-0.1random_100um_pitch_1plane_0umdistance_1000umdistance0_0intensityarray_contrast0.11907.png', '0.01random_100um': '/Users/qingjunwang/Documents/image files/PIC deisgn- large pitch simulation/sum_electric_field=1_note_1500um_image_size-0.01random_100um_pitch_1plane_0umdistance_1000umdistance0_0intensityarray_contrast0.13653.png', '0.1random_20um': '/Users/qingjunwang/Documents/image files/PIC deisgn- large pitch simulation/sum_electric_field=1_note_300um_image_size-0.1random_20um_pitch_1plane_0umdistance_1000umdistance0_0intensityarray_contrast0.27072.png', '0.05random_20um': '/Users/qingjunwang/Documents/image files/PIC deisgn- large pitch simulation/sum_electric_field=1_note_300um_image_size-0.05random_20um_pitch_1plane_0umdistance_1000umdistance0_0intensityarray_contrast0.27273.png', '0.05random_100um': '/Users/qingjunwang/Documents/image files/PIC deisgn- large pitch simulation/sum_electric_field=1_note_1500um_image_size-0.05random_100um_pitch_1plane_0umdistance_1000umdistance0_0intensityarray_contrast0.11888.png', '0.01random_60um':

'/Users/qingjunwang/Documents/image files/PIC deisgn- large pitch simulation/sum_electric_field=1_note_900um_image_size-0.01random_60um_pitch_1plane_0umdistance_1000umdistance0_0intensityarray_contrast0.14211.png', '0.05random_40um': '/Users/qingjunwang/Documents/image files/PIC deisgn- large pitch simulation/sum_electric_field=1_note_600um_image_size-0.05random_40um_pitch_1plane_0umdistance_1000umdistance0_0intensityarray_contrast0.36568.png', '0.05random_60um': '/Users/qingjunwang/Documents/image files/PIC deisgn- large pitch simulation/sum_electric_field=1_note_900um_image_size-0.05random_60um_pitch_1plane_0umdistance_1000umdistance0_0intensityarray_contrast0.19511.png', '0.01random_20um': '/Users/qingjunwang/Documents/image files/PIC deisgn- large pitch simulation/sum_electric_field=1_note_300um_image_size-0.01random_20um_pitch_1plane_0umdistance_1000umdistance0_0intensityarray_contrast0.29977.png', '0.1random_80um': '/Users/qingjunwang/Documents/image files/PIC deisgn- large pitch simulation/sum_electric_field=1_note_1200um_image_size-0.1random_80um_pitch_1plane_0umdistance_1000umdistance0_0intensityarray_contrast0.15831.png', '0.01random_80um': '/Users/qingjunwang/Documents/image files/PIC deisgn- large pitch simulation/sum_electric_field=1_note_1200um_image_size-0.01random_80um_pitch_1plane_0umdistance_1000umdistance0_0intensityarray_contrast0.10506.png', '0.01random_120um': '/Users/qingjunwang/Documents/image files/PIC deisgn- large pitch simulation/sum_electric_field=1_note_1800um_image_size-0.01random_120um_pitch_1plane_0umdistance_1000umdistance0_0intensityarray_contrast0.07983.png', '0.25random_20um': '/Users/qingjunwang/Documents/image files/PIC deisgn- large pitch simulation/sum_electric_field=1_note_300um_image_size-0.25random_20um_pitch_1plane_0umdistance_1000umdistance0_0intensityarray_contrast0.23901.png', '0.25random_140um': '/Users/qingjunwang/Documents/image files/PIC deisgn- large pitch simulation/sum_electric_field=1_note_2100um_image_size-0.25random_140um_pitch_1plane_0umdistance_1000umdistance0_0intensityarray_contrast0.13137.png', '0.1random_140um': '/Users/qingjunwang/Documents/image files/PIC deisgn- large pitch simulation/sum_electric_field=1_note_2100um_image_size-0.1random_140um_pitch_1plane_0umdistance_1000umdistance0_0intensityarray_contrast0.11242.png'}

In [2]:
```
import re
import cv2
# Define folders for each dataset
labels=[]
contrasts=[]
file_names = os.listdir(directory)
pattern = re.compile(r'-(\d+.\d+random_\d+um)')
for file in file_names:
    match = pattern.search(file)
    if match:
        label = (match.group(1))
        file_path = os.path.join(directory, file)  # Corrected to use the fu
        img = cv2.imread(file_path, cv2.IMREAD_GRAYSCALE)
        if img is not None:  # Check if the image is loaded properly
            if img.mean() != 0:  # Ensure the denominator is not zero
                contrast_value = img.std() / img.mean()
                labels.append(label)
                contrasts.append(contrast_value)
```

In [3]:
```
data = dict(zip(labels, contrasts))
```

In [4]:
```
data
```

Out[4]:
```
{'0.25random_120um': 0.09853358486769466,
 '0.05random_140um': 0.10990780176455472,
 '0.05random_120um': 0.08305356130932177,
 '0.1random_40um': 0.2930096028696876,
 '0.01random_140um': 0.10997917654037355,
 '0.01random_40um': 0.4787087106301881,
 '0.25random_60um': 0.19319634534794447,
 '0.1random_60um': 0.19496585500071728,
 '0.05random_80um': 0.1559215079888253,
 '0.25random_100um': 0.1290446688028979,
 '0.25random_80um': 0.15630962505892532,
 '0.25random_40um': 0.23537522527827617,
 '0.1random_120um': 0.08446819540052114,
 '0.1random_100um': 0.11952855110946768,
 '0.01random_100um': 0.13711119930923218,
 '0.1random_20um': 0.27208343753175374,
 '0.05random_20um': 0.2739648159897403,
 '0.05random_100um': 0.11934367206479347,
 '0.01random_60um': 0.14269335438743744,
 '0.05random_40um': 0.3682258855531133,
 '0.05random_60um': 0.19639444795800304,
 '0.01random_20um': 0.30119772075867374,
 '0.1random_80um': 0.1589971012922418,
 '0.01random_80um': 0.10543995088738096,
 '0.01random_120um': 0.08011543645435039,
 '0.25random_20um': 0.24031613551605854,
 '0.25random_140um': 0.1319472753444284,
 '0.1random_140um': 0.11289172350246265}
```

In [6]:
```python
import re

# Define a function to extract the "XXXum" part from each key
def extract_um(key):
    match = re.search(r'\d+um', key)
    return int(match.group(0).replace('um', ''))

# Define a function to extract the "XXXrandom" part from each key
def extract_random(key):
    match = re.search(r'\d+\.\d+random', key)
    return float(match.group(0).replace('random', ''))

# Define a custom order for the "XXXrandom" values
random_order = {0.01: 1, 0.05: 2, 0.1: 3, 0.25: 4}

# Sort the dictionary by the "XXXum" part first, and then by the "XXXrandom"
sorted_data = {k: v for k, v in sorted(data.items(), key=lambda item: (extra

print(sorted_data)

# Replace 'random' with 'r' in the keys
updated_data = {key.replace('random', 'r'): value for key, value in sorted_d
print(updated_data)
```

```
{'0.01random_20um': 0.30119772075867374, '0.05random_20um': 0.27396481598974
03, '0.1random_20um': 0.27208343753175374, '0.25random_20um': 0.240316135516
05854, '0.01random_40um': 0.4787087106301881, '0.05random_40um': 0.368225885
5531133, '0.1random_40um': 0.2930096028696876, '0.25random_40um': 0.23537522
527827617, '0.01random_60um': 0.14269335438743744, '0.05random_60um': 0.1963
9444795800304, '0.1random_60um': 0.19496585500071728, '0.25random_60um': 0.1
9319634534794447, '0.01random_80um': 0.10543995088738096, '0.05random_80um':
0.1559215079888253, '0.1random_80um': 0.1589971012922418, '0.25random_80um':
0.15630962505892532, '0.01random_100um': 0.13711119930923218, '0.05random_10
0um': 0.11934367206479347, '0.1random_100um': 0.11952855110946768, '0.25rand
om_100um': 0.1290446688028979, '0.01random_120um': 0.08011543645435039, '0.0
5random_120um': 0.08305356130932177, '0.1random_120um': 0.08446819540052114,
'0.25random_120um': 0.09853358486769466, '0.01random_140um': 0.1099791765403
7355, '0.05random_140um': 0.10990780176455472, '0.1random_140um': 0.11289172
350246265, '0.25random_140um': 0.1319472753444284}
{'0.01r_20um': 0.30119772075867374, '0.05r_20um': 0.2739648159897403, '0.1r_
20um': 0.27208343753175374, '0.25r_20um': 0.24031613551605854, '0.01r_40um':
0.4787087106301881, '0.05r_40um': 0.3682258855531133, '0.1r_40um': 0.2930096
028696876, '0.25r_40um': 0.23537522527827617, '0.01r_60um': 0.14269335438743
744, '0.05r_60um': 0.19639444795800304, '0.1r_60um': 0.19496585500071728,
'0.25r_60um': 0.19319634534794447, '0.01r_80um': 0.10543995088738096, '0.05r
_80um': 0.1559215079888253, '0.1r_80um': 0.1589971012922418, '0.25r_80um':
0.15630962505892532, '0.01r_100um': 0.13711119930923218, '0.05r_100um': 0.11
934367206479347, '0.1r_100um': 0.11952855110946768, '0.25r_100um': 0.1290446
688028979, '0.01r_120um': 0.08011543645435039, '0.05r_120um': 0.083053561309
32177, '0.1r_120um': 0.08446819540052114, '0.25r_120um': 0.0985335848676946
6, '0.01r_140um': 0.10997917654037355, '0.05r_140um': 0.10990780176455472,
'0.1r_140um': 0.11289172350246265, '0.25r_140um': 0.1319472753444284}
```

In [12]:

```python
import matplotlib.pyplot as plt
import re

# Assuming 'updated_data' is a dictionary with your data
labels = list(updated_data.keys())
values = list(updated_data.values())

# Extract unique 'um' values to categorize data
um_values = set(re.search(r'\d+um', label).group(0) for label in labels)
um_to_color = {um: plt.cm.tab20(i) for i, um in enumerate(um_values)}  # Ass

plt.figure(figsize=(35, 5))

# Plot each item with a color based on its 'um' value
for label, value in updated_data.items():
    um = re.search(r'\d+um', label).group(0)
    color = um_to_color[um]
    plt.bar(label, value, color=color, width=0.2)

plt.xlabel('PIC design')
plt.ylabel('Contrast')
plt.title('Different PIC design, scam 20um-140um pitch, 0.01-0.25randomn')
plt.xticks(rotation=90)  # Rotate labels to prevent overlap
plt.xticks(fontsize=14)  # You can adjust the size to your preference
plt.show()
```

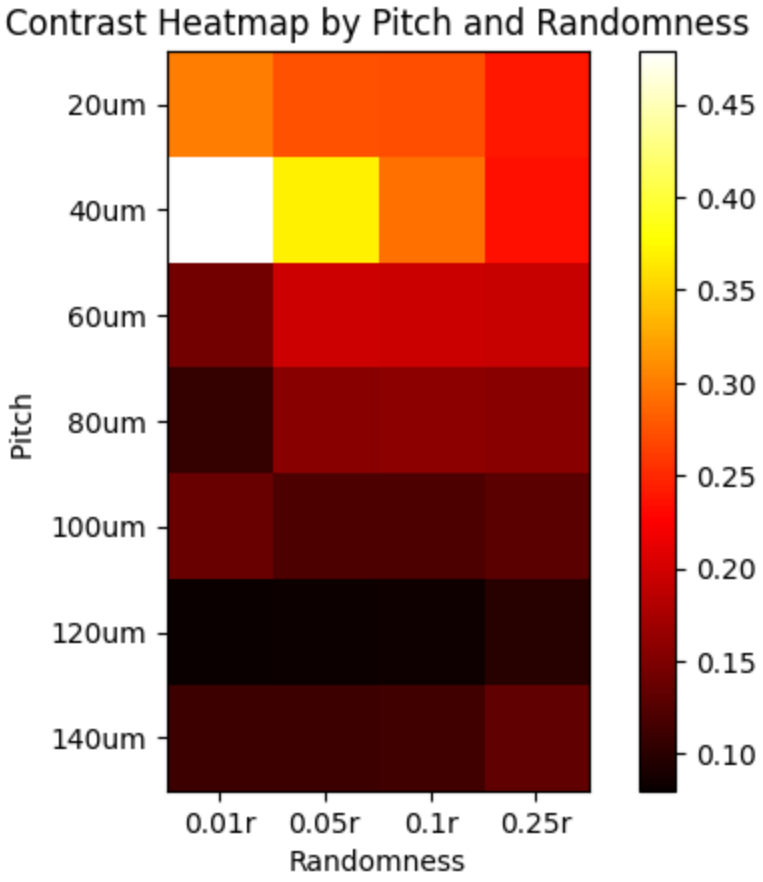
```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[12], line 5
      2 import re
      4 # Assuming 'updated_data' is a dictionary with your data
----> 5 labels = list(updated_data.keys())
      6 values = list(updated_data.values())
      8 # Extract unique 'um' values to categorize data

NameError: name 'updated_data' is not defined
```

Colormap

In [25]:

```
import os
import re
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
from PIL import Image
import numpy as np

# Define the directory paths
directories = [
    "/Users/qingjunwang/Documents/image files/PIC deisgn- large pitch simula
]

# Create an empty dictionary to store the image paths with labels
image_dict = {}

# Loop through all directories
for directory in directories:
    if not os.path.exists(directory):
        print(f"Directory does not exist: {directory}")
        continue
    # Loop through all files in the directory
    for filename in os.listdir(directory):
        # Check if the file is a PNG image
        if filename.endswith('.png'):
            # Extract the information from the filename using regular expres
            match = re.search(r'(\d+\.\d+random_\d+um)', filename)
            if match:
                label = match.group(1)  # Extract the label from the match
                image_path = os.path.join(directory, filename)  # Get the fu
                image_dict[label] = image_path  # Add the image path to the
            else:
                print(f"No match found for filename: {filename}")

if not image_dict:
    print("No images were loaded. Check the filename patterns and directory
    exit()

# Define the order of randomness and um values
random_order = ['0.01r', '0.05r', '0.1r', '0.25r']
um_order = ['20um', '40um', '60um', '80um', '100um', '120um', '140um']

# Initialize the contrast matrix
```

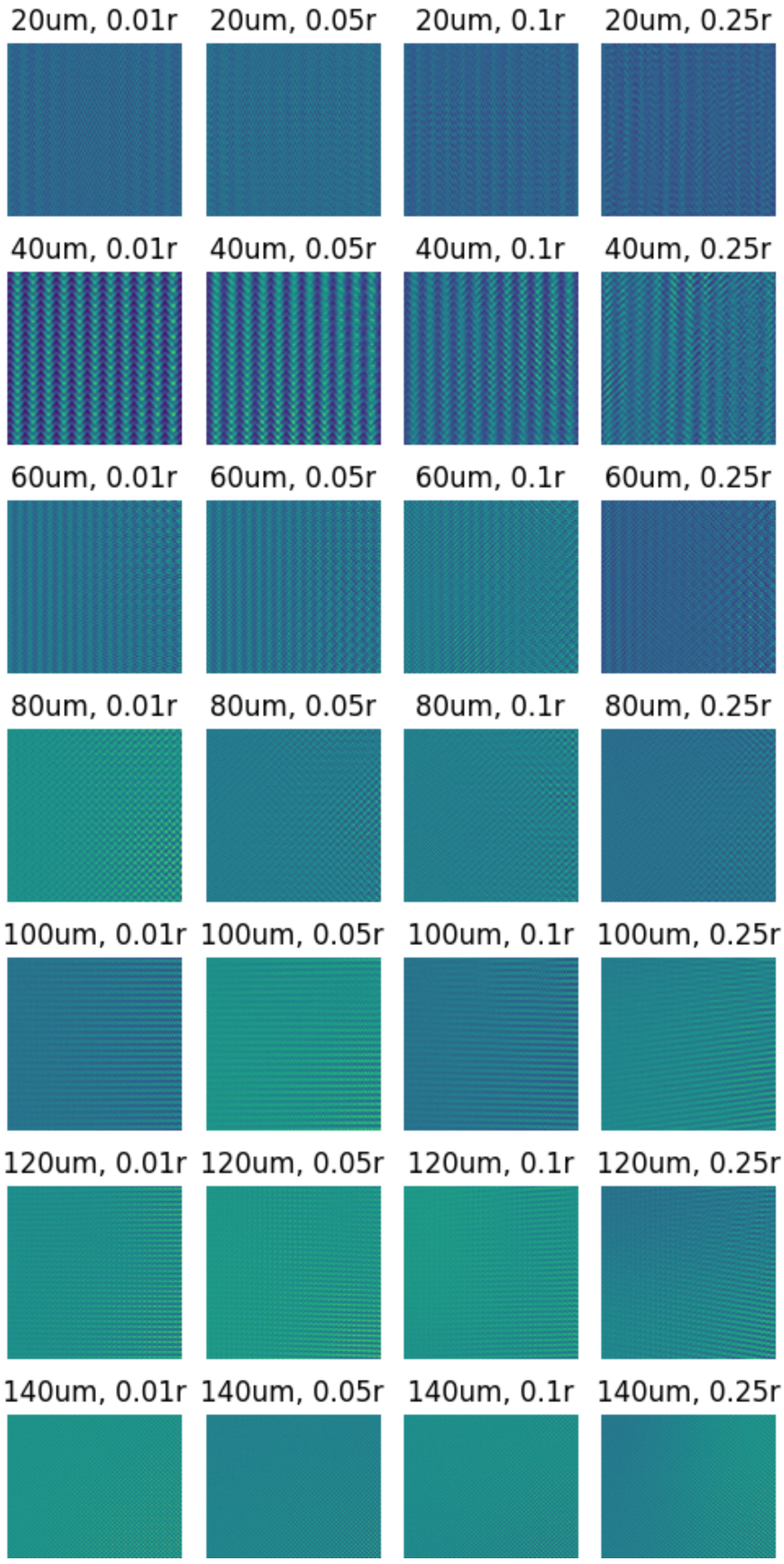
```
contrast_matrix = np.zeros((len(um_order), len(random_order)))

# Process images and calculate contrast
for label, path in image_dict.items():
    um = re.search(r'\d+um', label).group(0)
    random = re.search(r'\d+\.\d+r', label).group(0)
    img = Image.open(path)
    # if um != '20um':
    #     # Crop the middle 300um region for other images
    #     width, height = img.size
    #     left = (width - 300) // 2
    #     top = (height - 300) // 2
    #     right = left + 300
    #     bottom = top + 300
    #     img = img.crop((left, top, right, bottom))
    img_np = np.array(img)
    contrast = img_np.std() / img_np.mean()
    um_idx = um_order.index(um)
    random_idx = random_order.index(random)
    contrast_matrix[um_idx, random_idx] = contrast

# Plotting the colormap
fig, ax = plt.subplots()
cax = ax.imshow(contrast_matrix, cmap='hot', interpolation='nearest')
ax.set_xticks(np.arange(len(random_order)))
ax.set_yticks(np.arange(len(um_order)))
ax.set_xticklabels(random_order)
ax.set_yticklabels(um_order)
ax.set_xlabel('Randomness')
ax.set_ylabel('Pitch ')
plt.colorbar(cax)
plt.title('Contrast Heatmap by Pitch and Randomness')
plt.show()
```



In [24]:
```
import os
import re
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
from PIL import Image
import numpy as np

# Define the directory paths
directories = [
    '/Users/qingjunwang/Downloads/1000um',
    "/Users/qingjunwang/Documents/image files/PIC deisgn- large pitch simula
]

# Create an empty dictionary to store the image paths with labels
image_dict = {}

# Loop through all directories
for directory in directories:
    if not os.path.exists(directory):
        print(f"Directory does not exist: {directory}")
        continue
    # Loop through all files in the directory
    for filename in os.listdir(directory):
        # Check if the file is a PNG image
        if filename.endswith('.png'):
            # Extract the information from the filename using regular expres
            match = re.search(r'(\d+\.\d+random_\d+um)', filename)
            if match:
```

```
                label = match.group(1)  # Extract the label from the match
                image_path = os.path.join(directory, filename)  # Get the fu
                image_dict[label] = image_path  # Add the image path to the
            else:
                print(f"No match found for filename: {filename}")

if not image_dict:
    print("No images were loaded. Check the filename patterns and directory
    exit()

# Define the order of randomness
random_order = ['0.01r', '0.05r', '0.1r', '0.25r']

# Group images by 'um' and sort by randomness
sorted_images = {}
for label, path in image_dict.items():
    um = re.search(r'\d+um', label).group(0)
    random = re.search(r'\d+\.\d+r', label).group(0)
    if um not in sorted_images:
        sorted_images[um] = {}
    sorted_images[um][random] = path

# Define a custom order for the 'um' values
um_order = ['20um', '40um', '60um', '80um', '100um', '120um', '140um']

# Ensure there are rows to create subplots
num_rows = len(um_order)
num_cols = len(random_order)
if num_rows == 0 or num_cols == 0:
    print("No valid data to plot. Exiting.")
    exit()

# Plotting and calculating contrast
fig, axs = plt.subplots(num_rows, num_cols, figsize=(5, 10))
contrast_dict = {}

for i, um in enumerate(um_order):
    for j, random in enumerate(random_order):
        ax = axs[i, j]
        if um in sorted_images and random in sorted_images[um]:
            img = Image.open(sorted_images[um][random])
            if um != '20um':
                # Crop the middle 300um region for other images
                width, height = img.size
                left = (width - 300) // 2
                top = (height - 300) // 2
                right = left + 300
                bottom = top + 300
                img = img.crop((left, top, right, bottom))
            img_np = np.array(img)
            contrast = img_np.std() / img_np.mean()
            contrast_dict[f"{um}_{random}"] = contrast
            ax.imshow(img)
            ax.set_title(f"{um}, {random}")
        ax.axis('off')
```

```
plt.tight_layout()
plt.show()

# Print the contrast values
print("Contrast values:")
for key, contrast in contrast_dict.items():
    print(f"{key}: {contrast}")
```


20um, 0.01r
20um, 0.05r
20um, 0.1r
20um, 0.25r
40um, 0.01r
40um, 0.05r
40um, 0.1r
40um, 0.25r
60um, 0.01r
60um, 0.05r
60um, 0.1r
60um, 0.25r
80um, 0.01r
80um, 0.05r
80um, 0.1r
80um, 0.25r
100um, 0.01r
100um, 0.05r
100um, 0.1r
100um, 0.25r
120um, 0.01r
120um, 0.05r
120um, 0.1r
120um, 0.25r
140um, 0.01r
140um, 0.05r
140um, 0.1r
140um, 0.25r

```
Contrast values:
20um_0.01r: 0.30119772075867374
20um_0.05r: 0.2739648159897403
20um_0.1r: 0.27208343753175374
20um_0.25r: 0.24031613551605854
40um_0.01r: 0.5007552738698986
40um_0.05r: 0.3768764189564579
40um_0.1r: 0.3273209769233719
40um_0.25r: 0.23046716467611322
60um_0.01r: 0.14076822156257876
60um_0.05r: 0.20272937787269804
60um_0.1r: 0.18902489955713853
60um_0.25r: 0.2029899155219636
80um_0.01r: 0.11340616245246334
80um_0.05r: 0.1539290978574756
80um_0.1r: 0.16113432978204667
80um_0.25r: 0.17073515994015145
100um_0.01r: 0.17347056714938394
100um_0.05r: 0.15086614402969853
100um_0.1r: 0.15571490640259417
100um_0.25r: 0.15259373341017973
120um_0.01r: 0.08361990869828036
120um_0.05r: 0.08389361729391803
120um_0.1r: 0.09083718628933875
120um_0.25r: 0.0934384082452056
140um_0.01r: 0.2205242479262377
140um_0.05r: 0.21791859677662082
140um_0.1r: 0.21725157817281265
140um_0.25r: 0.23023167051582052
```

In [ ]:

In [14]:
```
#check the file scale bar of chipview
import cv2
file_path = os.path.join(folder, file_names[0])
img = cv2.imread('/Users/qingjunwang/Documents/image files/speckle 2 study/E
print(img.shape)
#20um
#(2056/44/8)*30
(2464/21.2/20)*30
```

```
(2056, 2464)
```

Out[14]: 174.33962264150944

In [33]:
```
file_names
```

Out[33]:
```
array(['EPIC-FL1_D0C0_NAP-8um-P20-0r_Green__X0.000_Y0.000_Z-0.450_Exp10000
0.png',
       'EPIC-FL1_D0C0_NAP-8um-P20-0r_Green__X0.000_Y0.000_Z-0.500_Exp10000
0.png',
       'EPIC-FL1_D0C0_NAP-8um-P20-0r_Green__X0.000_Y0.000_Z-1.350_Exp10000
0.png',
       'EPIC-FL1_D0C0_NAP-8um-P20-0r_Green__X0.000_Y0.000_Z-1.200_Exp10000
0.png',
       'EPIC-FL1_D0C0_NAP-8um-P20-0r_Green__X0.000_Y0.000_Z-1.450_Exp10000
0.png',
       'EPIC-FL1_D0C0_NAP-8um-P20-0r_Green__X0.000_Y0.000_Z-1.500_Exp10000
0.png',
       'EPIC-FL1_D0C0_NAP-8um-P20-0r_Green__X0.000_Y0.000_Z-0.350_Exp10000
0.png',
       'EPIC-FL1_D0C0_NAP-8um-P20-0r_Green__X0.000_Y0.000_Z-0.200_Exp10000
0.png',
       'EPIC-FL1_D0C0_NAP-8um-P20-0r_Green__X0.000_Y0.000_Z-1.550_Exp10000
0.png',
       'EPIC-FL1_D0C0_NAP-8um-P20-0r_Green__X0.000_Y0.000_Z-1.400_Exp10000
0.png',
       'EPIC-FL1_D0C0_NAP-8um-P20-0r_Green__X0.000_Y0.000_Z-0.250_Exp38261.
png',
       'EPIC-FL1_D0C0_NAP-8um-P20-0r_Green__X0.000_Y0.000_Z-0.300_Exp10000
0.png',
       'EPIC-FL1_D0C0_NAP-8um-P20-0r_Green__X0.000_Y0.000_Z-0.550_Exp10000
0.png',
       'EPIC-FL1_D0C0_NAP-8um-P20-0r_Green__X0.000_Y0.000_Z-0.400_Exp10000
0.png',
       'EPIC-FL1_D0C0_NAP-8um-P20-0r_Green__X0.000_Y0.000_Z-1.250_Exp10000
0.png',
       'EPIC-FL1_D0C0_NAP-8um-P20-0r_Green__X0.000_Y0.000_Z-1.300_Exp10000
0.png',
       'EPIC-FL1_D0C0_NAP-8um-P20-0r_Green__X0.000_Y0.000_Z-0.050_Exp10000
0.png',
       'EPIC-FL1_D0C0_NAP-8um-P20-0r_Green__X0.000_Y0.000_Z-0.100_Exp10000
0.png',
       'EPIC-FL1_D0C0_NAP-8um-P20-0r_Green__X0.000_Y0.000_Z-0.750_Exp10000
0.png',
       'EPIC-FL1_D0C0_NAP-8um-P20-0r_Green__X0.000_Y0.000_Z-0.600_Exp10000
0.png',
       'EPIC-FL1_D0C0_NAP-8um-P20-0r_Green__X0.000_Y0.000_Z-0.900_Exp10000
0.png',
       'EPIC-FL1_D0C0_NAP-8um-P20-0r_Green__X0.000_Y0.000_Z-0.850_Exp10000
0.png',
       'EPIC-FL1_D0C0_NAP-8um-P20-0r_Green__X0.000_Y0.000_Z-1.050_Exp10000
0.png',
       'EPIC-FL1_D0C0_NAP-8um-P20-0r_Green__X0.000_Y0.000_Z-1.100_Exp10000
0.png',
       'EPIC-FL1_D0C0_NAP-8um-P20-0r_Green__X0.000_Y0.000_Z-0.650_Exp10000
0.png',
       'EPIC-FL1_D0C0_NAP-8um-P20-0r_Green__X0.000_Y0.000_Z-0.700_Exp10000
0.png',
       'EPIC-FL1_D0C0_NAP-8um-P20-0r_Green__X0.000_Y0.000_Z-1.150_Exp10000
0.png',
       'EPIC-FL1_D0C0_NAP-8um-P20-0r_Green__X0.000_Y0.000_Z-1.000_Exp10000
0.png',
```

```
       'EPIC-FL1_D0C0_NAP-8um-P20-0r_Green__X0.000_Y0.000_Z-0.800_Exp10000
0.png',
       'EPIC-FL1_D0C0_NAP-8um-P20-0r_Green__X0.000_Y0.000_Z-0.950_Exp10000
0.png',
       'EPIC-FL1_D0C0_NAP-8um-P20-0r_Green__X0.000_Y0.000_Z-0.150_Exp10000
0.png',
       'EPIC-FL1_D0C0_NAP-8um-P20-0r_Green__X0.000_Y0.000_Z-0.000_Exp14639.
png'],
      dtype='<U71')
```

In [38]:
```
#find the files for shaking
focus_z=0
file_for_shaking = [
                    file_names for file_names in file_names if
                    'Z-0.000' in file_names or
                    'Z-0.200' in file_names or
                    'Z-0.400' in file_names or
                    'Z-0.600' in file_names or
                    'Z-0.800' in file_names
                     ]
file_for_shaking=np.array(file_for_shaking)
print(file_for_shaking)
```

```
['EPIC-FL1_D0C0_NAP-8um-P20-0r_Green__X0.000_Y0.000_Z-0.200_Exp100000.png'
 'EPIC-FL1_D0C0_NAP-8um-P20-0r_Green__X0.000_Y0.000_Z-0.400_Exp100000.png'
 'EPIC-FL1_D0C0_NAP-8um-P20-0r_Green__X0.000_Y0.000_Z-0.600_Exp100000.png'
 'EPIC-FL1_D0C0_NAP-8um-P20-0r_Green__X0.000_Y0.000_Z-0.800_Exp100000.png'
 'EPIC-FL1_D0C0_NAP-8um-P20-0r_Green__X0.000_Y0.000_Z-0.000_Exp14639.png']
```

In [ ]:
```
import cv2
import matplotlib.pyplot as plt
import numpy as np
from PIL import Image
from scipy.ndimage import shift


# Define the colorbar range
vmin = 0
vmax = 255

# # for shaker setup imaging
# radius = 129  # A30um
# Anum_r = ["A30", "A60", "A90", "A120"]
# radius_r = [radius, radius * 2, radius * 3, radius * 4]
# sampling_r = [100, 200, 300, 400]

# for chipview setup imaging
radius = 70  # A30um
Anum_r = ["A30", "A60", "A90"]
radius_r = [radius, radius * 2, radius * 3]
sampling_r = [100, 200, 300]

for i in file_for_shaking:
    for j in range(len(Anum_r)):
        img1 = cv2.imread(i, cv2.IMREAD_GRAYSCALE)
        img1 = img1.astype(np.float32)
```

```
        # Define the circular trajectory
        angle = np.linspace(0, 2 * np.pi, sampling_r[j])
        # Initialize an array to store the sum of the images
        sum_img = np.zeros_like(img1)

        # Move the image in a circular trajectory and sum the images
        for a in angle:
            x_shift = radius_r[j] * np.cos(a)
            y_shift = radius_r[j] * np.sin(a)
            shifted_img = shift(img1, [x_shift, y_shift], mode="wrap")
            if a % 10 == 0:
                print(f"x_shift, yshift={x_shift:.2f},{y_shift:.2f}")
            sum_img = np.add(sum_img, shifted_img)

        # Calculate the average of the images
        avg_img = sum_img / sampling_r[j]

        # Create a figure with 1 row and 2 columns
        fig, axs = plt.subplots(nrows=len(Anum_r), ncols=len(file_for_shakir

        # Display img1 in the first subplot
        im1 = axs[0].imshow(img1, cmap="gray", vmin=vmin, vmax=vmax)
        axs[0].set_title("orginal image")
        fig.colorbar(im1, ax=axs[0])
        axs[0].grid(False)  # Turn off the grid

        # Display the average of the images in the second subplot
        im2 = axs[1].imshow(avg_img, cmap="gray", vmin=vmin, vmax=vmax)
        axs[1].set_title("Average of images")
        fig.colorbar(im2, ax=axs[1])
        axs[1].grid(False)  # Turn off the grid

        # Show the plot
        plt.show()
        # cv2.imwrite('/home/qingjunwang/post'+i+'-100points-'+Anum+'um.png'

        # normalize
        max_value = avg_img.max()
        # Normalize the image to the range 0-1
        scale = 255 / 2 / max_value
        # Scale the image to the range 0-255
        scaled_img1 = avg_img * scale

        Image.fromarray(scaled_img1.astype("uint8")).save(
            "/home/qingjunwang/"
            + i
            + "-"
            + str(sampling_r[j])
            + "points-"
            + Anum_r[j]
            + "um-normalize.png"
        )

        print(i + "normalize.png")
        # Display the normalized image
```

```
        plt.imshow(scaled_img1, cmap="gray")
        plt.show()
```

In [ ]:
```
from scipy.ndimage import rotate
import matplotlib.pyplot as plt
import numpy as np
from PIL import Image


# Load the image
img1 = io.imread(filenames_r[0])

img1_array = np.array(img1)
# Find the minimum and maximum pixel values
min_value = img1_array.min()
max_value = img1_array.max()
# Normalize the image to the range 0-1
normalized_img1 = (img1_array - min_value) / (max_value - min_value)
# Scale the image to the range 0-255
scaled_img1 = (normalized_img1 * 255).astype(np.uint8)
Image.fromarray(scaled_img1.astype('uint8')).save(paths_r[0]+"post0um.png")

# Display the normalized image
plt.imshow(scaled_img1, cmap='gray')
plt.show()
```

Differet driving current v.s. contrast/spectrum

In [3]:
```
import pandas as pd

def read_spectrum_csv(file_path):
    # Open the file and read lines
    with open(file_path, 'r') as file:
        lines = file.readlines()
    # Find the start of the trace data
    start_index = 0
    for i, line in enumerate(lines):
        if '[TRACE DATA]' in line:
            start_index = i + 1
            break

    # Extract trace data
    trace_data = lines[start_index:]
    # Create a DataFrame from the trace data
    data = {
        'Wavelength': [],
        'Intensity': []
    }

    for line in trace_data:
        if line.strip():  # Ensure the line is not empty
            # Try splitting by tab, space, or comma
            parts = line.replace(',', ' ').split()
            if len(parts) == 2:
```

```
                try:
                    wavelength, intensity = parts
                    data['Wavelength'].append(float(wavelength))
                    data['Intensity'].append(float(intensity))
                except ValueError as e:
                    print(f"Error parsing line: {line.strip()} - {e}")

    df = pd.DataFrame(data)
    if df.empty:
        print("No data extracted. Check the file format and delimiter.")
    return df

# Example usage
file_path = '/Users/qingjunwang/Documents/image files/Spectrum_contrast/Spec
df = read_spectrum_csv(file_path)
df
```

Out[3]:

	Wavelength	Intensity
0	516.900	1.448000e-06
1	516.902	3.671000e-07
2	516.904	3.929000e-08
3	516.906	1.539000e-06
4	516.908	1.267000e-06
...	...	...
2496	521.892	4.234000e-06
2497	521.894	8.331000e-07
2498	521.896	2.570000e-06
2499	521.898	9.183000e-08
2500	521.900	8.537000e-07

2501 rows × 2 columns

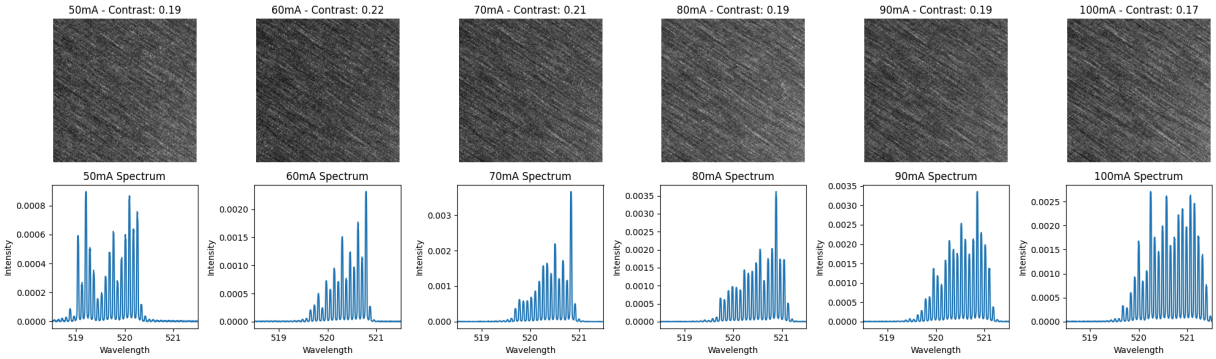
In [7]:

```
import cv2
import pandas as pd
import matplotlib.pyplot as plt
import glob
import os
import re

import cv2

def calculate_contrast(image):
    contrast = image.std() / image.mean()
    return contrast

def extract_power_value(filename, pattern):
    match = re.search(pattern, filename)
```

```
    if match:
        return match.group(1)
    return None

# Load all images from the folder
image_files = glob.glob('/Users/qingjunwang/Documents/image files/Spectrum_c
image_data = {}

for image_file in image_files:
    image = cv2.imread(image_file)
    image = image[:, :, 1]
    image= main.square_crop_img_upperleft(image,800)
    image = main.normalize_255(image)
    power_key = extract_power_value(image_file, r"(\d+)mA")
    if power_key:
        contrast = calculate_contrast(image)
        image_data[power_key] = (image, contrast)

# Load all CSV files from the folder
csv_files = glob.glob('/Users/qingjunwang/Documents/image files/Spectrum_cor
csv_data = {}

for csv_file in csv_files:
    power_key = extract_power_value(csv_file, r"(\d+)mW")
    if power_key:
        csv_data[power_key] = read_spectrum_csv(csv_file)
```

In [8]:

```
sorted_image_keys = sorted(image_data.keys(), key=int)
sorted_csv_keys = sorted(csv_data.keys(), key=int)
# Plotting
fig, axs = plt.subplots(2, max(len(image_data), len(csv_data)), figsize=(20,

for i, key in enumerate(sorted_image_keys):
    image, contrast = image_data[key]
    axs[0, i].imshow(image, cmap='gray')
    axs[0, i].set_title(f"{key}mA - Contrast: {contrast:.2f}")
    axs[0, i].axis('off')
for i, key in enumerate(sorted_csv_keys):
    df = csv_data[key]
    axs[1, i].plot(df['Wavelength'], df['Intensity'])
    axs[1, i].set_title(f"{key}mA Spectrum")
    axs[1, i].set_xlabel('Wavelength')
    axs[1, i].set_ylabel('Intensity')
    axs[1, i].set_xlim(518.5, 521.5)
plt.tight_layout()
plt.show()
```


50mA - Contrast: 0.19
60mA - Contrast: 0.22
70mA - Contrast: 0.21
80mA - Contrast: 0.19
90mA - Contrast: 0.19
100mA - Contrast: 0.17
50mA Spectrum
60mA Spectrum
70mA Spectrum
80mA Spectrum
90mA Spectrum
100mA Spectrum
Intensity
Wavelength

In [ ]: