

## Tugas TPCB 2

### Image Enhancement and Filtering

#### **Challenge :**

1. Try to implement the classic methods of histogram equalization and contrast-limited adaptive histogram equalization (CLAHE) on the following images. Compare and evaluate the quality of the medical image processing results with the original image.
2. Try implementing the filtering methods of Mean filter, Median filter, and Gaussian filter on the available images. Compare and evaluate the quality of the medical image processing results before and after being filtered.

#### **Solve :**

##### **1. Implementing Histogram Equalization and CLAHE**

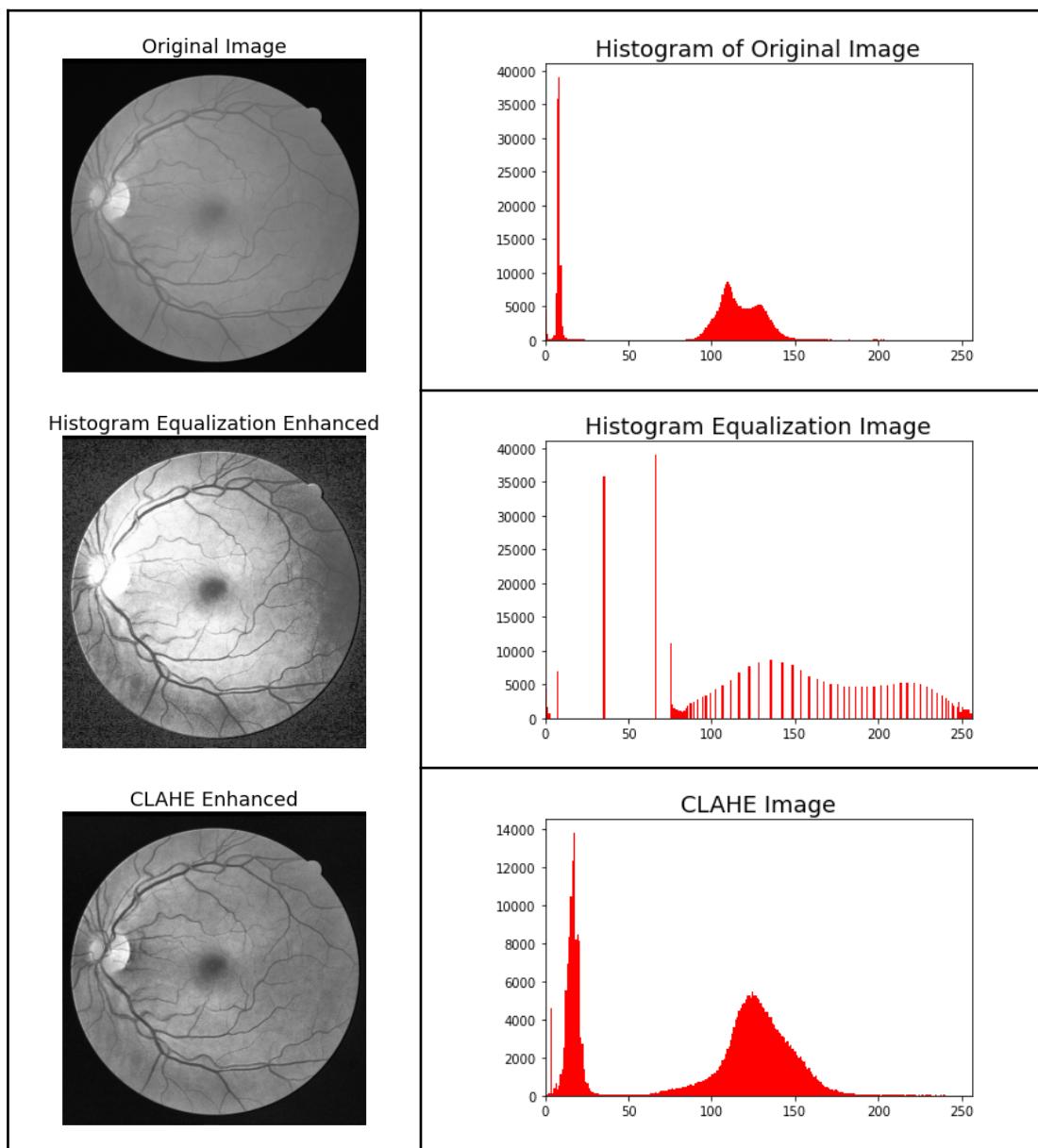
Histogram equalization (HE) dan Contrast Limited Adaptive Histogram Equalization (CLAHE) adalah teknik yang bertujuan untuk meningkatkan kontras citra dengan cara meratakan pendistribusian warna citra. Perbedaan keduanya terletak pada cara distribusi intensitas piksel citranya. Dalam HE, distribusi intensitas piksel dalam citra dibuat merata di sepanjang sumbu histogram. Sebagai hasilnya, histogram equalization dapat meningkatkan kontras gambar, tetapi mungkin tidak menghasilkan hasil yang optimal dalam kasus di mana gambar memiliki daerah dengan variasi kontras yang signifikan. Histogram equalization juga dapat menyebabkan over-enhancement di beberapa bagian gambar dan dapat membuat gambar terlihat tidak alami.

Sedangkan, CLAHE (Contrast Limited Adaptive Histogram Equalization) adalah variasi dari histogram equalization adaptif yang memungkinkan untuk mengatur nilai ambang agar tidak menghasilkan pergeseran terlalu besar pada intensitas piksel. CLAHE membagi gambar ke dalam beberapa blok kecil dan melakukan histogram equalization pada setiap blok secara independen. Sebagai hasilnya, CLAHE dapat meningkatkan kontras lokal pada gambar tanpa mengorbankan detail yang ada pada daerah kontras rendah. Oleh karena itu, CLAHE cenderung menghasilkan citra yang lebih baik dibandingkan HE pada gambar yang memiliki kontras yang signifikan di berbagai bagian citra.

Dalam mengimplementasikan CLAHE, kita perlu menentukan nilai batas kontras dan ukuran blok-blok kecil terlebih dahulu. Batas kontras ini berfungsi untuk menentukan seberapa banyak perubahan kontras yang akan dilakukan pada setiap blok citra sehingga kita bisa menghindari terjadinya *over-enhancement* yang signifikan. Nilai batas kontras yang terlalu rendah dapat menghasilkan citra yang tidak jauh berbeda dengan citra asli, sedangkan nilai batas kontras yang terlalu tinggi dapat menghasilkan citra yang terlalu kontras dan banyak derau.

HE dapat digunakan untuk meningkatkan kualitas citra untuk mempermudah identifikasi seperti identifikasi wajah, identifikasi plat nomor, dll. Sedangkan CLAHE dapat digunakan untuk meningkatkan kualitas citra pada area dengan variasi intensitas piksel yang besar, seperti citra radiografi, citra medis, dan citra satelit.

Berikut merupakan pengolahan citra untuk meningkatkan kontras beserta grafik distribusi histogram menggunakan metode Histogram Equalization dan CLAHE dengan nilai clipLimit 2.0 dan tileSize 8,8.

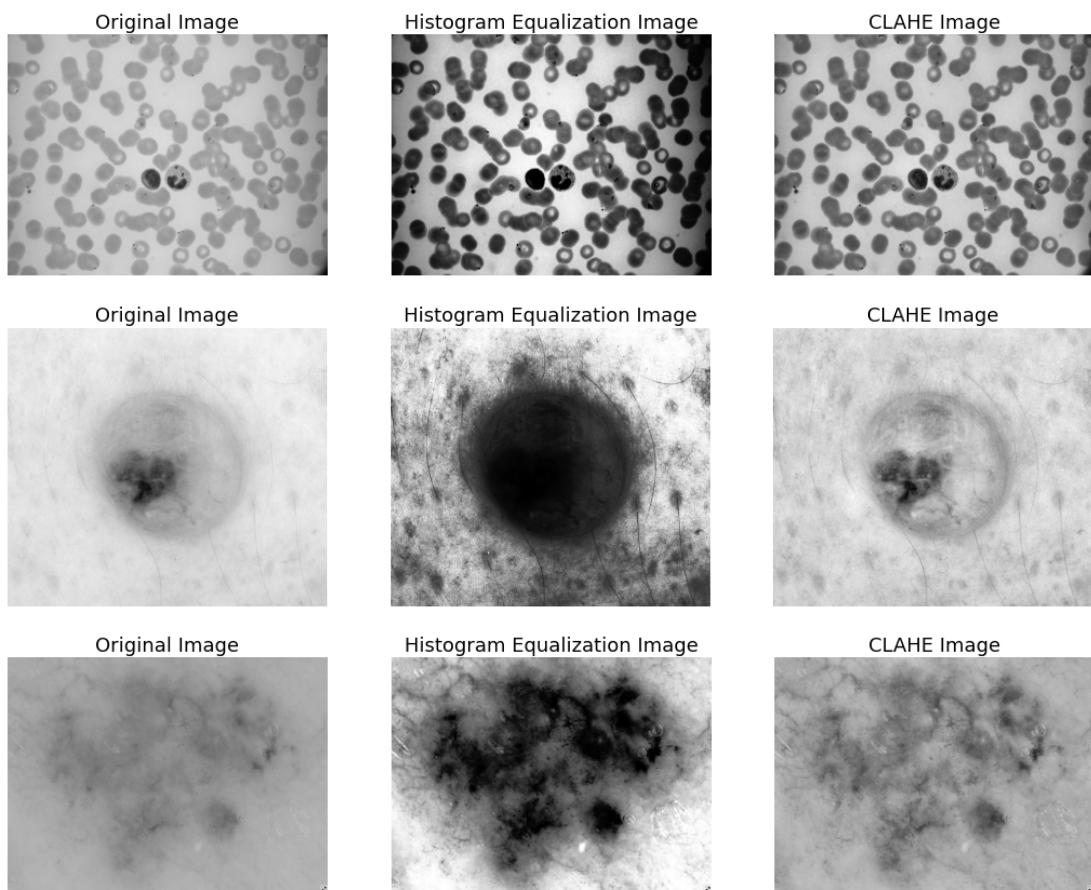


Berdasarkan hasil pengolahan, citra dengan kontras yang diolah menggunakan HE dan CLAHE memiliki detail yang lebih jelas dilihat. Terlihat pada grafik histogram, citra asli memiliki distribusi warna yang tidak merata sehingga detail kontras sulit dibedakan. Pada grafik histogram HE, distribusi intensitas kontras dibuat merata sepanjang sumbu x. Pada grafik histogram

CLAHE, memiliki grafik yang mirip seperti grafik histogram citra asli tetapi lebih merata dan nilainya lebih rendah. Hal ini disebabkan karena CLAHE menghasilkan distribusi intensitas piksel yang lebih merata secara lokal pada blok-blok kecil dalam gambar.

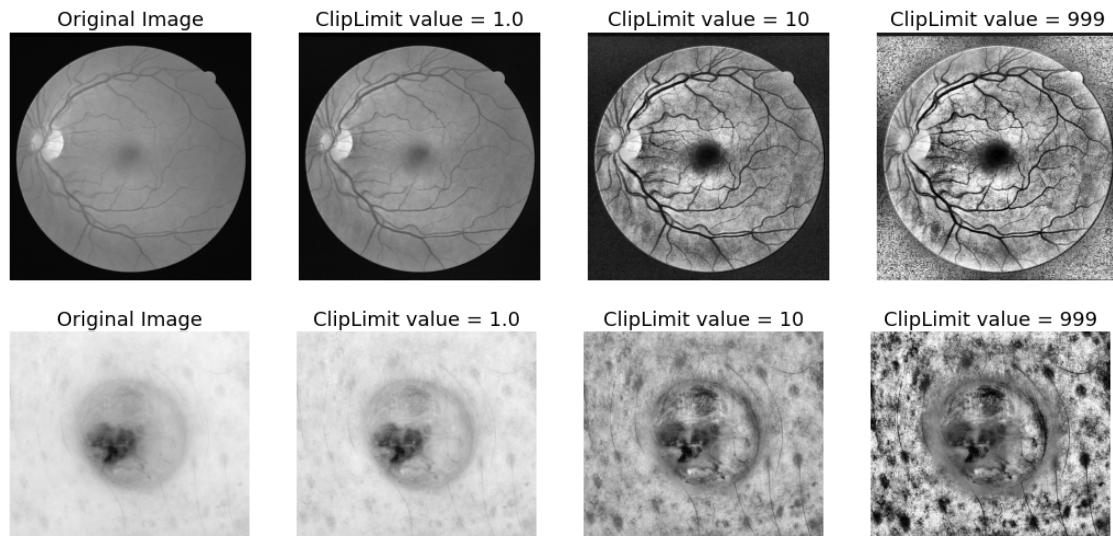
Dengan demikian, Histogram gambar asli memiliki puncak yang tajam pada beberapa intensitas piksel tertentu, histogram hasil histogram equalization memiliki distribusi intensitas piksel yang lebih merata secara global, serta histogram hasil CLAHE memiliki distribusi intensitas piksel yang lebih merata secara lokal pada blok-blok kecil dalam gambar.

Berikut perbandingan antara citra asli, citra hasil HE, dan citra hasil CLAHE pada gambar sample lainnya.



Seperti yang dapat kita lihat, citra hasil HE memiliki kontras warna pada pixel-pixel yang berbeda secara signifikan sehingga terlihat tidak alami dan detail pada kontras warna rendah tidak terlihat, sedangkan citra hasil CLAHE memiliki kontras warna yang lebih merata sehingga detail pada kontras warna rendah masih terlihat.

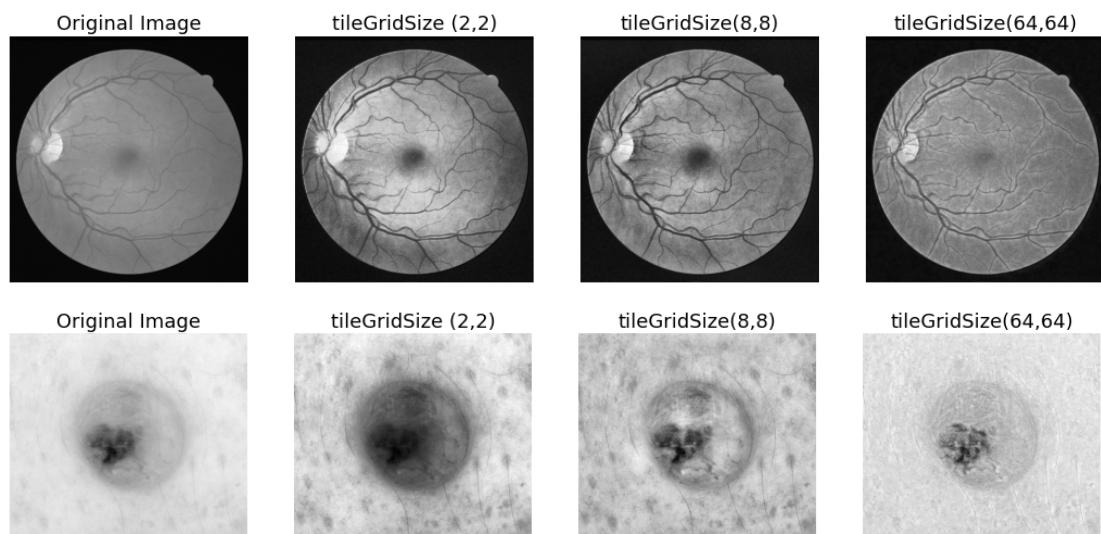
Pada CLAHE, terdapat parameter `clipLimit` dan `tileGridSize`. `clipLimit` digunakan untuk menentukan ambang batas kontras yang digunakan untuk membatasi distribusi intensitas piksel. Sedangkan `tileGridSize` digunakan untuk membagi citra menjadi blok-blok kecil yang akan dilakukan HE secara independen. Berikut merupakan eksperimen untuk melihat perbedaan yang dihasilkan ketika nilai parameter `clipLimit` ini diubah-ubah



Dapat kita lihat, nilai clipLimit yang terlalu rendah menghasilkan perbedaan kontras yang tidak signifikan dengan citra asli, sedangkan jika nilai clipLimit terlalu tinggi menghasilkan *over-enhancement* dimana kontras lokal terlalu tinggi, sehingga menghasilkan artefak derau yang tidak ada dalam citra asli.

Dengan demikian, untuk menghasilkan citra yang jelas dan tanpa artefak yang diinginkan kita perlu menentukan nilai clipLimit yang tepat. Nilai clipLimit yang optimal bervariasi tergantung pada karakteristik citra yang diproses sehingga kita mungkin perlu melakukan percobaan berulang kali untuk menemukan nilai yang optimal tersebut.

Parameter kedua yang dimiliki CLAHE adalah tileGridSize. Berikut merupakan perbedaan yang dihasilkan ketika nilai parameter tersebut diubah-ubah



Dapat kita lihat, ukuran blok yang lebih kecil meningkatkan detil pada citra yang dihasilkan, tetapi memerlukan waktu komputasi yang lebih lama. Sedangkan, penggunaan blok yang terlalu besar menghasilkan citra yang

kasar dan tidak detail, meskipun waktu komputasinya lebih cepat. menghasilkan citra yang kasar dan tidak detail.

Dengan demikian, untuk menghasilkan citra yang optimal mungkin memerlukan perulangan untuk mencari nilai parameter tileSize yang paling optimal pula.

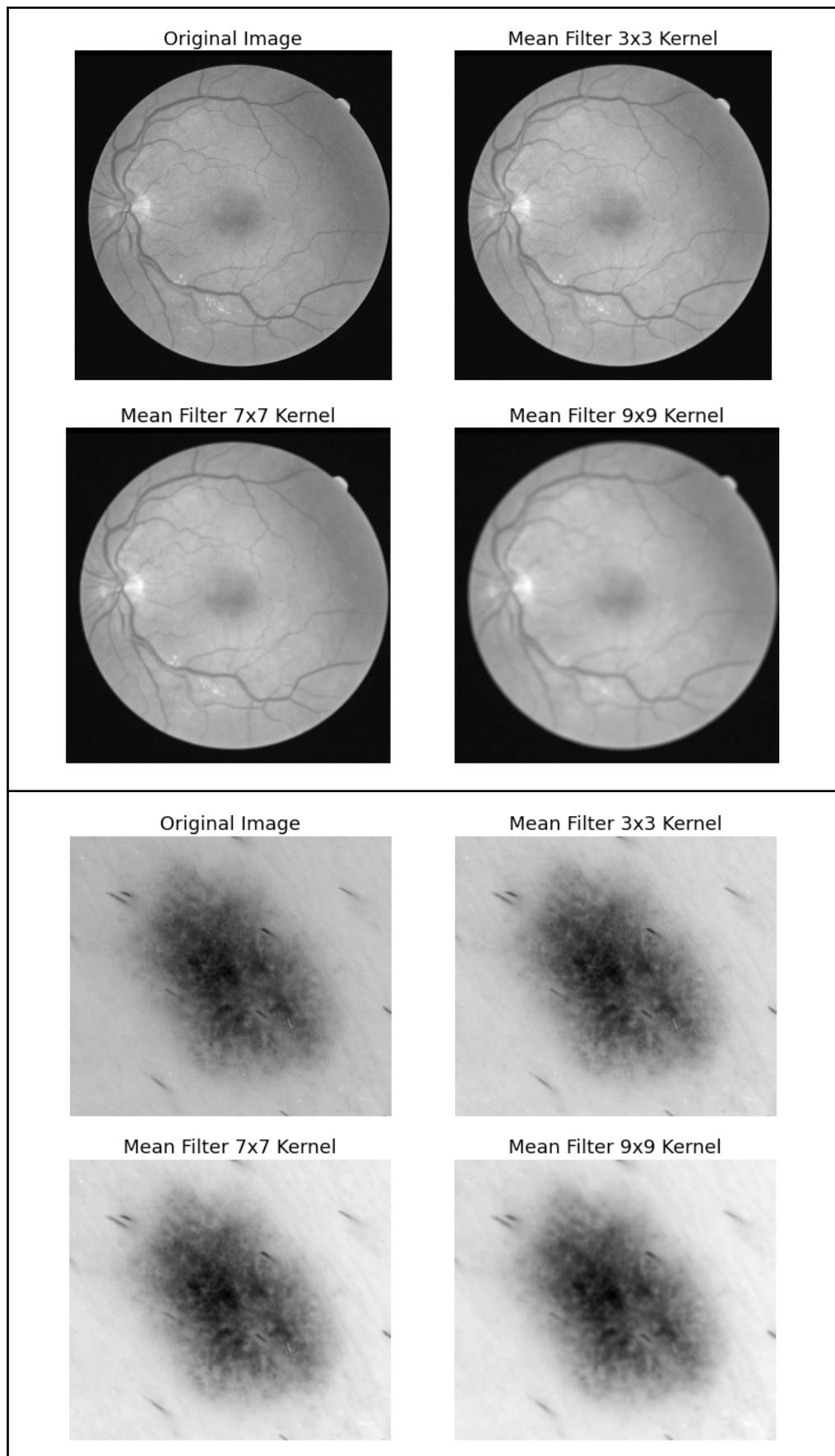
## 2. Implementing the filtering methods of Mean filter, Median filter, and Gaussian filter

### a. Mean Filter

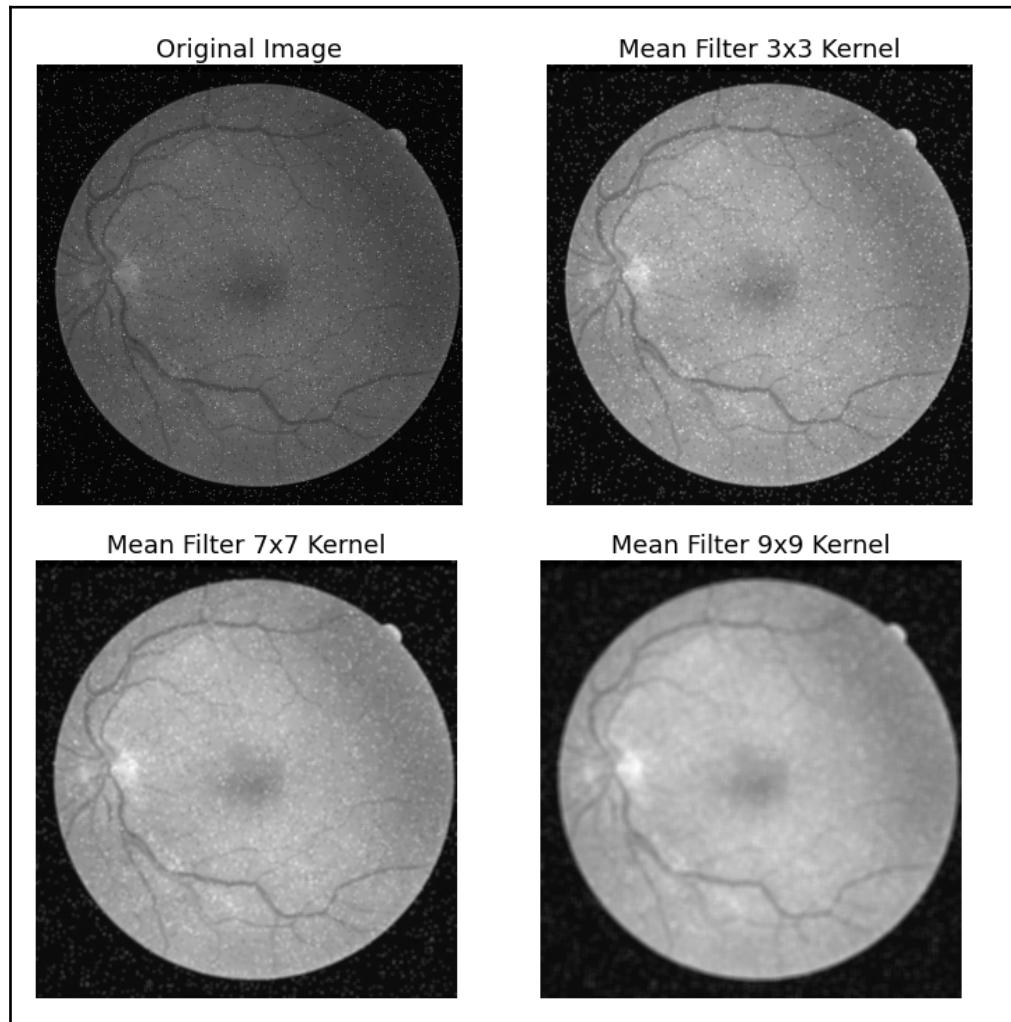
Mean filter adalah filter yang digunakan untuk mengurangi derau pada citra digital. Mean filter menerapkan konsep konvolusi. Konvolusi pada citra 2D adalah proses untuk memperoleh nilai piksel baru dengan memperhitungkan nilai piksel asal dan nilai piksel tetangga (neighbourhood). Untuk melakukan konvolusi, diperlukan sebuah matriks konvolusi atau yang disebut kernel dengan ukuran tertentu. Biasanya, kernel ini memiliki jumlah baris dan kolom yang sama serta merupakan bilangan ganjil seperti 3x3, 5x5, dan 7x7. Kernel yang digunakan untuk *mean filter* disebut *uniform blur* yang merupakan kernel dengan elemen yang seragam dengan jumlah dari semua elemennya sama dengan satu. Berikut merupakan contoh kernel uniform blur berukuran 3x3

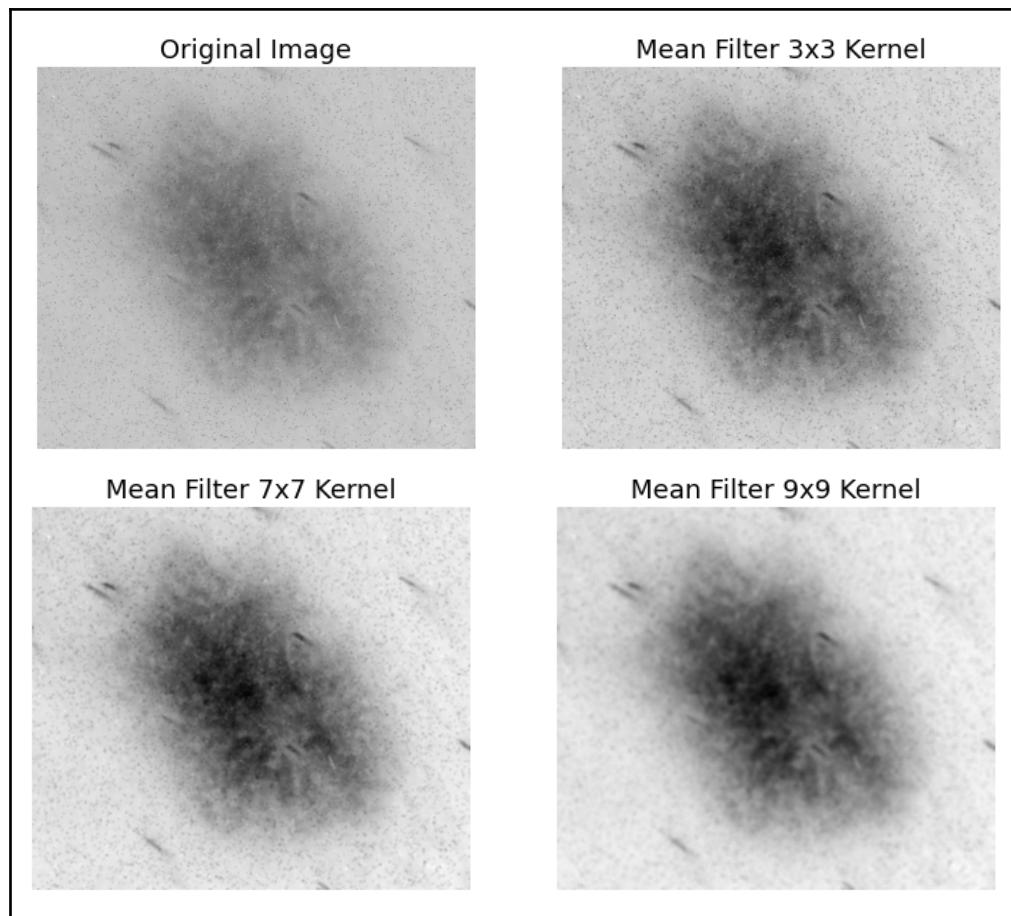
$$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Prinsip dasar proses konvolusi adalah dengan melakukan operasi *dot product* pada sebuah kernel dengan citra asli. Semakin besar ukuran kernel, semakin banyak piksel tetangga yang dihitung untuk mendapatkan nilai rata-rata, sehingga citra yang dihasilkan akan terlihat lebih halus. Namun, semakin besar ukuran kernel, semakin lambat juga waktu komputasi. Sebaliknya, semakin kecil ukuran kernel, semakin sedikit piksel tetangga yang diperhitungkan sehingga menghasilkan citra yang semakin kasar. Oleh karena itu, ukuran kernel harus disesuaikan dengan karakteristik citra yang akan diolah dan tujuan dari pengolahan citra tersebut. Untuk membuktikan hal tersebut, dilakukan percobaan *mean filter* dengan besar kernel yang berbeda-beda.



Dilihat dari citra asli dengan citra yang melalui mean filter, citra yang sudah diolah memiliki gambar yang lebih halus dan buram. Sedangkan, Perbedaan antara kernel 3x3, 5x5, dan 7x7 terlihat jelas pada hasil *mean filter* diatas, semakin besar ukuran kernel, semakin halus dan buram citra yang dihasilkan. Akan tetapi, dari citra tersebut sulit untuk melihat performa filter dalam menghilangkan noise karena citra tersebut sudah cukup bersih. Untuk itu, dilakukan uji coba ulang dengan citra yang sama tetapi diberi noise salt&papper terlebih dahulu.

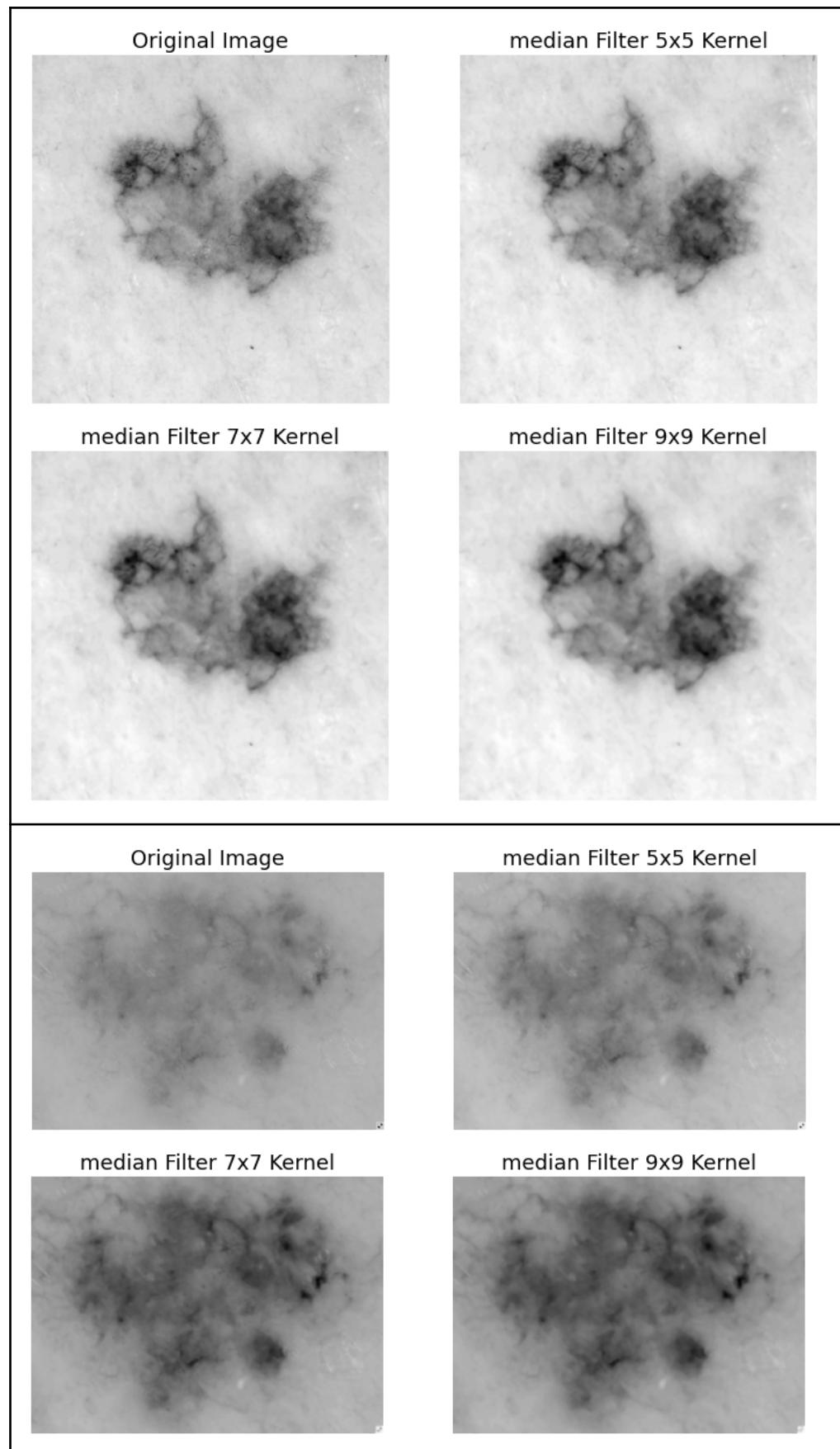




Dengan menambahkan noise pada citra, kita semakin mudah untuk membandingkan performa filter dalam menghilangkan noise. Terlihat pada kernel 3x3 noise masih terlihat dengan jelas, sedangkan pada kernel 9x9 noise-nya menjadi halus. Dengan demikian, terbukti bahwa semakin besar kernel yang digunakan, semakin halus noise yang terdapat pada citra.

b. Median Filter

Sama seperti *mean filter*, *Median filter* adalah teknik pengolahan citra yang digunakan untuk meningkatkan kualitas citra yang diproses dengan meminimalisir noise pada citra tersebut. *Median filter* ini bekerja dengan cara mengganti nilai piksel tertentu dengan nilai median dari piksel-piksel tetangga di sekitarnya. Hal ini menyebabkan perbedaan intensitas pada citra menjadi lebih halus dan detailnya tetap dipertahankan. Sehingga, Citra hasil pengolahan dengan median filter memiliki efek blur yang lebih halus dan tidak terlalu menghilangkan detail. Sama seperti sebelumnya, *Median Filter* menggunakan kernel dalam pengoperasiannya. Berikut merupakan hasil pengolahan menggunakan *Median Filter*.



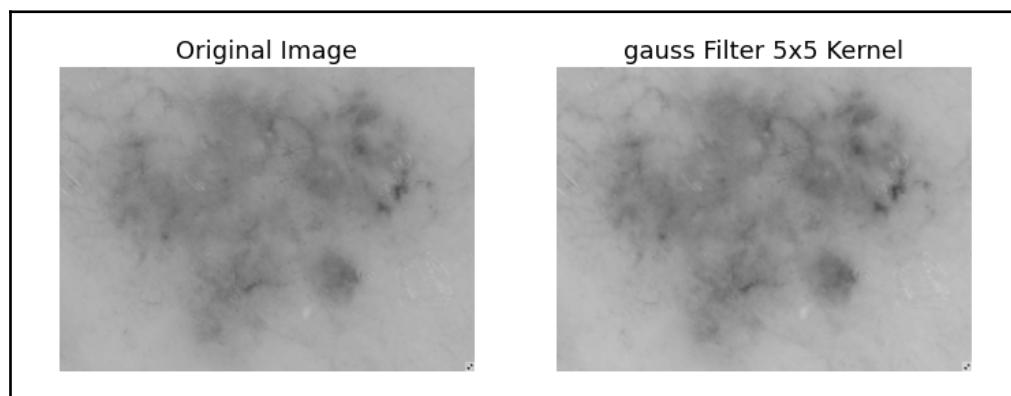
Terlihat pada citra-citra yang dihasilkan, citra yang melalui pengolahan median filter terlihat lebih halus dan kontrasnya semakin detail. Perbedaan nilai kernel berpengaruh terhadap tingkat derau yang dihilangkan. Apabila diperhatikan, citra yang diolah menggunakan kernel 9x9 terlihat lebih buram dan memiliki latar belakang sedikit lebih halus dari citra dengan kernel yang lebih kecil. Dengan demikian, semakin besar kernel, semakin besar juga performanya dalam menghilangkan noise. Akan tetapi, kernel yang terlalu besar dapat mengurangi detail pada citra.

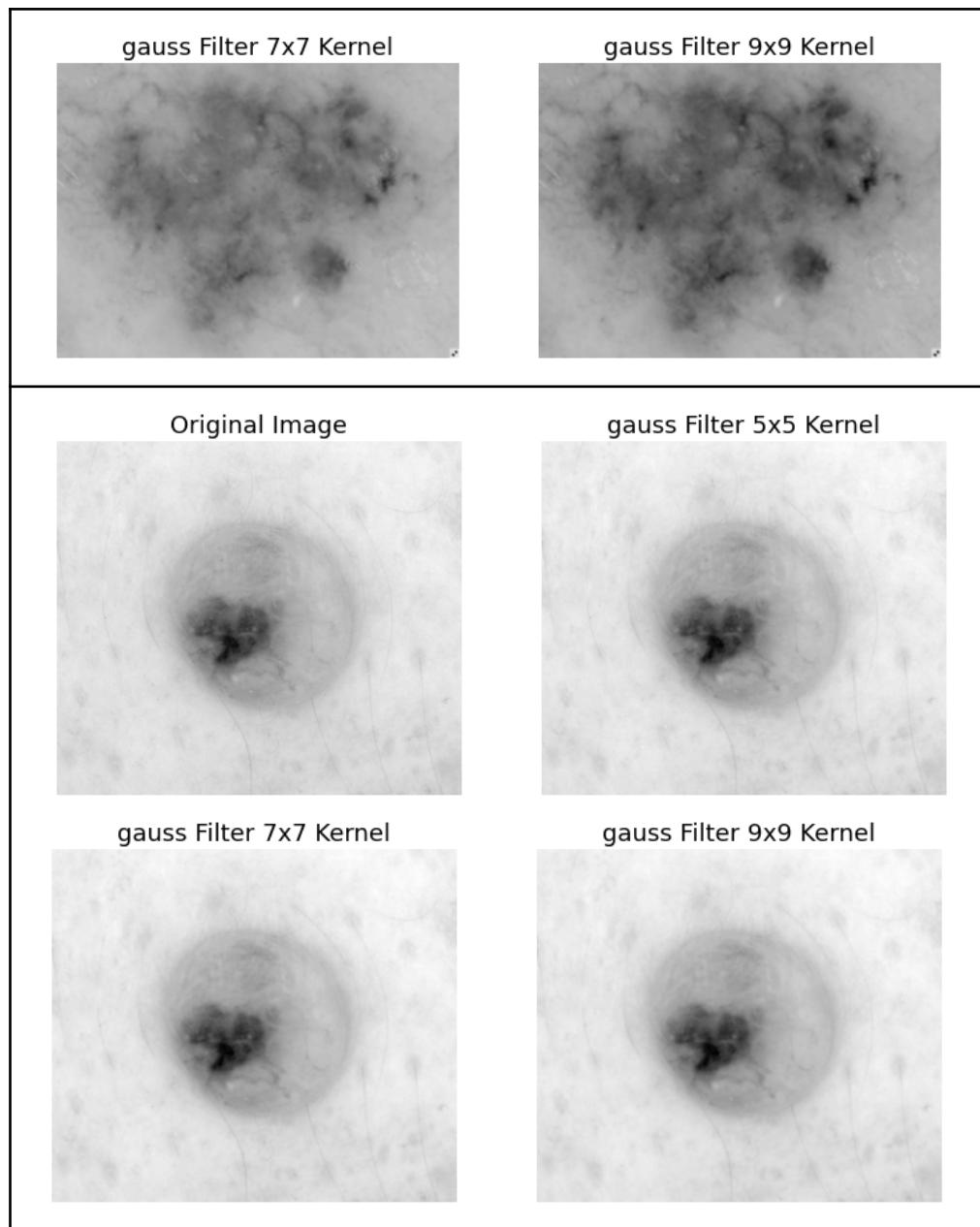
### c. Gaussian Filter

Gaussian filter memiliki algoritma yang sama dengan mean filter, yaitu dengan menghitung rerata piksel tetangga untuk mengganti nilai piksel tertentu. Perbedaannya terletak pada nilai atau bobot yang dimiliki oleh kernel. Berikut merupakan bobot kernel pada gaussian kernel

$$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Filter ini bekerja dengan cara mengambil nilai rata-rata bobot piksel-piksel pada citra sekitar titik yang sedang diolah, dengan bobot yang semakin besar di tengah kernel dan semakin kecil di tepi kernel. Hal ini menghasilkan *smoothing* yang lebih halus dan tanpa menghilangkan noise pada citra dibandingkan dengan mean filter. Berikut merupakan hasil pengolahan menggunakan Gaussian Filter



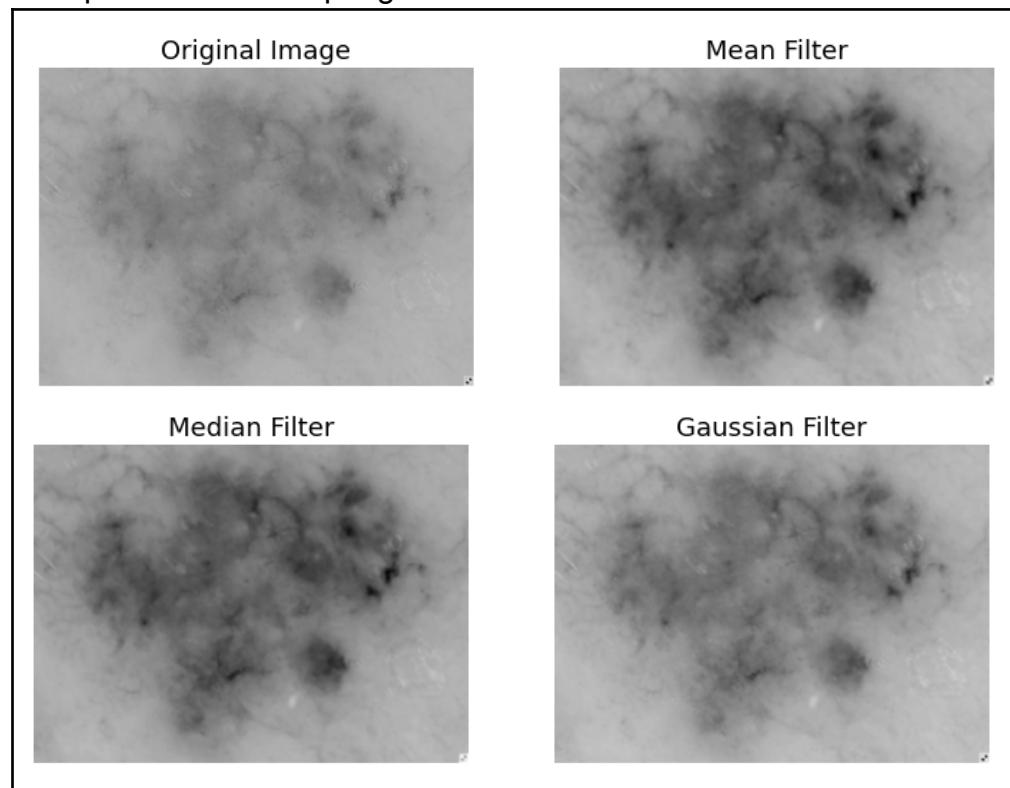


Terlihat citra yang sudah diolah dengan gaussian filter memiliki kontras yang sedikit lebih jelas dibandingkan dengan citra yang asli. Selain itu, citra hasil filtering ini juga terlihat lebih halus dan noisenya sedikit berkurang.

Perbedaan ukuran kernel terlihat pada tingkat kehalusan dan detail pada citra. Semakin besar ukuran kernel, semakin banyak pula piksel yang digunakan untuk menghitung nilai rerata pada titik yang diolah. Oleh karena itu, semakin besar ukuran kernel, akan menghasilkan citra yang semakin halus tetapi detailnya berkurang dan kurang tajam, seperti terlihat pada gambar hasil percobaan.

d. Perbandingan hasil mean filter, median filter, dan gaussian filter

Setiap algoritma filtering ini memiliki performa yang berbeda-beda. Untuk membandingkan dengan lebih mudah, berikut merupakan citra hasil pengolahan filter-filter tersebut.



Terlihat, dibandingkan citra asli, citra yang filter terlebih dahulu memiliki kontras dan garis tepi yang lebih jelas. Pada citra ini, urutan kontras dan garis tepi yang jelas terlihat pada citra hasil median filter, lalu mean filter, dan terakhir gaussian filter.

Sayyidan Muhamad Ikhsan  
20/460160/TK/50749

## **LAMPIRAN**

## **CODE AND RUNNING RESULT**

Link Google Colab :  
<https://colab.research.google.com/drive/1hrJe9OOcrTIZsVtpUt1qeGwiwmTVDv-b?usp=sharing>

## Assignment 2 : Image Enhancement and Filtering

Sayyidan Muhamad Ikhsan

20/460160/TK/50749

1. Try to implement the classic methods of histogram equalization and contrast-limited adaptive histogram equalization (CLAHE) on the following images. Compare and evaluate the quality of the medical image processing results with the original image (click this link).
2. Try implementing the filtering methods of Mean filter, Median filter, and Gaussian filter on the available images. Compare and evaluate the quality of the medical image processing results before and after being filtered.

### First Challenge : Implement Histogram Equalization and Contrast-Limited Adaptive Histogram Equalization (CLAHE)

```
#Librari
import cv2
import numpy as np
from google.colab.patches import cv2_imshow
import matplotlib.pyplot as plt

#fungsi hiseq ma clahe

def hiseq_clahe(img) :

    # Apply histogram equalization
    HisEq_img = cv2.equalizeHist(img)

    # Apply CLAHE
    clahe = cv2.createCLAHE(clipLimit=2.0, tileSize=(8,8))
    cl_img = clahe.apply(img)

    # Display the images

    fig, ax = plt.subplots(1, 3, figsize=(18,18))
    ax[0].set_title(f'Original Image', fontsize = 18)
    ax[0].imshow(img, cmap='gray')
    ax[0].set_axis_off()

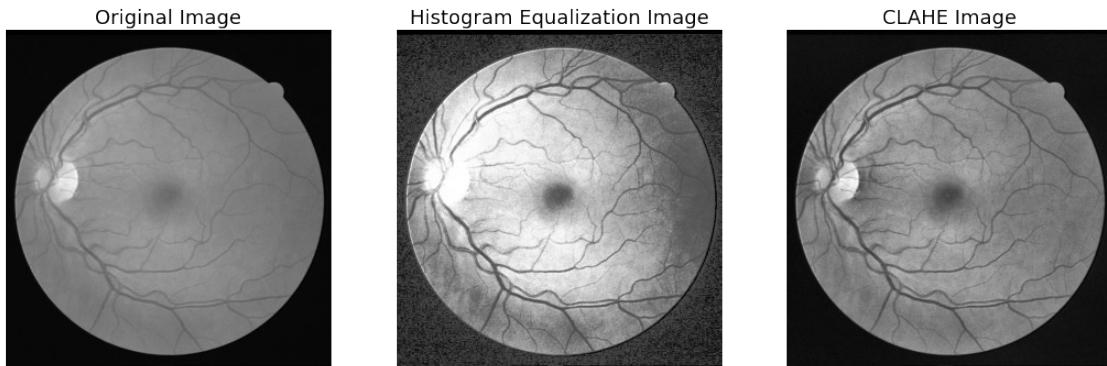
    ax[1].set_title(f'Histogram Equalization Image', fontsize = 18)
    ax[1].imshow(HisEq_img, cmap='gray')
    ax[1].set_axis_off()

    ax[2].set_title(f'CLAHE Image', fontsize = 18)
```

```
ax[2].imshow(cl_img, cmap='gray')
ax[2].set_axis_off()
```

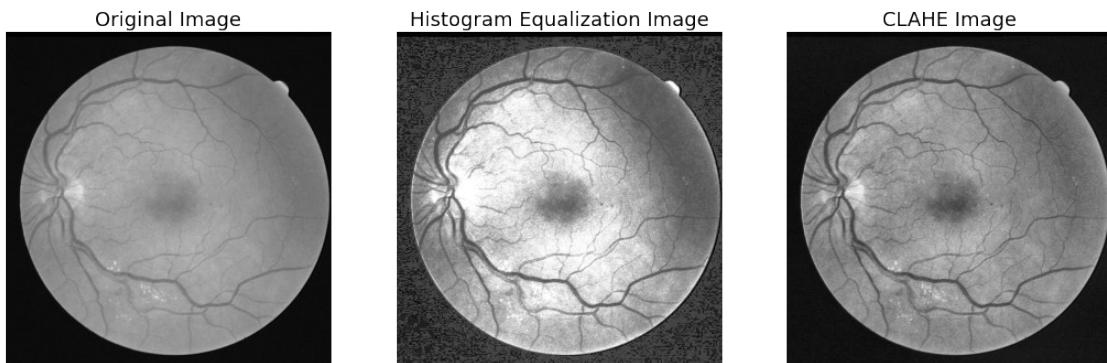
#### #Citra Pertama

```
img1 = cv2.imread('/content/drive/MyDrive/Tugas TPCB/Image
Data/01_test.jpg',0)
hiseq_clahe(img1)
```



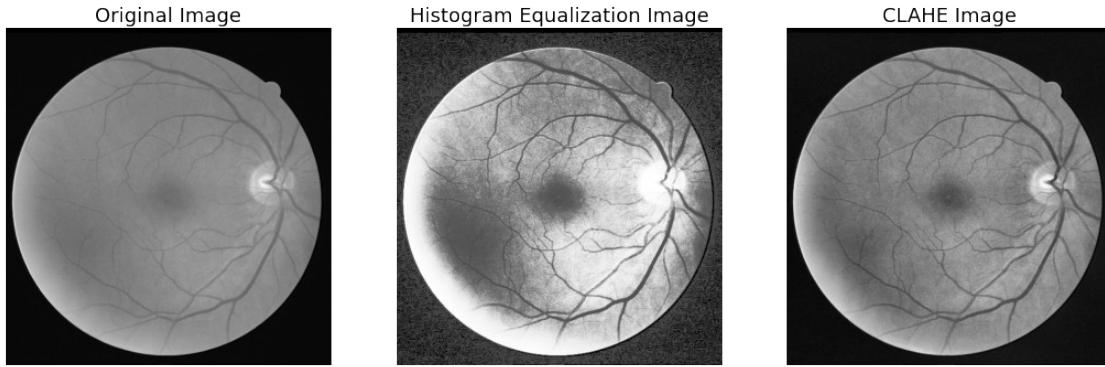
#### #Citra Kedua

```
img2 = cv2.imread('/content/drive/MyDrive/Tugas TPCB/Image
Data/03_test.jpg',0)
hiseq_clahe(img2)
```

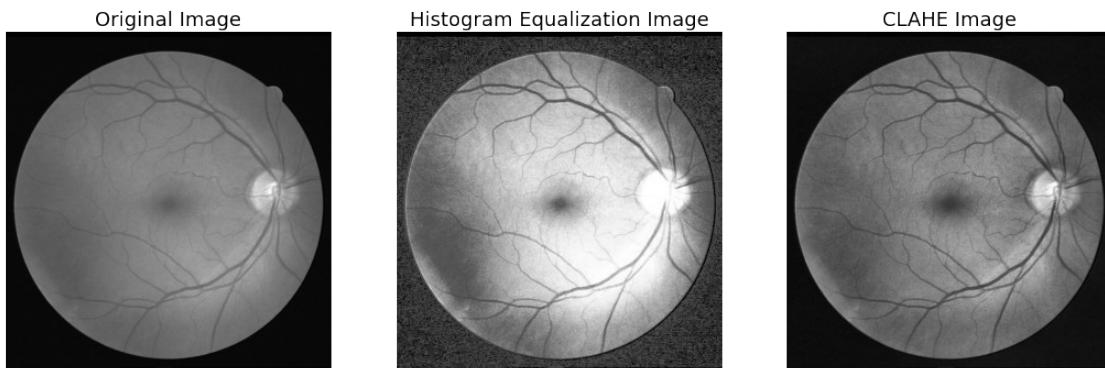


#### #Citra Ketiga

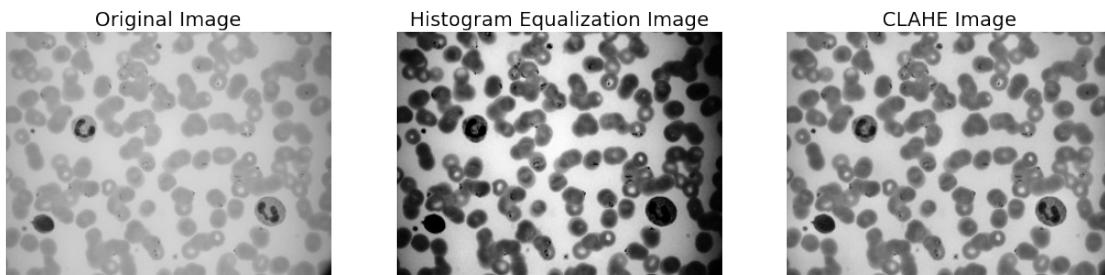
```
img3 = cv2.imread('/content/drive/MyDrive/Tugas TPCB/Image
Data/06_test.jpg',0)
hiseq_clahe(img3)
```



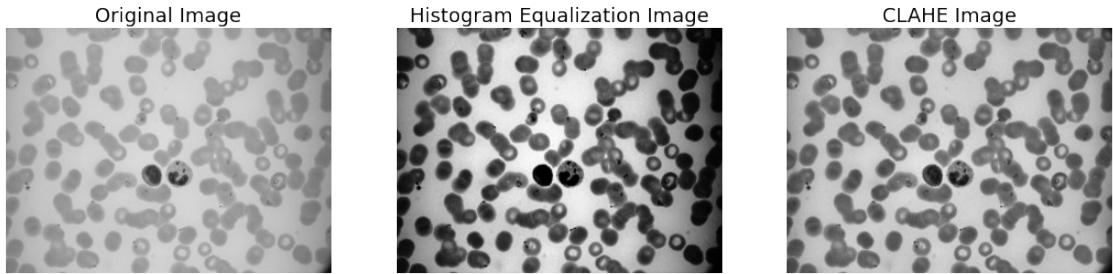
```
#citra keempat
img4 = cv2.imread('/content/drive/MyDrive/Tugas TPCB/Image
Data/10_test.jpg',0)
hiseq_clahe(img4)
```



```
#Citra Kelima
img5 = cv2.imread('/content/drive/MyDrive/Tugas TPCB/Image
Data/1307210661-0004-R.jpg',0)
hiseq_clahe(img5)
```

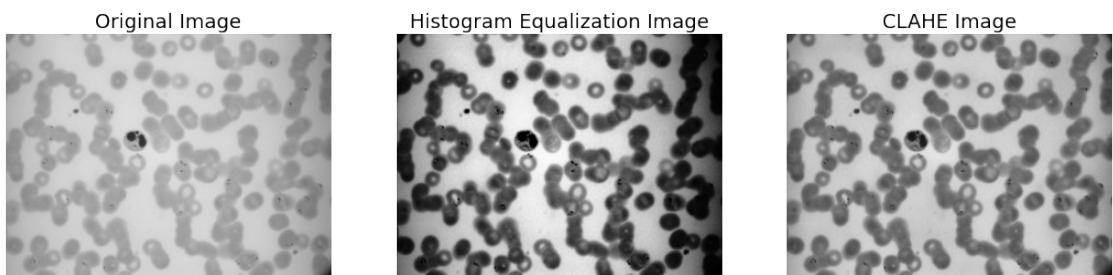


```
#Citra Keenam
img6 = cv2.imread('/content/drive/MyDrive/Tugas TPCB/Image
Data/1307210661-0005-R.jpg',0)
hiseq_clahe(img6)
```



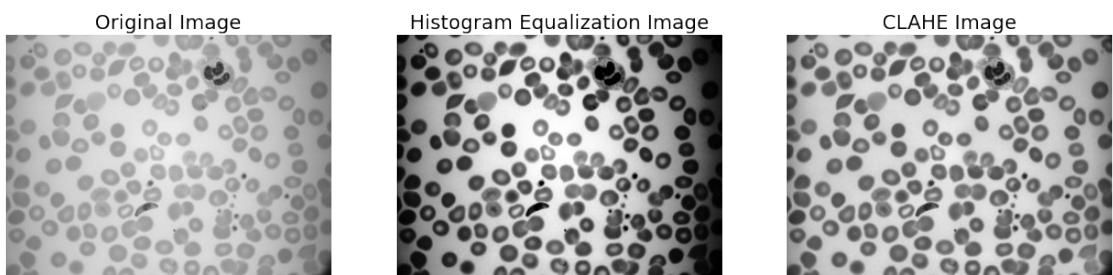
#Citra Ketujuh

```
img7 = cv2.imread('/content/drive/MyDrive/Tugas TPCB/Image Data/1307210661-0006-R.jpg',0)
hiseq_clahe(img7)
```



#Citra Kedelapan

```
img8 = cv2.imread('/content/drive/MyDrive/Tugas TPCB/Image Data/1603223711-0003-G_R.jpg',0)
hiseq_clahe(img8)
```

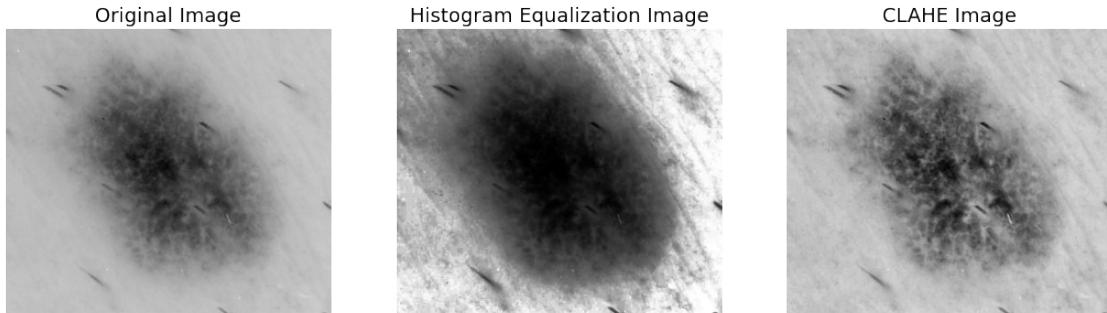


#citra kesembilan

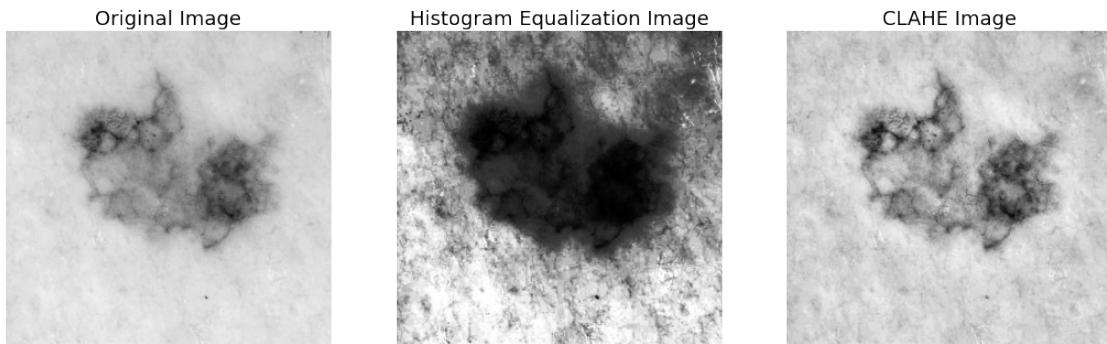
```
img9 = cv2.imread('/content/drive/MyDrive/Tugas TPCB/Image Data/ISIC_0000523.jpg',0)
hiseq_clahe(img9)
```



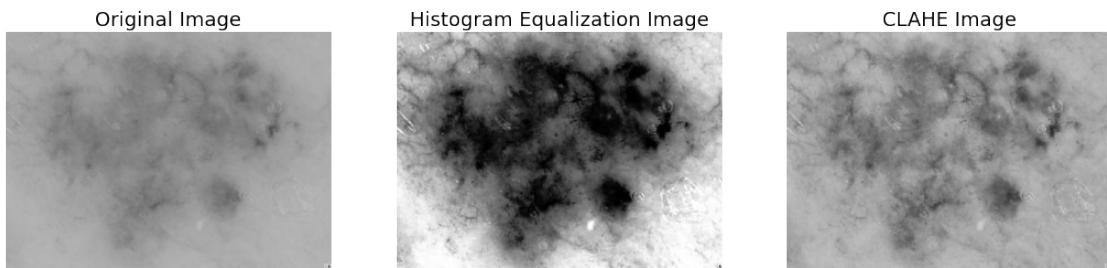
```
#citra kesepuluh
img10 = cv2.imread('/content/drive/MyDrive/Tugas TPCB/Image
Data/ISIC_0000546.jpg',0)
hiseq_clahe(img10)
```



```
#citra kesebelas
img11 = cv2.imread('/content/drive/MyDrive/Tugas TPCB/Image
Data/ISIC_0000548.jpg',0)
hiseq_clahe(img11)
```



```
#citra keduabelas
img12 = cv2.imread('/content/drive/MyDrive/Tugas TPCB/Image
Data/ISIC_0000551.jpg',0)
hiseq_clahe(img12)
```



## Implement HE and CLAHE with Histogram Graph

```
def HE_CL(img) :
```

```
# Calculate and display the histogram of the original image
hist,bins = np.histogram(img.flatten(),256,[0,256])
```

```

plt.hist(img.flatten(),256,[0,256], color = 'r')
plt.xlim([0,256])
plt.title('Histogram of Original Image',fontsize=18)
plt.show()

# Perform Histogram Equalization
img_he = cv2.equalizeHist(img)

# Calculate and display the histogram of the enhanced image
hist,bins = np.histogram(img_he.flatten(),256,[0,256])
plt.hist(img_he.flatten(),256,[0,256], color = 'r')
plt.xlim([0,256])
plt.title('Histogram Equalization Image',fontsize=18)
plt.show()

# Perform CLAHE
clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(16,16))
img_cl = clahe.apply(img)

# Calculate and display the CLAHE image
hist,bins = np.histogram(img_cl.flatten(),256,[0,256])
plt.hist(img_cl.flatten(),256,[0,256], color = 'r')
plt.xlim([0,256])
plt.title('CLAHE Image',fontsize=18)
plt.show()

# Display the original and enhanced images
fig, ax = plt.subplots(3, 1, figsize=(18,18))
ax[0].set_title(f'Original Image', fontsize = 18)
ax[0].imshow(img, cmap='gray')
ax[0].set_axis_off()

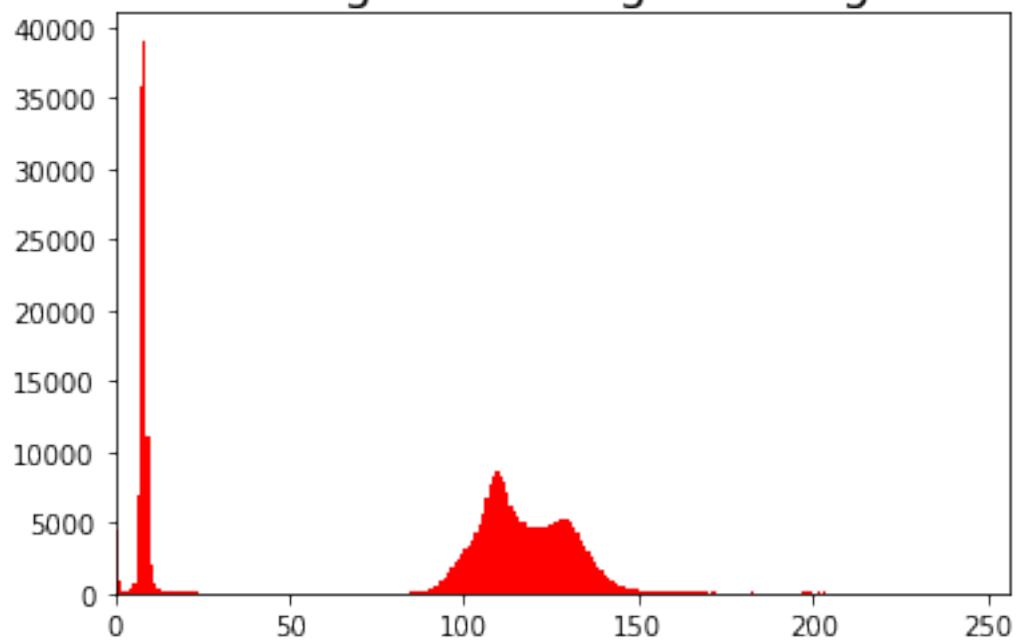
ax[1].set_title(f'Histogram Equalization Enhanced', fontsize = 18)
ax[1].imshow(img_he, cmap='gray')
ax[1].set_axis_off()

ax[2].set_title(f'CLAHE Enhanced', fontsize = 18)
ax[2].imshow(img_cl, cmap='gray')
ax[2].set_axis_off()

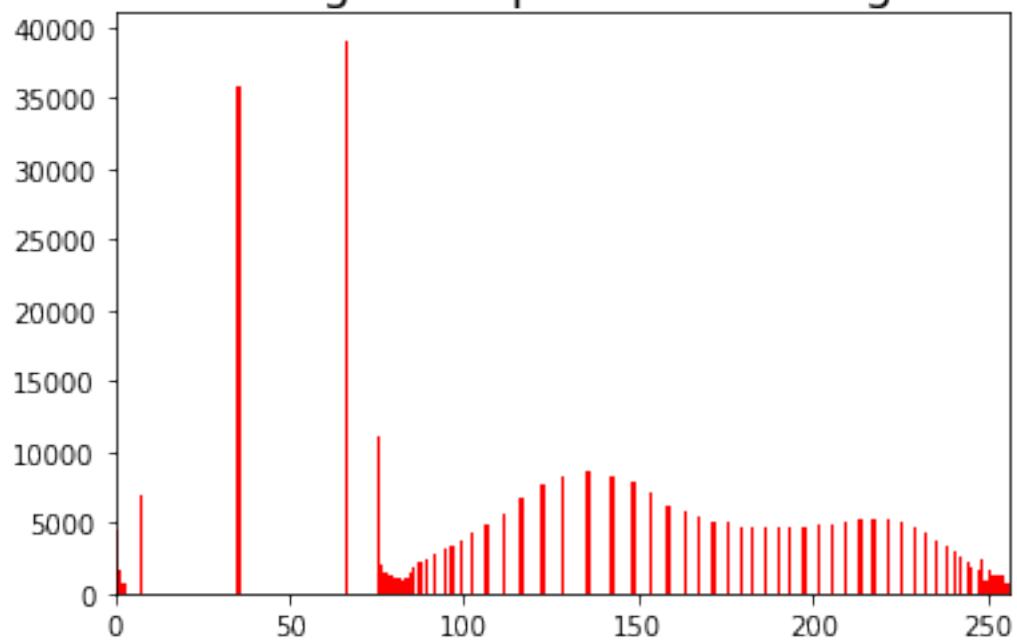
HE_CL(img1)

```

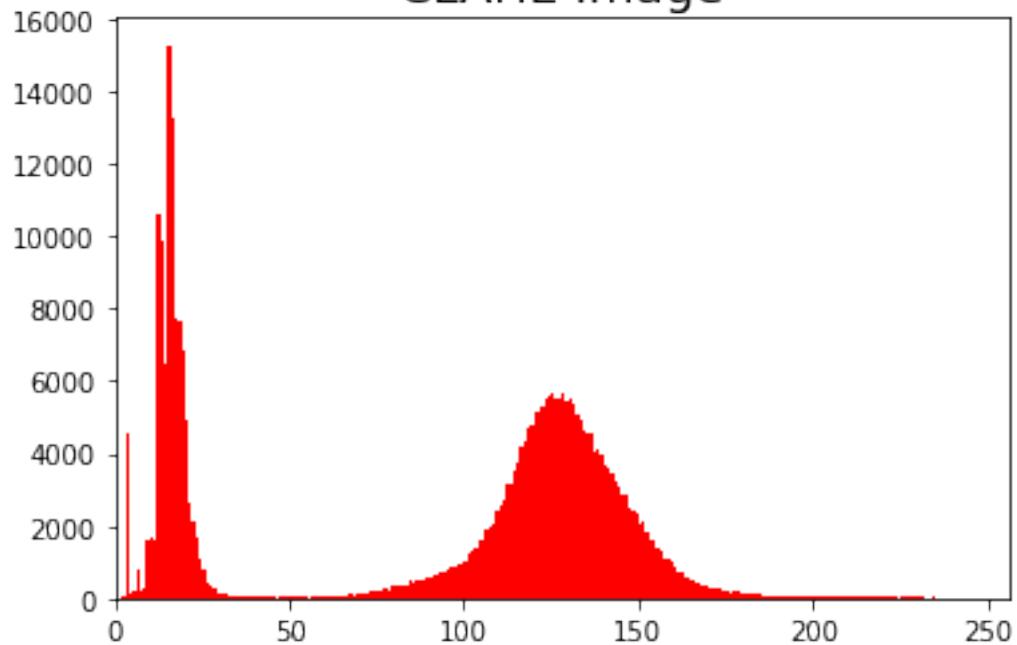
### Histogram of Original Image



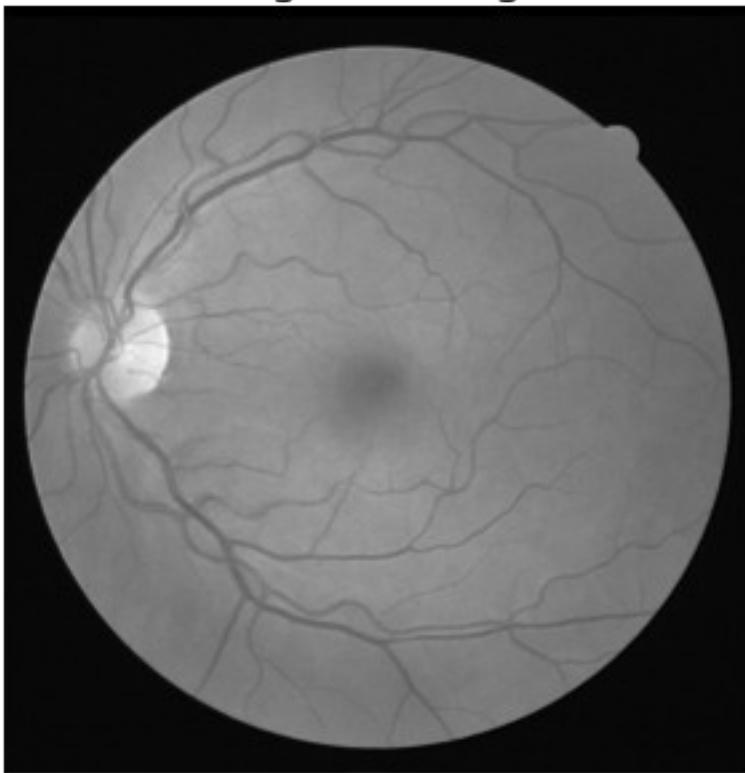
### Histogram Equalization Image



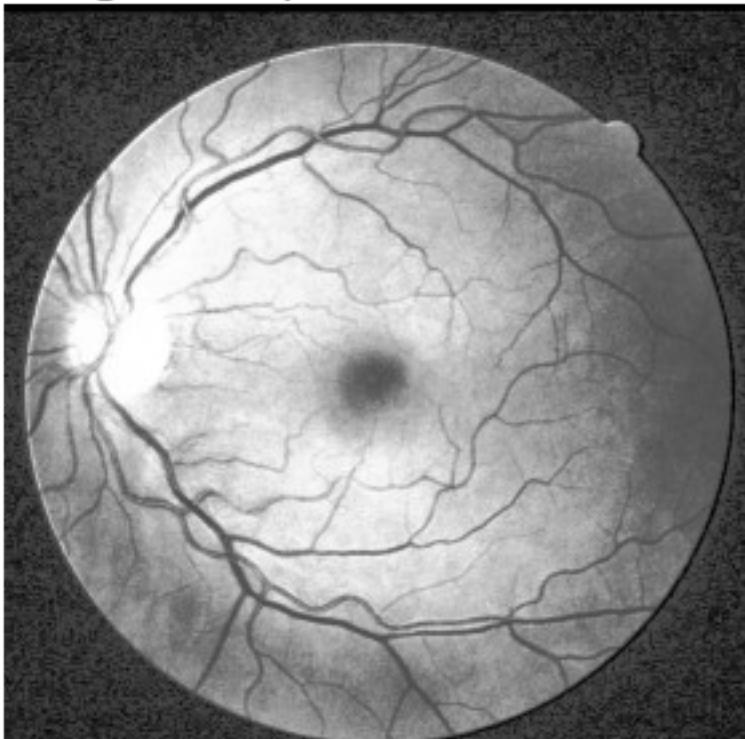
CLAHE Image



Original Image

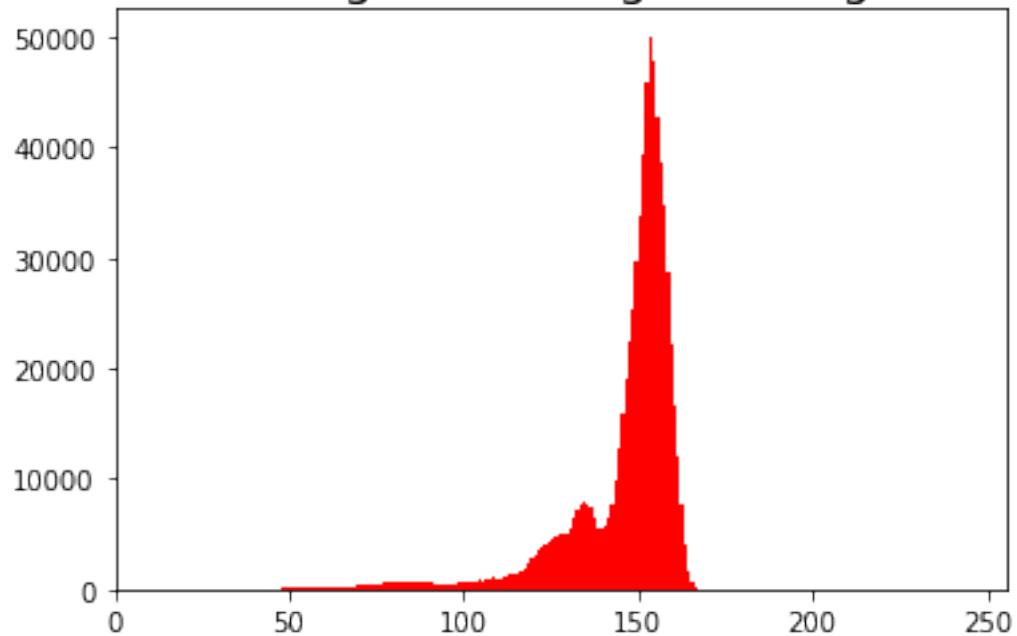


Histogram Equalization Enhanced

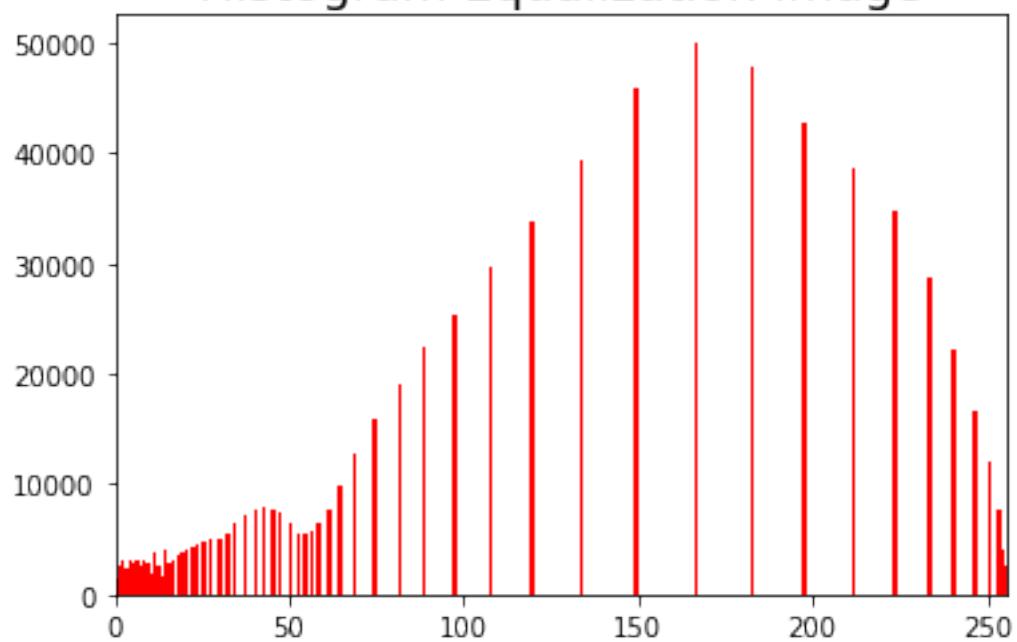


HE\_CL(img9)

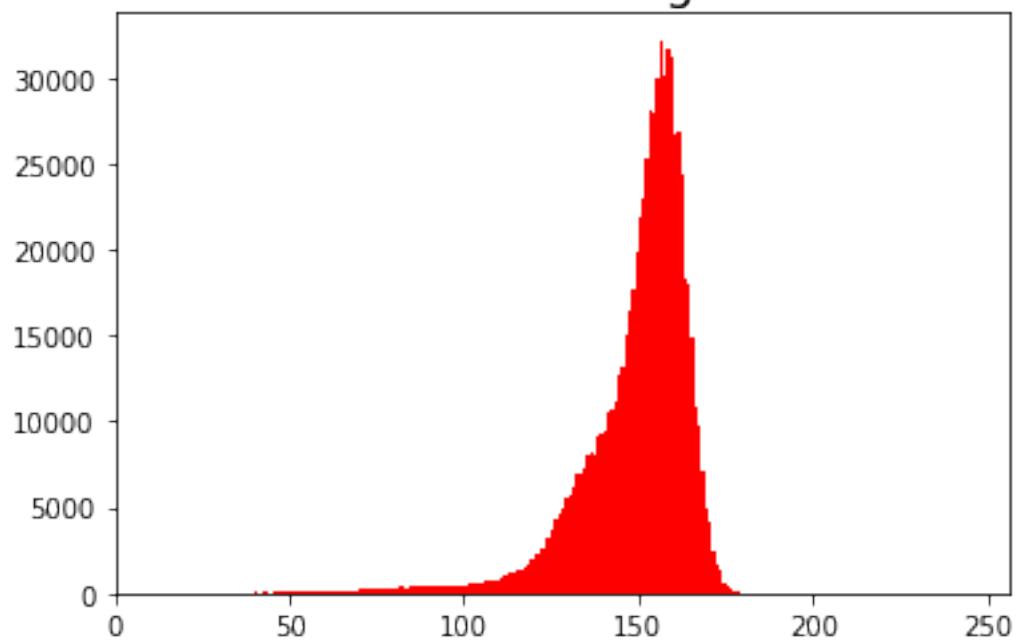
Histogram of Original Image



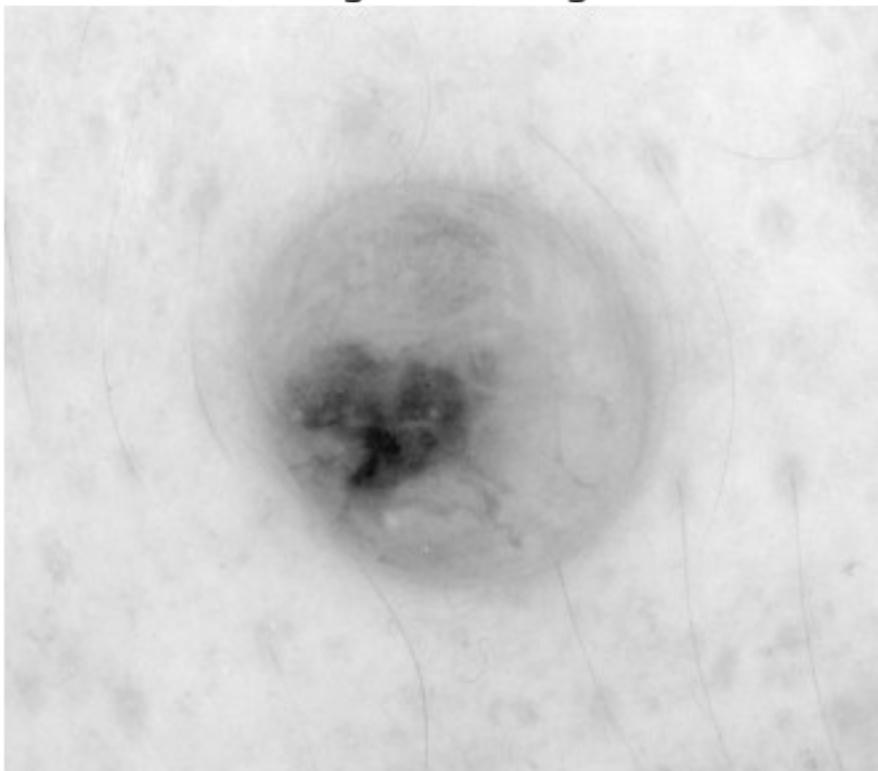
Histogram Equalization Image



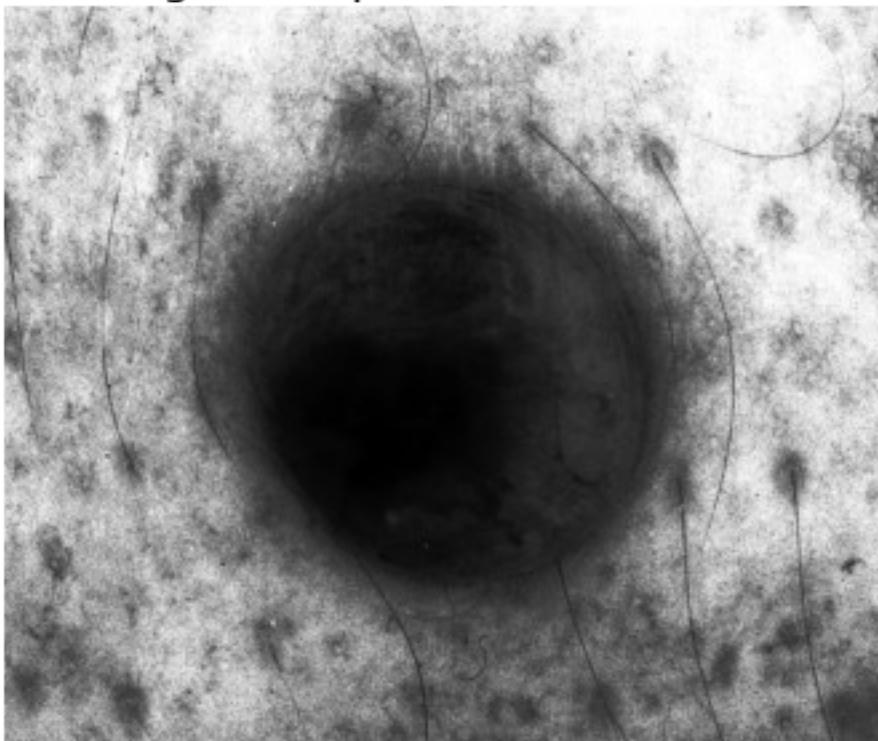
CLAHE Image



Original Image

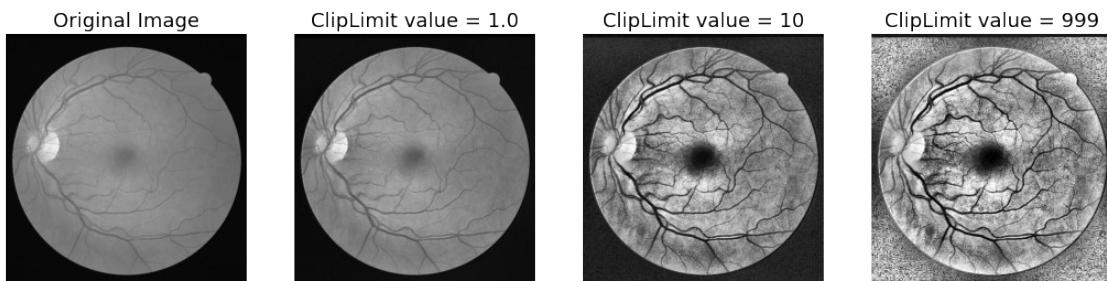


Histogram Equalization Enhanced

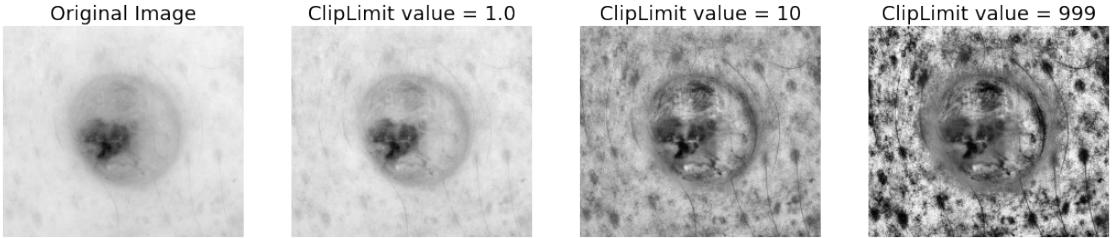


## Implement CLAHE with different clipLimit

```
def clahe2(img) :  
  
    # Apply CLAHE CL 1 TGS 8  
    clahe1 = cv2.createCLAHE(clipLimit=1.0, tileSize=(8,8))  
    cl_img1 = clahe1.apply(img)  
  
    # Apply CLAHE CL 2 TGS 8  
    clahe2 = cv2.createCLAHE(clipLimit=10, tileSize=(8,8))  
    cl_img2 = clahe2.apply(img)  
  
    # Apply CLAHE CL 4 TGS 8  
    clahe3 = cv2.createCLAHE(clipLimit=999, tileSize=(8,8))  
    cl_img3 = clahe3.apply(img)  
  
    # Display the images  
    fig, ax = plt.subplots(1, 4, figsize=(18,18))  
    ax[0].set_title(f'Original Image', fontsize = 18)  
    ax[0].imshow(img, cmap='gray')  
    ax[0].set_axis_off()  
  
    ax[1].set_title(f'ClipLimit value = 1.0', fontsize = 18)  
    ax[1].imshow(cl_img1, cmap='gray')  
    ax[1].set_axis_off()  
  
    ax[2].set_title(f'ClipLimit value = 10', fontsize = 18)  
    ax[2].imshow(cl_img2, cmap='gray')  
    ax[2].set_axis_off()  
  
    ax[3].set_title(f'ClipLimit value = 999', fontsize = 18)  
    ax[3].imshow(cl_img3, cmap='gray')  
    ax[3].set_axis_off()  
  
clahe2(img1)
```



```
clahe2(img9)
```



### Implement CLAHE with different tileSize

```

def clahe3(img) :

    # Apply CLAHE CL 1 TGS 8
    clahe1 = cv2.createCLAHE(clipLimit=4, tileSize=(2,2))
    cl_img1 = clahe1.apply(img)

    # Apply CLAHE CL 2 TGS 8
    clahe2 = cv2.createCLAHE(clipLimit=4, tileSize=(8,8))
    cl_img2 = clahe2.apply(img)

    # Apply CLAHE CL 4 TGS 8
    clahe3 = cv2.createCLAHE(clipLimit=4, tileSize=(64,64))
    cl_img3 = clahe3.apply(img)

    # Display the images
    fig, ax = plt.subplots(1, 4, figsize=(18,18))
    ax[0].set_title(f'Original Image', fontsize = 18)
    ax[0].imshow(img, cmap='gray')
    ax[0].set_axis_off()

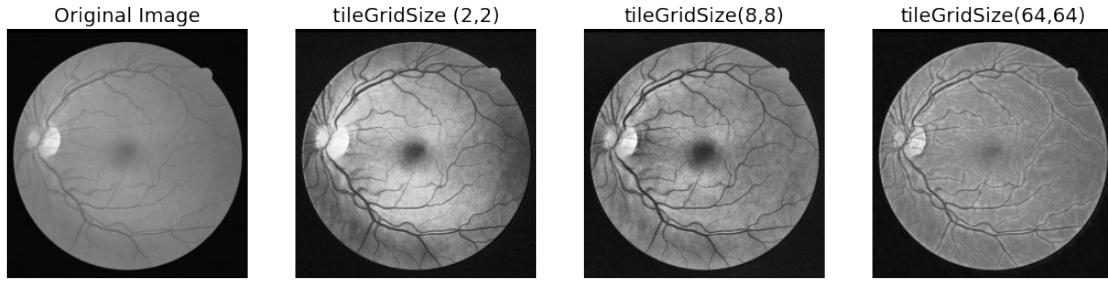
    ax[1].set_title(f'tileSize (2,2)', fontsize = 18)
    ax[1].imshow(cl_img1, cmap='gray')
    ax[1].set_axis_off()

    ax[2].set_title(f'tileSize(8,8)', fontsize = 18)
    ax[2].imshow(cl_img2, cmap='gray')
    ax[2].set_axis_off()

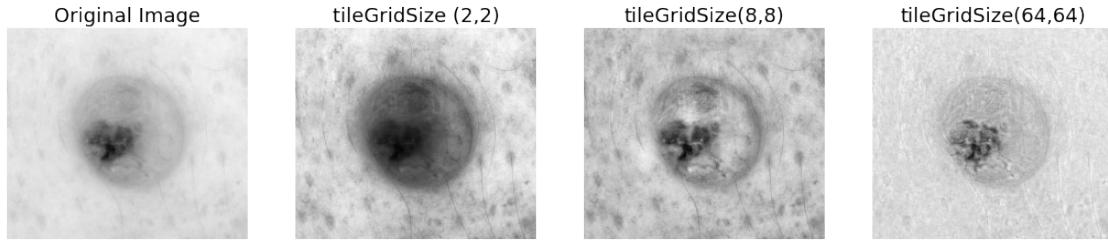
    ax[3].set_title(f'tileSize(64,64)', fontsize = 18)
    ax[3].imshow(cl_img3, cmap='gray')
    ax[3].set_axis_off()

clahe3(img1)

```



`clahe3(img9)`



## Second Challenge : Implementing the filtering methods of Mean filter, Median filter, and Gaussian filter

```
#Create function
def filter (img, ksize) :

    # Apply the mean filter using OpenCV
    img_mean = cv2.blur(img, (ksize,ksize))

    #Apply the median filter using OpenCV
    img_median = cv2.medianBlur(img, ksize)

    #Apply the gaussian filter using OpenCV
    img_gauss = cv2.GaussianBlur(img, (ksize,ksize),0)

    fig, ax = plt.subplots(1, 4, figsize=(25,25))
    ax[0].set_title(f'Original Image', fontsize = 18)
    ax[0].imshow(img, cmap='gray')
    ax[0].set_axis_off()

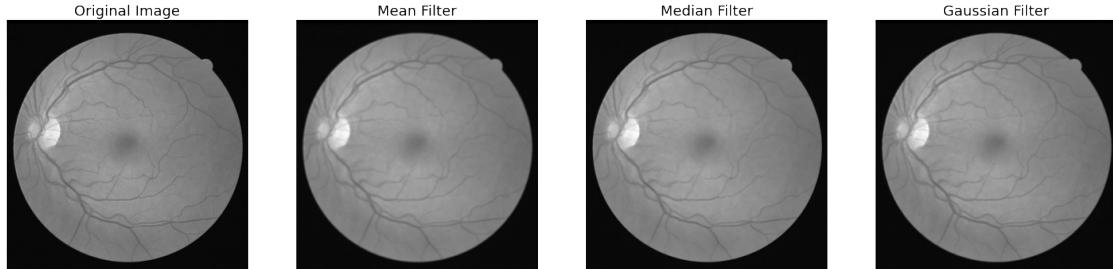
    ax[1].set_title(f'Mean Filter', fontsize = 18)
    ax[1].imshow(img_mean, cmap='gray')
    ax[1].set_axis_off()

    ax[2].set_title(f'Median Filter', fontsize = 18)
    ax[2].imshow(img_median, cmap='gray')
    ax[2].set_axis_off()

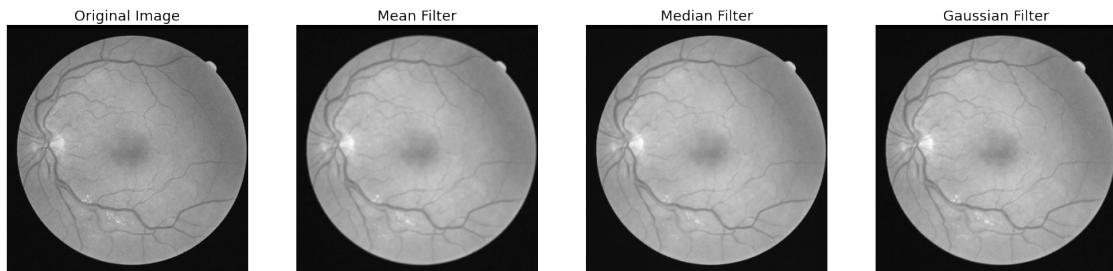
    ax[3].set_title(f'Gaussian Filter', fontsize = 18)
```

```
ax[3].imshow(img_gauss, cmap='gray')
ax[3].set_axis_off()
```

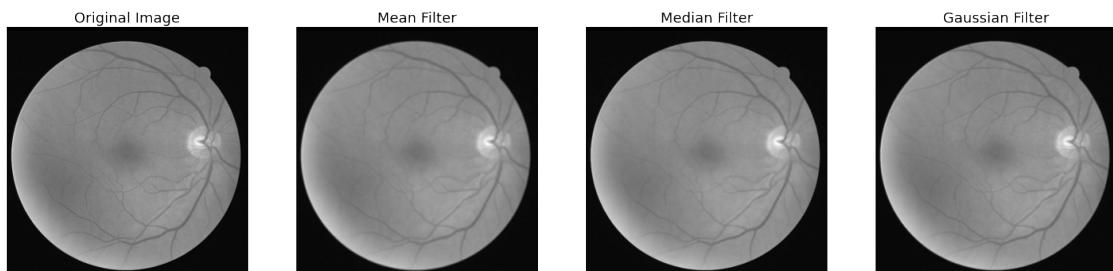
**5x5 Kernel**  
`filter(img1,5)`



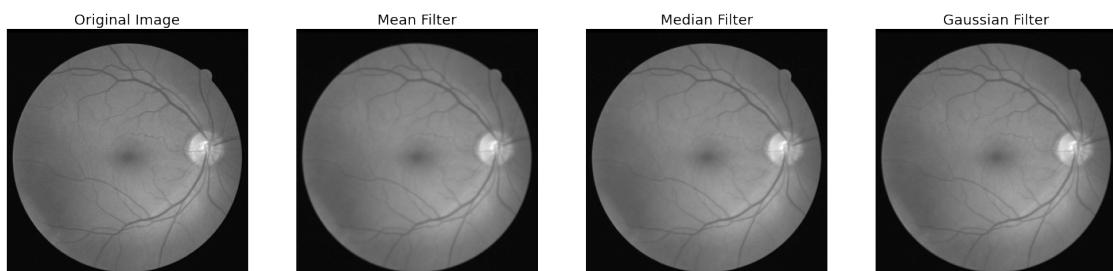
`filter(img2,5)`



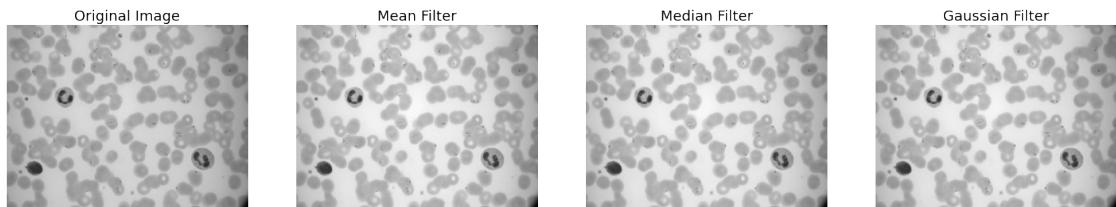
`filter(img3,5)`



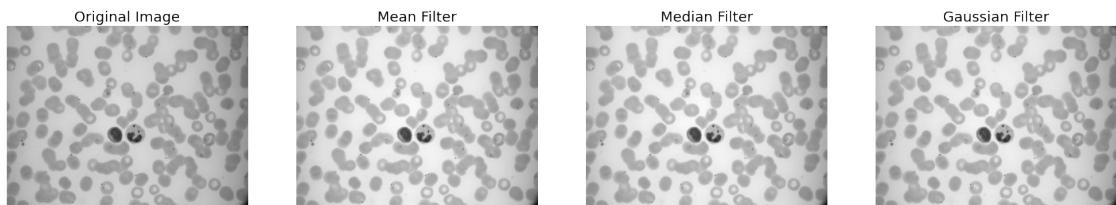
`filter(img4,5)`



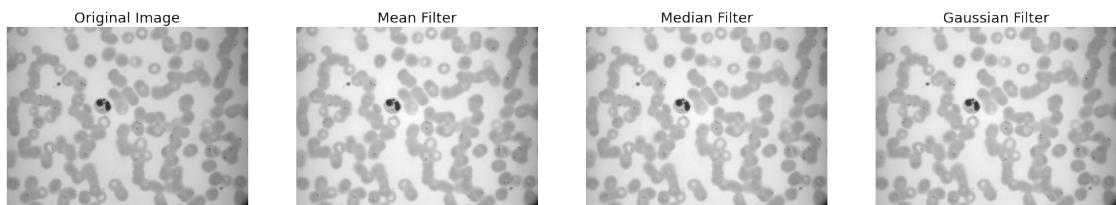
`filter(img5,5)`



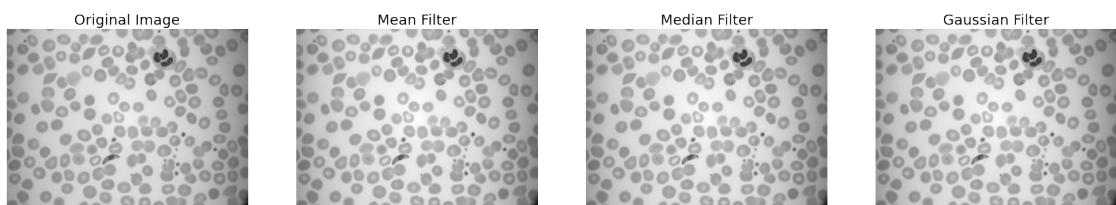
`filter(img6,5)`



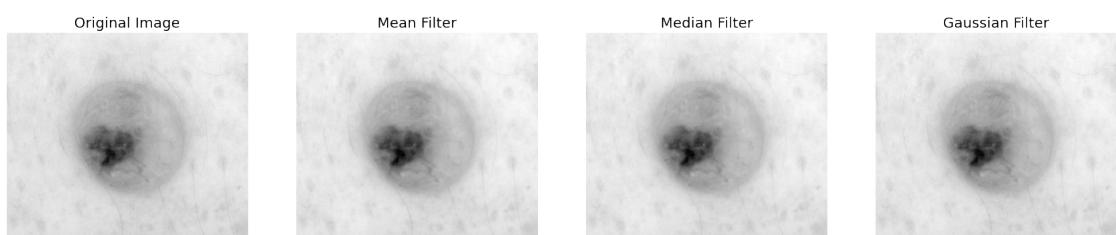
`filter(img7,5)`



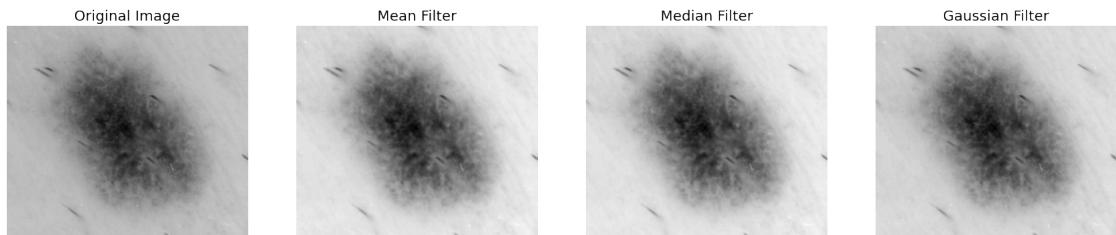
`filter(img8,5)`



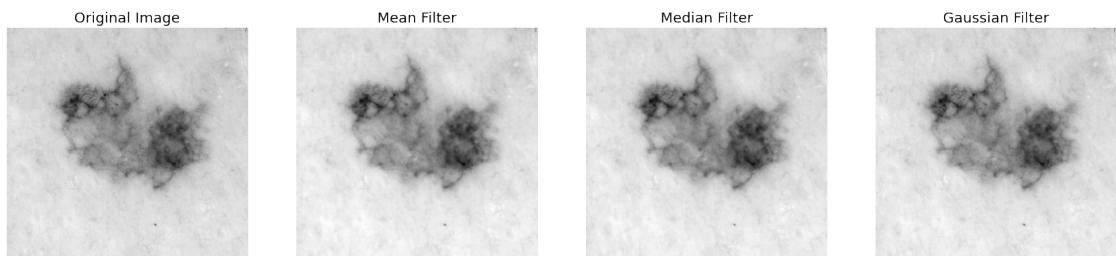
`filter(img9,5)`



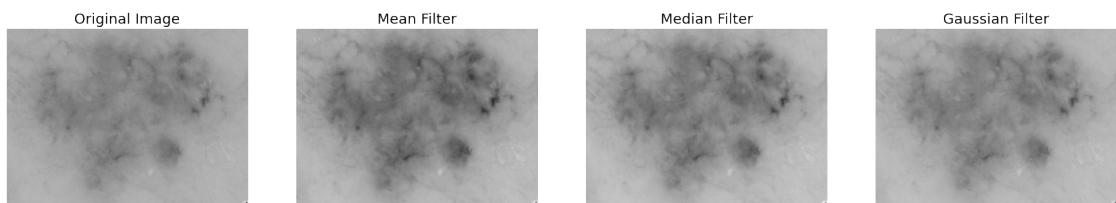
`filter(img10,5)`



`filter(img11, 5)`

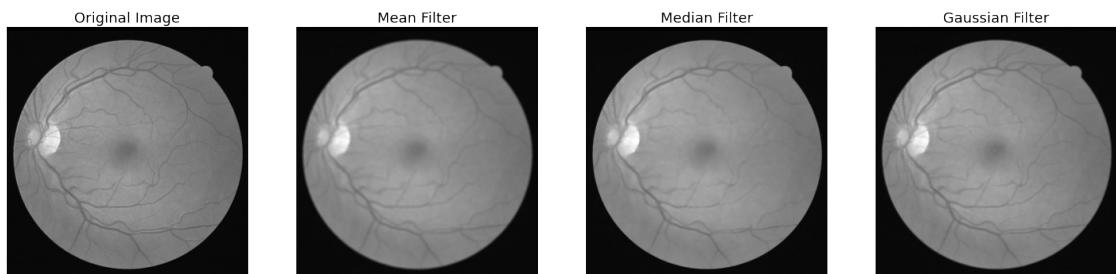


`filter(img12, 5)`

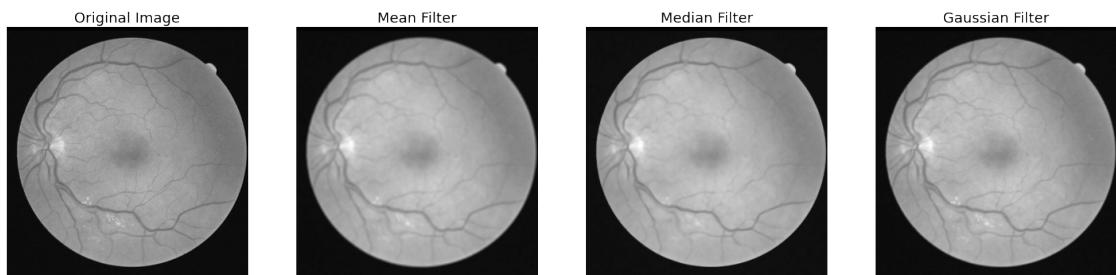


## 7x7 Kernel

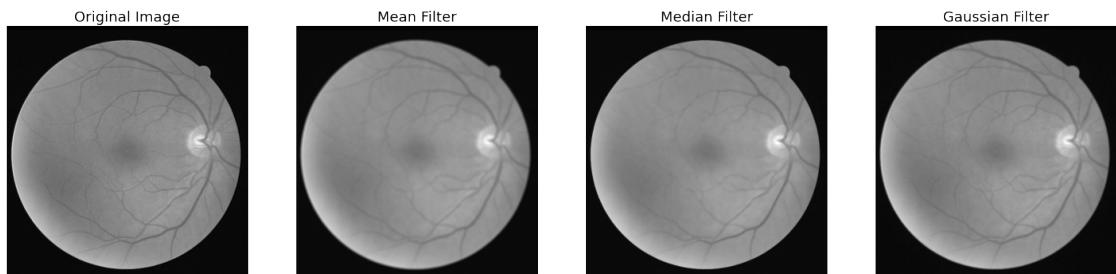
`filter(img1, 7)`



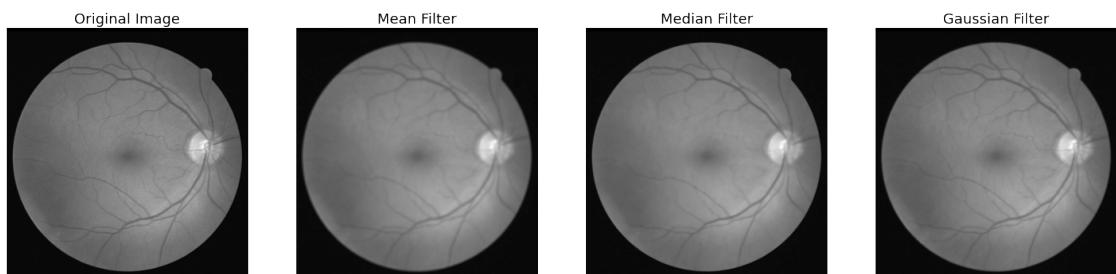
`filter(img2, 7)`



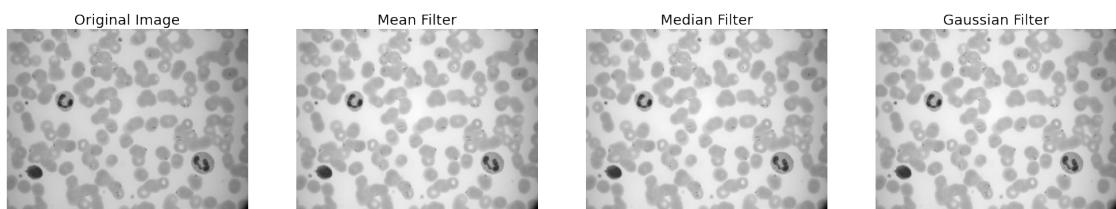
`filter(img3, 7)`



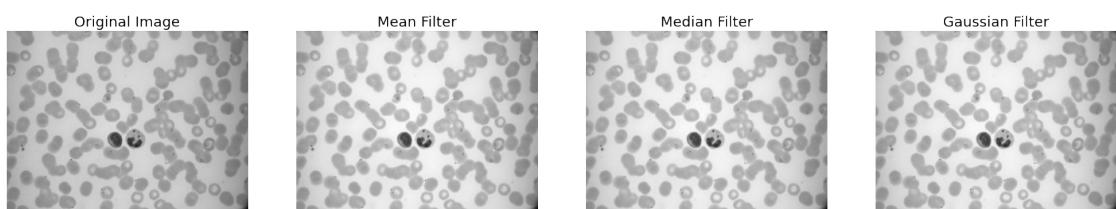
`filter(img4, 7)`



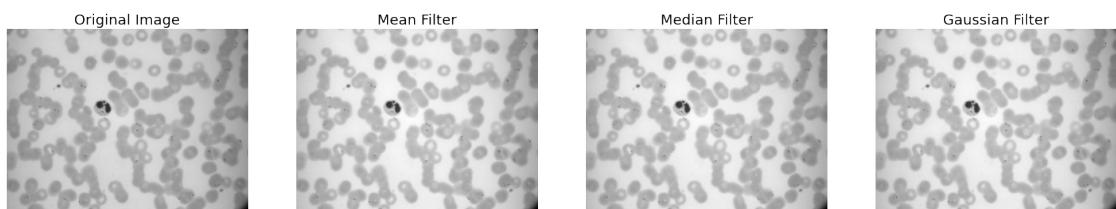
`filter(img5, 7)`



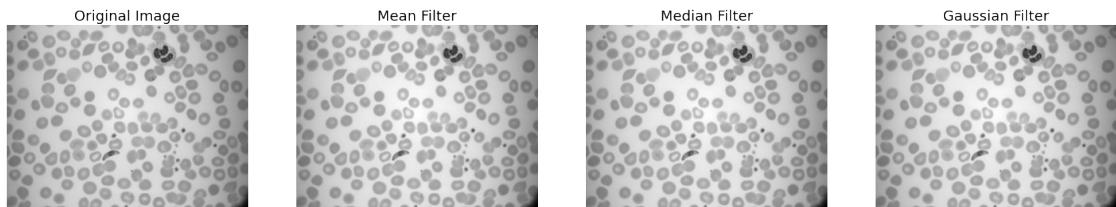
`filter(img6, 7)`



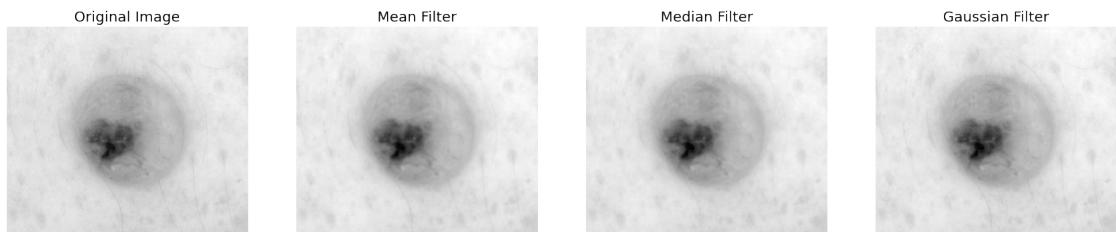
`filter(img7, 7)`



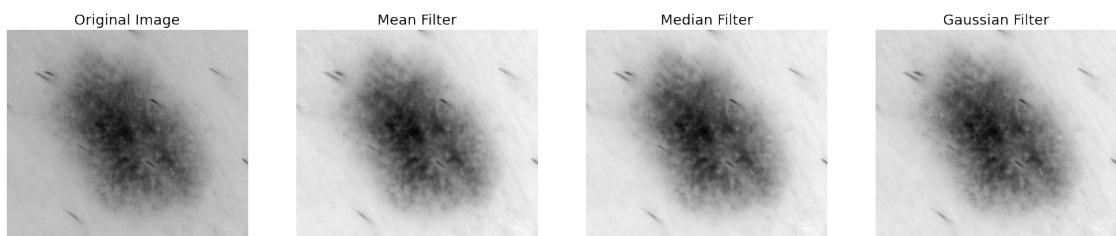
`filter(img8, 7)`



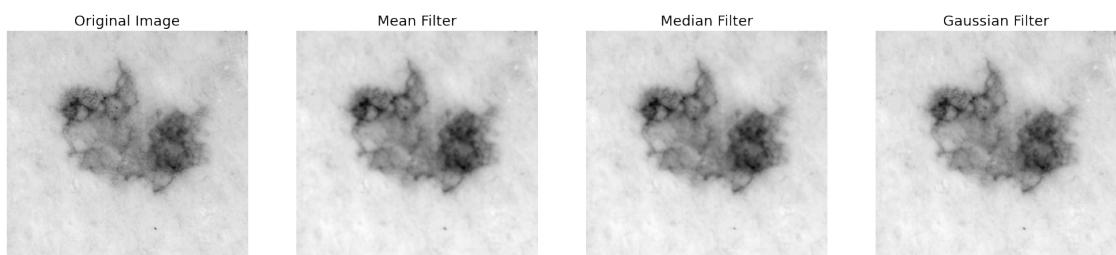
`filter(img9, 7)`



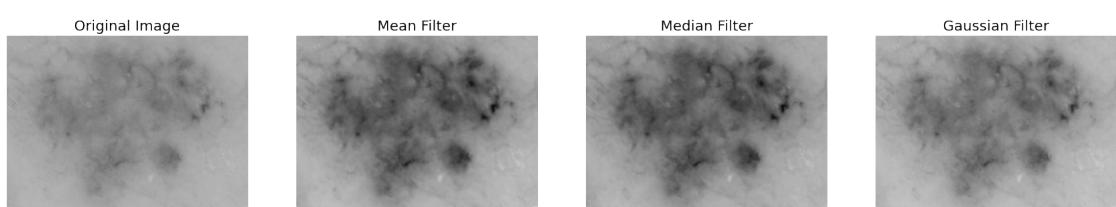
`filter(img10, 7)`



`filter(img11, 7)`

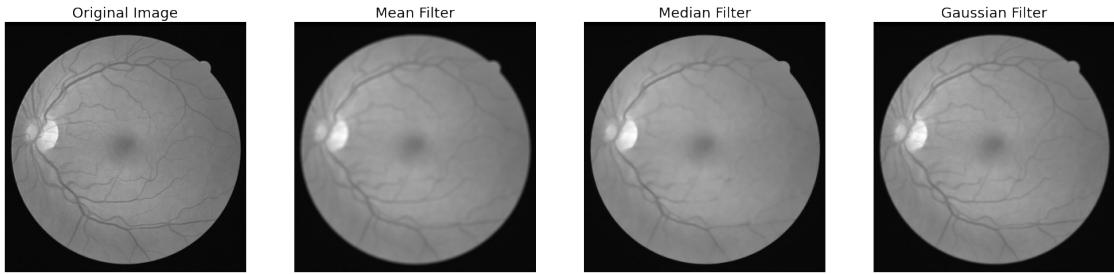


`filter(img12, 7)`

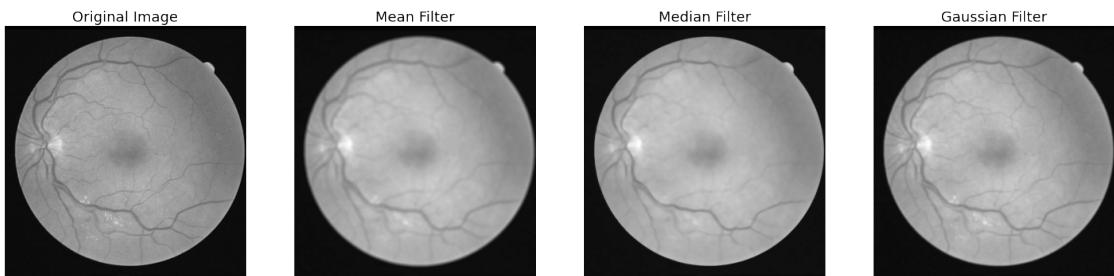


**9x9 Kernel**

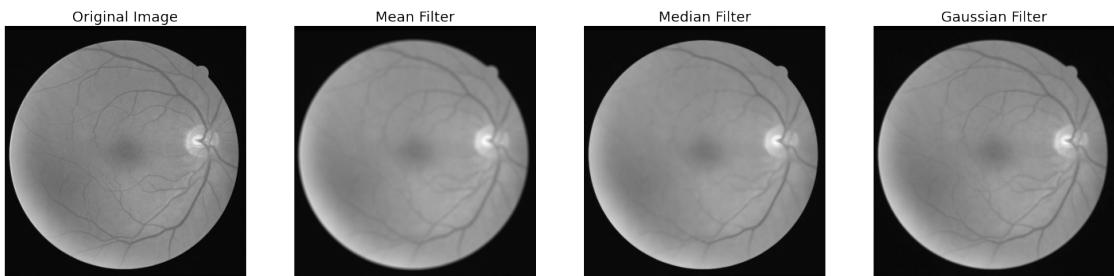
`filter(img1, 9)`



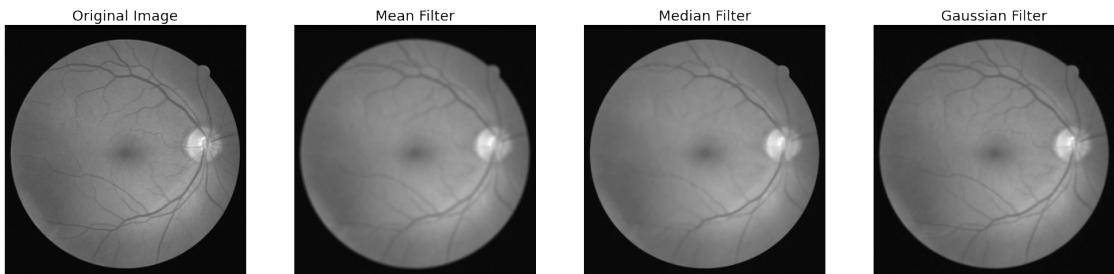
`filter(img2,9)`



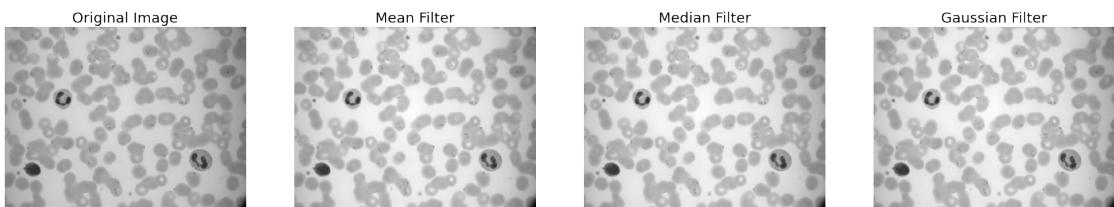
`filter(img3,9)`



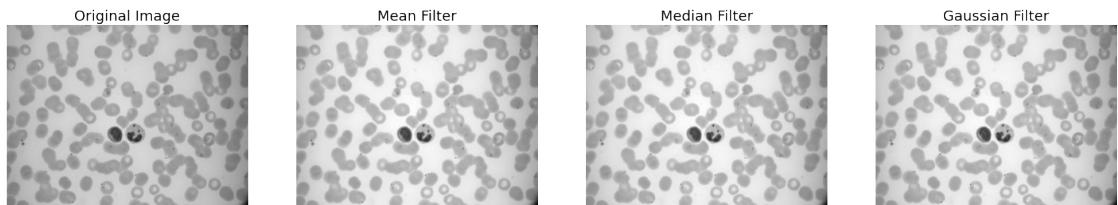
`filter(img4,9)`



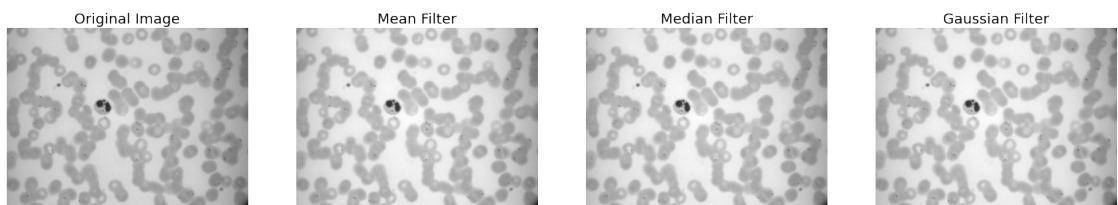
`filter(img5,9)`



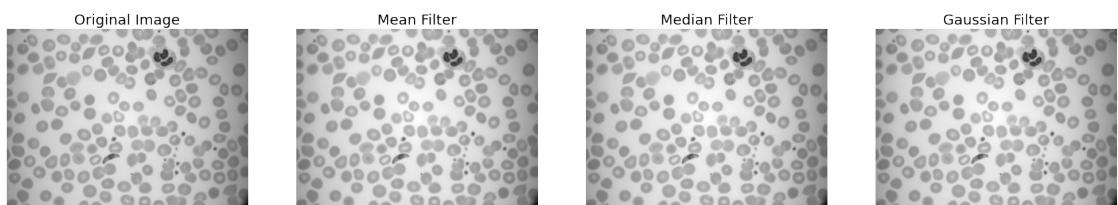
`filter(img6,9)`



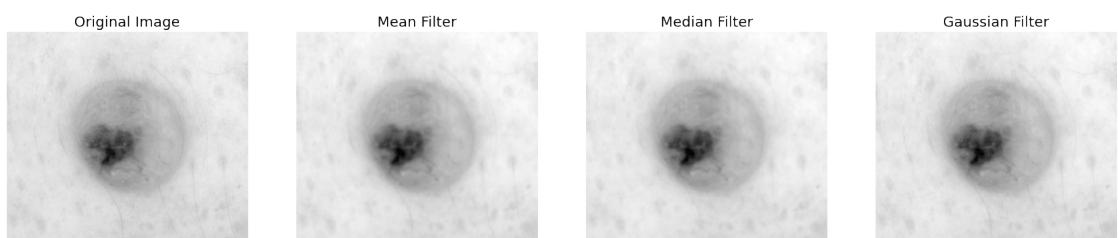
`filter(img7,9)`



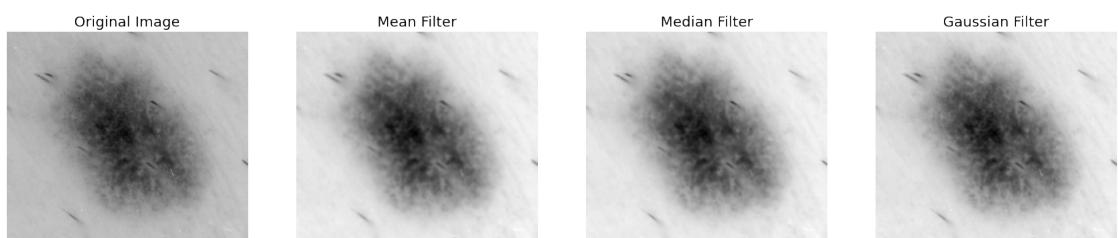
`filter(img8,9)`



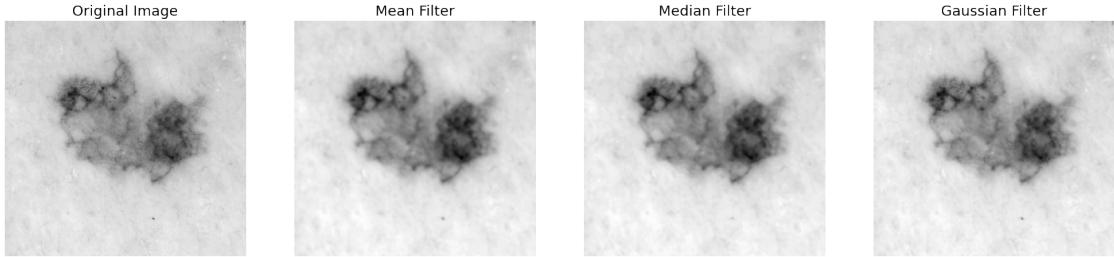
`filter(img9,9)`



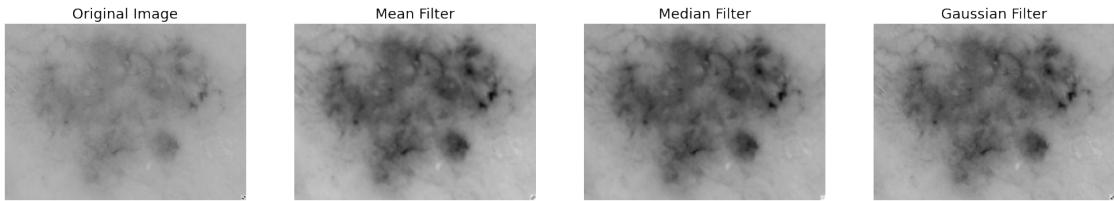
`filter(img10,9)`



`filter(img11,9)`



```
filter(img12,9)
```



Try to look at different kernel and different sigma value for gaussian blur

### Mean Filter

```
def mean_filter (img) :

    # Apply the mean filter with 5x5 kernel
    img_mean = cv2.blur(img,(3,3))

    #Apply the mean filter with 7x7 kernel
    img_mean2 = cv2.blur(img,(5,5))

    #Apply the mean filter with 9x9 kernel
    img_mean3 = cv2.blur(img,(9,9))

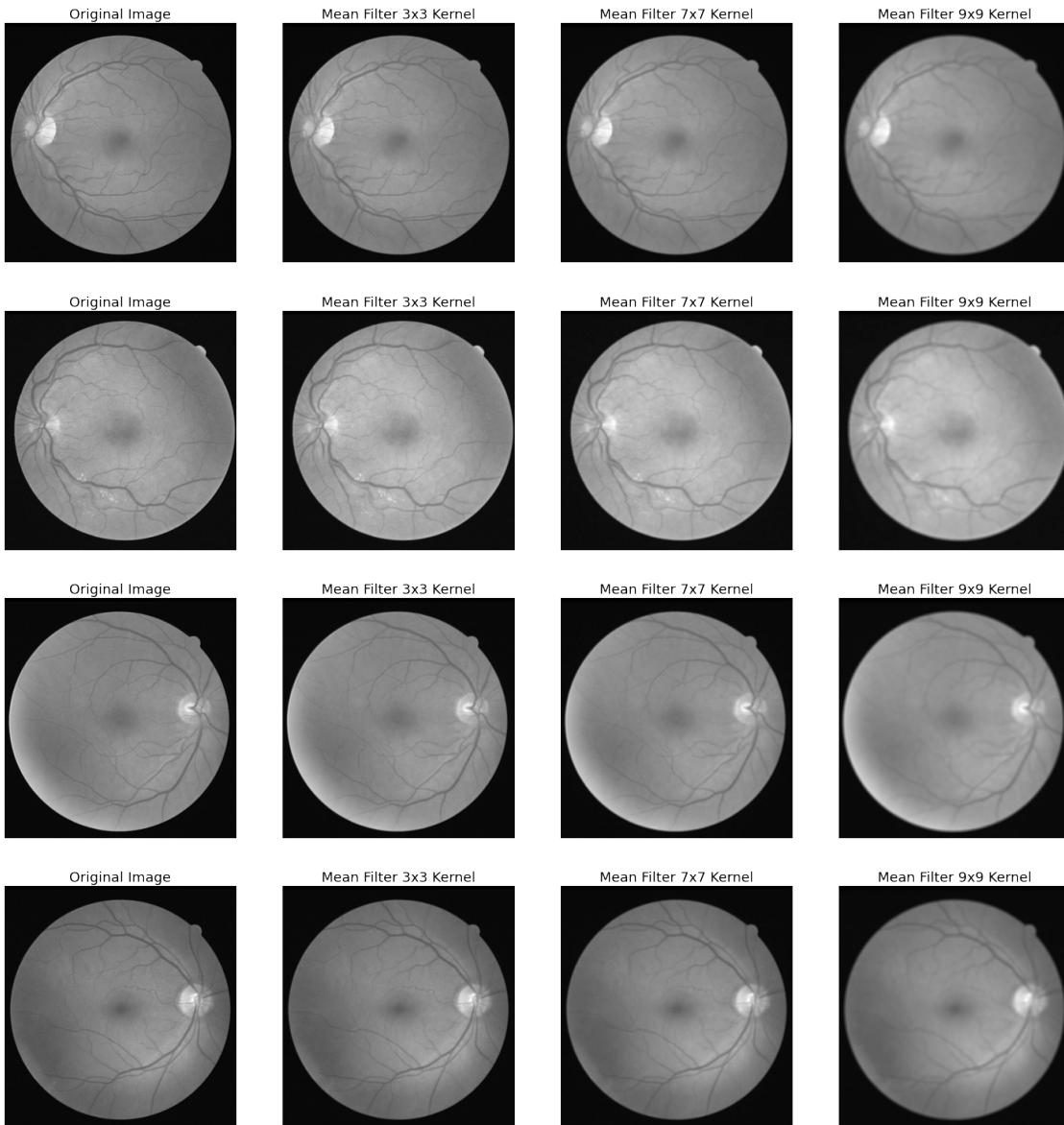
    fig, ax = plt.subplots(1, 4, figsize=(25,25))
    ax[0].set_title(f'Original Image', fontsize = 18)
    ax[0].imshow(img, cmap='gray')
    ax[0].set_axis_off()

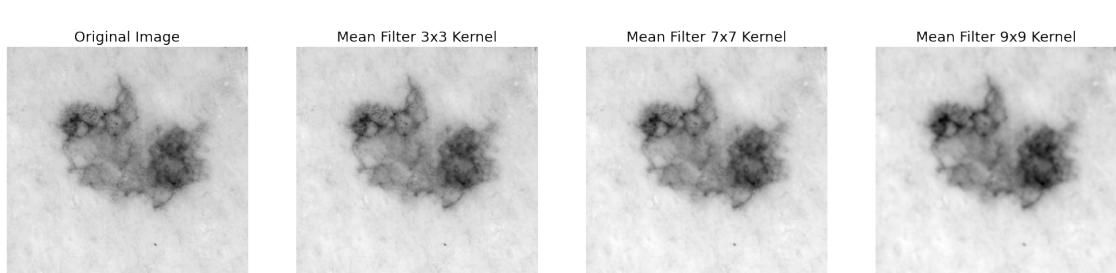
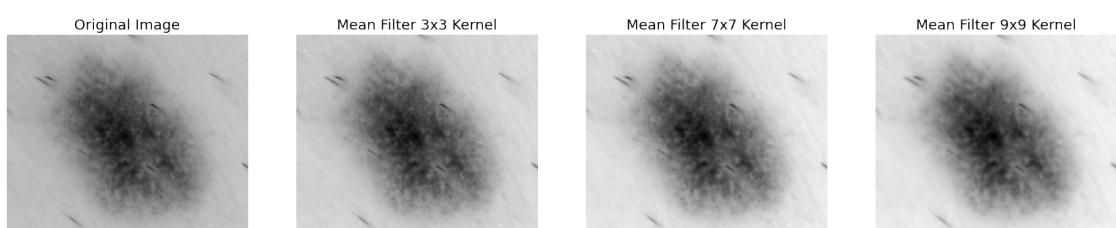
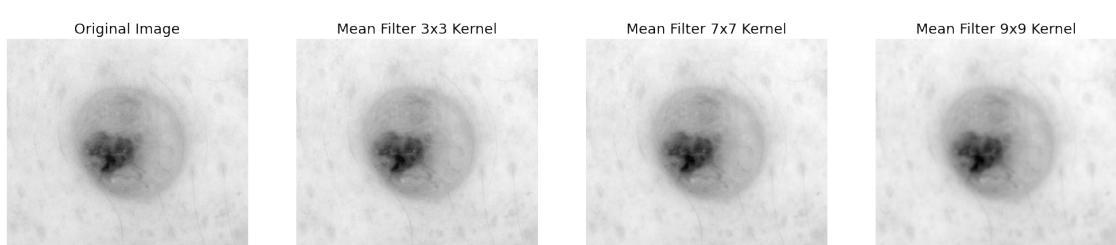
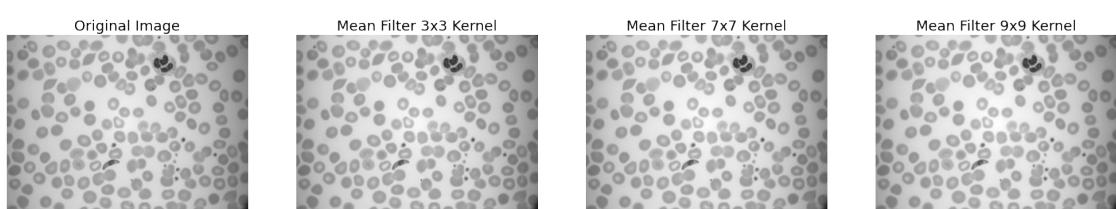
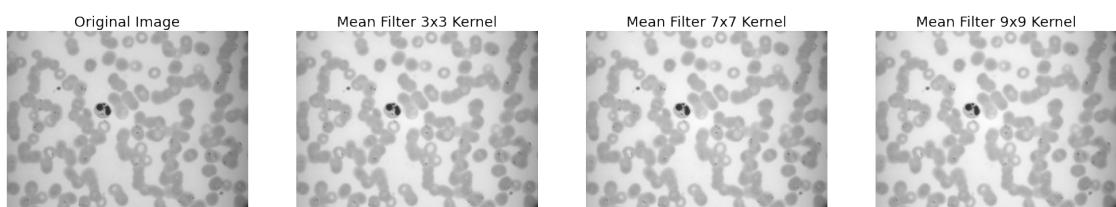
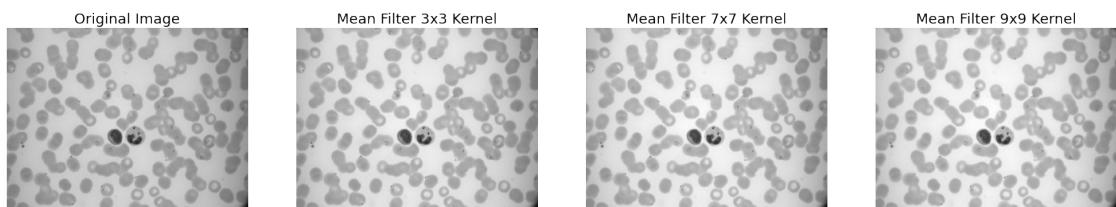
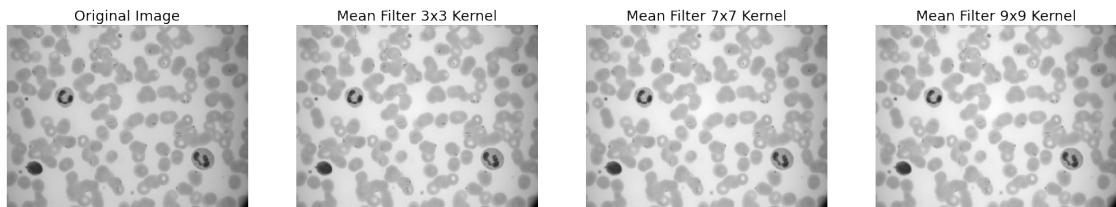
    ax[1].set_title(f'Mean Filter 3x3 Kernel', fontsize = 18)
    ax[1].imshow(img_mean, cmap='gray')
    ax[1].set_axis_off()

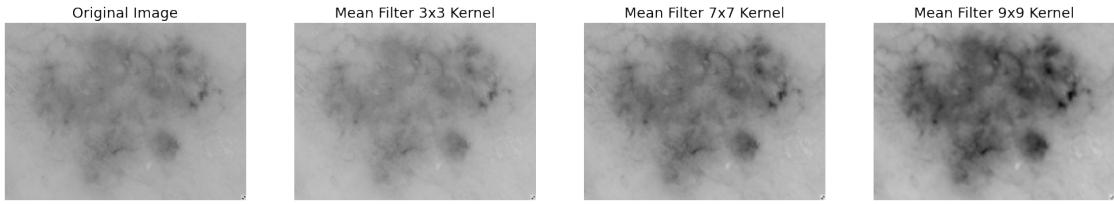
    ax[2].set_title(f'Mean Filter 7x7 Kernel', fontsize = 18)
    ax[2].imshow(img_mean2, cmap='gray')
    ax[2].set_axis_off()

    ax[3].set_title(f'Mean Filter 9x9 Kernel', fontsize = 18)
    ax[3].imshow(img_mean3, cmap='gray')
    ax[3].set_axis_off()
```

```
mean_filter(img1)
mean_filter(img2)
mean_filter(img3)
mean_filter(img4)
mean_filter(img5)
mean_filter(img6)
mean_filter(img7)
mean_filter(img8)
mean_filter(img9)
mean_filter(img10)
mean_filter(img11)
mean_filter(img12)
```

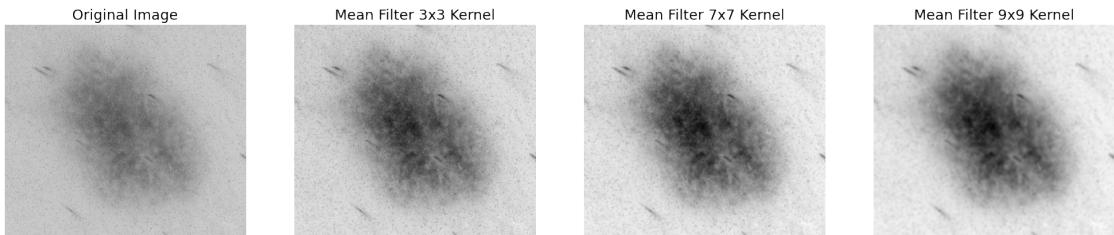
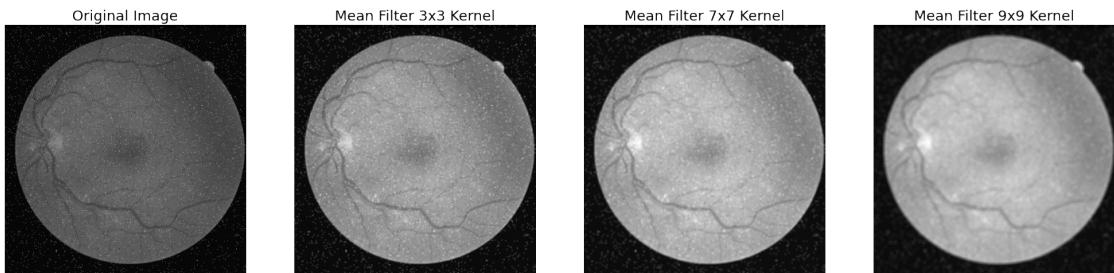






```
#Menambahkan noise untuk melihat performa algoritma penghilang noise
from skimage import io, util
```

```
mean_filter(util.random_noise(img2, mode='s&p', amount=0.02))
mean_filter(util.random_noise(img10, mode='s&p', amount=0.02))
```



```
###Median Filter
```

```
def median_filter (img) :

    # Apply the median filter with 5x5 kernel
    img_median = cv2.medianBlur(img,5)

    #Apply the median filter with 7x7 kernel
    img_median2 = cv2.medianBlur(img,7)

    #Apply the median filter with 9x9 kernel
    img_median3 = cv2.medianBlur(img,9)

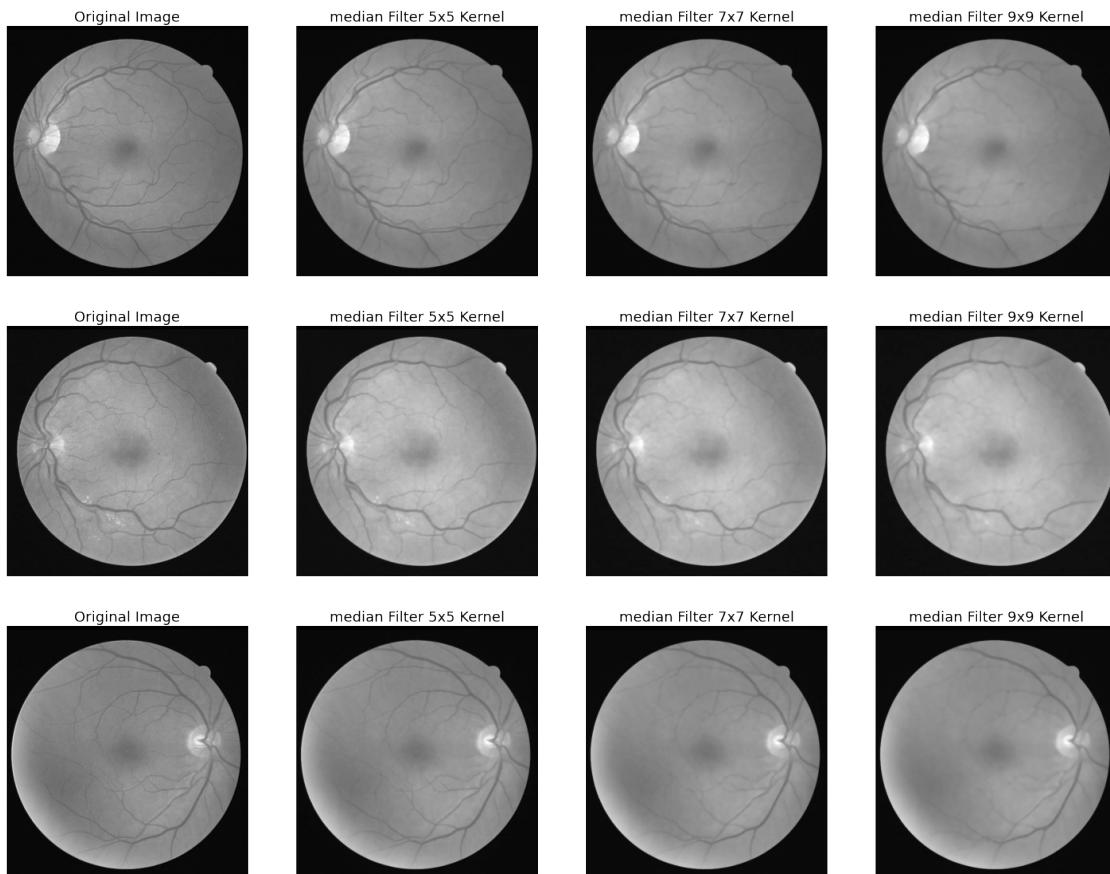
    fig, ax = plt.subplots(1, 4, figsize=(25,25))
    ax[0].set_title(f'Original Image', fontsize = 18)
    ax[0].imshow(img, cmap='gray')
    ax[0].set_axis_off()

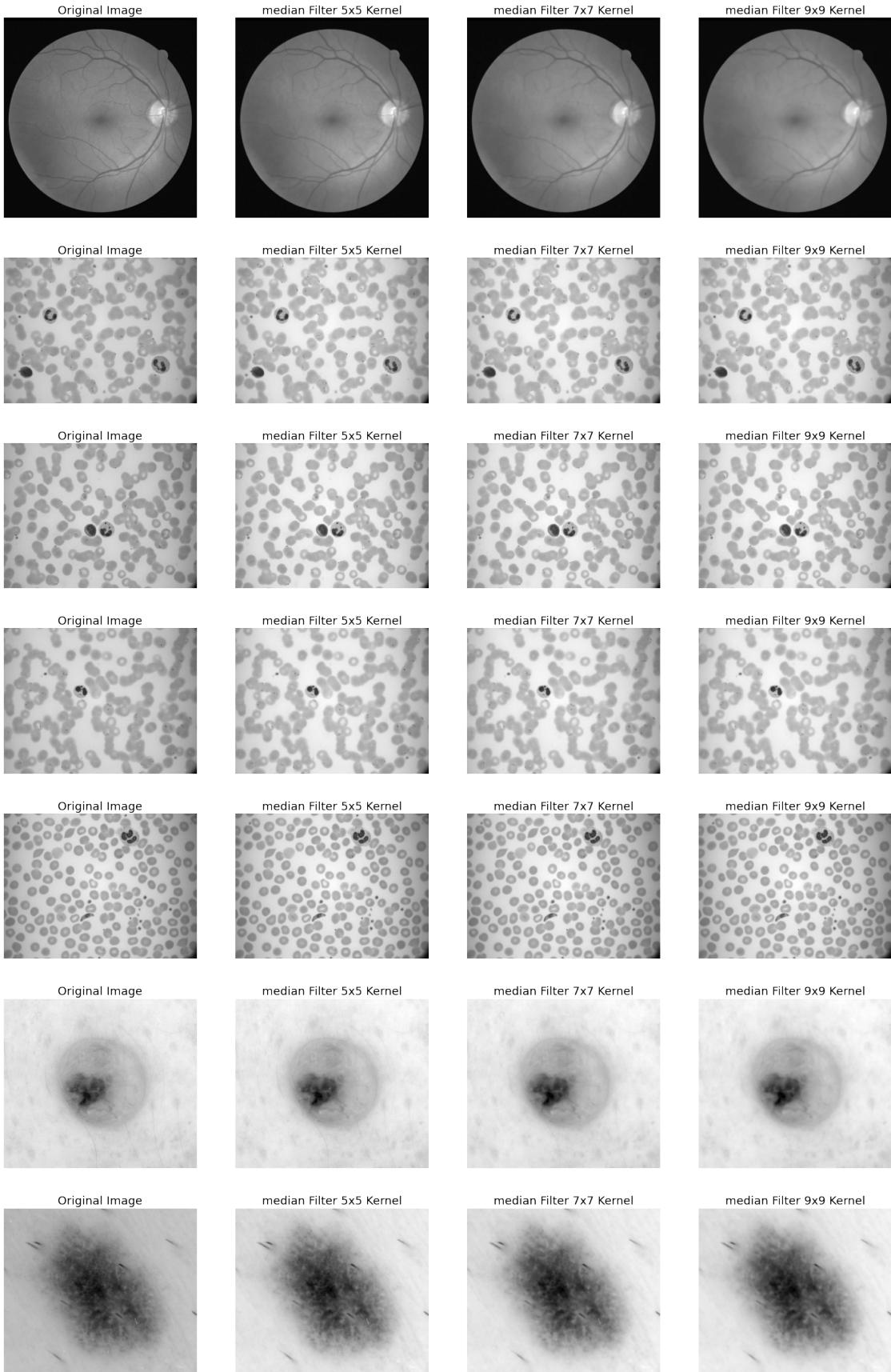
    ax[1].set_title(f'median Filter 5x5 Kernel', fontsize = 18)
    ax[1].imshow(img_median, cmap='gray')
    ax[1].set_axis_off()
```

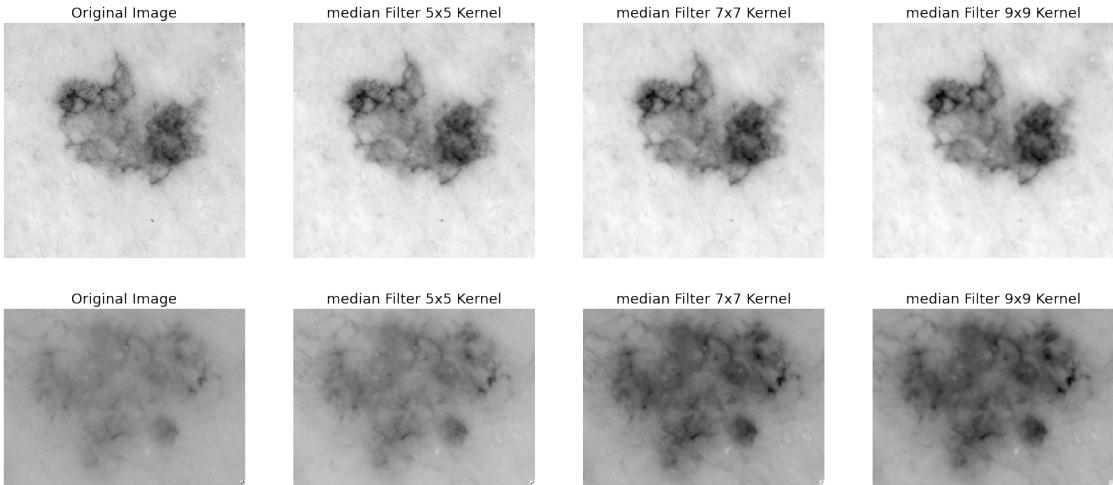
```
ax[2].set_title(f'median Filter 7x7 Kernel', fontsize = 18)
ax[2].imshow(img_median2, cmap='gray')
ax[2].set_axis_off()

ax[3].set_title(f'median Filter 9x9 Kernel', fontsize = 18)
ax[3].imshow(img_median3, cmap='gray')
ax[3].set_axis_off()

median_filter(img1)
median_filter(img2)
median_filter(img3)
median_filter(img4)
median_filter(img5)
median_filter(img6)
median_filter(img7)
median_filter(img8)
median_filter(img9)
median_filter(img10)
median_filter(img11)
median_filter(img12)
```







###Gaussian Blur Filter

```
def gauss_filter (img) :

    # Apply the gauss filter with 5x5 kernel
    img_gauss = cv2.GaussianBlur(img,(5,5),0)

    #Apply the gauss filter with 7x7 kernel
    img_gauss2 = cv2.GaussianBlur(img,(7,7),0)

    #Apply the gauss filter with 9x9 kernel
    img_gauss3 = cv2.GaussianBlur(img,(9,9),0)

    fig, ax = plt.subplots(1, 4, figsize=(25,25))
    ax[0].set_title(f'Original Image', fontsize = 18)
    ax[0].imshow(img, cmap='gray')
    ax[0].set_axis_off()

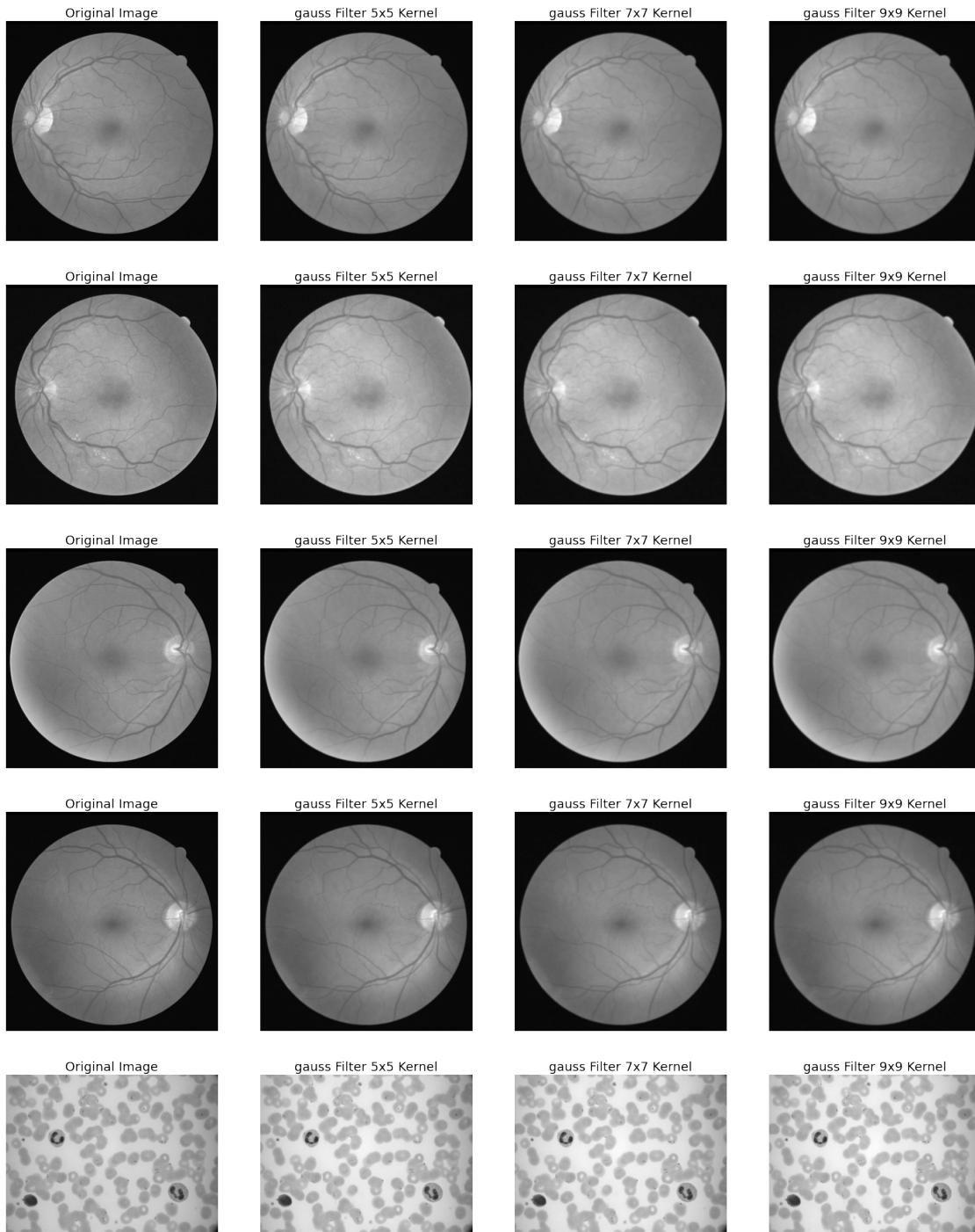
    ax[1].set_title(f'gauss Filter 5x5 Kernel', fontsize = 18)
    ax[1].imshow(img_gauss, cmap='gray')
    ax[1].set_axis_off()

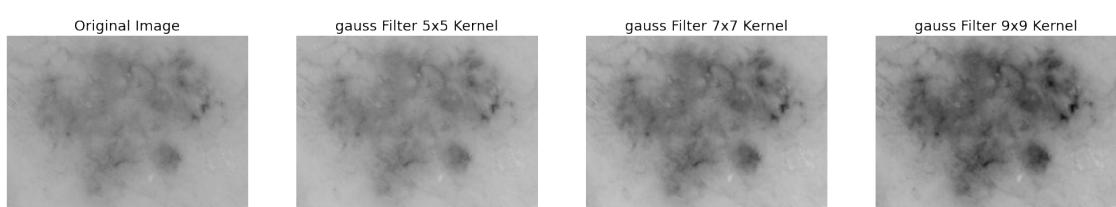
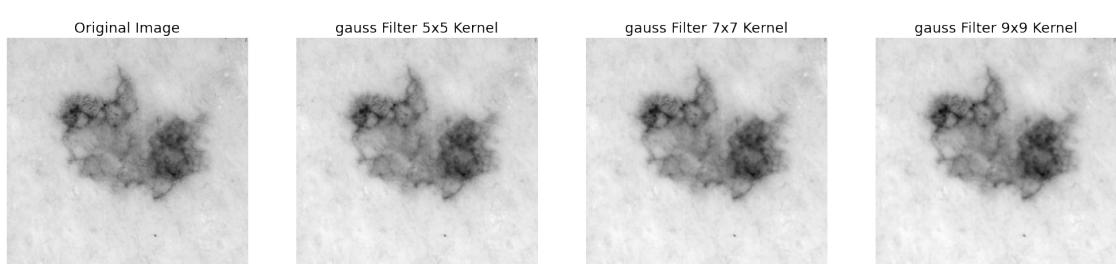
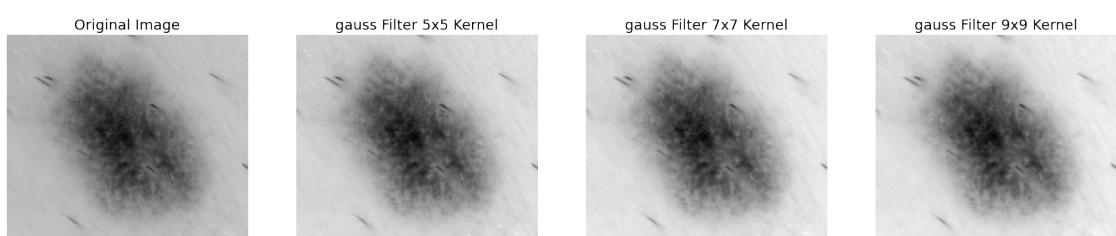
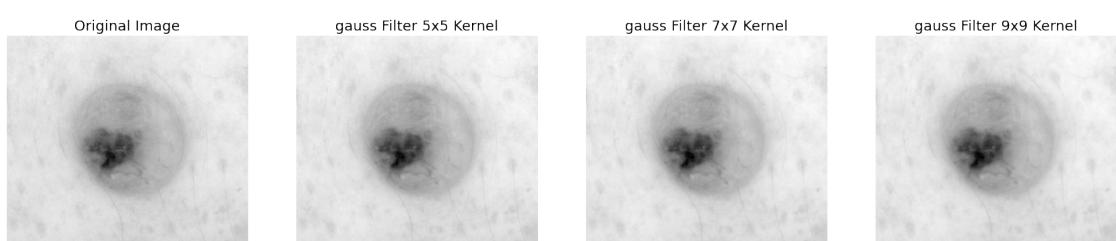
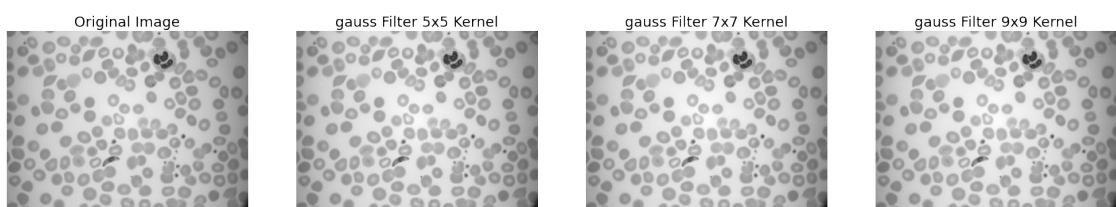
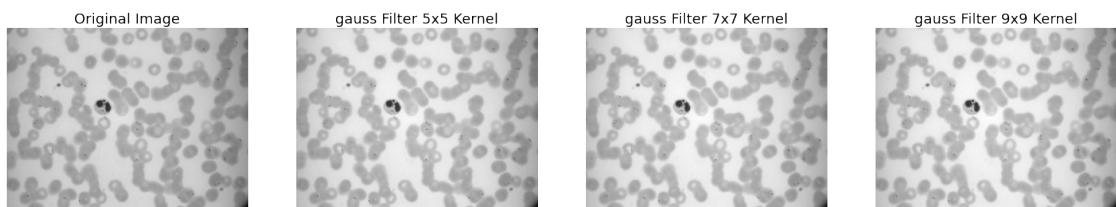
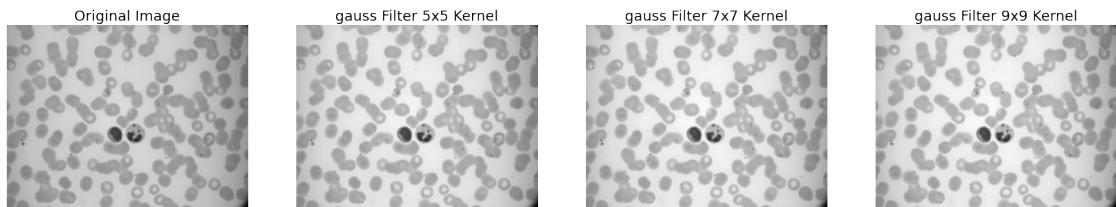
    ax[2].set_title(f'gauss Filter 7x7 Kernel', fontsize = 18)
    ax[2].imshow(img_gauss2, cmap='gray')
    ax[2].set_axis_off()

    ax[3].set_title(f'gauss Filter 9x9 Kernel', fontsize = 18)
    ax[3].imshow(img_gauss3, cmap='gray')
    ax[3].set_axis_off()

gauss_filter(img1)
gauss_filter(img2)
gauss_filter(img3)
gauss_filter(img4)
gauss_filter(img5)
```

```
gauss_filter(img6)
gauss_filter(img7)
gauss_filter(img8)
gauss_filter(img9)
gauss_filter(img10)
gauss_filter(img11)
gauss_filter(img12)
```





```
###Gaussian Blur Filter With different Sigma value

def gauss2_filter (img) :

    # Apply the gauss filter with 0 sigma
    img_gauss = cv2.GaussianBlur(img,(5,5),0)

    #Apply the gauss filter with 2 sigma
    img_gauss2 = cv2.GaussianBlur(img,(5,5),2)

    #Apply the gauss filter with 4 sigma
    img_gauss3 = cv2.GaussianBlur(img,(5,5),4)

    fig, ax = plt.subplots(1, 4, figsize=(25,25))
    ax[0].set_title(f'Original Image', fontsize = 18)
    ax[0].imshow(img, cmap='gray')
    ax[0].set_axis_off()

    ax[1].set_title(f'Gauss Filter with 0 sigma', fontsize = 18)
    ax[1].imshow(img_gauss, cmap='gray')
    ax[1].set_axis_off()

    ax[2].set_title(f'Gauss Filter with 2 sigma', fontsize = 18)
    ax[2].imshow(img_gauss2, cmap='gray')
    ax[2].set_axis_off()

    ax[3].set_title(f'Gauss Filter with 4 sigma', fontsize = 18)
    ax[3].imshow(img_gauss3, cmap='gray')
    ax[3].set_axis_off()

gauss2_filter(img1)
gauss2_filter(img2)
gauss2_filter(img3)
gauss2_filter(img4)
gauss2_filter(img5)
gauss2_filter(img6)
gauss2_filter(img7)
gauss2_filter(img8)
gauss2_filter(img9)
gauss2_filter(img10)
gauss2_filter(img11)
gauss2_filter(img12)
```

