BAB VII

RESTORASI CITRA

Materi:

- Restorasi Citra
- *Noise*

Tujuan Praktikum:

- Mahasiswa dapat melakukan mengembalikan citra yang terkena *noise* ke citra semula

A. PENYAJIAN

1. Restorasi Citra

Seperti dalam *image enhancement*, tujuan utama dari teknik restorasi adalah untuk meningkatkan citra dalam arti yang telah ditentukan sebelumnya. Perbedaan dengan *image enhancement* yaitu untuk *image enhancement* lebih subjektif, sementara restorasi citra merupakan proses yang objektif. Restorasi citra merupakan proses untuk mengurangi atau menghilangkan degradasi pada citra. Degradasi yang muncul dapat berupa *noise* atau efek optik atau distorsi saat melakukan akuisisi citra, seperti tidak fokusnya kamera yang menghasilkan citra *blur* [1]. Adapun perbedaan *image enhancement* dan *image restoration* dapat dilihat pada Tabel 1.

Tabel 1 Tabel perbandingan *image enhancement* dan *image restoration*

Image Enhancement	***Image Restoration***
Bersifat subjektif	Bersifat objektif
Digunakan untuk memanipulasi citra	Merekonstruksi citra untuk memperbaiki kualitas citra
Memperbaiki tampilan citra untuk tujuan tertentu	Memperbaiki citra yang sudah terkena *noise*

Degradasi citra dapat dimodelkan seperti berikut:

$$g(x, y) = h(x, y) * f(x, y) + n(x, y)$$

Keterangan:

g(x,y) = Gambar yang terdegradasi
h(x,y) = representasi spasial dari fungsi degradasi
* = *convolution, spatial filtering*
f(x,y) = input gambar
n(x,y) = additive noise term

Pada model tersebut, dilakukan *blur filtering* h(x, y) pada citra asli f(x, y). Kemudian terdapat fungsi *random error* n(x, y) atau kita kenal dengan *noise*. Maka, untuk menghasilkan kembali citra asli kita dapat melakukannya seperti berikut,

$$f(x, y) = g(x, y) - n(x, y) / h(x, y)$$

Contoh model degradasi diatas dapat direstorasi pada domain spasial, tetapi ada juga *noise* yang hanya bisa direstorasi di domain frekuensi. Biasanya restorasi citra dilakukan karena kita sudah mengetahui dan menduga model *noise* yang terdapat pada citra, sehingga penanganannya akan berbeda-beda tergantung model *noise*.

2. *Noise*

Sumber utama *noise* dalam gambar digital muncul selama akuisisi dan/atau transmisi gambar. Kinerja sensor pencitraan dipengaruhi oleh berbagai faktor lingkungan selama akuisisi gambar. Misalnya, dalam memperoleh gambar dengan kamera, level cahaya dan suhu sensor merupakan faktor utama yang mempengaruhi jumlah *noise* pada gambar yang dihasilkan. Gambar rusak selama transmisi terutama karena gangguan pada saluran transmisi. Misalnya, gambar yang dikirim menggunakan jaringan nirkabel mungkin rusak oleh petir atau gangguan atmosfer lainnya [1].

Noise merupakan degradasi pada sinyal citra yang terjadi akibat gangguan eksternal. *Error* yang terlihat akan bergantung pada jenis gangguan yang terjadi. Sumber dari *noise* pada citra dapat dilihat sebagai berikut:

- *Error* muncul ketika suatu citra dikirim secara elektronik dari suatu tempat ke tempat lain melalui satelit, kabel, atau nirkabel
- Sensor panas ketika mengambil gambar
- Nilai ISO yang tinggi pada kamera yang menyebabkan sensor kamera lebih cepat menangkap cahaya
- Ukuran sensor pada kamera.

Noise merupakan *random error* yang dapat dimodelkan sebagai fungsi aditif seperti yang dimodelkan pada model degradasi diatas. Beberapa tipe *noise* yang diperkenalkan adalah *Salt* and *pepper noise, gaussian noise*, *speckle noise*, dan *periodic noise*.

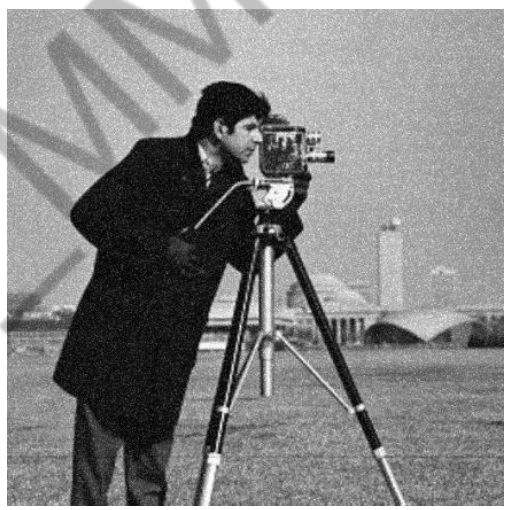
2.1. Salt and Pepper Noise

Secara visual, *noise salt* & *pepper* terdiri dari piksel hitam dan putih yang tersebar pada seluruh citra. Sering disebut *noise impuls*, karena disebabkan oleh gangguan yang tiba-tiba pada sinyal citra. *Noise* ini dapat diatasi dengan *filter mean*, dan *filter median*. Citra *salt and pepper noise* dapat dilihat pada Gambar 1.

Gambar 1 Citra *Salt and pepper noise*

2.2. Gaussian Noise

Penyebab adanya *gaussian noise* adalah fluktuasi acak pada sinyal yang ditambahkan pada citra. *Gaussian noise* merupakan *white noise* ideal yang disebabkan oleh fluktuasi acak dalam sinyal karena penyebarannya terdistribusi normal. *Noise* ini dapat dikurangi dengan *filter* spasial seperti *median filtering*, *gaussian smoothing*, dan *mean filtering*. Jika citra dinotasikan dengan I dan *gaussian noise* dengan N, maka model degradasinya yaitu (I + N). Citra *gaussian noise* dapat dilihat pada Gambar 2.



Gambar 2 Citra *gaussian noise*

2.3. Speckle Noise

Meskipun secara visual *gaussian* dan *speckle noise* terlihat mirip, tetapi keduanya memiliki pemodelan yang berbeda. *Speckle noise* dapat dimodelkan dengan nilai acak yang dikalikan dengan nilai pixel pada citra. Model degradasinya dapat dituliskan dengan persamaan I(1+N). Penanganan

untuk *speckle noise* dengan menggunakan *filter* spasial. Citra *speckle noise* dapat dilihat pada Gambar 3.

Gambar 3 Citra *speckle noise*

2.4. Periodic Noise

Berbeda dengan *noise* lain yang acak, periodic noise merupakan noise yang memiliki pola. Citra yang terdegradasi oleh *noise* ini akan tampak terdapat garis-garis pada citra. Untuk menghilangkan noise ini, dibutuhkan filtering pada domain frekuensi seperti *band reject filtering* dan *notch filter*. Citra *periodic noise* dapat dilihat pada Gambar 4.

Gambar 4 Citra *periodic noise*

B. LATIHAN

Latihan 1: Domain spasial

Berikut ini merupakan latihan membuat program yang dapat menambahkan dan merestorasi *noise* di domain spasial berupa *salt and pepper noise*, *gaussian noise*, dan *speckle noise*.

1. Membuat Noise pada Citra

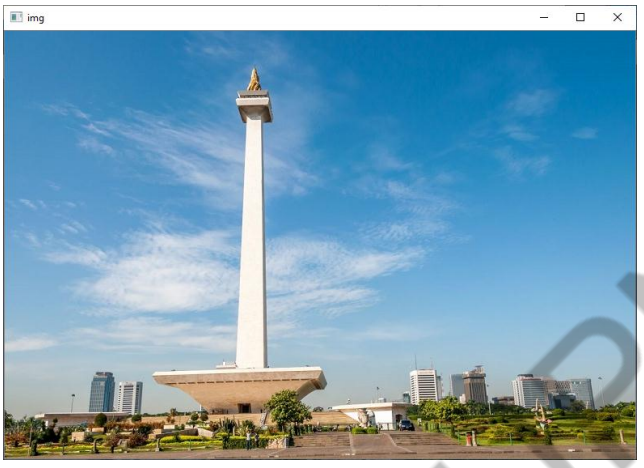
```
import cv2 as cv
import numpy as np
from skimage.util import random_noise

# Load the image
img = cv.imread("upn.jpg")

# Add salt-and-pepper noise to the image.
noise_img_snp = random_noise(img, mode='s&p',amount=0.2)

# Add gaussian noise to the image.
noise_img_gaussian = random_noise(img, mode='gaussian', mean=0, var=0.01)

# Add speckle noise to the image.
noise_img_speckle = random_noise(img, mode='speckle')

# The above function returns a floating-point image on the range [0, 1], thus we changed it to 'uint8'
and from [0,255]
noise_img_snp = np.array(255*noise_img_snp, dtype = 'uint8')
noise_img_gaussian = np.array(255*noise_img_gaussian, dtype = 'uint8')
noise_img_speckle = np.array(255*noise_img_speckle, dtype = 'uint8')

# Display the noise image
cv.imshow('img snp',noise_img_snp)
cv.imshow('img gaussian',noise_img_gaussian)
cv.imshow('img speckle',noise_img_speckle)
cv.waitKey(0)
cv.destroyAllWindows()
```

2. Merestorasi Citra yang telah diberi noise

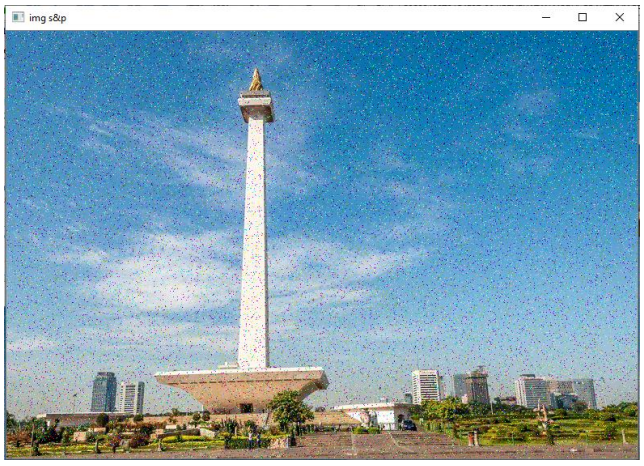
```
import cv2 as cv
import numpy as np
from skimage.util import random_noise

# Load the image
img = cv.imread("monas.jpg")

# Add salt-and-pepper noise to the image.
noise_img_snp = random_noise(img, mode='s&p',amount=0.05)

# Add gaussian noise to the image.
noise_img_gaussian = random_noise(img, mode='gaussian', mean=0, var=0.01)

# Add speckle noise to the image.
noise_img_speckle = random_noise(img, mode='speckle')

# The above function returns a floating-point image on the range [0, 1], thus we changed it to 'uint8'
and from [0,255]
noise_img_snp = np.array(255*noise_img_snp, dtype = 'uint8')
noise_img_gaussian = np.array(255*noise_img_gaussian, dtype = 'uint8')
noise_img_speckle = np.array(255*noise_img_speckle, dtype = 'uint8')

# image reduction
kernel_3_3 = np.ones((3,3),np.float32)/9

img_snp_average_filter = cv.filter2D(noise_img_snp,cv.CV_8U,kernel_3_3,(-1,-1), delta = 0,
borderType = cv.BORDER_DEFAULT)
img_snp_median_median = cv.medianBlur(noise_img_snp,3)

img_gaussian_average_filter = cv.filter2D(noise_img_gaussian,cv.CV_8U,kernel_3_3,(-1,-1), delta = 0,
borderType = cv.BORDER_DEFAULT)
img_gaussian_median_median = cv.medianBlur(noise_img_gaussian,3)

img_speckle_average_filter = cv.filter2D(noise_img_speckle,cv.CV_8U,kernel_3_3,(-1,-1), delta = 0,
borderType = cv.BORDER_DEFAULT)
img_speckle_median_median = cv.medianBlur(noise_img_speckle,3)

# Display the noise image
cv.imshow('img snp',noise_img_snp)
cv.imshow('img gaussian',noise_img_gaussian)
cv.imshow('img speckle',noise_img_speckle)


# Display the image reduction
cv.imshow('img s&p reduction with average filter', img_snp_average_filter)
cv.imshow('img s&p reduction with median filter', img_snp_median_median)

cv.imshow('img gaussian reduction with average filter', img_gaussian_average_filter)
cv.imshow('img gaussian reduction with median filter', img_gaussian_median_median)

cv.imshow('img speckle reduction with average filter', img_speckle_average_filter)
cv.imshow('img speckle reduction with median filter', img_speckle_median_median)

cv.waitKey(0)
cv.destroyAllWindows()
```

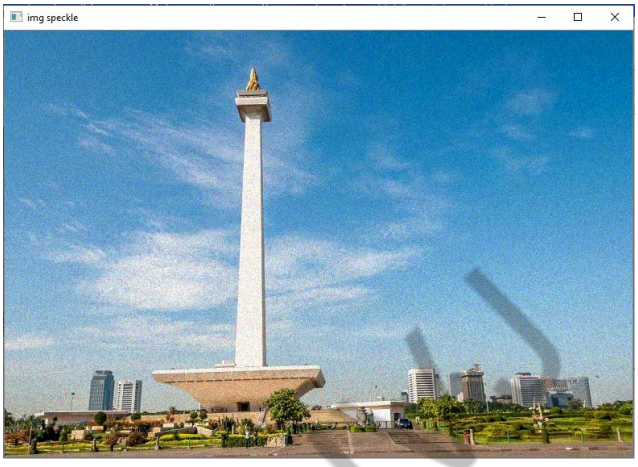
Restorasi yang digunakan pada latihan ini ialah *mean filter*, dan *median filter*. Hasil menjalankan program diatas dapat dilihat pada Gambar 5, Gambar 6, Gambar 7, Gambar 8, dan Gambar 9.

Output

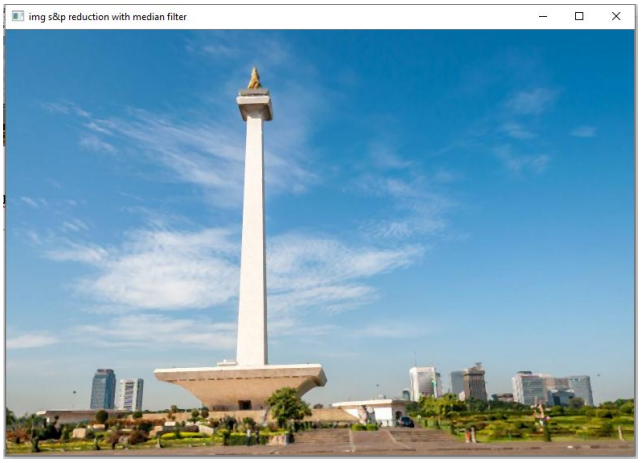


Gambar 5 Citra asli

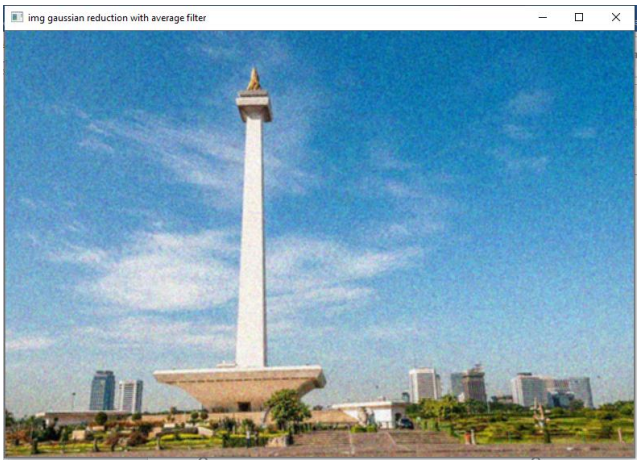


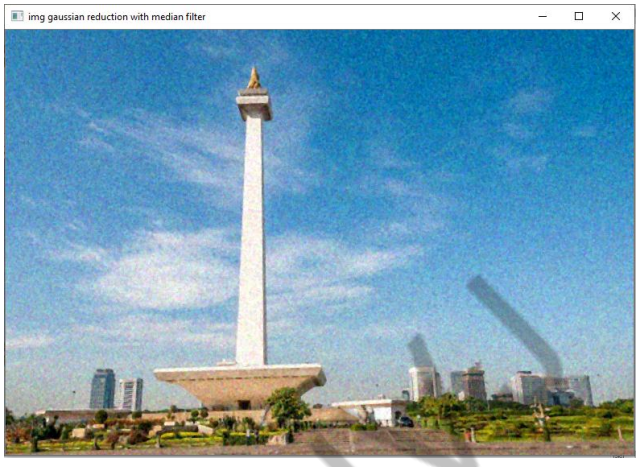


Gambar 6 Citra hasil penambahan *noise*

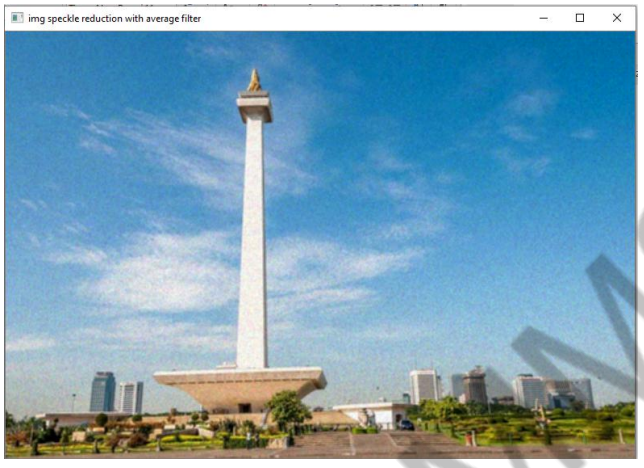


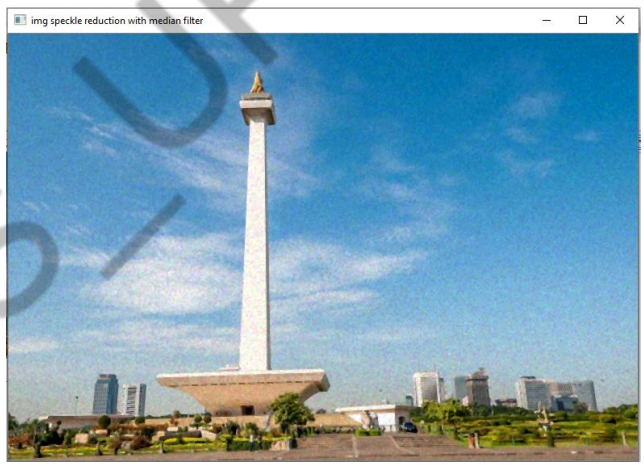
Gambar 7 Citra hasil restorasi pada *salt and pepper noise*





Gambar 8 Citra hasil restorasi pada *gaussian noise*





Gambar 9 Citra hasil restorasi pada *speckle noise*

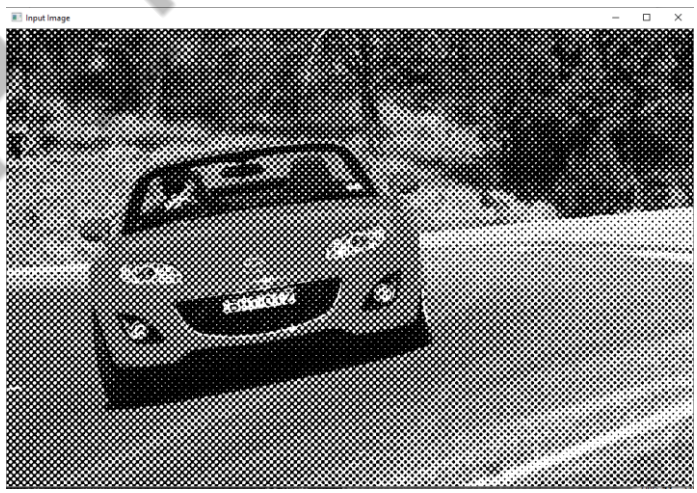
Latihan 2: Domain frekuensi

Berikut ini merupakan latihan membuat program untuk melakukan opearasi low pass filter (LPF), high pass filter (HPF), dan juga merestorasi *periodic noise* di domain frekuensi menggunakan Discrete Fourier Transform (DFT).

```
import numpy as np
import cv2
from matplotlib import pyplot as plt

img = cv2.imread('halftone.png',0)

img_float32 = np.float32(img)

dft = cv2.dft(img_float32, flags = cv2.DFT_COMPLEX_OUTPUT)
dft_shift = np.fft.fftshift(dft)

rows, cols = img.shape
crow, ccol = int(rows/2), int(cols/2)     # center

mask_lpf = np.zeros((rows, cols, 2), np.uint8)
mask_lpf[crow-30:crow+30, ccol-30:ccol+30] = 1

mask_hpf = np.ones((rows, cols, 2), np.uint8)
mask_hpf[crow-50:crow+50, ccol-50:ccol+50] = 0

mask_red = np.ones((rows, cols, 2), np.uint8)

for i in range(60, rows, 135):
  for j in range(100, cols, 200):
    if not (i == 330 and j == 500):
      mask_red[i-10:i+10, j-10:j+10] = 0
for i in range(0, rows, 135):
  for j in range(200, cols, 200):
    if not (i == 330 and j == 500):
      mask_red[max(0,i-15):min(rows,i+15), max(0,j-15):min(cols,j+15)] = 0

fshift_lpf = dft_shift*mask_lpf
f_ishift_lpf = np.fft.ifftshift(fshift_lpf)
img_back_lpf = cv2.idft(f_ishift_lpf)
img_back_lpf = cv2.magnitude(img_back_lpf[:,:,0],img_back_lpf[:,:,1])

fshift_hpf = dft_shift*mask_hpf
f_ishift_hpf = np.fft.ifftshift(fshift_hpf)
img_back_hpf = cv2.idft(f_ishift_hpf)
img_back_hpf = cv2.magnitude(img_back_hpf[:,:,0],img_back_hpf[:,:,1])

fshift_red = dft_shift*mask_red
f_ishift_red = np.fft.ifftshift(fshift_red)
img_back_red = cv2.idft(f_ishift_red)
img_back_red = cv2.magnitude(img_back_red[:,:,0],img_back_red[:,:,1])

I_out_lpf = cv2.normalize(img_back_lpf,None,0,255,cv2.NORM_MINMAX,cv2.CV_8U)
I_out_hpf = cv2.normalize(img_back_hpf,None,0,255,cv2.NORM_MINMAX,cv2.CV_8U)
I_out_red = cv2.normalize(img_back_red,None,0,255,cv2.NORM_MINMAX,cv2.CV_8U)
```

```
magnitude_spectrum = 20*np.log(cv2.magnitude(dft_shift[:,:,0],dft_shift[:,:,1]))

plt.subplot(421),plt.imshow(img, cmap = 'gray')
plt.title('Input Image'), plt.xticks([]), plt.yticks([])
plt.subplot(422),plt.imshow(magnitude_spectrum, cmap = 'gray')
plt.title('Magnitude Spectrum Input Image'), plt.xticks([]), plt.yticks([])

plt.subplot(423),plt.imshow(img_back_lpf, cmap = 'gray')
plt.title('Result Image LPF'), plt.xticks([]), plt.yticks([])
plt.subplot(424),plt.imshow(mask_lpf[:,:,0], cmap = 'gray')
plt.title('Filter LPF'), plt.xticks([]), plt.yticks([])

plt.subplot(425),plt.imshow(img_back_hpf, cmap = 'gray')
plt.title('Result Image HPF'), plt.xticks([]), plt.yticks([])
plt.subplot(426),plt.imshow(mask_hpf[:,:,0], cmap = 'gray')
plt.title('Filter HPF'), plt.xticks([]), plt.yticks([])

plt.subplot(427),plt.imshow(img_back_red, cmap = 'gray')
plt.title('Result Image Noise Reduction'), plt.xticks([]), plt.yticks([])
plt.subplot(428),plt.imshow(mask_red[:,:,0], cmap = 'gray')
plt.title('Filter Noise Reduction'), plt.xticks([]), plt.yticks([])

plt.show()

cv2.imshow('Result Image Filter LPF', I_out_lpf)
cv2.imshow('Result Image Filter HPF', I_out_hpf)
cv2.imshow('Result Image Noise Reduction', I_out_red)
cv2.imshow('Input Image', img)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

Output



Gambar 10 Citra input dengan *periodic noise*

Gambar 11 Citra hasil low pass filter noise di domain frekuensi menggunakan Discrete Fourier Transform (DFT).

Gambar 12 Citra hasil high pass filter noise di domain frekuensi menggunakan Discrete Fourier Transform (DFT).

Gambar 13 Citra hasil restorasi periodic noise di domain frekuensi menggunakan Discrete Fourier Transform (DFT).

C. Lembar Kerja Praktikum

Bung Tomo adalah pahlawan yang terkenal karena peranannya dalam membangkitkan semangat rakyat untuk melawan kembalinya penjajah Belanda melalui tentara NICA, yang berakhir dengan pertempuran 10 November 1945 yang hingga kini diperingati sebagai Hari Pahlawan [2]. Di mesin pencari terdapat salah satu foto Bung Tomo yang telah terdegradasi dengan *noise periodic*. Tugas Kita sebagai mahasiswa informatika yang paham akan ilmu pengolahan citra digital yaitu lakukan restorasi citra terhadap foto Bung Tomo tersebut. Buatlah program dengan menggunakan Bahasa Pemrograman Python dan library Open-CV untuk proses *image reduction* terhadap foto Bung Tomo yang telah terdegradasi dengan *periodic noise*.

Gambar 14. Sutomo.jpg

Referensi

[1] R. C. Gonzalez and R. E. Woods, "Digital Image Processing, 4th Edition," *J. Electron. Imaging*, p. 1019, Jan. 2018.

[2] Wikipedia, "Sutomo," *https://id.wikipedia.org/*, 2020. https://id.wikipedia.org/wiki/Sutomo.