## Ahmad Maulvi Alfansuri

## G64160081

|  |
| --- |
| import cv2  import numpy as np |

Import library yang digunakan pada praktikum ini

|  |
| --- |
| def calculate(b,g,r):  grey = r \* 0.299 + g \* 0.587 + b \* 0.114  return grey  def greyscaling\_image(image):  greyscale = np.zeros((row,col,1), np.uint8) # create dataframe  for y in range(row):  for x in range(col):  # get pixel  b, g, r = image[y, x]  # calculate  grey = calculate(b,g,r)  # assign  greyscale.itemset((y,x,0), grey) # assign  return greyscale |

Fungsi calculate adalah fungsi untuk menkonversi 3 channel pixel menjadi channel greyscale

Fungsi greyscaling\_image melakukan grayscaling pada image warna dan akan mengembalikan image dengan satu channel

|  |
| --- |
| def load\_image(filename="LennaInput.jpg"): # Baca file input  image = cv2.imread(filename) # Baca file  return image # return image object |

Load file gambar dengan nama sebagai argumen. Dan kembalikan image object.

|  |
| --- |
| # Fungsi untuk mendapatkan median dari array of pixel  def get\_median(array\_of\_pixel):  # implementation of median  # Sorting list pixel  array\_of\_pixel.sort()  panjang = len(array\_of\_pixel)  # ambil titik tengah (median)  mid = int(panjang / 2)  new\_pixel = array\_of\_pixel[mid]  # kembalikan nilai pixel baru  return new\_pixel |

Fungsi get\_median akan mengembalikan data yang berada ditengah dan akan membuang outlier. List pertama akan disort dan diambil data yang berada ditengah

|  |
| --- |
| def get\_list\_of\_adjescent(image,y,x,n):  # Inisialisasi list  box = []  # Cari batas untuk seleksi matriks tetangga  bound = n // 2  # nested loop untuk seleksi matriks tetangga  for i in range(-bound,bound+1,1):  for j in range(-bound,bound+1,1):  # Masukin pixel tetangga dan pixel sendiri kedalam list  # print i,j  # print y+i, x+j  try:  box.append(image[y+i, x+j])  except:  print y+i, x+j  exit()  """  Yang dimasukkan kedalam matriks adalah pixel image  image[y-n, x-n] . . image[y-n , x+n]  image[y-n+1,x-n] . . image[y-n+1, x+n]  .  image[y+n, x-1] . . image[y+n , x+n]  """  return box |

Fungsi menerima argumen image, posisi y, posisi x dan dimensi kernel. Dan akan mengembalikan sub\_box dari image yang akan diproses pada kernel

|  |
| --- |
| def create\_kernel(n):  dimension = n # membuat dimensi kernel  kernel = np.ones((dimension,dimension), np.float32) / 1 # buat kernel  return kernel |

Fungsi create\_kernel menerima argumen integer n dan akan mengembalikan array 2d berisi value 1 dengan dimensi n x n

|  |
| --- |
| # Fungsi utama pada median blur  def medianBlur(image, kernel, title="Baru"):  # Ambil dimensi picture  (row, col, chan) = image.shape  (mrow, mcol) = kernel.shape  # Ambil dimensi kernel  dim = mrow  # Ambil batas  bound = int(dim / 2)  # Buat objek baru untuk di return agar tidak menoverwrite gambar sebelumnya  new\_image = image.copy()  for y in range(0, row):  for x in range(0, col):  # Mengambil sub matrix sebanyak n x n dimensi kernel.  # sub matrix harus memenuhi bound  # Jika berada diluar itu isi dengan pixel hitam  if(x < bound or y < bound or y + bound > row - 1 or x + bound > col - 1):  new\_image.itemset((y, x, 0), 0)  continue  # buat list untuk menyimpan pixel tetangga dan pixel sendiri  adj\_box = []  adj\_box = get\_list\_of\_adjescent(image,y,x,dim)  # Mendapatkan median dari list  new\_pixel = get\_median(adj\_box)  new\_image.itemset((y, x, 0), new\_pixel)  cv2.imwrite(title, new\_image)  cv2.imshow(title, new\_image)  return new\_image |

Fungsi medianBlur adalah fungsi utama yang akan dilakukan. Menerima argumen image, kernel dan judul image yang akan disimpan dan dishow

|  |
| --- |
| image = load\_image() # Baca image yang ingin diblur. Dalam tugas adalah LennaInput.jpg  image\_greyscale = greyscaling\_image(image) # Ubah menjadi greyscale |

Load image dan merubah menjadi greyscaling

|  |
| --- |
| image = load\_image() # Baca image yang ingin diblur. Dalam tugas adalah LennaInput.jpg  image\_greyscale = greyscaling\_image(image) # Ubah menjadi greyscale  kernel = create\_kernel(3)  image\_blur = medianBlur(image\_greyscale, kernel, "Blur dengan kernel 3x3.jpg")# Fungsi utama untuk melakukan medianBlur  kernel = create\_kernel(5)  image\_blur = medianBlur(image\_greyscale, kernel, "Blur dengan kernel 5x5.jpg")  kernel = create\_kernel(7)  image\_blur = medianBlur(image\_greyscale, kernel, "Blur dengan kernel 7x7.jpg")    kernel = create\_kernel(3)  image\_blur = medianBlur(image\_greyscale, kernel, "Blur dengan kernel 3x3 sebanyak 1.jpg")  for i in range(2,6):  image\_blur = medianBlur(image\_blur, kernel, "Blur dengan kernel 3x3 sebanyak {}.jpg".format(i))  cv2.waitKey(0) |

Input



Lenna Input



Blur dengan kernel 3x3



Kernel dengan 5x5



Blur dengan kernel 7x7



Blur dengan kernel 3x3 dengan 1 kali iterasi



Blur dengan kernel 3x3 dengan 2 kali iterasi



Blur dengan kernel 3x3 dengan 3 kali iterasi



Blur dengan kernel 3x3 dengan 4 kali iterasi



Blur dengan kernel 3x3 dengan 5 kali iterasi

Dari hasil gambar yang dihasilkan. MedianBlur dengan kernel kecil dan iterasi banyak mengurangi noise dengan hanya mengubah detail sedikit pada gambar. Sedangkan kernel yang besar menghasilkan gambar yang lebih blur

Script akhir

|  |
| --- |
| import cv2  import numpy as np  from pprint import pprint  # bit.ly/LKP4-PCD  # Ahmad Maulvi Alfansuri  # G64160081  def calculate(b, g, r):  grey = r \* 0.299 + g \* 0.587 + b \* 0.114  return grey  def greyscaling\_image(image):  (row, col, chan) = image.shape  print (image.shape)  greyscale = np.zeros((row,col,1), np.uint8)  for y in range(row):  for x in range(col):  # get pixel  b, g, r = image[y, x]  # calculate  grey = calculate(b,g,r)  # assign  greyscale.itemset((y,x,0), grey) # assign  return greyscale  def load\_image(filename="LennaInput.jpg"):  image = cv2.imread(filename)  return image  # Fungsi untuk mendapatkan median dari array of pixel  def get\_median(array\_of\_pixel):  # implementation of median  # Sorting list pixel  array\_of\_pixel.sort()  panjang = len(array\_of\_pixel)  # ambil titik tengah (median)  mid = int(panjang / 2)  new\_pixel = array\_of\_pixel[mid]  # kembalikan nilai pixel baru  return new\_pixel  # Fungsi utama pada median blur  def medianBlur(image, kernel, title="Baru"):  # Ambil dimensi picture  (row, col, chan) = image.shape  (mrow, mcol) = kernel.shape  # Ambil dimensi kernel  dim = mrow  # Ambil batas  bound = int(dim / 2)  # Buat objek baru untuk di return agar tidak menoverwrite gambar sebelumnya  new\_image = image.copy()  for y in range(0, row):  for x in range(0, col):  # Mengambil sub matrix sebanyak n x n dimensi kernel.  # sub matrix harus memenuhi bound  # Jika berada diluar itu isi dengan pixel hitam  if(x < bound or y < bound or y + bound > row - 1 or x + bound > col - 1):  new\_image.itemset((y, x, 0), 0)  continue  # buat list untuk menyimpan pixel tetangga dan pixel sendiri  adj\_box = []  adj\_box = get\_list\_of\_adjescent(image,y,x,dim)  # Mendapatkan median dari list  new\_pixel = get\_median(adj\_box)  new\_image.itemset((y, x, 0), new\_pixel)  cv2.imwrite(title, new\_image)  cv2.imshow(title, new\_image)  return new\_image  def get\_list\_of\_adjescent(image,y,x,n):  # Inisialisasi list  box = []  # Cari batas untuk seleksi matriks tetangga  bound = n // 2  # nested loop untuk seleksi matriks tetangga  for i in range(-bound,bound+1,1):  for j in range(-bound,bound+1,1):  # Masukin pixel tetangga dan pixel sendiri kedalam list  # print i,j  # print y+i, x+j  try:  box.append(image[y+i, x+j])  except:  print y+i, x+j  exit()  """  Yang dimasukkan kedalam matriks adalah pixel image  image[y-n, x-n] . . image[y-n , x+n]  image[y-n+1,x-n] . . image[y-n+1, x+n]  .  image[y+n, x-1] . . image[y+n , x+n]  """  return box  def create\_kernel(n):  dimension = n # membuat dimensi kernel  kernel = np.ones((dimension,dimension), np.float32) / 1 # buat kernel  return kernel  image = load\_image() # Baca image yang ingin diblur. Dalam tugas adalah LennaInput.jpg  image\_greyscale = greyscaling\_image(image) # Ubah menjadi greyscale  kernel = create\_kernel(3)  image\_blur = medianBlur(image\_greyscale, kernel, "Blur dengan kernel 3x3.jpg")# Fungsi utama untuk melakukan medianBlur  kernel = create\_kernel(5)  image\_blur = medianBlur(image\_greyscale, kernel, "Blur dengan kernel 5x5.jpg")  kernel = create\_kernel(7)  image\_blur = medianBlur(image\_greyscale, kernel, "Blur dengan kernel 7x7.jpg")    kernel = create\_kernel(3)  image\_blur = medianBlur(image\_greyscale, kernel, "Blur dengan kernel 3x3 sebanyak 1.jpg")  for i in range(2,6):  image\_blur = medianBlur(image\_blur, kernel, "Blur dengan kernel 3x3 sebanyak {}.jpg".format(i))  cv2.waitKey(0) |