**LAPORAN**

**Visi Komputer dan Pengolahan Citra**

Pembahasan Jawaban UTS



Disusun oleh:

Nama : Ahmada Haiz Zakiyil Ilahi

NRP : 1123800010

Kelas : 1 S2 Teknik Elektro

Dosen : Dr Setiawardhana ST, MT

**PROGRAM PASCASARJANA TERAPAN**

**POLITEKNIK ELEKTRONIKA NEGERI SURABAYA**

**2023/2024**

**PEMBAHASAN JAWABAN UTS**

1. **Histogram Equalization (UTS Nomor 2)**

* **Kode Program:**

Program 1. perhitungan distribusi histogram equalization:

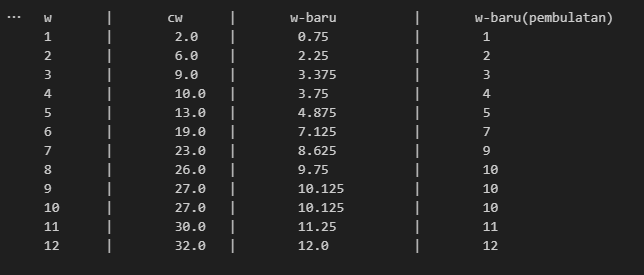
|  |
| --- |
| import numpy as np  h = np.array([2, 4, 3, 1, 3, 6, 4, 3, 1, 0, 3, 2])  cw = np.arange(float(12)) \* 0  wb = np.arange(float(12)) \* 0  cc = 0  print("w\t|\tcw\t|\tw-baru \t\t|\tw-baru(pembulatan)")  for i in range(0, 12):      cc += h[i]      cw[i] = cc      wb = (cw[i] \* 12) / np.sum(h)      print((i+1),"\t|\t", (cw[i]), "\t|\t", (wb), "  \t|\t", (round(wb))) |

Program 2. Menampilkan histogram equalization dengan input gambar :

|  |
| --- |
| # Histogram For Pixel Intensity  import matplotlib.pyplot as plt  import time  import numpy as np  from PIL import Image  # Direktori  Gambar = Image.open('Gambar1.png')  # Original  img = np.array(Gambar)  h, w, c = img.shape  print('width : ', w)  # kolom  print('height : ', h)  # baris  print('channel : ', c)  # channel  print('Data type : ', img.dtype)  # Grayscale  start\_time1 = time.time()  imgGray = np.array(Gambar)  imgGrayE = np.array(Gambar)  h, w, c = imgGray.shape  for x in range(0, w):      for y in range(0, h):          (R, G, B) = imgGray[y, x]          r = int(R)          g = int(G)          b = int(B)          xg = int((r + g + b)/3)          imgGray[y, x] = (xg, xg, xg)          imgGrayE[y, x] = (xg, xg, xg)  stop\_time1 = time.time()  exec\_time1 = (stop\_time1 - start\_time1)\*1000  # Auto Level  start\_time2 = time.time()  imgAuto = np.array(Gambar)  xgmax = np.amin(imgAuto)  xgmin = np.amax(imgAuto)  h, w, c = imgAuto.shape  for x in range(0, w):      for y in range(0, h):          (R, G, B) = imgAuto[y, x]          r = int(R)          if r < xgmin:              xgmin = r          if r > xgmax:              xgmax = r  for x in range(0, w):      for y in range(0, h):          (R, G, B) = imgAuto[y, x]          r = int(R)          xb = int(255\*(r-xgmin)/(xgmax-xgmin))          imgAuto[y, x] = (xb, xb, xb)  stop\_time2 = time.time()  exec\_time2 = (stop\_time2 - start\_time2)\*1000  # CDF dan Histogram Equalization  start\_time3 = time.time()  imgHe = imgGrayE  k = np.arange(float(256))  # Menyiapkan ruang array untuk nilai Histogram  l = np.arange(float(256))  # Menyiapkan ruang array untuk nilai CDF  h, w, c = imgHe.shape  for i in range(0, 256):  # Menyiapkan ruang untuk histogram sebanyak 256      k[i] = 0  for x in range(0, w):  # plot jumlah nilai Histogram H[nilai pixel R]      for y in range(0, h):          (R, G, B) = imgHe[y, x]          r = int(R)          k[r] = k[r] + 1  l[0] = k[0]  for i in range(1, 256):  # plot Kurva CDF nilai RGB 0 sd 255      l[i] = l[i-1] + k[i]  nx = int(w)  ny = int(h)  for x in range(0, w):      for y in range(0, h):          (R, G, B) = imgHe[y, x]          r = int(R)          xe = int(255\*l[r]/nx/ny)          imgHe[y, x] = (xe, xe, xe)  stop\_time3 = time.time()  exec\_time3 = (stop\_time3 - start\_time3)\*1000  # Tampilkan Gambar  fig0 = plt.figure(figsize=(10, 10))  fig0.add\_subplot(2, 2, 1)  plt.imshow(img)  plt.title("Original")  fig0.add\_subplot(2, 2, 2)  plt.imshow(imgGray)  plt.title("Grayscale")  fig0.add\_subplot(2, 2, 3)  plt.imshow(imgAuto)  plt.title("Auto Level")  fig0.add\_subplot(2, 2, 4)  plt.imshow(imgHe)  plt.title("Histogram Equalization")  plt.show()  # Hasil waktu pemrosesan  print('Waktu Proses Grayscale   : ', exec\_time1, 'mili detik')  print('Waktu Proses Auto-Level  : ', exec\_time2, 'mili detik')  print('Waktu Proses Equalization: ', exec\_time3, 'mili detik')  # Tampilkan Histogram & CDF  hist0, bins0 = np.histogram(img.flatten(), 256, [0, 256])  cdf0 = hist0.cumsum()  cdf\_normalized0 = cdf0 \* hist0.max() / cdf0.max()  plt.subplot(2, 2, 1)  plt.plot(cdf\_normalized0, color='b')  plt.hist(img.flatten(), 256, [0, 256], color='r')  plt.title("Histogram & CDF Original")  plt.xlim([0, 256])  plt.legend(('cdf', 'histogram'), loc='upper left')  print(' ')  print('Hasil Original')  for x in range(10):      print(x, hist0[x], cdf0[x])  hist1, bins1 = np.histogram(imgGray.flatten(), 256, [0, 256])  cdf1 = hist1.cumsum()  cdf\_normalized1 = cdf1 \* hist1.max() / cdf1.max()  plt.subplot(2, 2, 2)  plt.plot(cdf\_normalized1, color='b')  plt.hist(imgGray.flatten(), 256, [0, 256], color='r')  plt.title("Histogram & CDF Greyscale")  plt.xlim([0, 256])  plt.legend(('cdf', 'histogram'), loc='upper left')  print(' ')  print('Hasil Greysacle')  for x in range(10):      print(x, hist1[x], cdf1[x])  hist2, bins2 = np.histogram(imgAuto.flatten(), 256, [0, 256])  cdf2 = hist2.cumsum()  cdf\_normalized2 = cdf2 \* hist2.max() / cdf2.max()  plt.subplot(2, 2, 3)  plt.plot(cdf\_normalized2, color='b')  plt.hist(imgAuto.flatten(), 256, [0, 256], color='r')  plt.title("Histogram & CDF Auto Level")  plt.xlim([0, 256])  plt.legend(('cdf', 'histogram'), loc='upper left')  print(' ')  print('Hasil Auto Level')  for x in range(10):      print(x, hist2[x], cdf2[x])  hist3, bins3 = np.histogram(imgHe.flatten(), 256, [0, 256])  cdf3 = hist3.cumsum()  cdf\_normalized3 = cdf3 \* hist3.max() / cdf3.max()  plt.subplot(2, 2, 4)  plt.plot(cdf\_normalized3, color='b')  plt.hist(imgHe.flatten(), 256, [0, 256], color='r')  plt.title("Histogram & CDF Equalization")  plt.xlim([0, 256])  plt.legend(('cdf', 'histogram'), loc='upper left')  print(' ')  print('Hasil H. Equalization')  for x in range(10):      print(x, hist3[x], cdf3[x])  plt.show() |

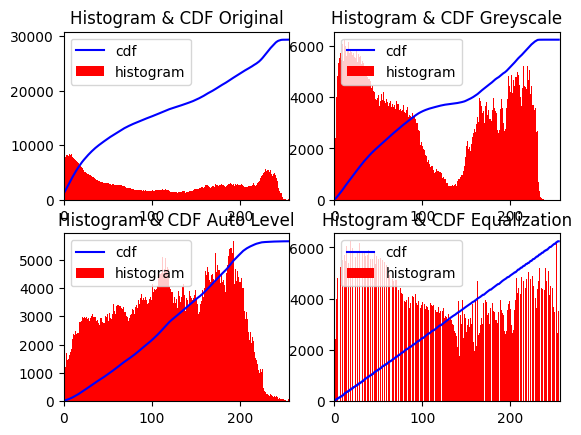
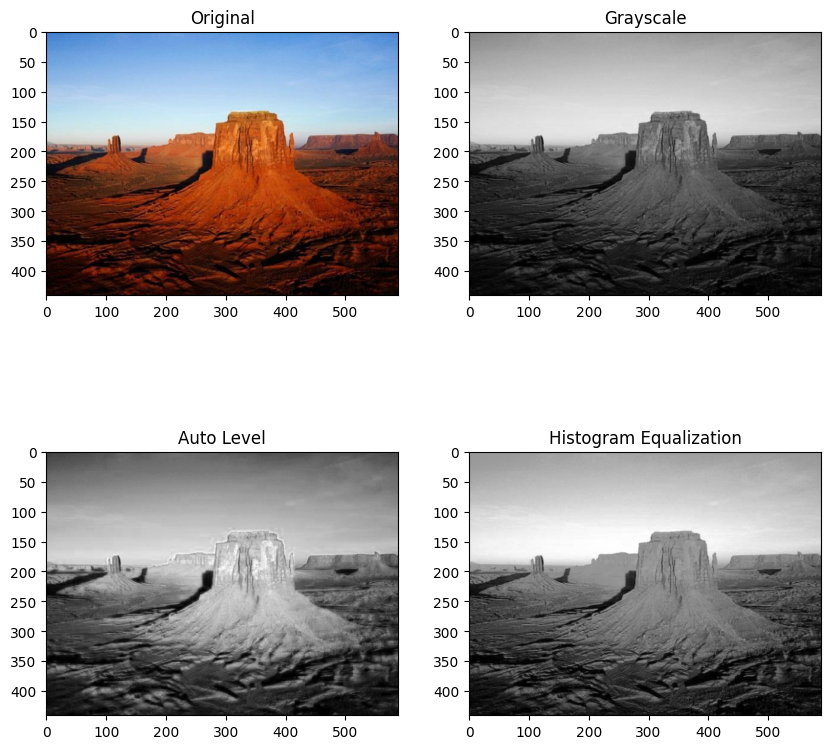
* **Output**:

Program 1:

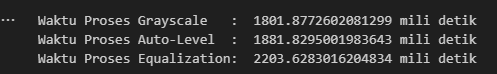


*Gambar hasil perhitugan soal nomor 2*

Program 2:



*Gambar tampilan histogram equalization dan CDF dengan input gambar*



*Gambar hasil waktu komputasi saat konversi gambar*

* **Analisa:**

Pada program pertama, histogram equalization tersebut mengimplementasikan metode equalisasi histogram pada distribusi diskrit yang direpresentasikan oleh array h. Variabel cw digunakan untuk menyimpan cumulative weight, wb untuk menyimpan bobot tertimbang, dan cc sebagai variabel bantu untuk mengakumulasi nilai distribusi diskrit. Program melakukan iterasi melalui array h, menghitung cumulative weight (cw), bobot tertimbang (wb), dan mencetak hasilnya dalam format yang terstruktur. Proses ini memperlihatkan perhitungan bobot tertimbang pada setiap titik distribusi, yang dapat digunakan untuk meratakan histogram dan meningkatkan kontras gambar.

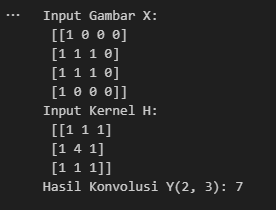
Program kedua, melakukan beberapa operasi pemrosesan citra pada file gambar. Pertama, gambar asli dimuat menggunakan modul PIL (Python Imaging Library), dan informasi dasar seperti lebar (width), tinggi (height), dan jumlah channel (channel) dicetak. Selanjutnya, gambar diubah menjadi citra grayscale, dan proses ini diukur waktunya. Setelah itu, dilakukan penyesuaian otomatis (auto level) terhadap tingkat kecerahan gambar, dan juga diukur waktunya. Terakhir, dilakukan histogram equalization pada citra grayscale, dan waktunya diukur. Semua hasil pemrosesan ditampilkan dalam empat subplot, yaitu gambar asli, grayscale, auto level, dan hasil histogram equalization. Selain itu, program juga menampilkan histogram dan cumulative distribution function (CDF) dari gambar asli dan hasil pemrosesan tersebut. Hasil histogram dan CDF dicetak untuk sepuluh nilai pertama. Waktu eksekusi masing-masing operasi juga dicetak pada akhir program yang menunjukkan bahwa proses komputasi histogram equalization membutuhkan waktu yang lebih lama dari yang lainnya.

1. **Konvolusi Gambar (UTS Nomor 3)**

* **Kode Program:**

|  |
| --- |
| import numpy as np  # Example image and kernel  imageX = np.array([[1, 0, 0, 0],                    [1, 1, 1, 0],                    [1, 1, 1, 0],                    [1, 0, 0, 0]])  print("Input Gambar X:\n",imageX)  kernelH = np.array([[1, 1, 1],                     [1, 4, 1],                     [1, 1, 1]])  print("Input Kernel H:\n",kernelH)  # Coordinates for the point (2, 3)  i, j = 2, 3  # Calculate convolution at (2, 3)  convolution\_result = np.sum(np.sum(kernelH \* imageX[i-2:i+1, j-2:j+1]))  print(f"Hasil Konvolusi Y({i}, {j}): {convolution\_result}") |

* **Output**:



*Gambar hasil perhitungan kovolusi pada persoalan nomor 3*

* **Analisa:**

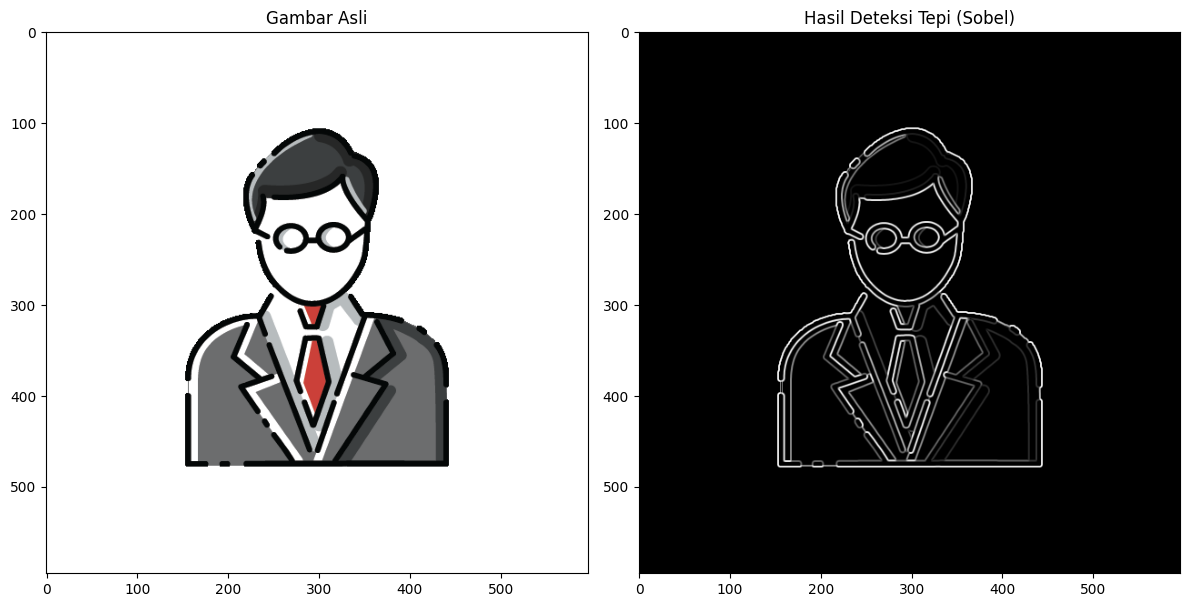
Program ini melakukan operasi konvolusi pada suatu gambar (imageX) menggunakan kernel tertentu (kernelH). Gambar dan kernel diinisialisasi sebagai array NumPy. Gambar dan kernel dicetak untuk memudahkan pemahaman. Kemudian, program menentukan koordinat titik tengah konvolusi, yaitu (2, 3). Hasil konvolusi pada titik tersebut dihitung dengan mengalikan matriks kernel dengan matriks gambar pada area sekitar titik tersebut, dan hasilnya adalah jumlah dari seluruh elemen hasil perkalian. Hasil konvolusi kemudian dicetak menghasilkan nilai 7. Program ini memberikan gambaran sederhana tentang operasi konvolusi pada pemrosesan citra, dan hasilnya dapat digunakan untuk berbagai tujuan, seperti deteksi fitur atau penyaringan gambar.

1. **Metode Sobel (UTS Nomor 4)**

* **Kode Program:**

|  |
| --- |
| import cv2  import numpy as np  import matplotlib.pyplot as plt  # Fungsi untuk melakukan deteksi tepi menggunakan metode Sobel  def sobel\_edge\_detection(image):      # Konversi gambar ke skala abu-abu      gray\_image = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)      # Terapkan filter Sobel      sobel\_x = cv2.Sobel(gray\_image, cv2.CV\_64F, 1, 0, ksize=3)      sobel\_y = cv2.Sobel(gray\_image, cv2.CV\_64F, 0, 1, ksize=3)      # Hitung magnitudo gradien      magnitude = np.sqrt(sobel\_x\*\*2 + sobel\_y\*\*2)      # Normalisasi magnitudo ke rentang 0-255      magnitude = np.uint8(255 \* magnitude / np.max(magnitude))      return magnitude  # Ubah path gambar sesuai dengan file "foto.png"  image\_path = "foto.png"  # Membaca gambar yang sudah didefinisikan  image = cv2.imread(image\_path)  # Melakukan deteksi tepi menggunakan metode Sobel  edge\_image = sobel\_edge\_detection(image)  # Menampilkan gambar asli dan hasil deteksi tepi  plt.figure(figsize=(12, 6))  plt.subplot(1, 2, 1)  plt.imshow(cv2.cvtColor(image, cv2.COLOR\_BGR2RGB))  plt.title('Gambar Asli')  plt.subplot(1, 2, 2)  plt.imshow(edge\_image, cmap='gray')  plt.title('Hasil Deteksi Tepi (Sobel)')  plt.tight\_layout()  plt.show() |

* **Output**:



*Gambar deteksi garis tepi menggunakan sobel pada input gambar*

* **Analisa:**

Program ini melakukan deteksi tepi pada suatu gambar menggunakan metode Sobel. Fungsi sobel\_edge\_detection didefinisikan untuk melakukan langkah-langkah deteksi tepi. Pertama, gambar diubah menjadi skala abu-abu, kemudian filter Sobel diterapkan secara terpisah pada sumbu x dan y. Magnitudo gradien dihitung dengan menggabungkan hasil dari kedua filter Sobel, dan hasilnya dinormalisasi ke rentang 0-255. Program kemudian membaca gambar dari path yang telah ditentukan, menerapkan fungsi deteksi tepi, dan menampilkan gambar asli serta hasil deteksi tepi menggunakan Matplotlib. Hasil deteksi tepi ditampilkan dalam skala abu-abu untuk menyoroti tepi-tepi gambar.