**PRAKTIKUM 5**

**HISTOGRAM**

1. Download citra car.png pada LMS

2. Baca citra car.png dalam format grayscale

3. Buatlah sebuah fungsi contrastStretching tanpa menggunakan fungsi OpenCV pada car.png (Fungsi opencv yang diperbolehkan: imread, waitKey, imshow, dan membaca citra grayscale).

4. Buatlah sebuah fungsi histogramEqualization tanpa menggunakan fungsi openCV dan lakukan proses histogram equalization pada car.png (Fungsi opencv yang diperbolehkan: imread, waitKey, imshow, dan membaca citra grayscale).

5. Algoritme histogramEqualization:

• Hitung jumlah kemunculan piksel setiap nilai derajat keabuan

• Hitunglah peluang nilai kemunculan setiap nilai derajat keabuan (normalized histogram)

• Hitung histogram kumulatif

• Lakukan histogram equalization

6. Tampilkan histogram hasil contrast stretching, histogram citra original, normalized histogram, cumulative histogram, dan equalized histogram

7. Berikan penjelasan singkat terhadap citra hasil histogram equalization dan bandingkan terhadap hasil contrast stretching.

|  |
| --- |
| **import** matplotlib.pyplot **as** plt **import** cv2 **import** numpy **as** np  img = cv2.imread(**"car.png"**,0) row, col = img.shape  *# print(np.array(frek))* max= np.max(img) min= np.min(img)  **def** ContrastStreching(img,max,min):  row, col = img.shape *# mengambil nilai dimensi dari "gambar"* canvas = np.zeros((row, col, 1), np.uint8)  **for** i **in** range(0, row):  **for** j **in** range(0, col):  hasil = ((img[i,j]-min)/(max-min))\*255  canvas.itemset((i,j,0), hasil)  **return** canvas  **def** HistogramChannel(img):  plt.hist(img.ravel(), 256, [0, 256])  **return** plt.show()  **def** equalization(img):  row, col = img.shape  nilaiPixel = [0]\*256  BIN = 256  canvas = np.zeros((row,col,1), np.uint8)  *#Standard Histogram = hitung banyaknya kemunculan tiap intensitas keabuan* **for** i **in** range(0, row):  **for** j **in** range(0, col):  index = int(img[i, j])  nilaiPixel[index] += 1  *#Normalized Histogram = Peluang nilai kemunculan setiap nilai derajat keabuan* **for** i **in** range(0, BIN):  nilaiPixel[i] = float(nilaiPixel[i])/float(row\*col)  *#Histogram Kumulatif* **for** i **in** range(0, BIN):  nilaiPixel[i] = nilaiPixel[i] + nilaiPixel[i - 1]  *#Histogram Equalization* **for** i **in** range(0, BIN):  nilaiPixel[i] = nilaiPixel[i] \* (BIN - 1)  *#memasukan nilai-nilai pixel di atas ke kanvas* **for** i **in** range(0, row):  **for** j **in** range(0, col):  index = int(img[i, j])  pixel = nilaiPixel[index]  canvas.itemset((i,j,0), pixel)   **return** canvas  *#Citra Original* cv2.imshow(**"citra original"**,img) HistogramChannel(img)  *#Contrastreching* contraststrech=ContrastStreching(img,max,min) cv2.imshow(**"Hasil ContrastStreching"**,contraststrech) HistogramChannel(contraststrech)  *#Histogram Equalization* equalization = equalization(img) cv2.imshow(**"Hasil Equalization"**, equalization) HistogramChannel(equalization)   cv2.waitKey(0) cv2.destroyAllWindows() |

**Citra Original**

|  |
| --- |
|  |

HASIL Contrast Streching

|  |
| --- |
|  |

**HASIL Histogram Equalization**

|  |
| --- |
|  |

Perbedaan antara Histogram Contrast Streching dan Histogram Equalization, terlihat bahwa histogram **contrast streching** memiliki kerenggangan yang sama antar persebaran frekuensi pixel, sedangkan pada **Histogram equalization** terlihat setiap persebaran frekuensi pixel memiliki kerenggangan yang berbeda. Hal ini dikarenakan pada **Equalization dimana distribusi nilai derajat keabuan pada suatu citra dibuat rata.** Lalu untuk dapat melakukan histogram equalization ini diperlukan suatu fungsi distribusi kumulatif yang merupakan kumulatif dari histogram. Sedangkan pada **Contrast Streching meningkatkan selisih nilai intensitas maksimal dan minimal pada sebuah citra**. Kontras rendah menggunakan sedikit bin, sedangkan kontras tinggi menggunakan banyak bin. Menambah atau mengurangi kontras dapat dilakukan dengan cara meregangkan atau memampatkan histogram.