|  |
| --- |
| **import** **cv2** **import** **math** **import** **numpy** **as** **np** |

Meing-*import library* yang dibutuhkan.

|  |
| --- |
| **def** **contrast\_stretching**(image):  row, col = image.shape  dst = np.zeros((row, col, **1**), np.uint8) |

Mendefinisikan fungsi contrast\_stretching. Didalamnya kita memasukkan jumlah baris pixel ke variabel row dan jumlah kolom pixel ke variable col. Kemudian kita mendefinisakan kanvas dengan nama dst.

|  |
| --- |
| maxi = ~(**1** << **31**) mini = **1** << **31** **for** i **in** range(row):  **for** j **in** range(col):  maxi = max(maxi, image[i][j])  mini = min(mini, image[i][j]) |

Menginisiasi variabel maxi dan mini dengan maxi diisi nilai maksimum dari int dan mini diisi dengan nilai minimum dari int. Kemudian melakukan iterasi pada setiap pixel yang ada di gambar kemudian meng-*update* variabel maxi dan mini nya apabila ada variabel yang lebih besar atau lebih kecil dari variabel maxi dan mini nya.

|  |
| --- |
| **for** i **in** range(row):  **for** j **in** range(col):  dst.itemset((i, j, **0**), ((image[i][j] - mini) / (maxi - mini)) \* **255**)  **return** dst |

Mengisikan tiap nilai pixel yang telah diubah dengan rumus *contrast stretching*, kemudian memasukkan nilainya ke tiap pixel pada kanvas yang telah dibuat sebelumnya. Kemudian mengembalikan gambar kanvas yang telah diisi pixel-pixel nya.

|  |
| --- |
| **def** **drawHistogram**(image):  row, col, dummy = image.shape  data = np.zeros(row \* col) |

Mendefinisikan fungsi untuk menggambar Histogram. Kemudian menampung nilai baris dan kolom ke variabel row dan col. Kemudian mendefinisikan canvas dengan nama data.

|  |
| --- |
| maxi = ~(**1** << **31**) **for** i **in** range(row):  **for** j **in** range(col):  data[image[i][j][**0**]] += **1**; maxi = max(maxi, data[image[i][j][**0**]]) |

Menghitung jumlah tiap value dari pixel. Kemudian mencari pixel dengan data terbanyak menggunakan variabel maxi.

|  |
| --- |
| histogram = np.zeros((row, col, **1**), np.uint8)  **for** j **in** range(col):  histvalue = col - math.ceil(data[j] \* (col - **1**) / maxi)  **for** i **in** range(row-**1**, histvalue-**1**, -**1**):  histogram.itemset((i, j, **0**), **255**)  **return** histogram |

Mendefinisikan kanvas dengan nama histogram. Kemudian melakukan iterasi pada kolom dan baris. Di setiap kolom di cari nilai histogram nya menggunakan rumus. Kemudian nilai histogram tersebut dimasukkan pada pixel histogram gambar. Kemudian gambar histogram dikembalikan.

|  |
| --- |
| **def** **equalization**(image):  row, col = image.shape  dst = np.zeros((row,col,**1**), np.uint8) |

Memasukkan nilai baris dan kolom ke variabel row, col. Kemudian mendefinisikan kanvas sesuai dengan ukuran baris dan kolom gambar.

|  |
| --- |
| pixel=np.zeros(**256**) **for** i **in** range(**0**, row):  **for** j **in** range(**0**, col):  nilai = int(image[i,j])  pixel[nilai]+=**1** |

Mendefinisikan array pixel sebanyak 256. Kemudian melakukan iterasi dan menghitung nilai kemunculan tiap pixel.

|  |
| --- |
| **for** i **in** range(**0**, **256**):  pixel[i] = float(pixel[i]/(row\*col))   **for** i **in** range(**1**, **256**):  pixel[i] = pixel[i]+pixel[i-**1**]   **for** i **in** range(**0**,**256**):  pixel[i]=pixel[i]\*(**256**-**1**) |

Mendapatkan peluang kemunculan tiap pixel. Kemudian mencari histogram kumulatif nya. Kemudian melakukan histogram equalization sesuai dengan rumus.

|  |
| --- |
| **for** i **in** range(**0**, row):  **for** j **in** range(**0**, col):  nilai = int(image[i,j])  nilai = pixel[nilai]  dst.itemset((i,j,**0**),nilai) **return** dst |

Memasukkan nilai pixel yang telah di proses ke kanvas dst. Kemudian mengembalikan gambar dst nya.