TUGAS PENGOLAHAN CITRA DIGITAL

Dosen Pembimbing:

Dr. Ricky Eka Putra, S.Kom., M.Kom.

Disusun oleh:

Cornelius Louis Nathan 23051204085

PROGRAM STUDI S1 TEKNIK INFORMATIKA

UNIVERSITAS NEGERI SURABAYA

2024

KATA PENGANTAR

Puji syukur saya haturkan kepada Tuhan Yang Maha Esa sehingga atas rahmat-Nya. Saya dapat menyelesaikan laporan tugas Pengolahan Citra Digital yang berjudul “Tugas Pengolahan Citra Digital Ekstraksi Matriks Piksel Menggunakkan Python” sebagai tugas pengolahan citra digital tepat pada waktunya.

Adapun tujuan penulisan laporan ini adalah untuk memahami penggunaan Python dalam pengolahan citra digital. Pembuatan laporan ini sekaligus menjadi tugas yang mengisi nilai dari Mata Kuliah Pengolahan Citra Digital.

Penyusunan laporan ini tidak lepas dari bantuan dan dukungan. Sehingga, saya menyampaikan terima kasih kepada pihak yang telah mendukung pembuatan laporan ini. Terutama kepada Bapak Dr. Ricky Eka Putra, S.Kom., M.Kom., selaku Dosen Pembimbing Mata Kuliah Pengolahan Citra Digital, atas petunjuk, didikan, dan arahan yang sangat membantu dalam penyusunan laporan ini.

Saya menyadari bahwasanya penulisan laporan ini jauh dari kata sempurna, sehingga saya menerima saran dan kritik yang membangun dari pembaca guna untuk perbaikan dan perkembangan di masa mendatang. Semoga penyusunan laporan ini juga berguna bagi para pembaca dalam memahami penggunaan Python dalam pengolahan citra digital.

Surabaya, 20 Februari 2025

Penulis

DAFTAR ISI

BAB I

PENDAHULUAN

1.1 Latar Belakang

Seiring dengan perkembangan teknologi, pengolahan citra digital (*digital image processing*) menjadi salah satu bidang yang semakin banyak digunakan dan diterapkan dalam kehidupan sehari-hari, seperti pengenalan wajah, analisis medis, pemrosesan video, dan sistem kecerdasan buatan. Salah satu teknik dasar dalam pengolahan citra adalah ekstraksi matriks piksel, yang memungkinkan analisis lebih lanjut terhadap struktur dan informasi visual dalam sebuah gambar.

Matriks piksel merupakan representasi numerik dari suatu citra digital, di mana setiap elemen dalam matriks merepresentasikan nilai intensitas warna pada suatu titik atau piksel (*pixel*). Dengan melakukan ekstraksi matriks piksel, pengguna dapat melakukan manipulasi gambar, seperti transformasi warna, memberi *filter* atau efek pada gambar, segmentasi, dll. Bahasa pemrograman yang paling banyak digunakan untuk pengolahan citra digital adalah menggunakkan Python. Python banyak digunakan karena menyediakan banyak *library* yang mempermudah proses pengolahan citra digital, yaitu OpenCV, Matplotlib, NumPy.

Untuk memahami lebih dalam mengenai konsep ekstraksi matriks piksel, maka tugas ini ditulis untuk menjelaskan proses ekstraksi tersebut menggunakkan Python. Diharapkan melalui tugas ini, mahasiswa dapat memahami dengan baik mengenai bagaimana sebuah citra direpresentasikan dalam bentuk matriks, serta bagaimana Python dapat digunakan untuk melakukan berbagai operasi dalam pengolahan citra digital.

1.2 Rumusan Masalah

1. Apa itu matriks piksel?
2. Apa saja *library* yang dibutuhkan dalam proses ekstraksi matriks piksel?
3. Bagaimana proses pengekstraksian matriks piksel pada suatu gambar?
4. Bagaimana cara mengetahui resolusi spasial dan tingkat keabuan pada suatu gambar?

1.3 Tujuan

1. Memahami apa itu matriks piksel.
2. Memahami *library* yang dibutuhkan dalam proses ekstraksi matriks piksel.
3. Memahami proses pengekstraksian matriks piksel pada suatu gambar.
4. Memahami cara mengetahui resolusi spasial dan tingkat keabuan pada suatu gambar.

BAB II

KAJIAN PUSTAKA

2.1 Matriks Piksel

Matriks piksel merupakan representasi numerik dari suatu citra digital, di mana setiap elemen dalam matriks merepresentasikan nilai intensitas warna pada suatu titik atau piksel (*pixel*). Matriks piksel memiliki banyak format, seperti *grayscale*, *RGB* (*Red, Green, Blue*), *BGR (Blue, Green, Red*), *CMYK* (*Cyan, Magenta, Yellow, Black*).

Pada format *RGB*, setiap piksel direpresentasikan dengan tiga nilai yang masing-masing menunjukkan tingkat intensitas warna, yaitu merah, hijau, dan biru dengan *range* antara 0 sampai 255. Nilai 0 berarti tidak ada cahaya warna (gelap atau hitam), sedangkan nilai 255 menunjukkan intensitas maksimum (terang). Semakin terang suatu intensitas warna pada suatu nilai, semakin besar angkanya.

Bentuk dari matriks piksel dengan format *RGB*:

[A, B, C]

A = nilai intensitas warna merah

B = nilai intensitas warna hijau

C = nilai intensitas warna biru

Untuk format *grayscale*, setiap piksel direpresentasikan hanya dengan satu nilai yang menunjukkan tingkat intensitas cahaya dengan *range* antara 0 sampai 255. Nilai 0 berarti tidak ada cahaya warna (gelap atau hitam), sedangkan nilai 255 menunjukkan intensitas maksimum (putih). Semakin terang atau putih suatu intensitas warna pada suatu nilai, semakin besar angkanya.

2.2 *Library* yang Dibutuhkan

Untuk mengekstrak nilai matriks piksel menggunakkan Python, Python menyediakan banyak *library*, seperti Matplotlib, OpenCV, Pillow, dan NumPy. Matplotlib dan Pillow membaca matriks piksel dengan format RGB, sedangkan OpenCV akan membaca matriks piksel dengan format BGR. Hal ini dapat terjadi karena OpenCV menempatkan nilai matriks warna biru pada bagian pertama, sementara Matplotlib dan Pillow tetap menggunakkan format RGB.

Walaupun begitu, OpenCV tetap dapat membaca, mengolah dan melakukan pengolahan citra digital dengan baik. NumPy digunakan untuk menangani operasi pada matriks piksel, yaitu mengambil nilai *range of interest* yang digunakan pada format *grayscale*. Pada kesempatan kali ini, saya akan menggunakkan Matplotlib dan OpenCV untuk pengambilan gambarnya karena fungsi yang digunakan hampir sama.

2.3 Proses Pengekstraksian Matriks Piksel

Sebelum melakukan pengekstraksian, hal pertama yang dapat saya lakukan adalah menyiapkan gambar yang ingin diekstrak. Saya menggunakkan foto pribadi saya dengan ukuran 1464 x 976. Saya sengaja memilih menggunakkan *background* berwarna merah untuk menonjolkan nilai piksel warna merah pada matriks piksel yang akan diekstrak.

Saya menjalankan proses pengekstraksian matriks piksel ini dalam *file* berformat .ipynb (Jupyter Notebook) agar hasil output dapat ditampilkan secara bertahap, memungkinkan analisis dan *debugging* secara mudah.

Lalu, kita melakukan pemanggilan *library* yang dibutuhkan

```
import cv2                          # image processing
import matplotlib.pylab as pltlib   # plot and visualization
import numpy as np                  # numerical computing
```

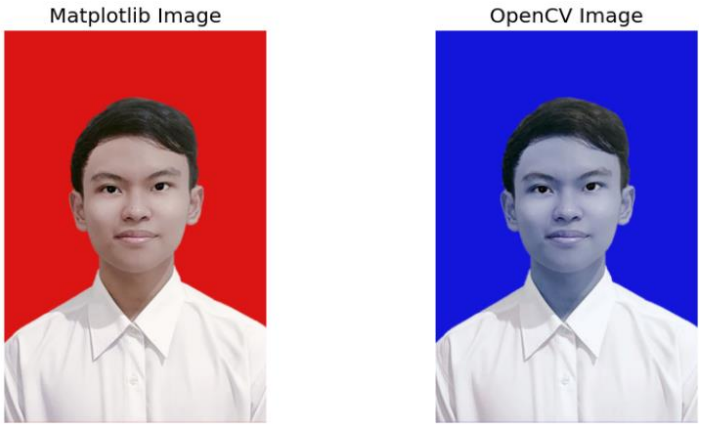
Setelah itu, kita membaca gambar menggunakkan fungsi imread(‘c:/Briefcase/Image Processing/image.jpg’).

```
# set the style for plots to 'ggplot'
pltlib.style.use('ggplot')

# read the image using matplotlib and cv2
img_matplotlib = pltlib.imread('C:\Briefcase\Image Processing\ImageProcess\image.jpg')
img_cv = cv2.imread('C:\Briefcase\Image Processing\ImageProcess\image.jpg')
```

Untuk melihat hasil gambar menggunakkan kedua *library* tersebut, kita menggunakkan fungsi imshow dan menonaktifkan *axis* agar hasil gambar terlihat lebih jelas. *Axis* pada tampilan gambar berfungsi untuk menunjukkan koordinat piksel berdasarkan sumbu x dan y.

```
fig, axs = pltlib.subplots(1,2, figsize = (10,5))
axs[0].imshow(img_matplotlib)
axs[0].axis('off')
axs[0].set_title("Matplotlib Image")
axs[1].imshow(img_cv)
axs[1].axis('off')
axs[1].set_title("OpenCV Image")
pltlib.show()
0.5s
```

Maka *output* nya adalah sebagai berikut



Jika diperhatikan, *OpenCV Image* menampilkan gambar berwarna biru. Hal ini dapat terjadi karena OpenCV secara *default* menggunakkan format *BGR*. Tetapi, format *BGR* tersebut dapat diatur menjadi format *RGB*.

Untuk melakukan pengekstraksian matriks piksel, kita langsung melakukan perintah print() terhadap gambar tersebut.

```
print("Pixel matrix extracting using Matplotlib")
print(img_matplotlib)
print()
print()
print("Pixel matrix extracting using OpenCV")
print(img_cv)
0.5s
```

Maka berikut adalah output nya

```
Pixel matrix using Matplotlib
[[[219  21  20]                                   [219  21  20]
  [219  21  20]                                   [219  21  20]
  [219  21  20]                                   [219  21  20]]
  ...
```

```
 [[219  21  20]
  [219  21  20]
  [219  21  20]
  ...
  [219  21  20]
  [219  21  20]
  [219  21  20]]

 [[219  21  20]
  [219  21  20]
  [219  21  20]
  ...
  [219  21  20]
  [219  21  20]
  [219  21  20]]

 ...

 [[183 150 157]
  [182 149 156]
  [179 146 153]
  ...
  [227 198 194]
  [227 198 194]
  [227 198 194]]

 [[183 140 149]
  [182 139 148]
  [182 136 146]
  ...
  [223 182 180]
  [223 182 180]
  [223 182 180]]]

Pixel matrix using OpenCV
[[[ 20  21 219]
  [ 20  21 219]
  [ 20  21 219]
  ...
  [ 20  21 219]
  [ 20  21 219]
  [ 20  21 219]]

 [[ 20  21 219]
  [ 20  21 219]
  [ 20  21 219]
  ...
  [ 20  21 219]
  [ 20  21 219]
  [ 20  21 219]]

 [[ 20  21 219]
  [ 20  21 219]
  [ 20  21 219]
  ...
  [ 20  21 219]
  [ 20  21 219]
  [ 20  21 219]]

 ...

 [[157 150 183]
  [156 149 182]
  [153 146 179]
  ...
  [194 197 228]
  [194 197 228]
  [194 197 228]]

 [[149 140 183]
  [148 139 182]
  [146 137 180]
  ...
  [180 182 223]
  [180 182 223]
  [180 182 223]]
```

Jika diperhatikan pada bagian pertama matriks, terlihat bahwa Matplotlib menampilkan nilai piksel `[219  21  20]`. Matriks piksel pertama yang ditampilkan adalah matriks piksel di pojok kiri atas gambar. Hal ini sesuai dengan fakta bahwa bagian kiri atas gambar berwarna merah. Karena Matplotlib menggunakan format *RGB (Red, Green, Blue)*, maka nilai tersebut benar mencerminkan warna merah pada gambar.

Sedangkan pada OpenCV, terlihat bahwa OpenCV menampilkan nilai piksel `[ 20  21 219]`. Sama seperti sebelumnya, matriks piksel pertama yang ditampilkan adalah matriks piksel di pojok kiri atas gambar. Namun, karena OpenCV secara *default* menggunakkan format *BGR*,

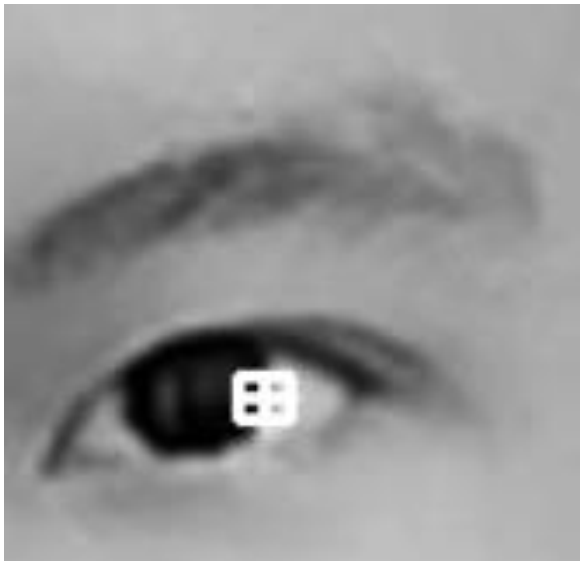
urutan angkanya sedikit berbeda dengan Matplotlib. Meski demikian, nilai intensitas warnanya tetap sama, hanya urutannya yang berbeda.

Untuk melakukan ekstraksi matriks piksel dalam format *grayscale*, langkah pertama yang harus kita pahami adalah *ROI (Region of Interest).* ROI merupakan area spesifik dalam suatu gambar yang ingin kita analisis atau olah lebih lanjut. Dalam pemrosesan citra digital, menentukan *ROI* sangat penting karena memungkinkan kita untuk fokus pada bagian yang relevan, mengabaikan area lain yang tidak diperlukan.

Sebenarnya, kita dapat mengimplementasikan *ROI* ini pada format *RGB*. Tetapi, untuk membuktikan variasi warna pada *ROI*, maka kita tidak menggunakkan *ROI* sebelumnya. Kita menggunakkan numpy untuk menampilkan matriks piksel yang sudah diambil menggunakkan *ROI* dan mengetik program sebagai berikut q

```
import numpy as np

img_cv_gray = cv2.cvtColor(img_cv, cv2.COLOR_BGR2GRAY)

# picking ROI
roi = cv2.selectROI("Set Region of Interest", img_cv_gray, showCrosshair=True, fromCenter=False)

# saving ROI coordinate
x,y,w,h = roi

# crop image based on ROI
cropped_image = img_cv_gray[y:y+h, x:x+w]

cv2.imshow("Cropped Image", cropped_image)

# print matrix pixel with greyscale format
print(np.array(cropped_image))

# close the window
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Ketika program dijalankan, maka akan muncul *windows* yang akan menampilkan gambar yang kita ambil sebelumnya. Lalu, kita dapat memilih *ROI* yang diinginkan. Untuk memperjelas perbedaan nilai matriks pikselnya, saya akan mengambil *ROI* di antara pupil mata dan bagian putih mata. Setelah memilih *ROI*, kita tekan '*Enter'* sebagai konfirmasi input dari gambar, yang akan dikalkulasikan oleh numpy untuk menampilkan nilai matriks pikselnya.



Maka berikut adalah outputnya

```
[[ 25  19  14  10   6  19  53 105 153 182 194 197 194]
 [ 26  21  16  14   4  18  54 107 156 187 200 204 202]
 [ 26  22  19  17   4  18  54 108 159 190 204 208 203]
 [ 25  22  20  19   7  21  57 110 160 191 204 208 204]
 [ 22  21  20  20  16  29  63 114 162 191 203 206 204]
 [ 19  19  19  19  26  38  71 120 166 192 202 204 202]
 [ 17  18  18  19  34  45  77 124 168 193 202 204 197]
 [ 11   7   9  22  36  58  96 138 174 196 203 203 199]
 [  9   7  11  25  47  74 114 153 183 200 204 203 199]
 [  3   3   8  24  60  94 138 172 193 202 203 201 198]
 [  2   2  10  29  73 113 159 188 199 203 201 196 194]]
```

Gambar di atas merupakan matriks piksel dengan format *greyscale*, yang dimana setiap angka merepresentasikan setiap piksel yang diambil pada gambar. Pada bagian kiri, terlihat bahwa nilai matriks memiliki nilai yang jauh lebih kecil dibandingkan dengan nilai matriks di sebelah kanan. Hal ini sesuai dengan konsep warna pada *greyscale*, di mana nilai 0 merepresentasikan warna hitam (tidak ada cahaya), sedangkan nilai 255 merepresentasikan warna putih (ada cahaya). Semakin terang suatu intensitas warna pada suatu piksel, semakin besar angka matriksnya.

Pada bagian *ROI*, terlihat bahwa pupil mata memiliki warna hitam, dengan beberapa nilai matriks mendekati nilai 0. Sedngkan pada bagian putih mata, nilai matriks lebih tinggi dan lebih mendekati nilai 255, sesuai dengan karakteristik format *greyscale*.

2.4 Informasi Resolusi Spasial dan Tingkat Keabuan

Jumlah bit yang diperlukan untuk menyimpan gambar hasil digitasi atau *digital image* dihitung dengan rumus :

$$b = M\ x\ N\ x\ k$$

Dimana , M = jumlah baris *pixel* pada suatu gambar (tinggi gambar)

N = jumlah kolom *pixel* pada suatu gambar (lebar gambar)

k = jumlah bit/*pixel* (menentukan jumlah warna atau *grayscale*)

Nilai M x N adalah resolusi spasial (*spatial resolution*) pada suatu gambar, yang menunjukkan jumlah total pixel pada suatu gambar.

Jumlah tingkat keabuan (*graylevel*) pada suatu gambar dinyatakan dengan nilai

$$L = 2^k$$

Semakin besar k, semakin tinggi tingkat warna yang tersedia, sehingga gambar memiliki semakin banyak detail. Jika k = 1, hanya terdapat 2^1 atau 2 tingkat warna, yaitu hitam dan putih, sehingga disebut gambar biner. Jika k = 8, terdapat 2^8 atau 256 tingkat warna, yang menghasilkan gambar dengan skala *grayscale*. Terdapat k = 16 untuk *High Bit-Depth Grayscale/RGB*, dan k = 32 untuk *True Color RGB*.

Untuk mengetahui ukuran atau *spatial resolution* pada gambar tersebut, kita menggunakkan *shape*. Sedangkan untuk mengetahui *graylevel* pada suatu gambar, kita menggunakkan len(np.unique()) yang menghitung jumlah nilai *pixel* unik pada suatu gambar yang tersimpan.

```
# Spatial Resolution info
# Grayscale Info

fig, ax = pltlib.subplots(figsize = (10,5))
ax.imshow(img_matplotlib)
ax.axis('off')
ax.set_title("Matplotlib Image")

print(img_matplotlib.shape)
L = len(np.unique(img_matplotlib))
print(L)
```

✓ 0.0s

```
(1464, 976, 3)
256
```

Terlihat *output* pertama adalah (1464,976,3), yang berarti *height* gambar adalah 1464 *pixel*, *weight* gambar adalah 976 *pixel*, dan angka terakhir menunjukkan *channel* warna, yaitu

3 (*Red, Green, Blue*). *Channel* disini mengacu pada banyaknya *layer* warna pada suatu gambar, berbeda dengan jumlah bit/*pixel* (k). *Channel* pada gambar tersebut. Dengan demikian, resolusi spasial gambar tersebut adalah 1464 x 976 *pixel*.

Sedangkan *output* terakhir adalah 256, yang menunjukkan jumlah tingkat keabuan atau *graylevel* pada suatu gambar. Ini menunjukkan bahwa gambar memiliki 256 variasi warna, yang merupakan karakteristik dari gambar *grayscale*. Maka, jumlah bit/*pixel* (k) dari gambar tersebut adalah 8, dibuktikan dengan $2^8 = 256$.

Namun, meski nilai *graylevel* menunjukkan skala *grayscale* (256), gambar foto pribadi tersebut memiliki *channel RGB*. Ini berarti bahwa gambar awalnya *grayscale* tetapi disimpan dalam format *RGB.*