LAPORAN TEORI

PENGOLAHAN CITRA DIGITAL

NAMA : Muh.Cholish Said

NIM : 202331076

KELAS : A

DOSEN : Dr. Dra. Dwina Kuswardani, M.Kom

NO.PC :

ASISTEN : 1. Clarenca Sweetdiva Pereira

2. Viana Salsabila Fairuz Syahla

3. Kashrina Masyid Azka

4. Sasikirana Ramadhanty Setiawan Putri

INSTITUT TEKNOLOGI PLN

TEKNIK INFORMATIKA

2025

1. Alur Elemen Pemrosesan Citra Digital:

Alur pemrosesan citra digital pada gambar menunjukkan proses sederhana akuisisi, penyimpanan, dan tampilan citra digital. Prosesnya dimulai dengan citra analog yang diubah menjadi citra digital oleh digitizer. Citra digital kemudian diproses oleh komputer digital, disimpan di media penyimpanan, dan akhirnya ditampilkan pada piranti tampilan.

2. Apa itu splitting citra?

Splitting citra adalah proses pembagian citra menjadi beberapa bagian atau blok yang lebih kecil. Tujuannya beragam, termasuk pemrosesan paralel, pemrosesan lokal, pengurangan kompleksitas, dan analisis regional.

3. Bagaimana cara men-splitting citra?

Cara men-splitting citra bergantung pada kebutuhan dan implementasi. Umumnya, citra dibagi menjadi blok-blok berukuran sama (misalnya, 8x8 piksel). Implementasinya dapat dilakukan dengan perulangan dan pengaksesan array elemen piksel, atau dengan fungsi-fungsi yang tersedia di library pemrograman seperti OpenCV.

4. 0.2989 * r[i,j] + 0.587 * g[i,j] + 0.1141 * b[i,j] Apa maksud baris kode di atas?

Baris kode tersebut menghitung intensitas grayscale dari sebuah piksel RGB. Koefisien 0.2989, 0.587, dan 0.1141 merepresentasikan bobot relatif dari komponen merah (r), hijau (g), dan biru (b) dalam persepsi warna manusia. Rumus ini merupakan pendekatan umum untuk konversi RGB ke grayscale.

5. Apakah kita bisa mengonversi gambar RGB langsung ke gambar binary? Jika iya, berikan alasan. Jika tidak, tahapan apa dulu yang harus dilakukan dan mengapa?

Tidak, kita tidak bisa langsung mengkonversi gambar RGB ke gambar biner. Gambar RGB memiliki tiga saluran warna per piksel, sedangkan gambar biner hanya memiliki satu saluran dengan nilai 0 atau 1. Sebelum konversi, perlu dilakukan proses thresholding. Thresholding menentukan nilai ambang; piksel dengan nilai intensitas di atas ambang menjadi 1, dan di bawah ambang menjadi 0. Proses ini mengurangi informasi warna menjadi informasi biner. Thresholding dapat dilakukan pada setiap saluran warna atau pada intensitas grayscale yang sudah dihitung.

LAPORAN PRAKTIKUM

PENGOLAHAN CITRA DIGITAL

NAMA : Muh.Cholish Said

NIM : 202331076

KELAS : A

DOSEN : Dr. Dra. Dwina Kuswardani, M.Kom

NO.PC :

ASISTEN : 1. Clarenca Sweetdiva Pereira

2. Viana Salsabila Fairuz Syahla

3. Kashrina Masyid Azka

4. Sasikirana Ramadhanty Setiawan Putri

INSTITUT TEKNOLOGI PLN

TEKNIK INFORMATIKA

2025

1.pertama kita import dulu library yang akan digunakan pada praktikum kali ini yaitu cv2 dan numpy as np

```
[1]: import cv2
     import numpy as np
     #MUH.CHOLISH SAID_202331076
```

2. selanjutnya kita membaca gambar yang namanya buah.jpg. citra = cv2.imread('buah.jpg') itu artinya: "Ambil gambar 'buah.jpg', terus simpan datanya ke dalam variabel yang namanya citra

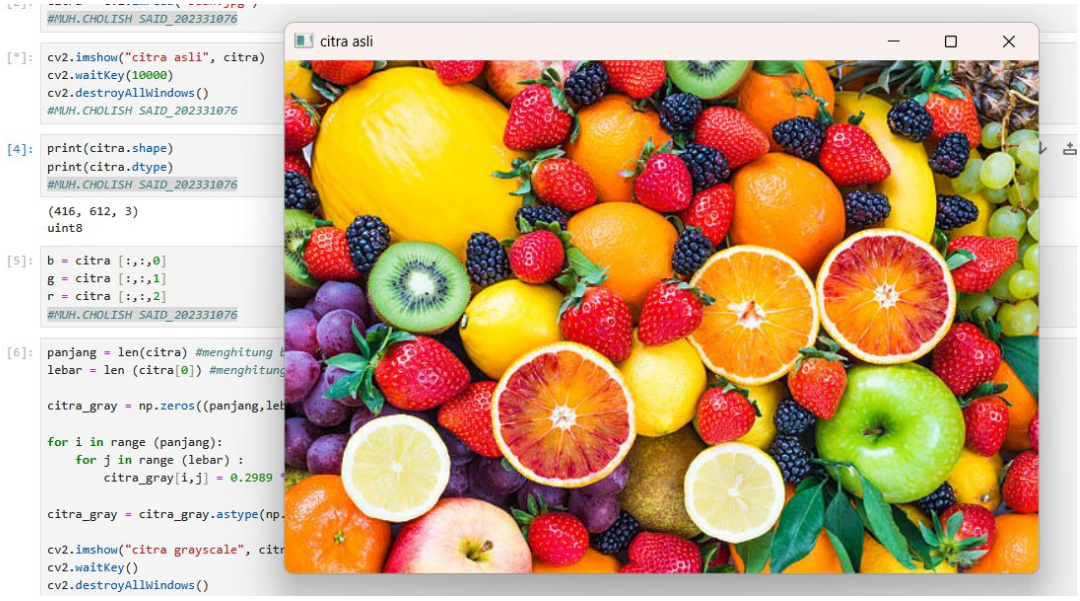
```
[2]: citra = cv2.imread('buah.jpg')
     #MUH.CHOLISH SAID_202331076
```

3. Setelah gambarnya dibaca, sekarang mau ditampilin.

cv2.imshow("citra asli", citra): Ini perintahnya: "Tampilkan gambar yang udah dibaca tadi (citra), dengan judul jendela 'citra asli'." Jadi, bakal muncul jendela baru, isinya gambar buahnya.

cv2.waitKey(10000): Ini buat nahan jendela gambarnya selama 10000 milidetik (10 detik). Jadi, kita bisa liat gambarnya sebelum programnya lanjut. Kalo gak ada ini, jendela gambarnya langsung ilang.

cv2.destroyAllWindows(): Setelah 10 detik, perintah ini bakal nutup jendela gambarnya. Bersih-bersih, biar gak banyak jendela yang terbuka.



4. Kode ini cuma ngecek ukuran dan tipe data gambarnya,

print(citra.shape): Ini buat ngeliat ukuran gambarnya. Hasilnya (416, 612, 3) artinya gambarnya punya tinggi 416 pixel, lebar 612 pixel, dan 3 channel warna (RGB).

print(citra.dtype): Ini buat ngeliat tipe datanya. uint8 artinya datanya berupa bilangan bulat tanpa tanda (unsigned integer) 8-bit. Jadi, setiap pixelnya direpresentasikan pake angka 0-255.

```
[4]: print(citra.shape)
     print(citra.dtype)
     #MUH.CHOLISH SAID_202331076

     (416, 612, 3)
     uint8
```

5. Ini dia bagian pisah-pisah warna. Gambar kan punya warna merah (R), hijau (G), dan biru (B). Kode ini lagi memisahkan ketiga warna itu.

b = citra[:,:,0] : Ambil semua pixel warna biru (0 itu index warna biru) dari gambar (citra), terus simpan ke variabel b.

g = citra[:,:,1] : Sama kayak yang atas, tapi ini ambil warna hijau (1 index warna hijau).

r = citra[:,:,2] : Ini ambil warna merah (2 index warna merah).

Jadi, sekarang kita punya tiga variabel: b, g, dan r, masing-masing berisi data warna biru, hijau, dan merah dari gambar buah.

```
[5]: b = citra [:,:,0]
     g = citra [:,:,1]
     r = citra [:,:,2]
     #MUH.CHOLISH SAID_202331076
```

6. Nah, ini dia bagian utamanya. Kode ini ngubah gambar warna (RGB) jadi gambar hitam putih (grayscale). Cara kerjanya:

- **panjang = len(citra) & lebar = len(citra[0])**: Ngecek ukuran gambar, panjang (tinggi) dan lebarnya berapa pixel.
- **citra_gray = np.zeros((panjang, lebar))**: Buat tempat kosong untuk gambar grayscale-nya, ukurannya sama kayak gambar aslinya, tapi isinya masih nol semua.
- **for i in range(panjang): for j in range(lebar):**: Ini loop (perulangan) buat ngolah setiap pixelnya satu per satu.
- **citra_gray[i,j] = 0.2989 * r[i,j] + 0.587 * g[i,j] + 0.1141 * b[i,j]**: Ini rumus utamanya! Dia ngitung nilai grayscale dari setiap pixel, dengan ngerata-ratain nilai merah, hijau, dan biru, pake bobot tertentu (0.2989, 0.587, 0.1141). Itu bobotnya disesuaikan sama cara mata kita ngeliat warna.
- **citra_gray = citra_gray.astype(np.uint8)**: Ubah tipe datanya jadi uint8, biar pas buat gambar.
- **cv2.imshow(...), cv2.waitKey(), cv2.destroyAllWindows()**: Ini sama kayak di kode sebelumnya, buat menampilkan gambar grayscale-nya selama beberapa saat, terus nutup jendelanya.

Jadi, intinya, kode ini lagi bikin gambar hitam putih dari gambar warnanya. Kayak lagi nge-filter foto di HP, tapi pake rumus matematika.

```
[*]: panjang = len(citra) #menghitung baris
     lebar = len (citra[0]) #menghitung kolom

     citra_gray = np.zeros((panjang,lebar))

     for i in range (panjang):
         for j in range (lebar) :
             citra_gray[i,j] = 0.2989 * r[i,j] + 0.587 * g[i,j] + 0.1141 *b[i,j]
             #merah = 29,89% ; hijau = 58,7% ; biru 11,41%

     citra_gray = citra_gray.astype(np.uint8)

     cv2.imshow("citra grayscale", citra_gray)
     cv2.waitKey()
     cv2.destroyAllWindows()
     #MUH.CHOLISH SAID_202331076
```


```
[7]: print (citra)
     #MUH.CHOLISH SAID_202331076

[[[ 96 224 255]
  [ 78 231 252]
  [140 224 250]
  ...
  [152 161 205]
  [117 148 187]
  [ 72 122 158]]

 [[ 54 217 250]
  [ 56 227 249]
  [137 223 253]
  ...
  [ 45  63  94]
```

7. bagian ini cuma print doang, nggak ada kerjaan berat. Dia cuma nunjukkin isi dari variabel citra. citra itu kan data gambar, nah, kode ini print aja datanya dalam bentuk angka-angka. Angka-angkanya itu ngerepresentasikan nilai warna (RGB) dari setiap pixel di gambar.

```
[8]: print (citra)
     #MUH.CHOLISH SAID_202331076

[[[ 96 224 255]
  [ 78 231 252]
  [140 224 250]
  ...
  [152 161 205]
  [117 148 187]
  [ 72 122 158]]

 [[ 54 217 250]
  [ 56 227 249]
  [137 223 253]
  ...
  [ 45  63  94]
  [ 33  42  75]
  [ 21  24  55]]

 [[ 22 214 251]
  [ 38 226 254]
  [127 227 255]
  ...
  [ 29  72  89]
  [ 42  55  77]
  [ 57  47  70]]

 ...
 [[  1 130 255]
  [  3 132 255]
  [  4 133 255]
  ...
  [ 36 172 254]
  [ 26 168 251]
  [ 15 165 249]]

 [[  0 129 254]
  [  2 131 255]
  [  3 132 255]
  ...
  [ 22 172 255]
  [ 14 169 254]
  [  8 167 254]]

 [[  0 128 253]
  [  0 129 254]
  [  1 130 255]
  ...
  [  7 171 254]
  [  6 171 255]
  [  7 172 255]]]
```

8. Sama kayak yang sebelumnya, ini cuma print aja. Cuma bedanya, yang di-print sekarang adalah isi dari variabel **citra_gray. citra_gray** itu kan gambar hitam putih (grayscale) yang udah diproses dari gambar warna. Nah, kode ini print aja nilai-nilai grayscale dari setiap pixelnya. Angka-angkanya itu ngerepresentasikan tingkat kecerahan (tingkat keabuan) dari setiap pixel.

```
[9]: print (citra_gray)
     #MUH.CHOLISH SAID_202331076

[[218 219 222 ... 173 156 127]
 [208 214 222 ...  70  50  32]
 [203 212 223 ...  72  60  55]
 ...
 [152 154 154 ... 180 176 172]
 [151 153 154 ... 179 176 174]
 [150 151 152 ... 177 177 177]]
```

9. Nah, ini dia proses bikin gambar biner. Bayangin kayak lagi nge-filter foto jadi cuma hitam putih doang, tapi lebih ekstrim.

- panjang, lebar: Sama kayak sebelumnya, ngecek ukuran gambar.

- ambang = 90: Ini nilai ambangnya. Nanti pixel yang lebih terang dari 90 jadi putih, yang lebih gelap jadi hitam.

- citra_biner = np.zeros((panjang, lebar)): Buat tempat kosong untuk gambar binernya.

- for i in range(panjang): for j in range(lebar):: Loop lagi, ngolah pixel satu per satu.

- if citra_gray[i,j] >= ambang:: Ini bagian inti! Kalo nilai grayscale pixelnya (citra_gray[i,j]) lebih besar atau sama dengan ambang (90), maka...

- citra_biner[i,j] = 1: ...pixelnya jadi putih (1).

- else: citra_biner[i,j] = 0: Kalo gak, pixelnya jadi hitam (0).

- cv2.imshow(...), cv2.waitKey(), cv2.destroyAllWindows(): Sama kayak biasa, buat menampilkan gambar binernya, terus nutup jendelanya.

Jadi, kode ini ngubah gambar grayscale jadi gambar biner (cuma hitam putih), pake metode thresholding. Kayak lagi bikin sketsa gambar, cuma pake komputer.

10. Kode itu cuma print aja lagi, nggak ada kerjaan lain. Yang di-print sekarang adalah isi dari variabel **citra_biner. citra_biner** itu kan gambar biner (hitam putih) yang udah dibuat sebelumnya. Nah, kode ini print aja nilai-nilai dari setiap pixelnya. Nilai 1 berarti putih, nilai 0 berarti hitam.