BAB 05

HISTOGRAM

Pokok bahasan pada bab ini mencakup hal-hal berikut:

- pengenalan histogram;
- instalasi Matplotlib;
- penyajian histogram;
- ekualisasi histogram.

5.1 Pengenalan Histogram

Histogram adalah grafik yang menggambarkan distribusi intensitas pada citra. Di grafik ini, frekuensi setiap nilai intensitas terlihat. Nilai besar menyatakan bahwa piksel-piksel yang mempunyai intensitas tersebut sering muncul di citra. Umumnya, histogram menunjukkan intensitas dari O hingga 255.

Saat ini, histogram menjadi perangkat yang sering dilibatkan untuk melakukan analisis citra. Histogram dapat dijadikan sebagai bahan pertimbangan untuk memilih objek tertentu dan menghilangkan bagian yang lain. Histogram juga bisa dimanfaatkan untuk meningkatkan kontras pada citra melalui ekualisasi histogram. Bahkan, histogram juga bisa digunakan untuk mendeteksi kecacatan suatu benda.

Histogram suatu citra dapat diperoleh dengan menggunakan `cv2. calcHist ()`. Bentuk pemakaian histogram untuk satu citra adalah seperti berikut:

cv2.calcHist(citra, *kanal, cadar, ukuranHist, jangkauan)*

Argumen pertama menyatakan citra yang hendak diproses. Argumen kedua berupa kanal yang hendak diproses. Dalam hal ini, [O] menyatakan kanal untuk komponen biru (B), [1] untuk hijau (G), dan [2] untuk merah (R). Argumen ketiga menentukan cadar area yang hendak diproses. Argumen ini perlu diisi dengan `None` sekiranya seluruh area dalam citra hendak diproses. Argumen keempat menentukan ukuran bin. Umumnya, satu bin menyatakan satu nilai intensitas. Namun, satu bin dapat mencakup beberapa intensitas. Untuk semua nilai intensitas, [256] perlu diberikan pada argumen ini. Argumen kelima menyatakan jangkauan yang diproses. Umumnya, nilai untuk argumen ini berupa [O, 256].

Conteh berikut menunjukkan cara menghitung histogram untuk citra `peppers .png`, dengan `histo` berupa larik berisi 256 elemen.

```
>>> citra = cv2.imread('peppers.png')  4)
>>> histo = cv2.ca1cBist(citra, [O],  None,
[256] , [0, 256]) 4)
>>>
```

5.2 Instalasi Matplotlib

Sayangnya, OpenCV tidak mendukung fitur untuk memvisualkan histogram dengan perintah yang pendek. Oleh karena itu, diperlukan modul lain yang memungkinkan penyajian histogram dengan perintah yang pendek. Matplotlib adalah solusinya.

Modul ini untuk Windows dapat diperoleh melalui:

```
http://www.lfd.uci.edu/~gohlke/pythonlibs/#scipy
```

Berkas yang digunakan pada contoh ini adalah matplotlib-3. 0. l-cp37-cp37m-win32. whl. Pertama-tama, berkas ini perlu diunduh. Setelah itu, instalasi dilakukan dengan menggunakan program pip dan dilakukan di Command Prompt. Caranya adalah seperti berikut.

1. Di Command Prompt, aktifkan folder tempat berkas matplotlib-3.0.l-cp37-cp37m-win32.whl berada. Biasanya, berkas ini diletakkan di folder C: \Users\namaPemakai \Downloads. Maka, perintah yang perlu diberikan berupa:

   ```
   CD C:\Users\namaPemakai\Downloads
   ```

2. Berikan perintah berikut untuk menginstalnya:

   ```
   pip install matplotlib-3.0.1-cp37-cp37m-win32.whl
   ```

3. Tunggu sampai pesan berikut muncul dan *prompt* terlihat lagi:

   ```
   Installing collected packages: six, python-
   dateutil, kiwisolver, cycler, pyparsing, matplotlib
   ```

```
Successfully installed cycler-0.10.0 kiwisolver-
1.0.1 matplotlib-3.0.1 pyparsing-2.3.0 python-
dateutil-2.7.5 six-1.11.0

C:\Users\namaPemakai\Downloads>
```

5.3 Penyajian Histogram

Setelah modul Matplotlib terpasang, penyajian histogram bisa dilakukan. Contoh ditunjukkan berikut ini:

```
>>> import matplotlib.pyplot as plt
>>> citra = cv2.imread('peppers.png')
>>> histo = cv2.calcHist([citra], [0], None,
[256], [0, 256])
>>> plt.plot(histo)
[<matplotlib.lines.Line2D object at 0x099D0E90>]
>>> plt.show()
```

Perintah berikut perlu diberikan sekali dan digunakan untuk memuat pustaka `Matplotlib.pyplot` ke memori komputer dan diberi nama singkat berupa `plt`:

```
import matplotlib.pyplot as plt
```

Perintah ini juga bisa ditulis dengan bentuk seperti berikut:

```
from matplotlib import pyplot as plt
```

Setelah `calcHist()` dipanggil, nilai baliknya digunakan pada:

```
plt.plot(histo)
```

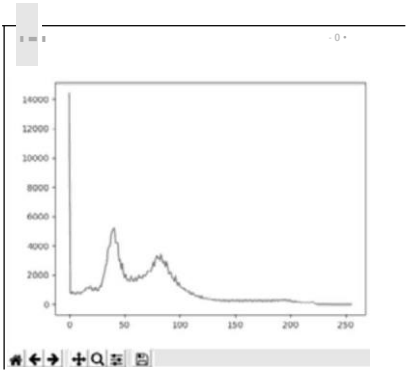
Perintah inilah yang membuat grafik histogram, tetapi masih berada di memori komputer.

Adapun perintah yang menampilkan grafik, yaitu:

```
plt.show()
```

Hasilnya diperlihatkan pada Gambar 5.1.

Jendela grafik tersebut tetap ditampilkan sampai dilakukan pengeklikan pada X.

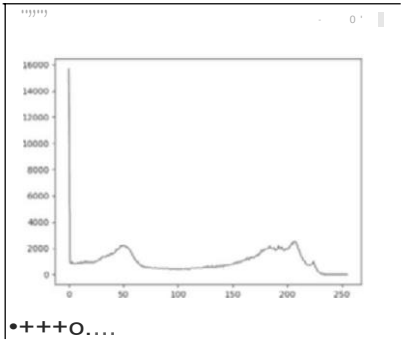


Gambar 5.1 Histogram citra peppers.png kanal biru

Contoh berikut digunakan untuk menampilkan histogram kanal hijau pada citra peppers .png:

```
>>> histo = cv2.calcHist([citra], [1], None,
[256] , [O, 256])
>>> plt.plot(histo)
[<matplotlib.li nes.Line2D object at Ox00770F10>]
>>> plt.show()
```

Pada `calcHist()`,[1] menyatakan kanal hijau. Hasilnya diperlihatkan pada Gambar 5.2.

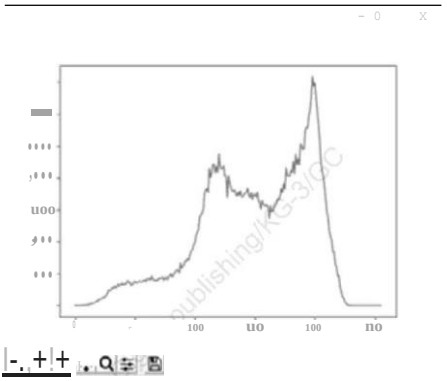


Gambar** 5.g **Histogram citra peppers.png kanal hijau

Contoh berikut digunakan untuk menampilkan histogram kanal merah pada citra peppers.png:

```
>>> histo = cv2.calcHist([citra], [2], None,
[256], [0, 256]) {}
>>> plt.plot(histo) {}
[<matplotlib.lines.Line2D object at 0x099D0CBO>]
>>> plt.show() {}
```

Pada calcHist(), [2] menyatakan kanal merah. Hasilnya diperlihatkan pada Gambar 5.3.



Gambar** 5.9 **Histogram citra peppers.png kanal merah

Histogram ketiga kanal dapat disatukan. Perwujudannya diperlihatkan pada skrip berikut:

Berkas : gabung3.py

```
# Pernbuatan histogram

import cv2
import sys
import matplotlib.pyplot as plt

if len(sys.argv) ==1:
    print('Masukkan nama berkas gambar')
else:
    berkas =sys.argv[l]
```

```
citra = cv2.imread(berkas, cv2.IMREAD_UNCHANGED)
if citra is None:
    print('Tidak dapat membaca berkas', berkas)
else:
    # Cek sebagai citra berwarna atau bukan
    if len(citra.shape) > 2:
       nkali   3
    else:
       nkali   1

    warna = ('b', 'g', 'r')
    for kanal in range(nkali):
        histo = cv2.calcHist([citra], [kanal],
                             None, [256],[0,256])
        plt.plot(histo, color= warna[kanal])

    plt.title('Histogram' + berkas)
    plt.show()

Akhir berkas
```

Skrip ini dapat dipakai untuk menyajikan histogram, baik untuk citra berwarna maupun citra berskala keabu-abuan. Pada citra berwarna, histogram ketiga kanal dipaparkan.

Skrip ini melibatkan tiga modul, yaitu cv2, sys, **dan** pyplot **milik** matplotlib. **Modul** sys **digunakan untuk menangani argumen baris perintah.**

Berkas citra dibaca melalui:

```
citra = cv2.imread(berkas, cv2.IMREAD_UNCHANGED)
```

Konstanta `cv2.IMREAD_UNCHANGED` **dimaksudkan agar citra dibaca apa adanya sehingga untuk citra berskala keabu-abuan tetap mengandung satu kanal, bukan tiga kanal.**

Pernyataan berikut digunakan untuk menugaskan nilai 1 atau 3 ke nkali bergantung pada **jenis citra yang dimuat:**

```
if len(citra.shape) > 2:
    nkali = 3
else:
```

```
    nkali = 1
```

Nilai 3 ditugaskan ke nkali jika citra adalah citra berwarna, sedangkan nilai 1 ditugaskan ke nkali jika citra adalah citra beraras keabu-abuan.

Pernyataan berikut digunakan untuk membuat tupel warna yang berisi tiga elemen yaitu 'b', 'g', dan 'r' yang masing-masing menyatakan warna biru, hijau, dan merah:

```
warna = ('b'' 'g'' 'r')
```

Variabel nkali digunakan pada pernyataan berikut:

```
for kanal in range(nkali):
    histo = cv2.calcHist([citra], [kanal],
                         None, [256],[O,256])
    plt.plot(histo, color= warna[kanal])
```

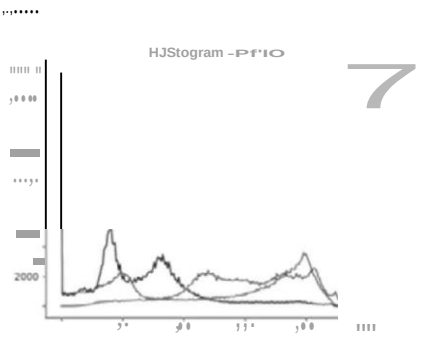
Pernyataan ini digunakan untuk menggambar histogram sebanyak nilai nkali. Tambahan argumen color pada plot (), memungkinkan warna yang digunakan diubah. Pada saat kanal bernilai 0, warna [0] berupa 'b'. Dengan demikian, warna yang digunakan biru. Pada saat kanal bernilai 1, warna [1] berupa 'g'. Dengan demikian, warna yang digunakan hijau. Pada saat kanal bernilai 2, warna [2] berupa 'b'. Dengan demikian, warna yang digunakan merah.

Pernyataan berikut digunakan untuk memberikan judul pada histogram dengan menggabungkan 'Histogram ' dengan nama berkas dan kemudian menampilkan grafik:

```
plt.title('Histogram' + berkas)
plt.show()
```

Gambar 5.4 menujukkan hasil pemanggilan:

```
python histogram.py peppers.png
```

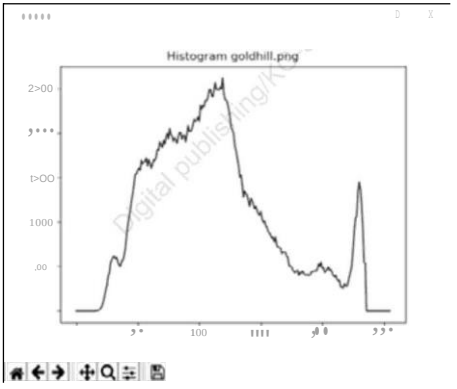


• +.+ +.Q.!.-. Ili,

Gambar d.4 Histogram tiga kanal pada citra peppers.png

Gambar 5.5 menujukkan hasil pemanggilan:

```
python histogram.py goldhill.png
```

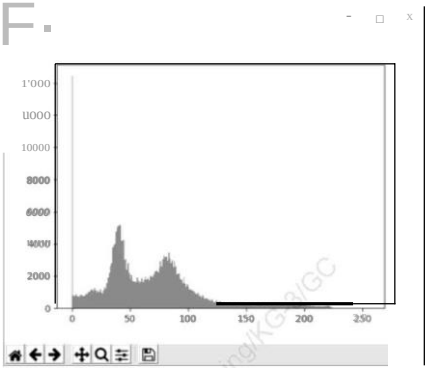


Gambar d.d Histogram citra goldhill.png

Selain menggunakan cara di depan, histogram juga bisa dibuat dengan menggunakan `hist ()` milik `pyplot`. Contoh berikut menunjukkan contoh pembuatan histogram komponen biru pada citra `peppers. png`:

```
>>> dtr = cv2.imread( 'peppers .png')  4)
>>> x = p1t.hist(citra[:, :, O] .ravel(), 256,
[0,256])  4)
>>> p1t. show()  4)
```

Pada contoh ini, ci tra [: , . , 0 J menyatakan kanal O atau kanal warna biru. Adapun metode ravel () digunakan untuk membuat larik satu dimensi. Argumen 256 menyatakan jumlah bin (256 intensitas) dan argumen [O, 256) menyatakan jangkauan intensitas yang diproses. Gambar 5.6 menunjukkan hasil ketika pl t. show () dieksekusi. Tampak bahwa model histogram yang dihasilkan berbeda dengan hasil pl t. plot().



Gambar 5.6 Histogram citra peppers.png kanal biru bentukan plt.histO

5.4 Ekualisasi Histogram

Ekualisasi histogram merupakan suatu cara yang bertujuan untuk memperoleh histogram dengan intensitas terdistribusi secara seragam pada citra. Cara ini ditujukan untuk mendapatkan aras yang lebih luas pada daerah yang memiliki banyak piksel dan mempersempit aras pada daerah berpiksel sedikit. Efeknya dapat digunakan untuk meningkatkan kontras sehingga citra seperti mendapat pencahayaan yang merata. Gambar 5.7 menunjukkan contoh citra asal dan hasil ekualisasi histogram. Gambar asal sangat redup, sedangkan hasilnya citra terlihat kontras. Bahkan, lantai yang semula tidak terlihat, sekarang menjadi tampak jelas.

Gambar d.7 Efek ekualisasi histogram

Ekualisasi histogram pada OpenCV ditangani dengan menggunakan `cv2.equalizeHist()`**.= dan diterapkan hanya untuk citra berskala keabu-abuan. Argumennya berupa citra yang hendak ditingkatkan kontrasnya dan nilai baliknya adalah hasil ekualisasi histogram.**

Contoh berikut memperlihatkan penggunaan metode

11!1 Berkas : ekualisasi.py

```
# Ekualisasi histogram

import cv2
import numpy as np

citra   cv2.imread('gembala.png', 0)
ekual   cv2.equalizeHist(citra)
hasil   np.hstack((citra, ekual))

# Tampilkan hasilnya
cv2.imshow('Hasil', hasil)

waitKey(0)
```

Akhir berkas

Pada skrip ini, modul numpy dilibatkan karena terdapat perintah untuk menggabungkan dua citra secara bersebelahan, yaitu menggunakan vs tack ().

Perintah berikut digunakan untuk membaca citra gembala. png:

```
citra = cv2.imread('gembala.png')
```

Berkas gembala. png dapat diunduh melalui pranala yang disebutkan dalam Kata Pengantar.

Adapun perintah yang melakukan ekualisasi histogram yaitu:

```
ekual = cv2.equalizeHist(citra)
```

Hasil skrip ekualisasi .pydapat dilihat pada Gambar 5.8.

Gambar 5.8 Ekualisasi histogram pada citra gembala.png

Penerapan ekualisasi histogram pada citra berwarna dapat dilakukan dengan mula-mula mengonversi citra berformat BGR ke format LAB. Selanjutnya, equalizeHist () diterapkan pada kanal pencahayaan (L) atau kanal 0. Hasilnya perlu dikonversi balik ke format BGR. Perwujudannya seperti berikut:

Berkas : ekualwarna.py

```
# Ekualisasi histogram untuk citra berwarna

import cv2
import numpy as np

citra = cv2.imread('taipei101.png')

lab= cv2.cvtColor(citra, cv2.COLOR_BGR2LAB)

kanalLAB = cv2.split(lab)

kanalLAB[0] = cv2.equalizeHist(kanalLAB[0])

lab   cv2.merge(kanalLAB)

bgr   cv2.cvtColor(lab, cv2.COLOR_LAB2BGR)

hasil = np.hstack((citra, bgr))

# Tampilkan hasilnya
cv2.imshow('Hasil',  hasil)

waitKey(0)
```

Akhir berkas

Pada skrip ini, perintah berikut digunakan untuk membaca citra taipeilOl .png**:**

```
citra = cv2.imread('taipei101.png')
```

Berkas taipeilOl .png **dapat diunduh melalui pranala yang disebutkan dalam Kata Pengantar.**

Perintah berikut digunakan untuk mengonversi ci tra **yang berformat BGR ke** lab **yang berformat LAB:**

```
lab= cv2.cvtColor(citra, cv2.COLOR_BGR2LAB)
```

Adapun perintah berikut digunakan untuk mengenakan ekualisasi histogram pada kanal 0:

```
kanalLAB[0] = cv2.equalizeHist(kanalLAB[0])
```

Kemudian, ketiga kanal disatukan kembali melalui:

```
lab= cv2.merge(kanalLAB)
```

Terakhir, citra berformat LAB dikonversi balik ke citra berformat BGR. Perintahnya berupa:

```
bgr = cv2.cvtColor(lab, cv2.COLOR_LAB2BGR)
```

Dengan demikian, bgr adalah citra berwarna dalam format BGR yang telah mengalami ekualisasi histogram. Hasilnya dapat dilihat pada Gambar 5.9.

Gambar 5.9 Ekualisasi histogram pada citra berwarna

Hasil ekualisasi histogram seringkali menimbulkan efek kontras yang berlebihan sehingga bagian tertentu pada hasil menjadi tidak alami. Untuk mengatasi hal ini, terdapat pendekatan yang dinamakan ekualisasi histogram adaptif yang dinamakan CLAHE *(Contrast limiting adaptive histogram equalization),* dengan menerapkan ekualisasi histogram pada area yang kecil, misalnya 8 x 8 piksel dan menerapkan pemotongan kontras yang melebihi suatu nilai. Jadi, pada cara ini terdapat dua parameter, yaitu ukuran segmen untuk penerapan

ekualisasi histogram dan limit kontras. Di OpenCV, hal ini dilaksanakan dengan menggunakan cv2. createCLAHE () dengan argumen berupa clipLimi t untuk menentukan ambang pemotongan piksel dan titleGridSize untuk menentukan ukuran segmen gambar.

Contoh berikut menunjukkan penggunaan CLAHE dengan clipLimi t sama dengan 12 dan titleGridSize sebesar 8 x 8:

11!1 Berkas : ekualw.irn.i.11v

```
# Ekualisasi histogram menggunakan CLAHE

import cv2
import numpy as np

citra   cv2.imread('gembala.png', 0)
clahe   cv2.createCLAHE(clipLimit = 12,
                        tileGridSize = (8,8))
ekual   clahe.apply(citra)
hasil   np.hstack((citra, ekual))

# Tampilkan hasilnya
cv2.imshow('Hasil', hasil)

    waitKey(O)        Q
```

Akhir berkas

Penentuan dua parameter CLAHE dilakukan melalui:

```
clahe = cv2.createCLAHE(clipLimit = 12,
                        tileGridSize = (8,8))
```

Kemudian, ekualisasi histogram dilaksanakan melalui:

```
ekual = clahe.apply(citra)
```

Dengan demikian, ekual berisi citra yang telah mengalami ekualisasi histogram. Hasilnya diperlihatkan pada Gambar 5.10.

Gambar 5.10 Contoh penerapan CLAHE pada citra gembala.png

Penerapan CLAHE pada citra berwarna dapat dilakukan dengan mula-mula mengonversi citra berformat BGR ke format LAB. Selanjutnya, CLAHE diterapkan pada kanal pencahayaan (L) atau kanal 0. Hasilnya perlu dikonversi balik ke format BGR. Perwujudannya seperti berikut:

11!1 Berkas : clahewarna.py

```
# Ekualisasi histogram menggunakan CLAHE
#    untuk citra berwarna

import cv2
import numpy as np

citra = cv2.imread('taipei101.png')
lab= cv2.cvtColor(citra, cv2.COLOR_BGR2LAB)

kanalLAB = cv2.split(lab)

clahe = cv2.createCLAHE(clipLimit = 2.0,
                        tileGridSize = (8, 8))

kanalLAB[0] = clahe.apply(kanalLAB[0])

lab   cv2.merge(kanalLAB)

bgr   cv2.cvtColor(lab, cv2.COLOR_LAB2BGR)

hasil = np.hstack((citra, bgr))

# Tampilkan hasilnya
cv2.imshow('Hasil', hasil)

.waitKey(0)
```

Akhir berkas

Pada skrip ini, perintah berikut digunakan untuk membaca citra taipeil01.png:

```
citra = cv2.imread('taipeil01.png')
```

Berkas taipeil01.png dapat diunduh melalui pranala yang disebutkan dalam Kata Pengantar.

Perintah berikut digunakan untuk mengonversi citra yang berformat BGR ke lab yang berformat LAB:

```
lab= cv2.cvtColor(citra, cv2.COLOR_BGR2LAB)
```

Penerapan dua parameter CLAHE dilakukan melalu:

```
clahe = cv2.createCLAHE(clipLimit = 2.0,
                        tileGridSize = (8, 8))
```

Adapun perintah berikut digunakan untuk mengenakan ekualisasi histogram pada kanal 0:

```
kanalLAB[0] = clahe.apply(kanalLAB[0])
```

Kemudian, ketiga kanal disatukan kembali melalui:

```
lab= cv2.merge(kanalLAB)
```

Terakhir, citra berformat LAB dikonversi balik ke citra berformat BGR. Perintahnya berupa:

```
bgr = cv2.cvtColor(lab, cv2.COLOR_LAB2BGR)
```

Dengan demikian, bgr adalah citra berwarna dalam format BGR yang telah mengalami ekualisasi histogram. Hasilnya dapat dilihat pada Gambar 5.11.

Gambar.5.11 Contohpenerapan CLAHEpada citra berwarna

Tampak bahwa hasil penggunaan CLAHE lebih baik daripada hasil yang menggunakan non-CLAHE (Gambar 5.9), terutama pada bagian pencahayaan di puncak menara.

5.5 Penggunaan subplot()

Mengingat bab ini memperkenalkan penggunaan Matplotlib, khususnya modul pyplot, maka akan diperkenalkan pula penggunaan subplot () yang memungkinkan satu jendela digunakan untuk menampilkan beberapa gambar. Hal ini dapat menjadi alternatif terhadap penggunaan hstack () dan cv2. imshow ().

Metode subplot () menggunakan bilangan bulat untuk menentukan jumlah baris dan kolom pada jendela grafik dan sekaligus nomor urut peletakan citra dalam grafik. Sebagai contoh, perintah berikut menyatakan bahwa terdapat 1 baris dan 2 kolom untuk citra pada jendela grafik dan citra akan diletakkan pada kolom pertama:

```
supbplot(121)
```

Adapun subplot (12 2) berarti terdapat 1 baris dan 2 kolom untuk citra pada jendela grafik dan citra akan diletakkan pada kolom pertama.

Contoh berikut menunjukkan cara meletakkan dua gambar berwarna

ndela grafik:

Berkas : **eku.ilwarn.i.11y**

```
# Satu baris dan dua kolom citra berwarna
#    menggunakan subplot()

import cv2
from matplotlib import pyplot as plt

citra1   cv2.imread('baboon.png')
citra2   cv2.imread('lena.png')

# Ubah BGR menjadi RGB
rgb1   citra1[...,     -1]
rgb2 = citra2[...,  ..  -1]

# Tampilkan gambar
plt.subplot(121)
plt.imshow(rgb1)                0
plt.xticks([]), plt.yticks([])
plt.title('Kolom 1')

plt.subplot(122)
plt.imshow(rgb2)
plt.xticks([]), plt.yticks([])
plt.title('Kolom 2')

show()
```

Akhir berkas

Hal yang perlu diperhatikan dalam menggunakan subplot (), **citra pada OpenCV yang berformat BGR perlu diubah menjadi RGB karena** imshow () **milik** pyplot **menggunakan format RGB. Jika terlupa dalam mengonversi ke RGB, akan diperoleh gambar yang cenderung biru.**

Konversi kedua citra menjadi RGB dilakukan oleh:

```
rgb1   citra1[...,     -1]
rgb2   citra2[...,     -1]
```

Kode ci **tral** [... , : : -1 J berarti membalik urutan yang semula berupa B,G, dan R menjadi R, G, dan B. Selanjutnya, citra **rgbl** dan rgb2 inilah yang akan ditampilkan melalui imshow () milik pyplot.

Perintah berikut digunakan untuk menentukan jumlah baris sebanyak 1 dan jumlah kolom sebanyak 2 dan citra akan diletakkan sebagai urutan pertama:

```
pH.subplot(121)
```

Perintah berikut menampilkan citra rgbl ke area yang telah ditentukan subplot():

```
plt.imshow(rgbl)
```

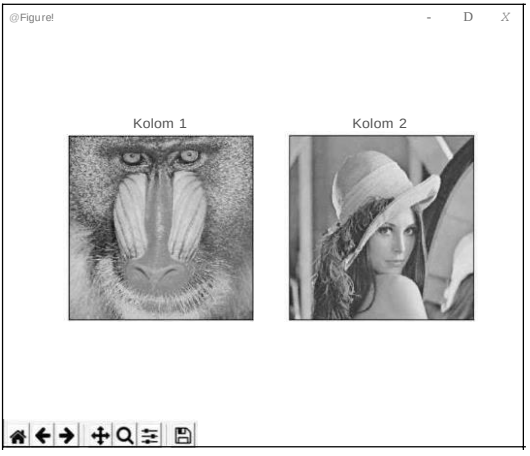
Perintah berikut digunakan untuk menghilangkan tanda garis dan angka pada sumbu X dan Y:

```
plt.xticks([]), plt.yticks([])
```

Perintah berikut digunakan untuk menampilkan judul di atas gambar:

```
plt.title('Kolom 1')
```

Gambar 5.12 memperlihatkan hasil skrip subplotl2.py.



Gambar 5.1!2 Dua citra ditampilkan beijajaran ke samping

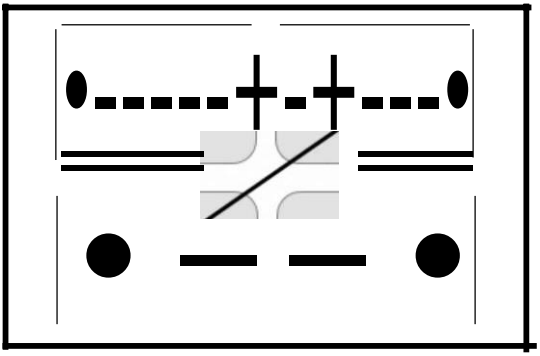
Adapun skrip berikut menunjukkan pembuatan grafik dengan empat m ng disusun dalam 2 baris dan 2 kolom:

Berkas : eku.ilwarn.i.11y

```
# Satu baris dan dua kolom citra berwarna
#    menggunakan subplot()

import cv2
from matplotlib import pyplot as plt

citra1   cv2.imread('baboon.png')
citra2   cv2.imread('lena.png')
citra3   cv2.imread('goldhill.png')
citra4   cv2.imread('boat.png')

# Ubah BGR menjadi RGB
rgb1   citra1[...,     -1]
rgb2   citra2[...,     -1]
rgb3   citra3[...,     -1]
rgb4   citra4[...,     -1]

# Tampilkan gambar
plt.subplot(221)
plt.imshow(rgb1)
plt.xticks([]), plt.yticks([])
plt.title('Baboon')

plt.subplot(222)
plt.imshow(rgb2)
plt.xticks([]), plt.yticks([])
plt.title('Lena')

plt.subplot(223)
plt.imshow(rgb3)
plt.xticks([]), plt.yticks([])
plt.title('Goldhill')

plt.subplot(224)
plt.imshow(rgb4)
plt.xticks([]), plt.yticks([])
plt.title('Boat')

   show()
```

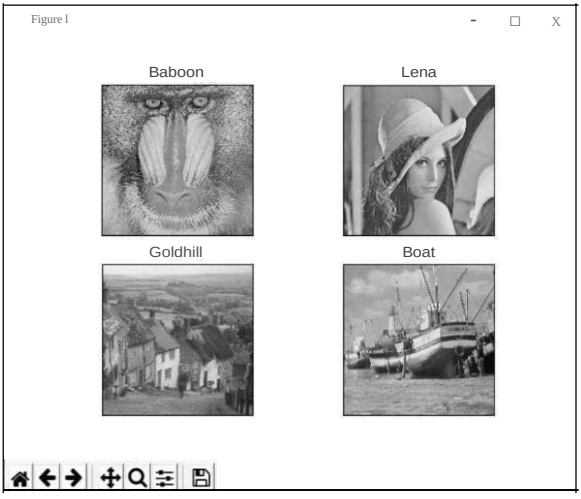
Akhir berkas

Berdasarkan contoh sebelum ini, perintah seperti pl t. subplot (221) mudah ditebak maknanya. Ya, artinya adalah posisi pertama pada area dengan 2 baris dan 2 kolom. Pada area seperti ini, penomorannya seperti terlihat pada Gambar 5.13.



Gambar .5.1s Penomoran gambar pada area g baris dan g kolom

Hasil skrip subplot22. pydiperlihatkan pada Gambar 5.14.



Gambar** .5.14 **Jendela dengang baris dang kolom

Catatan

Bentuk seperti

```
plt.subplot(234)
```

bisa ditulis menjadi

```
plt.subplot(2, 3, 4)
```

Hal seperti ini berguna jika penomoran gambar melebihi satu digit.

Nah, sekarang akan ditunjukkan contoh penerapan pada citra beraras keabu-abuan. Skrip berikut adalah hasil pengubahan terhadap skrip ekualisasi.py,**dengan bagian untuk menampilkan citra diganti**

m Berkas : ekualw:;irn.i.11y

```
# Ekualisasi histogram
                                    0
import cv2
import numpy as np
from matplotlib import pyplot as plt

citra   cv2.imread('gembala.png', 0)
ekual   cv2.equalizeHist(citra)

# Tampilkan gambar asal dan hasil
plt.subplot(121)
plt.imshow(citra, cmap = plt.get  cmap('gray'),
                  vmin = 0, vmax = 255)
plt.xticks([]), plt.yticks([])
plt.title('Citra semula')

plt.subplot(122)
plt.imshow(ekual, cmap = plt.get  cmap('gray'),
                  vmin = 0, vmax = 255)
plt.xticks([]), plt.yticks([])
plt.title('Citra hasil')

show()
```

Akhir berkas

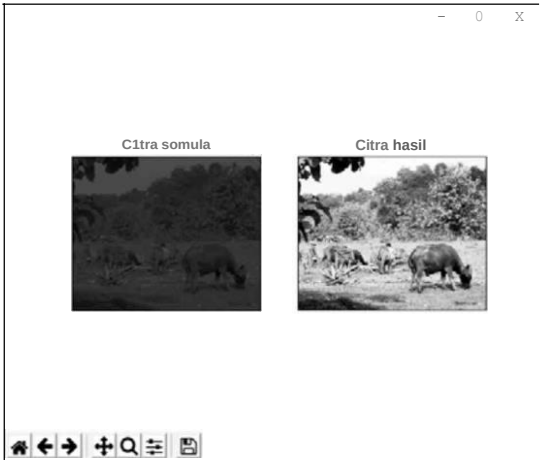
Pada skrip ini, hal paling penting yang perlu dibahas adalah hal yang berbeda dengan contoh sebelum ini, yakni pada pernyataan:

```
plt.imshow(citra, cmap = plt.get_cmap('gray'),
                  vmin = O, vmax = 255)
```

Perhatikan bahwa pada imshow () terdapat tambahan tiga argumen. Tambahan pertama berupa:

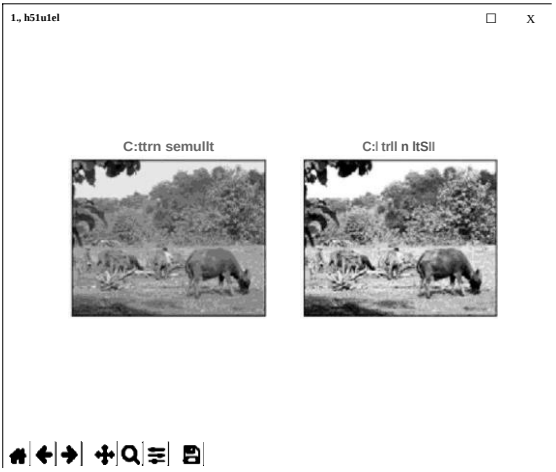
```
cmap = plt.get_cmap('gray')
```

Bagian ini menyatakan bahwa yang digunakan adalah warna beraras keabu-abuan. Argumen vmin = O dan vmax = 255 digunakan untuk memastikan bahwa jangkauan intensitas yang digunakan adalah dari O hingga 255. Hasilnya dapat dilihat pada Garnbar 5.15.



Gambar** 5.15 **Contoh penerapan pada ekualisasi citra

Apa yang terjadi sekiranya argumen vmin = O dan vmax 255 tidak disertakan? Pada keadaan seperti ini, perintah imshow () akan menggunakan pengaturan tersendiri sehingga intensitas yang ditampilkan dioptimalkan. Hasilnya, citra asal akan menjadi terang, seperti yang terlihat pada Gambar 5.16.



Gambar .5.16 Hasil kalau vmin dan vmax tidak disertakan