PRAKTIKUM VIII

SEGMENTASI CITRA (*IMAGE SEGMENTATION*)

Materi:

- Segementasi Citra
- *Thresholding*
- Deteksi Tepi (*edge detection*).

Tujuan Praktikum:

- Mahasiswa dapat mengimplementasikan teknik *thresholding* dan *edge detection* menggunakan OpenCV-Python.

A. PENYAJIAN

1. Segmentasi Citra

Segmentasi citra merupakan proses mempartisi citra menjadi beberapa daerah atau objek. Umumnya segmentasi bertujuan untuk memisahkan suatu objek citra dengan objek lainnya dan atau memisahkan objek citra dengan latar belajang citra. Segmentasi citra didasarkan pada dua hal yaitu diskontinuitas dan kemiripan intensitas piksel. Pendekatan diskontinuitas disebut juga pendekatan berbasis *edge* (*edge-based*), sementara pendekatan kemiripan intensitas, seperti *thresholding*, *clustering* (contoh: k-means), klasifikasi, pendekatan teori graf (contoh: GrabCut), *region growing*, dan *region splitting* & *merging*.

2. *Thresholding*

Thresholding merupakan teknik yang paling dasar untuk melakukan segmentasi citra. Teknik thresholding dapat digunakan untuk melakukan segmentasi citra berdasaran warna tertentu atau berdasarkan suatu nilai piksel tertentu.

3. Deteksi Tepi (*Edge Detection*)

Deteksi tepi adalah operasi yang dijalankan untuk mendeteksi garis tepi yang membatasi dua wilayah citra homogen yang memiliki tingkat kecerahan yang berbeda. Edge adalah bagian dari citra dimana intensitas kecerahan berubah secara drastis. Tujuan dari deteksi tepi pada umumnya yaitu untuk mengurangi jumlah data pada citra pada langkah *image processing* berikutnya. Deteksi tepi adalah pendekatan yang sering digunakan untuk segementasi citra digital dengan berdasarkan perubahan intensitas yang lokal. Ada beberapa pendekatan atau metode untuk deteksi tepi, yaitu *first order edge detection* dan *second order edge detection* [1].

a) First Order Derivative Edge Detection

Metode deteksi tepi yang termasuk kedalam first order derivative adalah Roberts operators, Prewitt operators, Sobel operators, dan First-order of gaussian (FDOG). Pada modul ini akan dicoba untuk operator Sobel, Prewitt, dan Roberts. Setiap operator memiliki row gradient dan column gradient.

i. Roberts Operator

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

0	0	0
0	–1	0
0	0	1

$G_x = Z_9 - Z_5$

0	0	0
0	0	–1
0	1	0

$G_x = Z_8 - Z_6$

ii. Prewitt Operator

-1	-1	-1
0	0	0
1	1	1

$$G_x = (Z_7 + Z_8 + Z_9) - (Z_1 + Z_2 + Z_3)$$

1	1	1
0	0	0
-1	-1	-1

$$G_y = (Z_7 + Z_8 + Z_9) - (Z_1 + Z_2 + Z_3)$$

OpenCV-Python tidak menyediakan fungsi khusus untuk operator **roberts** dan **prewitt**. Namun dapat menggunakan fungsi **filter2D** yang terdapat pada OpenCV-Python dengan mendefinisikan sendiri kernel-nya, sesuai dengan operator yang ingin digunakan. Fungsi **filter2D** telah dijelaskan pada Modul VI: *Image Enhancement* dengan *Mask Processing*:

```
dst = cv.filter2D(src, ddepth, kernel, anchor, delta, borderType)
```

iii. Sobel Operator

Operator Sobel adalah operasi smoothing Gausssian bersama diferensiasi, sehingga lebih tahan terhadap noise. Arah turunan dapat kita tentukan, baik secara vertikal maupun horizontal (masing-masing dengan argumen x order dan y order). Dokumen lebih detail terkait kernel yang digunakan dapat dilihat pada:

- https://docs.opencv.org/4.1.2/d4/d86/group__imgproc__filter.html#gacea54f142e81b6758cb6f375ce782c8d.

Berikut contoh dengan parameter sobel (xorder = 1, yorder = 0, ksize = 3) atau (xorder = 0, yorder = 1, ksize = 3).

-1	-2	-1
0	0	0
1	2	1

$$G_x = (Z_7 + 2Z_8 + Z_9) - (Z_1 + 2Z_2 + Z_3)$$

1	2	1
0	0	0
-1	-2	-1

$$G_y = (Z_7 + 2Z_8 + Z_9) - (Z_1 + 2Z_2 + Z_3)$$

Pada OpenCV-Python dapat menggunakan fungsi sebagai berikut [2]:

```
dst = cv.Sobel(src, ddepth, dx, dy, ksize)
```

keterangan

dst : citra *output* dengan ukuran yang sama dan jumlah *channel* yang sama dengan src.

src : citra input

dx : order dari derivative x.

dy : order dari derivative y.

ksize : ukuran kernel Sobel (1, 3, 5, atau 7).

b) *Second Order Derivative Edge Detection*

i. *Laplacian Derivatives*

Laplacian gambar dappat dihitung dengan,

$$\Delta src = \frac{\partial^2 src}{\partial x^2} + \frac{\partial^2 src}{\partial y^2}$$

di mana setiap turunan ditemukan menggunakan turunan Sobel. Jika ksize = 1, maka kernel berikut digunakan untuk memfilter:

$$kernel = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Pada OpenCV-Python dapat menggunakan fungsi sebagai berikut:

```
dst = cv.Laplacian(src, ddepth, ksize)
```

keterangan

dst : citra *output* dengan ukuran yang sama dan jumlah *channel* yang sama dengan src.

src : citra input

ksize : Ukuran aperture yang digunakan untuk menghitung filter derivatif kedua. Ukuran harus bernilai positif dan ganjil, untuk nilai default adalah 1.

ii. *Canny detection*

Materi lebih lanjut tentang Canny dapat dilihat pada file Canny Operator di LMS. Untuk melakukan canny, dilakukan 5 langkah berikut [2]:

1. *Smoothing*, melakukan *blurring* untuk meghilangkan *noise*.
   Karena deteksi tepi rentan terhadap noise pada gambar, langkah pertama adalah menghilangkan noise pada gambar dengan filter Gaussian 5x5.
2. Menemukan gradien

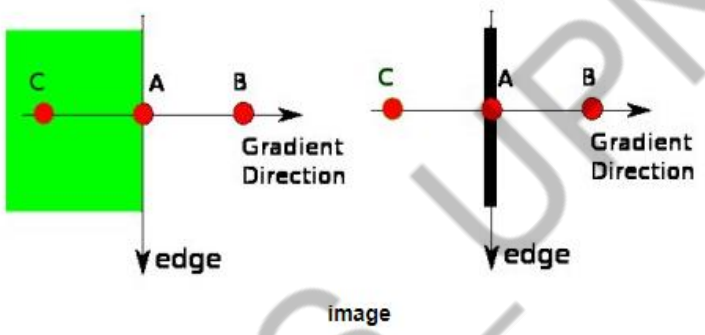
   Gambar yang dihaluskan kemudian dilakukan filter dengan kernel Sobel pada order horizontal (G_x) dan vertical (G_y) untuk mendapatkan turunan pertama. Dari dua gambar ini, kita dapat menemukan gradien tepi dan arah untuk setiap piksel sebagai berikut:

$$Edge\_Gradient\ (G) = \sqrt{G_x^2 + G_y^2}$$

$$Angle\ (\theta) = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$

3. *Non-maximum suppresion*

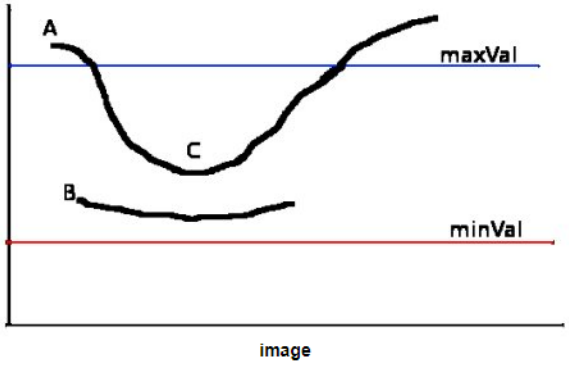
   Tujuan dari step ini adalah untuk meng-convert edge 'blur' dari step sebelumnya menjadi *edge* yang '*sharp*'. Algoritme ini dilakukan untuk setiap pixel pada citra gradient. Hanya maksimum lokal yang ditandai sebagai edge. Setiap piksel diperiksa jika itu adalah maksimum lokal di lingkungannya (neighborhood) dalam arah gradien (*gradient directions*). Lihat gambar di bawah ini:



   Titik A ada di edge (arah vertikal). Titik B dan C berada dalam *gradient directions*. Jadi titik A diperiksa dengan titik B dan C untuk melihat apakah itu membentuk maksimum lokal. Jika demikian, itu dipertimbangkan untuk tahap berikutnya, jika tidak, itu ditekan (nol). Singkatnya, hasil yang didapatkan pada tahapan ini adalah gambar biner dengan "tepi tipis".

4. *Hysteresis Thresholding*

   Tahap ini menentukan mana benar-benar sebagai tepi dan mana yang tidak. Untuk ini, kita membutuhkan dua nilai *threshold*, yaitu minVal dan maxVal. Setiap tepi dengan nilai intensitas gradien lebih dari maxVal adalah tepi dan yang di bawah minVal pasti non-edge. Intensitas yang berada di antara dua *threshold* ini dapat diklasifikasikan sebagai tepi atau non-tepi berdasarkan konektivitas mereka. Jika mereka terhubung ke piksel dapat diklasifikasikan sebagai tepi. Kalau tidak, mereka dapat dibuang. Lihat gambar di bawah ini:



Edge A berada di atas maxVal, sehingga dianggap sebagai “tepi”. Meskipun *edge* C berada di bawah maxVal, namun *edge* C terhubung ke edge A, sehingga dianggap sebagai *edge* yang valid dan didapatkan kurva penuh tersebut sebagai *edge*. Untuk *edge* B, meskipun berada di atas minVal dan berada di wilayah yang sama dengan tepi C, namun tidak terhubung ke “tepi”, sehingga *edge* tersebut dibuang. Maka sangat penting memilih minVal dan maxVal yang sesuai untuk mendapatkan hasil yang benar.

Tahap ini juga menghilangkan noise piksel kecil dengan asumsi tepi adalah garis panjang.

Pada OpenCV-Python dapat menggunakan fungsi sebagai berikut:

```
edges = cv.Canny(image, threshold1, threshold2)
```

keterangan

edges : output tepi (edge), channel tunggal 8-bit yang memiliki ukuran yang sama dengan citra input.

image : citra input 8-bit

threshold1 : *threshold* pertama untuk prosedur hysteresis.

threshold2 : *threshold* kedua untuk prosedur hysteresis.

B. Latihan

1. Baca gambar ‘monas.jpg’, lalu terapkan deteksi *edge* menggunakan operator sobel pada order horizontal (G_x), vertical (G_y), dan magnitude-nya, serta lakukan deteksi *edge* dengan menggunakan operator *Laplacian Derivatives*. Lalu tampilkan kelima gambar (gambar asli, gambar hasil filter sobel derivative x, gambar hasil filter sobel derivative y, gambar hasil magnitude, serta hasil filter Laplacian Derivatives) dengan menggunakan cv.imshow dan subplot matplotlib.

```
import numpy as np
import cv2 as cv
from matplotlib import pyplot as plt

img = cv.imread('monas.jpg',0)

laplacian64f = cv.Laplacian(img,cv.CV_64F)
abs_laplacian64f = np.absolute(laplacian64f)
laplacian_8u = np.uint8(abs_laplacian64f)

sobelx64f = cv.Sobel(img,cv.CV_64F,1,0,ksize=3)
abs_sobelx64f = np.absolute(sobelx64f)
sobelx_8u = np.uint8(abs_sobelx64f)

sobely64f = cv.Sobel(img,cv.CV_64F,0,1,ksize=3)
abs_sobely64f = np.absolute(sobely64f)
sobely_8u = np.uint8(abs_sobely64f)

magnitudesobel = cv.magnitude(sobelx64f,sobely64f)
abs_sobel64f = np.absolute(magnitudesobel)
sobel_8u = np.uint8(abs_sobel64f)

plt.subplot(3,2,1),plt.imshow(img,cmap = 'gray')
plt.title('Original'), plt.xticks([]), plt.yticks([])
plt.subplot(3,2,2),plt.imshow(laplacian_8u,cmap = 'gray')
plt.title('Laplacian'), plt.xticks([]), plt.yticks([])
plt.subplot(3,2,3),plt.imshow(sobelx_8u,cmap = 'gray')
plt.title('Sobel X'), plt.xticks([]), plt.yticks([])
plt.subplot(3,2,4),plt.imshow(sobely_8u,cmap = 'gray')
plt.title('Sobel Y'), plt.xticks([]), plt.yticks([])
plt.subplot(3,2,5),plt.imshow(sobel_8u,cmap = 'gray')
plt.title('Sobel Magnitude'), plt.xticks([]), plt.yticks([])
plt.show()

cv.imshow('Original', img)
cv.imshow('Laplacian', laplacian_8u)
cv.imshow('Sobel X', sobelx_8u)
cv.imshow('Sobel Y', sobely_8u)
cv.imshow('Sobel Magnitude', sobel_8u)
cv.waitKey(0)
cv.destroyAllWindows()
```

2. Baca gambar ‘monas.jpg’, lalu terapkan deteksi edge menggunakan *canny detection*.

```
import cv2 as cv
from matplotlib import pyplot as plt
img = cv.imread('monas.jpg',0)
edges = cv.Canny(img,100,200)
plt.subplot(121),plt.imshow(img,cmap = 'gray')
plt.title('Original Image'), plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(edges,cmap = 'gray')
plt.title('Edge Image'), plt.xticks([]), plt.yticks([])
plt.show()

cv.imshow('Original', img)
cv.imshow('Canny', edges)

cv.waitKey(0)
cv.destroyAllWindows()
```

C. Lembar Kerja Praktikum

1. Indonesia terkenal dengan keanekaragaman hayatinya, baik hewan dan tumbuhan. Banyak jenis tumbuhan khas Indonesia yang mungkin ada disekitar Kita. Tugas Anda adalah ambil foto disekitar terkait tumbuhan khas Indonesia. Foto yang diambil adalah bagian daun tumbuhan dengan latar belakang warna putih (daun dapat dipetik terlebih dahulu, lalu letakan di atas kertas putih). Daftar tumbuhan dapat dilihat pada halaman website ini (https://rimbakita.com/daftar-nama-tumbuhan-di-indonesia/).
   a) Lakukan deteksi tepi dengan menggunakan operator sobel pada *order horizontal* (G_x), *vertical* (G_y), dan *magnitude*-nya, lakukan juga deteksi *edge* dengan menggunakan operator *laplacian derivatives* dan *canny detection.*
   b) Lakukan segmentasi terhadap foto tersebut sedemikian sehingga latar foto bernilai 0 dan objek daun sesuai dengan intensitas objek tersebut. Pemrosesan citra menggunakan *grayscale*.

Referensi

[1] R. C. Gonzalez and R. E. Woods, “Digital Image Processing, 4th Edition,” *J. Electron. Imaging*, p. 1019, Jan. 2018.

[2] A. Mordvintsev and A. K, “OpenCV-Python Tutorials,” 2019. https://docs.opencv.org/4.1.2/d6/d00/tutorial_py_root.html.