SEGMENTASI CITRA

DISUSUN OLEH :

FARKHAN

NPM :

20081010060

MATA KULIAH :

PENGOLAHAN CITRA DIGITAL

PROGRAM STUDI INFORMATIKA

FAKULTAS ILMU KOMPUTER

UNIVERISTAS PEMBANGUNAN NASIONAL "VETERAN"

JAWA TIMUR

2022

Citra yang saya gunakan sebagai masukan dalam tugas segmentasi ini merupakan gambar daun papaya. Hal tersebut sesuai dengan paper yang kelompok saya gunakan. Citra tersebut dapat dilihat pada gambar di bawah ini.

Gambar 1. daun-pepaya.jpg

1. Deteksi tepi

   Deteksi tepi yang saya lakukan menggunakan fungsi yang sudah tersedia dari Bahasa pemrograman python.

   Pertama, lakukan import beberapa library yang dibutuhkan untuk melakukan deteksi tepi.

```
import cv2
import matplotlib.pyplot as plt
import numpy as np
from scipy import ndimage
```

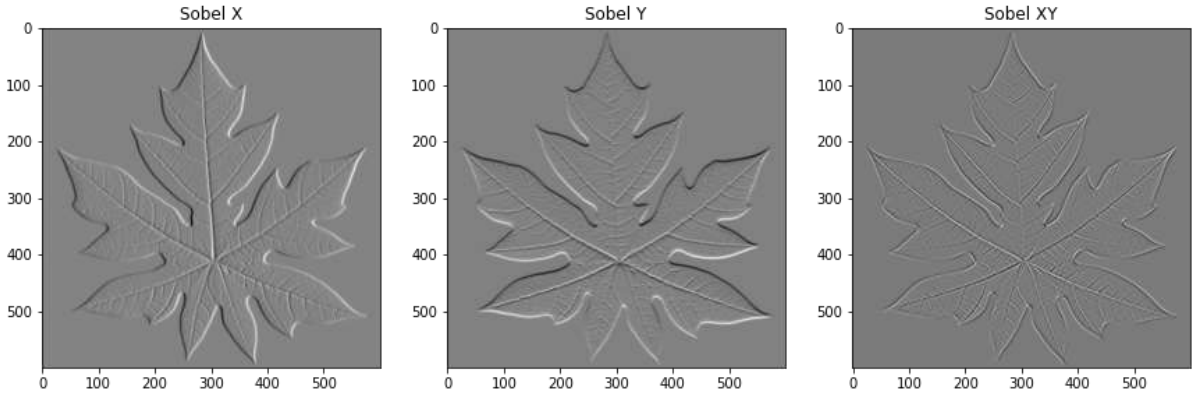
   Kemudian load citra yang digunakan dan dilanjut dengan mengubahnya menjadi grayscale dan menjadikannya blur.

```
img = cv2.imread('daun-pepaya.jpg')
# Convert to graycsale
img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
# Blur the image for better edge detection
img_blur = cv2.GaussianBlur(img_gray, (3,3), 0)
```

Kemudian lakukan deteksi tepi dengan metode sobel menggunakan fungsi yang sudah tersedia.

```
sobelx = cv2.Sobel(src=img_blur, ddepth=cv2.CV_64F, dx=1, dy=0, ksize=5)
sobely = cv2.Sobel(src=img_blur, ddepth=cv2.CV_64F, dx=0, dy=1, ksize=5)
sobelxy = cv2.Sobel(src=img_blur, ddepth=cv2.CV_64F, dx=1, dy=1, ksize=5)
```

Di bawah ini merupakan hasil gambar dari deteksi tepi sobel



Selanjutnya ialah menggunakan metode prewitt.

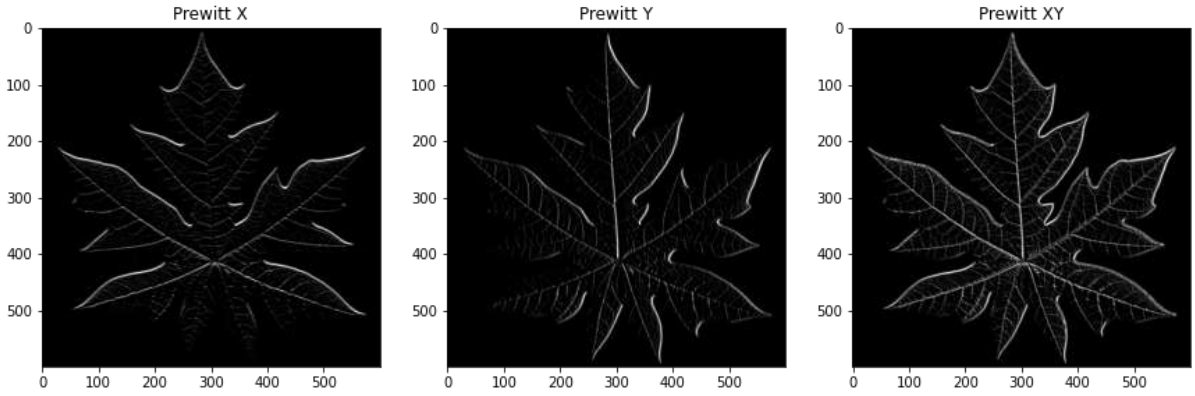
Pertama, buat dua kernel, yaitu kernel x dan kernel y

```
kernelx = np.array([[1,1,1],[0,0,0],[-1,-1,-1]])
kernely = np.array([[-1,0,1],[-1,0,1],[-1,0,1]])
```

Selanjutnya, di bawah ini syntax untuk melakukan deteksi tepi prewitt

```
img_prewittx = cv2.filter2D(img_blur, -1, kernelx)
img_prewitty = cv2.filter2D(img_blur, -1, kernely)
```

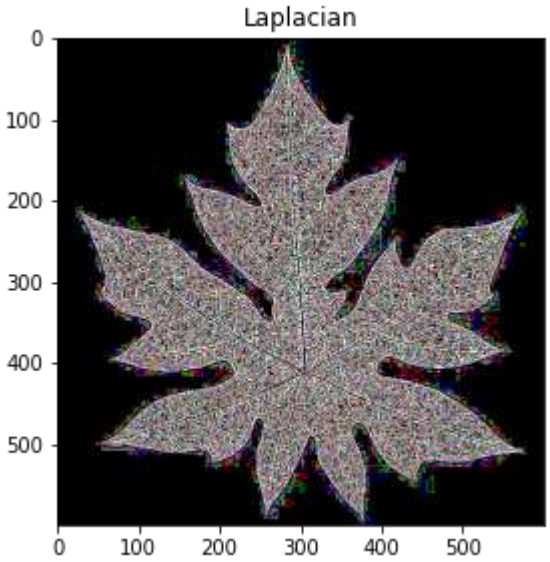
Di bawah ini hasil dari deteksi tepi prewitt



Untuk melakukan deteksi tepi laplacia, bisa menggunakan fungsi yang telah tersedia

```
laplacian = cv2.Laplacian(img,cv2.CV_64F)
```

Dengan hasilnya seperti di gambar di bawah ini.



Untuk melakukan deteksi tepi Roberts, pertama buat kernel untuk vertikal dan horizontal

```
roberts_cross_v = np.array( [[1, 0 ], [0,-1 ]] )
roberts_cross_h = np.array( [[ 0, 1 ], [ -1, 0 ]] )
```

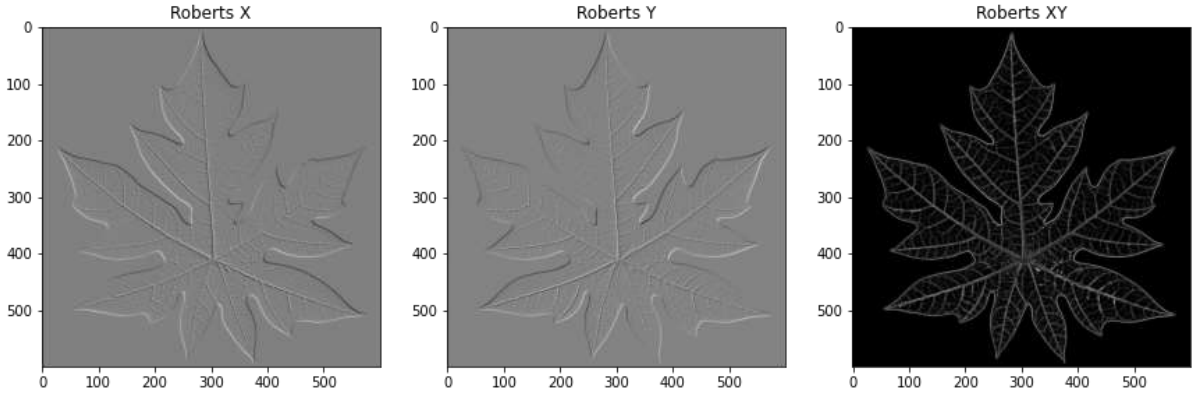
Selanjutnya citra yang digunakan diubah ke float dan dibagi dengan 255.0.

```
img = cv2.imread('daun-pepaya.jpg',0).astype('float64')
img /= 255.0
```

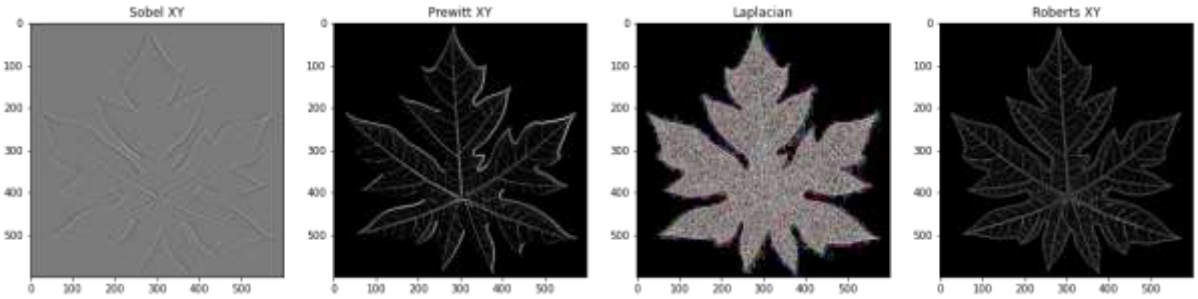
Syntax dibawah ini digunakan untuk melakukan deteksi tepi Roberts

```
vertical = ndimage.convolve( img, roberts_cross_v )
horizontal = ndimage.convolve( img, roberts_cross_h )
edged_img = np.sqrt( np.square(horizontal) + np.square(vertical))
edged_img*=255
```

Hasil dari deteksi tepi Roberts dapat dilihat pada gambar di bawah ini.



Apabila hasil dari keempat metode di atas disandingkan bersamaan, maka akan terlihat pada gambar di bawah ini.



Metode sobel memberikan hasil yang paling rendah di antara metode yang lain, hal ini dibuktikan dengan antara objek dan latar belakang yang tidak memiliki perbedaan warna serta tepi yang berhasil dideteksi sangat sedikit.

Deteksi tepi menggunakan prewitt memberikan hasil yang cukup baik. Namun, seluruh tepinya tidak terdeteksi secara merata, terdapat tepi yang memiliki ketebalan tinggi dan juga ada yang memiliki ketebalan yang rendah.

Laplacian memberikan hasil dengan perbedaan yang cukup jelas antara objek dengan latar belakang. Akan tetepi, terdapat banyak noise pada bagian dalam objek dan pada bagian tepi objek.

Metode roberts memberikan hasil yang paling baik di antara metode yang lain dalam percobaan ini. Tepi yang dideteksi didapatkan dengan sangat merata sehingga menunjukkan perbedaan yang jelas antara objek dengan latar belakang. Setiap tepi di dalam objek juga dideteksi dengan sangat baik dan tidak memberikan noise pada objek.

2. Otsu thresholding

Segmentasi citra merupakan proses pemisahan antara bagian *background* dan *foreground* dari suatu citra digital. Otsu thresholding merupakan satu di antara metode yang dapat digunakan untuk melakukan segmentasi citra. Ketika melakukan segmentasi citra, otsu thresholding menggunakan nilai ambang secara otomatis.

Pada percobaan otsu thresholding ini, saya masih menggunakan citra daun-pepaya.jpg. Pertama, saya import library yang akan digunakan.

```
import cv2
import matplotlib.pyplot as plt
```

Kemudian memanggil fungsi threshold dari python untuk melakukan otsu thresholding.

```
ret, thresh1 = cv2.threshold(image, 0, 255, cv2.THRESH_BINARY +
cv2.THRESH_OTSU)
```

thresh1 merupakan variabel yang menampung citra hasil dari otsu thresholding. Citra yang dihasilkan dari otsu thresholding ini dapat dilihat pada gambar di bawah ini.