Agung Prayoga
Muh Isfhani Ghiath

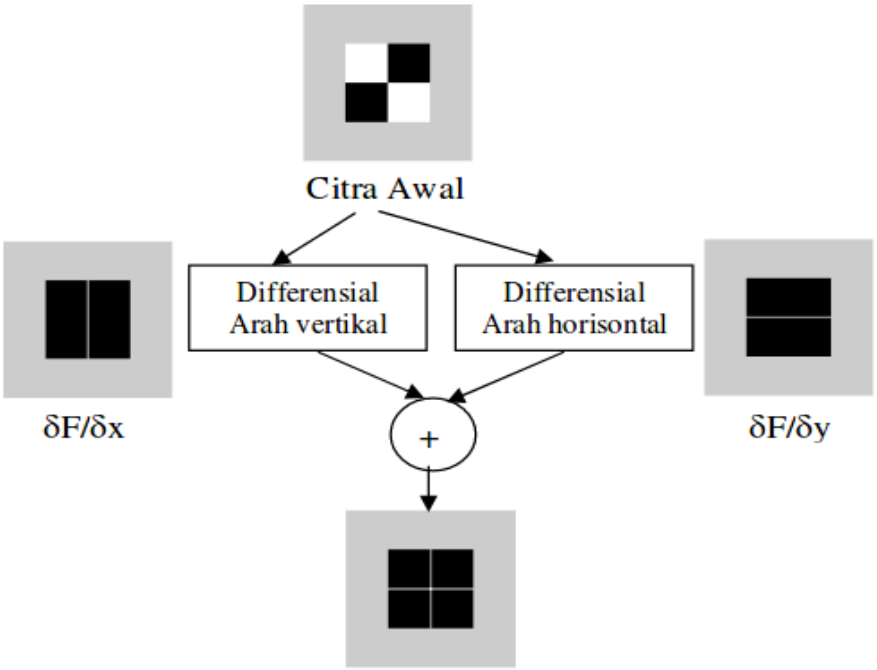
EDGE DETECTION

Edge detection adalah salah satu proses ekstraksi fitur yang mengidentifikasikan tepian citra, yaitu posisi dimana terjadi perubahan intensitas piksel secara tajam. Tepian dari suatu citra mengandung informasi penting dan mampu merepresentasikan objek-objek yang terkandung dalam citra tersebut meliputi bentuk, ukuran serta tekstur (Putra 2010).

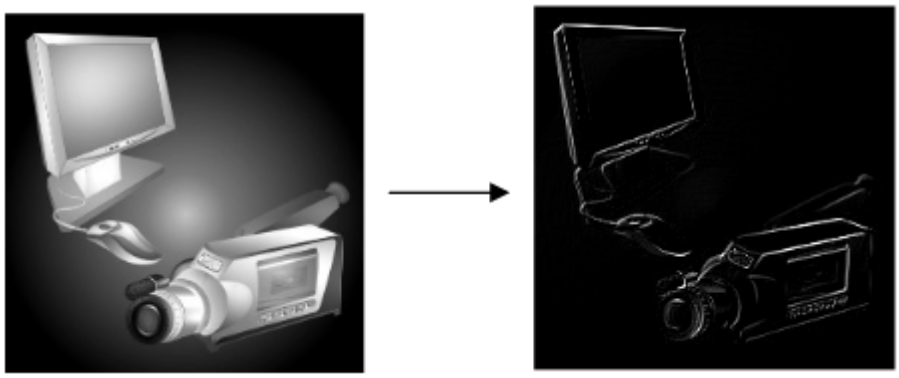
Tujuan dari edge detection ini adalah untuk:

1. Menandai bagian yang menjadi terperinci dari citra
2. Memperbaiki citra yang kabut, yang terjadi karena kesalahan atau adanya efek dari proses akuisisi citra

Suatu titik (x,y) dikatakan sebagai tepi (edge) dari suatu citra bila titik tersebut mempunyai perbedaan yang tinggi dengan tetangganya. Contoh gambar tepi dapat dilihat dibawah ini.



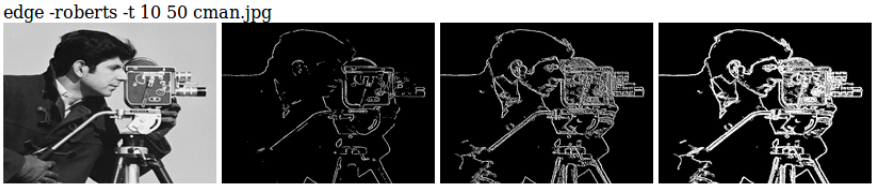
Agung Prayoga
Muh Isfhani Ghiath



Pada gambar diatas terlihat bahwa hasil deteksi tepi berupa tepi-tepi dari suatu gambar.

Terdapat macam-macam metode untuk proses deteksi tepi citra, diantaranya:

1. Metode Robert
   Metode Robert adalah nama lain dari teknik differensial yang dikembangkan di atas, yaitu differensial pada arah horisontal dan differensial pada arah vertikal, dengan ditambahkan proses konversi biner setelah dilakukan differensial.
   Teknik konversi biner yang disarankan adalah konversi biner dengan meratakan distribusi warna hitam dan putih. Metode Robert ini juga disamakan dengan teknik DPCM (Differential Pulse Code Modulation)



2. Metode Prewitt
   Metode Prewitt merupakan pengembangan metode robert dengan menggunakanfilter HPF yang diberi satu angka nol

penyangga. Metode ini mengambil prinsipdari fungsi laplacian yang dikenal sebagai fungsi untuk membangkitkan HPF.

edge -prewitts -t 10 50 cman.jpg

3. Metode Sobel

   Metode Sobel merupakan pengembangan metode robert dengan menggunakan filter HPF yang diberi satu angka nol penyangga. Metode ini mengambil prinsip dari fungsi laplacian dan gaussian yang dikenal sebagai fungsi untuk membangkitkan HPF. Kelebihan dari metode sobel ini adalah kemampuan untuk mengurangi noise sebelum melakukan perhitungan deteksi tepi.

edge -sobel -t 10 50 cman.jpg

4. Metode Canny

   Canny edge detector dikembangkan oleh John F. Canny pada tahun 1986 dan menggunakan algoritma multi-tahap untuk mendeteksi berbagai tepi dalam gambar. Walaupun metode tersebut telah berumur cukup lama, namun metode tersebut telah menjadi metode deteksi tepi standar dan masih dipakai dalam penelitan.

   Adapun kategori algoritma yang dikembangkan oleh John F. Canny adalah sebagai berikut:

   1. Deteksi: Kemungkinan mendeteksi titik tepi yang benar harus dimaksimalkan sementara kemungkinan salah mendeteksi titik tepi harus diminimalkan. Hal ini diamksudkan untuk memaksimalkan rasio signal-to-noise

Agung Prayoga
Muh Isfhani Ghiath

2. Lokalisasi: Tepi terdeteksi harus sedekat mungkin dengan tepi yang nyata.
3. Jumlah Tanggapan: Satu tepi nyata tidak harus menghasilkan lebih dari satu ujung yang terdeteksi.

edge -canny -t 10 50 cman.jpg

IMPLEMENTASI

Dalam implementasi ini, kami menggunakan OpenCV dalam ujicoba terhadap kelima metode tersebut. Lalu, untuk bahan foto ujicoba nya, sebagai berikut.

Agung Prayoga
Muh Isfhani Ghiath

Potongan kode dalam ujicoba nya, sebagai berikut.

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

filename = 'melonLap1.jpg'

# loading image and convert to gray
img0 = cv2.imread(filename, 0)
# converting to gray scale
gray = cv2.cvtColor(img0, cv2.COLOR_BGR2GRAY)
# remove noise
img = cv2.GaussianBlur(gray,(3,3),0)
# convolute with proper kernels
laplacian = cv2.Laplacian(img,cv2.CV_64F)
sobel = cv2.Sobel(img,cv2.CV_64F,1,0,ksize=5)
canny = cv2.Canny(img,cv2.CV_64F,100,200)

#prewitt
kernelx = np.array([[1,1,1],[0,0,0],[-1,-1,-1]])
kernely = np.array([[-1,0,1],[-1,0,1],[-1,0,1]])
img_prewittx = cv2.filter2D(img_gaussian, -1, kernelx)
img_prewitty = cv2.filter2D(img_gaussian, -1, kernely)

# robert
roberts_cross(filename, 'res_' + filename)

plt.subplot(2,2,1),plt.imshow(img,cmap = 'gray')
plt.title('Original Image'), plt.xticks([]), plt.yticks([])
plt.subplot(2,2,2),plt.imshow(laplacian,cmap = 'gray')
plt.title('Laplacian'), plt.xticks([]), plt.yticks([])
plt.subplot(2,2,3),plt.imshow(sobel,cmap = 'gray')
plt.title('Sobel'), plt.xticks([]), plt.yticks([])
plt.subplot(2,2,4),plt.imshow(canny,cmap = 'gray')
plt.title('Canny'), plt.xticks([]), plt.yticks([])

cv2.imshow("Prewitt X", img_prewittx)
cv2.imshow("Prewitt Y", img_prewitty)
cv2.imshow("Prewitt", img_prewittx + img_prewitty)

plt.show()
```

Lalu, edge detection untuk metode robertt dibuat secara terpisah dengan potongan kode sebagai berikut:

```python
import sys
import numpy as np
from scipy import ndimage
import Image

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

def load_image(infilename):
    img = Image.open(infilename)
    img.load()
    # note signed integer
    return np.asarray(img, dtype="int32")

def save_image(data, outfilename):
    img = Image.fromarray( np.asarray( np.clip(data,0,255), dtype="uint8"), "L")
    img.save(outfilename)

def roberts_cross(infilename, outfilename):
    image = load_image(infilename)
    vertical = ndimage.convolve(image, roberts_cross_v)
    horizontal = ndimage.convolve(image, roberts_cross_h)
    output_image = np.sqrt( np.square(horizontal) + np.square(vertical))
    save_image(output_image, outfilename)
```

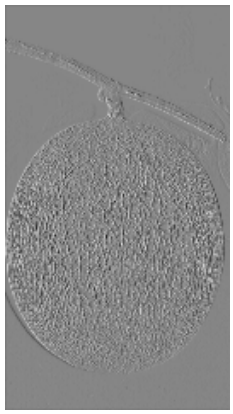
Agung Prayoga
Muh Isfhani Ghiath

HASIL UJICOBA

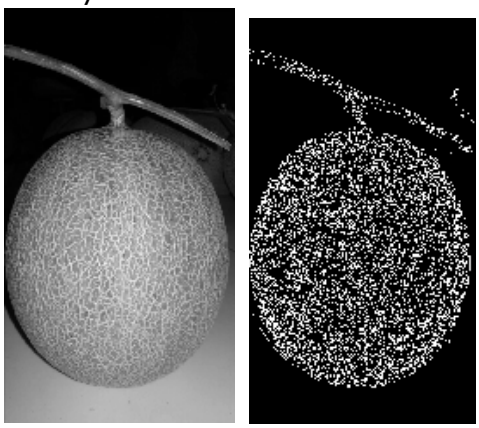
1. Robert

2. Prewitt

3. Sobel



4. Canny



5. Laplassian

DAFTAR PUSTAKA

1. Putra, Darma, Pengolahan Citra Digital, ANDI Jogjakarta, 2010
2. http://riyanto.lecturer.pens.ac.id/citra-bab8.pdf