APLIKASI DETEKSI TEPI VIDEO REALTIME DAN GAMBAR

ALGORITMA CANNY EDGE DETECTION

Mario aldi putra
Universitas Buana Perjuangan
Karawang, Indonesia
if19.marioputra@mhs.ubpkarawang.ac.id
(19416255201199)

Ayu Sri Rahayu
Universitas Buana Perjuangan
Karawang, Indonesia
if19.ayurahayu@mhs.ubpkarawang.ac.id
(19416255201197)

Indi Nurul Hassanah
Universitas Buana Perjuangan
Karawang, Indonesia
if19.indihassanah@mhs.ubpkarawang.ac.
(19416255201182)

Pemrosesan video dan gambar digunakan dalam berbagai macam aplikasi mulai dari pengawasan video dan manajemen lalu lintas hingga aplikasi pencitraan medis. Makalah ini menyajikan implementasi deteksi tepi cerdik menggunakan video realtime dari kamera. Algoritme Canny menggunakan detektor tepi yang optimal berdasarkan serangkaian kriteria yang mencakup menemukan tepi terbanyak dengan meminimalkan tingkat kesalahan, menandai tepi sedekat mungkin dengan tepi sebenarnya untuk memaksimalkan pengolahan, dan menandai tepi hanya sekali jika ada satu tepi. untuk respon minimal, dengan metode seperti Sobel,Prewitt Operator dan Laplacian Operator

Kata kunci — Deteksi tepi scanner, OpenCV, canny edge detection.

I.PENDAHULUAN

Deteksi tepi merupakan teknik analisis gambar yang penting ketika seseorang tertarik untuk mengenali objek dengan garis besarnya,. Algoritma Canny merupakan salah satu algoritma deteksi Tepi yang memiliki tingkat kesalahan yang minimum dan menghasilkan citra tepian yang optimal. Hasil algoritma Canny mendeteksi Tepi untuk menentukan wilayah video lalu diidentifkiasi menggunakan pengenalan karakter optis (OCR) akan dijadikan perhitungan untuk menilai kinerja algoritma deteksi Tepi Canny. Pemrosesan video dan gambar waktu nyata digunakan dalam berbagai macam aplikasi mulai dari pengawasan video , kontur wilayah , manajemen lalu lintas, dan tulisan pada Citra Digital. Hingga aplikasi pencitraan medis. Makalah ini menyajikan implementasi deteksi tepi cerdik menggunakan video realtime dari kamera. Algoritme Canny menggunakan detektor tepi yang optimal berdasarkan serangkaian kriteria yang mencakup menemukan tepi terbanyak dengan meminimalkan tingkat kesalahan, menandai tepi sedekat mungkin dengan tepi yang sebenarnya untuk memaksimalkan pelokalan, dan menandai tepi hanya sekali jika ada satu tepi. Canny Detektor tepi biasanya mengambil gambar skala abu-abu sebagai input dan menghasilkan gambar yang menunjukkan lokasi diskontinuitas intensitas sebagai output (yaitu tepi) . Kelebihan dari algortima Canny yaitu mendeteksi karakter dengan baik, melokalisasi karakter dengan baik, dan respon yang jelas dengan satu respon untuk setiap Tepi. Dalam beberapa penelitian sebelumnya algoritma Canny sudah digunakan dalam Deteksi Kanker, Deteksi Plat Nomor dan Automatically Detect and Recognize Text in Natural Image.

Metode

Dalam Pengolahan Citra Digital Deteksi Tepi dibutuhkan script atau code program yang digunakan untuk melengkapi dan menyempurnakan code dari program deteksi tepi gambar ini. Ada pun script yang digunakan dalam pembuatan program ini.

Hardware

Perangkat yang digunkan untuk membangun aplikasi dan penelitian ini menggunkanan computer 64 bit dengan spesifikasiIntel Core gen 5. 8 GB RAM,dan SSD 25 GB. Sebuah kamera / webcamyang digunakan untuk pengambilan gambar video.

Perangkat Lunak

Perangkat lunak yang digunakan dalam membangun aplikasi Deteksi tepi pada realtime video menggunakan algoritma Canny Detection adalah sebagai berikut:

1. Python Idle 3.8, Sebuah compiler yang dibuat menggunakan bahasa pemrograman python yang akan digunakan untuk aplikasi sistem ini.
2. OpenCV , Sebuah library untuk fungsi-fungsi computer vision dan opencv yang mendukung pembuatan program ini

Perancangan Aplikasi

merupakan data gambar atau videoyang diambil secara realtime menggunakan webcam yang dipasang pada obyek. Oleh Video Capture, data yang berupa gambar atau video realtime kemudian di ambil datanya untuk di olah atau d ubah dalam bentuk binery untuk menghasilkan gmabr atau video yang selanjutnya akan di proses dengan algotitma Canny Edge Detection untuk mendapatkan hasil berupa gambar video realtime yang sudah mengalami proses deteksi tepi.

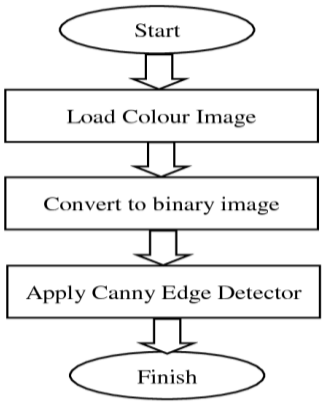

Figure 1 FLOW CHART CANYY DETECTIONS

Proses Aplikasi

Langkah selanjutnya ini melibatkan 3 sub pemindaian gambar. Berikut 3 sub pemindaian yang dilakukan secara bertahap:

- Deteksi Tepi *(Canny Edge Detection)*

Untuk mendeteksi tepi citra pada video webcam digunakan algoritma *Canny Edge Detection.*Algoritma detektor tepi populer yang dikembangkan pada tahun 1986 oleh John F.Canny. Algoritma feteksi tepi *Canny* terdiri beberapa langkah:

1) Pengurangan Noise

Karena deteksi tepi rentan terhadap noise pada gambar, langkah pertama adalah menghilangkan noise pada gambar dengan filter sesuai yang dibutuhkan pengguna. Pengurangan noise menggunakan fungsi Gaussian Blur yang terdapat pada perpustakaan OpenCV.

2) Perhitungan Gradient

Perhitungan gradient dilakukan dengan mengkonversi citra menjadi grayscale, agar dapat menentukan titik tepi pada citra yang ditangkap dengan nilai intensitas yang paling besar. Gradien gambar yang telah diperhalus dengan memperkirakan gradien arah *x* dan *y.*

3) Suppression Non-maximum

Setelah dapatkan arah gradien dan magnitudo, pemindaian penuh dilakukan untuk menghilangkan piksel yang tidak diinginkan yang mungkin bukan merupakan bagian tepi.

4) Double Thresholding

Langkah ini memutuskan semua tepi benar-benar tepi dan mana yang bukan. Untuk ini kita butuh dua nilai ambang, *minimum* dan *maks*. Setiap sisi dengan intensitas gradien lebih dari *maxVal* pasti menjadi tepi dan yang dibawah *minimum* tidak menjadi bagian, sehingga dibuang. Jadi sangant penting kita harus menyesuaikan *minVal* dan *maxVal* untuk mendapatkahn hasil yang benar. Tahap ini juga menghilangkan noise piksel kecil dengan asumsi bahwa tepi dan adalah garis panjang.

5) Menemukan Kontur atau *(*Edge Tracking by Hysteresis

hasil ambang batas, histeresis terdiri dari mengubah piksel lemah menjadi piksel kuat, jika dan hanya jika setidaknya satu piksel di sekitar piksel yang sedang diproses adalah piksel kuat

Setelah menerapkan langkah-langkah ini, Anda akan mendapatkan yang berikut ini:

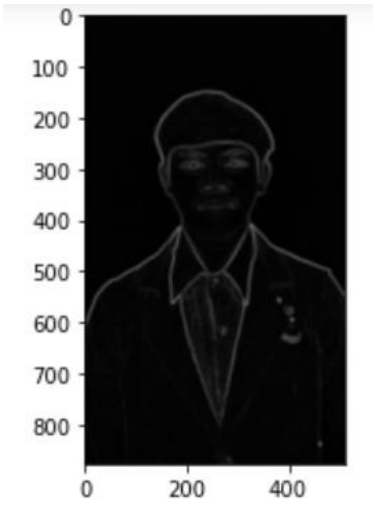

Figure 2 HASIL PENGOLAHAN CITRA CANNY

II.HASIL DAN PEMBAHASAN

Pembahasan

Algoritma Canny Edge Detection yang digunakan pada proses pengolahan citra gambar atau video secara realtime ini memiliki prinsip yang hampir sama seperti pada proses pendeteksian obyek citra gambar diam atau foto digital. Kelebihan aplikasi ini adalah dapat menerjemahkan atau mentranslasikan citra gambar dan video secara realtime menjadi tampilan gambar video yang sudah mendapatkan proses pendeteksian tepi menggunakan Canny Edge Detection, sehingga output dapat berjalan sesuai dengan kondisi realtime obyek yang diambil, meskipun dalam kondisi bergerak pada citra gambar maupun video. Namun dalam proses pengolahan data dari gambar atau video realtime menjadi tampilan gambar deteksi tepi yang berjalan secara realtime memerlukan kemampuan memori yang sangat besar, karena proses pengolahan translasi data yang *contiue* serta berjalan terus menerus mengakibatkan kebutuhan memory yang digunakan dalam computer cukup besar juga dan akan sangat berpengaruh pada proses pendeteksian frame demi frame dari sinyal video yang digunakan dalam penedeteksi garis tepi. Terlebih lagi menggunakan berbagai operator sobel, prewit dan Operator Laplacian. Yang masing masing cara penerapan nya berbeda.

Berdasarkan pada fungsiyang telah dibahas , maka dihasilkan citra yang diolah dengan 3 operator yang berbeda yaitu sebagai berikut:

Pertama Impor perpustakaan yang diperlukan

```
[9]: #import required library
     import numpy as np
     import cv2
     import matplotlib.pyplot as plt
     import matplotlib.image as mpig
     %matplotlib inline
```

Figure 3 libary

Membaca gambar dalam skala abu-abu

```
Di [10]: #read the gray scale image
         image = cv2.imread('output/mario.jpg',cv2.IMREAD_GRAYSCALE)
         plt.imshow(image,cmap='gray')

ar [10]: <matplotlib.image.AxesImage di 0x2304a39f2e0>
```

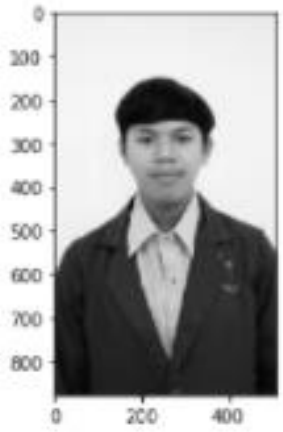

Figure 4 MEMBACA GAMBAR

1. Menggunakan Operator Sobel

Sobel adalah operator yang sangat umum untuk mendeteksi tepi gambar, yang merupakan pendekatan turunan gambar, yang terpisah dalam arah y dan x. Di sini Kami menggunakan matriks kernel 3 * 3, satu untuk setiap arah x dan y. Gradien untuk arah-x memiliki angka minus di kiri dan angka positif di kanan dan kami mempertahankan piksel tengah. Demikian pula, gradien untuk arah-y memiliki angka minus di bagian bawah dan angka positif di atas dan di sini kita tengah piksel baris.

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

Gx

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

Gy

Figure 5 Menentukan kernel Sobel x dan y

```
#menentukan kernel sobel horizontal dan Vertikal
Gx = np.array([[-1, 0, 1], [-2, 0, 2], [-1, 0, 1]])
Gy = np.array([[-1, -2, -1], [0, 0, 0], [1, 2, 1]])
```

Tentukan Fungsi Konvolusi Kernel

Convolution berfungsi untuk mengalikan dua larik angka, umumnya berukuran berbeda, di **sini gambar skala abu-abu dan kernel sobel / prewitt** , untuk menghasilkan deretan angka ketiga dengan dimensi yang sama. Konvolusi dilakukan dengan menggeser kernel yang diterapkan ke semua piksel gambar.
A adalah gambar sumber dan * menunjukkan operasi konvolusi.

$$G_x = \begin{vmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{vmatrix} * A$$

$$G_y = \begin{vmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{vmatrix} * A$$

```
#define kernal convolution function
# with image X and filter F
def convolve(X, F):
    # height and width of the image
    X_height = X.shape[0]
    X_width = X.shape[1]

    # height and width of the filter
    F_height = F.shape[0]
    F_width = F.shape[1]

    H = (F_height - 1) // 2
    W = (F_width - 1) // 2

    #output numpy matrix with height and width
    out = np.zeros((X_height, X_width))
    #iterate over all the pixel of image X
    for i in np.arange(H, X_height-H):
        for j in np.arange(W, X_width-W):
            sum = 0
            #iterate over the filter
            for k in np.arange(-H, H+1):
                for l in np.arange(-W, W+1):
                    #get the corresponding value from image and filter
                    a = X[i+k, j+l]
                    w = F[H+k, W+l]
                    sum += (w * a)
            out[i,j] = sum
    #return convolution
    return out
```

```
#normalisasi vektor
sob_x  =  convolve ( gambar , Gx ) /  8.0
sob_y  =  membelit ( gambar , Gy ) /  8.0
```

Besaran Gradien.

Komponen gradien di setiap orientasi kemudian digabungkan bersama untuk menemukan besaran absolut dari gradien pada setiap titik dan orientasi gradien tersebut.

```
#hitung besaran gradien vektor
sob_out  =  np . sqrt ( np . power ( sob_x , 2 ) +  np . power ( sob_y , 2 ))
# nilai pemetaan dari 0 hingga 255
sob_out  = ( sob_out  /  np . max ( sob_out )) *  255
```

```
#output gambar
cv2 . imwrite ( 'output / sobel_jet.jpg' , sob_out )
plt . imshow ( sob_out , cmap  =  'grey' , interpolation  =  'bicubic' )
plt . tampilkan ()
```

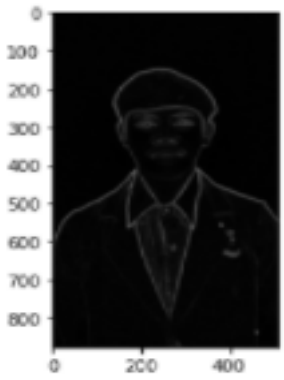

Figure 6 HASIL SOBEL

2. Menggunakan Operator Prewitt

Operator prewitt mirip dengan operator Sobel dan digunakan untuk mendeteksi tepi vertikal dan horizontal pada gambar. Operator ini memberi kita dua masker, satu untuk mendeteksi tepi dalam arah horizontal dan satu lagi untuk mendeteksi tepi dalam arah vertikal.

$$G_x = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} \quad G_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

Prewitt Mask

GAMBAR 4

```
Di [36]: #read the image in gray scale
         img = cv2.imread('output/mario.jpg',cv2.IMREAD_GRAYSCALE)
         plt.imshow(img,cmap='gray')

Keluar [36]: <matplotlib.image.AxesImage di 0x2304ab42340>
```

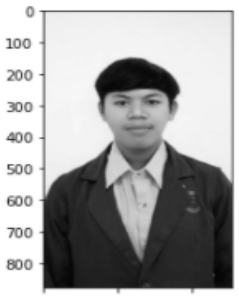

Figure 7 MEMBACA GAMBAR

```
Di [37]: #menentukan kernel sobel horizontal dan Vertikal
         Hx  =  np . larik ([[ - 1 , 0 , 1 ], [ - 1 , 0 , 1 ], [ - 1 , 0 , 1 ]])
         Hy  =  np . larik ([[ - 1 , - 1 , - 1 ], [ 0 , 0 , 0 ], [ 1 , 1 , 1 ]])
```

```
Di [38]: #normalisasi vektor
         pre_x  =  convolve ( img , Hx ) /  6.0
         pre_y  =  convolve ( img , Hy ) /  6.0
```

```
Di [39]: #calculate the gradient magnitude of vectors
         pre_out = np.sqrt(np.power(pre_x, 2) + np.power(pre_y, 2))
         # mapping values from 0 to 255
         pre_out = (pre_out / np.max(pre_out)) * 255
```

```
Di [40]: #output images
         cv2.imwrite('output/prewitt_knife.jpg', pre_out)
         plt.imshow(pre_out, cmap = 'gray', interpolation = 'bicubic')
         plt.show()
```

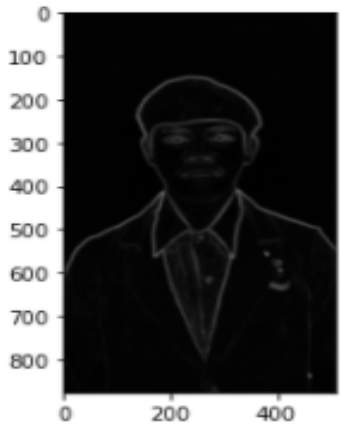

Figure 8 hasil prewitt

3. Menggunakan Operator Laplacian

Tidak seperti detektor tepi Sobel dan prewitt, detektor tepi Laplacian hanya menggunakan satu kernel. Ini menghitung turunan urutan kedua dalam satu lintasan. Inilah kernel yang digunakan untuk itu:

0	−1	0
−1	4	−1
0	−1	0

−1	−1	−1
−1	8	−1
−1	−1	−1

```
#read the image in gray scale
img2 = cv2.imread('output/mario.jpg',cv2.IMREAD_GRAYSCALE)
plt.imshow(img2,cmap='gray')
```

```
<matplotlib.image.AxesImage di 0x2304a6b4f70>
```

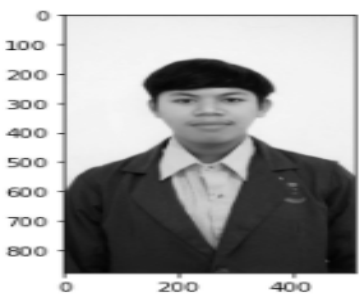

Figure 9 membaca gambar

Gaussian Blur

Karena topeng ini mendekati pengukuran turunan kedua pada gambar, maka topeng ini sangat sensitif terhadap noise. Untuk memperbaikinya, gambar sering kali dihaluskan dengan Gaussian sebelum menerapkan filter Laplacian.

```
: #apply gaussian blur
  blur_img = cv2.GaussianBlur(img2, (3, 3), 0)

: # Positive Laplacian Operator
  laplacian = cv2.Laplacian(blur_img, cv2.CV_64F)

: cv2.imwrite('output/lap_butterfly.jpg', laplacian)
  plt.imshow(laplacian, cmap = 'gray',interpolation = 'bicubic')
  plt.show()
```

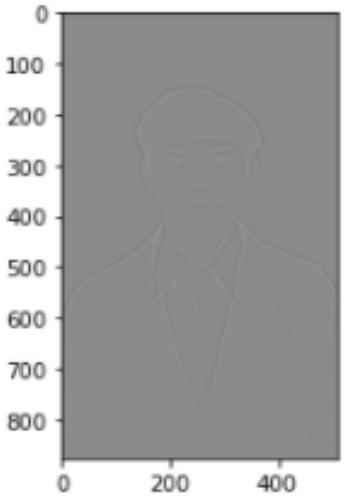

Figure 10 hasil Laplacian

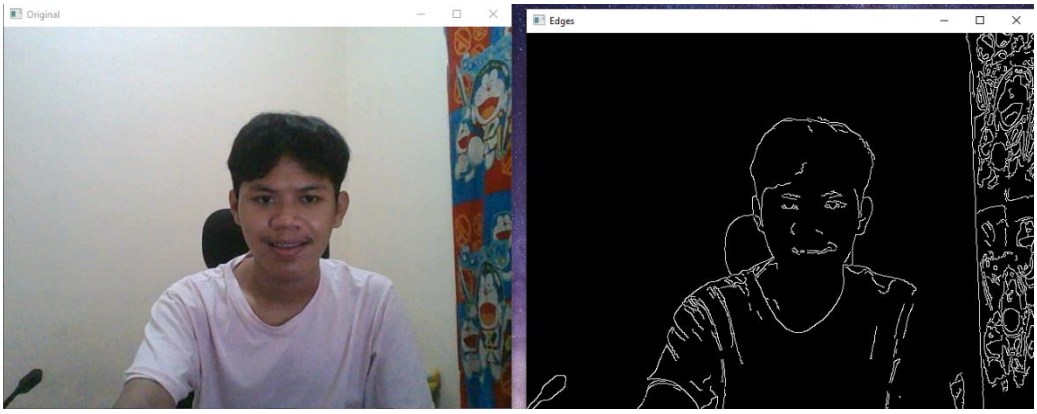

Figure 11 hasil video Realtime

III.KESIMPULAN

Dari hasil implementasi dan panelitian pendeteksi tepi pada realtime video menggunakan algoritma Canny Detection dan juga pengujian dan analisa dari penggunaan operator dapat disimpulkan bahwa:

1. Algoritma Canny Edge Detection merupakan salah satu dari beberapa algoritma deteksi tepi yang memiliki kelebihan yaitu memberikan hasil deteksi tepi yang optimal dan mampu memberikan hasil sesuai dengan pemilihan parameter-parameter

pixel konvolusi yang dilakukan. Sekaligus juga memberikan fleksibilitas yang sangat tinggi dalam hal menentukan tingkat deteksi ketebalan tepi sesuai yang diinginkan.

2. Canny Edge Detection dapat diterapkan dalam deteksi sebuah gambar / foto digital maupun gambar video realtime yang diambil dari webcam / kamera.

3. Operator berbasis gradient menghasilkan deteksi tepi yang lebih baik daripada operator berbasis turunan kedua (Laplacian)
4. Operator Prewitt dan menghasilkan deteksi tepi yang paling jelas diantara operator berbasis gradient lainnya

IV.DAFTAR PUSTAKA

1. BOVIK A, C., THE ESSENTIAL GUIDE TO VIDEO PROCESSING, ACADEMIC PRESS USA, 2009.
2. CASTLEMAN, KENNETH R., 2004, DIGITAL IMAGE PROCESSING, VOL. 1, ED.2, PRENTICE HALL, NEW JERSEY.
3. GONZALEZ. R. C, WOODS. R. E., DIGITAL IMAGE PROCESSING THIRD EDITION, PEARSON PRENTICE HALL, NEW JERSEY, 2008.
4. JAIN, A.K., FUNDAMENTAL OF DIGITAL IMAGE PROCESSING, PRENTICE HALL, INC., SINGAPORE,1989.
5. LOW, A., INTRODUCTORY COMPUTER VISION AND IMAGE PROCESSING, MCGRAW-HILL , UK, 1991.
6. MUNIR, R., PENGOLAHAN CITRA DIGITAL DENGAN PENDEKATAN ALGORITMIK , INFORMATIKABANDUNG , 2004..
7. MURNI, A. DAN S. SETIAWAN, PENGANTAR PENGOLAHAN CITRA,ELEX MEDIAKOMPUTINDO, JAKARTA,1992.
8. NIXON, M, S., AGUADO, A, S., FEATURE EXTRACTION AND IMAGE PROCESSING SECOND EDITION., ACADEMIC PRESS, 2008.
9. PITAS, I., DIGITAL IMAGE PROCESSING ALGORITHMS, PRENTICE HALL, SINGAPORE, 1993.
10. ADIPRANTA,R. 2005, PENELITIAN: PERANCANGAN DAN PEMBUATAN APLIKASI SEGMENTASI GAMBAR DENGAN MENGGUNAKAN METODE MORPHOLOGICAL WATERSHED.