UAS

PENGOLAHAN CITRA DIGITAL

SEGMENTASI CITRA

Sitti Nurlaili Qofifa

F55120107

C

PROGRAM STUDI S1 TEKNIK INFORMATIKA

JURURSAN TEKNOLOGI INFORMASI

FAKULTAS TEKNIK

UNIVERSITAS TADULAKO

2022

I. Alat dan bahan

A. laptop

B. matlab

II. Teori dasar

Segmentasi citra merupakan bagian dari proses pengolahan citra. Proses segmentasi citra ini lebih banyak merupakan suatu proses pra pengolahan pada sistem pengenalan objek dalam citra. Segmentasi citra *(image segmentation)* mempunyai arti membagi suatu citra menjadi wilayah-wilayah yang homogen berdasarkan kriteria keserupaan yang tertentu antara tingkat keabuan suatu piksel dengan tingkat keabuan piksel – piksel tetangganya, kemudian hasil dari proses segmentasi ini akan digunakan untuk proses tingkat tinggi lebih lanjut yang dapat dilakukan terhadap suatu citra, misalnya proses klasifikasi citra dan proses identifikasi objek. Adapun dalam proses segmentasi citra itu sendiri terdapat beberapa algoritma, diantaranya : algoritma Deteksi Titik, Deteksi Garis, dan Deteksi Sisi (berdasarkan Operator Robert dan Operator Sobel).

Gonzalez dan Wintz (1987) menyatakan bahwa segmentasi adalah proses pembagian sebuah citra kedalam sejumlah bagian atau obyek. Segmentasi merupakan suatu bagian yang sangat penting dalam analisis citra secara otomatis, sebab pada prosedur ini obyek yang diinginkan akan disadap untuk proses selanjutnya, misalnya: pada pengenalan pola. Algoritma segmentasi didasarkan pada 2 buah karakteristik nilai derajad kecerahan citra, yaitu: *discontinuity dan similarity*. Pada item pertama, citra dipisahkan/dibagi atas dasar perubahan yang mencolok dari derajad kecerahannya. Aplikasi yang umum adalah untuk deteksi titik, garis, area, dan sisi citra. Pada kategori kedua, didasarkan atas *thresholding*, region growing, dan region spiltting and merging. Prinsip segmentasi citra bisa diterapkan untuk citra yang statis maupun dinamis.

- *EDGE DETECTION* (DETEKSI TEPI)

Penentuan tepian suatu objek dalam citra merupakan salah satu wilayah pengolahan citra digital yang paling awal dan paling banyak diteliti. Proses ini seringkali ditempatkan sebagai langkah pertama dalam aplikasi segmentasi citra, yang bertujuan untuk mengenali objek-objek yang terdapat dalam citra ataupun konteks citra secara keseluruhan.

Deteksi tepi berfungsi untuk mengidentifikasi garis batas (boundary) dari suatu objek yang terdapat pada citra. Tepian dapat dipandang sebagai lokasi piksel dimana terdapat

nilai perbedaan intensitas citra secara ekstrem. Sebuah *edge detector* bekerja dengan cara mengidentifikasi dan menonjolkan lokasi-lokasi piksel yang memiliki karakteristik tersebut.

Ada banyak cara-cara untuk mengintifikasikan bagian tepi suatu citra, diantaranya adalah sebagai berikut :

$$\nabla f(x,y) = f_x + f_y = \frac{\partial}{\partial_x} f(x,y) + \frac{\partial}{\partial_y} f(x,y)$$

- OPERATOR *GRADIEN*

Pada citra digital f(x,y), turunan berarah sepanjang tepian objek akan bernilai maksimum pada arah normal dari kontur tepian yang bersesuaian. Sifat ini dipergunakan sebagai dasar pemanfaatan operator gradien sebagai edge detector.

Operator gradien citra konvensional melakukan diferensiasi intensitas piksel pada arah baris dan kolom, mengikuti persamaan *local intensity variation* berikut :

Nilai *magnitudo gradien*:

$$|\nabla(x,y)|$$

Dari persamaan di atas dapat dinyatakan sebagai berikut:

$$|\nabla(x,y)| = \sqrt{(f_x)^2 + (f_y)^2}$$

Operator gradien dapat direpresentasikan oleh dua buah kernel konvolusi Gx dan Gy, yang masing-masing mendefinisikan operasi penghitungan gradien dalam arah sumbu x dan sumbu y yang saling tegak lurus.

Dalam kasus penghitungan gradien dengan persamaan *local intensity variation*, maka kernel Gx dan Gy dapat dirumuskan seperti berikut:

$$G_x = [-1 \;\; 1]$$

$$G_y = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

Dari operator gradien konvensional di atas, dapat diturunkan berbagai *operator gradien* berikut :

1.Operator Roberts

2.Operator Prewit

3.Operator Sobel

- OPERATOR LAPLACIAN

Dalam kondisi transisi tepian yang lebih tidak ekstrem, penggunaan operator turunan kedua lebih dianjurkan.

$$\nabla^2 f(x,y) = f_{xx} + f_{yy} = \frac{\partial}{\partial_x} f_x(x,y) + \frac{\partial}{\partial_y} f_y(x,y)$$

$$\nabla^2 f(x,y) = \frac{\partial^2}{\partial_x^2} f_x(x,y) + \frac{\partial^2}{\partial_y^2} f_y(x,y)$$

Turunan kedua memiliki sifat lebih sensitif terhadap noise, selain itu juga menghasilkan double edge. Oleh karena itu, operator Laplacian dalam deteksi tepi pada umumnya tidak dipergunakan secara langsung, namun dikombinasikan dengan suatu kernel Gaussian menjadi sebuah operator Laplacian of Gaussian.

Fungsi transfer dari kernel Laplacian of Gaussian dapat dirumuskan sebagai berikut:

$$G(x,y) = -\frac{1}{2\pi\sigma^2} e^{-[x^2+y^2]/2\sigma^2}$$

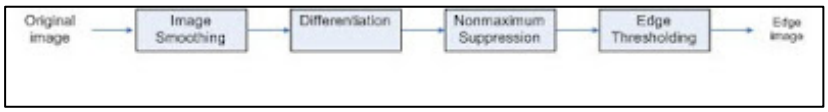
$$\nabla^2(G(x,y)) = \frac{1}{\pi\sigma^4}\left[1 - \frac{x^2+y^2}{2\sigma^2}\right] e^{-[x^2+y^2]/2\sigma^2}$$

- OPERATOR *ZERRO CROSS*

Metode *Zero-cross* menemukan edge dengan cara mencari *zero crossings* setelah memfilter I (Identitas) dengan filter a yang telah ditentukan.

- OPERATOR *CANNY*

Salah satu algoritma deteksi tepi modern adalah deteksi tepi dengan menggunakan metoda Canny. Berikut adalah diagram blok algoritma *Canny :*



III. Proses segmentasi citra

Pada percobaan segmentasi citra kali ini gambar yang akan diolah dan langsung dimasukkan pada kode program, gambar yang akan digunakan bersumber dari google dengan ekstensi png. https://images.app.goo.gl/QXbwLXWMGqyV5kU56

Gambar ini akan dilah menjadi Grayscale, hvs pictures, median filtring, opening, dan bitwise and.

IV. Source kode dan hasil program

A. kode program

Kode program :

```
# import library python package
import cv2
import numpy as np

# membaca file gambar upin ipin.png yang berada dalam folder  yang sama dengan file
image = cv2.imread("upin ipin.png")

# menampilkan data piksel citra
print('Image', image)

# menampilkan citra original
cv2.imshow("citra Original", image)

# menampilkan citra grayscale
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
cv2.imshow("Grayscale", gray)

# menampilkan citra HSV
hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
cv2.imshow("HSV Picture", hsv)

# median Filter
img_median = cv2.medianBlur(hsv, 5)  # menambahkan median filter ke image
cv2.imshow('Median Filter 0', img_median)

# Applying otau in Green channel
img = img_median[:, :, 1]
th, img = cv2.threshold(img, 0, 255, cv2.THRESH_OTSU)
cv2.imshow("otau Tresholding", th)
```

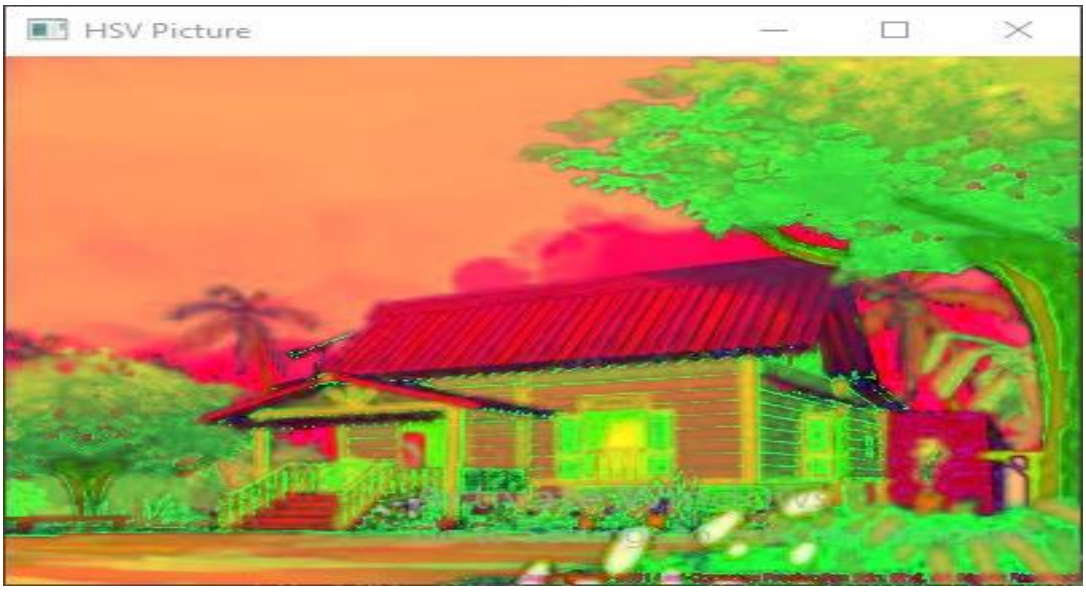
```
# opening
kernel = np.ones((7, 7), np.uint8)
opening = cv2.morphologyEx(img, cv2.MORPH_OPEN, kernel)
cv2.imshow("Opening", opening)

# aritmatic and operation
dest_and = cv2.bitwise_and(image, image, mask=opening)
cv2.imshow('bitwase and', dest_and)

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

B. Hasil program

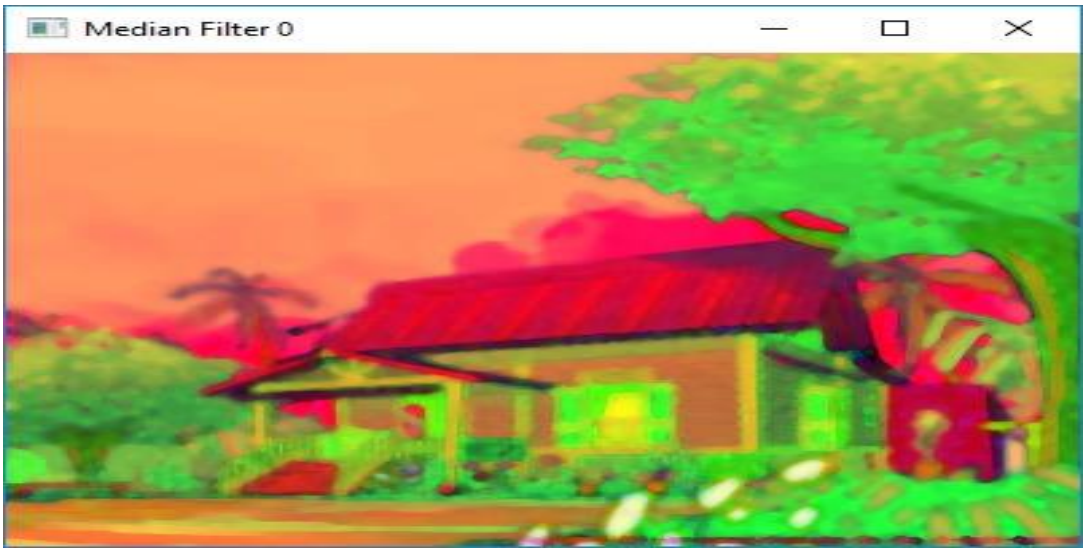
- Original

- *Grayscale*

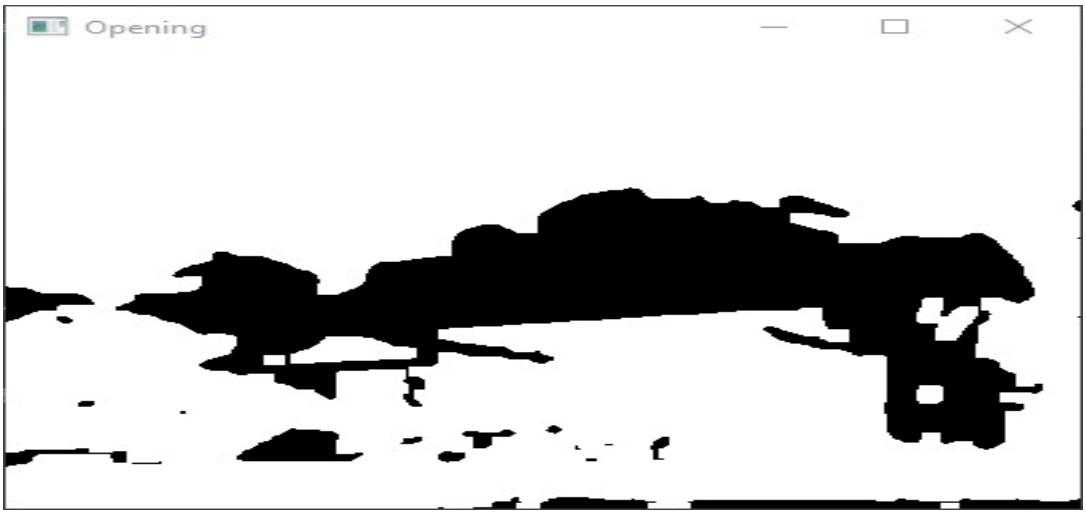
- *HSV Picture*



- *Median filtering*



- *Opening*



- *Bitwise and*

V. Analisis

seperti sebelumnya, yaitu mengimportkan library dulu. Lalu kemudian, masukkan kode cv2.imread pada variabel image untuk membaca gambar yang sudah disiapkan. Kemudian, tampilkan gambar ori dengan kode cv2.imshow(). Kemudian, untuk mengubah citra gambarnya menjadi grayscale masukkan kode ini gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY) dan untuk menampilkan citra hsv masukkan kode hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV). Lalu, buat variabel img_median untuk menjalankan method cv2.medianBlur yang berfungsi untuk memberikan efek blur pada gambar dengan menggunakan filter median. Kemudian, kode cv2.threshold yang dimasukkan pada variabel th, img. Dimana dilengkapi dengan cv.THRESH_OTSU yang memberikan ekstra algoritma yang menemukan nilai threshold optimal yangmana akan mengembalikan output yang pertama. Kemudian, terdapat proses opening yang sama seperti dimodul 5, dengan menggunkan kode cv2.morphologyEx. lalu terakhir, terdapat method dengan kode cv2.bitwise_and yang berfungsi untuk menampilkan area yang beririsan.