**Pengolahan Citra Digital**

**Dokumentasi**

****

Disusun Oleh :

Muhammad Rhifa Tjikoa (226201039)

**POLITEKNIK NEGERI SAMARINDA**

**2024**

1. **Erotion**
2. **Definisi**

*Erotion* atau Erosi merupakan proses pada morfologi citra yang mengecilkan batasan dari sebuah objek pada citra. Erosi menghilangkan piksel dalam daerah objek citra dengan cara menempatkan pusat tata elemen satu per satu dalam piksel *foreground* (bernilai 1). Jika piksel *neighbourhood* bernilai piksel *background* (bernilai 0), maka nilai *foreground* tersebut diubah ke *background*.

1. **Model Persamaan**

Berikut model persamaan atau notasi dari erosi :

Dimana SE merupakan kesalahan deviasi yang merupakan perkiraan dari deviasi standar tersebut. Adapun beberapa model persamaan dari erosi, yaitu :

Dimana A-b melambangkan translasi dari A oleh -b.

1. **Tujuan Penerapan**

Tujuan penerapan dari Erosi adalah untuk memperkecil struktur.

1. **Source Code**

import cv2

import numpy as np

gambar = cv2.imread(“C:\\Users\\ASUS\\Downloads\\presentasi\\gildedguy.webp”, 0)

kernel = np.ones((5,5) , np.uint8)

hasilErosi = cv2.erode(gambar, kernel, iterations = 1)

cv2.imshow("Erosi", hasilErosi)

cv2.waitKey(0)

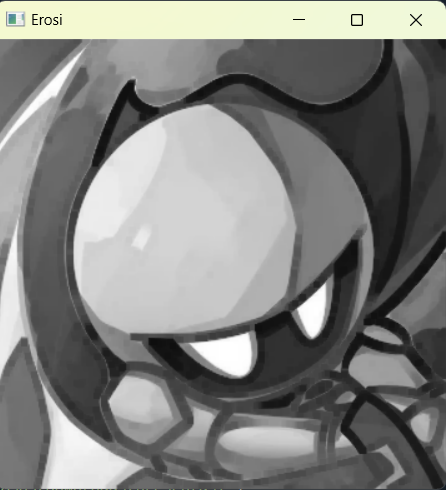
cv2.destroyAllWindows()

1. **Output Program**

Gambar awal sebelum di lakukan operasi Erosi :

****

Gambar setelah dilakukan operasi Erosi :

****

1. **Dilation**
2. **Definisi**

*Dilation* atau Dilasi merupakan proses didalam morfologi citra yang membesarkan batasan dari sebuah objek pada citra. Dilasi menambahkan piksel pada daerah citra asal, dengan cara menempatkan satu per satu pusat tata elemen untuk setiap pixel pada *background*. Bila salah satu piksel *background* bernilai piksel *foreground* (nilai 1) maka piksel *background* diubah ke *foreground*.

1. **Model Persamaan**

Berikut model persamaan *Dilation* atau dilasi :

SE berarti *Standard Error* atau kesalahan standar yang merupakan perkiraan dari deviasi standar tersebut. Adapun beberapa model persamaan dilasi yang lain, yaitu :

Dimana Ab merupakan translasi A oleh B.

1. **Tujuan Penerapan**

Salah satu penerapan Dilasi yaitu, objek-objek yang terputus yang disebabkan oleh hasil pengambilan citra yang terganggu oleh *noise*, kerusakan objek fisik yang dijasikan citra digitalm atau disebbkan resolusi yang jelek. Misalnya, teks pada kertas yang kelihatan agak rusak sehingga bentuk huruf terputus.

1. **Source Code**

import cv2

import numpy as np

gambar = cv2.imread(“C:\\Users\\ASUS\\Downloads\\presentasi\\gildedguy.webp”, 0)

kernel = np.ones((5,5) , np.uint8)

hasilDilasi = cv2.dilate(gambar, kernel, iterations = 1)

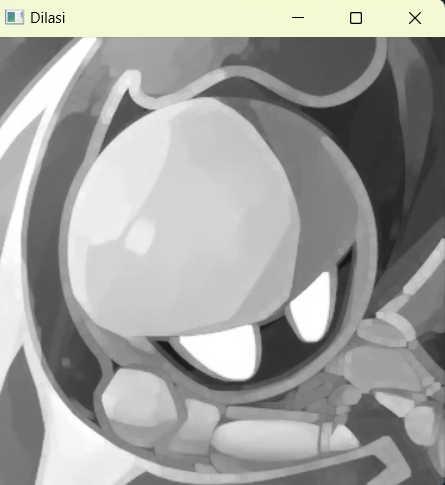
cv2.imshow("Dilasi", hasilDilasi)

cv2.waitKey(0)

cv2.destroyAllWindows()

1. **Output Program**

Berikut gambar setelah di lakukan operasi Dilasi :



1. **Opening**
2. **Definisi**

*Opening* menghaluskan garis-garis bentuk objek, menghilangkan bagian-bagian yang sempit dan menghilangkan penonjalan yang tipis.

1. **Model Persamaan**

Berikut model persamaan dari *Opening*, yaitu :

1. **Tujuan Penerapan**

Salah satu tujuan penerapan *Opening*, untuk mencari bentuk spesifik di dalam sebuah citra dan untuk mencari hal-hal yang dimana sebuah *structure element* dapat masuk seperti tepi dan sudut.

1. **Source Code**

import cv2

import numpy as np

gambar = cv2.imread(“C:\\Users\\ASUS\\Downloads\\presentasi\\gildedguy.webp”, 0)

kernel = np.ones((5,5) , np.uint8)

hasilOpening = cv2.morphologyEx(gambar, cv2.MORPH\_OPEN, kernel)

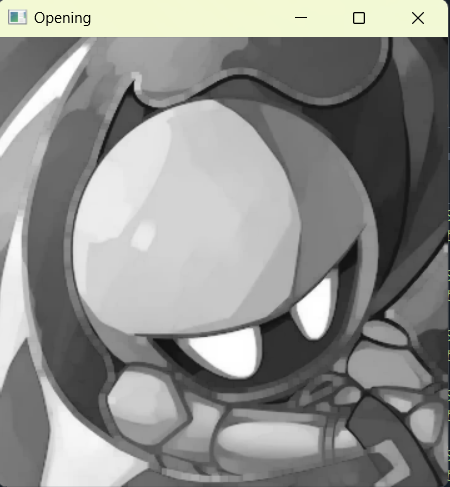
cv2.imshow("Opening", hasilOpening)

cv2.waitKey(0)

cv2.destroyAllWindows()

1. **Output Program**

Berikut gambar setelah dilakukan operasi *Opening* :



1. **Closing**
2. **Definisi**

*Closing* juga cenderung menghaluskan bagian-bagian dari garis bentuk objek, tetapi karena merupakan kebalikan dari *Opening*. Operasi ini biasanya menghilangkan celah kecil, mengisi celah-celah di garis bentuk objek, menggabungkan pecahan kecil dan jurang tipis panjang.

1. **Model Persamaan**

Berikut model persamaan dari *Closing* :

1. **Tujuan Penerapan**

Salah satu tujuan penerapan dari *Closing* yaitu untuk mengisi garis terputus pada pengenalan pola *fingerprint.*

1. **Source Code**

import cv2

import numpy as np

gambar = cv2.imread(“C:\\Users\\ASUS\\Downloads\\presentasi\\gildedguy.webp”, 0)

kernel = np.ones((5,5) , np.uint8)

hasilClosing = cv2.morphologyEx(gambar, cv2.MORPH\_CLOSE, kernel)

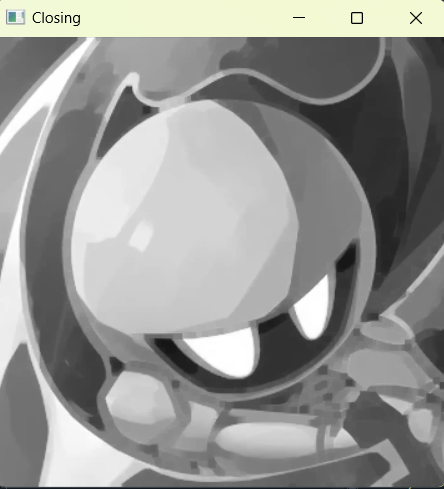
cv2.imshow("Closing", hasilClosing)

cv2.waitKey(0)

cv2.destroyAllWindows()

1. **Output Program**

Berikut gambar setelah dilakukan operasi *Closing* :

****

1. **Hit or Miss**
2. **Definisi**

*Hit-or-miss* merupakan sebuah operasi untuk mencari atau mendeteksi konfigurasi pola yang telah diberikan dengan menggunakan operasi Erosi dan sepasang elemen penataan yang terpisah-pisah

1. **Model Persamaan**

Berikut model persamaan dari *Hit-or-Miss* :

Dimana C dan D merupakan sepasang elemen penataan yang terpisah dan AC merupakan complement dari A.

1. **Tujuan Penerapan**

Penerapan dapat bervariasi seperti bagaimana operasi ini di tangani pada tepi dari citra dimana *structure* *element* tumpang tindih pada tepi dari citra. Solusi sederhananya adalah hanya cukup asumsi bahwa *structure element* apapun yang tumpang tindih citra tidak sama dengan piksel yang mendasari, dan oleh karena itu piksel yang sesuai pada output harus di ubah menjadi niol

1. **Source Code**

# importing required libraries

import mahotas as mh

import numpy as np

from pylab import imshow, show

# creating region

# numpy.ndarray

daerah = np.zeros((10, 10), bool)

# setting 1 value to the region

daerah[1, :4] = 1

daerah[1:8, 6: 10] = 1

daerah[4, 0] = 1

# showing the image with interpolation = 'nearest'

print("Gambae")

imshow(daerah, interpolation ='nearest')

show()

# template for hit miss

template = np.array([

[0, 1, 1],

[0, 1, 1],

[0, 1, 1]])

# hit miss transform

gambar = mh.hitmiss(daerah, template)

# showing image

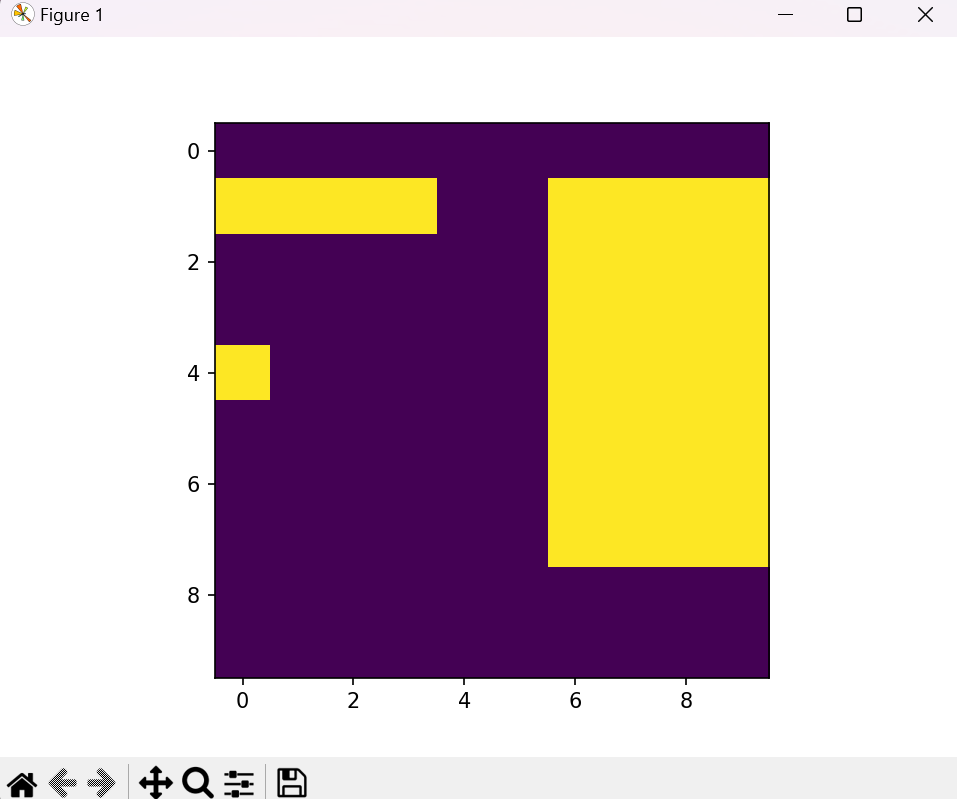
print("Gambar setelah transformasi hit miss")

imshow(gambar)

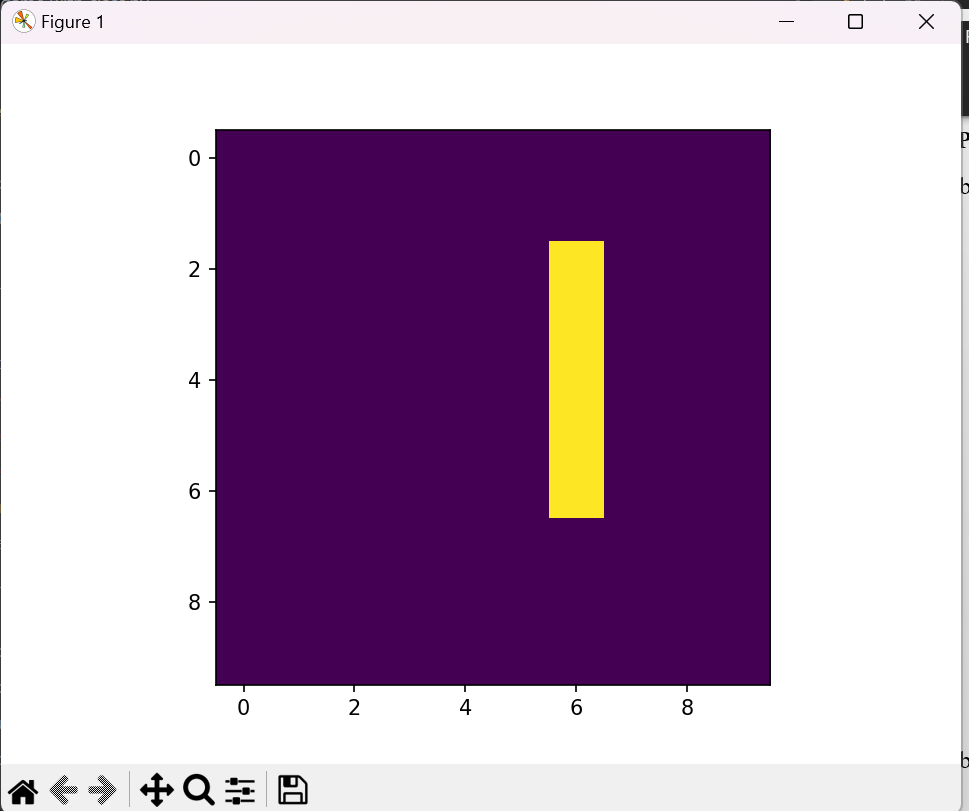
show()

1. **Output Program**

Berikut gambar sebelum *Hit-Or-Miss* :



Berikut gambar setelah *Hit-Or-Miss* :



1. **Thinning**
2. **Definisi**

*Thinning* merupakan sebuah operasi morfologi yang digunakan untuk menghilangkan piksel *foreground* yang dipilih dari citra. Operasi ini seperti operasi erosi atau *opening*.

1. **Model Persamaan**

Berikut model persamaan dari *Thinning* :

1. **Tujuan Penerapan**

Operasi ini dapat digunakan ke beberapa aplikasi, tetapi biasanya operasi ini berguna untuk *skeletonization*. Dalam mode ini biasanya digunakan untuk membersikan output dari pendeteksi tepi dengan cara mengurangi semua garis-garis menjadi ketebalan dengan ukuran satu piksel.

1. **Source Code**

import cv2

gambar = cv2.imread(“C:\\Users\\ASUS\\Downloads\\presentasi\\gildedguy.webp”, 0)

thinning = cv2.ximgproc.thinning(gambar)

cv2.imshow("Thinning", thinning)

cv2.waitKey(0)

cv2.destroyAllWindows()

1. **Output Program**

Berikut gambar setelah dilakukan operasi *Thinning*:



1. **Thickening**
2. **Definisi**

*Thickening* merupakan sebuah operasi morfologi yang digunakan untuk menumbuhkan daerah yang terpilih dari piksel *foreground* pada citra. Operasi ini seperti operasi dilasi atau *closing*.

1. **Model Persamaan**

Berikut model persamaan dari *Thickening* :

1. **Tujuan Penerapan**

Operasi ini memiliki penerapan, termasuk menentukan perkiraan *convex hull* dari sebuah bentuk, dan menghasilkan citra biner lain sebagai output.

1. **Source Code**

import cv2

import numpy as np

gambar = cv2.imread(“C:\\Users\\ASUS\\Downloads\\presentasi\\gildedguy.webp”, 0)

\_, gambarBiner = cv2.threshold(gambar, 128, 255, cv2.THRESH\_BINARY)

gambar\_cp = gambar.copy()

kernel = np.ones((3, 3), np.uint8)

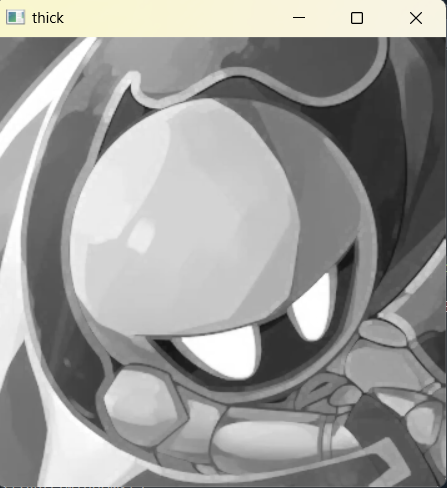
thick = cv2.dilate(gambar, kernel, iterations=1)

cv2.imshow("thick", thick)

cv2.waitKey(0)

cv2.destroyAllWindows()

1. **Output Program**

****

1. **Boundary Extraction**
2. **Definisi**

*Boundary Extraction* merupakan proses untuk mendapatkan batas tepi atau *boundary* objek citra. Proses ini pertama kali menggunakan proses erosi dengan blok penataan elemen yang kecil, lalu hasilnya dikurangi dengan citra asal.

1. **Model Persamaan**

Berikut model persamana dari *Boundary Extraction* :

1. **Tujuan Penerapan**

Salah satu penerapan operasi ini yaitu untuk mengambil tepi dari sebuah objek pada citra yang digunakan untuk pengenalan objek.

1. **Source Code**

import cv2

import numpy as np

gambar = cv2.imread(“C:\\Users\\ASUS\\Downloads\\presentasi\\gildedguy.webp”, 0)

\_, gambarBiner = cv2.threshold(gambar, 128, 255, cv2.THRESH\_BINARY)

gambar\_cp = gambar.copy()

kernel = np.ones((3, 3), np.uint8)

thick = cv2.dilate(gambar, kernel, iterations=1)

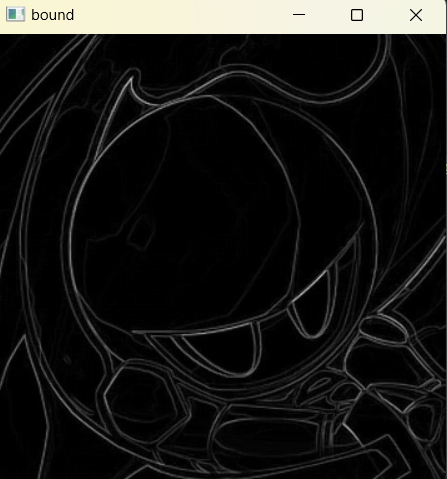
bound = cv2.subtract(thick, gambar)

cv2.imshow("bound", bound)

cv2.waitKey(0)

cv2.destroyAllWindows()

1. **Output Program**

******

1. **Region Filling**
2. **Definisi**

*Region Filling* merupakan operasi morfologi yang digunakan untuk mengisi daerah didalam *boundary* , daerah atau area didalam sebuah citra dengan warna, pola atai intesitas nilai.

1. **Model Persamaan**
2. **Tujuan Penerapan**

Penerapan operasi ini dapat ditemukan pada aplikasi pengolahan citra seperti *Photoshop* atau *GIMP*, yaitu mengisi suatu daerah didalam *boundary* dengan warna.

1. **Source Code**

import cv2

gambar = cv2.imread("C:\\Users\\ASUS\\Pictures\\square.png", 0)

garis = cv2.findContours(gambar, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)

garis = garis[0] if len(garis) == 2 else garis[1]

for g in garis:

cv2.drawContours(gambar,[g], 0, (255,255,255), -1)

kernel = cv2.getStructuringElement(cv2.MORPH\_RECT, (20,20))

regionFill = cv2.morphologyEx(gambar, cv2.MORPH\_OPEN, kernel, iterations=2)

cv2.imshow('gray', gambar)

cv2.imshow('region fill', regionFill)

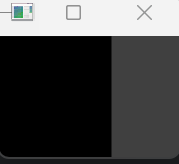
cv2.waitKey(0)

1. **Output Program**

Berikut gambar sebelum dilakukan operasi *Region Fill* :



Berikut gambar setelah dilakukan operasi *Region Fill* :



1. **Connected Components**
2. **Definisi**

Operasi *Connected Components* sama seperti Operasi *Region Filling*, hanya perbedaannya operasi ini menggunakan A bukan *complement* dari A.

1. **Model Persamaan**
2. **Tujuan Penerapan**

Salah penerapan operasi ini untuk mendeteksi objek asing didalam makanan kemasan.

1. **Source Code**

import cv2

import numpy as np

gambar = cv2.imread("C:\\Users\\ASUS\\Pictures\\square.png", 0)

gambar = cv2.threshold(gambar, 127, 255, cv2.THRESH\_BINARY)[1]

numberLabel, gambarLabel = cv2.connectedComponents(gambar)

hueLabel = np.uint8(179\*gambarLabel/np.max(gambarLabel))

emptyChannel = 255\*np.ones\_like(hueLabel)

gambarHasil = cv2.merge([hueLabel, emptyChannel, emptyChannel])

gambarHasil = cv2.cvtColor(gambarHasil, cv2.COLOR\_HSV2BGR)

gambarHasil[hueLabel==0] = 0

cv2.imshow("connectedComponents", gambarHasil)

cv2.waitKey(0)

cv2.destroyAllWindows()

1. **Output Program**

Berikut gambar setelah dilakukan proses *Connected Components* :

