LAPORAN EKSTRASI CITRA  
CITRA DIGITAL

****

**Ditulis oleh:**

**Renaldi Septian**

**NIM. 226201029**

**D3 TEKNIK KOMPUTER**

**TEKNOLOGI INFORMASI**

**POLITEKNIK NEGERI SAMARINDA**

**2024**

# DAFTAR ISI

[DAFTAR ISI i](#_Toc181745130)

[BAB I PENDAHULUAN 2](#_Toc181745131)

[1.1 Teori Ekstraksi 2](#_Toc181745132)

[BAB II SOURCE CODE 4](#_Toc181745133)

[2.1 Program 1 4](#_Toc181745134)

[2.1.1 Penjelasan 4](#_Toc181745135)

[2.1.2 Code Program 4](#_Toc181745136)

[2.1.3 Output Program 8](#_Toc181745137)

[2.2 Program 2 8](#_Toc181745138)

[2.2.1 Penjelasan 8](#_Toc181745139)

[2.2.2 Code Program 9](#_Toc181745140)

[2.2.3 Output Program 12](#_Toc181745141)

[2.3 Program 3 12](#_Toc181745142)

[2.3.1 Penjelasan 12](#_Toc181745143)

[2.3.2 Code Program 13](#_Toc181745144)

[2.3.3 Output Program 17](#_Toc181745145)

[BAB III PENUTUP 18](#_Toc181745146)

[3.1 Kesimpulan 18](#_Toc181745147)

[3.2 Opini Penulis 18](#_Toc181745148)

[DAFTAR PUSTAKA 19](#_Toc181745149)

# PENDAHULUAN

## Teori Ekstraksi

Merupakan tahapan mengekstrak ciri/informasi dari objek di dalam citra yang ingin dikenali/dibedakan dengan objek lainnya.

1. Ekstraksi Ciri Bentuk

Ekstraksi ciri Bentuk Edge detection merupakan menemukan bagian pada citra yang mengalami perubahan intensitas secara drastis. Ada dua cara yang digunakan untuk menemukan bagian tersebut yaitu menggunakan turunan pertama, dimana intensitas magnitudonya lebih besar dari threshold yang didefinisikan dan menggunakan turunan kedua, dimana intensitas warnanya mempunyai zero crossing. Dalam penelitian ini menggunakan Canny Edge Detection yang secara umum menggunakan algoritma umum sebagai berikut: 1. Penghalusan untuk mengurangi dampak noise terhadap pendeteksian edge. 2. Menghitung potensi gradient citra 3. Non-maximal suppression dari gradient citra untuk melokalisasi edge secara presisi. 4. hysteresis thresholding untuk melakukan klasifikasi akhir Tahap berikutnya adalah perhitungan edge direction histogram menggunakan 5 bin arah yaitu 00 , 450 , 900 , 1350 , dan 1800 dengan nilai piksel ketetanggan yang sama sebanyak 3 piksel.

1. Ciri Tekstur

Ciri tekstur merupakan ciri penting dalam sebuah gambar yang merupakan informasi berupa susunan struktur permukaan suatu gambar. Dalam penelitian ini menggunakan Gray Level oCcurance Matrix (GLCM) sebagai matrik pengambilan nilai keabuan dari sebuah gambar. Berikut merupakan tahapan yang digunakan dalam pengambilan ciri tekstur dari sebuah gambar. 1. Citra warna dirubah menjadi citra grayscale. 2. Segmentasi nilai warna ke dalam 16 bin. 3. Hitung nilai-nilai co-occurance matrix dalam empat arah masing-masing 00 , 450 , 900 , dan 1350 4. Hitung informasi ciri tekstur yaitu yaitu contrast, correlation, energy, homogeneity, dan entropy 5. Masing-masing matriks akan dihitung tekstur citra yaitu : Contrast, Correlation, Energy, Homogeneity, dan Entropy. Jeremiah (2007).

1. Ciri Jarak

Titik koordinat titik akhir dan titik percabangan yang telah diperoleh dikelompokkan dan menjadi nilai masukan dalam

persamaan Euclidean Distance (1). Euclidean distance merupakan generalisasi dari teorema phytagoras[7]. Pada persamaan (1), x1 , x2 merupakan oordinat sumbu x dari sebuah titik dan y1, y2 merupakan oordinat sumbu y dari sebuah titik . 𝑑 = √(𝑥1 − 𝑥2) 2 + (𝑦1 − 𝑦2) 2 (1) Pada penelitian ini jarak dikelompokkan yaitu jarak dari tangan ke poros dada, jarak kepala ke poros dada, jarak kaki depan ke kaki belakang, jarak sudut tungkai depan ke sudut tungkai belakang.

1. Ciri Geomtri

Ciri geometri merupakan ciri yang didasarkan pada hubungan antara dua buah titik, garis, atau bidang dalam citra digital. Ciri geometri di antaranya adalah jarak dan sudut. Jarak antara dua buah titik (dengan satuan piksel) dapat ditentukan menggunakan persamaan euclidean, minkowski, manhattan, dll. Jarak dengan satuan piksel tersebut dapat dikonversi menjadi satuan panjang seperti milimeter, centimeter, meter, dll dengan cara membaginya dengan resolusi spasial. Sedangkan sudut antara dua buah garis dapat ditentukan dengan perhitungan trigonometri maupun dengan analisis vektor.

# SOURCE CODE

## Program 1

### Penjelasan

Kode di bawah memuat dan mengubah gambar menjadi skala abu-abu, kemudian diterapkan Gaussian Blur untuk mengurangi noise dan meningkatkan deteksi kontur. Selanjutnya, gambar diubah menjadi biner menggunakan thresholding. Operasi morfologi dilakukan untuk menghilangkan noise kecil dan mengisi celah. Deteksi tepi dilakukan menggunakan metode Canny, dan kontur ditemukan dari gambar hasil deteksi tepi. Kontur digambar pada gambar asli, dan parameter seperti eksentrisitas, luas, dan keliling dihitung untuk setiap kontur. Akhirnya, bentuk-bentuk diidentifikasi dan diberi label sebagai "Bulat" atau "Elips" berdasarkan nilai metrik yang dihitung. Gambar dengan bentuk-bentuk yang terdeteksi dan diberi label ditampilkan pada akhir proses.

### Code Program

import cv2

import numpy as np

# Load the image

image = cv2.imread('list kendaraan.png') # Ganti dengan path file gambar Anda

# Resize image for consistency

image = cv2.resize(image, (600, 600)) # Ubah ukuran gambar agar konsisten

# Convert to grayscale

gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY) # Konversi gambar ke skala abu-abu

# Apply GaussianBlur to reduce noise and improve contour detection

blurred = cv2.GaussianBlur(gray, (5, 5), 0) # Terapkan GaussianBlur untuk mengurangi noise dan meningkatkan deteksi kontur

# Threshold the image

\_, binary = cv2.threshold(blurred, 60, 255, cv2.THRESH\_BINARY\_INV) # Terapkan threshold untuk mengubah gambar ke biner

# Define kernels for morphological operations

kernelOpen = np.ones((5, 5), np.uint8) # Definisikan kernel untuk operasi morfologi open

kernelClose = np.ones((7, 7), np.uint8) # Definisikan kernel untuk operasi morfologi close

# Perform morphological operations

closing = cv2.morphologyEx(binary, cv2.MORPH\_CLOSE, kernelClose) # Lakukan operasi morfologi closing

opening = cv2.morphologyEx(closing, cv2.MORPH\_OPEN, kernelOpen) # Lakukan operasi morfologi opening

# Segment the image

segment = cv2.Canny(opening, 30, 200) # Segmentasi gambar menggunakan deteksi tepi Canny

contours, \_ = cv2.findContours(segment, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE) # Temukan kontur dalam gambar

# Draw contours on the original image

segment = cv2.cvtColor(segment, cv2.COLOR\_GRAY2RGB) # Konversi segmen ke RGB

cv2.drawContours(image, contours, -1, (0, 0, 255), 1) # Gambar kontur pada gambar asli

# Segmentation, clustering, labeling, and calculating parameters

centers = []

for j in range(len(contours)):

# Clustering and labeling

moments = cv2.moments(contours[j]) # Hitung momen dari setiap kontur

if moments['m00'] == 0:

continue

centers.append((int(moments['m10'] / moments['m00']), int(moments['m01'] / moments['m00']))) # Temukan pusat massa (centroid)

cv2.circle(image, centers[-1], 1, (0, 0, 255), -1) # Gambar lingkaran pada centroid

cv2.putText(image, str(j + 1), centers[-1], cv2.FONT\_HERSHEY\_DUPLEX, 0.75, (0, 255, 255), 1) # Beri label pada centroid

# Calculate parameters

(x, y), (minorAxis, majorAxis), angle = cv2.fitEllipse(contours[j]) # Dapatkan sumbu minor dan mayor

eccentricity = np.sqrt(1 - (minorAxis\*\*2 / majorAxis\*\*2)) # Hitung eksentrisitas

area = cv2.contourArea(contours[j]) # Hitung luas kontur

perimeter = cv2.arcLength(contours[j], True) # Hitung keliling kontur

metric = (4 \* np.pi \* area) / perimeter\*\*2 # Hitung metrik bulat

print("----------------------")

print(f'Eksentrisitas objek {j + 1} : {eccentricity}')

print(f'Luas objek {j + 1} : {area}')

print(f'Keliling objek {j + 1} : {perimeter}')

print(f'Metrik objek {j + 1} : {metric}')

# Label shapes based on metric

offset = (centers[-1][0], centers[-1][1] + 20) # Offset untuk penempatan teks

form = "Bulat" if round(metric, 1) >= 0.7 else "Elips" # Tentukan bentuk berdasarkan metrik

cv2.putText(image, str(form), offset, cv2.FONT\_HERSHEY\_DUPLEX, 0.5, (255, 0, 0), 1) # Beri label bentuk pada gambar

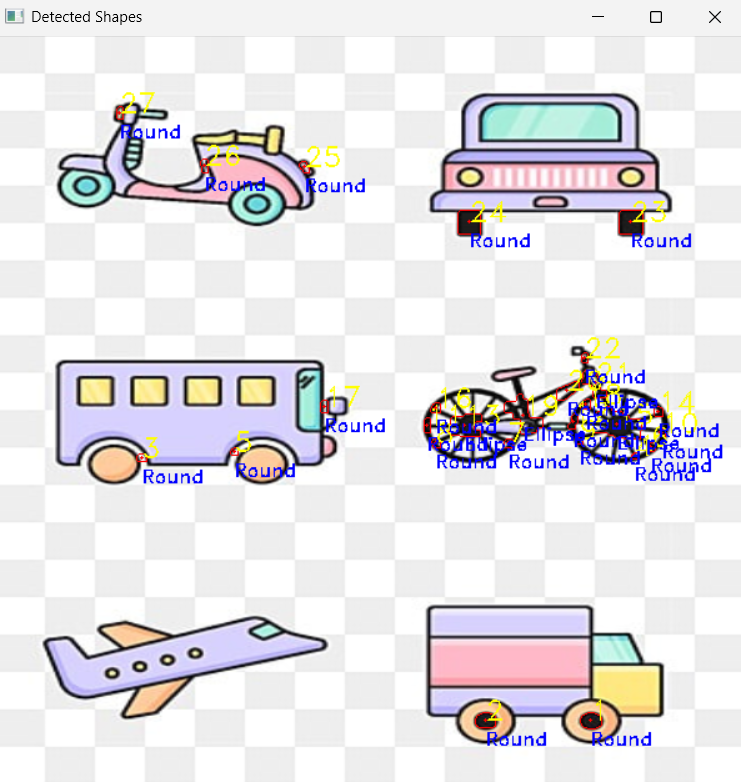
# Display the image

cv2.imshow('Detected Shapes', image) # Tampilkan gambar dengan bentuk terdeteksi

cv2.waitKey(0) # Tunggu input dari pengguna

cv2.destroyAllWindows() # Tutup semua jendela

### Output Program



## Program 2

### Penjelasan

Kode di bawah melakukan pemrosesan citra untuk mendeteksi dan menganalisis objek dalam dua gambar. Pertama, gambar dibaca dan diubah ukurannya menjadi 640x480 piksel. Kemudian, gambar diubah menjadi grayscale dan diterapkan thresholding untuk menghasilkan citra biner. Setelah itu, dilakukan operasi morfologi (closing dan opening) untuk menghilangkan noise dan mengisi lubang pada objek. Segmentasi dilakukan menggunakan deteksi tepi Canny, diikuti dengan pencarian kontur pada citra yang telah diproses. Setiap kontur dihitung centroid-nya, dan parameter geometris seperti area dan perimeter objek dihitung dan ditampilkan pada gambar. Hasil akhir adalah gambar yang telah digambar dengan kontur, centroid, serta informasi area dan perimeter untuk setiap objek yang terdeteksi.

### Code Program

import cv2

import numpy as np

# Membaca gambar dari file

images = [

cv2.imread("lahan1.png"), # Membaca gambar pertama

cv2.imread("lahan2.png") # Membaca gambar kedua

]

# Menentukan parameter untuk operasi morfologi

closeIterations = [4, 7] # Iterasi untuk operasi closing pada setiap gambar

openIterations = [20, 1] # Iterasi untuk operasi opening pada setiap gambar

kernelOpen = np.ones((5, 5), np.uint8) # Kernel untuk operasi opening (5x5 matriks)

kernelClose = np.ones((5, 5), np.uint8) # Kernel untuk operasi closing (5x5 matriks)

# Proses untuk setiap gambar dalam list 'images'

for i in range(len(images)):

# Mengubah ukuran gambar menjadi 640x480

images[i] = cv2.resize(images[i], (640, 480)) # Menyesuaikan ukuran gambar agar lebih kecil

# Mengubah gambar menjadi grayscale (hitam putih)

gray = cv2.cvtColor(images[i], cv2.COLOR\_BGR2GRAY) # Mengubah gambar RGB ke grayscale

# Melakukan thresholding untuk mendapatkan citra biner (hitam-putih)

\_, binary = cv2.threshold(gray, 127, 255, cv2.THRESH\_BINARY) # Thresholding dengan nilai 127

# Melakukan operasi closing dan opening pada citra biner

closing = cv2.morphologyEx(binary, cv2.MORPH\_CLOSE, kernelClose, iterations=closeIterations[i]) # Operasi closing

opening = cv2.morphologyEx(closing, cv2.MORPH\_OPEN, kernelOpen, iterations=openIterations[i]) # Operasi opening

# Segmentasi citra menggunakan Canny edge detection

segment = cv2.Canny(opening, 200, 200) # Menyaring tepi objek menggunakan Canny dengan threshold 200

# Menemukan kontur (garis tepi) dari hasil segmentasi

contours, hierarchy = cv2.findContours(segment, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_NONE) # Mencari kontur

# Mengonversi citra segmentasi ke format RGB agar bisa menggambar kontur

segment = cv2.cvtColor(segment, cv2.COLOR\_GRAY2RGB) # Konversi ke RGB untuk visualisasi

# Menggambar semua kontur pada gambar asli

cv2.drawContours(images[i], contours, -1, (0, 0, 255), 1) # Menggambar kontur berwarna merah pada gambar asli

# Segmentasi, clustering, pemberian label, dan menghitung parameter pada citra

centers = [] # Menyimpan koordinat centroid (titik tengah) dari setiap kontur

for j in range(len(contours)):

# Menghitung centroid (titik tengah) dari setiap kontur

moments = cv2.moments(contours[j]) # Menghitung moment kontur untuk menemukan centroid

if moments['m00'] != 0: # Memastikan tidak ada pembagian dengan nol

center\_x = int(moments['m10'] / moments['m00']) # Menghitung koordinat x centroid

center\_y = int(moments['m01'] / moments['m00']) # Menghitung koordinat y centroid

centers.append((center\_x, center\_y)) # Menambahkan centroid ke list

# Menggambar titik bulat pada posisi centroid

cv2.circle(images[i], centers[-1], 1, (0, 0, 255), -1) # Menggambar titik merah di centroid

# Memberi nomor pada centroid untuk identifikasi

cv2.putText(images[i], str(j + 1), centers[-1], cv2.FONT\_HERSHEY\_DUPLEX, 0.75, (0, 255, 255), 1) # Nomor urut centroid

# Menghitung parameter kontur: area dan perimeter

area = round(cv2.contourArea(contours[j]), 2) # Menghitung area kontur

perimeter = round(cv2.arcLength(contours[j], True), 2) # Menghitung perimeter kontur

# Menampilkan nilai area dan perimeter pada gambar di dekat centroid

offset = (centers[-1][0], centers[-1][1] + 20) # Menentukan posisi teks untuk area

cv2.putText(images[i], f"Area = {area}", offset, cv2.FONT\_HERSHEY\_DUPLEX, 0.5, (255, 0, 0), 1) # Menampilkan area

offset = (centers[-1][0], centers[-1][1] + 40) # Menentukan posisi teks untuk perimeter

cv2.putText(images[i], f"Perimeter = {perimeter}", offset, cv2.FONT\_HERSHEY\_DUPLEX, 0.5, (0, 255, 0), 1) # Menampilkan perimeter

# Menampilkan gambar yang telah diproses

for i in range(len(images)):

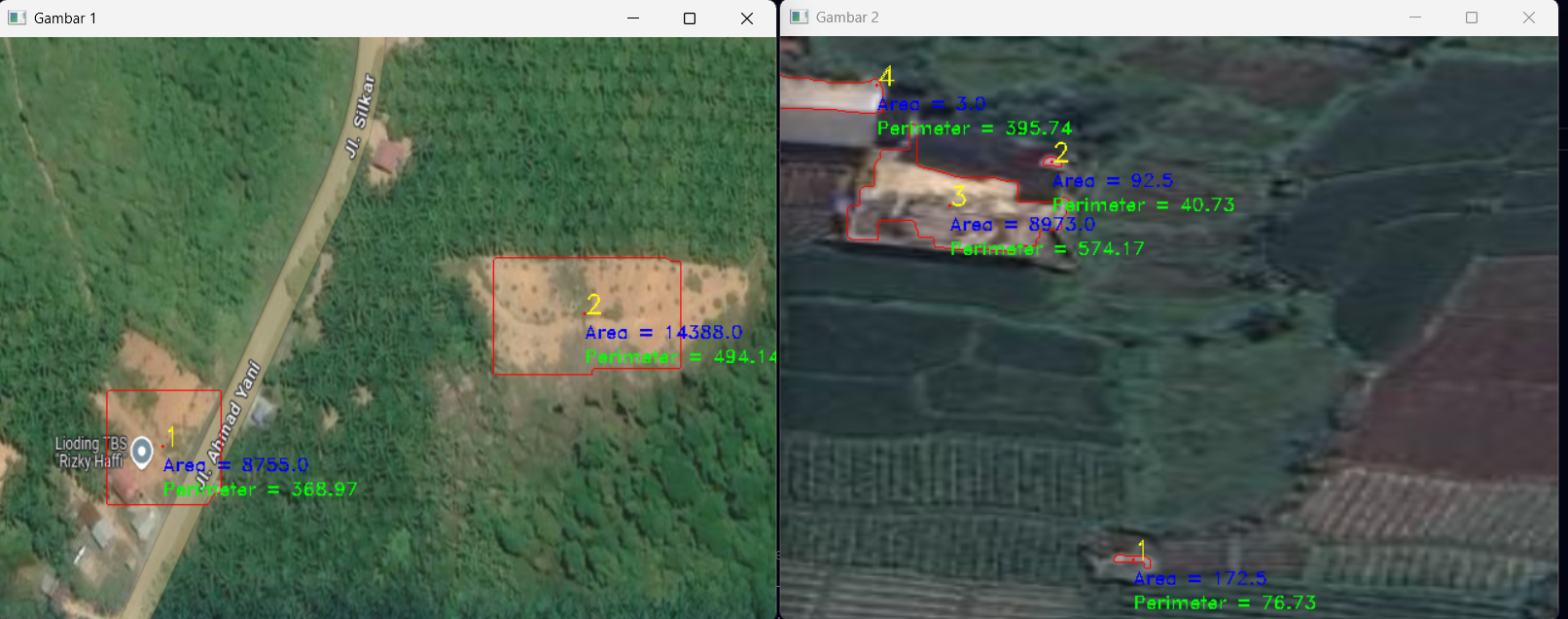
cv2.imshow(f'Gambar {i+1}', images[i]) # Menampilkan gambar dengan kontur dan parameter

# Menunggu hingga tombol ditekan dan menutup jendela

cv2.waitKey(0) # Menunggu input pengguna

cv2.destroyAllWindows() # Menutup semua jendela gambar

### Output Program



## Program 3

### Penjelasan

Kode ini melakukan pemrosesan citra untuk mendeteksi objek dan menghitung jarak antara centroid objek pada dua gambar menggunakan OpenCV. Pertama, gambar dibaca dan diubah ukurannya menjadi 640x480 piksel. Kemudian, gambar diubah menjadi grayscale dan dilakukan thresholding untuk menghasilkan citra biner. Setelah itu, operasi morfologi seperti closing dan opening diterapkan untuk membersihkan citra, diikuti dengan segmentasi menggunakan deteksi tepi Canny. Kontur objek dalam citra ditemukan, dan centroid (titik tengah) dari setiap kontur dihitung dan ditandai pada gambar. Jarak Euclidean antara dua centroid dihitung dalam satuan piksel dan kemudian dikonversi menjadi kilometer atau meter menggunakan faktor skala yang telah ditentukan. Akhirnya, hasilnya, termasuk jarak antar objek, digambarkan pada gambar dan ditampilkan.

### Code Program

import cv2

import numpy as np

# # Membaca gambar pertama dan kedua

images = [

cv2.imread("lahan1.png"), # Membaca gambar pertama

cv2.imread("lahan2.png") # Membaca gambar kedua

]

# # Fungsi untuk menghitung titik tengah antara dua titik (centroid)

def getCenterofLine(object1, object2, x=0, y=0):

offset = (int(object1[0] / 2) + int(object2[0] / 2) + x,

int(object1[1] / 2) + int(object2[1] / 2) + y)

return offset

# # Menentukan iterasi untuk operasi morfologi

closeIterations = [7, 2]

openIterations = [1, 4]

kernelOpen = np.ones((5, 5), np.uint8) # # Kernel untuk operasi opening (5x5 matriks)

kernelClose = np.ones((5, 5), np.uint8) # # Kernel untuk operasi closing (5x5 matriks)

centers = [] # # List untuk menyimpan posisi centroid

# # Proses untuk setiap gambar

for i in range(len(images)):

images[i] = cv2.resize(images[i], (640, 480)) # # Mengubah ukuran gambar menjadi 640x480

gray = cv2.cvtColor(images[i], cv2.COLOR\_BGR2GRAY) # # Mengubah gambar menjadi grayscale

\_, binary = cv2.threshold(gray, 127, 255, cv2.THRESH\_BINARY) # # Melakukan thresholding

# # Melakukan operasi closing dan opening pada citra

closing = cv2.morphologyEx(binary, cv2.MORPH\_CLOSE, kernelClose, iterations=closeIterations[i])

opening = cv2.morphologyEx(closing, cv2.MORPH\_OPEN, kernelOpen, iterations=openIterations[i])

# # Segmentasi citra dengan Canny edge detection

segment = cv2.Canny(opening, 200, 200)

# # Menemukan kontur dari hasil segmentasi

contours, hierarchy = cv2.findContours(segment, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_NONE)

segment = cv2.cvtColor(segment, cv2.COLOR\_GRAY2RGB) # # Mengonversi citra segmentasi menjadi RGB

# # Menggambar setiap kontur pada gambar

for j in range(len(contours)):

moments = cv2.moments(contours[j]) # # Menghitung momen kontur untuk mendapatkan centroid

if moments['m00'] != 0: # # Menghindari pembagian dengan nol

center\_x = int(moments['m10'] / moments['m00']) # # Menghitung koordinat x centroid

center\_y = int(moments['m01'] / moments['m00']) # # Menghitung koordinat y centroid

centers.append((center\_x, center\_y)) # # Menambahkan centroid ke list

# # Menggambar titik pada posisi centroid

cv2.circle(images[i], centers[-1], 1, (0, 0, 255), -1) # Titik merah di centroid

# # Memberikan nomor pada setiap centroid

cv2.putText(images[i], str(j + 1), centers[-1], cv2.FONT\_HERSHEY\_DUPLEX, 0.75, (255, 0, 0), 1)

# # Menghitung jarak antara centroid pada gambar pertama (dalam piksel dan kilometer)

if len(centers) > 1:

pixelDistance = cv2.norm(centers[0], centers[1], cv2.NORM\_L2) # # Menghitung jarak Euclidean antara centroid

kmDistance = pixelDistance / 40.1836969001148 # # Mengkonversi jarak piksel ke kilometer (1 km = 40 px)

# # Menggambar garis antara kedua centroid dan menampilkan jarak dalam piksel

cv2.line(images[i], centers[0], centers[1], (0, 255, 255), 2)

cv2.putText(images[i], f'd = {pixelDistance:.2f} px', getCenterofLine(centers[0], centers[1]), cv2.FONT\_HERSHEY\_DUPLEX, 0.5, (255, 0, 255), 1)

cv2.putText(images[i], f'd = {kmDistance:.4f} km', getCenterofLine(centers[0], centers[1], y=-20), cv2.FONT\_HERSHEY\_DUPLEX, 0.5, (255, 0, 255), 1)

# # Menghitung jarak antara centroid pada gambar kedua (dalam piksel dan meter)

if len(centers) > 3:

pixelDistance = cv2.norm(centers[2], centers[4], cv2.NORM\_L2) # # Menghitung jarak Euclidean antara centroid

kmDistance = pixelDistance / 0.24705882352941178 # # Mengkonversi jarak piksel ke meter (1 meter = 0.24 piksel)

# # Menggambar garis antara kedua centroid dan menampilkan jarak dalam piksel

cv2.line(images[i], centers[2], centers[4], (0, 255, 255), 2)

cv2.putText(images[i], f'd = {pixelDistance:.2f} px', getCenterofLine(centers[2], centers[4]), cv2.FONT\_HERSHEY\_DUPLEX, 0.5, (0, 255, 255), 1)

cv2.putText(images[i], f'd = {kmDistance:.4f} m', getCenterofLine(centers[2], centers[4], y=-20), cv2.FONT\_HERSHEY\_DUPLEX, 0.5, (0, 255, 255), 1)

# # Menampilkan gambar yang telah diproses

for i in range(len(images)):

cv2.imshow(f'Gambar {i+1}', images[i]) # # Menampilkan gambar

# # Menunggu hingga tombol ditekan dan menutup jendela

cv2.waitKey(0)

cv2.destroyAllWindows()

### Output Program



# PENUTUP

## Kesimpulan

Teknik yang efektif, dengan fokus pada geometri, pelabelan, dan deteksi bentuk. Proses dimulai dengan konversi gambar ke grayscale dan penerapan thresholding untuk memisahkan objek dari latar belakang. Penghapusan objek kecil dan penutupan morfologis kemudian memperbaiki struktur objek. Langkah selanjutnya melibatkan pengisian lubang dalam kontur dan perhitungan geometri objek, seperti area dan perimeter, untuk menilai bentuknya. Akhirnya, setiap objek diberi label berdasarkan metrik bentuk yang dihitung.

## Opini Penulis

Opini penulis, metode ini sangat berguna dalam analisis citra karena memberikan wawasan mendalam tentang karakteristik objek. Penggunaan parameter geometris seperti area dan perimeter memungkinkan identifikasi bentuk yang lebih akurat. Namun, penulis menyarankan penambahan fleksibilitas pada ambang batas ukuran objek dan opsi untuk menyimpan hasil pemrosesan. Ini akan meningkatkan kemampuan kodingan untuk beradaptasi dengan berbagai jenis gambar, menjadikannya alat yang lebih praktis dalam aplikasi nyata. Secara keseluruhan, kodingan ini adalah representasi yang solid dari teknik-teknik penting dalam pengolahan citra digital.

# DAFTAR PUSTAKA

[1]https://pemrogramanmatlab.com/pengolahan-citra-digital/ekstraksi-ciri-citra-digital/#:~:text=Ekstraksi%20ciri%20citra%20merupakan%20tahapan,lainnya%20pada%20tahapan%20identifikasi%2F%20klasifikasi.

[2]<https://ojs.amikom.ac.id/index.php/semnasteknomedia/article/viewFile/1353/1271?form=MG0AV3>

[3] https://ejournal.gunadarma.ac.id/index.php/infokom/article/download/2368/1874