LAPORAN AKHIR TUGAS AKHIR KULIAH
PENGOLAHAN CITRA DIGITAL
PEMBACAAN PLAT NOMOR

Disusun oleh:

1.	Husnul Latifah	17/409375/PA/17682	2017
2.	Ratih Suci Lestari	20/456556/PA/19743	2020
3.	Datu Maulana Ahmad	20/462077/PA/20049	2020
4.	Khairunas Rhamadani W.	20/462085/PA/20057	2020

UNIVERSITAS GADJAH MADA

YOGYAKARTA

2022

BAB I
PENDAHULUAN

A. Latar Belakang

Kendaraan bermotor sudah menjadi kebutuhan primer bagi seluruh masyarakat dimana dapat ditunjukkan bahwa ada lebih dari 500.000 kendaraan yang masuk di jogja pada 25 April hingga 1 Mei 2022. Hal tersebut juga dirasakan pada Jl. Kaliurang yang selalu padat ketika hari kerja.

Dengan jumlah kendaraan tersebut pasti diperlukan adanya pencatatan data yang sangat banyak. Pada pencatatan ini plat nomor memiliki peran penting sehingga dapat mengetahui jumlah mobil yang masuk dan keluar pada suatu lahan parkir atau tempat perbelanjaan. Selain itu plat nomor dapat membantu polisi untuk mengetahui data dari pemilik kendaraan tersebut.

Dengan jumlah yang banyak pencatatan plat nomor kendaraan sangat diperlukan sehingga dapat memudahkan para pemilik pariwisata untuk mengetahui seberapa banyak kendaraan di dalam lahan parkir. Dengan demikian diperlukan pendeteksi plat nomor menggunakan citra digital. Sehingga plat akan dengan mudah tersimpan dan dapat memudahkan dalam transaksi ketika kendaraan keluar atau masuk lahan parkir.

B. Tujuan Penelitian

Pada penelitian ini didapatkan tujuan penelitian sebagai berikut :

1. Membuat program untuk mendeteksi plat nomor.
2. Menyimpan hasil dari pembacaan plat nomor.

C. Rumusan Masalah

Dari latar belakang yang sudah disampaikan dapat diketahui bahwa ada lebih dari ratusan ribu kendaraan yang ada di Jogja. Dengan jumlah sebanyak ini jika tidak dilakukan pendataan maka plat nomor tidak akan berguna untuk mengetahui pemilik dari kendaraan tersebut, begitu juga pada pemilik pariwisata yang tidak dapat mengetahui plat kendaraan yang ada di lahan parkiran. Dan jika dilakukan pencatatan secara manual akan memakan waktu yang sangat lama.

D. Batasan Masalah

Penelitian ini memiliki batasan sebagai berikut

1. Warna plat nomor tidak sama dengan warna mobil
2. Plat nomor yang memiliki bentuk persegi empat
3. Dalam satu gambar hanya ada satu mobil

E. Manfaat Penelitian

Penelitian ini adalah

1. Membantu polisi dalam pendeteksian plat nomor kendaraan
2. Memvisualisasikan plat nomor yang terdeteksi
3. Dapat di export dan dijadikan sebagai dataset plat nomor kendaraan

BAB II
METODE

A. Pra Pemprosesan

Proses ini diawali dengan proses akuisisi data citra. Data yang kami gunakan adalah *Car License Plate Detection* Dataset yang berasal dari kaggle. Setelah itu citra RGB dikonversi lagi menjadi citra BGR. Proses ini menukar nilai dari *channel* merah dan *channel* biru. Citra yang sudah kembali ke warna aslinya ini kemudian diubah menjadi citra *grayscale.*

Metode *grayscaling* yang dilakukan adalah dengan mengubah nilai ketiga *channel,* menjadi satu *channel*. Caranya adalah dengan mencari nilai *Luminance* relatif yang didapatkan dengan rumus:

$$\text{RGB[A] to Gray:} \quad Y \leftarrow 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B$$

Keterangan:

- Y: Nilai *Luminance* relatif
- R: Nilai dari *channel* merah
- B: Nilai dari *channel* biru
- G: Nilai dari *channel* hijau, nilai pengali hijau paling besar karena mata manusia paling sensitif dengan warna hijau.

Tahapan selanjutnya adalah proses peningkatan kualitas citra dengan filter. Kami menggunakan gaussian *low pass filter* untuk mengurangi noise yang terjadi ketika proses *thresholding*. Kernel yang kami gunakan berukuran 5x5.

B. Segmentasi

Tujuan dari metode otsu adalah membagi histogram citra *grayscale* ke dalam dua daerah yang berbeda secara otomatis tanpa membutuhkan bantuan user untuk memasukkan nilai ambang (*threshold*). Pendekatan yang dilakukan oleh metode otsu adalah dengan melakukan analisis diskriminan yaitu menentukan suatu variabel yang dapat membedakan antara dua atau lebih kelompok yang muncul secara alami. Analisis Diskriminan akan memaksimumkan variable tersebut agar dapat membagi objek latar depan (*foreground*) dan latar belakang (*background*).

Metode Otsu melanjutkan proses peningkatan citra dan memroses gambar yang telah diperhalus dengan *gaussian smoothing*. Hasil dari proses ini adalah citra dengan warna hitam (0,0,0) dan putih (255,255,255).

C. Operasi Morfologi

Pada penelitian kali ini kali melakukan operasi morfologi erosi. Sesuai namanya, proses ini mengikis *foreground* yang telah didapatkan dari metode *thresholding* sesuai dengan ukuran kernel yang telah ditentukan. Kami memilih erosi karena dalam percobaan, banyak tepi plat nomor yang kurang jelas dan menyatu dengan bagian mobil lainnya. Operasi erosi kami lakukan menggunakan kernel yang berukuran 5x5.

D. Ekstraksi Fitur

Ekstraksi Fitur mengacu pada pada menemukan sudut suatu wilayah atau citra yang ingin dikenali dengan objek lainnya. Ciri yang telah diekstrak kemudian digunakan sebagai parameter/nilai masukan untuk membedakan antara objek satu dengan lainnya pada tahapan klasifikasi. Dalam pembacaan plat nomor ini menggunakan pendekatan *Edge-Based* yang berfungsi untuk mendeteksi tepian dari citra dengan metode canny sehingga huruf dan angka dapat terdeteksi.

Pada metode tersebut dilakukan proses deteksi tepi dengan operator gradien. Tepi atau sisi dari sebuah objek tersebut yaitu daerah di mana terdapat perubahan intensitas warna yang cukup tinggi. Proses deteksi tepi akan melakukan konversi terhadap daerah tersebut menjadi dua macam nilai yaitu intensitas warna rendah atau tinggi, contoh bernilai nol atau satu. Metode Canny *edge detection* akan menghasilkan sebuah tampilan gambar yang berbeda dengan menampilkan efek relief di dalamnya. Efek relief terbentuk dari bayangan terang dan gelap. Kelebihan dari metode Canny ini adalah kemampuan untuk mengurangi noise sebelum melakukan perhitungan deteksi tepi sehingga tepi-tepi yang dihasilkan lebih banyak.

Masukannya citra berupa citra biner yang sudah melewati tahap erosi. Setelah itu citra di filter dengan kernel sobel dengan arah vertikal (*Gy*) dan horizontal (*Gx*). Kedua nilai ini digunakan untuk mendapatkan *edge gradient* dan arahnya dengan rumus:

$$Edge\_Gradient\ (G) = \sqrt{G_x^2 + G_y^2}$$

$$Angle\ (\theta) = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$

Setelah didapatkan *edge gradient* dan arahnya dilakukan pemindaian penuh gambar dilakukan untuk menghilangkan piksel yang tidak diinginkan yang mungkin bukan merupakan tepi. Untuk ini, pada setiap piksel, piksel diperiksa apakah maksimum lokal di lingkungannya dalam arah gradien.

Proses selanjutnya adalah *hysteresis thresholding.* Tahap ini memutuskan mana semua sisi yang benar-benar sisi dan mana yang tidak. Untuk ini, kita memerlukan dua nilai ambang batas, minVal dan maxVal. Setiap tepi dengan gradien intensitas lebih dari maxVal pasti menjadi tepi dan yang di bawah minVal pasti bukan tepi, jadi dibuang. Mereka yang berada di antara dua ambang ini diklasifikasikan sebagai tepi atau bukan tepi berdasarkan konektivitasnya. Jika mereka terhubung ke piksel "tepi pasti", mereka dianggap sebagai bagian dari tepi. Jika tidak, mereka juga dibuang. Pada penelitian ini, kami menentukan minVal sebesar 70 dan maxVal sebesar 400.

E. Deteksi

Klasifikasi citra digital merupakan proses pengelompokkan piksel ke dalam kelas-kelas tertentu. Pada penelitian ini, proses klasifikasi yang dilakukan adalah menentukan bagian mana yang merupakan plat nomor. Kami menggunakan metode representasi citra digital kontur yang berfungsi sebagai penghubung antar tepi sehingga dapat membentuk informasi relief. Selanjutnya kontur-kontur yang ada direduksi untuk mencari kontur yang dominan.

Pada tahap ini menggunakan *metode edge counting* dengan perulangan, untuk mencari titik-titik yang disinyalir sebagai plat kendaraan. Kami menentukan jumlah tepi minimal 4 karena bentuk plat nomor yang persegi empat, dengan batasan tepi terbanyak 7 untuk mengantisipasi tepi yang kurang rata. Batasan tepi ini tidak membatasi posisi dan orientasi plat nomor. Sehingga plat nomor berbentuk jajar genjang, dan bentuk persegi empat lainnya masih dapat terdeteksi. Hasil akhir dari proses ini adalah citra yang sudah dipotong sesuai dengan plat nomor yang akan dideteksi.

BAB III
HASIL PENELITIAN DAN PEMBAHASAN

A. Pra pemrosesan

Mengubah citra RGB ke BGR

```
rgb = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
plot_image = image_read(image, rgb, title1 = "RGB", title2 =
"Original")
```

Gambar 1. Hasil konversi citra BGR ke RGB

Mengubah citra RGB ke grayscale

```
gray = cv2.cvtColor(rgb, cv2.COLOR_BGR2GRAY)
image_read(rgb, gray, title1 = "RGB", title2 = "Gray")
```

Gambar 2. Hasil konversi citra RGB ke Grayscale

Proses Smoothing

```
smooth = cv2.GaussianBlur(gray,(5,5),cv2.BORDER_DEFAULT)
image_read(gray, smooth, title1 = "Grayscale", title2 =
"Gaussian Smoothing")
```

Gambar 3. Hasil citra yang diperhalus dengan *gaussian smoothing* (kernel 5x5)

B. Segmentasi

```
 _, thresh =
cv2.threshold(smooth,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)
image_read(smooth, thresh, title1 = "Gaussian Smoothing",
title2 = "Otsu Thresholding")
```

Gambar 4. Hasil segmentasi Otsu

C. Operasi Morfologi

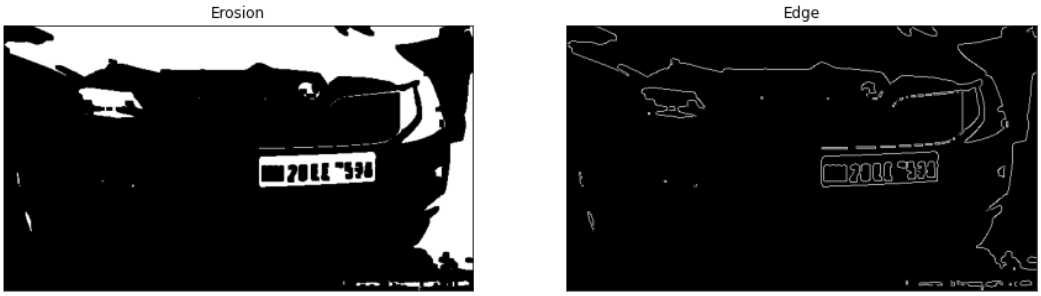
```
kernel = np.ones((5, 5), np.uint8)
erosion = cv2.erode(th_image, kernel,iterations=1)
```

Gambar 5. Hasil operasi erosi (kernel 5x5)

D. Ekstraksi Ciri

Metode Deteksi tepi Canny

```
edge = cv2.Canny(erosion, 70, 400)
image_read(dilation, edge, title1 = "Dilation", title2 = "Edge")
```



Gambar 6. Hasil metode *canny edge detection*

Pembacaan Kontur

```
contours, new = cv2.findContours(edge.copy(), cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)
image_copy = rgb.copy()
_ = cv2.drawContours(image_copy, contours, -1, (255, 0, 0), 2)
image_read(edge, image_copy, title1 = "Edge", title2 = "Contours")
```

Gambar 7. Hasil pembacaan kontur citra

Reduksi Kontur

```
contours = sorted(contours, key = cv2.contourArea, reverse = True)[:4]
image_reduced = edge.copy()
_ = cv2.drawContours(image_reduced, contours, -1, (255, 0, 0), 2)
image_read(image_copy, image_reduced, title1 = "Original", title2 = "Reduced")
```

Gambar 8. Hasil reduksi kontur

E. Deteksi

```
plate = None

for i in contours:
    a = cv2.arcLength(i, True)
    edge_count = cv2.approxPolyDP(i, 0.005 * a, True)
    if len(edge_count) >= 4 and len(edge_count) <= 7:
        x, y, w, h = cv2.boundingRect(i)
        plate = image[y:y+h, x:x+w]
        cv2.imwrite("plate.png", plate)
```

Gambar 9. Hasil akhir pendeteksian plat nomor

BAB IV
KESIMPULAN

Dari hasil diatas dapat disimpulkan metode ini cukup optimal dan cukup mudah untuk diimplementasikan. Hal tersebut dikarenakan metode ini menggunakan metode *grayscale* sebagai *enhancement*, metode Otsu sebagai segmentasi, metode erosi sebagai morfologi, dan metode *canny edge detection* sebagai ekstraksi fitur. Hasil yang didapat menurut kami memuaskan walau masih sedikit sulit diterapkan untuk citra yang pada bagian plat sudah blur dari asalnya.

Metode pendeteksian plat ini juga memiliki kekurangan yaitu tidak dapat mendeteksi plat nomor yang memiliki warna mirip dengan cat kendaraan. Hal tersebut dikarenakan citra akan menyatu dengan kendaraan sejak tahap *enhancement* dan hingga proses-proses selanjutnya tidak akan terdeteksi.

BAB V

LAMPIRAN

A. Listing program

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import cv2
import os
import pathlib
from os import listdir

def load_images_from_folder(folder):
    images = []
    for (idx, filename) in enumerate(os.listdir(folder)):
        img = cv2.imread(os.path.join(folder,filename), 0)
        if img is not None:
            images.append(img)
        rgb = cv2.cvtColor(img, cv2.COLOR_RGB2BGR)
        gray = cv2.cvtColor(rgb, cv2.COLOR_BGR2GRAY)
        smooth =
cv2.GaussianBlur(gray,(5,5),cv2.BORDER_DEFAULT)
        _, thresh =
cv2.threshold(smooth,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)
        kernel = np.ones((5, 5), np.uint8)
        erosion = cv2.erode(thresh, kernel,iterations=1)
        edge = cv2.Canny(erosion, 70, 400)
        contours, new = cv2.findContours(edge.copy(),
cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)
        image_copy = rgb.copy()
        _ = cv2.drawContours(image_copy, contours, -1, (255,
0, 0), 2)
        contours = sorted(contours, key = cv2.contourArea,
reverse = True)[:4]
        image_reduced = edge.copy()
        _ = cv2.drawContours(image_reduced, contours, -1,
(255, 0, 0), 2)
        plate = None
        count = 0
        for i in contours:
```

```
            a = cv2.arcLength(i, True)
            edge_count = cv2.approxPolyDP(i, 0.005 * a, True)
            #print('edge_count: ', edge_count)
            if len(edge_count) >= 4 and len(edge_count) <= 7:
                x, y, w, h = cv2.boundingRect(i)
                plate = rgb[y:y+h, x:x+w]
                cv2.imwrite(f"plate_{idx}.png", plate)
                image_read(plate, rgb, title1 =
f"Plate_{idx}", title2 = f"Original_{idx}")
                break
            count += 1
load_images_from_folder(mobil)
```