Vehicle Number Plate Detection

Introduction

The automatic detection of vehicle number plates has gained significant importance in recent years due to its diverse applications, including toll collection, parking management, and law enforcement. The use of computer vision techniques and machine learning algorithms has made it possible to detect and recognize number plates automatically. In this report, we will discuss the implementation of a vehicle number plate detection system using OpenCV and Tesseract OCR.

Methodology

The system uses a cascade classifier to detect number plates in the frames captured by the camera. The classifier is trained using the Haar-like features, which are used to detect edges, corners, and other significant features of an object. Once the number plate is detected, the system extracts the region of interest (ROI) and converts it to grayscale. The grayscale ROI is then threshold using Otsu's method, which automatically calculates the threshold value. Finally, Tesseract OCR is used to extract the text from the threshold image.

Following are the steps to implement **Vehicle Number Plate Detection:**

1. Imports required libraries - cv2, numpy, and pytesseract.
2. Sets the path for pytesseract.
3. Sets the frame width and height and minimum area required for a license plate to be detected.
4. Loads the classifier for detecting license plates.
5. Opens the default camera and sets the frame width, height, and brightness.
6. Initializes a counter for the saved images.
7. Reads a frame from the camera, converts it to grayscale, and detects license plates in the grayscale image.
8. Loops through each license plate found, calculates the area of the license plate, and if the area is greater than the minimum area, draws a rectangle around the license plate and displays a label.
9. Extracts the region of interest (ROI) containing the license plate, converts the ROI to grayscale, applies thresholding to the grayscale image to extract the text, and uses pytesseract to extract the text from the thresholded image.

10. If the 's' key is pressed, it displays number plate as text on console, saves the ROI as an image file and displays a message indicating that the scan has been saved.
11. If the 'q' key is pressed, exits the program.
12. Releases the capture and destroys all windows.

Below is the implemented code in Python Programming Language:

```python
import cv2
import numpy as np
import pytesseract

pytesseract.pytesseract.tesseract_cmd = "C:/Program Files/Tesseract-
OCR/tesseract.exe"

# Set the frame width and height
FRAME_WIDTH = 640
FRAME_HEIGHT = 480

# Set the minimum area required for a license plate to be detected
MIN_AREA = 500

# Load the classifier for detecting license plates
PLATE_CASCADE = cv2.CascadeClassifier(
    "C:/Users/Desktop/testing/haarcascade_russian_plate_number.xml")

# Open the default camera (id=0)
cap = cv2.VideoCapture(0)

# Set the frame width, height, and brightness
cap.set(3, FRAME_WIDTH)
cap.set(4, FRAME_HEIGHT)
cap.set(10, 150)

# Initialize a counter for the saved images
count = 0

# Loop until the user presses the 'q' key
while True:
    # Read a frame from the camera
    success, frame = cap.read()

    # Convert the frame to grayscale
    frame_gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Detect license plates in the grayscale image
    number_plates = PLATE_CASCADE.detectMultiScale(frame_gray, 1.1, 4)

    # Loop through each license plate found
    for (x, y, w, h) in number_plates:
        # Calculate the area of the license plate
        area = w * h
        # If the area is greater than the minimum area, draw a rectangle around
the license plate and display a label
        if area > MIN_AREA:
            cv2.rectangle(frame, (x, y), (x + w, y + h), (255, 0, 0), 2)
            cv2.putText(frame, "NumberPlate", (x, y - 5),
                        cv2.FONT_HERSHEY_COMPLEX, 1, (0, 0, 255), 2)
            # Extract the region of interest (ROI) containing the license plate
            roi = frame[y:y + h, x:x + w]
            # Convert the ROI to grayscale
            roi_gray = cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY)
            # Apply thresholding to the grayscale image to extract the text
            _, roi_thresh = cv2.threshold(
                roi_gray, 0, 255, cv2.THRESH_BINARY+cv2.THRESH_OTSU)
            # Use pytesseract to extract the text from the thresholded image
            text = pytesseract.image_to_string(roi_thresh, config='--psm 11')

            # Display the text on the license plate
            cv2.putText(frame, text, (x, y - 30),
                        cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)

            cv2.imshow("ROI", roi)

    # Display the result with license plates highlighted
    cv2.imshow("Result", frame)

    # If the 's' key is pressed, save the ROI as an image file
    if cv2.waitKey(1) & 0xFF == ord('s'):
        print(text)
        cv2.imwrite("C:/Users/Desktop/testing/IMAGES" +
                    str(count) + ".jpg", roi)
        # Display a message indicating that the scan has been saved
        cv2.rectangle(frame, (0, 200), (640, 300), (0, 255, 0), cv2.FILLED)
        cv2.putText(frame, "Scan Saved", (15, 265),
                    cv2.FONT_HERSHEY_COMPLEX, 2, (0, 0, 255), 2)
        cv2.imshow("Result", frame)
        cv2.waitKey(500)
        count += 1

    # If the 'q' key is pressed, exit the program
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Release the capture and destroy all windows
cap.release()
cv2.destroyAllWindows()
```

Results

The implemented system was tested on a live camera feed and was able to detect number plates accurately. The system was able to handle different lighting conditions and backgrounds, and was able to detect number plates from different angles. The extracted text was also accurate, although there were some cases where the text was not recognized due to poor image quality or occlusion.

Discussion

The implemented system has some limitations, including the requirement for a high-quality camera and adequate lighting conditions. The system may also have difficulties in detecting number plates in situations where there is significant occlusion or if the number plate is not visible from the camera's perspective. Additionally, the system may be affected by the speed of the vehicle, which could result in motion blur and affect the accuracy of the detection.

Conclusion

The implemented system demonstrated the feasibility of using computer vision techniques and machine learning algorithms for vehicle number plate detection. With further optimization, the system could be used in a variety of applications, including traffic management, parking management, and law enforcement. Overall, the system has shown promising results and could be a valuable addition to the field of computer vision and machine learning.

References

1. "License Plate Recognition using OpenCV in Python" by Sahil Garg, published in the International Journal of Computer Science and Mobile Computing in 2015.
2. "An Improved License Plate Recognition System Based on Edge Detection and Hough Transform" by Wei-Li Chen and Meng-Yen Hsieh, published in the International Journal of Innovative Computing, Information and Control in 2015.
3. "License Plate Recognition for Indian Vehicles using Haar Cascade Classifier and Template Matching" by Pranjal Shukla and Jayshree Bapat, published in the International Journal of Advanced Research in Computer and Communication Engineering in 2014.
4. "Optical Character Recognition using Tesseract OCR Engine: An Overview" by Yugal Kumar and Ankur Singh Bist, published in the International Journal of Computer Science and Mobile Computing in 2015.