Universitatea Națională de Știință și Tehnologie Politehnica din București
Facultatea de Electronică, Telecomunicații și Tehnologia Informației

Detect and Recognize Car License Plate from a video in real time

Proiect Python

Nume student:
Marinescu Silviu-Andrei
Baroiu Silvian
Nistor Flaviu-Cristian
Grupa:
422B

2023

1. Cerința

Proiectul satisface o sarcină importantă în sistemele de securitate bazate pe camere video, mai ales cele aflate prin izolarea și citirii informațiilor de pe o plăcuță de înmatriculare dintr-un videoclip în timp real. Programul folosește tehnici de izolare a plăcuței de restul obiectelor din imagine și prin recunoaștere optică a caracterelor (OCR) poate citi numerele de înmatriculare bazate pe un șablon impus de acea țară.

2. Tehnologii utilizate

Am folosit multiple tehnologii în crearea proiectului, cea principală fiind limbajul Python. În crearea programului am folosit și o tehnologie de recunoaștere optică a caracterelor (OCR), aceasta fiind „Tesseract”.

De asemenea am folosit un simulator auto numit „BeamNG.drive” pentru a avea un mediu din care putem face rost ușor de filmări cu mașini care să satisfacă șablonul pentru care am optimizat programul.

Pentru Python am folosit ca biblioteci „cv2”, de unde am importat un xml pentru „russiancarnumberplate”, care ne-a oferit template-ul pentru numerele de înmatriculare rusești care pot fi recunoscute de OCR, bibliotecile „re” și „deque” pentru a crea o listă și un dicționar și biblioteca “panda” pentru a putea introduce datele de ieșire într-un Excel, pentru a testa mai ușor eficiența programului.

În plus, proiectul prezintă o multitudine de extensii integrate în softul Pycharm care ajută la executarea anumitor funcții necesare împlinirii scopului programului.

Pillow	10.1.0 → 10.2.0
et-xmlfile	1.1.0
imageio	2.33.0 → 2.33.1
imutils	0.5.4
lazy_loader	0.3
networkx	3.2.1
numpy	1.26.2 → 1.26.3
opencv-python	4.8.1.78 → 4.9.0.80
openpyxl	3.1.2 → 3.2.0b1

packaging	23.2
pandas	2.1.4 → 2.2.0rc0
pip	23.3.2
pytesseract	0.3.10
python-dateutil	2.8.2
pytz	2023.3.post1
scikit-image	0.22.0
scipy	1.11.4 → 1.12.0rc1
six	1.16.0
tf	1.0.0
tifffile	2023.12.9

```
import tkinter as tk
from tkinter import filedialog
import cv2
import pytesseract
import re
from collections import deque
import pandas as pd
```

3. Descrierea aplicației

3.1. Interfața

Mai jos este prezentarea codului pentru interfața simplă folosită de program.

```
root = tk.Tk()
root.title("CAR_LICENSE_PLATE_RECOGNITION")

video_file_path = ""
excel_file_path = ""

def run_code():
    if not video_file_path:
        print("Alege fisierul video")
        return
```

Acest cod stabilește titlul ferestrei care se deshide la rularea programului și definește 2 variabile care ajută programatorul să aleagă mai eficient fișierele necesare pentru videoclip-ul selectat, respectiv Excelul în care vor fi trecute datele, fără accesarea interfației.

Comanda „def run_code()” marchează începutul funcției care execută cerința proiectului.

```
def open_video_file():
    global video_file_path
    video_file_path = filedialog.askopenfilename(title="Selecteaza video", filetypes=[("Video files", "*.mp4")])

def open_excel_file():
    global excel_file_path
    excel_file_path = filedialog.askopenfilename(title="Selecteaza fisierul Excel pentru afisare", filetypes=[("Excel files", "*.xlsx")])

label1 = tk.Label(root, text="Universitatea Națională de Știință și Tehnologie Politehnica București", font=("Helvetica", 16), bg="#f2d3a7")
label1.pack(pady=20)

label2 = tk.Label(root, text="PROIECT PYTHON - RECUNOASTEREA NUMERELOR DE INMATRICULARE DIN VIDEO ", font=("Helvetica", 14), bg="#f2d3a7")
label2.pack(pady=10)
label2 = tk.Label(root, text="2024 ", font=("Helvetica", 14), bg="#f2d3a7")
label2.pack(pady=10)
label2 = tk.Label(root, text="PROIECT REALIZAT DE BAROIU SILVIAN, MARINESCU SILVIU-ANDREI, NISTOR FLAVIU-CRISTIAN ", font=("Helvetica", 14), bg="#f2d3a7")
label2.pack(pady=10)
label2 = tk.Label(root, text="  ", font=("Helvetica", 14), bg="#f2d3a7")
label2.pack(pady=10)
label2 = tk.Label(root, text="PASUL 1:", font=("Helvetica", 14), bg="#f2d3a7")
label2.pack(pady=10)
btn_open_video = tk.Button(root, text="Alege fisierul video", command=open_video_file, width=20, height=2)
btn_open_video.pack(pady=10)
label2 = tk.Label(root, text="PASUL 2:", font=("Helvetica", 14), bg="#f2d3a7")
label2.pack(pady=10)

btn_open_excel = tk.Button(root, text="Alege fisierul Excel", command=open_excel_file, width=20, height=2)
btn_open_excel.pack(pady=10)
label2 = tk.Label(root, text="PASUL 3:", font=("Helvetica", 14), bg="#f2d3a7")
label2.pack(pady=10)
```

Acesta este codul complet pentru programarea interfației care prezintă modalitatea alegerii videoclipului și a documentrului Excel prin funcția „filedialog.askopenfilename” din biblioteca „Tkinter”.

Titlul și restul textului prezentate în fereastra deschisă sunt realizate prin funția label, din aceeași librărie.

Cele trei butoane prezente în interfață sunt concepute tot cu ajutorul librăriei „Tkinter” și oferă utilizatorului posibilitatea de a folosi orice videoclip pentru recunoașterea plăcuțelor de înmatriculare și introducerea datelor obținute într-un fișier de tip Excel. Ultimul buton rulează codul.

Aceste funcții de la final configurează dimensiunena ferestrei și culoarea fundalului.

```
btn_run = tk.Button(root, text="START", command=run_code, width=20, height=2)
btn_run.pack(pady=10)

root.configure(bg="#f2d3a7")
root.geometry("1000x650")
btn_run.pack(pady=10)
btn_open_video.pack(pady=10)
btn_open_excel.pack(pady=10)


root.mainloop()
```

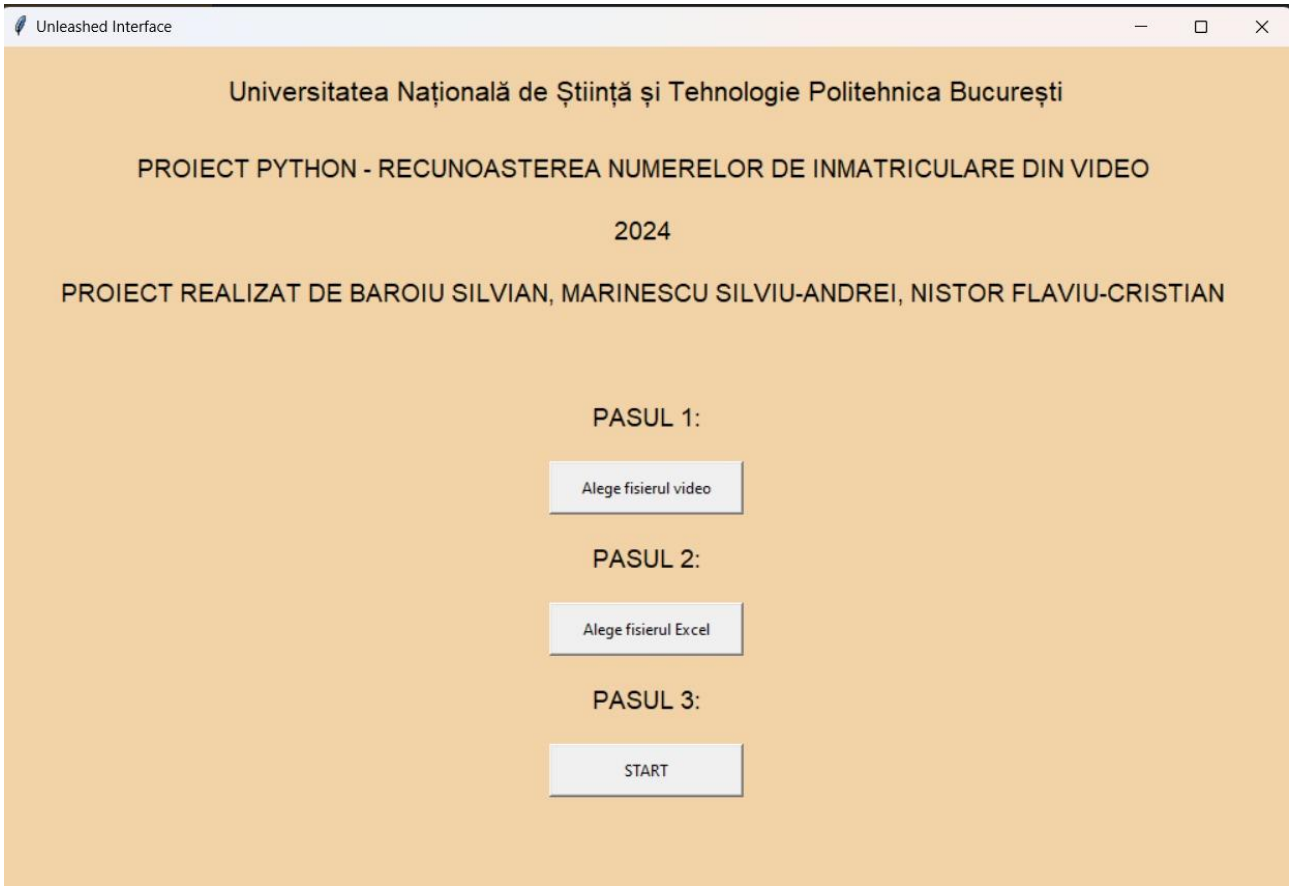

Figura 3.1

Interfața de operare a proiectului

3.2.Arhitectura Sistem

Proiectul se desfășoară pe baza unui singur fișier de tip „main.py” care prezintă codul principal și funcționalitatea programului. Proiectul prezintă și un fișier de tip .xml care prezintă funcționalitatea de „haar-cascade” cărui funționalitate este prezentată la secțiunea următoare.

3.3. Prezentare funcționalități

Prima funție prezentă în program este definită pentru extragerea literelor și cifrelor dintr-un text.

```
# Funcție pentru extragerea caracterelor alfanumerice dintr-un text
def extract_alphanumeric(text):
    return re.sub(r'\W+', '', text)
```

Apoi avem funcția de inițializarea a OCR-ului, care are ca scop recunoașterea unui text din filmări.

```
pytesseract.pytesseract.tesseract_cmd = r'C:\Program Files\Tesseract-OCR\tesseract.exe'
```

Această funție este urmată de un „haarcascade” xml creat pentru a ajuta programul să distingă unde se află o plăcuță de înmatriculare în acel cadru al filmării, pentru a putea recunoaște Tesseract textul plăcuței. Acest haarcascade este optimizat pentru plăcuțe de înmatriculare rusești, deoarece fiecare țară are un format diferit de text pentru plăcuțele lor de înmatriculare, unele având multiple formate pentru o singură țară/regiune (ex. SUA).

Un Haar Cascade este un clasificator obiect pre-antrenat, instruit să identifice caractere distinctive ale unor obiecte, în cazul nostru plăcuțe de înmatriculare. Un Haar Cascade funcționează astfel:

1.Antrenarea clasificatorului prin seturi de imagini:

Prin oferirea de seturi pozitive și negative de date, putem antrena clasificatorul să recunoască doar anumite lucruri.

2.Crearea Etapizată a Clasificatorului

Clasificatorul rezultat este organizat într-o serie de etape (sau cascade), unde fiecare etapă conține un set de clasificatori. Etapele sunt aranjate într-o structură ierarhică și sunt proiectate să reducă numărul de regiuni care trebuie analizate în mod intensiv, concentrându-se pe regiunile mai promițătoare. (Exemplu: prelucrarea imaginii ca să fie alb-negru, reducerea zgomotului din imagine, oferirea unor dimensiuni pentru obiectele căutate, etc.)

```
frame = cv2.resize(frame, (display_width, display_height))
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

vehicles = vehicle_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5, minSize
    =(20, 20))
```

3.Caracteristici Haar:

Caracteristicile Haar sunt ferestre de evaluare rectangulare care sunt plasate pe imagine și sunt utilizate pentru a calcula diferențele de intensitate între sub-regiuni ale imaginii. Aceste caracteristici sunt eficiente în detectarea de modele, cum ar fi marginile sau schimbările bruște de intensitate, care sunt semnificative pentru obiectul de interes.

```
for (x, y, w, h) in vehicles:
    cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)

    center_x = x + w // 2
    center_y = y + h // 2

    rectangle_width = int(w * 0.84)
    rectangle_height = int(h * 0.9)
```

4.Fereastră glisantă (Sliding Window):

O fereastră glisantă este folosită pentru a parcurge imaginea și a aplica clasificatorul în regiuni mici ale imaginii. La fiecare pas al ferestrei glisante, clasificatorul examinează caracteristicile Haar și decide dacă regiunea curentă conține sau nu obiectul de interes.

```
top_left = (center_x - rectangle_width // 2, center_y - rectangle_height // 2)
bottom_right = (center_x + rectangle_width // 2, center_y + rectangle_height // 2)

cv2.rectangle(frame, top_left, bottom_right, (0, 255, 0), 2)

roi = frame[top_left[1]:bottom_right[1], top_left[0]:bottom_right[0]]
gray_roi = cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY)
```

5.Impunerea unui prag (Thresholding):

În funcție de rezultatele clasificatorului, este aplicat un prag (threshold) pentru a decide dacă o regiune este sau nu un obiect. Pentru a evita detecții multiple pentru același obiect, se aplică o tehnică numită non-maximum suppression, unde doar detecțiile cele mai puternice sunt reținute.

```
_, thresh = cv2.threshold(gray_roi, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)
```

6. Integrarea în sistemul de supraveghere:

După ce plăcuța de înmatriculare este detectată și localizată, folosim tehnici suplimentare, precum segmentarea caracterelor și recunoașterea optică a caracterelor (OCR), pentru a extrage numărul de înmatriculare.

```
vehicle_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_russian_plate_number
    .xml')
```

Metodele explicite care au fost apelate în codul nostru se pot găsi în xml-ul luat din biblioteca Cv2.

Următoarea secțiune de cod folosește OCR-ul selectat, Tesseract, pentru a transforma secțiunea de imagine aleasă de Haar Cascade (un cadru din videoclip) în text, verificând dacă respectă formatul căutat pentru numere de înmatriculare rusești.

```
# Verifică dacă conturul conține caractere
number_plate_text = pytesseract.image_to_string(thresh, config='--psm 8')
number_plate_text = extract_alphanumeric(number_plate_text)

# Verifică dacă șirul respectă formatul specific
if len(number_plate_text) == 8 and \
   number_plate_text[0].isalpha() and \
   number_plate_text[1:4].isdigit() and \
   number_plate_text[4:6].isalpha() and \
   number_plate_text[6:].isdigit():
```

Figura 3.2

Model de mașină cu plăcuță de înmatriculare care respectă formatul impus

Figura 3.3

Model de mașină cu plăcuță de înmatriculare care nu respectă formatul impus

În continuare programul detectează dacă numărul de înmatriculare nu este duplicat pentru nu a afișa același număr de înmatriculare de mai multe ori. Deoarece programul primește videoclipuri care le segmentează pe cadre care le analizează, un număr de înmatriculare va apărea în mai multe cadre consecutive și aceste condiții sunt impuse pentru a nu se întâmpla o afișare duplicată.

```
# Verificăm noul șir în raport cu cele detectate recent
is_unique = number_plate_text not in recently_detected_strings

# Adaugă filtrare duplicată
if is_unique:
    recently_detected_strings.append(number_plate_text)

    # Verifică condiția pentru afișarea numărului
    if len(recently_detected_strings) > 1:
        prev_number = recently_detected_strings[-2]
        common_chars = set(number_plate_text).intersection(prev_number)
        if len(common_chars) <= 5:
            counter += 1
            data['Masina'].append(counter)
            data['Numar_inmatriculare'].append(recently_detected_strings[-2])
    else:
        counter += 1
        data['Masina'].append(counter)
        data['Numar_inmatriculare'].append("")
```

Această secventă de cod are de utilitate, pentru a opri programul prematur apăsând tasta „q” de pe tastatură.

```
cv2.imshow('Number Plate Recognition', frame)

if cv2.waitKey(30) & 0xFF == ord('q'):
    break
```

Datele returnate de program sunt stocate aferente într-un document excel pentru vizualizarea lor mai simplă.

```
# Creează un DataFrame din dicționarul de date
df = pd.DataFrame(data)

# Elimină rândurile cu valori nule și salvează DataFrame-ul într-un fișier Excel
df_cleaned = df.dropna(subset=['Numar_inmatriculare'])
df_cleaned.to_excel("C:\\Users\\flavi\\rezultate_recunoastere_numar.xlsx", index=False)
```

4. Bibliografie

https://github.com/opencv/opencv

https://sourceforge.net/projects/tesseract-ocr.mirror/

https://www.geeksforgeeks.org/license-plate-recognition-with-opencv-and-tesseract-ocr/

https://www.youtube.com/watch?v=fyJB1t0o0ms

Arhive extra:

Tesseract-OCR.zip

Cod Python.txt (Codul total)