UNIVERSITATEA DE VEST DIN TIMIȘOARA

FACULTATEA DE MATEMATICA SI INFORMATICA

DEPARTAMENTUL: INFORMATICA APLICATA

PROIECT VA

RECUNOASTERE PLACUTA INMATRICULARE

STUDENT:

BOGAI STEFAN-TUDOR

TIMIȘOARA

2022

1.Introducere

In domeniul Vederii Artificiale (Computer Vision), se incearca diferite proiecte ce au ca scop usurarea activitatii umane prin automatizare. Printre posibilele proiecte eu am ales sa rezolv problema identificarii placutelor de inmatriculare. Ca si utilizare practica, un astfel de proiect poate fi integrat in monitorizarea traficului, urmarirea unei masini, verificarea parcarilor, etc.

Ca si motivatie pentru acest proiect, in anul II de facultate in cadrul altor materii am incercat sa dezvolt o aplicatie care monitorizeaza si automatizeaza parcarile in Sibiu, iar un punct cheie pentru a verifica daca un loc de parcare este ocupat este detectarea unei masini pe spatiul respectiv iar pentru eficientizare si indexare, identificarea placutei de inmatriculare este necesara.

2.Resurse

Proiectul meu de Identificare Automata a Placutelor de Inmatriculare este scris in limbajul Python, folosind libraria OpenCV pentru prelucrarea imaginilor in vederea separarii placutelor de inmatriculare din cadru iar pentru extragerea textului am folosit initial EasyOCR care ulterior din motive de performanta l-am inlocuit cu Python Tesseract. Am folosit de asemenea libraria NumPY pentru a crea un fond negru pe care sa salvam placutele de inmatriculare separate si ulterior sa fie citite.

3.Functionalitate

- In prima faza, aplicatia incarca un fisier video (de tip .mp4 sau .mov) pe care incepe sa prelucreze fiecare al doilea frame (din motive de viteza de procesare).
- Pe frame se aplica un filtru de transformare in scara alb-negru.

```
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
```

- Dupa transformarea in alb-negru, se mai aplica un filtru de “reducere a zgomotului”

```
bfilter = cv2.bilateralFilter(gray,11,17,17)
```

- Apoi, din frame se extrag liniile de contur

```
edged = cv2.Canny(bfilter, 80, 200)
```

- Se prelucreaza imaginea in asa fel incat toate contururile sa fie salvate intr-un set de tuple

```
keypoints = cv2.findContours(edged.copy(), cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
contours = imutils.grab_contours(keypoints)
contours = sorted(contours, key=cv2.contourArea, reverse = True) [:10]
```

- Din setul de tuple se extrage poligonul regulat cu 4 laturi, cel mai aproape de a fi un numar de inmatriculare

```
location = NONE
for contour in contours:
    peri = cv2.arcLength(contour,True)
    approx = cv2.approxPolyDP(contour, 0.09*peri, closed = True)
    if len(approx) == 4:
        location = approx
        break
```

- In cazul in care exista, se va separa din imaginea initiala si va fi extras textul

```
if location!= NONE :
    mask = np.zeros(gray.shape, np.uint8)
    new_image = cv2.drawContours(mask, [location], 0, 255, -1)
    new_image = cv2.bitwise_and(frame,frame,mask=mask)

    (x,y) = np.where(mask==255)
    (x1,y1) = (np.min(x), np.min(y))
    (x2,y2) = (np.max(x), np.max(y))
    cropped_image = gray[x1:x2+1, y1:y2+1]

    text = pytesseract.image_to_string(cropped_image, config='--psm 11 ')
```

- Ulterior, textul extras va fi adaugat pe cadrul initial si salvat separat drept fisier “.jpg”

```
if text != '' :
        font = cv2.FONT_HERSHEY_SIMPLEX
        res = cv2.putText(frame, text=text, org=(approx[0][0][0], approx[1][0][1]+60),
        res = cv2.rectangle(frame, tuple(approx[0][0]), tuple(approx[2][0]), (0,255,0), 3)
        cv2.imwrite(os.path.join('frames','frame' + str(frame_idx) + '.jpg'), res)
```

4.Cazuri netratate

In prezent, actuala versiune de aplicatie nu ia in considerare multiple cazuri particulare care incurca citirea si determinarea numarului de inmatriculare. Printre ele sunt, cazul in care am mai multe placute de inmatriculare, cazul in care am mai multe forme similare precum cea a unei placute, cazul in care imaginea este blurata, etc.