Prelucrarea informațiilor video preluate de la o cameră WEB

-identificarea poziției obiectelor aflate în mișcare-

Man Cristina

Universitatea Tehnică din Cluj-Napoca

15/10/2023

Table of Contents

1. Introducere

1.1 Context

Scopul acestui proiect este de a identifica obiecte aflate în mișcare prin intermediul unei camere WEB sau prin alegerea unui videoclip preîncărcat în cadrul proiectului. Aplicația pune la îndemâna utilizatorului un meniu în care acesta poate alege tipul de videoclip pe care să se realizeze recunoașterea mișcării și mărimea obiectelor observate.

1.2 Motivație

Aceasta aplicație poate fi folosită în numeroase cazuri, cum ar fi numărarea oamenilor aflați în mișcare într-o încăpere, numărarea oamenilor care trec strada într-un interval de timp, alertarea utilizatorului în cazul în care are loc o mișcare în spațiul supravegheat etc. Există și numeroase posibilități de dezvoltare ulterioară, cum ar fi integrarea funcționalității într-o aplicație Web care sa permită oricărei persoane interesate de o anumită locație să vizualizeze obiectele aflate în mișcare din acel loc. Totodată, acest proiect permite integrarea unei logici pentru determinarea direcției și sensului de mișcare a obiectelor din scena observată sau reținerea traiectoriei obiectelor și, prin agregarea datelor, tragerea unor concluzii (cea mai folosită bandă de circulație, locul în care ar fi potrivit să fie construită o potecă etc.)

1.3 Obiective

Obiectivele urmărite în acest proiect sunt realizarea, prelucrarea și testarea informațiilor provenite de la camera WEB integrată a calculatorului. Aceste informații trebuie prelucrate prin intermediul librăriei *OpenCV* pentru a detecta mișcarea obiectelor aflate în cadru și încadrarea acestora într-un chenar pentru o recunoaștere mai ușoară a mișcării acestora.

2. Studiu bibliografic

Detectarea mișcării reprezintă un domeniu vast și bine documentat care oferă doritorilor numeroase posibilități de explorare. Tehnologia folosită în cadrul proiectului este *OpenCV* (Open Source Computer Vision Library), o librărie software de computer vision și machine learning care oferă utilizatorilor mai mult de 2500 de algoritmi care pot fi folosiți pentru prelucrarea informațiilor provenite de la o cameră WEB sau dintr-un videoclip. Acești algoritmi pot fi folosiți pentru

recunoașterea fețelor umane, urmărirea mișcării ochiului, identificarea obiectelor și urmărirea obiectelor aflate în mișcare.

OpenCV are o interfață pentru limbajul de programare Python, suportând, de asemenea, sistemele de operare Windows, Linux, Android si Mac OS.

3. Design & Analiză

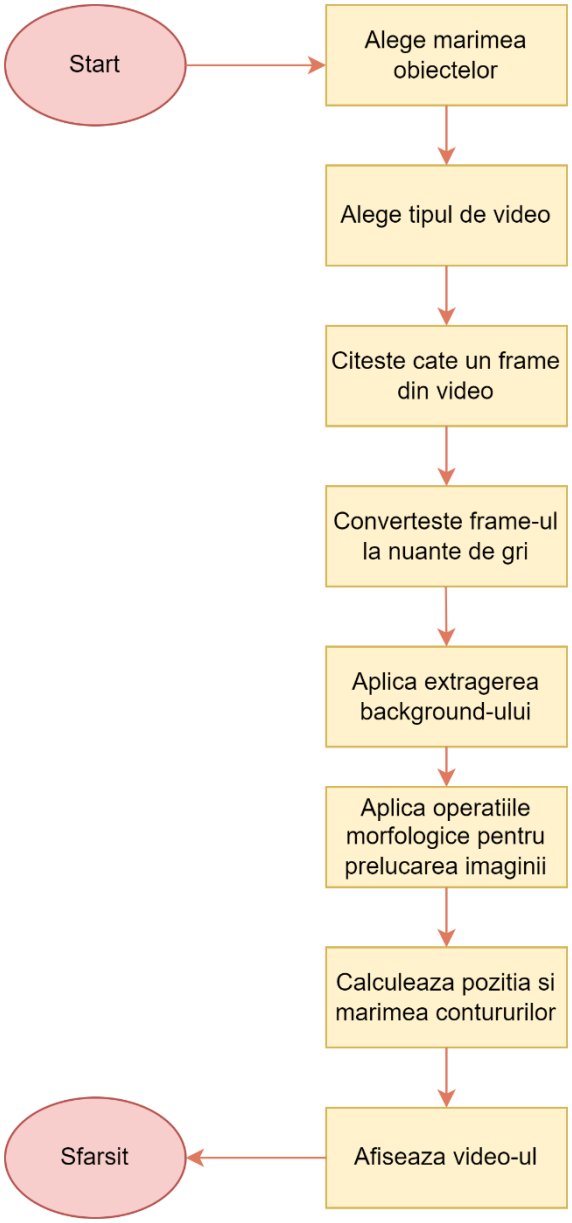
3.1 Analiză

Analizând algoritmul de detectare a mișcării folosit în acest proiect, se poate observa nevoia de a importa pachetul *cv2*, o extensie a librăriei *OpenCV* pentru limbajul de programare Python.

Pentru că adaptarea aplicației la diferențe mari între mărimile obiectelor poate îngreuna întreg procesul de recunoaștere a mișcării, este nevoie de stabilirea încă de la început a mărimii obiectelor pe care utilizatorul dorește a le observa.

Odată stabilite mărimea obiectelor și tipul de videoclip dorite, este nevoie de aplicarea algoritmului de recunoaștere a mișcării. Algoritmul constă în capturarea imaginii curente, transformarea acesteia într-o imagine la scară gri și încețoșarea celei din urma, ca mai apoi să se aplice extragerea fundalului și aplicarea operațiilor morfologice.

Operațiile morfologice sunt simple operații bazate pe forma unei imagini binare sau la scara gri. O operație are nevoie de doua intrări, anume imaginea originală și elementul de structurare (kernel), cel din urma determinând natura operației. Cele



Figură 1 – Diagrama de stări pentru algoritmul de funcționare a aplicației

mai simple și folosite operații sunt eroziunea și dilatarea, acestea fiind folosite și în proiectul prezentat. Eroziunea constă în parcurgerea imaginii bit cu bit cu ajutorul unui kernel. Un pixel din imaginea originală (formată din ‚1' si ‚0') va fi considerat ‚1' doar daca toți pixelii de sub kernel sunt ‚1'; în caz contrar, acesta va fi considerat ‚0'. Ceea ce rezultă este o imagine a cărei obiecte sunt mai puțin groase. Dilatarea, fiind operația opusa eroziunii, va parcurge imaginea bit cu bit, iar un bit va fi considerat ‚1' daca cel puțin un bit aflat sub kernel este ‚1'. Rezultatul dilatării este un obiect mai gros. Ordinea normală a celor două operații este eroziunea urmată de dilatare, fapt determinat de caracteristica eroziunii de a înlătura zgomotele albe și de a perfecționa forma obiectului observat. Odată ce zgomotele au fost înlăturate, asupra obiectului se poate efectua dilatarea cu scopul de a-l aduce la forma inițială.

Odată ce au fost efectuate toate operațiile morfologice, se poate calcula dimensiunea și poziția fiecărui contur caracteristic unui obiect aflat in mișcare.

3.2 Design

Atunci când aplicația se deschide, aceasta afișează un meniu în cadrul căruia îi sunt prezentate utilizatorului următoarele opțiuni: dimensiunea obiectelor pe care dorește să le observe si tipul de video în care apar aceste obiecte. Dimensiunea poate fi mare (6000 de unități) sau mica (600 de unități), iar video-ul poate fi reprezentat de camera Web sau de un video preîncărcat. Aceste opțiuni sunt afișate sub forma unor butoane, iar, odată alese, utilizatorul trebuie să apese butonul de start. In cazul în care butonul de start este apăsat fără a alege nicio opțiune, acestea vor avea valorile lor predefinite, anume obiecte mari și camera Web.

Rolul existentei opțiunii pentru recunoașterea mișcării obiectelor mici este pentru a exemplifica și prezenta funcționarea, respectiv acuratețea algoritmului pe un set de date bine definite.

Frame-urile provenite din videoclip trebuie citite și procesate într-o buclă infinită, până când utilizatorul apasă tasta „q" (*quit*) pentru a închide aplicația sau videoclipul ales a ajuns la sfârșit. Pentru a oferi posibilitatea vizualizării modului în care funcționează algoritmul, utilizatorul poate apăsa tasta „g" pentru a vedea cum arata imaginea după convertirea acesteia la scara gri și începțoșare, respectiv „f" pentru a vedea imaginea obținută după aplicarea operațiilor morfologice. Apăsarea tastei ‚a' permite vizualizarea cadrului din videoclip de la momentul respectiv atât în forma sa finala, incluzând conturul aferent obiectului, cat si în forma rezultată din aplicarea operațiilor morfologice.

4. Implementare

Proiectul prezentat a fost realizat folosind limbajul de programare *Python* și biblioteca *cv2*. Implementarea meniului care apare după lansarea aplicației s-a bazat pe folosirea bibliotecii *tkinter*. Cu ajutorul acestei biblioteci, s-a realizat fereastra și au fost așezate textele și butoanele pentru a permite utilizatorului să aleagă dimensiunea obiectelor și tipul de videoclip. S-a ales o implementare simplă a interfeței grafice, aceasta constând in doua porțiuni de text si patru butoane, fiecare buton reprezentând o preferința pentru dimensiunea obiectelor observate, respectiv videoclipul care se dorește a fi analizat. Butoanele au atașată, de asemenea, câte o comandă prin intermediul căreia sunt setate variabilele globale aferente preferințelor.

```
Label(text="Choose the size of the objects", width=25).grid(row=0, column=0)
Label(text="Choose the type of video", width=25).grid(row=1, column=0)

button1 = Button(menu, text="Big", command=big_threshold)
button2 = Button(menu, text="Small", command=small_threshold)
button3 = Button(menu, text="Web camera", command=web_cam)
button4 = Button(menu, text="Uploaded video", command=uploaded_video)
button5 = Button(menu, text="Start", command=start)
```

Figură 2 - Etichetele si butoanele interfeței grafice

Procesarea fiecărui cadru din videoclip are loc într-o buclă infinită *while*, ieșindu-se din aceasta doar atunci când videoclipul preîncărcat a ajuns la sfârșit sau atunci când utilizatorul apasă tasta „q”.

Prelucrarea imaginii cu scopul de a ajunge la forma obiectului aflat în mișcare are loc prin convertirea la scară gri, încețoșarea acesteia și aplicarea operațiilor morfologice în felul următor: prin aplicarea funcției *cvtColor(frame, cv2.COLOR_BGR2GRAY)* din cadrul bibliotecii *cv2* are loc convertirea cadrului curent la scara gri, iar prin intermediul funcției *GaussianBlur(gray_frame, (15, 15), 0)*, tot din biblioteca *cv2*, se realizează încețoșarea cadrului obținut anterior. Parametrii funcției *GaussianBlur* sunt: gray_frame (cadrul pe care se aplica transformarea), tupla (15, 15) (dimensiunea matricei care parcurge cadrul, aceasta determinând gradul de încețoșare) si 0 (deviația standard in direcțiile X si Y; în cazul de față, este

specificata doar deviația in direcția X, așadar, deviația în direcția Y se considera egala cu omoloaga sa).

Înainte de intrarea în bucla infinita cu scopul de a procesa constant videoclipul dorit, are loc crearea unui obiect de extragere a fundalului de tipul *MOG2* prin expresia *mog = cv2.createBackgroundSubtractorMOG2()*. Funcția *mog.apply(gray_frame)* este folosita apoi pentru fiecare cadru cu scopul extragerii fundalului din cadrul deja convertit la scara gri și încețoșat.

```
gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)  # Convert the frame to grayscale
gray_frame = cv2.GaussianBlur(gray_frame, ksize: (15, 15), sigmaX: 0)

fg_mask = mog.apply(gray_frame)  # Apply background subtraction

# Apply morphological operations to reduce noise and fill gaps
kernel = cv2.getStructuringElement(cv2.MORPH_RECT, ksize: (4, 4))
fg_mask = cv2.erode(fg_mask, kernel, iterations=3)
fg_mask = cv2.dilate(fg_mask, kernel, iterations=5)
```

Figură 3 - Prelucrarea fiecărui cadru

S-a ajuns la concluzia că, în cazul camerei Web folosite pentru testarea acestui proiect, aplicarea eroziunii de trei ori urmată de aplicarea dilatării de cinci ori va duce la o detectare a obiectelor aflate în mișcare optimă, minimizând zgomotele care pot apărea în cadrul videoclipului. Totodată, dimensiunea matricei de kernel aduce rezultatul dorit dacă aceasta are o dimensiune de 4x4 și forma de patrulater (fapt determinat de constanta *cv.MORPH_RECT*). Astfel, matricea de kernel cu care se va parcurge imaginea va arata in felul următor:

```
array([[0, 0, 1, 0, 0],
       [1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1],
       [0, 0, 1, 0, 0]], dtype=uint8)
```

Figură 4 - Forma matricei de kernel

Calcularea conturului fiecărui obiect aflat în mișcare se realizează cu ajutorul funcției cv2.findContours(), lista obținută in urma aplicării acestei funcții pe întreaga

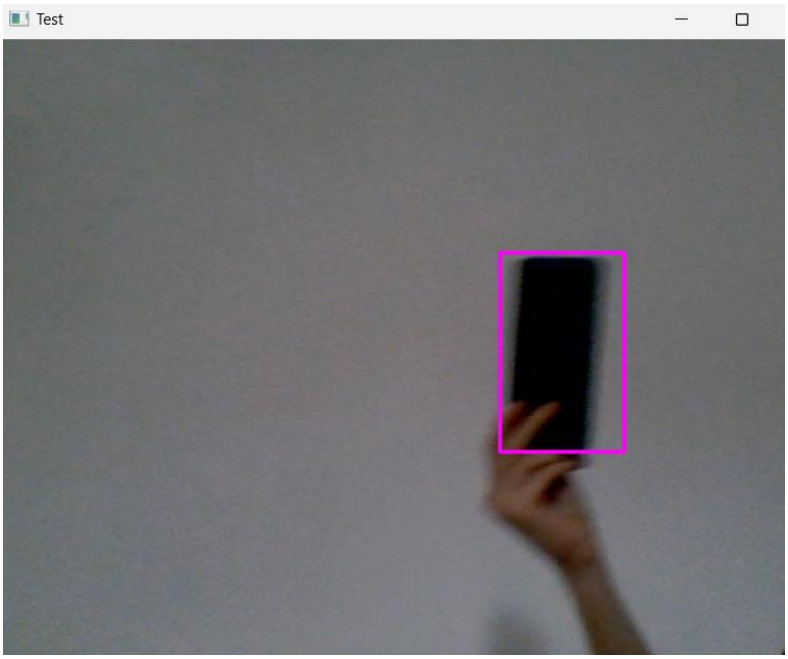
imagine fiind apoi parcursă cu scopul de a calcula mărimea patrulaterului care va cuprinde obiectele.

```
contours, _ = cv2.findContours(fg_mask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

for contour in contours:

    if cv2.contourArea(contour) < contour_threshold:  # Ignore small contours
        continue

    x, y, w, h = cv2.boundingRect(contour)  # Draw bounding box around contour
    cv2.rectangle(frame, (x, y), (x + w, y + h), (255, 0, 255), 2)
```

Figură 5 - Calcularea conturului obiectelor

5. Testare

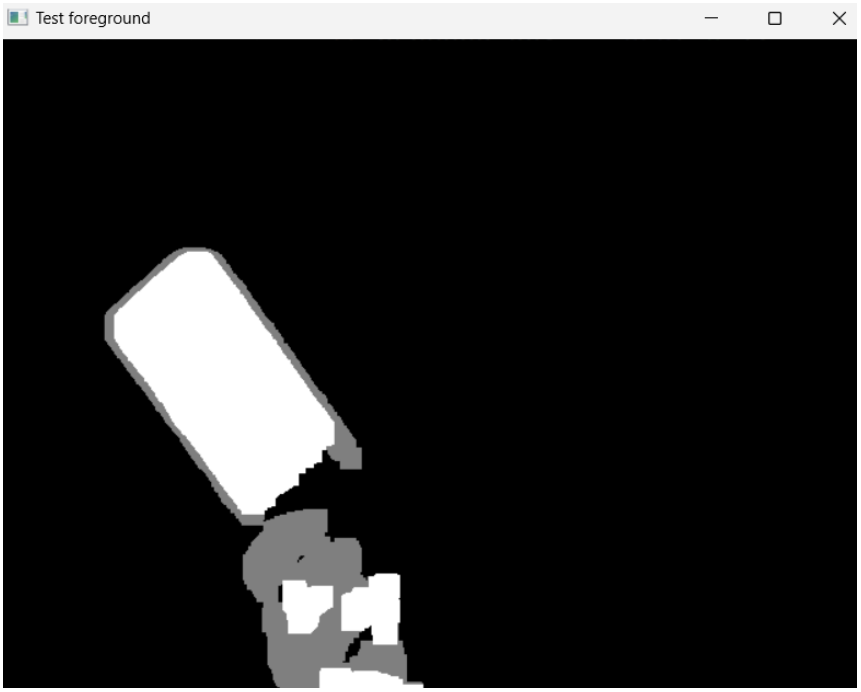
Pentru a testa funcționalitatea aplicației, am folosit două scenarii, anume folosirea camerei Web, respectiv a unui videoclip preîncărcat pentru a detecta mișcarea. Totodată, după pornirea programul, am apăsat tasta ‚a' cu scopul de a obține o imagine cu cadrul din videoclip de la momentul dorit.



Figură 6 - Încadrarea obiectului aflat in mișcare observat de camera Web

Am așezat camera în fața unui fundal alb și am folosit un obiect dreptunghiular negru pentru a ușura observarea obiectelor aflate în mișcare pe camera Web. După cum se ilustrează figura 6, obiectul este observat cu ușurință și încadrat ca atare.

Pentru a vedea forma finală a cadrului curent în urma aplicării operațiilor morfologice, apăsarea tastei ‚a' duce la apariția unei imagini cu cadrul curent după aplicarea operațiilor morfologice, așadar, înaintea calculării dimensiunii și poziției conturului.



Figură 7 - Imaginea observata de camera Web in urma operațiilor morfologice

In ceea ce privește analiza unui videoclip preîncărcat, am ales unul în care se pot observa oameni aflați în mișcare în cadrul unui spațiu aglomerat, iar rezultatele la care s-a ajuns în urma aplicării algoritmului se regăsesc în figura 7. Se poate observa că viteza de deplasare a unui obiect este direct proporțională cu mărimea conturului care îl încadrează.

Figură 5 - Obiecte aflate in mișcare in cadrul unui video preîncărcat

6. Concluzie

Scopul acestui proiect a fost proiectarea, implementarea și testarea unei aplicații de detectare a obiectelor aflate în mișcare prin intermediul unei camere Web sau a unui videoclip preîncărcat. Am adăugat, de asemenea, posibilitatea de a alege dimensiunea obiectelor pe care utilizatorul dorește a le observa pentru obținerea unor rezultate mulate pe nevoile acestuia.

Ceea ce am învățat pe parcursul dezvoltării acestei aplicatii a fost modul de utilizare si numeroasele transformări care se pot aplica unor imagini cu ajutorul bibliotecii *cv2*. Totodată, am apreciat oportunitatea de a lucra in limbajul de dezvoltare *Python*, deoarece am avut posibilitatea de a experimenta atât cu utilizarea bibliotecilor disponibile, cat si cu realizarea interfețelor grafice cu scopul de a oferi utilizatorului o experiență cat mai directa.

Totodată, întregul proces de documentare, urmat de design, analiză și, intr-un sfârșit, implementare și testare a reprezentat o șansă de a acumula cunoștințe noi legate de procesarea imaginilor, dar și de a consolida unele deja existente, precum folosirea limbajului *Python.*

7. Bibliografie

[1] *„Image Processing in OpenCV”*,
https://docs.opencv.org/4.x/d2/d96/tutorial_py_table_of_contents_imgproc.html

[2] *Motion Detection OpenCV Python*,

https://hackthedeveloper.com/motion-detection-opencv-python/