Urmarirea obiectelor in timp real

**Cerinte**

Se doreste implementarea unei aplicatii care primeste ca input un fisier video sau un flux video (camera), pe care il afiseaza si ii permite utilizatorului sa selecteze un obiect, care va fi urmarit pe toata durata fluxului. Fluxul va fi 2D si se va putea urmari orice tip de obiecte, ci nu doar un set predefinit.

**Modulele principale**

Pentru implementarea acestei aplicatii, am construit o arhitectura care este compusa din 2 elemente:

1. Supervisor – este modulul care citeste imaginile dintr-un flux video, manipuleaza inputul primit de la utilizator, se ocupa de selectarea manuala a obiectelor si creearea trackerelor. Este intr-o relatie de 1 to many fata de modulul Mosse, adica un supervisor poate avea mai multe trackere (module Mosse), cu care face schimb de informatii.

2. Mosse – este modulul care se ocupa de trackingul obiectelor. Fiecare modul se ocupa de un singur obiect, este creat odata cu selectarea obiectului de catre utilizator si moare odata cu supervisorul. Un modul Mosse primeste continuu ca input frame-ul curent, calculeaza pozitia obiectului in acesta si expune o functie care incadreaza (deseneaza) obiectul intr-un dreptunghi in imagine.

**Implementarea**

Pentru implementarea acestui proiect, am folosit indicatiile lui *David Bolme*, din paper-ul *Visual Object Tracking using Adaptive Correlation Filters[1]*.

Am implementat algoritmul MOSSE (Minimum Output Sum of Squared Error), care utilizeaza ca principiu de baza filtrul de corelare si se antreneaza primind un singur frame. Ideea de baza este identificarea obiectului prin utilizarea filtrelor de corelare si utilizarea unei etape asemanatoare celei de convolutii din retelele neuronale convolutionale (prin care se cauta un sablon in imagine prin suprapunerea continua a acestuia si inmultirii valorilor pixelilor, obtinandu-se la final o matrice a imaginii de dimensiuni putin mai reduse dar din care se poate spune unde se afla sablonul in imagine, prin gasirea valorii maxime in matricea rezultat). Acest algoritm este capabil sa detecteze obiectul, chiar si in momentele in care este rotit, scalat, partial acoperit sau este intr-un mediu cu lumina puternica sau slaba, intr-un timp scurt si reusind astfel sa tina pasul cu framerate-ul fluxurilor actuale de imagini.

*Acuratete*

Pentru masurarea acuratetii localizarii obiectului in imagine, am utilizat ca metrica PSR (Peak-to-Sidelobe Ratio), care masoara varful de corelatie si care ne indica cu cata precizie a fost identificat obiectul (o valoare < 8, indica faptul ca obiectul este acoperit sau nu a putut fi identificat. Valorile normale ar trebui sa se situeze intre 20 si 60, dar este perfect daca sunt > 60).

PSR se calculeaza astfel:

Gmax = valoarea maxima din matricea de corelare

reprezinta media si deviatia standard a valorilor matricii din care se exclude punctul in care se gaseste valoarea maxima Gmax si punctele dintr-o matrice de 11x11 din jurul lui Gmax.

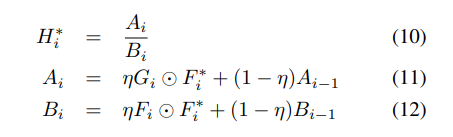
Noi am afisat PSR pentru fiecare tracker, in jurul dreptunghiului care incadreaza obiectul.

*Algoritm*

Se selecteaza un obiect dintr-un frame si se creeaza tracker-ul care primeste initial frame-ul in care s-a facut selectia si punctele care identifica selectia. Pentru a crea un tracker mai eficient, corelatia se va efectua in domeniul Fourier. Pentru a aplica Fourier, am calculat mai intai dimensiunea optima a ferestrei care incadreaza obiectul (cele mai bune performante se obtin daca dimensiunile sunt multiplii de 2, 3 sau 5, de aceea este posibil ca selectia utilizatorului sa fie putin mai mare pentru a indeplinii aceste cerinte) si am aplicat un filtru Gaussian de blur peste frame-ul curent, pentru a reduce zgomotul si detaliile. Dupa aplicarea filtrului, selectez portiunea din imagine care reprezinta obiectul.

Dupa ce am primul frame si obiectul blurate, trec printr-o etapa intermediara de preprocesare, care aplica functia log peste toate valorile pixelilor si normalizeaza valorile acestora intre 0 si 1, pentru a tine cont de cazul in care imaginea are o lumina cu contrast redus.

In continuare, pentru fiecare frame, se calculeaza pozitia obiectului tinand cont de pozitia lui in frame-urile precedente (de aici apare si problema cand obiectul se misca foarte repede), se trece prin etapa de preprocesare prin care se logaritmeaza si normalizeaza valorile pixelilor, se face corelatia si se calculeaza valoarea PSR si diferenta pozitiei curente a obiectului fata de cea anterioara (se obtine prin aflarea pozitiei a pixelului de valoare maxima din matricea de corelatie Gi). Daca valoarea PSR-ului calculat anterior este acceptabila (in caz contrar, asteptam pana cand gasim un frame cu o valoare buna a PSR-ului), atunci se va actualiza pozitia obiectului, se va trece din nou prin etapa de preprocesare si se vor aplica recursiv la nivel de frame-uri, formulele urmatoare:



Hi = filtrul care reprezinta obiectul (\* = conjugat)

Fi = frame-ul curent

Gi = Fi produs convolutional cu Hi\*

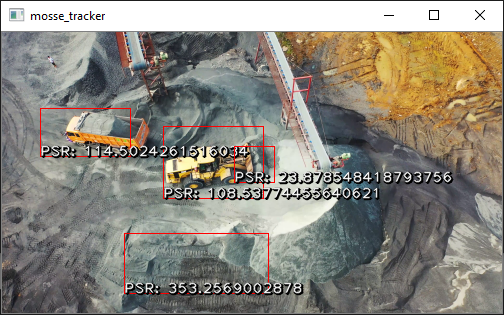
n = rata de invatare si este folosit pentru a pune mai mult accent pe frame-ul curent, scazand exponential importanta frame-urilor mai vechi. Noi am lucrat cu n=0.125

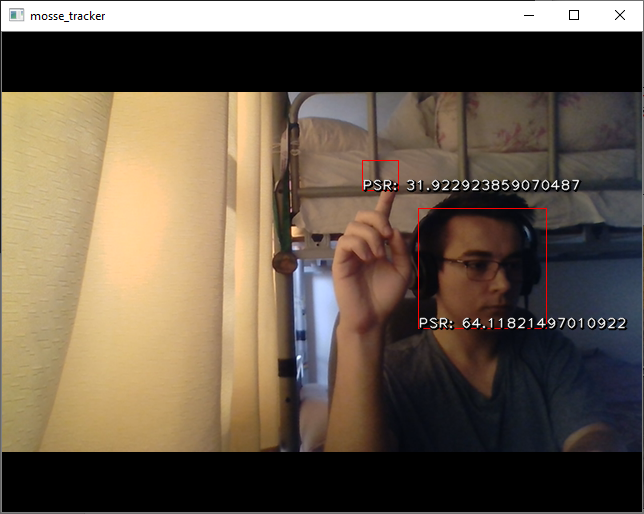
Aceste formule reprezinta o imbunatatire a gasita de autorul paper-ului si pe care le-am utilizat si noi in proiect.

*Alte detalii legate de implementare*

Am utilizat NumPy si OpenCV pentru ca calcula transformatele fourier, produsele de convolutie si alte calcule matematice, precum si pentru a citi fluxurile de imagini si selecta obiectele.

*Rezultate*





Bibliografie

[1] <https://www.cs.colostate.edu/~draper/papers/bolme_cvpr10.pdf>