**Descriere Generală**

Acest proiect realizează procesarea unei imagini utilizând tehnici de multi-threading și transmite actualizări live către o pagină web folosind Flask și WebSockets. Imaginea este supusă urmatoarelor transformări: conversie la gri, inversare pe orizontală și estompare (blur).

**Tehnologii Utilizate**

* **Flask**: folosit pentru a crea un server web care afișează imaginea procesată
* **Flask-SocketIO**: permite trimiterea actualizărilor imaginii către interfața web
* **OpenCV (cv2)**: utilizat pentru manipularea imaginilor (citire, conversie, blur, salvare)
* **NumPy**: folosit pentru manipularea matricelor de pixeli
* **Threading**: permite execuția paralelă a diferitelor etape de procesare

**Funcții Implementate**

**1. gray\_out(pixel\_list, width, height, lock, events)**

**Descriere:**

* Converteste imaginea la tonuri de gri, linie cu linie.
* Pentru conversie folosește formula: gray\_value = 0.21 \* R + 0.72 \* G + 0.07 \* B.
* După procesarea fiecărei linii, semnalează evenimentul events[i] finalizarea prelucrarii unei linii pentru sincronizare cu următorul fir de execuție.

**2. flip\_image(pixel\_list, width, height, lock, events, flip\_events)**

**Descriere:**

* Asteaptă ca fiecare linie să fie convertită la gri (events[i].wait()).
* Inversează pe orizontală fiecare linie a imaginii.
* Semnalează evenimentul flip\_events[i] pentru a permite aplicarea blurului.

**3. blur\_image(pixel\_list, width, height, lock, flip\_events)**

**Descriere:**

* Asteaptă ca fiecare linie să fie inversată (flip\_events[i].wait()).
* Aplică un filtru Gaussian Blur utilizând cv2.GaussianBlur(). Acesta pastrează forma generală a imaginii fara să rotunjească colțurile . ( Alte optiuni MedianBlur, mai rapid ).
* Trimite actualizări către interfața web utilizând send\_image\_update().

**4. send\_image\_update(pixel\_list, width, height)**

**Descriere:**

* Convertește matricea de pixeli într-un format de imagine utilizabil (cv2.imencode()).
* Transformă imaginea într-un string Base64 pentru a fi transmis prin WebSockets.
* Utilizează socketio.emit() pentru a trimite imaginea actualizată paginii web.

**5. main()**

**Descriere:**

* Încarcă imaginea inițială (cv2.imread()).
* Inițializează o listă de pixeli din imagine.
* Creează și pornește cele trei fire de execuție pentru procesare.
* Așteaptă finalizarea tuturor firelor de execuție (‘join’).
* Salvează imaginea procesată utilizând cv2.imwrite().

**Cum Funcționează Threading-ul**

* **Thread 1** (grayscale): procesează fiecare linie și semnalează că este gata o linie de procesat.
* **Thread 2** (flip): așteaptă semnalizarea din Thread 1, inversează linia și semnalează catre Thread 3 că poate prelucra rândul respectiv.
* **Thread 3** (blur): așteaptă semnalizarea din Thread 2, aplică blur și trimite actualizări paginii web.

**Observatii**

* **Imaginea:** Programul este structurat pentru gestiunea formatului JPG, este compatibil și cu PNG cu anumite incoveniente legate preponderent de performanță.
* **Host:** Programul rulează pe localhost pe portul 5000, în funcție de preferințe se poate edita adăugand linia de cod :

**socketio.run(app, debug=True, host='127.0.0.1', port=”xyz”)**

* **Pentru o mai buna vizualizarea a proceselor din cadrul programului,** se poate muta funcția: **Send\_image\_update(pixel\_list**) la finalul funcției oricărui Thread

.