**PROGRAM DE ANALIZA A TRASATURILOR MORFOLOGICE ALE FETEI**

Universitatea Tehnica din Cluj-Napoca

Studenta**:** Hirean Roxana-Maria

**Continut**

1.Introducere................................................................................................................................................1

1.1 Context................................................................................................................................................1

1.2 Specificatii…………................................................................................................................................1

1.3 Obiective.............................................................................................................................................2

2.Studiu bibliografic.....................................................................................................................................2 3.Analiza.......................................................................................................................................................3 4.Design .......................................................................................................................................................5 5.Implementare….........................................................................................................................................7 6.Testare…………………….................................................................................................................................8 7.Concluzii…..................................................................................................................................................9

**Introducere**

* **Context**

Scopul acestui proiect este de a proiecta, implementa si testa detectarea fetei persoanelor din anumite imagini, pentru a extrage trasaturi faciale. Exista o serie de lucruri pe care le putem face cu aceste informatii ca pas de preprocesare, precum etichetarea oamenilor in fotografii, crearea unor efecte sau filtre care sa “imbunatateasca” fotografiile. Un alt exemplu ar mai fi analizarea sentimentelor si emotiilor de pe fete in functie de expresia faciala.

* **Specificatii**

Programul va fi proiectat si implementat pentru recunoasterea fetei in imagini, folosind:

-OpenCV

-DLib

-Python

Fața noastră are mai multe caracteristici care pot fi identificate. Ne vom folosi de librariile cv2 si face\_recognition pentru a detecta fetele care apar intr-o fotografie, intr-un video si chiar si in timp real. Programul este capabil sa detecteze o fata sau mai multe, specificand numarul lor si pozitia(top, bottom, left, right) in timp real (pozitia se va modifica odata cu miscarea fetei persoanei respective). Programul detecteaza fetele chiar si la o distanta mai mare si chiar daca fata este blurata. Fetele detectate vor fi incadrate intr-un chenar rosu.

* **Obiective**

Dintre obiectivele proiectului face parte implementarea unui algoritm de detectare a fetei cu ajutorul functiilor esentiale OpenCV, a librariei DLib in Python, dar si a librariei face\_recognition, care a facut implementarea putin mai usoara. Fetele se detecteaza dintr-o imagine, video sau direct din camera video a laptopului. De asemenea, inca un obiectiv al proiectului este detectarea emotiilor de pe fetele persoanelor in functie de expresia faciala.

**Studiu bibliografic**

Pentru implementarea functiilor specific programului am ales biblioteca OpenCV din Python si biblioteca DLib.

**OpenCV** sau Open Computer Vision este o bibliotecă open-source disponibilă pe [www.opencv.org](http://www.opencv.org/). Bibilioteca este scrisa in limbajul de programare C și C++ și poate fi rulată sub următoarele sisteme de operare: Linux, Windows si Mac OS X. Biblioteca este în permanentă dezvoltare pentru inferfețe precum Python, Ruby, Matlab sau alte limbaje de programare.

La începuturile OpenCV, scopurile proiectului erau descrise astfel:

* Cercetarea avansată a vederii prin asigurarea nu doar de cod sursă deschis ci și optimizat pentru infrastructura de bază a vederii.
* Promovarea cunoștințelor despre vedere prin asigurarea unei infrastructuri comune pentru programatori, astfel ca codul sursă să fie mai inteligibil și transferabil.
* Avansarea de aplicații comerciale bazate pe vedere prin furnizarea gratuită de cod sursă portabil și optimizat – cu o licență care nu solicită ca codul în sine să fie deschis sau gratuit.

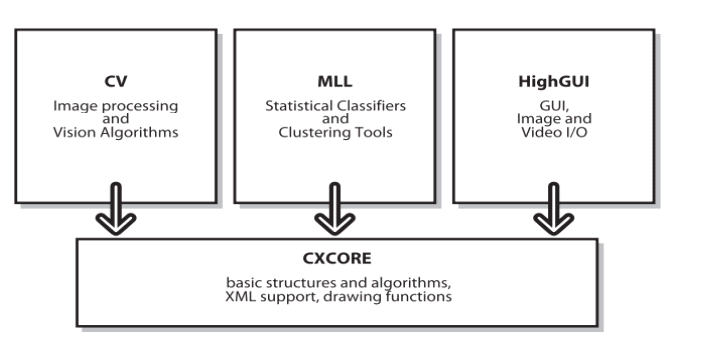
OpenCV a fost proiectat pentru performanță computațională redusă și cu un accent deosebit pe aplicațiile în timp real. OpenCV este scris în C optimizat și poate profita de procesoarele multicore. Unul dintre obiectivele OpenCV este acela de a oferi o platformă software de vedere artificială (machine vision) care ajută oamenii să creeze rapid aplicații de viziune destul de sofisticate. Biblioteca OpenCV conține peste 500 de funcții care acoperă multe zone ale vederii artificiale, inclusiv inspecția vizuală, imagistică medicală, securitate, interfața cu utilizatorul, calibrarea camerei.

Deoarece computer vision și machine learning nu sunt automatizate, OpenCV conține, de asemenea, o bibliotecă de machine learning (MLL), cu scop general.

Drept definiție ideea de computer vision este procesul de tranformare a datelor dintr-o cameră foto sau o cameră video într-o decizie sau o nouă reprezentare. Aceste transformări sunt efectuate pentru a obținute un anumit rezultat dorit.

Componentele OpenCV conțin procesarea de bază a imaginii și algoritmi de vedere artificială la nivel superior. Biblioteca ML (machine learning) include clasificatoare statistice și instrumente de grupare a datelor. HighGUI conține rutine I/O și funcții pentru stocarea și încărcarea imaginilor, iar CXCore conține structurile de date și conținutul de bază.

Figura de mai jos nu include componenta CvAux, care conține recunoașterea chipului încorporat HMM (Hidden Markov model), cât și algoritmi experimentali (segmentare fundal/prim-plan).



**Dlib** este o colectie de aplicatii open source și biblioteci scrise în C++ sub o licență Boost permisivă. Dlib oferă o gamă largă de instrumente de procesare a imaginilor. Este un detector facial al unui reper cu modele pre-trained si este folosit pentru a estima locația a 68 de coordonate (x, y) care mapeaza punctele faciale de pe fața unei persoane, ca imaginea de mai jos.

**Design**

Pentru inceput, voi prezenta putin clasele pe care pe care le-am implementat cu in aplicatia Anaconda3 in Spyder(Python 3.9).

Face detection:

Face detection este un proces care localizeaza automat fetele umane in mediul vizual(imagini sau video).

Doua tipuri populare de suporturi de librarii Face Detectors image\_recognition sunt:

-HoG Face Detector

-CNN based Face Detector inclus in DLib

HoG Face Detector este in general folosit pentru detectia obiectelor si este mai rapid pe CPU si se blocheaza mai greu.

CNN Face Detector din DLib este o metoda care foloseste Object Detector with features CNN. CNN detecteaza orientarea mai multor fete si e rapid pe GPU, dar e super slow pe CPU.

Prima clasa implementata de mine a fost image\_face\_detection.py, prin intermediul careia se detecteaza orice fata dintr-o imagine incarcata de mine. Dupa apasarea butonului de run, programul detecteaza toate fetele din imagine, le decupeaza si afiseaza pe ecran intreaga poza, dar si fetele decupate. In momentul rularii, in consola se afiseaza numarul fetelor detectate si pozitiile acestora(top, bottom, right, left) in functie de unde au fost gasite.

Urmatoarea clasa implementata este realtime\_face\_detection.py, cu ajutorul careia se detecteaza automat o fata sau mai multe. Acestea se detecteaza cu ajutorul camerei laptopului, care se porneste odata cu apasarea butonului de run. Aplicatia va detecta orice numar de fete, nu conteaza distanta la care se afla de camera(daca e mai mare sau mai mica). De asemenea, se afiseaza din nou in consola numarul de fete detectat si pozitia fiecareia. Daca fata se misca, automat in consola se actualizeaza in timp real si pozitiile top, bottom, left si right. Atunci cand apasam tasta 'q', camera se inchide.

Cea de-a treia clasa implementata de mine este video\_face\_detection.py. Aceasta clasa ne ajuta sa detectam fete dintr-un video incarcat de mine, si ca si la celelalte clase implementate, pozitia se modifica in timp real odata cu persoana din video. La fel, atunci cand apasam tasta 'q', video-ul se opreste.

Urmatoarea clasa implementata este realtime\_face\_detection\_blur.py , care extrage fata din frame, o blureaza si apoi o pune inapoi in frame. Are aceleasi caracteristici ca si clasa realtime\_face\_detection. La apasarea tastei 'q', camera se inchide.

Urmatoarea clasa este realtime\_face\_emotion\_detection.py, unde am un label in care imi setez cateva emotii precum bucurie, furie, tristete, frica, dezgust, surprins. In functie de emotia citita pe fata, programul stabileste emotia de pe fata fiecarui om.

Pentru aceasta clasa va trebui sa avem Dataset-ul "Kaggle Facial Expression Recognition Challenge", care consta din imagini cu fete in tonuri de gri de 48x48 pixeli. Fiecare imagine corespunde uneia din 7 categorii de expresii. Intregul set de date contine aproximativ 36000 de imagini. Dataseturile sunt model structure file si model weight file. Aplicatia porneste si deschide camera web, detecteaza emotiile de pe fata in timp real, in functie de schimbarile expresiei fetei. Camera se opreste la apasarea tastei 'q'.

Mai departe, am implementat clasa image\_face\_emotion\_detection.py, care cu ajutorul label-ului care stocheaza emotiile prestabilite, detecteaza emotiile de pe fetele oamenilor dintr-o imagine incarcata de mine. Si in cadrul acestei clase folosim datasetul "Kaggle Facial Expression Recognition Challenge".

Clasa urmatoare implementata este video\_face\_emotion\_detection.py, care detecteaza in timp real emotiile de pe fata persoanelor dintr-un video incarcat de catre mine. Este folosit, de asemenea, label-ul care stocheaza emotiile. Emotiile de pe fata persoanei se schimba in functie de schimbarile expresiei fetei, iar atunci cand apasam tasta 'q', programul se opreste.

**Implementare**

Prima clasa implementata este **image\_face\_detection.py**.

Acest cod folosește bibliotecile cv2 și face\_recognition pentru a detecta și afișa locațiile tuturor fețelor dintr-o imagine.

Fișierul imagine este citit folosind funcția cv2.imread() și stocat în variabila image\_to\_detect. Funcția cv2.imshow() este apoi folosită pentru a afișa imaginea.

Apoi, funcția face\_recognition.face\_locations() este utilizată pentru a detecta toate fețele din imagine. Parametrul model poate fi setat fie la „cnn”, fie la „hog”, iar parametrul number\_of\_times\_to\_unsample poate fi folosit pentru a crește numărul de ori când imaginea este redimensionată pentru a detecta mai multe fețe. Locațiile tuturor fețelor detectate sunt stocate în variabila all\_face\_locations ca o listă de tuple.

Codul tipărește apoi numărul de fețe detectate și parcurge lista de locații ale fețelor. Pentru fiecare locație, codul imprimă coordonatele locației și feliază imaginea pentru a afișa doar fața din acea locație folosind funcția cv2.imshow().

Cea de-a doua clasa implementata este **realtime\_face\_detection.py**.

Acest cod folosește bibliotecile cv2 și face\_recognition pentru a detecta și afișa locațiile tuturor fețelor dintr-un flux video de la o cameră web.

În primul rând, codul folosește funcția cv2.VideoCapture() pentru a inițializa fluxul webcam și a-l stoca în variabila webcam\_video\_stream. O listă goală este apoi inițializată pentru a stoca locațiile feței pentru fiecare cadru al videoclipului.

Codul intră apoi într-o buclă while pentru a procesa fiecare cadru al videoclipului. Funcția cv2.VideoCapture.read() este utilizată pentru a citi în cadrul curent al videoclipului ca o imagine, care este stocată în variabila current\_frame. Cadrul este apoi redimensionat la un sfert din dimensiunea sa originală folosind funcția cv2.resize() și stocat în variabila current\_frame\_small. Acest lucru se face pentru a accelera procesul de detectare a feței.

Funcția face\_recognition.face\_locations() este apoi utilizată pentru a detecta toate fețele din cadrul curent. Parametrul number\_of\_times\_to\_upsample poate fi folosit pentru a mări numărul de redimensionări ale imaginii pentru a detecta mai multe fețe, iar parametrul model poate fi setat fie la „cnn”, fie la „hog”. Locațiile tuturor fețelor detectate sunt stocate în lista all\_face\_locations.

Codul trece apoi prin lista de locații ale fețelor și desenează un dreptunghi în jurul fiecărei fețe folosind funcția cv2.rectangle(). Cadrul redimensionat este apoi afișat folosind funcția cv2.imshow().

Bucla while este setată să ruleze până când utilizatorul apasă tasta „q” de pe tastatură, moment în care codul iese din buclă și eliberează resursele webcam folosind funcția cv2.VideoCapture.release(). Codul închide apoi toate ferestrele deschise folosind funcția cv2.destroyAllWindows().

Cea de-a treia clasa implementata este **video\_face\_detection.py**.

Acest cod este similar cu codul anterior, principala diferență fiind că citește un fișier video dintr-o cale de fișier, mai degrabă decât să folosească o cameră web ca sursă video.

Codul folosește funcția cv2.VideoCapture() pentru a inițializa fluxul video și a-l stoca în variabila webcam\_video\_stream. Calea fișierului către fișierul video este transmisă ca argument funcției. O listă goală este apoi inițializată pentru a stoca locațiile feței pentru fiecare cadru al videoclipului.

Codul intră apoi într-o buclă while pentru a procesa fiecare cadru al videoclipului. Funcția cv2.VideoCapture.read() este utilizată pentru a citi în cadrul curent al videoclipului ca o imagine, care este stocată în variabila current\_frame. Cadrul este apoi redimensionat la un sfert din dimensiunea sa originală folosind funcția cv2.resize() și stocat în variabila current\_frame\_small. Acest lucru se face pentru a accelera procesul de detectare a feței.

Funcția face\_recognition.face\_locations() este apoi utilizată pentru a detecta toate fețele din cadrul curent. Parametrul number\_of\_times\_to\_upsample poate fi folosit pentru a mări numărul de redimensionări ale imaginii pentru a detecta mai multe fețe, iar parametrul model poate fi setat fie la „cnn”, fie la „hog”. Locațiile tuturor fețelor detectate sunt stocate în lista all\_face\_locations.

Codul trece apoi prin lista de locații ale fețelor și desenează un dreptunghi în jurul fiecărei fețe folosind funcția cv2.rectangle(). Cadrul redimensionat este apoi afișat folosind funcția cv2.imshow().

Bucla while este setată să ruleze până când utilizatorul apasă tasta „q” de pe tastatură, moment în care codul iese din buclă și eliberează resursele video folosind funcția cv2.VideoCapture.release(). Codul închide apoi toate ferestrele deschise folosind funcția cv2.destroyAllWindows().

Urmatoarea clasa implementata este **realtime\_face\_detection\_blur.py.**

Acest cod este similar cu cel implementat in clasa realtime\_face\_detection.py, diferenta fiind faptul ca fața este extrasă din cadrul original cu tăiere, estompată cu cv2.GaussianBlur() și apoi lipită înapoi în cadrul original. Un dreptunghi este desenat în jurul locației feței cu cv2.rectangle(). Cadrul modificat este afișat cu cv2.imshow().

Urmatoarea clasa implementata este **image\_face\_emotion\_detection.py.**

Acest cod realizează recunoașterea expresiei faciale pe videoclipuri live de la o cameră web folosind un model pre-antrenat. Folosește biblioteca face\_recognition pentru a detecta locațiile tuturor fețelor din fiecare cadru al videoclipului și apoi folosește biblioteca cv2 pentru a desena dreptunghiuri în jurul fețelor detectate. Apoi folosește biblioteca Keras pentru a preprocesa imaginile feței detectate și pentru a face predicții despre expresiile lor folosind modelul pre-antrenat. În cele din urmă, afișează videoclipul cu fețele detectate și expresiile prezise ale acestora.

Bibliotecile necesare, cv2, numpy, face\_recognition și keras, sunt importate. Videoclipul implicit al camerei web este capturat cu cv2.VideoCapture().

Modelul de recunoaștere a expresiei faciale este inițializat cu model\_from\_json() și greutățile sale sunt încărcate cu load\_weights(). O listă de etichete pentru cele 7 expresii posibile este definită ca emotions\_label.

Este inițializată o matrice goală pentru locațiile feței.

O buclă while infinită începe să circule prin fiecare cadru al videoclipului. Un singur cadru al videoclipului este capturat ca imagine cu webcam\_video\_stream.read(). Cadrul este redimensionat la un sfert din dimensiunea sa originală pentru a accelera procesarea cu cv2.resize(). Locațiile tuturor fețelor din cadru sunt detectate cu face\_recognition.face\_locations().

O buclă for începe să circule prin locațiile feței detectate. Valorile de poziție pentru locația curentă a feței sunt extrase din tuplu și înmulțite cu 4 pentru a se potrivi cu dimensiunea inițială a cadrului. Un dreptunghi este desenat în jurul locației feței cu cv2.rectangle().

Imaginea feței este convertită în tonuri de gri, redimensionată la 48x48 pixeli și apoi convertită într-o matrice numpy 3D cu img\_to\_array(). Forma matricei este extinsă cu np.expand\_dims(). Valorile pixelilor sunt normalizate la intervalul [0, 1] cu img\_pixels /= 255. Modelul este folosit pentru a face o predicție despre expresia feței cu face\_exp\_model.predict(). Indicele celei mai mari valori de predicție este găsit cu np.argmax() și folosit pentru a căuta eticheta corespunzătoare în emotions\_label.

Label-ul este afișat pe imagine.

Urmatoarea clasa implementata este **video\_face\_emotion\_detection.py.**

Acest cod folosește bibliotecile OpenCV și face\_recognition pentru a detecta și eticheta emoțiile fețelor dintr-un flux video. Face acest lucru folosind un model Keras pre-antrenat pentru detectarea emoțiilor, pe care îl încarcă din arhitectura și greutățile modelului salvate.

Codul începe prin configurarea fluxului video folosind OpenCV, cu camera web implicită ca sursă. Apoi încarcă modelul de detectare a emoțiilor pre-antrenat și etichetele acestuia.

Începe apoi bucla principală de procesare. În fiecare iterație, codul:

1.Citește un cadru din fluxul video

2.Redimensionează cadrul la un sfert din dimensiunea inițială pentru a face procesarea mai rapidă

3.Detectează toate fețele din cadru folosind funcția face\_locations din biblioteca face\_recognition

4.Parcurge locațiile feței detectate

5.Pentru fiecare față, aceasta:

1.Desenează un dreptunghi în jurul feței de pe cadru

2.Preprocesează imaginea feței transformându-o în tonuri de gri și redimensionând-o la dimensiunea de intrare a modelului

3.Trece imaginea procesată prin modelul de detectare a emoțiilor pentru a obține o predicție pentru fiecare dintre cele 7 etichete de emoție

4.Alege eticheta cu cea mai mare probabilitate estimată și o afișează pe cadru

6.Afișează cadrul modificat cu emoțiile etichetate

Bucla se repetă apoi până la sfârșitul fluxului video.

Ultima clasa implementata este **interface.py**. Aceasta reprezinta o interfata care este folosita pentru a alege ce file vreau sa rulez, in functie de tasta apasata.

Acest cod este un program condus de meniu care permite utilizatorului să selecteze dintr-o listă de opțiuni pentru a rula diferite programe de detectare a feței și de detectare a emoțiilor. Opțiunile sunt:

1. Detectează fețele într-o singură imagine

2. Detectează fețele în fluxul video în timp real

3. Detectează fețele într-un fișier video

4. Detectează fețele în fluxul video în timp real și estompează-le

5. Detectează emoțiile într-o singură imagine

6. Detectează emoțiile în fluxul video în timp real

7. Detectează emoțiile într-un fișier video

Pentru fiecare opțiune, scriptul Python corespunzător este apelat folosind funcția subprocess.run().

Codul importă mai întâi modulele necesare, inclusiv cv2, face\_recognition, numpy, keras și subprocess. Apoi definește o funcție run\_file() care preia o intrare cheie de la utilizator și rulează programul corespunzător de detectare a feței sau de detectare a emoțiilor pe baza valorii cheii.

Funcția verifică mai întâi valoarea cheii și rulează programul corespunzător. De exemplu, dacă cheia este „1”, citește un fișier imagine și detectează toate fețele din acesta folosind funcția face\_recognition.face\_locations(). Apoi parcurge locațiile feței detectate și afișează fiecare față într-o fereastră separată. Dacă cheia este „5”, rulează scriptul „image\_face\_emotion\_detection.py”, care detectează emoțiile fețelor într-o singură imagine.

În cele din urmă, codul solicită utilizatorului să apese o tastă și transmite valoarea cheii la funcția run\_file() pentru a rula programul selectat.

**Testare**

Proiectul acesta este implementat si testat in Spyder Anaconda. Spyder, Scientific Python Development Environment, este un mediu de dezvoltare integrat (IDE) care este inclus cu Anaconda. Include funcții de editare, testare interactivă, depanare și introspecție.

In cadrul testarii proiectului, am folosit o imagine din computerul personal, pe care am incarcat-o in file-urile image\_face\_detection.py si image\_face\_emotion\_detection.py. Acea imagine contine un numar de 10 fete pe care programul nostru le detecteaza si totodata ne spune pozitia lor( top, bottom, right, left).

O alta modalitate de testare a proiectului este incarcarea unui video tot din computer. Video-ul este incarcat in clasele video\_face\_detection.py si video\_face\_emotion\_detection.py. Programul nostru detecteaza fata din video-ul incarcat, chiar daca fata se misca si de asemenea, ii scrie coordonatele fetei in timp real, in timp ce aceasta se misca. La fel, coordonate top, bottm, left si right.

A treia modalitate de testare este prin intermediul camerei web a laptopului. Aceasta se deschide la rularea programului, in clasele realtime\_face\_detection.py, realtime\_face\_detection\_blur.py si realtime\_face\_emotion\_detection.py. In momentul deschiderii camerei web, programul detecteaza fetele aparute in cadru si daca acestea se afla la apropiere si la distanta. Programul afiseaza in consola coordonatele fetei( top, bottom, left si right) in functie de miscarile fetei(capului) persoanei/persoanelor. Programul nu mai detecteaza fetele daca acestea sunt acoperite sau ascunse din cadru. Acesta este capabil sa detecteze oricate fete.

**Concluzii**

Acest proiect este util in multe domenii de activitate, cele mai multe tinand de tehnologie.Un exemplu ar fi detectarea detinatorului unui telefon prin detectarea fetei lui cu ajutorul acestui program.

Cateva dezvoltari ulterioare ale proiectului ar fi implementarea unui algoritm de detectare a genului sau varstei unei persoane prin intermediul camerei web, care, la fel, ar putea fi de folos in mai multe domenii din zilele noastre. O alta idee ar fi un algoritm care este capabil sa vizualizeze reperele de pe fata.

Recunoașterea faciala este utilizata si la eliberarea documentelor de identitate si, cel mai adesea, combinata cu alte tehnologii biometrice precum amprentele digitale (prevenirea fraudei de identitate și a furtului de identitate). Potrivirea fetei este utilizata la controalele la frontiera pentru a compara portretul de pe un pașaport biometric digitizat cu chipul titularului.