



**UNIVERSITATEA DIN
BUCUREȘTI**



**FACULTATEA DE
MATEMATICĂ ȘI INFORMATICĂ**

**SPECIALIZAREA CALCULATOARE ȘI
TEHNOLOGIA INFORMAȚIEI**

Proiect de diplomă

APLICAȚIE WEB PENTRU ANIMAȚIA 2D A IMAGINILOR ȘI CLIPURILOR VIDEO

Absolvent

Iancu Florentina-Mihaela

Coordonator științific

Lect.dr. Petru Soviany

București, iunie 2024

Rezumat

Scopul acestei lucrări este dezvoltarea și prezentarea unei aplicații web care combină tehnologia și arta, oferind o modalitate simplă de a experimenta cu animația 2D. Motivația din spatele alegerii acestui subiect este interesul în potențialul inteligenței artificiale de a deveni un instrument util pentru artiști în procesul lor creativ, precum și pasiunea personală atât pentru vederea artificială, cât și pentru artă. Lucrarea se bazează pe un algoritm de detecție a pozițiilor corpului uman și pe diverse metode de manipulare a imaginilor pentru a genera animații simple. Acest proces este furnizat sub forma unei aplicații web care oferă utilizatorului mai multe modalități de a transmite datele necesare și permite personalizarea anumitor elemente. Rezultatele pot reprezenta un punct de interes pentru dezvoltatorii care doresc să experimenteze cu aceste două medii în viitor, dar și pentru artiștii care doresc să se folosească de asistență digitală. Folosind limbajul Python, modelul de detecție MediaPipe și biblioteci specializate, am creat o aplicație ușor de folosit pentru a transforma videoclipuri sau imagini în animații 2D, având opțiunea de a genera aceste animații în timp real. Pentru dezvoltarea acestui proiect am utilizat IDE-ul PyCharm de la JetBrains, care a facilitat integrarea diverselor biblioteci și organizarea finală a fișierelor.

Abstract

The purpose of this paper is to develop and present a web application that combines technology and art, offering a simple way to experiment with 2D animation. The motivation behind choosing this topic is the interest in the potential of artificial intelligence to become a useful tool for artists in their creative process, as well as my personal passion for both computer vision and art. The project utilizes a human body position detection algorithm and various image manipulation methods to generate simple animations. This process is provided in the form of a web application that offers the user multiple ways to input necessary data and allows customization of certain elements. The results can be of interest to developers who wish to experiment with these two mediums in the future, as well as for artists who want to utilize digital assistance. Using the Python language, the MediaPipe detection model, and specialized libraries, I created a user-friendly application for transforming videos or images into 2D animations, with the option to generate these animations in real-time. For the development of this project, we used the PyCharm IDE from JetBrains, which aided in integrating various libraries and organizing the final files.

Cuprins

1	Introducere	5
1.1	Prezentarea generală a temei	5
1.2	Scopul și motivația alegerii temei.....	6
1.3	Contribuția proprie în realizarea lucrării	7
1.4	Structura lucrării.....	8
2	Noțiuni științifice	10
2.1	Inteligența artificială	10
2.1.1	Învățarea automată	11
2.1.2	Vederea artificială.....	12
2.2	Dezvoltarea web folosind Python	12
3	Tehnologii utilizate	15
3.1	Limbaje de programare și biblioteci	15
3.1.1	Python	15
3.1.2	Bibliotecile Python	16
3.1.3	HTML/CSS.....	17
3.1.4	JavaScript.....	18
3.2	Mediul de lucru: PyCharm.....	19
3.3	Baza de date	19
3.4	Modelul de detecție: MediaPipe	21
4	Comparație între modele de detecție	24
4.1	MediaPipe vs OpenPose.....	24
4.2	MediaPipe vs YOLOv7	25
4.3	Diverse configurări.....	28

5	Prezentarea aplicației web	31
5.1	Pagina principală.....	31
5.2	Pagina de încărcare fișiere	33
5.3	Pagina de înregistrare video	35
5.4	Pagina de animație în timp real	36
5.5	Pagina de procesare	37
5.6	Pagina de descărcare a rezultatului.....	38
6	Prezentarea algoritmului de animație 2D	39
6.1	Prezentarea generală	39
6.2	Funcțiile <code>anime_frame()</code> și <code>animate_video()</code>	40
6.3	Funcția de procesare a animației 2D.....	42
6.3.1	Adăugarea capului	43
6.3.2	Adăugarea trunchiului.....	45
6.3.3	Adăugarea încălțămintei	46
6.3.4	Adăugarea membrelor.....	47
6.4	Funcțiile ajutătoare	48
7	Concluzii	50
	Bibliografie	52

Capitolul 1

Introducere

1.1 Prezentarea generală a proiectului

Lucrarea de licență este alcătuită din două componente distincte: aplicația web, care reprezintă o interfață user-friendly pentru diverse opțiuni și metode de generare a animațiilor, și programul de procesare a imaginilor sau clipurilor video, care prezintă o abordare nouă asupra legăturii dintre tehnologie și abilități artistice.

Aplicația nu necesită autentificare sau acces special pentru ca utilizatorul să poată beneficia în totalitate de funcționalitățile acesteia, ci este simplu de folosit și accesibilă oricărei categorii de vârstă sau experiență. Astfel, există un singur tip de interfață pentru toți utilizatorii care permite procesarea unei imagini sau a unui videoclip pentru a obține o animație 2D simplă, dar unică a persoanei aflată în cadru.

Dezvoltarea a avut ca obiectiv crearea unei aplicații care convertește datele de intrare într-o imagine sau o animație, ambele folosind același model 2D. Scopul a fost obținerea unui program eficient și a unei interfețe web care să complementeze procesul din fundal. Pentru a crea un mediu în care utilizatorului îi este permis să experimenteze cu diverse elemente, aplicația web a fost centrată pe ideea de personalizare și abilitatea de a crea animații unice pentru a inspira curiozitate utilizatorilor.

Aplicația web oferă trei modalități de încărcare a datelor de input, printre care se numără: încărcarea unui fișier, opțiunea de animație în timp real și înregistrarea unui videoclip folosind camera web pentru a fi procesat ulterior. Două dintre funcționalități, animația în timp real și înregistrarea video, se folosesc de camera web a utilizatorului pentru a prelua datele necesare într-o perioadă delimitată de utilizator prin folosirea unor butoane. Între aceste două metode există câteva diferențe de execuție care vor fi explicate într-un capitol viitor.

Pe lângă necesitatea de a utiliza videoclipuri sau imagini pentru a obține rezultatul final, utilizatorul are libertatea de a alege un model 2D și o imagine de fundal din cele puse la

dispoziție. De altfel, utilizatorii au opțiunea de a încărca o imagine de fundal pentru a personaliza și mai mult rezultatul.

În ceea ce privește aspectul interfeței web, această are un design simplu și ușor de utilizat cu butoane mari și descrieri clare. Paginile web nu sunt aglomerate cu diverse butoane sau meniuri dezorganizate, ci toate opțiunile sunt separate în categorii clar delimitate. Pentru a acomoda modificările făcute asupra ferestrei aplicației, bara de navigare care apare pe toate paginile se restrânge automat într-un meniu dropdown dacă nu mai există destul spațiu pentru toate opțiunile.

1.2 Scopul și motivația alegerii temei

Scopul aplicației este de a oferi un mediu artiștilor pentru a descoperi potențialul unui asistent digital, precum și testarea metodelor prin care se poate combina domeniului de programare, îndeosebi inteligența artificială, cu domeniul artistic. Această lucrare nu își limitează grupul posibil de utilizatori, fiind ușor de folosit de către orice individ, indiferent de experiență sau interes în oricare dintre domeniile abordate. Mai mult, dezvoltarea acestei aplicații a avut și un scop personal de cercetare a legăturii dintre animație și inteligență artificială, a metodelor deja existente și posibilelor dezvoltări pe viitor.

Motivația principală din spatele alegerii acestui subiect o reprezintă interesul în potențialul inteligenței artificiale de a deveni un instrument util pentru artiști în procesul lor creativ. Din acest motiv, aplicația încorporează mai multe instanțe de personalizare a animațiilor care pot fi utilizate cel mai eficient de o persoană cu îndemânare artistică, obținând astfel rezultate unice.

Totodată, una dintre inspirațiile folosite pentru alegerea temei a fost popularitatea în ultimii ani ai conceptului de VTuber, o categorie de creator de conținut online care folosește un personaj animat în loc de camera web. În ciuda diferenței de complexitate între aplicația folosită de aceștia și ceea ce mi-a propus în această lucrare, consider că o variantă simplificată a modelelor 2D pe care o pot modifica utilizatorii fără a trebui să plătească pentru echipamentul necesar reprezintă un pas în direcția corectă spre a inova domeniul de animație.

Pe de altă parte, aplicația își propune să familiarizeze toți utilizatorii cu conceptele de bază ale vederii artificiale, oferindu-le posibilitatea de a modifica procesul de animație și de a se „juca” cu opțiunea de animație în timp real.

1.3 Contribuția proprie în realizarea lucrării

Contribuția proprie în realizarea lucrării de licență constă în dezvoltarea procesului de animație, cu excepția modelului de detecție, crearea interfeței web pentru soluția obținută și desenarea imaginilor necesare procesului.

În ceea ce privește modelele de detecție existente și cel ales de mine, contribuția personală se reflectă în cercetarea și testarea modelelor pre-antrenate existente pentru a descoperi cel mai potrivit model care să îndeplinească cerințele impuse de proiect. După identificarea metodei potrivite, am efectuat teste asupra modelului selectat pentru a optimiza rezultatele.

În sfera vederii artificiale și a procesării datelor de intrare, am dezvoltat în totalitate funcțiile care compun animația, printre care se numără: adăugarea imaginilor .png (cu excepția capului), adăugarea capului ținând cont de orientarea în spațiu și înclinare, obținerea unui cadran din două sau trei puncte și organizarea coordonatelor necesare.

Am realizat integral și dezvoltarea aplicației web, pentru a asigura conectarea acesteia la procesul de generare a animației pentru a putea utiliza funcțiile necesare. Acest pas a inclus construirea funcționalităților paginilor web în funcție de implementarea generării animațiilor pentru a obține o soluție eficientă și intuitivă.

Partea creativă a proiectului include crearea modelelor 2D. Acest pas a fost efectuat manual: am desenat personal imaginile folosind aplicația MediBang Paint, folosind o referință suport (Fig. 1.1) ca bază pentru primul model. Pe parcursul dezvoltării proiectului, am efectuat modificări asupra modelului 2D pentru a asigura o calitate înaltă și un rezultat care corespunde pe deplin cerințelor proiectului.

Astfel, contribuția mea în acest proiect nu se limitează doar la aspectele tehnice, ci se extinde și la partea artistică pentru a asigura o soluție completă și bine integrată.

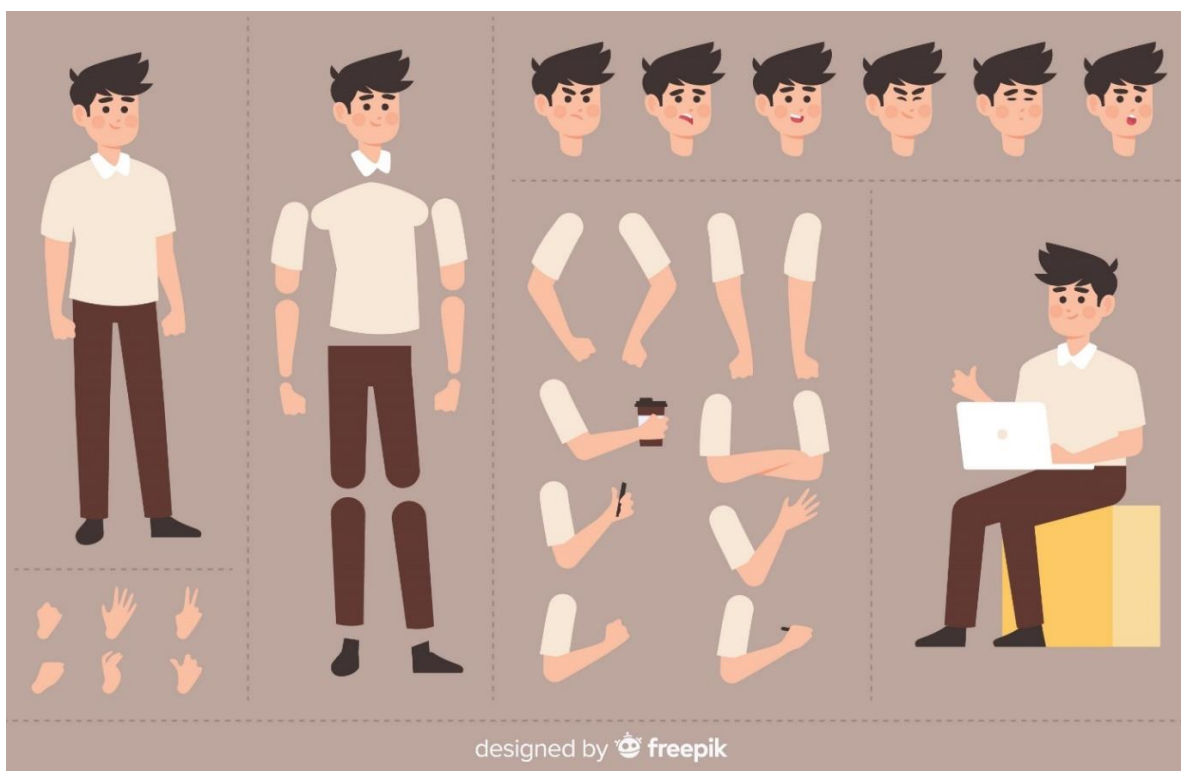


Fig. 1.1 – Caracter animat pentru motion design (Pikisuperstar, f.a.)

1.4 Structura lucrării

Lucrarea este structurată în jurul programului de generare a animațiilor și al modul de funcționare a acestuia. Din acest motiv, proiectul cuprinde explicația modelului de detecție folosit și potențialele alternative, precum și motivarea alegerii finale. Pe parcursul lucrării, am intrat în detalii în ceea ce privește tehnologiile utilizate, fundamentele teoretice, implementarea algoritmului folosit la animație, funcționalitățile din aplicația web, concluziile obținute, dar și posibilele oportunități de completare a acestui proiect.

Al doilea capitol, *Noțiuni științifice*, prezintă informații generale despre domeniul de studiu ales, precum și tehnici relevante în contextul acestui proiect. Printre definițiile și detaliile oferite, am menționat și modul de utilizare a tehnologiilor utilizate în lucrare.

Cu ajutorul contextului stabilit în capitolul anterior, *Tehnologii utilizate* prezintă mediul de lucru, baza de date, limbajele de programare și bibliotecile folosite în procesul de dezvoltarea al aplicației web. Mai mult, sunt prezentate detalii despre modul în care au fost utilizate aceste componente pentru a obține proiectul final. Ultima parte introduce modelul pre-antrenat de detecție a poziției corpului uman.

În capitolul al patrulea, se continuă cu mai multe detalii despre modelul de la MediaPipe (*MediaPipe Solutions guide*, f.a.), comparându-l cu diverse modele, cât și cu diferite configurări ale acestuia. Acest lucru ajută în evidențierea avantajelor și dezavantajelor soluției alese. În prezentarea rezultatelor obținute, am precizat motivele pentru care aplicația utilizează metoda aleasă.

Următorul capitol, *Prezentarea aplicației* conține o descriere a fiecărei pagini web și funcționalitățile acestora atât din punct de vedere al dezvoltării, cât și din cel al utilizatorului care folosește aplicația. În ceea ce privește dezvoltarea și procesul prin care s-au obținut rezultatele, se intră în detalii tehnice și se prezintă părțile relevante ale programului. Pe cealaltă parte, va prezenta experiența unui utilizator normal pe aplicația finală și cum poate fi folosită fiecare funcționalitate.

În continuarea lucrării, este analizat programul responsabil de obținerea animației. Acesta stă la baza aplicației și influențează în mod direct modul de funcționare al website-ului. Se prezintă algoritmul din spate, modul de gândire care a dus la acest rezultat, precum și problemele întâmpinate pe parcursul dezvoltării proiectului.

În cele din urmă, sunt prezentate concluziile obținute după finalizarea lucrării, funcționalitățile care pot îmbunătăți aplicația și posibilitatea de continuare a lucrării în viitor pe partea de cercetare.

Capitolul 2

Noțiuni științifice

2.1 Inteligența artificială

Inteligența artificială (IA) reprezintă simularea proceselor folosind gândirea umană de către mașini, în special sisteme de calcul. Aplicațiile specifice ale IA includ sisteme expert, învățare automată, procesare a limbajului natural, recunoaștere vocală și vederea artificială. (Laskowski et al., 2024)

Expansiunea rapidă a domeniului inteligenței artificiale, care s-a făcut remarcant prin capacitatea de a oferi soluții inovatoare la probleme complexe, a deschis calea către noi frontiere în tehnologie și cercetare. Aceasta a revoluționat o multitudine de domenii, printre care: medicină, prin diagnosticarea bolilor, analizarea imaginilor medicale și monitorizarea pacienților; educație, prin evaluarea automată și analiza performanțelor elevilor, dar și finanțe, deoarece este folosită pentru detectarea fraudelor, analizarea riscurilor și automatizarea tranzacțiilor. Acestea sunt numai câteva exemple.

Datorită avansurilor făcute cu ajutorul inteligenței artificiale, putem asuma că acest sector al programării va continua să se dezvolte și să modeleze viitorul umanității în numeroase industrii. Aceasta a devenit deja principalul motor al tehnologiilor precum big data și este adeseori asociată cu robotica sau IoT (internet of things), motiv pentru care multe companii au început să integreze inteligența artificială în activitățile lor.

Metodele de aplicare a inteligenței artificiale care au fost folosite direct sau indirect în dezvoltarea proiectului sunt: învățarea automată (machine learning) și vederea artificială (computer vision). Cea din urmă se folosește de diverse ramuri ale domeniului pentru a altera datele de intrare oferite și a obține rezultate vizuale sau numerice.

În continuare, voi prezenta informații despre aceste două tehnologii și modurile lor de utilizare în contextul dezvoltării acestei lucrări.

2.1.1 Învățarea automată

În ceea ce privește automatizarea, aceasta era una dintre ramurile în continuă dezvoltare a domeniului, ceea ce crează oportunități pentru folosirea învățării automate. Replicarea unor modele clar definite, antrenate pe seturi de date oficiale, poate fi cu adevărat utilă și pentru multiple domenii, printre care și cel al animației.

Pentru a defini ce este și la ce folosește aceasta am apelat la articolul *Introduction to Machine Learning, Neural Networks, and Deep Learning*: „Învățarea automată (ML) este un domeniu care se concentrează pe aspectul de învățare al inteligenței artificiale prin dezvoltarea algoritmilor care reprezintă cel mai bine un set de date. Aceasta utilizează submulțimi de date pentru a genera algoritmi, iar aceștia pot să folosească combinații diferite de caracteristici și greutatea (weights) derivate din principii fundamentale ale inteligenței artificiale. În învățarea automată, există patru metode frecvent utilizate, fiecare folosită pentru a rezolva diferite sarcini: învățare supervizată, învățare nesupervizată, învățare semisupervizată și învățare prin consolidare (reinforcement learning).” (Choi et al., 2020)

Folosind datele din articolul menționat, am formulat următoarele definiții ale metodelor de învățare automată. Învățarea supervizată se concentrează pe antrenarea unui algoritm pentru a prezice rezultate bazate pe date etichetate. În schimb, învățarea nesupervizată detectează modele într-un set de date fără a fi ghidată de date etichetate, și se folosește de gruparea datelor în grupuri bazate pe similarități neașteptate pentru a crea o logică. Învățarea semisupervizată reprezintă o combinație dintre primele două metode și utilizează date parțial etichetate și parțial neetichetate. Învățarea prin consolidare antrenează un algoritm să obțină rezultate dorite printr-un proces de încercare și eroare, similar cu modul în care o persoană învață.

Pe de altă parte, învățarea automată s-a dezvoltat și a condus la apariția învățării profunde (deep learning), care este utilizată în diverse aplicații, inclusiv recunoașterea imaginilor.

Învățarea profundă este un subset al învățării automate care utilizează rețele neuronale cu multiple straturi de procesare neliniară. Aceste arhitecturi sunt mai eficiente în reprezentarea funcțiilor complexe decât posibilele alternative. Antrenarea rețelelor profunde implică pre-antrenarea strat-cu-strat nesupravegheată, urmată de o ajustare fină supravegheată a întregii rețele, pentru a depăși dificultățile asociate cu metodele clasice. (Arnold et al., 2011)

2.1.2 Vederea artificială

Conform articolului *Everything You Ever Wanted To Know About Computer Vision*, vederea artificială este una dintre cele mai influente ramuri ale programării, aceasta fiind „un domeniu al informaticii care se concentrează pe imitarea anumitor aspecte ale complexității sistemului de viziune umană și pe capacitatea calculatoarelor de a identifica și procesa obiecte în imagini și videoclipuri într-un mod similar cu cel al oamenilor.” (Mihajlovic, 2019)

Rețelele neuronale convoluționale (CNN) sunt utilizate, în primul rând, pentru procesarea datelor vizuale, motiv pentru care sunt des asociate cu vederea artificială. Acestea sunt alcătuite din trei tipuri de straturi. Acestea sunt straturi convoluționale, straturi de pooling și straturi complet conectate, iar atunci când aceste straturi sunt suprapuse, se formează o arhitectură de tip CNN.

Aceste rețele neuronale convoluționale se remarcă în sarcini precum clasificarea imaginilor, detectarea obiectelor, segmentarea și alte task-uri de recunoaștere vizuală datorită abilității lor de a învăța automat reprezentări ierarhice ale datelor direct din valorile pixelilor, fără a avea nevoie de extragerea manuală a caracteristicilor. (O'Shea et al., 2015)

În ceea ce privește situația curentă a dezvoltării acestui subdomeniu, unul dintre factorii care contribuie la avansul tehnologic este cantitatea enormă de date disponibile în ziua de astăzi. Ne referim la imagini și videoclipuri folosite pentru a crea seturi de date pe baza cărora sunt antrenate și îmbunătățite performanțele modelelor de vedere artificială. Mai mult, principiile de bază ale vederii artificiale se regăsesc integrate în produse pe care le utilizăm în mod cotidian, cum ar fi mașinile autonome, recunoaștere facială sau a amprente.

2.2 Dezvoltarea web folosind Python

Dezvoltarea web se referă la crearea și întreținerea site-urilor, funcționalitățile acestora dictează complexitatea produsului final. Impresia inițială față de orice aplicație web este, în general, legată de elementele sale vizuale, iar din acest motiv, design-ul reprezintă o parte la fel de importantă precum calitățile tehnice ale unui site. Pentru obținerea unei aplicație web care satisface cât mai multe dintre caracteristicile dorite, este bine de atins un anumit echilibru între elementele estetice și cele funcționale.

În referință cu dezvoltarea unei aplicații web în Python, există numeroase framework-uri care contribuie la implementarea fișierelor necesare pentru menținerea unui website: bottle.py,

Flask, CherryPy, Pyramid, Django și web2py. Multe dintre aceste framework-uri sunt folosite de aplicații populare precum Reddit sau Spotify.

Django este, la momentul actual, cel mai popular framework pentru dezvoltare web în Python. Conform paginii oficiale a Django, acesta este „un framework web de nivel înalt care încurajează dezvoltarea rapidă și un design curat, pragmatic. Construit de dezvoltatori experimentați, se ocupă de multe dintre dificultățile dezvoltării web, astfel încât să vă puteți concentra pe scrierea aplicației dvs. fără a fi nevoie să reinventați roata. Este gratuit și open-source.” (*Meet Django*, f.a.)

Pe de altă parte, unul dintre cele mai recomandate framework-uri pentru începători este Flask, dat fiind faptul că este ușor de folosit și, totodată, destul de versatil pentru a crea proiecte complexe. Folosind cuvintele lui Deery M. din articolul *What Is Flask and How Do Developers Use It? A Quick Guide*: „Flask este un micro-framework pentru dezvoltatori, conceput pentru a le permite să creeze și să scaleze aplicații web rapid și simplu. Este considerat un "micro-framework" deoarece nu se bazează pe alte instrumente sau biblioteci de programare pentru a funcționa.” (Deery, 2023)

În ceea ce privește popularitatea fiecărui framework din Python, există următoarele statistici prezentate în figura 2.1, de pe site-ul oficial de la JetBrains. (*The State of Developer Ecosystem 2022*, 2022)

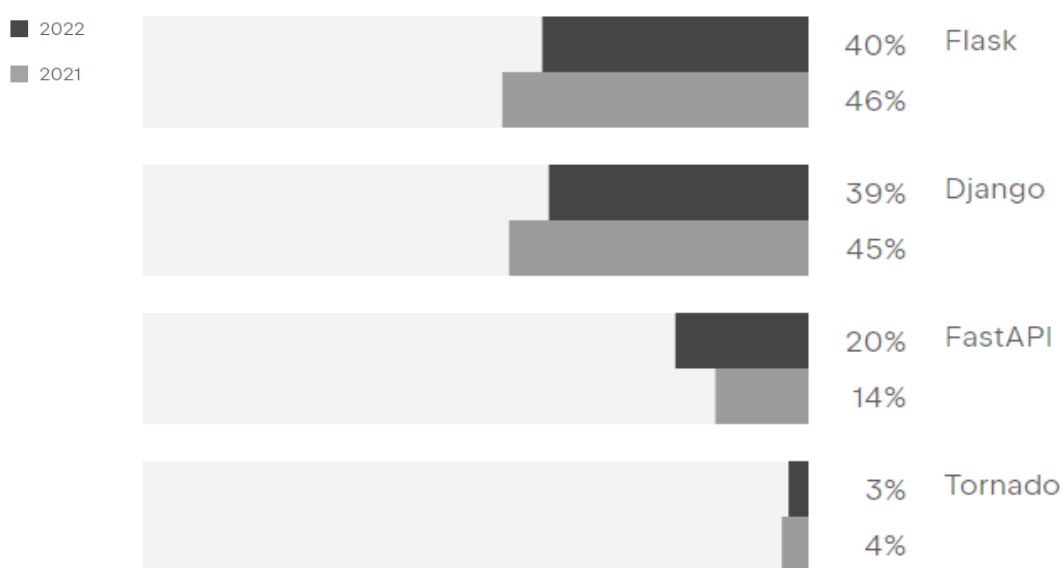


Fig. 2.1 – Ce framework-uri se folosesc pentru dezvoltare web în Python? (*The State of Developer Ecosystem 2022*, 2022)

Din cauza simplității Flask există diferențe clare în ceea ce privește comparația acestuia cu competiția, iar unele dintre aspectele la care excelează Django sunt securitate și accesul mai ușor la baza de date.

În ciuda acestor dezavantaje, am considerat că Flask se potrivește mai mult aplicației acesteia, deoarece funcționalitățile website-ului dezvoltat nu includ o comunitate bazată pe utilizatori care trebuie întreținută și securizată cu atenție. Tot din acest motiv, baza de date nu ține cont de utilizatorii care folosesc aplicația, ci numai de fișierele încărcate de aceștia. Astfel, Flask reprezintă o modalitate mai simplă pentru a atinge obiectivele propuse.

Capitolul 3

Tehnologii utilizate

3.1 Limbaje de programare și biblioteci

3.1.1 Python

În dezvoltarea lucrării am utilizat Python, versiunea 3.9. Acest limbaj de programare este extrem de versatil și conține în arsenalul său un număr mare de biblioteci bine optimizate, multe fiind create în alte limbaje, precum C/C++. Din acest motiv, Python este una dintre cele mai ușoare limbaje pentru a programa și, de multe ori, oferă performanțe similare cu celelalte opțiuni.

Conform GeeksforGeeks și nu numai: „Python este unul dintre cele mai ușor de învățat, dar și cele mai utile limbaje de programare, utilizat pe scară largă în industria software. Oamenii folosesc Python pentru Programare Competitivă, Dezvoltare Web și crearea de software. Datorită sintaxei sale simple, este recomandat începătorilor din domeniul ingineriei software.” (*What is Python? Its uses and applications*, 2024)

Python se remarcă prin simplitatea și lizibilitatea sa, iar sintaxa sa curată și intuitivă seamănă cu limba engleză, permițând dezvoltatorilor să se concentreze pe rezolvarea problemelor. În plus, versatilitatea Python este de neegalat, deoarece acesta suportă multiple paradigme de programare și vine cu o bibliotecă standard extinsă, făcându-l adaptabil pentru o gamă largă de sarcini, de la dezvoltare web la inteligență artificială. Un alt avantaj al Python este comunitatea sa robustă. Aspectul open-source a limbajului a cultivat o comunitate mare și activă, care contribuie la îmbunătățirea continuă a acestuia și la mărirea bibliotecii de module și instrumente accesibile.

Toate avantajele limbajului Python sunt evidente prin simplul fapt că este unul dintre cele mai folosite limbaje. Conform figurii 3.1, preluată de pe website-ul Statistica, Python este al treilea cel mai folosit limbaj de programare după JavaScript și HTML/CSS. (Vailshery, 2024)

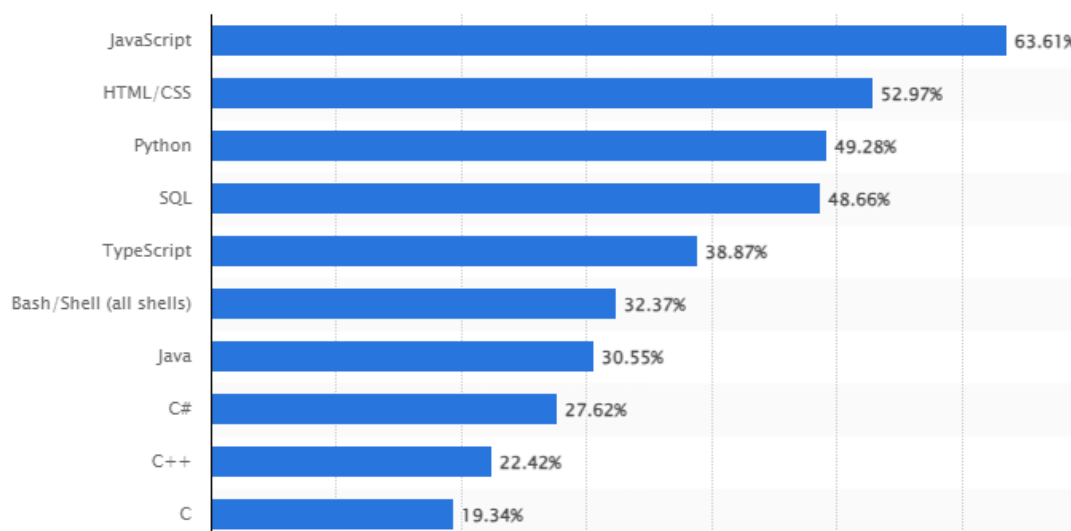


Fig. 3.1 – Cele mai folosite limbaje de programare de către programatori în 2023 (Vailshery, 2024)

3.1.2 Bibliotecile Python

Bibliotecile utilizate în implementarea algoritmului de generare a animației sunt: math, time, tempfile, numpy (1.26.4), opencv-python (4.9.0.80), mediapipe (0.10.14). Datorită utilizării MediaPipe, nu au fost necesare pachete precum scikit-learn care se ocupă de antrenarea modelului de detecție.

Din bibliotecile elementare ale Python, am folosit math pentru că oferă numeroase funcții matematice pentru operațiile aritmetice de bază și calculele numerice. Biblioteca time este folosită pentru calcularea timpului de procesare a imaginilor, necesar pentru anumite funcții. Tempfile oferă funcții pentru crearea fișierelor și folderelor temporare care, în cazul acestei lucrări, au fost utilizate pentru gestionarea unor variabile înainte de salvarea lor în baza de date.

OpenCV este o bibliotecă esențială pentru modificările și procesarea imaginilor, permițând dezvoltatorului să manipuleze eficient imaginile. Pe website-ul oficial de la OpenCV este menționat că aceasta „conține peste 2500 de algoritmi optimizați, care includ un set cuprinzător de algoritmi clasici și de ultimă generație pentru vederea computerizată și învățare automată.” (About OpenCV, f.a.)

Pe de altă parte, NumPy își îndeplinește scopul în calcul numeric, fiind renumită pentru capacitatea sa de manipulare eficientă a matricelor și tablourilor, alături de o gamă largă de funcții matematice pentru operațiuni complexe. Așa cum precizează documentația oficială pentru NumPy: „La baza pachetului NumPy se află obiectul de tip ndarray. Acesta încapsulează

tablouri n-dimensionale de tipuri de date omogene, cu multe operații efectuate în cod compilat pentru performanță.” (*What is NumPy?*, f.a.)

Pe lângă bibliotecile deja prezentate, dezvoltarea aplicației web a necesitat și următoarele: io, base64, Flask (3.0.3), Flask-SQLAlchemy (3.1.1).

Biblioteca io din Python oferă instrumente pentru gestionarea operațiunilor de intrare și ieșire prin fluxuri, cum ar fi citirea și scrierea fișierelor și lucrul cu buffer-uri în memorie. Suportă atât date text, cât și binare, motiv pentru care a fost util în tranziția de la tipul de date blob din baza de date și NumPy array-urile necesare pentru utilizarea OpenCV.

Base64 oferă funcții pentru codificarea și decodificarea datelor binare, această schemă de codificare convertește datele binare într-un format de șir ASCII. A fost folosit pentru codificarea și decodificarea imaginilor din baza de date.

Prezentat anterior, Flask este un framework ușor de folosit și flexibil. Acesta asistă în gestionarea sarcinilor precum rutarea, organizarea cererilor și randarea șabloanelor, permițând dezvoltatorilor să construiască aplicații web rapid și eficient. Drept extensie a Flask, am folosit Flask-SQLAlchemy pentru a simplifica operațiile dintre aplicație și baza de date. În documentația oficială pentru SQLAlchemy, care stă la baza extensiei din Flask, este precizat că aceasta „oferă o suită completă de modele bine-cunoscute de persistență la nivel de întreprindere, concepute pentru accesul eficient și de înaltă performanță la bazele de date.” (*The Python SQL Toolkit and Object Relational Mapper*, f.a.)

3.1.3 HTML/CSS

HTML (HyperText Markup Language) reprezintă un limbaj de marcare pentru documente, cel mai des utilizat pentru crearea și organizarea paginilor web. Acesta oferă blocurile de bază pentru conținutul web și folosește elemente precum titluri, paragrafe, imagini și link-uri pentru a defini structura și aspectul paginilor create.

CSS (Cascading Style Sheets) este un limbaj de stiluri folosit pe partea de design, pentru îmbunătățirea aspectului documentelor HTML. Astfel, dezvoltatorii pot să modifice o multitudine de elemente: culorile, fundalul, fonturile și așezarea în pagină.

Faptul că acestea sunt compatibile și utilizate pe majoritatea platformelor existente nu reprezintă singurul motiv pentru care sunt atât de populare. În articolul *Benefits of using CSS*

and HTML in Web Development de pe Medium, sunt elaborate multiple avantaje ale utilizării acestora. Câteva dintre acestea sunt: îmbunătățirea vitezei de încărcare a paginii, asigură consistența design-ului, împărțirea clară și semnificativă a conținutului și crearea de funcționalități interactive. (Khan, 2023)

În ciuda faptului că acestea nu sunt limbaje de programare, contribuția lor la crearea aplicației web nu poate fi ignorată. Am utilizat HTML / CSS pentru a crea și modifica fiecare pagină din website și pentru a organiza elementele într-un mod plăcut și ușor de navigat. Câteva dintre elementele implementate cu ajutorul acestora sunt: pagina principală care folosește un grid cu elemente de navigare pentru o prezentare mai simpatică și descriptivă a paginilor web, bara de navigare care se poate restrânge în funcție de dimensiunea ferestrei, precum și utilizarea de fișiere .gif care cresc complexitatea estetică a aplicației.

3.1.4 JavaScript

JavaScript este un limbaj de programare versatil folosit în mod obișnuit pentru crearea elementelor interactive și dinamice pe aplicațiile web. Acesta este folosit pentru îmbunătățirea experienței utilizatorului prin adăugarea unor funcții precum validarea formularelor, animații și actualizări de conținut dinamice, făcând paginile web mai captivante și mai responsive.

Printre avantajele utilizării acestui limbaj se numără și cele precizate în articolul *Advantages of JavaScript* de pe Code Institute. Unul dintre acestea este viteza, deoarece este un limbaj „interpretat” care reduce timpul necesar pentru compilare. Mai mult, rulează la nivel de client, reducând astfel încărcarea pe server și permițând acestuia să se concentreze pe alte sarcini importante. (Tuama, f.a.)

Cunoscut pentru ușurința sa de utilizare, oferă o interfață bogată și versatilă pentru dezvoltatorii web, dar și interoperabilitate cu alte limbaje de programare, ceea ce face din JavaScript una dintre cele mai populare și puternice tehnologii pentru dezvoltarea web. (Tuama, f.a.)

Această lucrare folosește JavaScript în majoritatea fișierelor .html pentru implementarea unor funcționalități diverse: schimbarea dintre bara de navigare normală și meniul dropdown, apelarea diverselor funcții și redirecționarea la o pagină specifică după îndeplinirea condiției specificate, blocarea butoanelor când utilizatorul nu ar trebui să aibă acces la ele, dar și apariția dinamică a unor elemente.

3.2 Mediul de lucru: PyCharm

În dezvoltarea aplicației web, am folosit PyCharm pentru a asigura un mediu de lucru optim care oferă asistența necesară. Din documentația oficială de la JetBrains, PyCharm este „un mediu de dezvoltare integrat (IDE) dedicat Python, oferind o gamă largă de instrumente esențiale pentru dezvoltatori, strâns integrate pentru a crea un mediu convenabil pentru dezvoltarea productivă a programelor Python, web și știința datelor.” (*Quick start guide*, 2024)

Cele două versiuni ale IDE-ului asigură că orice programator se poate folosi de tehnologia de la JetBrains pentru a își ușura procesul de dezvoltare. Interfața aplicației este complexă, dar nu aglomerată, fiind destul de intuitivă pentru începători.

Totodată, acesta oferă o multitudine de opțiuni, dezvoltatorii putând personaliza mediul de lucru în funcție de preferințele personale, configurând setările IDE-ului conform nevoilor. Pe lângă acestea, aceștia pot adăuga sau elimina plugin-uri. Asistența oferită de către PyCharm pe parcursul dezvoltării oricărui proiect poate salva timp prețios dezvoltatorului, asigurându-se că există oportunitatea de a îndepli cât mai multe obiective.

Instrumentele puse la dispoziție de către PyCharm, cum ar fi evidențierea sintaxei, completarea automată a codului, verificarea codului și debugger-ul, sunt bine implementate și ajută în evitarea unor erori simple, dar care pot consuma timp. Funcțiile precum refactoring-ul inteligent, navigarea rapidă și completarea automată a codului, duc la sporirea productivității în timpul dezvoltării proiectelor Python.

PyCharm se evidențiază ca unul dintre cele mai puternice și complete medii de dezvoltare integrate pentru Python. Suportul său pentru framework-uri populare, capacitatea de personalizare și actualizările constante, împreună cu o comunitate activă, îl fac o alegere ideală pentru gestionarea și dezvoltarea proiectelor complexe.

3.3 Baza de date

Pentru a asigura conexiunea și stocarea eficientă a datelor, am utilizat o bază de date .sqlite3. În ciuda funcționalităților limitate și a scalabilității restrânse, bazele de date .sqlite3 conferă destule avantaje în ceea ce privește portabilitatea, performanța și compatibilitatea cu majoritatea platformelor.

Pentru a vizualiza mai ușor schimbările din baza de date pe parcursul testării funcționalităților, am utilizat DB Browser for SQLite (DB4S). Folosirea acestei aplicații sau a unei alternative nu este o necesitate pentru dezvoltarea proiectului, însă această tehnologie vizuală, folosită pentru proiectarea și editarea tabelelor și datelor introduse în baza de date, ajută pe parcursul activității. (*DB Browser for SQLite*, f.a.)

Această aplicație permite observarea rezultatelor în timpul procesului și verificarea datelor din baza de date. Din cauza modului de stocare, interfața nu permite vizualizarea videoclipurilor salvate, însă imaginile pot să fie afișate fără nici o problemă.

În contextul dezvoltării unei aplicații web, implementarea și integrarea unei baze de date asigură persistența și integritatea datelor pe termen lung. Mai mult, baza de date facilitează gestionarea datelor, permițând organizarea și manipularea acestora în mod structurat.

Considerând articolul *The Role of Database in Web Application Development*, am realizat că aplicațiile web trebuie să poată prelucra mai multe cereri în același timp, deoarece sunt folosite de mai mulți utilizatori simultan. Pentru a face față acestui volum de cereri, bazele de date ale aplicațiilor web folosesc o arhitectură distribuită, ceea ce le permite să scaleze rapid și să gestioneze cantitatea mare de solicitări fără probleme. (*The Role of Database in Web Application Development*, 2022)

Chiar dacă proiectul nu este în punctul în care necesită o bază de date puternică, iar echipamentul folosit nu permite implementarea unei arhitecturi distribuite, este un aspect important care trebuie luat în considerare. Pe lângă toate avantajele organizării fișierelor într-o bază de date, motivul principal pentru care am integrat o bază de date este abilitatea de a stoca imagini și videoclipuri sub formatul blob (binary large object). Acest tip de date asigură integritatea fișierelor în timp ce simplifică recuperarea și manipularea structurilor de date complexe, în cazul acesta imagini și clipuri video.

În dezvoltarea proiectului a fost utilizat următorul tabel pentru legătura cu aplicația web: un tabel pentru upload și procesare care stochează un id, un nume de fișier, datele de intrare și, ulterior, rezultatul, denumirea modelului și a imaginii de fundal alese, dar și o imagine de fundal încărcată de utilizator. Majoritatea datelor sunt opționale și pot să fie salvate aleator în baza de date, lucru care a fost folosit pentru pagini precum animația în timp real care necesita imaginea de fundal și modelul 2D ales înainte de orice altă variabilă.

3.4 Model de detecție: MediaPipe

MediaPipe (*MediaPipe Solutions guide*, f.a.) a fost creat de către Google, fiind dezvoltat inițial de către Google Research în cadrul Google AI & Machine Perception. Este o platformă open-source și cross-platform, care oferă instrumente și modele pre-antrenate pentru construirea de aplicații cu percepție de calcul în timp real.

Această tehnologie oferă o gamă largă de modele pre-antrenate, inclusiv modele pentru detectarea scheletului uman. Modelele utilizează rețele neurale pentru a identifica și a localiza punctele-cheie ale corpului uman în imagini sau în fluxuri video. (*Pose landmark detection guide*, f.a.)

Această capacitate este esențială pentru o varietate de aplicații, de la recunoașterea gesturilor la analiza posturii în timp real în domenii precum realitatea augmentată, jocurile video și asistența medicală. Cu ajutorul modelelor MediaPipe pentru detectarea scheletului uman, dezvoltatorii pot crea aplicații inovatoare și interactive care îmbunătățesc experiența utilizatorului și extind posibilitățile interacțiunii om-calculator.

Unul dintre motivele principale pentru alegerea acestui model pre-antrenat, este eficiența și viteza acestuia. Rezultatele sunt obținute rapid și sunt, de cele mai multe ori, corecte. Detaliile privind datele de antrenare utilizate pentru modelele acestea nu sunt specificate în documentația oficială. Cu toate acestea, Google utilizează adesea o combinație de seturi de date publice, date deținute de companie și date augmentate pentru scopuri de antrenare.

În ceea ce privește concurența în domeniul detectării posturii, MediaPipe nu este singurul jucător de pe piață, iar câțiva dintre competitori includ framework-uri și platforme precum OpenPose (*OpenPose*, 2017), PoseNet (*PoseNet: Pose Estimation*, 2018) și DensePose (*DensePose*, f.a.). Toate aceste soluții oferă, de asemenea, capabilități de detectare a poziției corporale și de urmărire a mișcării, și sunt folosite în diverse aplicații, inclusiv realitate augmentată, animație, asistență medicală și fitness.

Extras din repository-ul oficial al MediaPipe Pose, aceasta este descrisă drept „o soluție de învățare automată pentru urmărirea cu fidelitate ridicată a posturii corpului, deducând 33 de repere 3D și o mască de segmentare a fundalului pentru întregul corp din cadre video RGB utilizând BlazePose.” (*MediaPipe Pose*, 2019)

În ciuda existenței reperelor 3D, a treia coordonată este adeseori incorectă, imprevizibilă în timpul mișcării și duce la rezultate eronate. Din acest motiv, pentru lucrarea de licență nu am folosit coordonatele 3D, ci am extras numai versiunea 2D.

Conform documentației de pe Github (*MediaPipe Pose*, 2019), acestea sunt rezultatele celor trei modele MediaPipe Pose în comparație cu alte două modele:

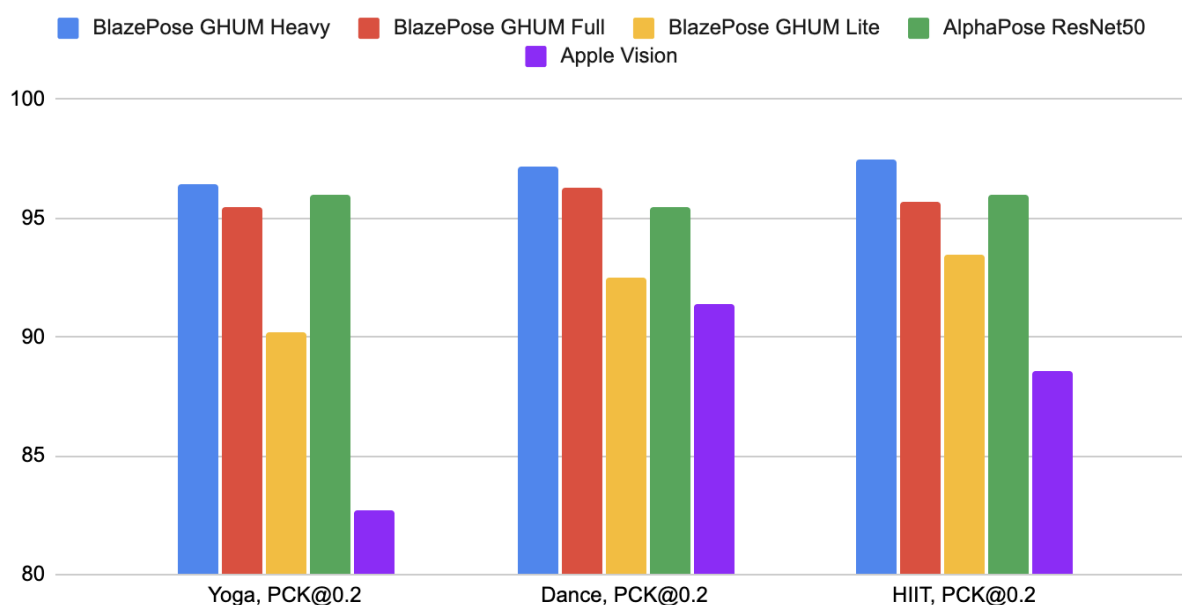


Fig. 3.2 - Calitatea estimării poziției umane (*MediaPipe Pose*, 2019)

Pentru a clarifica grafic acesta (Fig. 3.2), BlazePose reprezintă o arhitectură de rețele convoluționale pentru estimarea poziției umane (Bazarevsky et al., 2020). GHUM este un sistem de modelare 3D a formei umane bazat pe învățare profundă, care antrenează modelele pentru estimarea poziției și formei corpului uman (Xu et al., 2020). Acestea au stat la baza dezvoltării MediaPipe Pose.

AlphaPose (*AlphaPose*, 2017) și Apple Vision (*Apple Vision: Detecting Human Body Poses in Images*, 2017) sunt două modele de estimare a scheletului uman. Termenii *yoga*, *dance* și *HIIT* (high-intensity interval training) descriu tipul de date de testare folosite, acestea fiind alese pentru diversitatea pozițiilor corpului în timpul activităților sportive.

În această lucrare, am ales să folosesc varianta Full a modelelor pre-antrenate. Deoarece la baza proiectului stă implementarea animației în timp real, este necesar un anumit echilibru între performanță și acuratețea rezultatelor. Pentru a explica mai în detaliu, echipamentul folosit

pentru dezvoltarea și testarea lucrării este un laptop, cu Windows 10 ProN, placă video GeForce GTX 1660 Ti, procesor AMD Ryzen 7, 2300 gHz, 4 nuclee, 8 procesoare logice și 16GB RAM.

Din cauza limitărilor impuse de echipament, am decis să nu folosesc versiunea Heavy care este mai lentă și produce rezultate similare cu versiunea Full. Următorul capitol prezintă mai multe detalii despre diferențele dintre modelele oferite de MediaPipe și configurația finală folosită în acest proiect.

Capitolul 4

Comparații cu modelul de detecție

4.1 MediaPipe vs OpenPose

Pentru a compara OpenPose și MediaPipe trebuie luate în considerare abordarea soluțiilor acestora, performanța în relație cu rezultatele și posibilitatea ca mecanismele folosite de acestea să fie prea complexe pentru aplicația web. Pornind această discuție de la documentul *Comparative Study of Human Pose Estimation* (Vishnu et al., 2023) care testează și compară patru dintre metodele existente de estimare a poziției umane.

De fapt și de drept, abordarea OpenPose folosește o metodă ascendentă (bottom-up), identificând părțile individuale ale corpului și apoi asamblându-le. Această abordare asigură o acuratețe ridicată în detectarea pozelor complexe și a mai multor persoane simultan. În schimb, MediaPipe folosește o abordare descendentă (top-down), detectând mai întâi întreaga persoană și apoi identificând punctele cheie. Această metodă simplifică procesul și îmbunătățește eficiența.

Cu toate acestea, rezultatele bune ale OpenPose sunt obținute datorită utilizării mai multor resurse, procesul devenind mai lent. În contrast, MediaPipe oferă o performanță echilibrată, furnizând o acuratețe bună în timp ce este mult mai eficientă din punct de vedere al resurselor. Modelele lor folosesc accelerarea hardware pentru a obține viteze de procesare în timp real, făcând-o ideală pentru aplicații care necesită răspunsuri rapide. (*MediaPipe Pose*, 2019)

Un alt avantaj al modelului OpenPose este posibilitatea de a detecta multiple schelete în același cadru, pe când MediaPipe, chiar dacă oferă această opțiune, nu se descurcă excepțional în această arie. Pe de altă parte MediaPipe obține, de cele mai multe ori, rezultate mai bune în situații dificile, precum lumină puternică sau întuneric, persoane la distanță, unghiuri ciudate ale camerei sau poziții complexe ale corpului.

Per total, OpenPose implică o post-procesare mai complexă datorită abordării sale, ceea ce poate îngreuna implementarea rapidă a soluțiilor pentru a obține rezultate mai precise.

MediaPipe, cu metoda sa ierarhică, simplifică post-procesarea și oferă o experiență de dezvoltare mai fluidă.

Din păcate, testarea modelului OpenPose este imposibilă la momentul actual, deoarece există probleme de compatibilitate, iar versiunile mai vechi nu mai funcționează. De pe Github-ul oficial al modelului OpenPose (*OpenPose*, 2017), există următoarele analize ale modelelor:




Build Type	Linux	MacOS	Windows
Build Status	 CI failing	 CI failing	 build failing

Fig. 4.1 – Statusul de funcționare al OpenPose pe diferite sisteme de operare (*OpenPose*, 2017)

Din aceste motive, pentru aplicațiile în timp real în care eficiența performanței și consumul redus de resurse sunt cruciale, MediaPipe se evidențiază ca fiind opțiunea superioară.

4.2 MediaPipe vs YOLOv7

În această secțiune, am folosit din nou articolul suport *A Comparative Study of Human Pose Estimation* (Vishnu et al., 2023) pentru a găsi diferențele dintre MediaPipe Pose și YOLOv7.

Pentru a prezenta pe scurt modelul MoveNet din articolul suport, voi folosi informațiile oferite de TensorFlow, dezvoltatorul acestui model. „MoveNet este un model ultra rapid și precis care detectează 17 puncte cheie ale corpului. Modelul este oferit pe TF Hub cu două variante, cunoscute sub numele de Lightning și Thunder. Lightning este destinat aplicațiilor unde latența este critică, în timp ce Thunder este destinat aplicațiilor care necesită o acuratețe ridicată.” (*MoveNet: Ultra fast and accurate pose detection model*, 2021)

MediaPipe	MoveNet	OpenPose	YOLOv7
Single User	Single User	Multiuser	Multiuser

Fig. 4.2 – Single-use sau multi-use pentru metodele prezentate (Vishnu et al., 2023)

Una dintre diferențele prezentate în articol este numărul de utilizatori (Fig. 4.2) care pot fi detectați de fiecare model. Pe când MediaPipe a fost creat pentru a detecta o singură entitate, YOLOv7 permite mai multe detecții în același cadru. Chiar dacă MediaPipe Pose are opțiunea

de a detecta utilizatori multipli (*Pose landmark detection guide*, f.a.), acest lucru încetinește procesul. În altă ordine de idei, MediaPipe se evidențiază prin acuratețea mult mai bună în imagini cu intensitate scăzută a luminii, în diferite orientări, la diferite distanțe de cameră și în videoclipuri cu mișcare. (Vishnu et al., 2023)

Pentru a mă convinge de rezultatele prezentate anterior, am decis să implementez YOLOv7 (yolov7-w6-pose) și să compar modelele între ele. Astfel, am ajuns la următoarele diferențe de detecție și de timp de procesare:

Fișier	Dimensiune	Durată	FPS	MediaPipe	YOLOv7
Image1.jpg	540 x 360	-	-	0.24 sec	1.75 sec
Image2.jpg	736 x 860	-	-	0.27 sec	2.26 sec
Video1.mp4	960 x 540	17.25 sec	24	16.30 sec	722 sec (~12min)
Video2.mp4	640 x 360	2.56 sec	25	2.61 sec	109 sec (~2min)

Tabel 4.1 – Comparație între modelele MediaPipe Pose și YOLOv7

Unica problemă a implementării YOLOv7 este, de fapt, timpul de execuție mult mai lent, așa cum se poate observa în tabelul 4.1. Rezultatele puțin mai bune ale detecției nu justifică diferența de timp, iar pentru o aplicație de animație timpul este foarte important.

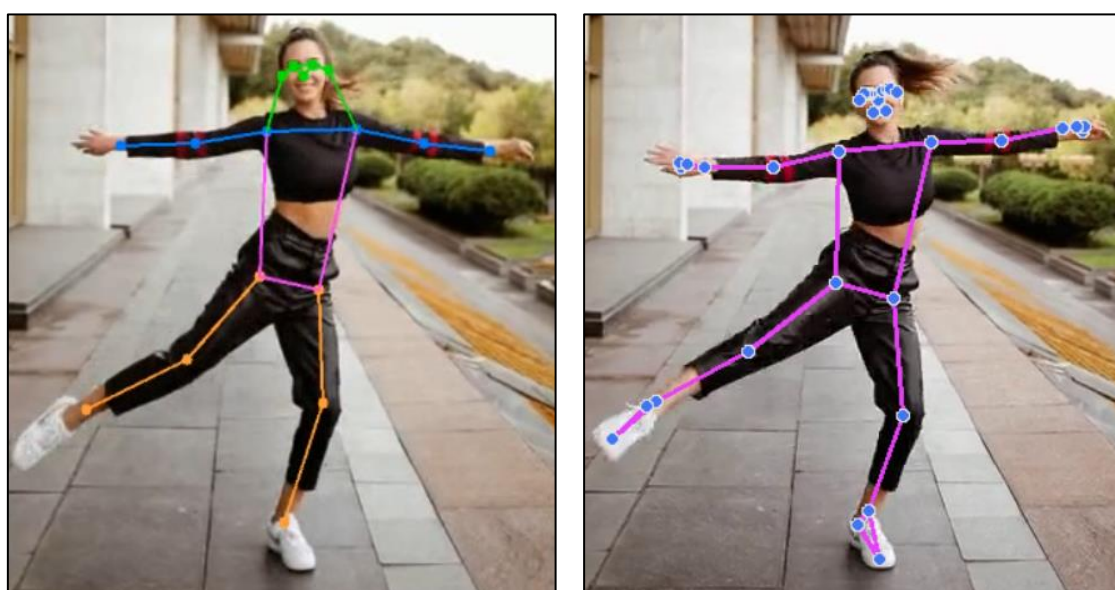


Fig. 4.3 – Rezultate ale detecției scheletului pe clipuri video, YOLOv7 (strânga) și MediaPipe (dreapta)

(Sursă clip video: Tankilevitch, f.a.)



Fig. 4.4 – Rezultate ale detecției scheletului pe imagini, YOLOv7 (strânga) și MediaPipe (dreapta)

(Surse imagini: Deagrez, f.a. și Sapaka, f.a.)

În ceea ce privește rezultatele vizuale, în ciuda similarităților, detecția YOLOv7 obține un set de puncte mai corecte decât MediaPipe. Acest lucru este vizibil în figurile 4.3 și 4.4, unde poziția corpului este interpretată mai bine de către modelul YOLOv7: trunchiul corpului are o formă mai naturală în toate cele trei exemple, coatele sunt detectate mai bine în cea de a doua poză din figura 4.4 și încheieturile sunt poziționate mai bine, deoarece mâinile și picioarele sunt proporționate corect și totodată țin cont de perspectivă. În apărarea modelului MediaPipe, acesta detectează mai bine glezna, deoarece este un punct de legătură între gambă și picior, vizibil în prima imagine din figura 4.4, dar și în figura 4.3.

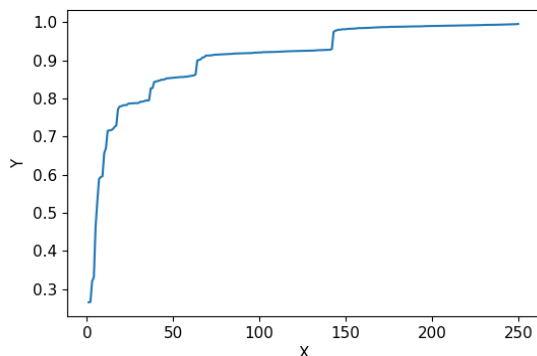


Fig.4.5 – Acuratețea YOLOv7 90.81%

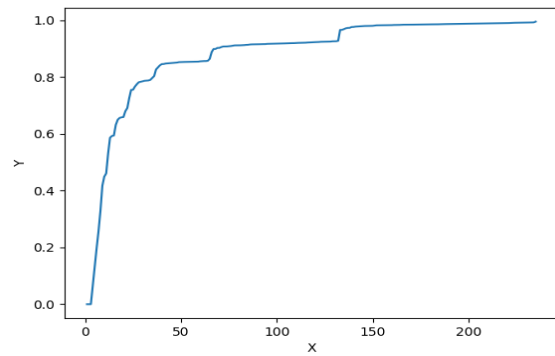


Fig. 4.6 – Acuratețea MediaPipe Pose 88.33%

Pentru a confirma diferențele dintre cele două modele, m-am folosit de 260 de imagini din setul de date *Human pose dataset (sit & stand pose classes)* (Toit, 2023) pentru a efectua un test de acuratețe. Am utilizat setările standard ale MediaPipe Pose (Full) și același model de YOLOv7 (yolov7-w6-pose), iar acuratețea medie obținută a fost 90.81% pentru YOLOv7 și 88.33% pentru MediaPipe Pose. Pentru obținerea acestora, am calculat distanța dintre punctul corect

(ground truth) și punctele rezultate de fiecare model, iar diferența (un număr între 0 și 1) reprezenta acuratețea pentru acel punct. Punctele folosite au fost: nas, urechi, umeri, coate, încheietura mâinii, șolduri, genunchi și glezne, în total 15 puncte. Cu datele găsite am generat și două grafice pentru acuratețea modelelor (Fig. 4.5 și 4.6).

În concluzie, chiar dacă atât rezultatele calitative cât și cele cantitative arată o superioritate a modelului YOLOv7, consider că aceste avantaje sunt nesemnificative în comparație cu resursele suplimentare pe care modelul YOLOv7 le necesită. Deoarece reducerea timpului de procesare este un aspect important al generării animațiilor, am ales să folosesc MediaPipe Pose în dezvoltarea proiectului.

4.3 Diverse configurații

Pornind de la secțiunea Pose a proiectului MediaPipe oficial de pe Github (*MediaPipe Pose*, 2019), am observat parametrii pe care îi pot modifica. Modelul are o opțiune pentru imagini statice (*static_image_mode*), alegerea dintre cele trei versiuni care au complexități diferite (*model_complexity*), valorile personalizate pentru detectarea și urmărirea scheletului uman (*min_detection_confidence*, *min_tracking_confidence*), dar și altele.

Următoarele teste au fost efectuate fără a folosi aplicația web, ci doar programul de detecție al MediaPipe Pose implementat în Python. După modificarea argumentelor funcției Pose() am ajuns la concluzia că modificarea minimului de încredere pentru detecția și urmărirea persoanei nu afectează foarte mult imaginile rezultate.

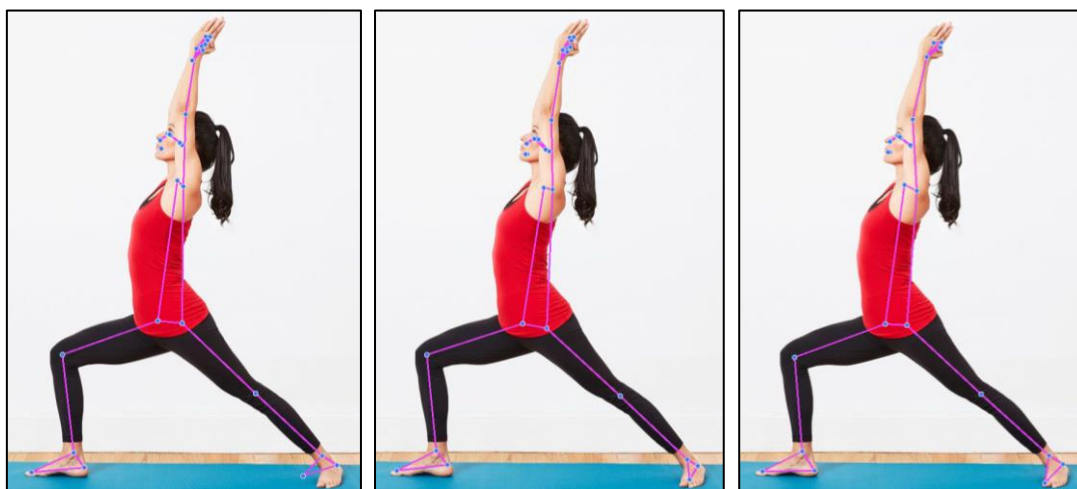


Fig. 4.7 - Comparație între modelele MediaPipe Pose folosind imagini: Lite (stânga), Full (mijloc) și Heavy (dreapta)

(Sursă imagine: Goldstein, f.a.)

În ceea ce privește alegerea unui model diferit dintre cele trei existente (Lite, Full sau Heavy), opțiunile produc rezultate vizibil diferite, așa cum putem observa în figura 4.7 unde piciorul, trunchiul, încheietura mâini și palma nu sunt detectate corect în versiunea Lite. Detecția este îmbunătățită în versiunea Full, unde piciorul și trunchiul sunt corecte. În final, poziționarea tuturor punctelor este mai aproape de realitate în rezultatele modelului Heavy.

Referitor la procesarea clipurilor video, există diferențe substanțiale între rezultatele obținute. Modificarea minimului de încredere pentru detecția și urmărirea persoanei nu cauzează schimbări vizibile asupra videoclipurilor, așa că am decis să folosesc valoarea medie de 50%.

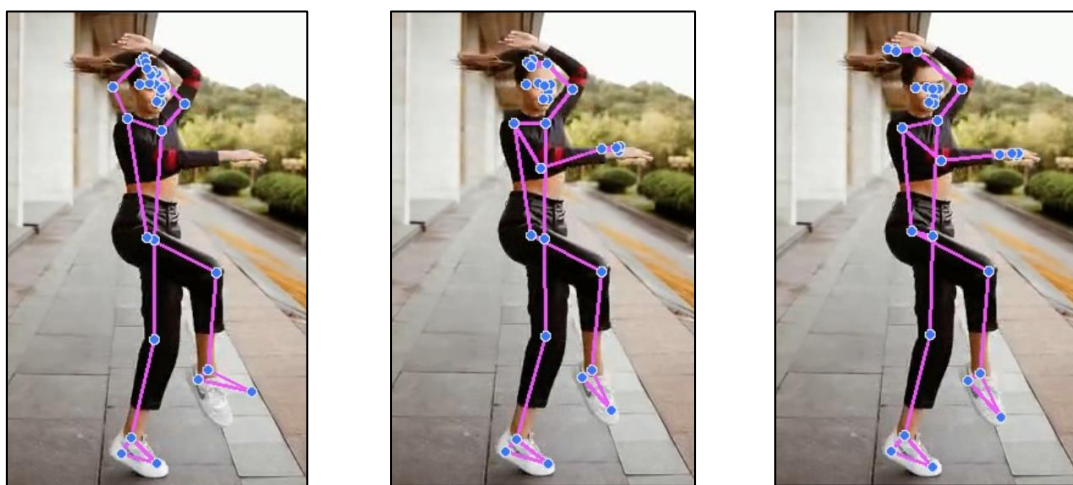


Fig. 4.8 - Comparație între cele trei modele MediaPipe Pose folosind clipuri video:

Lite (stânga), Full (mijloc) și Heavy (dreapta)

(Sursă clip video: Tankilevitch, f.a.)

Rezultatele obținute de modelul Lite prin comparație cu celelalte două sunt clar mai slabe, având momente în care scheletul deviază de la așteptări, lucru vizibil în figura 4.8. Astfel, în imaginea procesată cu Lite brațele și unul dintre picioarele persoanei sunt în poziții total greșite, iar expresia facială se contopește cu aceste puncte. Rezultatul acestei versiuni este extrem de slab. Prin comparație, Full oferă un schelet natural care reușește să urmărească forma persoanei din videoclip. Micile probleme vizuale ale modelului Full sunt rezolvate în cazul Heavy, una dintre diferențe fiind detecția trunchiului și a mâinii din fața acestuia, care devia puțin în versiunea anterioară.

Mai mult, putem observa diferența dintre modelele MediaPipe Pose în tabelul 4.2, unde am folosit o imagine și două clipuri video de durată diferită pentru a compara detecțiile obținute. Datele folosite sunt prezentate în figurile de mai sus: yoga.png din figura 4.7 și videoclipul din figura 4.8.

Fișier	Dimensiuni	Durată	FPS	Lite	Full	Heavy
Yoga.jpg	1500 x 1000	-	-	0.30 sec	0.34 sec	0.49 sec
Video1.mp4	960 x 540	17.25 sec	24	12.97 sec	18.48 sec	54.87 sec
Video2.mp4	640 x 360	2.56 sec	25	2.00 sec	2.62 sec	7.27 sec

Tabel 4.2 – Comparație între vitezele de execuție ale modelelor MediaPipe Pose

Pe lângă toate acestea, am efectuat câte un test de acuratețe pentru fiecare versiune a MediaPipe Pose, și am obținut următoarele procentaje de acuratețe: am început cu 84.27% pe versiunea Lite, urmată de 88.33% pe modelul Full și o creștere mică până la 89.84% pe versiunea Heavy. Aceste rezultate sunt reprezentate vizual în figura 4.9, unde rezultatele slabe, de sub 20% acuratețe, care sunt prezente în graficul pentru versiunea Lite, încep să dispară în celelalte două modele.

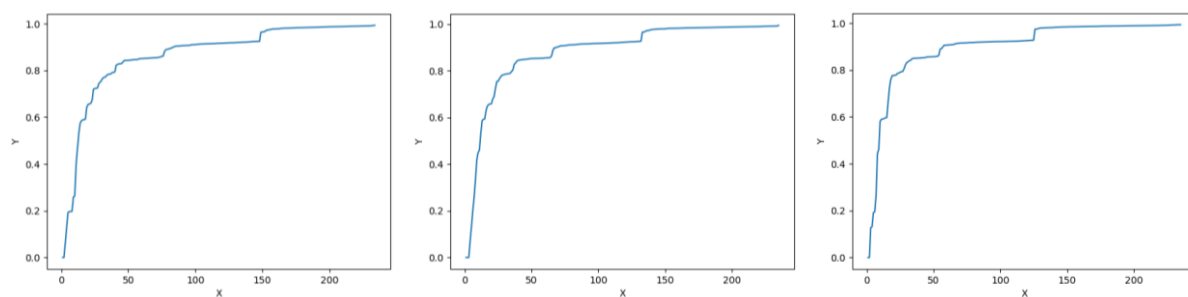


Fig. 4.9 – Acuratețea modelelor MediaPipe Pose

Lite 84.27% (stânga), Full 88.33% (mijloc) și Heavy 89.84% (dreapta)

În concluzie, datorită echilibrului dintre timpul de execuție și acuratețea detecțiilor care se apropie de 90%, am ales să folosesc versiunea Full cu setările standard de minim 50% (0.5) încredere în detectarea și urmărirea scheletului. Acest model a fost folosit în dezvoltarea aplicației web.

Capitolul 5

Prezentarea aplicației web

5.1 Pagina principală

Pagina principală a aplicației web (Fig. 5.1) conține următoarele elemente: o bară de navigare cu logo-ul website-ului și butoanele pentru redirecționarea la celelalte pagini, butoane organizate pe un grid pentru a explica pe scurt fiecare pagină de generare a animațiilor, un link de redirecționare la Github-ul proiectului (acesta va fi postat după prezentarea lucrării de licență) și trimeri la website-urile: Universitatea din București, PyCharm și MediaPipe pentru a mulțumi contribuției lor la finalizarea lucrării.

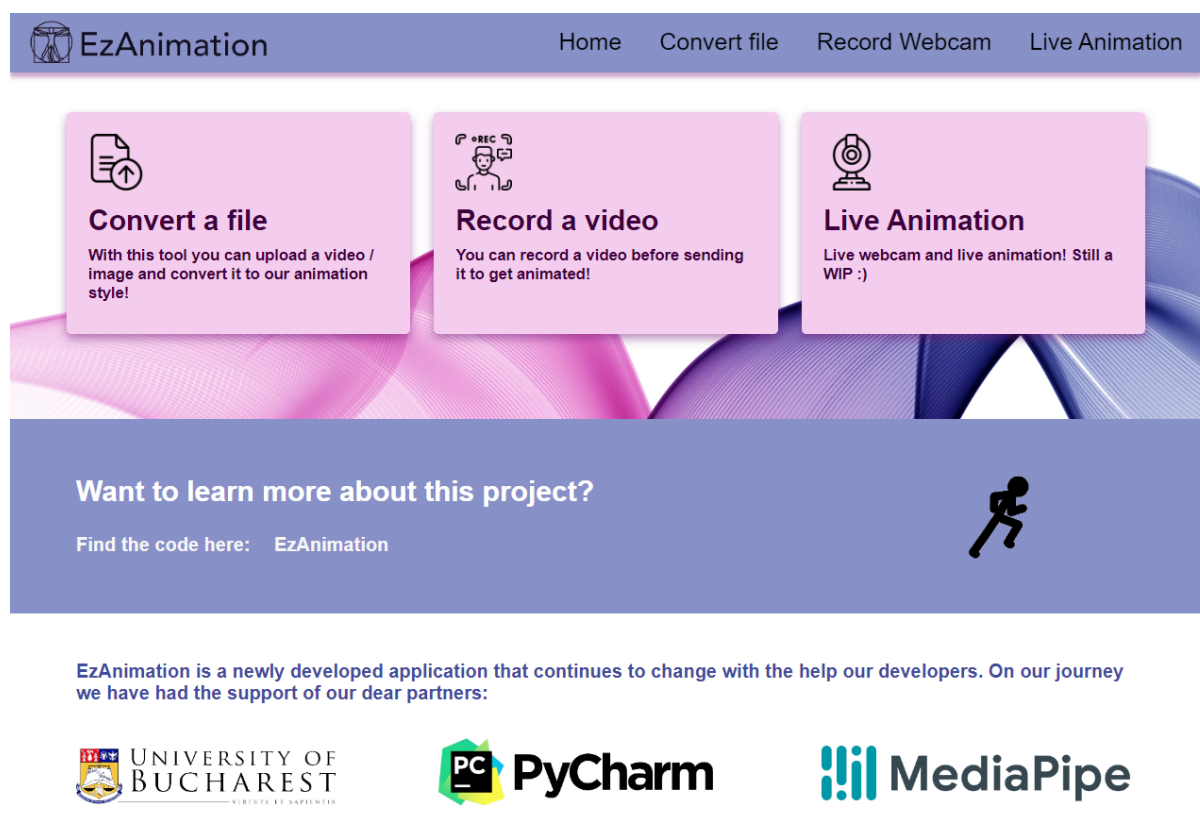


Fig. 5.1 - Pagină principală (folosind meniul principal)

Această pagină a fost concepută pentru a îndruma utilizatorul și a explica conținutul funcționalităților aplicației. Cu scopul de a semăna cu un website oficial, am adăugat elementele din partea de jos a paginii.

Chiar dacă nu sunt necesare pentru folosire corectă a aplicației web, consider că elementele din pagină conferă un spațiu familiar pentru orice persoană care dorește să folosească website-ul, având un aspect plăcut și direcții simple de navigare între pagini.

Bara de navigare este un element comun între toate paginile și include o variantă restrânsă a acesteia care apare automat când fereastra devine prea mică pentru a încadra butoanele meniului original. Pentru a minimiza spațiul ocupat de meniul secundar, am folosit un format dropdown, acesta putând fi accesat printr-un buton (Fig. 5.2).

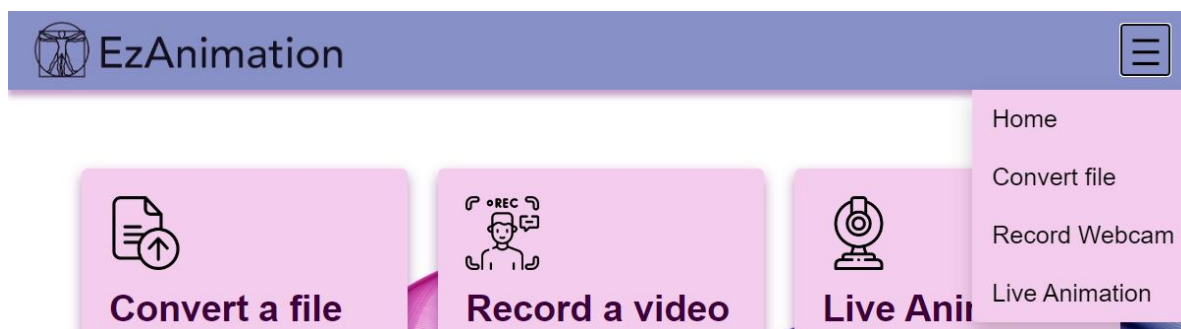


Fig. 5.2 – Pagină principală (folosind meniul restrâns)

Implementarea acestei schimbări utilizează un toggle cu token-ul „show” asupra conținutului meniului secundar (cel mic). Această acțiune este realizată folosind JavaScript. Când condiția de dimensiune este îndeplinită (@media (max-width: 1100px)) și meniul acesta este vizibil, schimb din fișierul index.css vizibilitatea celor două meniuri, făcându-l pe unul să apară și celalalt să dispară.

Cadranele de culoare roz de pe pagina principală reprezintă o alternativă la opțiunile din meniu, având informații adiționale despre cum se utilizează fiecare funcționalitate. Imaginile care reprezintă cele trei pagini au o animație de mărire (zoom) când utilizatorul pune cursorul peste ele (hover).

Partea de jos a paginii nu are funcționalități speciale, fiind simple trimiteri la website-uri externe. Imaginea din cadranul cu redirecționarea la codul sursă, așa cum se poate observa în figura 5.3, este o animație a unui omuleț alergând, aceasta folosind extensia gif.



Fig. 5.3 – Redirecționare la repository-ul de pe Github

5.2 Pagina de încărcare a fișierelor

Prima metodă de creare a animației 2D este încărcarea unui fișier pentru a fi procesat după apăsarea unui buton. Pagina de încărcare a fișierelor (Fig. 5.4) conține un titlu, o descriere scurtă, butoanele *Upload file* pentru a alege fișierul dorit, *Submit* care nu este activ înainte ca utilizatorul să selecteze fișierul necesar și butoanele de personalizare.

Pagina are un design simplu, folosind același stil pentru toate butoanele, de culoarea bării de navigare, și utilizând o casetă parțial transparentă pentru a evidenția și delimita elementele de personalizare a animației.

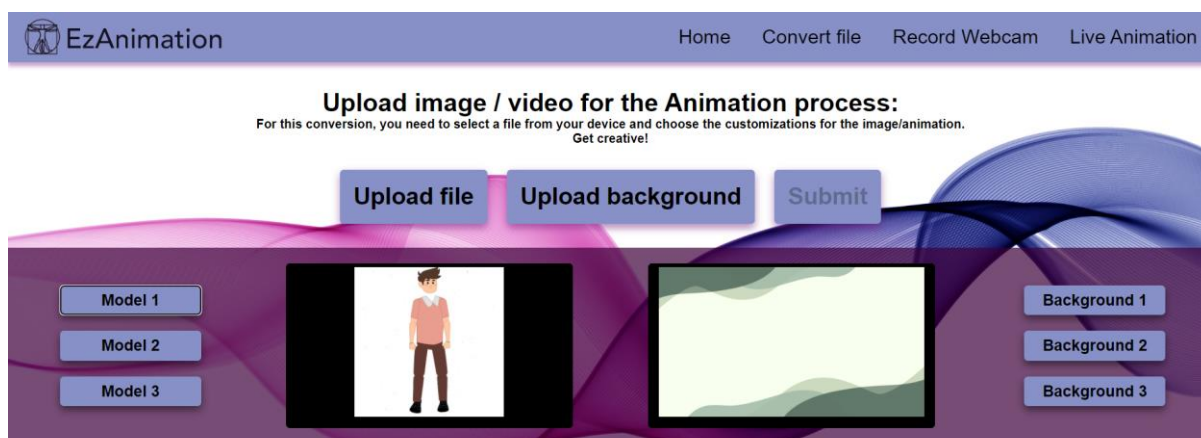


Fig. 5.4 – Pagină de încărcare fișiere

Cu *Upload background* utilizatorul poate încărca o imagine pentru fundalul folosit în animație, iar butoanele pentru model și background oferă opțiunea de a alege modelul 2D utilizat și una dintre imaginile de fundal implicite.

În momentul alegerii unei opțiuni noi, imaginile poziționate între butoanele de selecție se schimbă pentru a reflecta noua configurare. Imaginea de fundal (Fig. 5.5) nu se schimbă în funcție de fundalul încărcat de utilizator, ci doar în funcție de opțiunile oferite pe website.

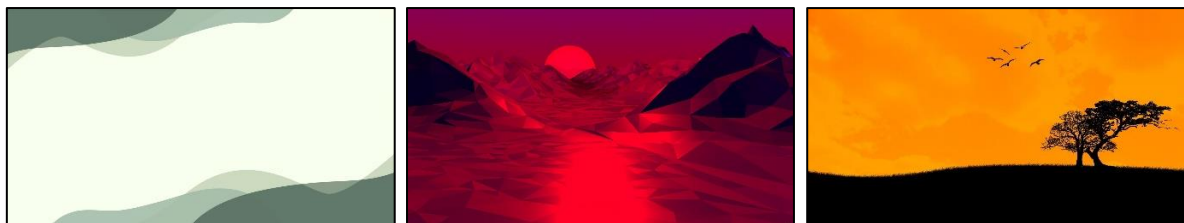


Fig. 5.5 – Imaginile de fundal disponibile pe website

După încărcarea fișierului care trebuie animat și, opțional, a fundalului, pagina menționează fișierele alese (Fig. 5.6). Această afișare se face dinamic în funcție de ce fișier a fost încărcat și se modifică în cazul în care fișierele sunt alese din nou.



Fig. 5.6 – Afișarea fișierelor încărcate

Odată apăsăat buutonul de *Submit*, toate datele strânse de pe acesată pagină sunt trimise printr-un formular la funcția *upload()* care salvează toate variabilele în baza de date, pentru a fi ușor accesibile pentru viitoarele alterări.

Această funcție redirecționează datele la pagina de procesare, unde se pun toate informațiile necesare cap la coadă pentru a apela funcțiile de generare a animației 2D. Înainte de procesare, se ține cont de posibilitatea ca utilizatorul să fi ales două imagini de fundal, una din opțiunile website-ului și una încărcată de acesta. În cazul în care există ambele variabile, imaginea încărcată de utilizator este cea folosită de program.

Timpul de așteptare pentru generarea animației variază în funcție de mărimea fișierului primit. Este direct proporțional cu fps-ul (frames per second), în cazul clipurilor, și dimensiunile imaginilor procesate. Videoclipurile de calitate înaltă durează extrem de mult, pe când imaginile sunt procesate în câteva secunde.

5.3 Pagina de înregistrare video

Funcționalitatea de înregistrare a unui clip video funcționează similar cu încărcarea unui fișier, însă fișierul este rezultatul înregistrării. Pagina aceasta (Fig. 5.7) conține un cadran pentru redarea camerei web pe parcursul înregistrării, aceasta fiind înlocuită de un cadran negru cât timp nu se înregistrează nimic.

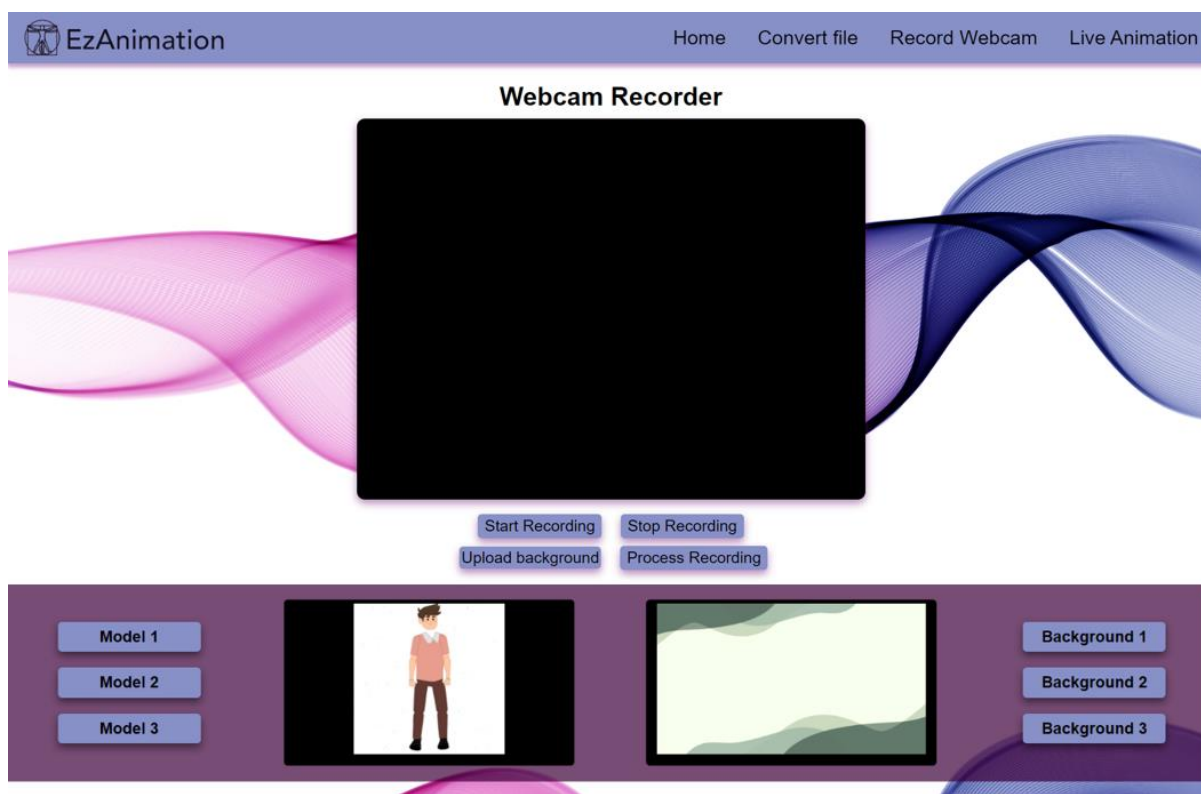


Fig. 5.7 – Pagină de înregistrare a unui clip video

Butoanele *Start Recording*, respectiv *Stop Recording* lucrează împreună pentru a obține datele de intrare necesare pentru procesare. Apelarea funcției *start_recording()* setează variabila în

care va fi salvat videoclipul, iar *stop_recording()* închide procesul de salvare a cadrelor video și salvează ce a fost înregistrat până la momentul respectiv.

În timpul dintre aceste două acțiuni, programul preia datele de la camera web și salvează fiecare cadru. Totodată, cadrul salvat este redirecționat pentru a fi vizibil și pe pagina web. Astfel, utilizatorul are control total asupra duratei înregistrării făcute, putând să vizualizeze înregistrarea pe parcurs și având la îndemână butoanele care încep și opresc videoclipul.

După obținerea clipului video, utilizatorul poate alege să proceseze această înregistrare folosind butonul *Process Recording*. Există și opțiunea de a începe o nouă înregistrare dacă utilizatorul nu este mulțumit de primul rezultat. Alegerea modelului 2D și a fundalului poate fi făcută oricând înainte de apăsarea butonului de procesare. Similar cu pagina anterioară, după apăsarea butonului, se redirecționează la pagina de procesare și, pe urmă, la pagina de descărcare.

5.4 Pagina de animație în timp real

Această pagină, deși similară cu cea prezentată anterior, reprezintă o abordare diferită asupra generării animației 2D. Pe când pagina de înregistrare procesa animația după obținerea datelor de intrare, acum animația se realizează în același timp cu afișarea camerei web.

Pagina de animație în timp real (Fig. 5.8) reprezintă o metodă ambițioasă de procesare și obținere a videoclipului de animație 2D. Am încercat să procesez fiecare cadru în parte înainte de a îl salva, astfel procesarea imaginilor în timpul redării camerei web, rezultă în animația în timp real a utilizatorului.

Dezavantajele acestei metode țin de viteza de execuție a algoritmului utilizat, care încetinește rezultatul și, astfel, generează animații lente, mai deranjante din punct de vedere vizual. Pe de altă parte, avantajul acestei metode este faptul că împreună cu apăsarea butonului de stop, animația este deja gata de descărcat. Procesarea se face pe parcursul înregistrării, ținând cont de timpul de generare a fiecărui cadru pentru a păstra numărul de imagini pe secundă.

La momentul actual, această versiune de animație live nu este perfectă, însă consider că reprezintă o funcționalitate atractivă pentru utilizatori și o experiență interesantă pe parcursul animației, deoarece se poate vedea procesul în timpul execuției sale.

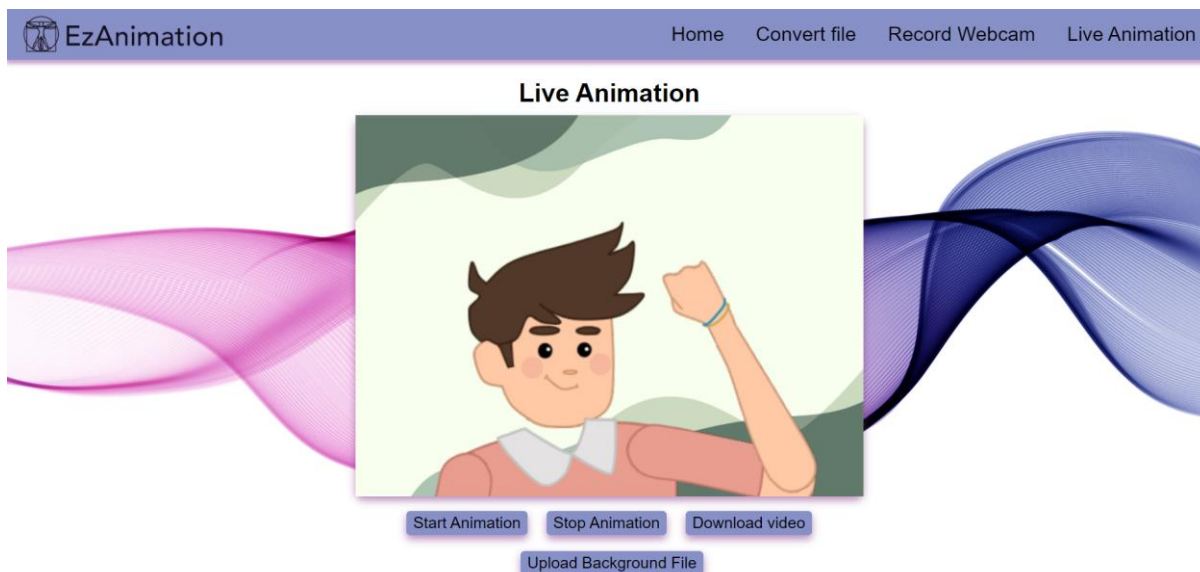


Fig. 5.8 – Pagină de animație în timp real

Pagina aceasta, la fel ca celelalte, beneficiază de diversele modificări asupra fundalului și modelului 2D. Pentru a transmite imaginea de fundal, în cazul încărcării acesteia, am transmis fișierul prin codul de JavaScript. Din acest motiv, au existat diferențe semnificative între dezvoltarea paginii de înregistrare video și aceasta.

Deoarece nu pot obține toate datele înainte de procesare, toate opțiunile alese de utilizator sunt transmise după apăsarea butonului de *Start Animation* și sunt păstrate în acest format în timpul întregii înregistrări. Modificările nu pot fi făcute după începerea animației.

5.5 Pagina de procesare

Pagina de procesare (Fig. 5.9) apelează funcțiile de generare a animației din fișierul *utils.py* în funcție de tipul de fișier de intrare: imagine sau videoclip. Procesarea decurge diferit în funcție de aceste două tipuri de fișier, apelând funcții diferite *animate_video()* sau *animate_frame()*. La rândul lor, acestea utilizează alte funcții pentru a obține varianta animată a fișierului de intrare.

Un aspect importat al aplicației în versiunea sa curentă este faptul că poate procesa numai imagini în format jpg, jpeg și png, iar videoclipurile în format mp4. Clipurile video înregistrare prin funcționalitățile website-ului sunt salvate implicit sub formatul mp4, pentru a nu exista

probleme la procesare. Detaliile despre cum funcționează generarea animației vor fi prezentate în capitolul următor.

Pentru a simula animația de încărcare a rezultatului, am utilizat un fișier gif. Prin comparație cu implementarea unei animații similare folosind HTML/CSS, această variantă a fost simplu de obținut și adăugat în pagină. Timpul de încărcare nu este afectat de fișierele tip gif.

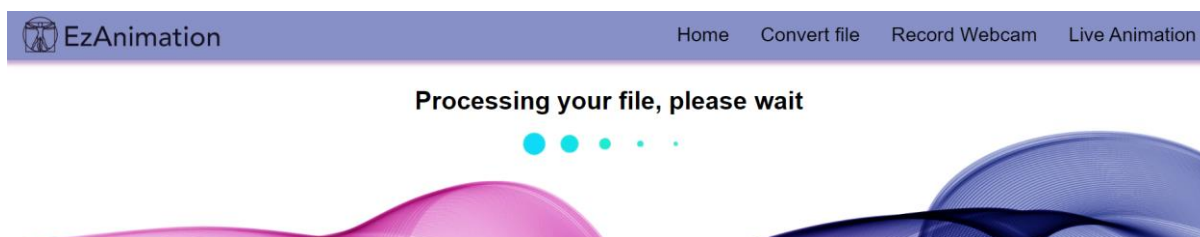


Fig. 5.9– Pagină de procesare / loading

5.6 Pagina de descărcare a rezultatului

Pagina de descărcare, prezentată în figura 5.10, reprezintă finalizarea procesului de animație, după care utilizatorul poate obține rezultatul dorit. Această pagină se folosește de o variabilă de sesiune pentru a extrage id-ul datelor procesate din baza de date. Datorită organizării informațiilor necesare animației în baza de date într-un tabel unic, pagina de descărcare poate extrage rezultatul fără nicio problemă.

Descărcarea fișierelor se face folosind aceeași denumire și extensie ca fișierul încărcat de către utilizator. De exemplu, imagine.jpg va rezulta un fișier cu denumire identică.

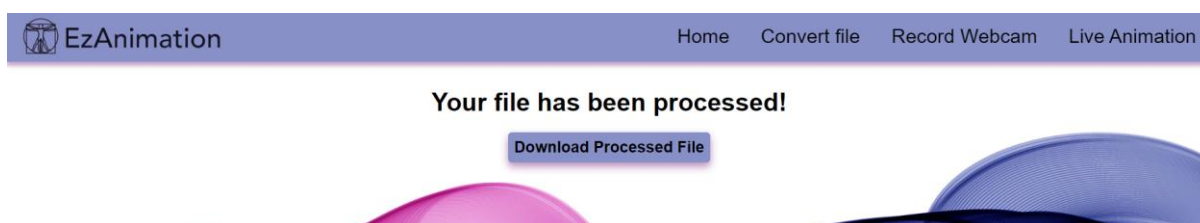


Fig. 5.10 – Pagină de descărcare a animației

Capitolul 6

Prezentarea algoritmului de animație 2D

6.1 Prezentarea generală

Algoritmul folosit pentru generarea animațiilor este unul simplu în teorie și complex în practică. Ideea principală se bazează pe utilizarea MediaPipe Pose pentru a obține coordonatele 2D ale corpului, acestea fiind asociate cu legăturile dintre diferite părți ale corpului, cum ar fi umerii, coatele, încheieturile etc.

Odată obținute aceste date, am încercat să iau în considerare diverse poziții ale corpului uman și schimbările care vin cu acestea. Astfel, am ajuns la următoarea soluție: folosind coordonatele 2D pentru majoritatea punctelor oferite de MediaPipe Pose, am delimitat pentru fiecare piesă a modelului 2D un cadran în care urmează să fie plasată.



Fig. 6.1 – Modelul 2D folosit pentru animație

Pentru a lămurii ce conține modelul 2D (Fig. 6.1), am luat în considerare aceste părți ale corpului: cap, trunchi, brațele formate din două părți, fără animație individuală a mâinilor, picioarele formate din trei părți (coapsa, gamba și laba piciorului). Modelul este asamblat sub o formă similară cu o marionetă, capul nefiind atașat de corp.

Pentru încălțăminte caracterului există o verificare a orientării acestuia în spațiu și, în funcție de rezultat, se determină dacă piciorul este văzut din față sau lateral. Astfel, există două variante de picior în generarea animația 2D, frontal și lateral.

Similar cu cele două variante ale piciorului există și o variantă laterală a trunchiului. Pe lângă toate acestea, sunt utilizate diverse verificări pe parcursul procesării pentru a determina poziția altor piese, precum gradul de înclinare al capului sau dacă acesta se află sub corp, caracterul stând în mâini sau într-o poziție similară.

În ceea ce privește plasarea bucăților modelului în cadranele respective, majoritatea elementelor folosesc o funcție comună pentru a încadra piesa peste fundalul selectat, cu excepția capului care are propria funcție prin care este adăugat în cadru. În funcție de tipul de procesare, după asamblarea imaginii, aceasta este trimisă mai departe sub forma unei imagini sau salvată într-o variabilă pentru a obține un clip video.

Generarea animației împreună cu toate funcțiile necesare aparțin în totalitate de fișierul *utils.py*, pe când salvarea în baza de date și interacțiunea cu utilizatorul sunt separate în fișierul *__init__.py*, unde se folosește biblioteca Flask.

6.2 Funcțiile *animate_frame()* și *animate_video()*

Aceste două funcții reprezintă cele două configurări pentru procesarea unei imagini sau a unui cadru dintr-un video (folosit pentru animația în timp real) sau a unui clip video. Funcțiile reprezintă și legătura cu interfața web, acestea fiind apelate pe pagina de procesare.

Ele inițializează spațiul de lucru și după apelarea funcției principale de generare a animației, memorează rezultatul în formatul dorit, după care returnează acest rezultat funcției de procesare din *__init__.py*.

Ambele funcții primesc argumente similare, *animate_frame()* primește o imagine și *animate_video()* primește un clip video, amândouă folosind un string pentru alegerea modelului 2D și un string sau o imagine pentru imaginea de fundal.

Pentru a mă asigura că modelul și imaginea de fundal folosite sunt cele corecte, le verific și modific la începutul execuției. Astfel, determin path-ul pentru model și verific existența folder-ului respectiv, precum și ce tip de variabilă este fundalul pentru a putea fi obținut fundalul selectat sau pentru a modifica fundalul încărcat de utilizator. În cazul în care există vreo problemă în oricare dintre acești pași, fundalul de bază este o imagine albă, iar modelul 2D folosit este modelului numărul 1, cel prezentat în lucrare în figura 6.1.1.

Aceste funcții apelează MediaPipe pentru a obține setul de landmarks. Prin conversia de la BGR la RGB a cadrului inițial, se apelează *pose.process()*, acesta fiind la rândul său o funcție a modelului pre-antrenat din *mp.solutions.pose.Pose* folosind următoarea configurare:

model_complexity=1, min_detection_confidence=0.5, min_tracking_confidence=0.5

Deoarece nu toate cadrele obținute conțin un om, dar și din cauza eventualității de eșuare a detecției, se verifică dacă se pot extrage rezultatele procesării detecției umane prin *landmarks = results.pose_landmarks.landmark*. În cazul în care variabila *landmarks* nu este creată, deci nu există în *locals()* imaginea rezultată o să fie redusă la imaginea de fundal.

Din cauza utilizării în cazuri diverse a *animate_frame()* nu am putut să setez această funcție pentru a procesa numai imagini. Din acest motiv, nu am modificat setarea *static_image_mode* să fie adevărată. Pentru eficientizarea acestui proces, aş putea adăuga un argument în apelarea funcției care precizează dacă imaginea este separată sau partea dintr-un videoclip.

Printre diferențele dintre cele două funcții, se numără inițializarea diferită a variabilei în care este stocat rezultatul. Pe când *animate_frame()* nu trebuie să folosească decât un array NumPy, *animate_video()* folosește OpenCV pentru a inițializa o captură video cu *cv2.VideoCapture()* și o variabilă de salvare a clipului *cv2.VideoWriter()*. Pentru a putea returna un videoclip înapoi la baza de date pentru a putea fi accesat de utilizator, am utilizat o variabilă temporară în care am salvat animația:

temp_output = tempfile.NamedTemporaryFile(suffix='.mp4', delete=False) (Formula 6.1)

Această variabilă permite transmiterea rezultatului, dar și modificarea acestuia sub forma de bytes necesară pentru adăugarea în baza de date. Conversia în bytes este făcută cu ajutorul *open(temp_output.name, 'rb')*, unde *temp_output.name* este numele variabilei noastre temporare și *rb* este modul de citirea a acesteia.

6.3 Funcția de procesare a animației 2D

Funcția de procesare 2D, *process_frame()*, care este apelată de cele două funcții prezentate anterior, reprezintă pivotul acestei aplicații. Această parte a implementării folosește toate funcțiile ajutoare din *utils.py* pentru determinarea cadranelor fiecărei piese din modelul 2D și pentru a poziționa pieselor în relație cu aceste delimitări.

Pentru a începe generarea animației este nevoie de imaginile care compun modelul 2D. Acestea sunt extrase folosind path-ul generat în funcțiile anterioare și trebuie reîncărcate de fiecare dată când sunt utilizate pentru a asigura calitatea acestora. Piese modelului 2D sunt de fapt imagini png decupate cât mai precis, așa că trebuie folosit *cv2.IMREAD_UNCHANGED* la citirea acestora pentru a asigura fundalul transparent a modelului. Altfel, animația ar fi înconjurată de margini albe care ar acoperi diverse porțiuni ale caracterului.

În timpul procesării, am folosit o metodă de separare a datelor din formatul BGRA pentru a obține culorile și o mască pentru fundalul transparent. Voi prezenta aceste detalii mai târziu în acest subcapitol. Părțile corpului sunt procesate în această ordine: capul, încălțăminte, partea de jos a piciorului, partea de sus a piciorului, trunchiul corpului, partea de sus a brațului și partea de jos a brațului.

În ceea ce privește optimizările procesului de animație, am încercat să utilizez câteva metode prin algoritmi de calcul paralel, însă din cauza structurii programului, a trebuit să fac câteva modificări pentru a putea apela funcții diferite în paralel.

După efectuarea schimbărilor și implementarea unei arhitecturi paralele simple, rezultatele au fost chiar mai lente. În încercarea de a optimiza procesarea imaginilor, am folosit următoarele metode: *multithreading.Process* și *concurrent.futures*, dintre care ambele au dat rezultate mai lente decât programul inițial. Aceste teste au fost efectuate fără utilizarea interfeței web, iar datele obținute pot fi observate în tabelul 6.1.

Fișier	Dimensiuni (fps și timp)	Original	Futures	Process
Image2.jpg	736 x 860	0.68 sec	0.92 sec	0.71 sec
Video2.mp4	260 x 640 (24fps, 2.56sec)	16.62 sec	20.60 sec	18.05 sec

Tabel 6.1 – Comparație între timpurile de procesare ale implementării originale și celor două implementări paralele

6.3.1 Adăugarea capului

Acest proces se face total separat de celelalte părți ale modelului, având propria funcție *add_head()*.

Funcția pentru adăugare a capului își începe procesul prin calcularea centrului feței folosind centrul unui triunghi format din punctul din vârful nasului și punctele pentru fiecare ureche. După determinarea aproximativă a acestui punct, urmează să aproximăm mărimea capului. Din cauza inconsecvenței detecției faciale, mai ales pe date de calitate scăzută, am decis să fac o comparație între distanțele dintre cele trei puncte folosite anterior pentru a decide lățimea capului.

Când distanța dintre urechi este mai mare, aceasta se înmulțește cu 2.0 pentru a obține lățimea, dar când una din celelalte distanțe este mai mare, este înmulțită cu 2.25 pentru un rezultat similar. Această diferență ia în considerare unghiul la care se întoarce capul, motiv pentru care distanța inițială se micșorează între perspectiva frontală și cea din profil.

Acum că există o variabilă pentru lățimea capului, se modifică imaginea png pentru a se încadra în dimensiunile noi. Calculând diferența dintre lățimea actuală a imaginii și variabila obținută, folosesc *cv2.resize()* pentru a redimensiona piesa. Din acest moment, trebuie efectuate câteva verificări. În primul rând, verific dacă coordonatele pentru nas sunt deasupra sau dedesubtul umerilor, dacă măcar unul dintre umeri este mai sus de cap, imaginea png este oglindită pentru a fi cu capul în jos.

În al doilea rând, se calculează gradul de înclinare a capului bazat pe coordonatele urechilor:

$$head_tilt = -1 * round(np.rad2deg(np.arctan((y2 - y1) / (x2 - x1)))) \quad (Formula\ 6.2)$$

Dacă nu poate fi obținut, programul trece mai departe, dar dacă se calculează *head_tilt*, atunci este folosită funcția *rotate_image()* pentru a roti imaginea fără alterarea dimensiunilor acesteia. Voi explica cum funcționează această funcție în subcapitolul *Funcțiile ajutoare*.

Fiindcă acum avem varianta finală a imaginii png pentru cap, putem să calculăm cele patru coordonate în care va fi încadrată. Pornind de la punctul din centrul feței, extindem în cele patru direcții pentru a obține două puncte *start_x*, *start_y*, respectiv *end_x*, *end_y*.

Coordonatele de pe axa Ox sunt modificate egal cu câte jumătate din lățimea capului, pe când pe axa Oy se ține cont de poziționarea capului pentru a lăsa distanță mai mare între cap și restul corpului. Această împărțire se face: 2/3 în direcția opusă trunchiului și 1/3 înspre trunchi.

Astfel, se obține și cadranul necesar. După acest proces trebuie să ne asigurăm că poziția noastră nu este mai mult de jumătate în exteriorul imaginii de fundal și că centrul feței nu se apropie de margini. Verific fiecare dintre cele patru coordonate, împreună cu punctul de centru și fac modificările necesare, decupând din png pentru a nu exista probleme la inserarea acestuia.

Partea interesantă la combinarea unei imagini BGR cu o imagine BGRA este diferența dintre numărul de canale. Soluția pentru această problemă este separarea png-ului în culori (*head_overlay*) și crearea unei măști pentru canalul alpha (*head_alpha_mask*). Cu toate aceste informații, decupăm secțiunea din imaginea de fundal (*img_subsection*) și compunem rezultatul:

$$bg[start_y:end_y, start_x:end_x] = img_subsection * (1 - head_alpha_mask) + head_overlay * head_alpha_mask \quad (Formula\ 6.3)$$

Pentru a demonstra abilitățile poziționării acestei piese, am decis să procesez o poză cu o persoană care stă în mâini, iar rezultatul este prezentat în figura 6.2:



Fig. 6.2 – Testarea poziționării capului asupra unei imagini

(Sursă imagine: Deagreetz, f.a.)

6.3.2 Adăugarea trunchiului

Adăugarea trunchiului reprezintă un proces mult mai simplu decât cel descris în funcția *add_head()*. Astfel, tot ce trebuie făcut pentru adăugarea acestei piese este citirea coordonatelor oferite de MediaPipe Pose și extindere acestora prin folosirea *extend_points()*.

Printre puținele verificări făcute asupra trunchiului se numără și determinarea poziției generale a corpului. Dacă trunchiul este vizibil din lateral, ci nu din față, imaginea png folosită și argumentul folosit pentru modificarea punctelor se schimbă. Pentru a exemplifica aceste piese ale modelului 2D am folosit figura 6.3.

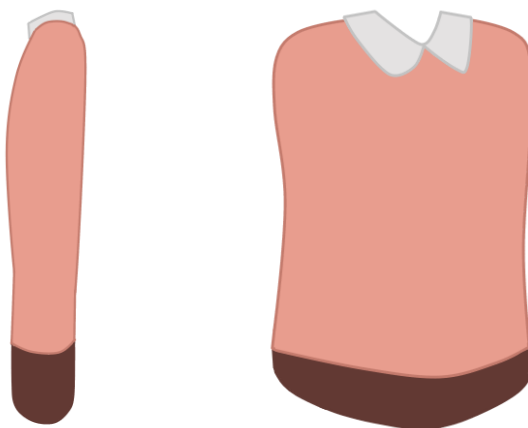


Fig. 6.3 - Diferitele perspective ale trunchiului

O altă alterare a punctelor folosite este extinderea punctelor de la șold pentru a se potrivi cu mărimea picioarelor de care se conectează. Am modificat partea aceasta pentru ca animația să fie mai plăcută din punct de vedere vizual.

Mai mult, pentru a fi sigur că următoarea funcție folosită primește datele corecte, am utilizat funcția *sort_pts()* pentru a ordona coordonatele cadranelor. În interiorul funcției am așezat punctele în așa fel încât să nu se strice așezarea png-ului în interior, punctele de la umeri nu se pot amesteca cu cele de la șold.

Acum că am delimitat zona în care trebuie adăugată imaginea, m-am folosit de funcția *add_png()* pentru a modifica imaginea de fundal și a adăuga piesa curentă.

Funcția aceasta este utilizată de toate piesele în afară de cap și folosește elemente din biblioteca OpenCV, anume *cv2.getPerspectiveTransform()* și *cv2.warpPerspective()*, pentru a modifica imaginea png și a o poziționa corespunzător. După această transformare, se utilizează aceeași metodă de separare a imaginii BGRA pentru a combina imaginea de fundal cu png-ul alterat.

6.3.3 Adăugarea încălțăminteii

Încălțăminteii caracterului 2D, similar cu trunchiul, conține două variante ale aceleiași piese. Perspectiva frontală și cea din profil a piciorului caracterului determină care dintre cele două imagini este folosită (Fig. 6.4). Datorită celor trei puncte ale piciorului (gleznă, călcâi și vârful degetelor), am fost nevoit să creez funcții unice pentru determinarea perspectivei, *face_fronting_foot()* și pentru obținerea cadranelor, *triangle_to_rectangle()*. De altfel, posibilitatea de răsucire a imaginii în funcție de coordonate, adică 8 orientări posibile pentru varianta laterală a piciorului, a creat necesitatea unei funcții de modificarea a imaginii png, *foot_pos()*.



Fig. 6.4 – Diferitele perspective ale piciorului

Datorită relevanței, voi explica funcții specifice piciorului în acest subcapitol. Funcția *face_fronting_foot()* returnează o variabilă de tip bool în funcție de perspectiva la care se află piciorul. Această concluzie este obținută prin compararea distanțelor dintre călcâi și gleznă, respectiv călcâi și degete, cu o valoare primită ca argument. Dacă valoarea a cel puțin una dintre distanțe este mai mică, piciorul este considerat a fi în perspectivă frontală.

Chiar dacă denumirea funcției *triangle_to_rectangle()* explică funcționalitatea acesteia, o să explic pe scurt cum am făcut acest lucru. Considerând cele trei puncte componente ale piciorului, am reprezentat una dintre diagonalele dreptunghiului dorit ca fiind dreapta de la gleznă la degete. Astfel, am calculat punctul opus folosind coordonatele de la călcâi și luând în considerare orientarea acestor puncte.

Modificarea imaginii png a piciorului, în cazul perspectivei laterale, s-a dovedit a fi mult mai dificilă decât am asumat inițial. Astfel, pentru funcția *foot_pos()*, am separat cazurile posibile în două categorii: când glezna e deasupra tălpii și când este dedesubt. Categoriile acestea au fost separate, la rândul lor, în câte patru posibilități de alterare în funcție de poziția călcâiului.

Trebuie menționat că aceste poziții sunt de fapt poziția fiecărui punct după efectuarea ordonării lor cu *sort_pts()*, fiecare corespunzând unui număr de la 0 la 3, unde 0 reprezintă punctul din stânga sus, iar următoarele continuă în sensul acelor de ceasornic. Pentru modificarea propriuzisă a piesei, am utilizat o combinație dintre funcțiile *cv2.rotate* și *cv2.flip*.

În cazul perspectivei frontale, în schimb, nu se apelează această funcție, ci m-am folosit de linia formată de la gleznă la mijlocul dreptei dintre călcâi și degete pentru a crea cadranul. Pentru a obține această delimitare, există funcția *line_to_rectangle()*.

Acum că am obținut imaginea și punctele corespunzătoare, urmează apelarea funcției *add_png()*, la fel ca în cazul celorlalte piese.

6.3.4 Adăugarea membrilor

Membrele corpului au fost cele mai ușor de implementat, datorită formei cilindrice din viața reală. Din acest motiv, forma lor 2D poate fi simplificată într-un dreptunghi din aproximativ orice unghi. Modificat destul, acel dreptunghi ajunge să fie un pătrat și chiar dacă nu este o alternativă perfectă a cercului necesar cilindrului, își face treaba destul de bine.

Problema pe care am întâmpinat-o la implementarea membrilor a fost însă scalarea acestora în funcție de datele de intrare. Cum MediaPipe îmi oferă câte două puncte pentru piesele de la mâini și picioare, nu exista nicio metodă prin care puteam să determin grosimea membrilor.

Astfel am ajuns la concluzia că ar fi ușor să aproximez această variabilă în funcție de lățimea capului calculată mai devreme. Cunoscând câteva aproximări folosite în desene pentru a asigura corectitudinea anatomică, am aproximat grosimea brațului ca fiind jumătate și piciorul fiind fix cât capul.

Desigur că a mai existat o problemă în această proces, deoarece modelele aveau deja o lățime proprie, iar eu încercam să le scalez, nu să le redimensionez. După câteva calcule, considerând proporțiile imaginilor png, am ajuns la următoarele numere: 0.12 pentru brațe, 0.20 pentru partea de jos a piciorului și 0.25 pentru partea de sus a piciorului. Aceste rezultate respectă regula inițială cum că piciorul este dublu în grosime față de brațe.

După obținerea acestor aproximări, procesul a devenit mult mai simplu. Pentru fiecare piesă se extrag cel două coordonate și se folosește funcția *line_to_rectangle()* împreună cu coeficienții de scalare.

Funcția ține cont de orientarea drepte pentru a crea două linii perpendiculare la capetele acesteia. Lungimea acestor linii este determinată de numerele calculate mai devreme, astfel am obținut patru puncte care sunt, de fapt și de drept, coordonatele cadranelor. Pentru a crea toate membrele de o parte și cealaltă a corpului, am folosit *cv2.flip()* pentru a oglindi imaginile png. Într-un final, se apelează funcția *add_png()* pentru fiecare dintre membre.

6.4 Funcțiile ajutătoare

În cele din urmă, funcțiile ajutătoare care nu au fost explicate sunt: *rotate_image()*, *extend_points()*, *add_margins()* și *sort_pts()*.

Începând cu funcția *rotate_image()* aceasta are ca scop rotirea unei imagini fără a schimba dimensiunile conținutului sau a decupa din imagine. Pentru a obține rezultatul dorit, am luat în considerare modul de funcționare a următoarelor funcții din OpenCV: *cv2.copyMakeBorder()*, *cv2.getRotationMatrix2D()* și *cv2.warpAffine()*.

Ideea cu care am venit a fost simplă, adaug o margine transparentă imaginii mele, destul de mare încât atunci când este rotită să nu fie decupată. După calculele mele, marginea ar trebui să fie aproximativ jumătate din latura cea mai lungă a imaginii. Acest număr se obține rapid și este folosit drept variabila *border* în adăugarea marginii:

```
cv.copyMakeBorder(img, top=border, bottom=border, left=border, right=border, borderType=cv.BORDER_CONSTANT, value=[0, 0, 0, 0]) (Formula 6.4)
```

Odată adăugată aceasta, folosesc celelalte două funcții OpenCV pentru a roti imaginea. Unghiul la care trebuie rotit png-ul este, de fapt, variabila *head_tilt* calculată în funcția *add_head()*.

Mai departe, funcția *extend_points()* primește două coordonate *a*, *b* și prelungește dreapta acestora pentru a obține coordonate noi. Practic, calculează și returnează două puncte noi care sunt interpolate liniar din punctele date și care folosesc un coeficient suplimentar, *margin*. În interiorul funcției, m-am folosit de două variabile bazate pe *margin*, *t0* și *t1*, pentru a asigura simetria punctelor noi în jurul punctului de mijloc al segmentului [ab]. Noile coordonate sunt calculate folosind *t0* și *t1* cu formula de interpolare liniară.

Funcția următoare, *add_margins()* apelează funcția *extend_points()* pentru a scala cadrul înainte de adăugarea imaginii png. Modul de funcționare este simplu, se face câte o apelare de

funcție pentru fiecare dintre laturile cadranelor, astfel încât rezultatul final să fie o variantă mărită sau micșorată a acestuia. În procesarea imaginilor, am folosit numai varianta de mărire a datelor.

Sort_pts() sortează un set de patru puncte într-o ordine specifică, astfel încât să fie mai ușor de utilizat ulterior pentru operațiuni de transformare geometrică. Ordinea rezultată după apelarea funcției este: colțul din stânga sus, colțul din dreapta sus, colțul din dreapta jos și colțul din stânga jos.

Această ordine se schimbă puțin deoarece am dorit să păstrez ordinea stabilită la citirea coordonatelor și, din acest motiv, funcția verifică dacă punctele reordonate în colțurile din dreapta jos și stânga jos sunt corect identificate ca puncte inițiale 2 și 3 (și-au păstrat parțial poziționarea). Dacă condiția este îndeplinită, este returnată ordinea nouă, altfel ordinea rămâne cea inițială.

Pentru ordonarea propriu-zisă a punctelor, am folosit funcțiile din NumPy: `np.sum()`, `np.argmax()` și `np.argmin()` pentru a determina de dreapta sus și stânga jos, iar pentru celelalte două am folosit `np.diff()` împreună cu aceleași două funcții.

Capitolul 7

Concluzii

Lucrare aceasta explorează intersecția dintre învățare automată, vedere artificială, dezvoltarea web și animație, în timp ce evidențiază modul în care aceste tehnologii pot fi integrate pentru a crea soluții inovatoare și eficiente. Există câteva puncte care pot fi îmbunătățite și, desigur, există posibilități de cercetare pe viitor.

Prin acest proiect de diplomă, consider că am reușit să prezint o abordare diferită asupra utilizării vederii artificiale în realizarea unor animații 2D și să dezvolt o aplicație web interesantă și creativă din punct de vedere al utilizatorului. De altfel, această lucrare promovează prezentarea tehnicilor din inteligență artificială într-un format simplificat și interactiv pentru a putea fi înțeles de orice tip de persoană. Unul dintre avantajele aplicației este faptul că prezintă un proces complex de transformare a datelor vizuale în animații și îl împachetează într-o interfață ușor de navigat care permite personalizarea rezultatelor.

Ca să abordăm pe rând posibilitățile de continuare a proiectului, consider că aplicația web poate să fie optimizată până în punctul de generare a imaginilor în mai puțin de 0.1 secunde și animațiilor cu aproximativ 12 cadre pe secundă. Acest lucru ar putea fi realizat prin rescrierea programului curent pentru a eficientiza rezultatele și prin reimplementarea procesului de animație folosind o arhitectură paralelă precum un sistem de framework-uri distribuite sau una dintre metodele prezentate în lucrare, multiprocessing și concurrent.futures.

Pe partea de dezvoltarea a interfeței web, cu cât există mai multe opțiuni de personalizare, cu atât mai bine. Permitearea utilizatorilor de a adăuga sau edita modelul 2D, precum și opțiuni mai amuzante precum desenarea fundalului propriu. Legat de design-ul paginilor, există mereu posibilitatea de îmbunătățire a diverselor elemente sau a culorilor alese. Un website mai complex ar putea conține setări precum lumină de noapte sau poate ar exista abilitatea de a îți crea cont și a salva datele procesate într-un dosar digital.

O altă perspectivă de dezvoltare este introducerea unor moduri diferite de animație: implicită, cu expresii faciale sau cu adăugarea degetelor de la mâini. Modificarea animației pentru a fi mai complexă și expresivă este una dintre schimbările care ar ajuta enorm această aplicație.

În ceea ce privește posibilitatea de cercetare în domeniu, combinația dintre artă și inteligență artificială reprezintă un punct de interes pentru multe dintre companiile de animație, acestea utilizând deja unelte bazate pe tehnologii de învățare automată. Posibilitatea de aproximare a unui spațiu 3D din date de intrare 2D este, de altfel, un subiect care merită abordat.

Datorită abilității calculatoarelor de a interpreta lumea într-un mod vizual utilizând vederea artificială și utilizând metode de învățare automată, posibilitățile de dezvoltare în domeniu sunt numeroase. Proiecte de îmbunătățire a sarcinilor necesare în procesul de animație, cum ar fi rotoscopia, animația cadru cu cadru și generarea de efecte speciale, dar și automatizare stilurilor unice de animație care folosesc o combinație dintre 2D și 3D.

Avantajul principal ar fi reducerea timpului prin asistarea artiștilor sau eliminarea acțiunilor repetitive care nu necesită input creativ. Un exemplu bun este automatizarea animației pentru mișcările gurii (lipsync) care necesită atenție și în majoritatea instanțelor nu necesită creativitate.

În concluzie, lucrarea subliniază importanța continuării cercetării și dezvoltării în domeniile abordate pentru a valorifica pe deplin potențialul acestora în îmbunătățirea procesului de animație. Soluția implementată în această lucrare reprezintă o abordare simplă, dar accesibilă prin care utilizatorii se pot familiariza cu diverse concepte în timp ce crează animații unice.

Bibliografie

- *About OpenCV*, (f.a.). OpenCV, accesat în 8 iunie 2024, din <https://opencv.org/about/>
- *AlphaPose*, (2017). Machine Vision and Intelligence Group, Shanghai Jiao Tong University, GitHub, accesat în 14 iunie 2024 din <https://github.com/MVIG-SJTU/AlphaPose>
- *Apple Vision: Detecting Human Body Poses in Images*, (2017). Apple Developers, accesat în 14 iunie 2024, din https://developer.apple.com/documentation/vision/original_objective-c_and_swift_api/detecting_human_body_poses_in_images
- Arnold. L., Rebecchi. S., Chevallier. S., Paugam-Moisy. H., (2011). *An Introduction to Deep Learning*, European Symposium on Artificial Neural Networks (ESANN), accesat în 12 iunie 2024, din <https://hal.science/hal-01352061/file/publishedversion.pdf>
- Bazarevsky. V., Grishchenko. I., Raveendran. K., Zhu. T., Zhang. F., Grundmann. M., (2020). *BlazePose: On-device Real-time Body Pose tracking*, Google Research, accesat în 12 iunie 2024, din <https://arxiv.org/pdf/2006.10204>
- Choi. R.Y., Coyner. A.S., Cramer, J.K., Chiang, M.F., Campbell, J.P., (2020). *Introduction to Machine Learning, Neural Networks, and Deep Learning*, Translational Vision Science & Technology, accesat în 10 iunie 2024, din <https://tvst.arvojournals.org/article.aspx?articleid=2762344>
- *DB Browser for SQLite*, (f.a.). DB Browser for SQLite, accesat în 11 iunie 2024, din <https://sqlitebrowser.org/>
- Deagreez, (f.a.). *Full length photo of funky funny guy*, Adobe Stock, accesat în 10 iunie 2024, din https://stock.adobe.com/ro/images/full-length-photo-of-funky-funny-guy-dressed-orange-sweater-dancing-walking-arms-isolated-turquoise-color-background/507075753?prev_url=detail
- Deery, M., (2023). *What Is Flask and How Do Developers Use It? A Quick Guide*, CareerFoundry, accesat în 7 iunie 2024, din <https://careerfoundry.com/en/blog/web-development/what-is-flask/>
- *DensePose*, (f.a.). DensePose, GitHub, accesat în 10 iunie 2024, din <https://github.com/facebookresearch/DensePose/tree/main>
- Goldstein. B., (f.a.). *Yoga poses*, Verywell Fit, accesat în 12 iunie 2024, din <https://www.verywellfit.com/essential-yoga-poses-for-beginners-3566747>

- Khan, S., (2023). *Benefits of using CSS and HTML in Web Development*, Medium, accesat în 9 iunie 2024, din <https://medium.com/@sumerak819/benefits-of-using-css-and-html-in-web-development-addd90365c2c>
- Laskowski, N.,Tucci. L., (2024). *What is artificial intelligence (AI)? Everything you need to know*, TechTarget, accesat în 7 iunie 2024, din <https://www.techtarget.com/searchenterpriseai/definition/AI-Artificial-Intelligence>
- *MediaPipe Pose*, (2019). MediaPipe, GitHub, accesat în 9 iunie 2024, din <https://github.com/google-ai-edge/mediapipe/blob/master/docs/solutions/pose.md>
- *MediaPipe Solutions guide*, (f.a.). MediaPipe, accesat în 7 iunie 2024, din <https://ai.google.dev/edge/mediapipe/solutions/guide>
- *Meet Django*, (f.a.). Django, accesat în 7 iunie 2024, din <https://www.djangoproject.com/>
- Mihajlovic, I., (2019). *Everything You Ever Wanted To Know About Computer Vision*, Towards Data Science, accesat în 7 iunie 2024, din <https://towardsdatascience.com/everything-you-ever-wanted-to-know-about-computer-vision-heres-a-look-why-it-s-so-awesome-e8a58dfb641e>
- *MoveNet: Ultra fast and accurate pose detection model*, (2021). TensorFlow, accesat în 13 iunie 2024, din <https://www.tensorflow.org/hub/tutorials/movenet>
- O'Shea. K., Ryan Nash. R., (2015). *An Introduction to Convolutional Neural Networks*, ArXiv, accesat în 14 iunie 2024, din <https://arxiv.org/pdf/1511.08458>
- *OpenPose*, (2017). OpenPose, GitHub, accesat în 11 iunie 2024, din <https://github.com/CMU-Perceptual-Computing-Lab/openpose>
- Pikisuperstar, (f.a.). *Cartoon character for motion design*, FreePik, accesat în 26 mai 2024, din https://www.freepik.com/free-vector/cartoon-character-motion-design_4221448.htm
- *Pose landmark detection guide*, (f.a.). Google AI for Developers, accesat în 12 iunie 2024, din https://ai.google.dev/edge/mediapipe/solutions/vision/pose_landmarker
- *PoseNet: Pose Estimation*, (2018). PoseNet, GitHub, accesat în 10 iunie 2024, din <https://github.com/dusty-nv/jetson-inference/blob/master/docs/posenet.md>
- *Quick start guide*, (2024). JetBrains, accesat în 9 iunie 2024, din <https://www.jetbrains.com/help/pycharm/quick-start-guide.html>
- Sapaka, (f.a.) *Rendering - Jeans Background*, CleanPNG, accesat în 6 iunie 2024, din <https://www.cleanpng.com/png-rendering-computer-graphics-texture-mapping-1764108/>

- Tankilevitch. P., (f.a.). *A Woman Showing Her Ballet Skill In Turning One Footed*, Pexels, accesat în 6 iunie 2024, din <https://www.pexels.com/video/a-woman-showing-her-ballet-skill-in-turning-one-footed-5385885/>
- *The Python SQL Toolkit and Object Relational Mapper*, (f.a.). SQLAlchemy, accesat în 8 iunie 2024, din <https://www.sqlalchemy.org/>
- *The Role of Database in Web Application Development*, (2022). Ramotion, accesat în 11 iunie 2024, din <https://www.ramotion.com/blog/database-in-web-app-development/#section-why-do-web-app-developers-need-a-database>
- *The State of Developer Ecosystem 2022*, (2022). JetBrains, accesat în 8 iunie 2024, din <https://www.jetbrains.com/lp/devecosystem-2022/python/>
- Toit. J., (2023). *Human pose dataset (sit & stand pose classes)*, DaYta, accesat în 15 iunie 2024, din https://dayta.nwu.ac.za/articles/dataset/Human_pose_dataset_sit_stand_pose_classes_/23290937?file=41049788
- Tuama, D.Ó., (f.a.). *Advantages of JavaScript*, Code Institute, accesat în 9 iunie 2024, din <https://codeinstitute.net/global/blog/advantages-of-javascript/>
- Vailshery, L.S., (2024). *Most widely utilized programming languages among developers worldwide 2023*, Statistica, accesat în 8 iunie 2024, din <https://www.statista.com/statistics/793628/worldwide-developer-survey-most-used-languages/>
- Vishnu, J.G., Divya. S.J., (2023). *A Comparative Study of Human Pose Estimation*, International Journal of Innovative Research in Science, Engineering and Technology (IJIRSET), accesat în 11 iunie 2024, din https://www.researchgate.net/profile/Jban-Dulan/publication/366956503_A_Comparative_Study_of_Human_Pose_Estimation/links/63bad94f03aad5368e75ac17/A-Comparative-Study-of-Human-Pose-Estimation.pdf
- *What is NumPy?*, (f.a.). NumPy, accesat în 8 iunie 2024, din <https://numpy.org/doc/stable/user/whatisnumpy.html>
- *What is Python? Its uses and applications.*, (2024). GeeksforGeeks, accesat în 8 iunie 2024, din <https://www.geeksforgeeks.org/what-is-python/>
- Xu. H., Bazavan, E.G., Zafir. A., Freeman. W.T., Sukthankar. R., Sminchisescu. C., (2020). *GHUM & GHUML: Generative 3D Human Shape and Articulated Pose Models*, Google Research, accesat în 12 iunie 2024, din https://openaccess.thecvf.com/content_CVPR_2020/papers/Xu_GHUM_GHUML_Generative_3D_Human_Shape_and_Articulated_Pose_CVPR_2020_paper.pdf

(Acest document se leagă împreună cu lucrarea de diplomă)

Declarație

Subsemnatul/Subsemnata IANCU FLORENTINA-MIHAELA, candidat la examenul de diplomă, sesiunea IUNIE - IULIE 2024, la Facultatea de Matematică și Informatică, programul de studii Tehnologia informației, declar pe propria răspundere că lucrarea de față este rezultatul muncii mele, pe baza cercetărilor mele și pe baza informațiilor obținute din surse care au fost citate și indicate, conform normelor etice, în note și în bibliografie. Declar că nu am folosit în mod tacit sau ilegal munca altora și că nici o parte din teză nu încalcă drepturile de proprietate intelectuală ale altcuiva, persoană fizică sau juridică. Declar că lucrarea nu a mai fost prezentată sub această formă vreunei instituții de învățământ superior în vederea obținerii unui grad sau titlu științific ori didactic.

Data

15.06.2024

Semnătura

A handwritten signature in black ink, appearing to read 'Iancu', is written over a horizontal line.