

Universitatea "Politehnica" din București
Facultatea de Electronică, Telecomunicații și Tehnologia Informației

Sinteza de imagini pentru îmbunătățirea estimării posturii capului

Proiect de diplomă

prezentat ca cerință parțială pentru obținerea titlului de *Inginer* în domeniul *Inginerie electronică, telecomunicații și tehnologii informaționale* programul de studii de licență
Rețele și software de telecomunicații

Conducător științific
Conf. Dr. Ing. Corneliu-Nicolae Florea

Absolvent
Ungureanu Andrei

Anul 2019

TEMA PROIECTULUI DE DIPLOMĂ
a studentului **UNGUREANU D. Andrei , 443D**

1. Titlul temei: Sinteza de imagini pentru imbunatatirea estimarii posturii capului

2. Descrierea contribuției originale a studentului (în afara părții de documentare) și specificații de proiectare:

Se construiește o soluție automată pentru estimarea unghiului de inclinare a capului (postura) în imagini sau secvențe video cu potrete. Lucrarea are două componente. O prima componentă se referă la estimarea propriu-zisă. Aici se descriu imaginile cu Modelul Local Binar (LBP), iar decizia se ia cu ajutorul Masinilor cu Vector Suport (SVM). Calitatea soluției este analizată pe baza de date UPNA care conține diverse posturi. Această componentă se implementează în Python. O a doua componentă se referă la cantitatea insuficientă de date adnotate și presupune sinteza de imagini cu ajutorul programului ZBRUSH. Aici, se încarcă un model generic al capului, peste el se suprapune o imagine frontală și prin rotații programate se generează imagini cu posturi arbitrare. Aceste imagini se vor folosi pentru a crește calitatea primei componente.

3. Resurse folosite la dezvoltarea proiectului:

Python, OpenCv, ZBrush, baza de date UPNA

4. Proiectul se bazează pe cunoștințe dobândite în principal la următoarele 3-4 discipline:

PC, SDA, DEPI

5. Proprietatea intelectuală asupra proiectului aparține: U.P.B.

6. Data înregistrării temei: 2019-02-08 16:03:21

Conducător(i) lucrare,

Conf. dr. ing. Corneliu FLOREA

semnătura:

Director departament,

Conf. dr. ing. Eduard POPOVICI

semnătura:

Student,

semnătura:

Decan,

Prof. dr. ing. Cristian NEGRESCU

semnătura:

Cod Validare: **21a55c58f9**

Declarație de onestitate academică

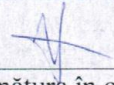
Prin prezenta declar că lucrarea cu titlul "*Sinteza de imagini pentru îmbunătățirea estimării posturii capului*", prezentată în cadrul Facultății de Electronică, Telecomunicații și Tehnologia Informației a Universității "Politehnica" din București ca cerință parțială pentru obținerea titlului de *Inginer* în domeniul *Inginerie electronică, telecomunicații și tehnologii informaționale*, programul de studii *Rețele și software de telecomunicații* este scrisă de mine și nu a mai fost prezentată niciodată la o facultate sau instituție de învățământ superior din țară sau străinătate.

Declar că toate sursele utilizate, inclusiv cele de pe Internet, sunt indicate în lucrare, ca referințe bibliografice. Fragmentele de text din alte surse, reproduse exact, chiar și în traducere proprie din altă limbă, sunt scrise între ghilimele și fac referință la sursă. Reformularea în cuvinte proprii a textelor scrise de către alți autori face referință la sursă. Înțeleg că plagiatul constituie infracțiune și se sancționează conform legilor în vigoare.

Declar că toate rezultatele simulărilor, experimentelor și măsurărilor pe care le prezint ca fiind făcute de mine, precum și metodele prin care au fost obținute, sunt reale și provin din respectivele simulări, experimente și măsurători. Înțeleg că falsificarea datelor și rezultatelor constituie fraudă și se sancționează conform regulamentelor în vigoare.

București, 05/09/2019

Absolvent Andrei UNGUREANU


(semnătură în original)

Cuprins

Lista figurilor	9
Lista acronimelor	11
Introducere	13
Capitolul 1. Concepte relevante și limbaje de programare utilizate	14
1.1 Programul Zbrush și limbajul de scriptare Zscript	15
1.2 Limbajul de programare Python	16
Capitolul 2. Selecția, pregătirea și sinteza imaginilor în Zbrush	17
2.1 Selecția punctelor de interes din imagini	18
2.2 Interfața programului Zbrush și procesul de sinteză	20
2.2.1 Interfața Zbrush	20
2.2.2 Scriptul Zbrush	21
Capitolul 3. Transformarea secvențelor video în baze de date	26
Capitolul 4. Procesul de învățare automată și detecție facială	28
4.1 Modelul local binar și algoritmul de funcționare	28
4.2 Mașinile cu vectori suport și regresie	31
4.3 Implementarea scriptului Python	34
Capitolul 5. Rezultate și concluzii	36
5.1 Concluzii sinteză	36
5.2 Concluzii estimare	38
Bibliografie	41

Lista figurilor

Fig 1.1 Axele de rotatie ale capului	14
Fig 1.2 Exemple de senzori folosiți pentru realizarea bazei de date UPNA.....	15
Fig 1.3 Metodă de realizare a imaginilor folosind markere [15]	15
Fig 1.4 Organigrama lucrării	17
Fig 2.1 Exemple de imagini originale	18
Fig 2.2 Poziția și ordinea punctelor de interes folosind biblioteca dlib [6]	18
Fig 2.3 Exemplu coordonate puncte de interes.....	19
Fig 2.4 Interfața programului Zbrush	20
Fig 2.5 Coordonatele mouse-ului în Zbrush	20
Fig 2.6 Meniul modifiers.....	21
Fig 2.7 Încărcarea scriptului Zbrush.....	21
Fig 2.8 Organigrama procesului de sinteză în Zbrush.....	25
Fig 3.1 Trecerea din format video în imagini si organizarea în dosare.....	27
Fig 3.2 Exemple de imagini sintetice generate	27
Fig 4.1 Comparăția vecinilor în procesul LBP [8]	28
Fig 4.2 Generarea codurilor LBP [8].....	29
Fig 4.3 Conversia imaginii în matricea locală binară [8]	29
Fig 4.4 Comparăție între imaginea originală și matricea LBP formată [8].....	30
Fig 4.5 Hiperplan în cazul reprezentării datelor în 2D și 3D [9]	31
Fig 4.6 Marginea de separăție dintre două clase [17]	32
Fig 4.7 Influența parametrului gamma asupra delimitării claselor [9].....	33
Fig 4.8 Reprezentare a pragului de decizie pentru regresie [14]	33
Fig 4.9 Procesul de creare a imaginii LBP	34
Fig 4.10 Histograma LBP normalizată.....	34
Fig 4.11 Organigrama procesului de învățare	36
Fig 5.1 Exemplu de rotație folosind două axe	37
Fig 5.2 Rezultate finale	38
Fig 5.3 Diferențe între imaginea originală și cea sintetizată	39
Fig 5.4 Erori și imperfecțiuni apărute în procesul de sinteză	40

Lista acronimelor

LBP – Local binary pattern (model local binar)

ML – Machine learning (învățare automată)

MSE - Mean squared error (eroare pătratică medie)

SVM – Support vector machine (mașini cu vector suport)

SVR – Support vector regression (regresie cu vector suport)

UI – User interface (interfața cu utilizatorul)

UPNA - Universidad Pública de Navarra

Introducere

În domeniul inteligenței artificiale, învățarea automată este un proces foarte important pentru dezvoltarea multor aplicații care ușurează interacțiunea om-mașină. Recunoașterea facială este, în prezent, larg utilizată în domenii precum securitate, transporturi sau domeniul medical.

Un algoritm de învățare automată supervizat primește de la utilizator perechi de date împreună cu o etichetă ce reprezintă rezultatul corect, astfel încât să înțeleagă cum se asociază datele de intrare cu cele de ieșire, și să ofere rezultate cât mai corecte.

Așadar, învățarea automată supervizată are nevoie de perechi de date de intrare, numite și lot de antrenare, cât mai corecte. În cazul analizei expresiilor faciale, este nevoie de o imagine cu expresia facială a unei persoane simțind o anumită emoție, împreună cu o etichetă care să clasifice acea emoție. În cazul acestei lucrări, este nevoie de un lot de antrenare complex, fiecare imagine având ca etichetă un unghi precis de înclinare al capului. În practică, obținerea unui astfel de lot de antrenare este dificilă, fiind nevoie de multe imagini având ca etichetă unghiul exact de înclinare care se pot obține folosind niște procese ineficiente din punct de vedere al timpului sau al costului. Astfel, rezultă baze de date nebalansate, care conduc la rezultate eronate.

Contribuții personale

Scopul lucrării este de a propune o metodă de creare a unei baze de date balansate, pentru estimarea unghiului de înclinare a capului, împreună cu unghiul corespunzător. Apoi se va verifica dacă această metodă oferă rezultate mai bune decât dacă se folosesc imagini din baza de date UPNA, care conține diverse posturi. Această bază este formată din secvențe video cu utilizatori ce fac diferite mișcări, iar unghiurile de rotație precum și poziția capului sunt salvate într-un fișier. Așadar trebuie extrase imaginile din baza de date și etichetate în funcție de unghiul de rotație corespunzător.

Lucrarea este împărțită în mai multe capitole, în fiecare fiind descris în detaliu principiile de funcționare, limbajele de programare utilizate, dificultăți care au apărut la implementare și concluzii.

În primul capitol se prezintă programele și limbajele de programare utilizate, motivele utilizării lor, și o imagine generală a lucrării.

În cel de-al doilea capitol se descrie procesul de sinteză a imaginilor și obținerea secvențelor video din care se vor extrage imaginile finale.

În capitolul trei se ilustrează extragerea propriu-zisă a imaginilor din secvențele video, ordonarea lor și pregătirea pentru a fi transformate în date de intrare pentru algoritmul de învățare automată.

În capitolul patru sunt descrise modelul local binar (LBP) și mașinile cu vector suport (SVM) care se folosesc pentru a lua decizii în cadrul învățării automate. Este prezentat algoritmul de funcționare a întregului program.

În ultimul capitol sunt prezentate rezultatele și concluziile lucrării, ilustrând metode de îmbunătățire a rezultatelor și a timpilor de procesare în viitor.

Capitolul 1. Concepte relevante și limbaje de programare utilizate

Scopul lucrării a fost crearea unui flux de lucru eficient pentru aplicațiile de detecție automată a unghiului de rotație a posturii capului care au baze de date incomplete sau nebalansate. Soluția se folosește de un subgrup de imagini din orice bază de date, în cazul nostru baza de date UPNA, pentru a genera imagini sintetice la orice unghi de rotație.

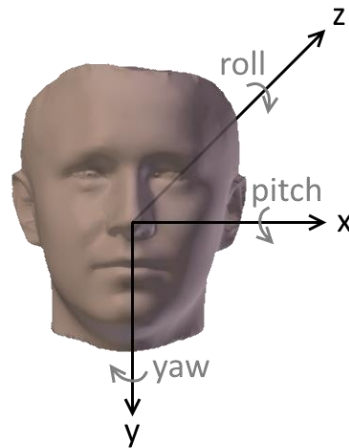


Fig 1.1 Axele de rotație ale capului

În acest stadiu al lucrării m-am concentrat asupra rotației de yaw, adică în jurul axei Y. O rotație care presupune două axe ar genera mult prea multe imagini și reprezintă o problemă pentru un sistem de calcul nespecializat precum cel utilizat. Metode de îmbunătățire vor fi elaborate într-un capitol ulterior.

În prezent, metodele pentru a realiza o astfel de bază de date sunt foarte ineficiente din punct de vedere al timpului de realizare și al costului. Pentru a realiza o bază de date precum UPNA, persoanele cărora le sunt făcute pozele trebuie să miște capul la unghiuri exacte. Acest lucru se realizează cu un senzor atașat de capul persoanei care ar determina unghiul de rotație. Acest senzor nu este foarte precis și este relativ scump. De asemenea se mai poate folosi o cameră mobilă ce se rotește în jurul subiectului, însă acest lucru este de asemenea scump și ineficient din punct de vedere al spațiului pe care îl ocupă.



Fig 1.2 Exemple de senzori folosiți pentru realizarea bazei de date UPNA

O altă metodă pentru realizarea unei astfel de baze de date este folosirea unei camere fixe, iar subiectul își îndreaptă privirea către markere plasate în încăperea în care se află. Această metodă nu este foarte exactă și pot apărea erori de etichetare.

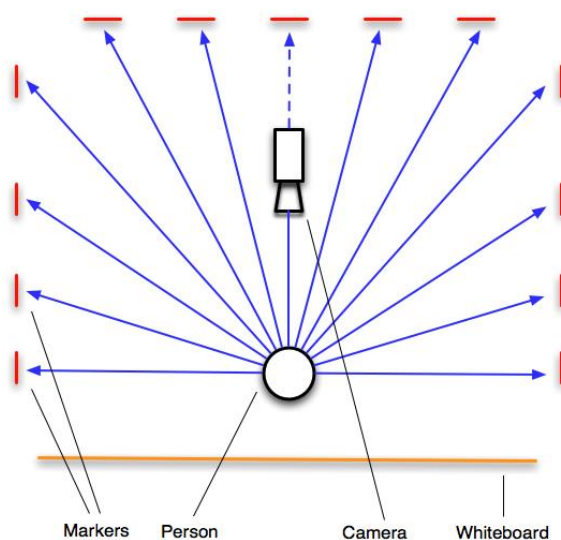


Fig 1.3 Metodă de realizare a imaginilor folosind markere [15]

1.1 Programul Zbrush și limbajul de scriptare Zscript

În aceasta faza am ales folosirea programului Zbrush. Acesta este un program de modelare 3D, dar care are și capabilități de texturare și automatizare prin script-uri în limbajul specific Zscript, facilități care au fost indispensabile pentru realizarea lucrării.

Programul a fost realizat de către compania Pixologic. Prima versiune a apărut în anul 2003, iar versiunea folosită pentru realizarea aplicației este 4R8 apărută în 2019.

Limbajul Zscript este un limbaj de programare funcțional, însemnând ca se bazează pe evaluarea funcțiilor matematice în defavoarea folosirii claselor și a instanțelor de obiecte folosite în programarea orientată pe obiect. Așadar, Zscript folosește variabile și funcții predefinite pentru a efectua operațiuni ce pot fi făcute, în mod normal, manual de către utilizator.

De asemenea, se pot reține valori în variabile, și chiar și scrierea și citirea din fișier sunt posibile. Programul Zbrush a fost ales în favoarea altor programe de modelare 3D deoarece este acoperită întreaga funcționalitate dorită în realizarea aplicației, și anume: texturarea unei poze 2D peste un model 3D, automatizare relativ simplă cu un limbaj de programare propriu care nu are nevoie de un mediu specific de dezvoltare și poate fi realizat în orice editor text, rotirea modelului la unghiuri exacte și exportarea secvențelor video a acestui model care se rotește la unghiuri bine definite și cunoscute de utilizator.

Documentația pentru limbajul Zscript este valabilă pe pagina oficială a programului Zbrush , împreună cu exemple de cod folosite pentru diferite aplicații.

1.2 Limbajul de programare Python

Cea de-a doua parte a lucrării, respectiv crearea bazei de date ordonate în funcție de imaginea originală din care provin imaginile sintetizate, precum și unghiul de rotație, eliminarea imaginilor neimportante pentru procesul de învățare, imagini în care nu este vizibilă fața utilizatorului, ci numai partea din spate a capului, transformarea imaginilor în vectori binari locali, crearea clasificatorului și introducerea datelor, a fost realizată în totalitate în limbajul Python.

Am ales acest limbaj de programare deoarece oferă cele mai multe funcționalități în procesarea imaginilor precum și biblioteci specifice pentru învățarea automată. Cele mai importante biblioteci din această lucrare sunt **sklearn**, care conține funcții pentru crearea mașinilor cu vectori suport folosind descriptorii binari locali, **numpy**, o bibliotecă ce ușurează semnificativ lucrul cu vectori și matrice și **OpenCV**.

OpenCV (Open Source Computer Vision Library) este o librărie care conține funcții specializate pe procesare de imagine și învățare automată. În această lucrare este folosită pentru a extrage cadre din secvențe video, procesarea acestor cadre pentru a putea fi folosite ca date de intrare pentru un model de învățare și modificarea lor pentru a ușura încărcarea de memorie a algoritmului.

Deși la baza sa, Python este un limbaj de programare orientat pe obiect, este adesea folosit în mod procedural datorită vitezei cu care se pot scrie și rula script-uri simple, precum cele de preprocesare a imaginilor și clasificare folosite în această lucrare.

Pentru a testa eficiența metodei propuse se folosește un model de învățare automată supervizat cu vectori suport. Datele de intrare pentru acest algoritm sunt imaginile obținute anterior, transformate cu ajutorul modelului local binar, care este un descriptor vizual folosit pentru clasificare. Comparția se face între rezultatele obținute cu metoda propusă și rezultate obținute folosind imagini originale din baza de date UPNA. Toate aceste funcții sunt prezente în bibliotecile prezentate anterior.

De asemenea, înainte de întregul proces de sinteză a imaginilor, este nevoie de un script Python pentru a localiza puncte de interes de pe față, puncte care vor fi încărcate în Zbrush. Scriptul utilizat folosește biblioteca **dlib** existentă în Python.

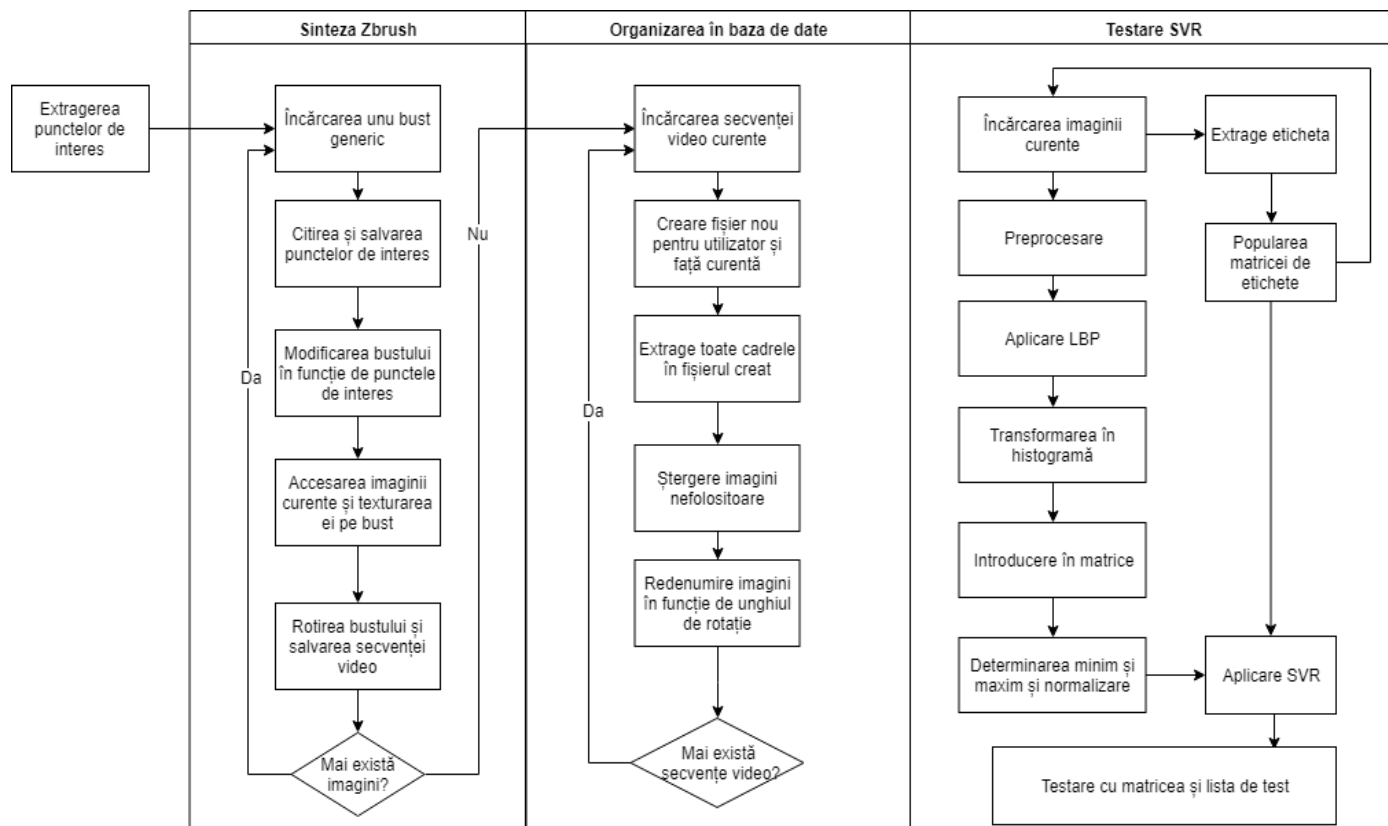


Fig 1.4 Organigrama lucrării

Capitolul 2. Selecția, pregătirea și sinteza imaginilor în Zbrush

Lucrarea folosește patruzeci de imagini extrase din baza de date UPNA. Toate imaginile sunt frontale, câte patru imagini ale aceiași persoane, fiind zece persoane diferite. Această distribuție de poze este luată în așa fel încât datele să fie destul de dispersate pentru a atinge o plajă cât mai largă, dar totuși suficient de specifică pentru sarcina dorită. De asemenea, dintr-un număr mare de poze inițiale ar rezulta foarte multe imagini la ieșirea procesului de sinteză, ceea ce ar fi un procedeu complex, inefficient de gestionat.

2.1 Selecția punctelor de interes din imagini



Fig 2.1 Exemple de imagini originale

Fiecare imagine este analizată de un script python care extrage șaiszeci și opt de coordonate ale punctelor de interes ale feței. Aceste puncte reprezintă poziția ochilor, a sprâncenelor, a nasului, a gurii și conturul feței.

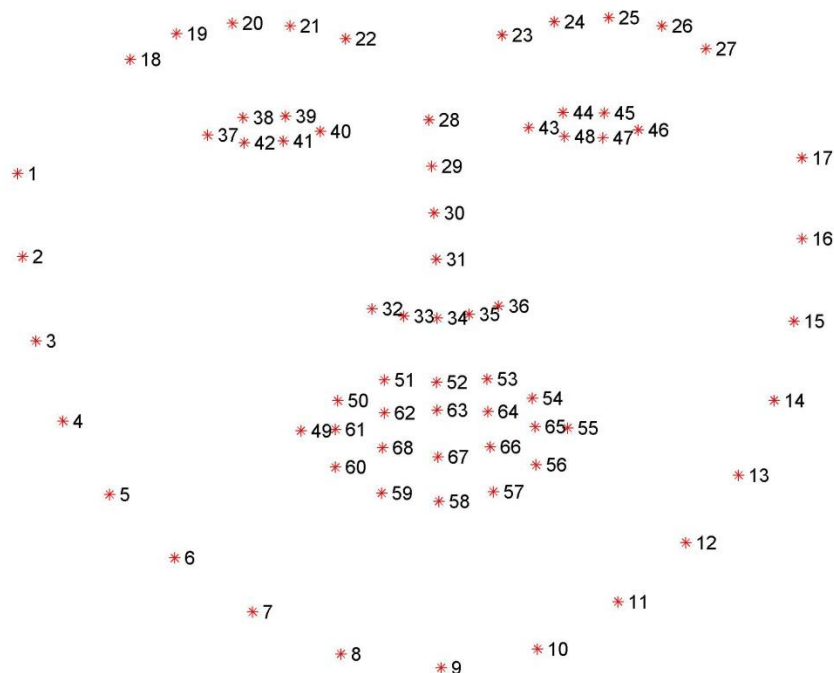


Fig 2.2 Poziția și ordinea punctelor de interes folosind biblioteca dlib [6]

Modul în care funcțiile bibliotecii dlib estimează punctele de interes este următorul, în imaginea dată se caută locația în care este cea mai mare probabilitate să existe o față a unei persoane. În acea arie sunt evaluați toți pixelii, mai exact diferențele de intensitate dintre aceștia. De aceea, pentru rezultate mai bune, este indicat ca înainte de această procesare, imaginea de intrare să fie transformată în grayscale, pentru ca diferențele să fie mai evidente.

Coordonatele sunt scrise într-un fișier text într-un mod în care pot fi citite ușor de către programul Zbrush. Începând de la primul punct, coordonata x este scrisă pe primul rând, iar coordonata y pe cel de-al doilea. În continuare, cel de-al doilea punct ocupă liniile trei și patru. Astfel un punct "n" are coordonatele x și y pe liniile "2n-1", respectiv "2n".

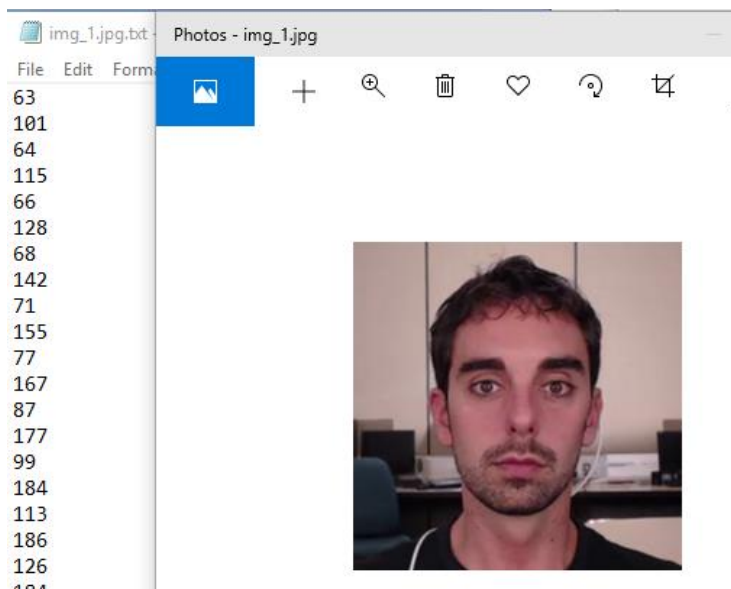


Fig 2.3 Exemplu coordonate puncte de interes

Numele fișierului text format păstrează același nume ca imaginea din care provine, chiar și extensia sa jpg, pentru a putea fi accesat și citit mai ușor ulterior.

Procesul este iterativ, astfel se obțin patruzeci de fișiere text din cele patruzeci de imagini.

2.2 Interfața programului Zbrush si procesul de sinteză

2.2.1 Interfața Zbrush

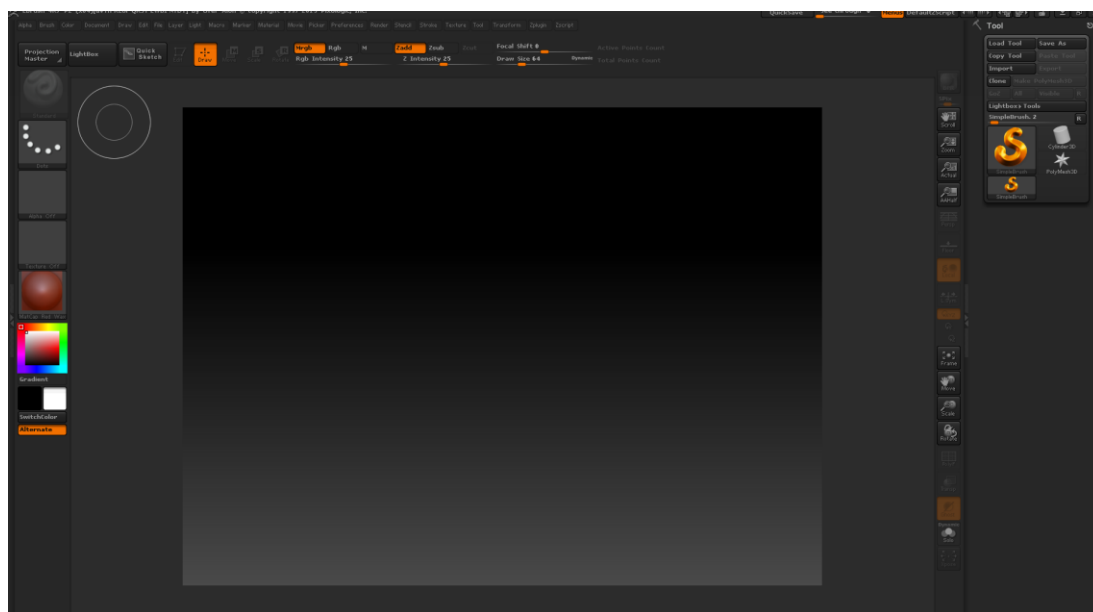


Fig 2.4 Interfața programului Zbrush

La prima deschidere a programului Zbrush se vor activa multe elemente ale interfeței care nu sunt folosite pentru aplicație. Acestea trebuie închise pentru ca aplicația să ruleze cu succes. Se poate observa că spațiul de lucru este mai mic decât întreaga fereastră a aplicației. Pentru aceasta se vor face niște modificări în script. Coordonatele mouse-ului sunt afișate în timp real în meniul Utilities. Folosind coordonatele x și y ale mouse-ului putem determina coordonatele la care începe și se termină fereastra efectivă de lucru în cadrul programului.

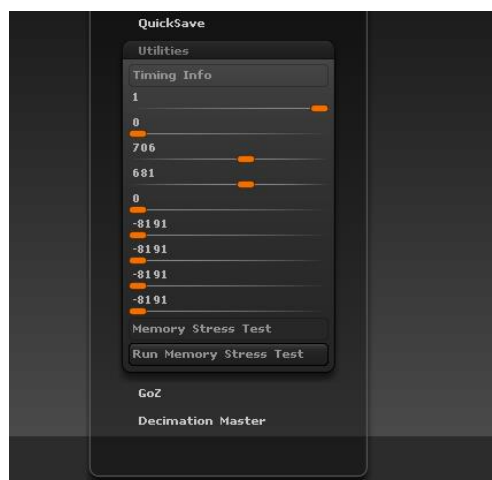


Fig 2.5 Coordonatele mouse-ului în Zbrush

Zbrush fiind în principal un program de modelare 3D are o interfață încărcată cu multe facilități care nu sunt folosite în cadrul acestui proiect.

Din meniul Modifiers se vor determina axa, sau axele de rotație și numărul de cadre pe care îl vor avea secvențele video. Dacă numărul de cadre este 360, fiecare cadru va corespunde unui unghi. În cazul nostru, am ales ca secvențele video să aibă 1440 de cadre, fiecare cadru fiind astfel 0.25 dintr-un unghi. Acest lucru a fost ales pentru a crea o bază de date cât mai mare și cât mai exactă.

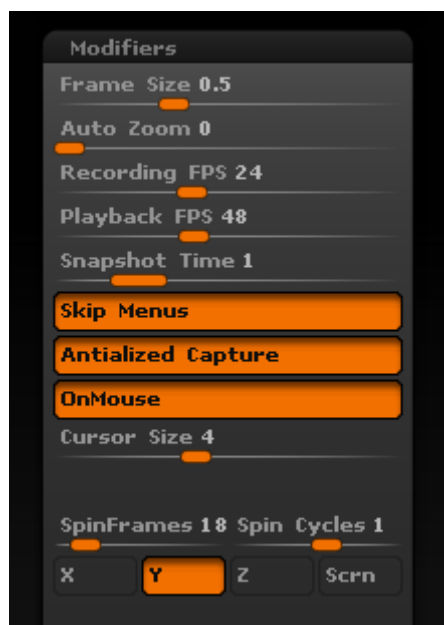


Fig 2.6 Meniul modifiers

Pentru a rula un script se apasă butonul Load script din meniul Zscript. Acestea sunt singurele acțiuni ce trebuie făcute manual de către utilizator.

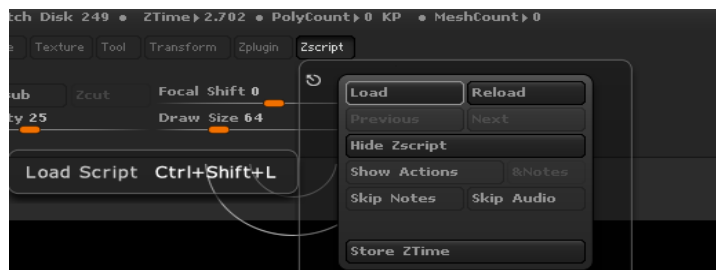


Fig 2.7 Încărcarea scriptului Zbrush

Odată ales fișierul text în care se află scriptul programul va rula instrucțiunile automat.

2.2.2 Scriptul Zbrush

Odată încărcat, primul lucru pe care scriptul îl va face este să simplifice interfața programului, oferind mai mult spațiu de lucru.

În continuare se deschide un bust 3D implicit. Acesta va fi folosit pentru a imprima fiecare imagine. Bustul este centrat pentru a fi posibile operațiuni exacte pe el. Se obține numele fișierului în care se află punctele de interes corespunzătoare imaginii care este sintetizată. Se definesc doi vectori, *coordx* și *coordy*, de lungime 68, care vor conține cele 68 de puncte de interes. Se citește fiecare linie din fișier și este adăugată în vectorul *coordx*, respectiv *coordy*.

```
[If, [MemGetSize, ZB_TextInputMem],  
  [Loop, 68,  
    //citeste o linie intr-o variabila de tip string  
    [VarSet, gBytes, [MemReadString, ZB_TextInputMem, lStr, gOffset, 1]]  
    [varset, coordx(i), lStr]  
    [VarSet, gOffset, gOffset+gBytes]  
    [VarSet, gBytes, [MemReadString, ZB_TextInputMem, lStr, gOffset, 1]]  
    [varset, coordy(i), lStr]  
    [VarSet, gOffset, gOffset+gBytes]  
    [varinc, i]  
    [If, gOffset >= [MemGetSize, ZB_TextInputMem], [LoopExit]]  
  ]
```

O linie nouă este definită ca fiind mărimea de memorie a tuturor linilor până la cea curentă plus mărimea cuvântului pe care vrem să îl citim. De aceea a fost necesară scrierea coordonatelor pe linii distincte, altfel mai era nevoie de un proces de separare a coordonatelor x și y. Variabila *gBytes* memorează câți octeți au fost parcurși până în acest punct, iar *gOffset* reprezintă dimensiunea liniei pe care o citim în acel moment. Indexul *i* definește coordonata la care s-a ajuns. La fiecare iterație se verifică dacă *gOffset* este mai mare sau egal cu dimensiunea blocului de memorie în care a fost introdus întregul fișier, *ZB_TextInputMem*. În cazul în care se depășește această dimensiune, scriptul poate continua, având în vectorii *coordx* și *coordy* toate coordonatele de interes a imaginii pe care se lucrează.

În continuare scriptul va lucra la ochiul stâng al modelului implicit încărcat. Mai întâi se maschează întreaga porțiune care va fi modificată, însemnând un cerc în jurul ochiului stâng.

```
[IPress,Tool:Masking:MaskAll]
```

```
[IPress,Tool:Masking:Inverse]
```

```
[IPress,Transform:Draw Pointer]
```

```
[CanvasStroke, (ZObjStrokeV02n37=Yh221C5v12D7EK2Xh2262Cv132FFh2312Dv147E7h23B14v15A9Bh244FBv168E9h25116v17851h25D30v1846Ch263CAv189ECh26B7Ev19086h270FFv193D3h27D19v19ED4h285E7v1A455h28A4Dv1A9D5h29435v1B06Fh29BE8v1B93Dh2A39Cv1C20Ah2A91Dv1C9BEh2AC6Av1CE25h2AD84v1CE25h2AE9Dv1CF3Eh2B0D1v1D172h2B304v1D28Bh2B41Ev1D3A5h2B651v1D5D9h2B76Bv1D6F2h2B99Ev1D6F2h2BAB8v1D6F2h2BE05v1D5D9h2C038v1D3A5h2C26Cv1D172h2C5B9v1CF3Eh2C6D2v1CE25h2C7ECv1CE25h2C7ECv1CD0Bh2C906v1CD0Bh2CA1Fv1CD0Bh2CA1Fv1CD0B) ]
```

```
[CanvasStroke, (ZObjStrokeV03n2%p2E372D4p1C6E5D6PNNn-FFB5s15F0E66s15F0E66s15F0E66Z=YH2B7V19BK2XH2B7V19B) ]
```

Comenzile de mai sus reprezintă un cod rezultat în urma unei mișcări de mouse. Astfel se produce mascarea ochiului. Apoi masca se inversează pentru a putea fi făcute modificări numai în acea porțiune a bustului.

Pentru a putea muta ochiul stâng, sau în continuare alte elemente ale feței, la coordonatele dorite este nevoie de un punct central al feței de unde mutările să se facă relativ la distanța de la acel punct la coordonata dorită.

```
//centrare pe varful nasului
```

```
[CanvasClick, 665, 380, 740, ((coordy(34)*920)/227) ]
```

```
//mutare
```

```
[CanvasClick, 665, 380, 710-(((coordx(28)-coordx(41))*1479)/227), 380-(((coordy(34)-coordy(40))*920)/227)-40]
```

Coordonatele punctului central sunt 710 și 380 asupra cărora se fac niște modificări. Aceste coordonate sunt obținute incluzând interfața programului, astfel trebuie făcute modificări la coordonatele punctelor de interes pentru a putea fi folosite. Funcția *CanvasClick* simulează o apăsare de buton la coordonatele oferite. Cea de-a doua funcție *CanvasClick* aduce ochiul în poziția dorită, la niște coordonate obținute din multe încercări experimentale. Același proces este parcurs și pentru ochiul drept.

Coordonatele 665 și 380 reprezintă poziția unui element al interfeței programului de unde se pot face mutări de obiecte. Aceste coordonate au fost găsite experimental, iar elementul de interfață revine în acea poziție după fiecare modificare datorită unei comenzi rulate precedent. Este important să avem acest element mereu în aceeași poziție, deoarece orice modificare a modelului 3D este făcută relativ la modificările

anterioare. Dacă nu avem un punct de referință bine stabilit pe care putem să îl folosim, modificările ar fi imposibil de realizat în mod automatizat.

Gura este mutată într-un mod similar, însă poziția acesteia nu trebuie modificată pe axa x, ci numai pe axa y. Toate modificările sunt făcute având în vedere coordonatele obținute din imaginile originale, pentru ca bustul final să fie cât mai asemănător cu fizionomia persoanei din imaginea originală.

După modificarea acestor elemente urmează etapa de proiectare a pozei pe acest bust nou obținut. Programul Zbrush are o unealtă care facilitează texturarea. Într-o variabilă se salvează numele imaginii pe care urmează să o aplicăm asupra bustului. Identificatorul aflat în numele imaginii va fi iterat pentru a se parcurge toate imaginile. Simulând acțiuni făcute în mod normal de către un utilizator folosind funcțiile oferite de program se încarcă imaginea în interfața de texturare și se simulează mișcări de mouse care aplică imaginea asupra bustului. După texturare imaginea trebuie ștearsă din interfață pentru ca una nouă să fie încărcată corect. În acest moment se incrementează contorul global care ține evidența de imagine și fișierul de coordonate pe care le prelucrăm.

În final, bustul texturat este centrat, și este folosită funcția `[IPress,Movie:Turntable]` pentru a realiza o secvență video. Secvența este salvată cu numele "frames" și numărul imaginii care a fost prelucrată și conține 1440 de cadre, fiecare corespunzând unei poziții de unghi de rotație pe axa care a fost aleasă la începutul procesului.

Înainte ca următoarea imagine să fie încărcată, programul trebuie adus la starea inițială, vectorii de coordonate trebuie goliți, iar secvența video trebuie ștearsă din memorie.

Întregul proces prezentat este repetat pentru toate imaginile alese din baza de date UPNA. Numărul de iterații este specificat în scriptul Zbrush. După ce toate imaginile au fost sintetizate, se va obține un fișier care conține patruzeci de secvențe video de 1440 de cadre fiecare, și se poate trece la următoarea etapă a procesului.

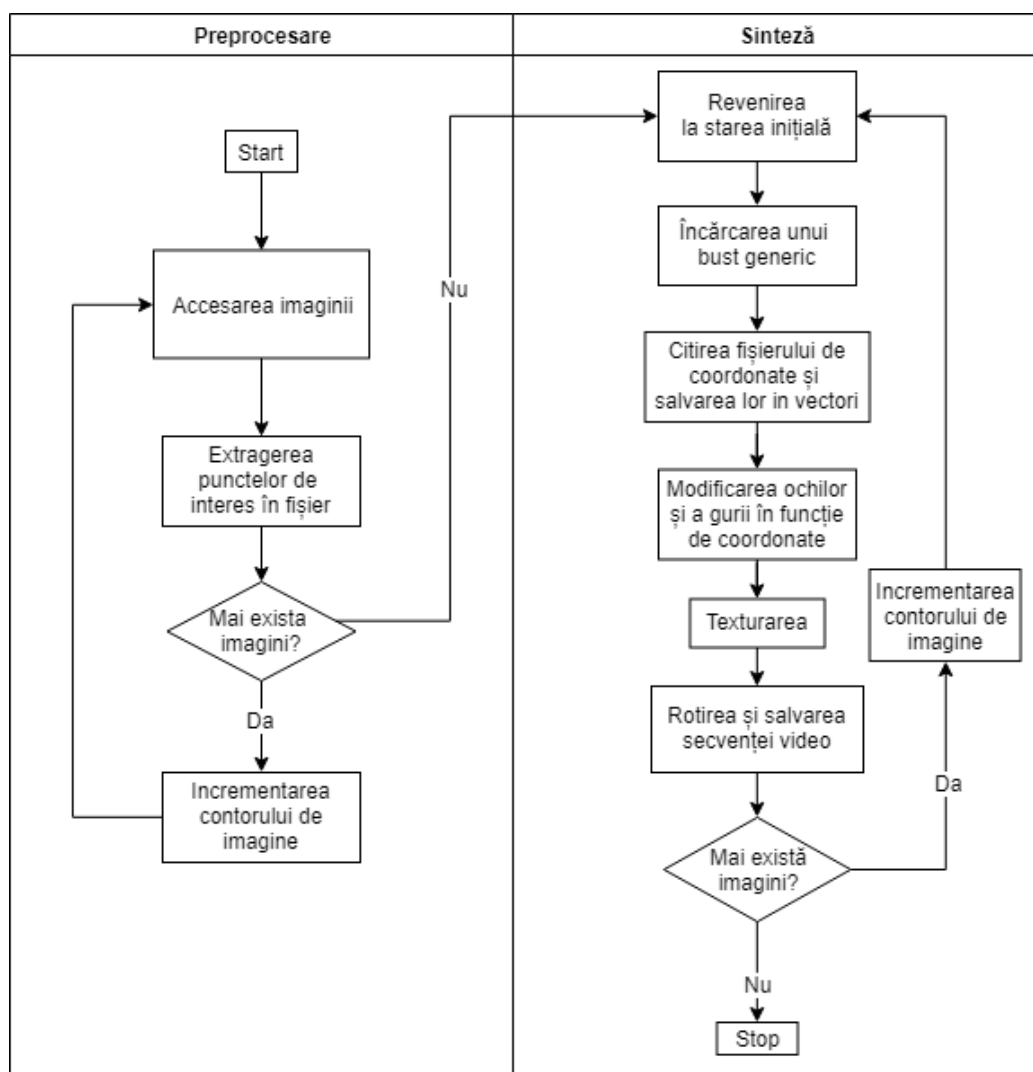


Fig 2.8 Organigrama procesului de sinteză în Zbrush

Capitolul 3. Transformarea secvențelor video în baze de date

Scopul principal al lucrării este de a crea baze de date ordonate, folosite ca date de intrare pentru un algoritm de învățare automată. De aceea acest pas al lucrării este foarte important pentru procesul care va urma.

Deoarece cele patruzeci de secvențe video obținute reprezintă zece utilizatori diferiți, fiecare având câte patru secvențe, fiecare de câte o mie patru sute patruzeci de imagini, acestea trebuie aranjate în mod corespunzător, iar fiecare imagine trebuie să conțină eticheta unghiului la care se află.

Scriptul python care va fi rulat în continuare conține o buclă în care se preia fiecare secvență video și se prelucrează. Se definesc două variabile **user** și **fata**, reprezentând un identificator pentru utilizatorul curent, respectiv secvența aceluși utilizator. Variabila **user** este definită ca identificatorul secvenței video care este prelucrată minus unu, divizată la patru, la care este adunat unu. Variabila **fata** este definită similar, însă diviziunea este înlocuită de operațiunea modulo patru.

```
user = (movies-1)//4 + 1  
fata = (movies-1) % 4 + 1
```

În continuare se verifică dacă în dosarul secvențelor video există un subdosar cu identificatorul utilizatorului curent, iar dacă în acest subdosar există un dosar cu identificatorul secvenței curente. În cazul în care acestea nu există, ele vor fi create.

Utilizând biblioteca OpenCV, se deschide secvența video curentă și se extrag, în interiorul directorului creat, toate cadrele sale. Momentan aceste cadre vor avea nume de la unu la o mie patru sute patruzeci, acesta nereflecând unghiul corect de rotație. Acesta va fi corectat în continuare.

Dupa ce toate cadrele unei secvențe video au fost extrase, se vor șterge cadrele de la trei sute șaiszeci și unu la o mie șaptezeci și noua, deoarece în aceste cadre nu este vizibilă fața utilizatorului, ci doar partea din spate a capului.

Înainte de a putea fi folosite ca date de intrare pentru algoritmul de învățare automată, trebuie corectate etichetele, adică numele imaginilor, să reflecte unghiul de rotație corect. Așadar, se parcurg toate imaginile din toate directoarele și se extrage din etichetă numărul curent.

```
imp=path[5:-4]
```

Path reprezintă numele imaginii, în cazul nostru **frame1200.jpg**. Cu ajutorul sintaxei python putem extrage numărul 1200 deoarece cunoaștem că toate imaginile vor avea cinci caractere înainte de număr și încă patru după acesta, acelea fiind **frame** respectiv **.jpg**. După ce numărul a fost extras, se verifică dacă acesta este mai mare de 360. Acest lucru ar însemna că unghiul de rotație este de fapt unul negativ față de punctul de zero. Așadar, dacă afirmația precedentă este corectă, se va scădea 1440 din el. În final, toate numerele sunt împărțite la patru, deoarece fiecare cadru reprezintă un sfert de unghi de rotație, având 1440 de imagini.

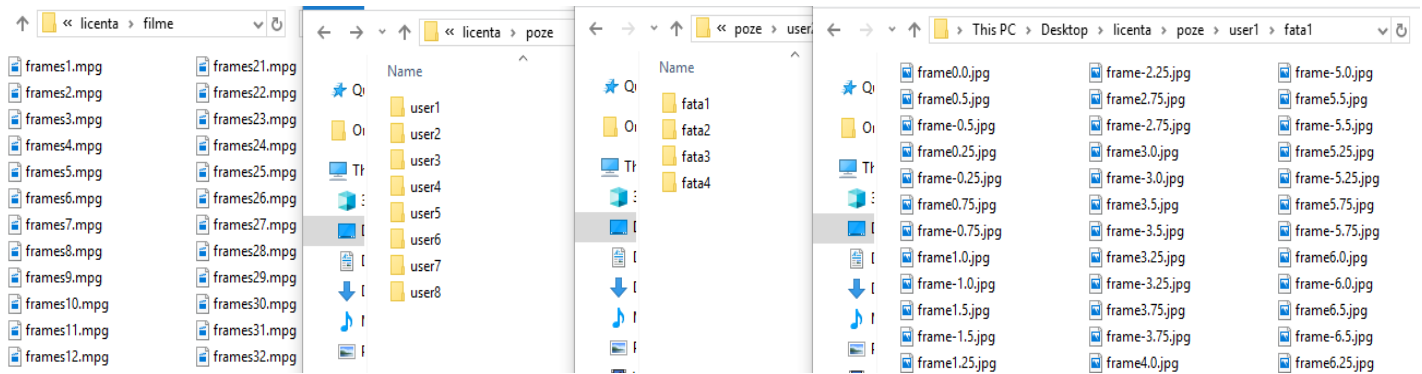


Fig 3.1 Trecerea din format video în imagini si organizarea în dosare

În final, imaginile generate vor arăta astfel

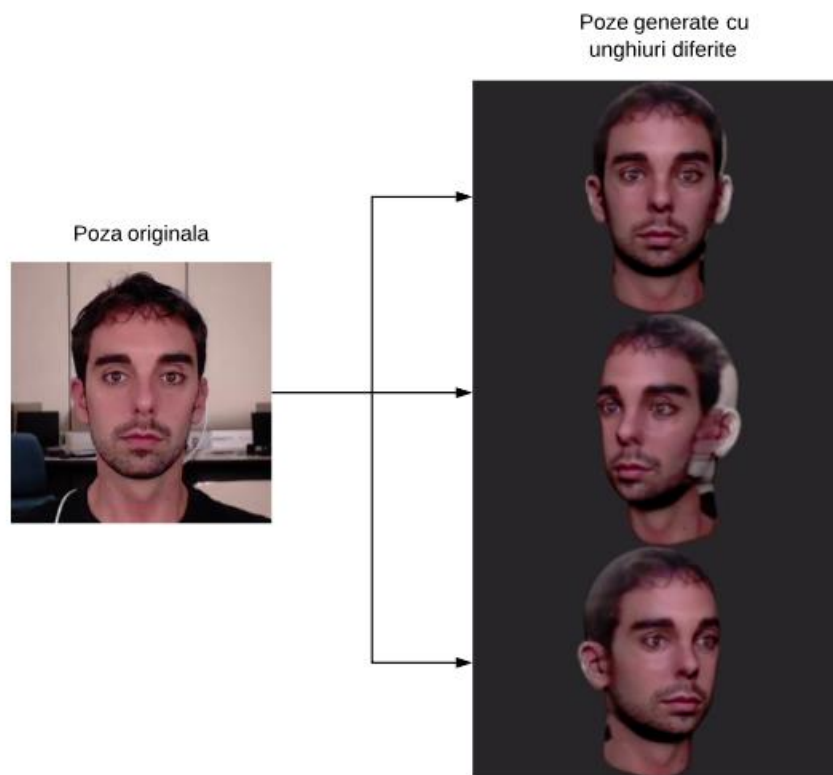


Fig 3.2 Exemple de imagini sintetice generate

Capitolul 4. Procesul de învățare automată și detecție facială

În acest capitol se vor prezenta mașinile cu vectori suport care vor lua deciziile cu privire la unghiul de rotație al fețelor oferite ca imagini de testare, și modelul local binar care se folosește pentru a converti imaginile de intrare în vectori suport pentru antrenare.

4.1 Modelul local binar și algoritmul de funcționare

Modelul local binar este un descriptor vizual utilizat pentru clasificare. Cu ajutorul algoritmului imaginile de intrare sunt convertite în vectori linie ce descriu acea imagine și pot fi folosiți în numeroși algoritmi de învățare automată. Spre deosebire de alți descriptori care procesează o reprezentare globală a unei texturi, modelul local binar crează o reprezentare locală. Această reprezentare este definită de către diferențele pixelilor adiacenți.

Primul pas pentru a genera descriptorul este de a converti imaginea într-una alb-negru. Pentru fiecare pixel din imaginea alb-negru se alege o vecinătate de mărime “ r ” care are ca centru pixelul curent. Pentru acest pixel este calculată o valoare LBP care este stocată într-un vector bidimensional cu lungime și adâncime egale cu a imaginii prelucrate.

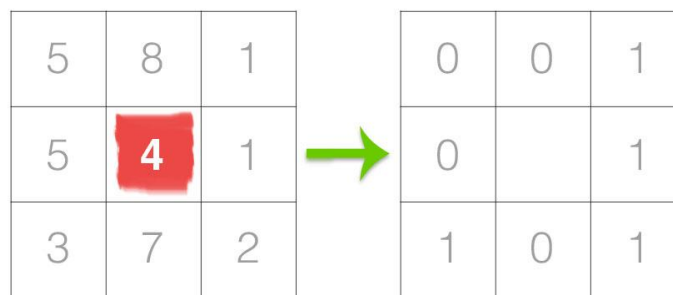


Fig 4.1 Comparația vecinilor în procesul LBP [8]

În figură mărimea r este considerată a fi 8, astfel se va lua ca vecinătate toți pixelii direct adiacenți pixelului prelucrat, în cazul acesta pixelul colorat cu roșu. Numerele de pe fiecare pixel reprezintă intensitatea lor, imaginea fiind alb-negru. Pixelul central se compară cu fiecare pixel din vecinătatea sa, iar dacă valoarea intensității sale este mai mare sau egală, se va înlocui valoarea intensității vecinului comparat cu 1, iar dacă este strict mai mică, cu 0. Având o vecinătate de 8 pixeli, se calculează $2^8=256$ combinații posibile de coduri locale binare.

În continuare trebuie calculat valoarea locală binară a pixelului central. Se va alege o direcție de parcurgere, iar valorile binare se stochează într-un vector de lungime 8.



Fig 4.4 Comparație între imaginea originală și matricea LBP formată [8]

Ultimul pas al procesului este de a calcula histograma matricei LBP creată. Această histogramă va evidenția numărul de apariții al fiecărui cod local binar. Aceasta poate fi folosită ca un vector suport pentru algoritmul de învățare.

În acest exemplu, precum și în lucrare, vecinătatea a fost aleasă de 8 pixeli direct adiacenți pixelului central, însă numărul de puncte și distanța față de pixelul central sunt valori ce pot fi modificate în funcție de nivelul de detalii dorit. În cazul vecinătății de 8, detaliile sunt păstrate, însă în unele cazuri este nevoie de o imagine mai puțin detaliată.

În ultimul rând, în lucrare se folosește un vector LBP invariant la rotație. Conceptul de uniformitate se referă la un cod local binar care are cel mult două treceri de la 0 la 1 sau de la 1 la 0. Spre exemplu, codul 01010010 obținut după procesul LBP nu este uniform deoarece are mai mult de 2 tranziții. Este dovedit experimental că apariția codurilor neuniforme este mult mai mică decât a celor uniforme. Astfel, toate codurile neuniforme au aceeași valoare, iar fiecare cod uniform are câte o valoare separată. Acest lucru este folositor deoarece reduce lungimea vectorului suport de la 255 la 59. Codurile uniforme se pot interpreta ca fiind un colț sau o latură.

4.2 Mașinile cu vectori suport și regresie

Mașinile cu vectori suport sunt modele supervizate de învățare automată folosite pentru clasificare și regresie bazate pe conceptul de maximizare a „marginii” ce separă date aparținând unor clase distincte. În cazul unei probleme de clasificare liniară, pentru un set de date de intrare aparținând uneia din două clase distincte, algoritmul SVM este capabil să separe printr-un hiperplan datele astfel încât toate datele dintr-o anumită clasă să fie pe o parte a hiperplanului, separate de date din cealaltă clasă. Un hiperplan este un plan de dimensiuni $n-1$ unde n este dimensiunea planului în care sunt reprezentate datele. Algoritmul încearcă să găsească hiperplanul care are cea mai mare distanță față de cel mai apropiat punct aparținând unei clase.

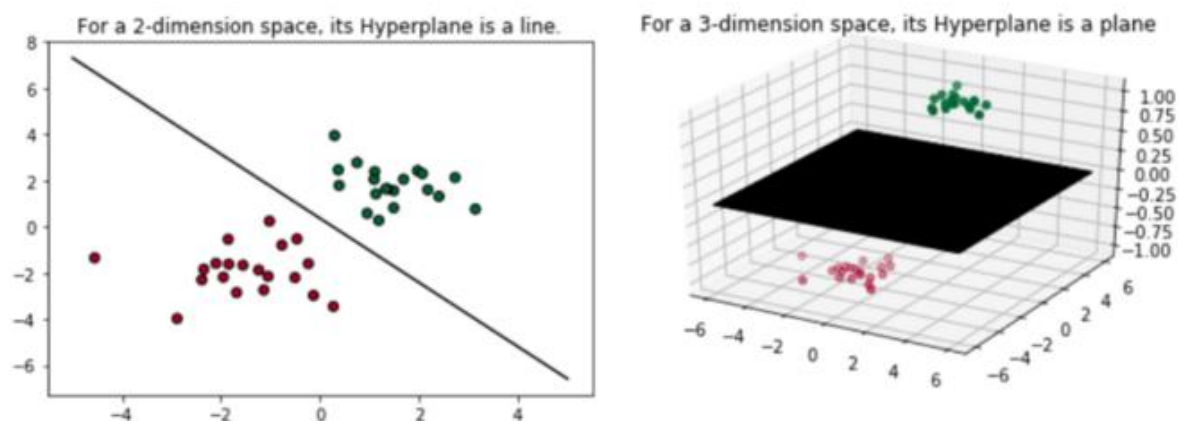


Fig 4.5 Hiperplan in cazul reprezentării datelor în 2D și 3D [9]

Un hiperplan poate fi exprimat matematic cu următoarea formulă:

$$a_0 + a_1x_1 + \dots + a_nx_n = b$$

Astfel, cele două clase pot fi definite ca:

$$a_0 + a_1x_1 + \dots + a_nx_n > 0$$

$$a_0 + a_1x_1 + \dots + a_nx_n < 0$$

Se poate considera că modelul care oferă cele mai bune rezultate pentru majoritatea problemelor este acela care desparte cel mai strict clasele, oferind cea mai mare margine de separație între ele. Această margine este distanța dintre două drepte paralele cu dreapta de separare care ating cel puțin o valoare din fiecare clasă.

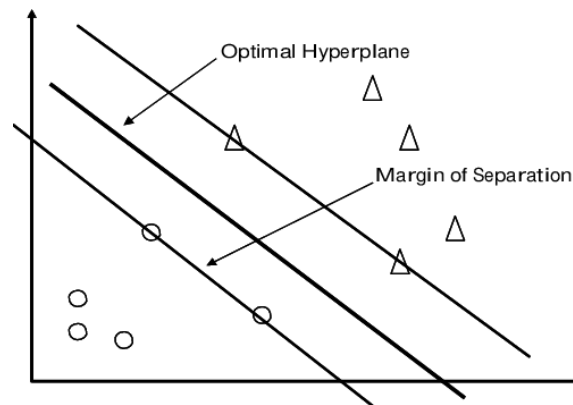


Fig 4.6 Marginea de separație dintre două clase [17]

Aceste puncte se numesc vectori suport, care dau și numele metodei

În majoritatea timpului, datele nu vor putea fi separate în mod liniar, ci vor fi separate prin metode neliniare precum **Soft Margin** și **Trucuri kernel**.

Soft Margin reprezintă o metodă de clasificare neliniară în care se permite o eroare de clasificare. Toleranța la această eroare este definită de către utilizator printr-un cost de eroare.

Trucurile kernel sunt metode de clasificare ce modifică numărul de dimensiuni ale spațiului în care sunt descrise datele, pentru a găsi delimitarea corectă. Cu ajutorul acestora, datele se transformă din spațiul original, numit spațiul atributelor, într-un spațiu multi-dimensional numit spațiul trăsăturilor. În lucrare se utilizează RBF kernel, acesta fiind :

$$\phi(x, center) = \exp(-\gamma \|x - center\|^2)$$

Această funcție este folosită pentru a genera date noi în alte spații. Variabila gamma controlează influența acestor noi date. Cu cât este acesta mai mare cu atât va fi mai ondulantă linia ce delimitează clasele. Similar cu metoda soft margin, gamma este un parametru definit de către utilizator.

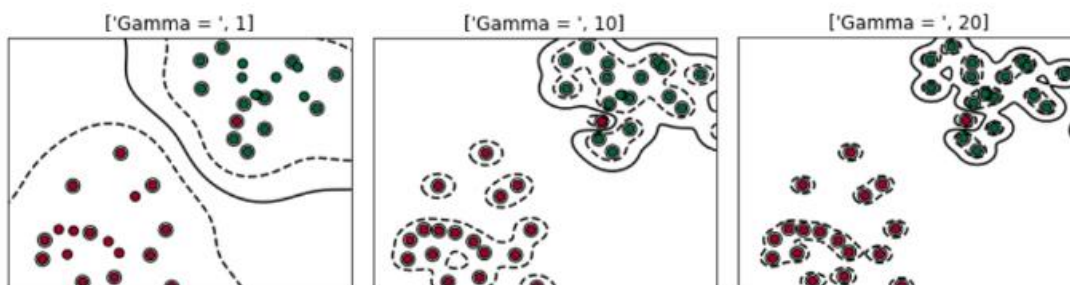


Fig 4.7 Influența parametrului gamma asupra delimitării claselor [9]

Aceste metode sunt folositoare pentru a ordona date într-un număr finit de clase. Deseori este nevoie de o operație de regresie pentru a aranja date într-un număr infinit de clase, clasa aleasă fiind cea mai apropiată de clasele oferite în procesul de învățare al algoritmului. Și în cazul nostru, avem nevoie ca rezultatul unei imagini să fie cel mai apropiat unghi la care este rotită fața. Pentru aceasta se folosește o variantă de SVM numită SVR, support vector regression.

Considerând funcția hiperplanului ca fiind

$$wx + b = 0$$

Vom defini ecuațiile celor două linii ce delimitează datele ca " $wx + b = -e$ " și " $wx + b = +e$ " unde " e " reprezintă distanța de la hiperplan la cel mai îndepărtat punct de date pe care îl vom considera corect în algoritmul nostru. Astfel putem considera $2e$ distanța de decizie, și cele două linii paralele cu hiperplanul și care se află la distanțele e și $-e$ de el, pragul de decizie.

Support Vector Machine for Regression - Radial Basis Kernel

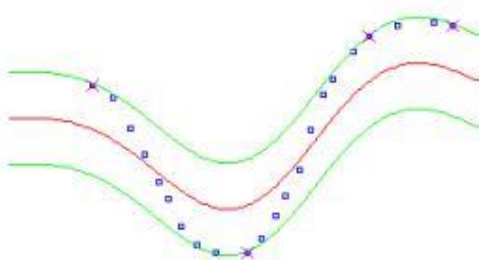


Fig 4.8 Reprezentare a pragului de decizie pentru regresie [14]

4.3 Implementarea scriptului Python

În lucrarea prezentată, implementarea unui SVM este o sarcină relativ simplă, deoarece avem la dispoziție biblioteci cu funcții specializate. Adevărata problemă este preprocesarea datelor.

Inițial se parcurg toate imaginile din baza de date creată precedent. Fiecare imagine este convertită în alb-negru și este decupată, pentru a elimina porțiunile neimportante. Apoi, asupra fiecărei imagini este aplicat algoritmul LBP. Pentru aceasta, se folosește constructorul :

```
feature.local_binary_pattern(image, points, radius, method)
```

din librăria **skimage**. Astfel, se generează un obiect de tip LBP care va transforma fiecare imagine. Parametrii săi sunt imaginea pe care dorim să o convertim, numărul de pixeli în jurul pixelului central pe care îi comparăm în algoritmul LBP, distanța pixelilor față de cel central și metoda pentru a determina modelul. În final se generează histograma folosind funcția `ravel()` pe imaginea LBP creată, pentru a transla datele dintr-o matrice bidimensională, într-un vector cu o singură dimensiune.

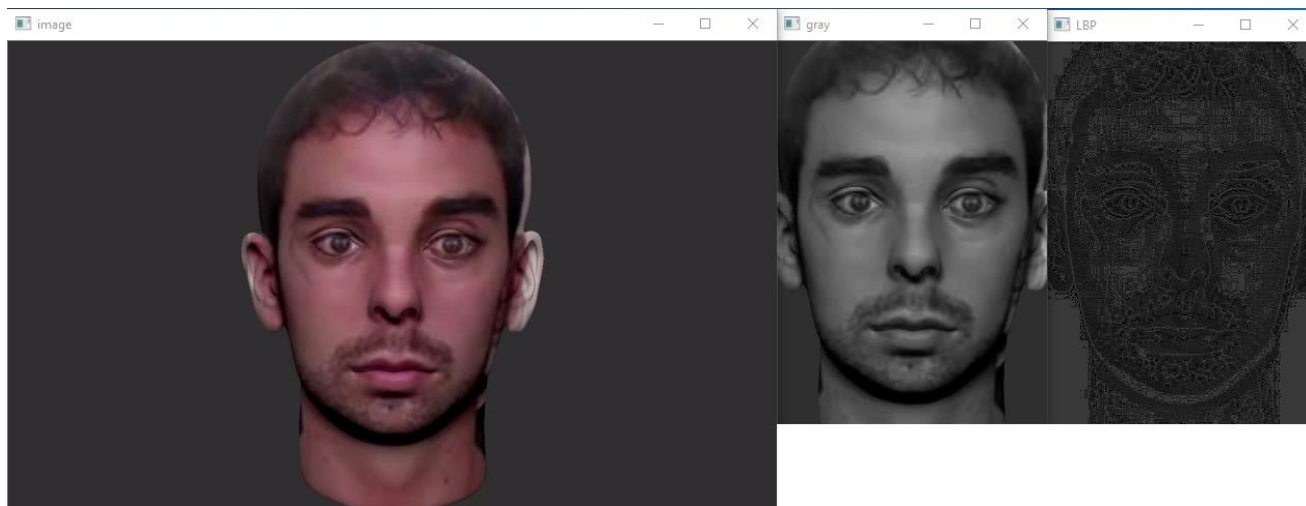


Fig 4.9 Procesul de creare a imaginii LBP

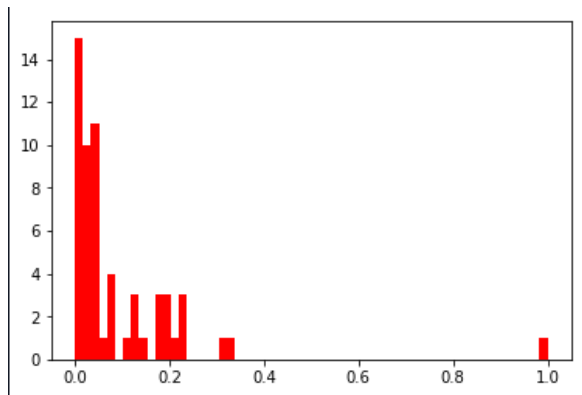


Fig 4.10 Histograma LBP normalizată

Următorul pas este de a extrage etichetele fiecărei imagini pentru a putea fi folosite pentru învățare. Acest proces este similar cu cel descris anterior. Etichetele se extrag folosind concatenarea și se introduc într-un vector. Astfel se știe că etichetele păstrează aceeași ordine cu cea în care au fost citite imaginile.

Histogramele generate de către procesul LBP sunt introduse într-o matrice . Deoarece aceste histograme au fost generate folosind modelul uniform, ele vor avea lungimea de 59 de valori posibile. Adâncimea matricii este numărul de imagini procesate. Această matrice este necesară ca dată de intrare pentru mașina cu vector suport.

După ce toate imaginile au fost transformate în histograme și introduse în matrice, se vor calcula valoarea minimă și maximă, care se vor folosi pentru normalizarea datelor. Parcurgem întreaga matrice și normalizăm datele folosind formula

$$\text{date}[x][y] = (\text{date}[x][y] - \text{dmin}[y]) / (\text{dmax}[y] - \text{dmin}[y])$$

astfel datele vor fi cuprinse între 0 și 1. Acest lucru reduce volumul de calcul.

Astfel este finalizat procesul de creare a matricii de învățare. În continuare se crează matricea de test, parcurgând aceași pași, însă pentru niște imagini ce nu se află în lotul de antrenare. În final această matrice este normalizată folosind aceleași valori minim și maxim, pentru a păstra integritatea datelor.

Acum se inițializează GridSearchCV. Acesta primește ca parametri un estimator, în cazul nostru un SVR și o listă de parametri pe care să îi folosească. GridSearchCV are rolul de a căuta cea mai bună combinație de parametri Cost, Gamma și Kernel pentru datele oferite estimatorului. În cazul nostru Kernelul folosit este RBF, Costul poate să ia valorile

2**-5, 2**-3, 2**-1, 2**1, 2**3, 2**5, 2**7, 2**9, 2**11, 2**13, 2**15,

iar Gamma valorile

2**-15, 2**-13, 2**-11, 2**-9, 2**-7, 2**-5, 2**-3, 2**-1, 2**1, 2**3.

GridSearchCV va calcula scorul estimării pentru fiecare caz și va anunța care este cea mai bună combinație de parametri.

Folosind funcția fit cu parametrii matricea de date și lista de etichete, fiecărei linie de date îi este asociată o etichetă, iar astfel clasificatorul clf a "învățat". În final se populează lista de etichete de test folosind funcția predict cu parametru matricea datelor de test. În cazul în care procesul a avut succes, valorile din lista etichetelor de test corecte și cele din lista etichetelor de test prezise de către estimator ar trebui să fie cât mai apropiate.

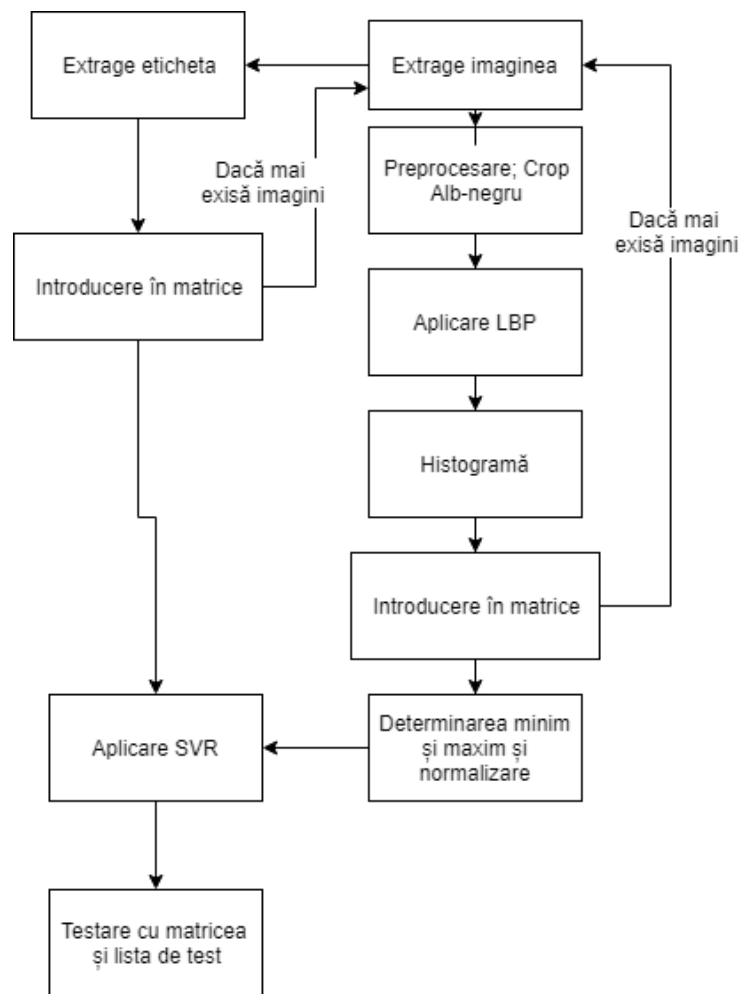


Fig 4.11 Organigrama procesului de învățare

Capitolul 5. Rezultate și concluzii

Deoarece lucrarea a avut două componente, sinteza imaginilor și luarea de decizii, voi prezenta concluziile pentru cele două părți separat.

5.1 Concluzii sinteză

Partea de sinteză a lucrării a fost realizată cu succes. Programul Zbrush s-a dovedit a fi foarte folositor pentru acest proiect, și nu am găsit o soluție mai eficientă pentru texturare automată. Desigur, programul

Zbrush este făcut pentru modelare 3D, aşadar vor exista nişte performanţe scăzute faţă de un program special realizat pentru sinteză. O posibilă îmbunătăţire este crearea unui program specializat pentru astfel de aplicaţii.

Codul utilizat pentru sinteză poate fi îmbunătăţit pentru a scădea timpul de rulare. În momentul de faţă, patruzeci de imagini sunt sintetizate în cincisprezece minute de către Zbrush. În acest timp calculatorul nu poate fi folosit, deoarece aplicaţia foloseşte mouse-ul pentru diferite acţiuni.

În toate aceste cazuri am vorbit despre sinteza imaginilor rotite pe o singură axă, Y. Dacă dorim două grade de libertate în rotaţie ca rezulta o creştere exponenţială a timpului de rulare. Pentru fiecare imagine sunt şaizeci de filme în plus, pentru fiecare unghi pe ce-a de-a doua axă. Astfel timpul de rulare creşte la cincisprezece ore. Acest lucru nu ar fi o problemă, deoarece calculatorul poate fi lăsat să ruleze, însă după o perioadă lungă de operare, apar probleme cu programul Zbrush, iar unele comenzi din script nu mai sunt rulate corect. Acest lucru poate fi remediat fie prin reducerea numărului de imagini iniţiale, prin rularea scriptului de mai multe ori la intervale de o oră, sau prin proiectarea unui program specializat. De asemenea, dimensiunea spaţiului necesară pe disc devine foarte mare. Am încercat să rulez scriptul de python ce extrage imaginile şi apoi şterge filmele în acelaşi timp cu scriptul de sinteză, însă nu a fost posibil. Se poate specifica din Zbrush ca filmele să aibă o dimensiune mai mică, însă cea mai bună soluţie rămâne proiectarea unui program specializat care realizează întreg procesul de sinteză şi de extragere a imaginilor în acelaşi timp.



Fig 5.1 Exemplu de rotaţie folosind două axe

Deşi limbajul de scriptare Zscript este foarte folositor, au apărut nişte probleme care ar fi putut fi rezolvate foarte simplu într-un limbaj de programare tradiţional. Au apărut probleme majore când a trebuit să calculez diferenţa de pixeli dintre documentul unde se poate lucra propriu-zis, şi interfaţa grafică care nu era folositoare pentru aplicaţie. De asemenea, orice operaţiune de mutare, scalare sau rotaţie a fost imposibilă

doar din cod. Cea mai mare problemă întâmpinată a fost alinierea modelului cu imaginea ce trebuie texturată. Acest lucru a fost realizat prin încercări succesive și este relativă la dimensiunea documentului, astfel dimensiunea monitorului. Așadar codul nu este universal pentru orice calculator și pot apărea probleme în unele cazuri. O aplicație special realizată pentru această sarcină ar putea rezolva aceste probleme.

Procesul de extragere a imaginilor din secvențe video a fost realizat cu succes. Organizarea în baza de date este eficientă pentru aplicația în care o folosesc, însă pentru alte aplicații poate fi necesară alt fel de organizare. Procesul este foarte rapid și nu prezintă probleme de timpi de rulare. Singura problemă este că scriptul trebuie rulat manual după încheierea generării de secvențe video. De asemenea, codul python nu este relativ la mașina pe care rulează, astfel nu va funcționa pe alte calculatoare. Soluția ar fi scrierea unui cod universal care primește ca parametru locația secvențelor video și locația unde se dorește a fi generată baza de date.

În concluzie, soluția propusă este eficientă pentru a genera imagini sintetice, însă multe îmbunătățiri se pot face proiectând un program special pentru aceste lucruri în locul folosirii programului Zbrush. S-a arătat că este posibilă sinteza cu programul existent și este un proces relativ eficient, depinzând de numărul de poze inițiale și numărul de imagini dorite la ieșire.

5.2 Concluzii estimare

Partea de estimare a lucrării a fost realizată cu succes, iar rezultatele obținute pentru estimarea unor imagini ce nu se află în lotul de antrenare sunt bune și dovedesc eficiența metodei. Am determinat prin experimente cu diferite dimensiuni ale lotului de antrenare faptul că cei mai buni parametri sunt $\text{Cost}=2^{13}$ iar $\text{Gamma}=2^{-3}$. Pentru acești parametri pe întreg lotul de antrenare am obținut următoarele rezultate

```
[-22.9924162 -33.93001075 -40.71589022 -5.11776127 -62.75269624
 -90.07357931 -4.57465343 17.46877633 28.6967417 43.9690013
 5.47322429 68.55455698 78.28131267]
*****
['-20.25', '-32.5', '-47.5', '-5.25', '-64.0', '-87.0', '0.0', '20.25', '32.25', '47.5', '5.75',
 '64.0', '87.0']
Estimation score: 0.9361263492493036
MSE: 16.707879560071866
```

Fig 5.2 Rezultate finale

Prima listă prezentată în figură este lista etichetelor estimate. Acestea au fost realizate de către SVR pe baza a treisprezece imagini date în lotul de testare.

Cea de-a doua listă reprezintă etichetele reale ale imaginilor din lotul de testare.

Scorul de estimare este determinat de către GridSearchCV și reprezintă distanța datelor estimate față de hiperplanul ce a fost ales de către estimator pentru a separa datele. Acesta este folosit pentru a determina cei mai buni parametri pentru estimare.

MSE reprezintă eroarea pătratică medie.

Deși s-a dovedit că imaginile realizate sintetic se pot folosi pentru a crea o bază de date ce dă rezultate satisfăcătoare când se aplică acest proces de estimare prin regresie, nu s-a testat dacă, completând baza de date UPNA cu aceste imagini obținem rezultate mai bune decât dacă s-ar folosi baza de date originală. Sunt de părere că rezultatele s-ar fi îmbunătățit, deoarece baza de date creată sintetic conține multe imagini cu fețe la unghiuri care nu se află în baza de date originală. Așadar în viitor, dacă se observă experimental că procesul de sinteză îmbunătățește rezultatele estimării, codul python trebuie standardizat pentru a putea funcționa cu orice imagini. Momentan acesta este optimizat să funcționeze cu imaginile din baza de date UPNA.

De asemenea, imaginile obținute în urma procesului de sinteză sunt sintetice, iar în unele cazuri deformat, neputând fi folosite într-o bază de date. Aceste lucruri pot deteriora performanțele sistemului de estimare. Era de așteptat ca imaginile să pară sintetice, iar acest lucru nu poate fi remediat, fiind legat de natura întregului proces de sinteză.



Fig 5.3 Diferențe între imaginea originală și cea sintetizată

Aceste imperfecțiuni ce apar sunt datorate procesului de sinteză, astfel orice îmbunătățire a aceluia proces ar aduce rezultate mai bune în procesul de estimare. Un număr mic de imperfecțiuni în baza de date este acceptabil deoarece nu influențează datele într-un mod major, însă dacă un utilizator întreg este sintetizat în mod eronat, pot apărea probleme.



Fig 5.4 Erori și imperfecțiuni apărute în procesul de sinteză

În concluzie, s-a dovedit că pentru problema generării bazelor de date echilibrate pentru detecția unghiului de rotație al capului, metoda de sinteză propusă este realizabilă, este mai eficientă din punct de vedere al timpului de realizare al imaginilor și al costului de realizare, estimările făcute pe un subgrup din această bază de date oferă rezultate satisfăcătoare și este necesar să se determine dacă baza de date creată conduce la rezultate mai bune în estimarea unghiului de rotație al capului când este adăugată unei baze de date deja existentă.

Bibliografie

- [1] https://en.wikipedia.org/wiki/Support-vector_machine accesat la data: 01.08.2019
- [2] https://en.wikipedia.org/wiki/Supervised_learning accesat la data: 01.08.2019
- [3] <https://www.analyticsindiamag.com/5-important-techniques-to-process-imbalanced-data-in-machine-learning/> accesat la data: 01.08.2019
- [4] https://en.wikipedia.org/wiki/Local_binary_patterns accesat la data: 03.08.2019
- [5] <https://www.analyticsindiamag.com/5-important-techniques-to-process-imbalanced-data-in-machine-learning/> accesat la data: 03.08.2019
- [6] <https://www.pyimagesearch.com/2017/04/03/facial-landmarks-dlib-opencv-python/> accesat la data: 04.08.2019
- [7] <https://opencv.org/about/> accesat la data: 08.08.2019
- [8] <https://www.pyimagesearch.com/2015/12/07/local-binary-patterns-with-python-opencv/> accesat la data: 09.08.2019
- [9] <https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47> accesat la data: 09.08.2019
- [10] Leon, Florin. (2014). Inteligența artificială: mașini cu vectori suport.
- [11] Denisa, Ispan & Cojocaru, Magda. (2018). Clasificarea imaginilor ce conțin cifre folosind baza de date MNIST, operatorul LBP și rețeaua neuronală MLP. 10.13140/RG.2.2.21130.11205.
- [12] <https://skybiometry.com/faq/how-do-head-rotation-angles-look-like-and-which-angles-our-api-can-find/> accesat la data: 03.09.2019
- [13] T. Ojala, M. Pietikäinen, and D. Harwood (1996), "A Comparative Study of Texture Measures with Classification Based on Feature Distributions", Pattern Recognition, vol. 29, pp. 51-59.
- [14] <http://www.prima.inrialpes.fr/perso/Gourier/Faces/HPDatabase.html> accesat la data: 04.09.2019
- [15] N. Gourier, D. Hall, J. L. Crowley *Estimating Face Orientation from Robust Detection of Salient Facial Features* Proceedings of Pointing 2004, ICPR, International Workshop on Visual Observation of Deictic Gestures, Cambridge, UK
- [16] <http://kernelsvm.tripod.com/> accesat la data: 05.09.2019
- [17] Williams, Nigel & Zander, Sebastian & Armitage, Grenville. (2006). Evaluating machine learning algorithms for automated network application identification.

[18] Mikel Ariz, José J. Bengoechea, Arantxa Villanueva, Rafael Cabeza, [A novel 2D/3D database with automatic face annotation for head tracking and pose estimation](#), Computer Vision and Image Understanding, Volume 148, July 2016, Pages 201-210, ISSN 1077-3142

Anexa A

Script Zbrush

```
[VarDef,contor,1]

[Vardef,contorfisier,1]

[vardef,contorfilm,1]

[Loop,1,

//Deschidere model

[FileNameSetNext,"C:\Users\Andrei\Desktop\licenta\capf.ZPR"][IPress,File:Open]

[IUnPress,Draw:Perspective]

[IPress, 7524]

]

//setare cate poze sunt

[Loop,2,

//sticky on

[IPress,Transform:Move]

[CanvasClick, 700,290]

[IPress,Transform:Draw Pointer]

[IPress,Tool:SubTool:PM3D_FemaleHead_2]

[ISet,Tool:Geometry:SDiv,1]

[IPress,Material:MatCap Gray]

[IPress,Material:SkinShade4]

//editare model

[Vardef,i,1]

[varset,i,1]

[VarSet,gFilePath,[strmerge,"img",[var,contorfisier],".jpg.txt"]]

[varinc,contorfisier]

[If,[FileExists,gFilePath],

//create memory block from the file

[MemCreateFromFile,ZB_TextInputMem,gFilePath]

]

[VarSet,gOffset,0]

[VarDef,lStr,""]

[VarSet,lStr,""]

[VarDef,coordx(69)]

[VarDef,coordy(69)]

[If,[MemGetSize,ZB_TextInputMem],

[Loop,68,

//citeste o linie intr-o variabila de tip string

[VarSet,gBytes,[MemReadString,ZB_TextInputMem,lStr,gOffset,1]]

[varset,coordx(i),lStr]

[VarSet,gOffset,gOffset+gBytes]

[VarSet,gBytes,[MemReadString,ZB_TextInputMem,lStr,gOffset,1]]

[varset,coordy(i),lStr]

[VarSet,gOffset,gOffset+gBytes]

[varinc,i]

[If,gOffset >=

[MemGetSize,ZB_TextInputMem],[LoopExit]]

]

[IKeyPress,CTRL,[IPress,Stroke:Circle]]

[IUnPress,Transform:Activate Symmetry]

//ochi stang

[IPress,Tool:Masking:MaskAll]

[IPress,Tool:Masking:Inverse]

[IPress,Transform:Draw Pointer]

[CanvasStroke,(ZObjStrokeV02n37=Yh221C5v12D7EK2Xh2262Cv132FFh2312Dv147E7h23B14v15A9Bh244FBv168E9h25116v17851h25D30v1846Ch263CAv189ECh26B7Ev19086h270FFv193D3h27D19v19ED4h285E7v1A455h28A4Dv1A9D5h29435v1B06Fh29BE8v1B93Dh2A39Cv1C20Ah2A91Dv1C9BEh2AC6Av1CE25h2AD84v1CE25h2AE9Dv1CF3Eh2B0D1v1D172h2B304v1D28Bh2B41Ev1D3A5h2B651v1D5D9h2B76Bv1D6F2h2B99Ev1D6F2h2BAB8v1D6F2h2BE05v1D5D9h2C038v1D3A5h2C26Cv1D172h2C5B9v1CF3Eh2C6D2v1CE25h2C7ECv1CE25h2C7ECv1CD0Bh2C906v1CD0Bh2CA1Fv1CD0Bh2CA1Fv1CD0B)]

[CanvasStroke,(ZObjStrokeV03n2%p2E372D4p1C6E5D6PNNn-FFB5s15F0E66s15F0E66s15F0E66Z=
```

```

YH2B7V19BK2XH2B7V19B)]

[IPress,Tool:Masking:Inverse]

[IPress,Transform:Move]

//centrare pe varful nasului

[CanvasClick, 665, 380,
740, ((coordy(34)*920)/227)]

//mutare

[CanvasClick, 665, 380, 710-(((coordx(28)-
coordx(41))*1479)/227), 380-(((coordy(34)-
coordy(40))*920)/227)-40]

[IPress,Transform:Draw Pointer]

[IPress,Tool:Masking:MaskAll]

[IPress,Tool:Masking:Inverse]

//ochi drept

[IPress,Transform:Draw Pointer]

[CanvasStroke, (ZObjStrokeV02n44=Yh3AA98v13999K2X
h3A97Ev13BCCCh3A1CAv1414Dh39E7Dv1449Ah398FDv146CD
h38CE2v152E8h3852Ev15982h37D7Bv16136h36F2Dv16E6A
h3665Fv17504h35FC5v17A84h353AAv18352h35291v1846C
h34BF7v187B9h3455Dv18B06h340F6v18D39h33DA9v18F6C
h3318Ev19720h32AF4v19954h31DC0v19FEEh312BFv1A56E
h30B0Bv1A8BBh3058Bv1AAEFh2FEF0v1AE3Ch2FBA3v1B06F
h2F970v1B189h2F73Dv1B3BCh2F623v1B4D6h2F3F0v1B709
h2F2D6v1B823h2F0A3v1B823h2EE6Fv1BA56h2EC3Cv1BB70
h2EC3Cv1BDA3h2EC3Cv1BEBDh2ED55v1C0F0h2EE6Fv1C20A
h2EF89v1C324h2F0A3v1C324h2F2D6v1C43Eh2F3F0v1C43E
h2F509v1C43Eh2F623v1C557h2F623v1C557)]

[CanvasStroke, (ZObjStrokeV03n2%p2E38140p1A7CB7EP
NNn-
FC05s147043As147043As147043AZ=YH358V155K2XH358V1
55)]

[IPress,Tool:Masking:Inverse]

[IPress,Transform:Move]

[CanvasClick, 665, 380,
590, ((coordy(34)*920)/227)]

[CanvasClick, 665, 380, 620+(((coordx(48)-
coordx(28))*1479)/227), 380-(((coordy(34)-
coordy(43))*920)/227)-40]

[IPress,Transform:Draw Pointer]

[IPress,Tool:Masking:MaskAll]

[IPress,Tool:Masking:Inverse]

//gura

[IPress,Transform:Draw Pointer]

```

```

[CanvasStroke, (ZObjStrokeV02n80=Yh353AAv220ABK2X
h3505Dv222DFh34BF7v2285Fh3455Dv23013h3370Fv24094
h32F5Bv24615h3245Av24EE2h311A5v25C17h3000Av2694B
h2EE6Fv2744Ch2DDEEV27Ch2C5B9V287h2B1EAv28EB4h29F
36v29435h290E8v299B5h28180v2A283h2744Cv2A803h26A
65v2A91Dh25C17v2AC6Ah25696v2AD84h24CAFv2AE9Dh244
FBv2AFB7h23F7Bv2AFB7h236ADv2B41Eh22CC6v2B76Bh229
79v2B99Eh22745v2BAB8h222DFv2BCEBh21E78v2BE05h21D
5Ev2BF1Fh21C45v2C038h21B2Bv2C038h21B2Bv2BF1Fh223
F8v2BCEBh2347Av2BAB8h243E1v2B651h2522Fv2B1EAh25D
30v2AE9Dh26A65v2AB50h26C98v2AA36h26FE5v2A91Dh270
FFv2A803h27332v2A803h2744Cv2A6E9h27565v2A6E9h277
99v2A6E9h27D19v2A5D0h2881Av2A4B6h29668v2A283h2A1
69v2A283h2A5D0v2A283h2A6E9v2A283h2A5D0v2A39Ch2A3
9Cv2A39Ch2A169v2A39Ch29F36v2A5D0h29D02v2A5D0h29B
E8v2A5D0h299B5v2A5D0h2954Ev2A5D0h29435v2A5D0h292
01v2A5D0h290E8v2A6E9h28EB4v2A803h28D9Bv2A803h28C
81v2A91Dh28B67v2A91Dh28A4Dv2A91Dh28934v2A91Dh287
v2A91Dh283B3v2AA36h27F4Dv2AA36h27AE6v2AA36h2767F
v2A91Dh27565v2A91Dh2744Cv2A91Dh2744Cv2A803h27332
v2A6E9h26FE5v2A5D0h26FE5v2A5D0)]

[CanvasStroke, (ZObjStrokeV02n2=YH2E0V234K2XH2E0V
234)]

[IPress,Tool:Masking:Inverse]

[IPress,Transform:Move]

[CanvasClick, 665, 380, 665,300]

//[note, [val, (coordy(67)*920)/227]]

[CanvasClick, 665, 380,
665, 380+(((coordy(65)*920)/227)-530)]

[IPress,Transform:Draw Pointer]

[IPress,Tool:Masking:MaskAll]

[IPress,Tool:Masking:Inverse]

]

//proiectare poza

[IPress,Tool:SubTool:PM3D_FemaleHead_2]

[ISet,Tool:Geometry:SDiv,4]

[IPress,Brush:Standard]

[FileNameSetNext, [StrMerge, "C:\Users\Andrei\Desk
top\licenta\poze2\img_", [var, contor], ".jpg"]]

[IPress,Texture:Import]

[IPress, [StrMerge, "Texture: img_", [val, contor]]]

[IPress,Texture:Add To Spotlight]

[IPress,Preferences:LightBox:LightBox]

[IPress,Transform:Draw Pointer]

```

```

[IPress,Draw:Mrgb]

[ISet,Draw:Draw Size,1000]

//[CanvasZoomSet,0.90]

//[CanvasPanSet,1045,435]

[CanvasPanSet,1045,435]

[IUnPress,Transform:EditSpotlight]

[IUnPress,Draw:Zadd]

[IUnPress,Transform:Activate Symmetry]

[CanvasStroke,(ZObjStrokeV02n370=h2FC81v6AA7h2FC
81v6B82h2FC81v6D37h2FC81v6E11h2FC81v6EEBh2FC81v7
0A0h2FC81v71B1h2FC81v72C2h2FC81v739Dh2FC81v74E4h
2FC81v7663h2FC4Av784Eh2FC14v7A3Ah2FBDDv7BB8h2FBA
6v7CFFh2FBA6v7E11h2FBA6v7F22h2FBA6v8033h2FBA6v81
7Ah2FBA6v828Bh2FBA6v83D3h2FBA6v851Bh2FBA6v8662h2
FBA6v87AAh2FBA6v88F2h2FBA6v8A39h2FBA6v8B81h2FBA6
v8CC8h2FBA6v8DD9h2FBA6v8EEBh2FBA6v8FC5h2FBA6v910
Dh2FBA6v9254h2FBA6v93D3h2FBA6v9551h2FBA6v9662h2F
BA6v9773h2FBA6v9884h2FBA6v99CCh2FBA6v9B13h2FBA6v
9C91h2FBA6v9DD9h2FBA6v9EEAh2FBA6vA032h2FBA6vA143
h2FBA6vA2C1h2FBA6vA476h2FB70vA5F4h2FB39vA7A9h2FB
03vA927h2FACCvAADCh2FACCvACC8h2FA5FvAE7Ch2F9F2vB
031h2F984vB179h2F917vB2C1h2F917vB43Fh2F917vB5BDh
2F917vB73Bh2F917vB8BAh2F917vBA38h2F917vBBB6h2F91
7vBDA2h2F917vBF8Dh2F917vC1AFh2F917vC3D1h2F917vC5
BDh2F917vC7DFh2F917vC994h2F917vCB48h2F917vCD34h2
F917vCF1Fh2F917vD10Bh2F94EvD32Dh2F984vD54Fh2F9F2
vD771h2FA95vD9CAh2FB03vDBB5h2FB70vDD33h2FBDDvDE7
Bh2FC14vDFC3h2FC4AvE141h2FCB7vE2BFh2FD25vE43Eh2F
DC8vE585h2FE6CvE6CDh2FF10vE8B8h2FFB4vEA6Dh30021v
EC59h300C5vEE44h30132vEF8Ch3019FvF09D

h3020DvF1AEh30243vF288h3027AvF399h3027AvF518h302
7AvF696h3027AvF881h3027AvFADAh3027AvFD69h3027Av1
0066h3027Av10362h3027Av10628h3027Av10925h3027Av1
0BEh3027Av10EB0h3027Av11176h3027Av11406h3027Av1
165Eh3027Av118EEh3027Av11B46h3027Av11E0Ch3027Av1
2109h3027Av12405h3027Av127A6h3027Av12A6Bh3027Av1
2D31h3027Av12FC1h3027Av13219h302B0v13516h302E7v1
3812h3031Ev13AA2h30354v13D31h30354v13FF7h30354v1
42F3h30354v14694h30354v149FDh30354v14CFAh3038Bv1
4FC0h303C1v15218h303F8v1543Ah3042Fv1565Dh3042Fv1
5848h3042Fv159FDh3042Fv15BB2h3042Fv15CF9h3042Fv1
5E0Ah3042Fv15F89h3042Fv160D0h3042Fv1624Fh3042Fv1
6403h3042Fv16582h3042Fv16737h3042Fv168EBh3042Fv1
6AA0h3042Fv16C8Ch3042Fv16EAEh3042Fv1713Dh3042Fv1
7403h30465v176FFh304D3v179FCh30540v17CC2h305ADv1
7F1Bh3061Av1813Dh30651v1835Fh30687v185B7h306BEv1
8847h306BEv18B43h306BEv18E09h306BEv190CFh306BEv1
93CCh306BEv19691h306BEv199FBh306F5v19D2Eh3072Bv1
A098h30798v1A3CBh30806v1A691h3083Cv1A957h30873v1
AC1Dh30873v1AF50h308A9v1B2F0h308E0v1B6C7h30917v1
BA9Eh3094Dv1BE08h30984v1C13Bh309F1v1C46Eh30A5Ev1
C6FDh30ACCv1C9FA

h30B02v1CCC0h30B6Fv1CFBCh30C13v1D326h30CB7v1D5EC
h30D5Bv1D8B2h30D91v1DB0Ah30D91v1DD63h30D91v1E029

```

```

h30D91v1E326h30D91v1E622h30D91v1E955h30D91v1EC52
h30D91v1EF4Eh30D91v1F2B8h30D91v1F5B4h30D91v1F8B1
h30D91v1FBADh30D5Bv1FE3Dh30D24v200CCh30CEEv20392
h30C80v20658h30C4Av2091Eh30BDDv20C51h30B39v20F4D
h30ACCv21213h30A5Ev214D9h309F1v21732h30984v2198A
h30917v21C1Ah308A9v21EA9h3083Cv221A6h307CFv224D9
h30762v2279Fh306BEv22AD2h30651v22DCEh3061Av230CB
h305E4v233FEh30576v236FAh30509v239C0h30465v23CBD
h303C1v23FF0h3038Bv24359h3031Ev24767h3027Av24B75
h301D6v24F4Ch30132v252ECh3008Ev25656h30058v25952
h2FFEBv25D29h2FF47v260C9h2FED9v264A0h2FE6Cv268AE
h2FE36v26C18h2FE36v26FEEh2FE36v2738Fh2FE36v2772F
h2FE36v27A99h2FE36v27D95h2FE36v27FEEh2FE36v28247
h2FE36v2849Fh2FE36v2872Fh2FE36v28A2Bh2FE36v28D28
h2FE36v28FEEh2FE36v292B3h2FE36v29543h2FE36v29765
h2FE36v299BEh2FE36v29BA9h2FE36v29DCBh2FE36v29FED
h2FE36v2A1D9h2FE36v2A3FBh2FE36v2A61Dh2FE36v2A808
h2FE36v2AA2Ah2FE36v2AC4Ch2FEA3v2AE01h2FF47v2B023
h2FFEBv2B20Fh3008Ev2B38Dh300C5v2B542h300FCv2B6C0
h30169v2B875h30243v2BA97

h3031Ev2BD26h30465v2BFECCh305ADv2C2E9h306BEv2C5AF
h30806v2C875h308E0v2CACDh30984v2CD26h30A28v2CFB5
h30ACCv2D1D7h30B39v2D467h30BDDv2D689h30C80v2D874
h30CEEv2DA96h30D5Bv2DC4Bh30D91v2DE37h30D91v2E022
h30D91v2E20Dh30DC8v2E466h30DFFv2E652h30E35v2E83D
h30EA2v2E9BBh30F10v2EB03h30F7Dv2ECB8h30FEAv2EE6D
h31021v2F08Fh31021v2F27Ah31021v2F49Ch31021v2F6BE
h30FEAv2F8AAh30FB3v2FACCh30F7Dv2FCB7h30F10v2FEA3
h30ED9v30058h30EA2v3020Dh30E6Cv3038Bh30E6Cv30509
h30E6Cv30687h30E6Cv30806h30E6Cv30984h30E6Cv30ACC
h30E6Cv30C4Ah30E6Cv30DC8h30E6Cv30F7Dh30E6Cv31132
h30E6Cv312E7h30E6Cv3149Bh30E6Cv315E3h30E6Cv3172B
h30E6Cv31872h30E6Cv31983h30E6Cv31A94h30E6Cv31BDC
h30E6Cv31D24h30E6Cv31E6Bh30E6Cv31FB3h30E6Cv320C4
h30E6Cv3219Fh30E35v32279h30DFFv3238Ah30DC8v324D2
h30D91v32619h30D91v32798h30D91v328DFh30D5Bv32A27
h30D24v32B6Eh30CB7v32CB6h30C4Av32DFEh30C13v32F45
h30BDDv3308Dh30BDDv331D5h30BDDv3331Ch30BDDv3342D
h30BDDv3353Eh30BDDv33619h30BDDv3372Ah30BDDv3383B
h30BDDv33983h30BDDv33ACAh30BDDv33C12h30BDDv33D59
h30BDDv33EA1h30BDDv33FE9h30BDDv34130h30BDDv34278
h30BDDv343C0h30BDDv34507

h30BDDv3464Fh30BDDv34797h30BDDv348DEh30BDDv34A26
h30BDDv34B6Eh30BDDv34CB5h30BDDv34DFDh30BDDv34F45
h30BDDv350C3h30C13v35241h30C4Av353BFh30C80v3553E
h30CB7v35685h30CB7v35803h30CB7v3594Bh30CB7v35A93
h30CB7v35BA4h30CEEv35C48h30D5Bv35CB5h30DC8v35CEB
)]

//stergere poza si resetare model

[IPress,[StrMerge,"Texture: img_",[val,contor]]]

[IUnPress,Texture:Add To Spotlight]

[ISet,Texture:Texture Off,0]

[IPress,Texture:R]

[VarInc,contor]

//sticky off

```

```

[IPress,Transform:Move]

[CanvasClick, 700,290]

[CanvasClick, 725,290]

[IPress,Transform:Draw Pointer]

[IPress,Movie:Turntable]

[FileNameSetNext,[StrMerge,"C:\Users\Andrei\Desktop\licenta\filme\frames",[var,contorfilm],".mpg"]]

[IPress,Movie:Export]

[IKeyPress,'N',[IPress,Movie:Delete]]

[varinc,contorfilm]

[IPress,Tool:SubTool:PM3D_FemaleHead_2]

[CanvasStroke,(ZObjStrokeV03n11%p2D74934p1CF5158
PNNn-
FF52s1D0214Es1D0214Es1D0214EZ=Yh4E317v1624FK1Xh4
E431v1624Fh4E665v16369h4E77Ev16369h4E77Ev16483h4
E9B2v16483h4EACBv16483h4EBE5v1659Ch4ECFFv166B6h4
EE18v166B6h4EE18v166B6)]

[ISet,Edit:Tool:UndoCounter,1]

[IPress,Edit:Tool:DelUH]

[MemDelete,ZB_TextInputMem] //all done, delete
memblock]

/*End of ZScript*/

```

Anexa B

Crearea bazei de date în python

```
import cv2

import os

file = 1

contor=1

for movies in range (1,41):

    user = (movies-1)//4 + 1

    fata = (movies-1) % 4 + 1

    newpath = 'C:\\Users\\User-
practica\\Desktop\\poze\\filme\\user'+str(user)+
'\\fata'+str(fata)

    if not os.path.exists(newpath):

        os.makedirs(newpath)

    vidcap =
cv2.VideoCapture('frames'+str(movies)+'.mpg')

    success,image = vidcap.read()

    count = 1

    while success:

        cv2.imwrite(newpath +
"\\frame%d.jpg" % count, image)      # save frame
as JPEG file

        success,image = vidcap.read()

        count += 1

    print('done user '+str(user))
```

Anexa C

Estimarea în python

```
from skimage import feature

import cv2

import numpy as np

from sklearn.model_selection import GridSearchCV

from sklearn import svm

import os

labels=[]

labelstest=[]

labelstestadev=[]

date = np.zeros(59)

datetest=np.zeros(59)

files_all='C:\\Users\\Andrei\\Desktop\\licenta\\
poze\\user1'

index_files=0

for _,_,files in os.walk(files_all):

    for imagine in files:

        image=cv2.imread(files_all+"\\fata"+str(
index_files)+"\\"+imagine)

        crop_img=image[40:410, 240:500]

        gray = cv2.cvtColor(crop_img,
cv2.COLOR_BGR2GRAY)

        lbp = feature.local_binary_pattern(gray,
8,1, method='nri_uniform')

        hist,bins = np.histogram(lbp.ravel(),59)

        #return array cu value of the hist , si
bin_edges care e length de hist+1 pe care il
ignoram

        path=imagine

        l=len(path)

        imp=path[5:-4]

        labels.append(imp)

        date=np.vstack((date,hist))

    index_files+=1
```

```

date=np.delete(date,0,0)

date = date.astype("float")

#normalizare

dmin=np.amin(date,axis=0)

dmax=np.amax(date,axis=0)

for x in range(0,len(files)*(index_files-1)):

    for y in range(0,59):

        date[x][y]=(date[x][y]-
dmin[y])/(dmax[y]-dmin[y])

files_test_all='C:\\Users\\Andrei\\Desktop\\licenta\\fataltest'

index_files_test=0

for _,_,files in os.walk(files_test_all):

    for imagine in files:

        image=cv2.imread(files_test_all+'\\fata'+
str(index_files_test)+"\\"+imagine)

        # crop_img=image[40:410, 240:500]

        gray = cv2.cvtColor(image,
cv2.COLOR_BGR2GRAY)

        lbp = feature.local_binary_pattern(gray,
8,1, method='nri_uniform')

        hist,bins = np.histogram(lbp.ravel(),59)

        path=imagine

        l=len(path)

        imp=path[5:-4]

        labelstestadev.append(imp)

        datetest=np.vstack((datetest,hist))

    index_files_test+=1

datetest=np.delete(datetest,0,0)

datetest = datetest.astype("float")

for x in range(0,len(files)*(index_files_test-
1)):

    for y in range(0,59):

        datetest[x][y]=(datetest[x][y]-
dmin[y])/(dmax[y]-dmin[y])

#for c in [2**-5,2**-3,2**-
1,2**1,2**3,2**5,2**7,2**9,2**11,2**13,2**15]:

#

# for ga in [2**-15,2**-13,2**-11,2**-9,2**-
7,2**-5,2**-3,2**-1,2**1,2**3]:

parametrii= {'kernel':['rbf'], 'C': [2**13],
'gamma': [2**-3]}

clf=GridSearchCV(estimator=svm.SVR(),param_grid=
parametrii,n_jobs=-1)

#clf = svm.SVR(kernel= 'rbf', C= 8192
,gamma=0.125)

for i in range(len(labels)):

    labels[i] = float(labels[i])

for i in range(len(labelstestadev)):

    labels[i] = float(labelstestadev[i])

clf.fit(date,labels)

labelstest=clf.predict(datetest)

for i in range(len(labelstest)):

    labels[i] = float(labelstest[i])

print(labelstest)

print("*****")

print(labelstestadev)

#print(clf.score(datetest,date))

#print(gridsearch)

#print( np.unravel_index(gridsearch.argmin(),
gridsearch.shape))

#svr.predict(datel)

print('Best score for data:', clf.best_score_)

#print('Best C:',clf.best_estimator_.C)

#print('Best
Kernel:',clf.best_estimator_.kernel)

#print('Best Gamma:',clf.best_estimator_.gamma)

```