X. Detectia si recunoasterea gesturilor si posturii corporale/faciale cu ajutorul unui senzor Kinect.

X.1. Senzorul Kinect. Facilitati si unelte.

In continuare se prezinta o abordare bazata pe senzorul Kinect, care, pe langa imaginea 2D color furnizeaza si o imagine de profunzime generata prin intermediul senzorului optic integrat. SDK-ul Kinenct pune la dispozitia utilizatorilor unelte utile pentru dezvoltarea de aplicatii de urmarire a capului si a partilor corpului si pentru recunoasterea de gesturi cum ar fi: estimarea skeletonului persoanei [20], detectia si urmarirea fetei / capului [21] etc.

Skeletonul este modelat ca si o lista de puncte 3D (X, Y, Z) corespunzatoare la 20 (Kinect-v1) de tipuri de incheieturi („joints") [30]: JointType.HandLeft, JointType.WristLeft, JointType.ElbowLeft etc.

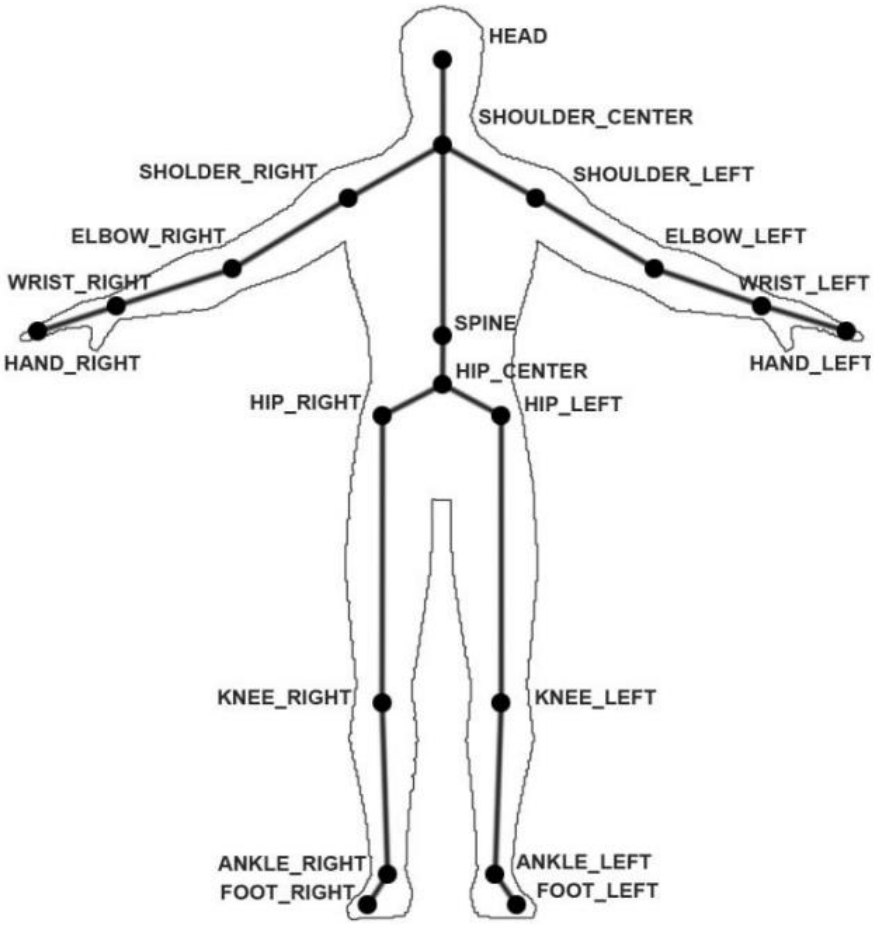


Fig. X.1. Modelul de reprezentare al skeletonului unei persoane (20 incheieturi) [30]

http://www.youtube.com/watch?v=YTBvjLGDluY

Pe baza reprezentarii skeletonului se pot infera gesturi ale persoanei urmarite, cea mai utilizata thenica fiind DTW (Dynamic Time Warping). In [22] este descrisa implementarea unei unelte pentru recunosaterea gesturilor aferente partii superioare a trunchiului + membre prin aplicarea metodei DTW pe coordonatele 2D ale incheieturilor (proiectia de tip front-view a punctelor 3D). In [23] este raportata o metoda similara bazata pe coordonatele 3D.

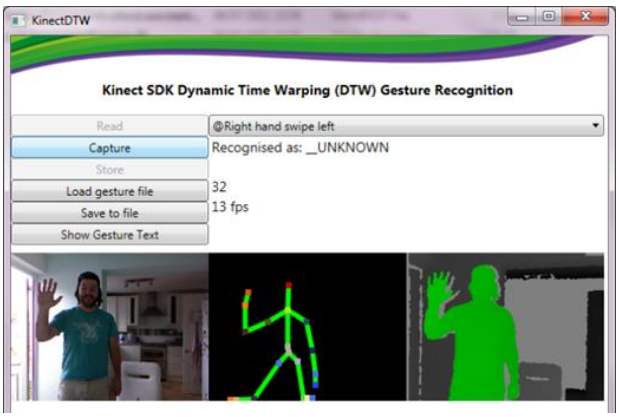
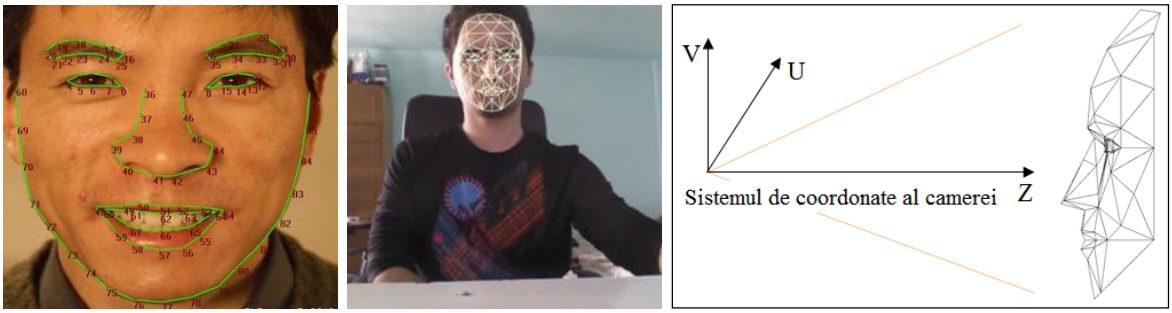


Fig. X.2. Unealta (toolkit) de recunoastere a gesturilor pe baza skeletonului si DTW [22].

Odata cu aparitia Kinect SDK 1.5 a fost lansat si un framework pentru **detectia si urmarirea fetei**. Acest framework foloseste doua librarii : Microsoft.Kinect.Toolkit si libraria derivata FaceTracking [21]. Ceea ce ne pune la dispozitie este un set de **87 puncte 2D** de pe fata (fig. X.3.a). Aditional rutina de Face Tracking mai identifica alte **13 puncte** (inferate din cele existente, ca de exemplu: centrul ochiului, centrul nasului), care nu sunt ilustrate pe fig. X.3. Aceste puncte pot fi folosite ca date senzoriale primare pentru diverse aplicati de tip HMI (Uman Machine Interface), ca de exemplu: recunoasterea unei anumite expresii faciale), detectia si urmarirea gesturilor realizate cu mainile, cu capul etc. Avand pozitia varfului capului (Joint of the Head), senzorul Kinect recunoaste o suprafata virtuala („masca parametrizata” in jurul fetei (fig. X.3.b) pe baza modelului CANDIDE-3 [26], iar clasele de Face Tracking ne returneaza un vector de puncte 2D (proiectia de tip front-view a punctelor 3D) sau 3D prin extrapolare (conform modelului CANDIDE-3).



a. b. c.

Fig. X.3. Punctele de pe fata detectate prin unealta FaceTracking (Kinekt Toolkit) [21]: a. puncte 2D; b. suprafata virtuala 3D obtinuta prin extrapolare din 2D conform modelului CANDIDE-3.

Modulul FaceTracking ne pune la dispozitie si obiectul *frame.Rotation* ce contine cele 3 unghiuri de rotatie *(pitch, yaw si roll)* ale capului in jurul celor 3 axe de coordonate ale senzorului Kinect (Fig. X.3.c) cat si obiectul *frame.Translation* ce contine cele trei componente ale vectorului de translatie *[Tx, Ty, Tz]* ale capului relativ la acelasi sistem de coordonate. Prin analiza valorilor acestor vectori se poate infera pozitia si directia in care priveste subiectul uman (utila in aplicatiile de monitorizare a directiei de privire a persoanei ca de ex. in aplicatii ADAS).

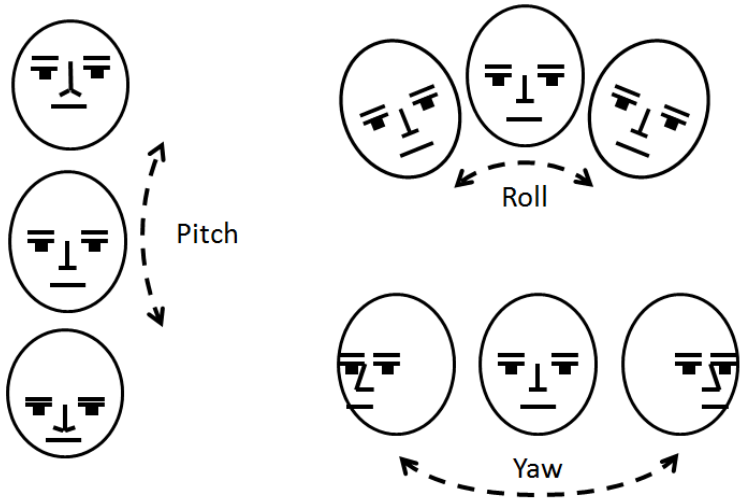


Fig. X.4. Estimarea orientarii capului relativa la sistemul de coordonate al senzorului [21].

Un alt domeniu de interes, in special in domeniul aplicatiilor de supravegherea starii conducatorului autovehiculului dar si in interfetele perceptuale este cel al **recunoasterii gesturilor faciale**. Gesturile faciale constau in **schimbarea dinamica a aparentei faciale** si s-ar putea infera pe baza unor trasaturi care masoara variabilitatea unor componente faciale: pozitia buzelor, sprancenelor, mandibulei etc. Aceastea se pot realiza pe baza modelului CANDIDE-3 [26] implementat in [21]. Acest model pune la dispozitia utilizatorului un set de **9 unitati de forma SU (Shape Units)** si **6 unitati de animatie AU (Animation Units).** In timp ce majoritatea eforturilor de valorificare a capabilitatilor Kinect s-au orientat spre aplicatii comeciale de animatie (avatari) [29], [30], o utlizare mai interesanta ar fi in aplicatii de monitorizare a starii subiectilor umani cu aplicabilitate potentiala in sisteme ADAS (monitorizarea starii de atentie a conducatorului auto) sau interfete perceptuale (detectia starii emotionale etc.)

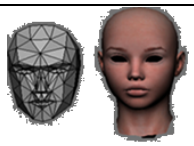
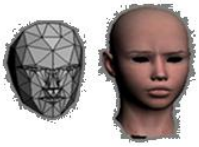
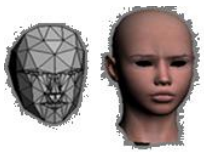
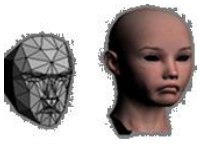
Unitatile de forma (SU) estimeaza forma particulara a capului/feţei unui subiect uman: pozitia, intr-o stare neutra, a gurii, sprancenelor, ochilor etc.

Unitatile de animatie (AU) reprezinta abaterile SU de la pozitia neutra si sunt urmarite in imagini succesive. Marimile AU constau in valori normalizate in intervalul -1 ... +1.

Tabel X.1. Trasturile SU ale modelului CANDIDE-3 ([21], [26])

Index [21], [26]	**Feature type**	**Tip trasatura**
SU-0	Head height	Inaltime capului
SU-1	Eyebrows vertical position	Pozitia verticala a sprancenelor
SU-2	Eyes vertical position	Pozitia verticala a ochilor
SU-3	Eyes, width	Latimea ochilor
SU-4	Eyes, height	Inaltimea ochilor
SU-5	Eye separation distance	Distanta dintre ochi
SU-8	Nose vertical position	Pozitia verticala a nasului
SU-10	Mouth vertical position	Pozitia verticala a gurii
SU-11	Mouth width	Latimea gurii

Tabel X.2. Unitatile de Animatie (AU) si semnificatia lor [21]

AU [21]	**AU [26]**	**Semnificatie**	**Ilustrare fata-virtuala / avatar**	**Interpretare valori AU**
		Fata neutra		AU_i =0, i=0 .. 5
AU0	AU10	Pozitie buza superioara (Upper Lip Raiser)		0 = neutra, dinti acoperiti 1 = descoperire totala dantura -1 = buze comprimate (maxim)
AU1	AU26 AU27	Coborare mandibula (Jaw Lowerer)		0 = inchisa 1 = deschisa total -1 = inchisa (≈ 0)
AU2	AU20	Forma buze/gura (Lip Stretcher)		0 = neutra 1 = latire maxima (joker's smile) -0.5= rotunjite (pout) -1 = rotunjite la maxim (kissing mouth)
AU3	AU4	Pozitie sprancene (Brow Lowerer)		0 = neutra -1 = ridicate aproape in totalitate +1= coborate la maxim (la limita ochilor)
AU4	AU13 AU15	Curbura buze (Lip Corner Depressor)		0 = neutra -1 = zambet fericit +1= trista
AU5	AU2	Ridicare exterior sprancene (Outer Brow Raiser)		0 = neutra -1 = coborate (fata trista) +1= ridicate (surprindere)

X.2. Recunosate gesturilor corporale dinamice (trunchi, cap si dinamica privirii)

In cele ce urmeaza se prezinta algoritmi de recunoastere a unor gesturi corporale elementare realizate cu mainile si cu capul si implicit a dinamicii directiei de privire. Metoda utilizata pentru recunoastere se bazeaza pe algoritmul DTW (Dynamic Time Warping).

Algoritmul *Dynamic Time Warping* [24] este foarte cunoscut in multe domenii. A fost introdus in anii 60 si a fost foarte folosit in anii 70 in aplicatii de recunoastere a vorbirii. In zilele noastre este folosit pentru: **recunoasterea scrisului de mana si a semnaturii online, recunoasterea semnelor, recunoasterea gesturilor, in animatia pe calculator, in supraveghere, alinierea secventelor de proteine, procesarea semnalelor audio si a semnalelor in general**. Algoritmul DTW si-a castigat popularitatea de a fi extrem de eficient prin masurarea similaritatii in timp a doua serii, masura ce minimizeaza efectul deplasarii si distorsiunilor in timp permitand transformari elastice ale serilor pentru a putea detecta forme asemanatoare in faze diferite.

Dandu-se doua serii $X = (x_1, x_2, ...x_N)$; $N \in \mathbf{N}$ and $Y = (y_1, y_2, ...y_M)$; $M \in \mathbf{N}$ reprezentate de secventa de valori (sau curbe reprezentate printr-o secventa de varfuri), DTW gaseste solutia optima in O(MN), timp ce poate fi imbunatatit prin diferite tehnici.

Daca seventele iau valori din spatiul de trasaturi Φ, atunci pentru a compara cele doua secvente X, $Y \in \Phi$ trebuie sa folosim masura de distanta locala care este definita ca fiind o functie :

$$d: \Phi \times \Phi \rightarrow R \geq 0 \quad (1)$$

d avand o valoare mica atunci cand secventele sunt similare si o valoare mare cand sunt diferite. Cum DTW este un algoritm de programare putem numi aceasta functie de distanta ca fiind functia de cost si taskul de aranjare optima a secventelor va deveni taskul de aranjare a punctelor secventei pentru a minimiza functia de cost.

Exemplu: un element yi din lista Y pentru recunoasterea gesturilor realizate cu mana ar arata astfel:

Tabel X.3 Exemplu cu valorile normalizate ale coordonatelor punctelor încheieturilor membrele superioare pentru o instanță temporală i a unui gest realizat cu mâinile:

i	U_i^N	V_I^N	Z_i^n
1 - HandLeft	-0,81115095467971	0,76509962541820	-1,0184617322390
2 - WristLeft	-0,83346899225044	0,55132286420475	-0,9685865898909
3 - ElbowLeft	-0,95456558370226	-0,0633539243305	-0,6606921065685
4 - ElbowRight	0,545383953584578	-0,8731627336406	-0,1662174573874
5 - WristRight	0,608084590373526	-1,4554127018135	-0,5075000844907
6 - HandRight	0,642824381675729	-1,7058843790186	-0,6661649722653

Setul Y al tuturor instantelor yi ar putea reprezenta setul de antrenare al tuturor pozitiilor incheieturilor pentru un gest q.

Setul X ar fi instantele xi al al tuturor pozitiilor incheieturilor pentru un gest care se doreste a fi recunoscut.

Algoritmul începe prin construirea matricei de distanță C ∈ R^{NxM} reprezentând toate perechile distanță între X și Y. Această matrice de distanță este numită matricea de cost local a două secvențe X și Y:

$$C_t \in \mathbb{R}^{N\,x\,M} : c_{i,j} = ||x_i - y_j||, i \in [1 : N], j \in [1: M] \quad (2)$$

Odată ce matricea de cost local este construită, algoritmul găsește calea de aliniere care trece prin "văile" de cost minim. Acest drum de aliniere (warping path / warping function) definește corespondența unui element x_i ∈ X la y_j ∈ Y urmărind condiția de limită care asignează primul și ultimul element din X, Y unul altuia.

Calea de aliniere construită de DTW este o secvență de puncte p = (p1,p2, ..., pk) cu p_l = (pi, pj) ∈ $[1: N]\ x\ [1: M]$ pentru l ∈ $[1: k]$ care trebuie să satisfacă următoarele condiții :

1. **Criteriul limită** (de frontieră): pl = (1,1) si pk = (N, M). Punctul de start și final al drumului trebuie să fie primul și ultimul punct din secvențele aliniate
2. **Monotonicitatea condiției** : n1 ≤ n2 ≤ ... ≤ nk și m1≤ m2 ≤ ... ≤ mk. Această condiție păstrează ordinea în timp a punctelor.
3. **Condiția lungimii pasului (step size condition)** : acest criteriu limitează posibilitatea salturilor mari (deplasare în timp) în timpul alinierii secvențelor. Pentru început vom folosi funcția p_{l+1}- p_l ∈ {(1, 1), (1, 0), (0, 1)}.

Funcția de cost asociată cu calea de cost minim (warping path) va fi:

$$c_p(X, Y) = \sum_{l=1}^{K} c(x_{n_l}, y_{ml}) \quad (3)$$

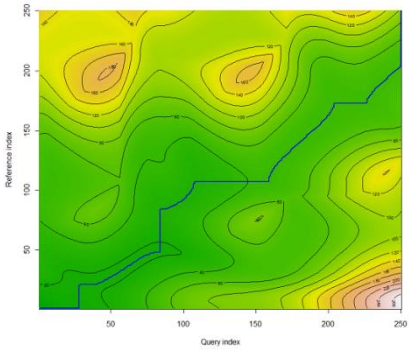


Fig. X.5. Ilustrarea matricii de cost un forma unei harti termice/elevatie si a caii de cost minim [24]

Drumul de aliniere de cost minim se va numi drumul de aliniere minim și se va nota cu P^*. Pentru a găsi o cale de aliniere de cost minim trebuie să testăm toate căile de aliniere dintre X și Y. Această soluție este foarte costisitoare din punct de vedere computațional deoarece căile de aliniere posibile cresc exponențial ca număr atunci când X și Y cresc liniar. Pentru a preveni această provocare se folosește *Programarea Dinamica* [31] pentru a implementa algoritmul DTW. Complexitatea va fi deci **O(MN).**

Partea de progamare dinamică a algoritmului DTW folosește funcția de distanță :

$$\textbf{DTW (X, Y)} = \boldsymbol{C_p} * (\boldsymbol{X}, \boldsymbol{Y}) = \textbf{min}\{\, \boldsymbol{C_p}(\boldsymbol{X}, \boldsymbol{Y}), \boldsymbol{p} \in \boldsymbol{P^{N\,x\,M}}\} \qquad (4)$$

unde $\boldsymbol{P^{NxM}}$ este mulțimea tuturor căilor de aliniere posibile și construiește **matricea de cost acumulat** sau **matricea de cost global** definită mai jos:

1. Primul rand : $\boldsymbol{DTW}(\mathbf{1}, \boldsymbol{j}) = \sum_{\boldsymbol{k}=\mathbf{1}}^{\boldsymbol{j}} \boldsymbol{c}(\boldsymbol{x_1}, \boldsymbol{y_k}), \boldsymbol{j} \in [\mathbf{1}, \boldsymbol{M}]$
2. Prima coloana : $\boldsymbol{DTW}(\boldsymbol{i}, \mathbf{1}) = \sum_{\boldsymbol{k}=\mathbf{1}}^{\boldsymbol{j}} \boldsymbol{c}(\boldsymbol{x_k}, \boldsymbol{y_1}), \boldsymbol{i} \in [\mathbf{1}, \boldsymbol{N}]$
3. Toate celelalte elemente sunt :

$\boldsymbol{DTW}(\boldsymbol{i}, \boldsymbol{j}) = \textbf{min}\{\, \boldsymbol{DTW}(\boldsymbol{i}-\mathbf{1}, \boldsymbol{j}-\mathbf{1}), \boldsymbol{DTW}(\boldsymbol{i}-\mathbf{1}, \boldsymbol{j}), \boldsymbol{DTW}(\boldsymbol{i}, \boldsymbol{j}-\mathbf{1})\} + \boldsymbol{c}(\boldsymbol{x_i}, \boldsymbol{y_j}), \boldsymbol{i} \in [\mathbf{1}, \boldsymbol{N}], \boldsymbol{j} \in \mathbf{1}, \boldsymbol{M}])$. (5)

Timpul necesar construirii matricii este O(NM) care este egal cu costul următorului algoritm, unde intrările X și Y sunt serile de timp, iar C este matricea de cost local și reprezintă toate distanțele posibile dintre X și Y.

În continuare se va prezenta pseudocodul pentru matricea de cost acumulat și pentru găsirea drumului minim [24]:

```
1:    n := |X|
2:    m := |Y|
3:    dtw[] := new [n x m]
4:    dtw(0,0) := 0
5:    for i = 1; i ≤ n; i++ do
6:          dtw(i,1) := dtw (i-1,1) + c(i,1)
7:    end for
8:    for j = 1;   j ≤ m; j++ do
9:          dtw(1,j) := dtw(1,j-1) + c(1,j)
10:   end for
11:   for i = 1; i ≤ n; i++ do
12:         for  j = 1; j ≤ m; j++ do
13:               dtw(i,j) := c(i,j) + min {dtw(i-1, j); dtw(i,j-1); dtw(i-1,j-1)}
14:         end for
15:   end for
```

```
16:     return dtw
```

Odată ce matricea de cost acumulat a fost construită drumul minim poate fi găsit prin simpla parcurgere înapoi de la punctul $p_{end} = (M, N)$ până la $p_{start} = (1, 1)$ folosind metoda greedy descrisă de următorul algoritm [13].

```
1:      path[] := new array
2:      i = rows(dtw)
3:      j = columns(dtw)
4:      while (i >1 ) & (j > 1) do
5:              if i == 1 then
6:                      j = j - 1
7:              else if j == 1 then
8:                      i = i -1
9:              else
10:                     if dtw(i-1, j) == min {dtw(i-1,j); dtw(i,j-1); dtw(i-1;j-1)}
11:                     then
12:                             i = i - 1
13:                     else if dtw(i, j-1) == min {dtw(i-1, j); dtw(i, j - 1); dtw (i-1, j-1)}
14:                     then
15:                             j = j - 1
16:                     else
17:                             i = i - 1; j = j-1
18:                     end if
19:                     path.add((i,j))
20:             end if
21:     end while
22:     return path
```

Optimizări DTW (scalarea secventelor ⇒ reducerea dimensionalitatii (secventelor)) - vezi [24]

X.3. Recunoașterea gesturilor folosind algoritmul DTW

Recunoașterea gesturilor corporale

1. Se alege un set de puncte de control / încheieturi (joints) specific categoriei de gesturi care se dorește a fi recunoscut
2. Se alege un punct de referință (mijlocul segmentului care unește cei doi umeri) $\Rightarrow (U_r, V_r, Z_r)$
3. Se centrează coordonatele (U_i, V_i, Z_i) ($i = 1 .. J$ / J - numărul de puncte de control) ale încheiturilor alese față de punctul de referință:

$$U_i^c = U_i - U_r$$
$$V_i^c = V_i - V_r$$
$$Z_i^c = Z_i - Z_r$$

4. Se normalizează coordonatele centrate față de o caracteristică antropomorfică specifică persoanei (distanța euclidiană dintre încheieturile umerilor) $\Rightarrow$ set de puncte centrate și normalizate $(U_i^n, V_i^n, Z_i^n) \Rightarrow$ setul/vectorul de trăsături pentru o instanță temporală :

$$\mathbf{v}_{3D\text{-}skeleton} = \{(U_i^n, V_i^n, Z_i^n), i = 1 .. J\} \qquad (6)$$

5. Se crează secvența de referință (antrenare) pentru fiecare gest care se dorește a fi recunoscut. Se înregistrează M instanțe temporale ale pozițiilor succesive ale corpului/membrelor în timpul gestului respectiv:

$$Y = \{\cup_{k=1}^{M}(U_i^n, V_i^n, Z_i^n), i = 1 .. J\} \qquad (7)$$

6. Se creează secvențe de referință pentru toate gesturile care se doresc a fi recunoscute: Y_q.
7. Sa apelează procedura de recunoaștere DTW pe o secvență de test X. Această procedură încearcă să găsească cea mai bună potrivire (calea de cost minim / warping path) a unui subset de K instanțe temporale ale lui X peste fiecare set Y_q.

Tabel X.3 Exemplu cu valorile normalizate ale coordonatelor punctelor încheieturilor membrel superioare pentru o instanță temporală i a unui gest realizat cu mâinile:

i	U_i^n	V_I^n	Z_i^n
1 - HandLeft	-0,81115095467971	0,76509962541820	-1,0184617322390
2 - WristLeft	-0,83346899225044	0,55132286420475	-0,9685865898909
3 - ElbowLeft	-0,95456558370226	-0,0633539243305	-0,6606921065685
4 - ElbowRight	0,545383953584578	-0,8731627336406	-0,1662174573874
5 - WristRight	0,608084590373526	-1,4554127018135	-0,5075000844907
6 - HandRight	0,642824381675729	-1,7058843790186	-0,6661649722653

Recunoașterea gesturilor faciale

1. Se crează setul/vectorul de trăsături al unităților de animație (AU) pentru o instanță temporală: $\mathbf{v}_{2D\text{-}face\text{-}AU} = (AU_0, AU_1, AU_2, AU_3, AU_4, AU_5)$ (8)

2. Se creează secvența de referință (antrenare) pentru fiecare gest care se dorește a fi recunoscut. Se înregistrează M instanțe temporale ale valorilor succesive ale vectorului de trăsături (cele 6 unități de animație, AU, ale feței) în timpul gestului respectiv : $Y = \bigcup_{K=1}^{M}(AU_{0,}AU_{1,}AU_{2,}AU_{3,}AU_{4,}AU_{5})$ (9)

3. Se creează secvențe de referință pentru toate gesturile care se doresc a fi recunoscute : Y_q.

4. Se apelează procedure de recunoaștere DTW pe o secvență de test X.

Recunoașterea miscărilor dinamice ale capului

1. Se crează setul/vectorul de trăsături al unghiurilor de rotație pentru o instanță temporală: $\mathbf{v}_{Head\_Rotation} = (X, Y, Z)$ (8')

2. Se creează secvența de referință (antrenare) pentru fiecare mișcare care se dorește a fi recunoscută. Se înregistrează M instanțe temporale ale valorilor succesive ale vectorului de trăsături (cele 3 unghiuri de rotație ale capului) în timpul miscării respective : $Y = \bigcup_{K=1}^{M}(X, Y, Z)$ (9')

3. Se creează secvențe de referință pentru toate mișcările capului care se doresc a fi recunoscute : Y_q.

4. Se apelează procedure de recunoaștere DTW pe o secvență de test X.

Recunoașterea gesturilor dinamice făcute cu degetele mâinii

1. Se alege ca și vector de trăsături setul celor 5 puncte corespunzătoare vârfurilor degetelor
2. Se alege un punct de referință centrul palmei $\Rightarrow (U_r, V_r, Z_r)$
3. Se centrează coordonatele (U_i, V_i, Z_i) ($i = 1$ .. J / J - numărul de puncte de control) ale vârfurilor degetelor față de centrul palmei:

$$U_i^c = U_i - U_r$$
$$V_i^c = V_i - V_r$$
$$Z_i^c = Z_i - Z_r$$

4. Se normalizează coordonatele centrate față de o caracteristică antropomorfică specifică persoanei (ex. distanța euclidiană dintre centrul palmei și cotul persoanei) $\Rightarrow$ set de puncte

centrate și normalizate $(U_i^n, V_i^n, Z_i^n) \Rightarrow$ setul/vectorul de trăsături pentru o instanță temporală :

$$\mathbf{v}_{3D\text{-}fingers} = \{(U_i^n, V_i^n, Z_i^n), i = 1 .. J\} \quad (6')$$

5. Se crează secvența de referință (antrenare) pentru fiecare gest care se dorește a fi recunoscut. Se înregistrează *M* instanțe temporale ale pozițiilor succesive ale degetelor în timpul gestului respectiv:

$$Y = \{\cup_{k=1}^{M}(U_i^n, V_i^n, Z_i^n), i = 1 .. j\} \quad (7')$$

6. Se creează secvențe de referință pentru toate gesturile care se doresc a fi recunoscute: Y_q.
7. Sa apelează procedura de recunoaștere DTW pe o secvență de test *X*. Această procedură încearcă să găsească cea mai bună potrivire (calea de cost minim / warping path) a unui subset de *K* instanțe temporale ale lui *X* peste fiecare set Y_q.

Nota: detectia varfului degetelor nu este inclusa in toolkit-ul Kinect dar se pot folosi librarii 3-rd party:

[32] F. Trapero Cerezo 3D Hand and Finger Recognition using Kinect, Universidad de Granada (UGR), Spain (cited May 2013), https://www.youtube.com/watch?v=rrUW-Z3fHkk

http://cvrlcode.ics.forth.gr/handtracking/,
http://users.ics.forth.gr/~argyros/research/kinecthandtracking.htm

https://social.msdn.microsoft.com/Forums/en-US/55395acd-cb38-4193-9f9a-814480c98b65/3d-hand-tracking?forum=kinectsdk

Rezultate

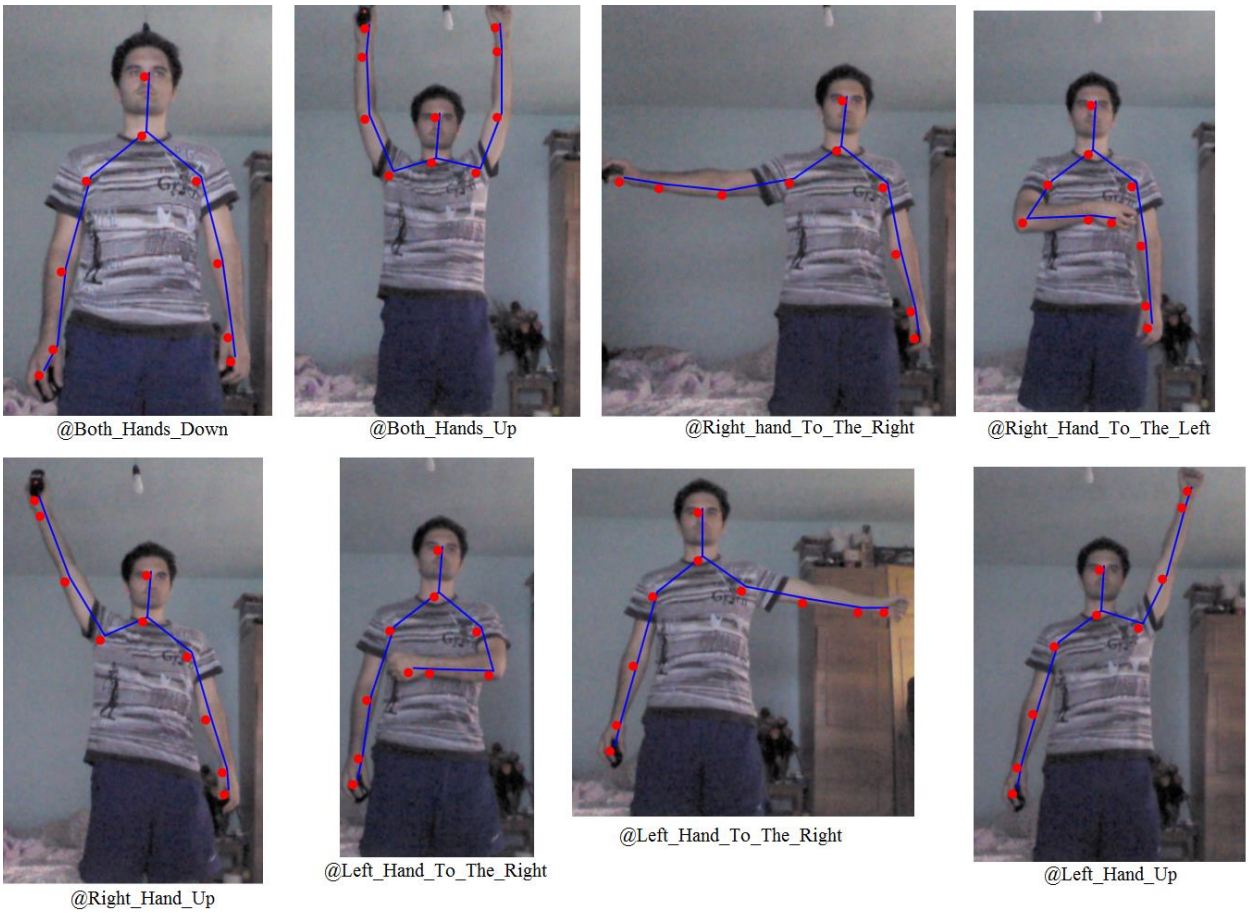


Fig. X.6 Posturi/Gesturi corporale dinamice înregistrate și testate

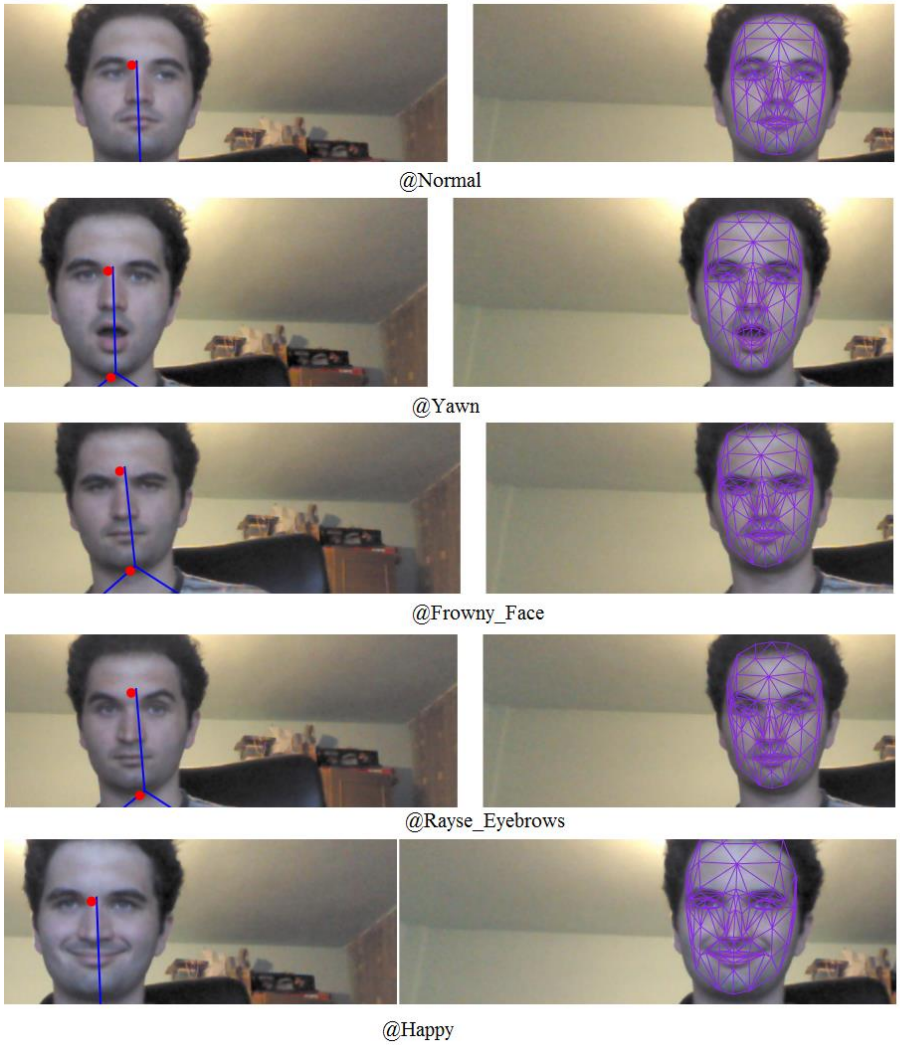


Fig. X.7 Posturi/Gesturi dinamice faciale înregistrate și testate

X.3. Recunosate gesturilor corporale statice (trunchi, fata, mana/degete)

- **Trasaturi corporale: Skeletonul** este modelat ca si o lista de puncte 3D (X, Y, Z) corespunzatoare la **20 de tipuri de incheieturi** („joints") – centrate si normalizate
- **Trasaturi faciale: 6** unitati de animatie AU **(Animation Units)** – valori normalizate (-1 ..1)
- **Trasaturi ale palmei** – pozitia 3D varfului degetelor – valori centrate si normalizate

Centrarea si normalizarea coordonatelor pt. incheieturi si degete – identic ca si la pct. X.2 (vezi recunoasterea gesturilor dinamice cu DTW).

X.3.1. Detectarea mâinilor și a degetelor

F. Trapero Cerezo (3D Hand and Finger Recognition using Kinect)[32] propune o abordare de recunoaștere a conturului mâinilor și a degetelor folosind un senzor Kinect. Dispozitivul Kinect ne permite obținerea unei imaginii de adâncime, în locul unei imaginii RGB normale, ceea ce ne dă mai multe informații față de o cameră normală. Aceste tipuri de imaginii sunt mult mai potrivite pentru urmărire, și în plus putem reprezenta punctele relevante într-un spațiu 3D.

Fluxul funcționalității de recunoaștere a mâinilor și a degetelor [32]:

1. Generearea masca pt. pixelilor apropiați (pe baza imaginii de adancime)
2. Reducerea zgomotului din imagine
3. Clasificarea punctelor ce aparțin mâinii
4. Diferențierea mâinilor și găsirea conturului
5. Alocarea punctelor interioare
6. Găsirea centrului palmei
7. Găsirea vârfurilor degetelor
8. Alocarea punctelor într-un spațiu 3D

1. Generare mascapentru pixelii apropiați $\Rightarrow$ ROI 3D $\Rightarrow$ ROI 2D (pixeli utili) (o imagine 2D binara ce va contine doar pixelii care apartin mainii)

Pentru început trebuie să construim o matrice a pixelilor utili (2D). Vom păstra doar pixeli a caror coordonate 3D sunt la o distanță cuprinsă în intervalul [minDist..maxDist] față de cameră.

Pentru a alege pixelii utili se pot folosi două metode :

- adâncime absolută - un pixel este aproape dacă adâncimea sa este mai mică decât o valoare constantă

- adâncime relativă - se calculează adâncimea minimă după care se adaugă o valoare constantă la această adâncime pentru a afla adâncimea maximă. Prin urmare dacă, adâncimea pixelului este cuprinsă între cele două valori tocmai găsite, atunci pixelul este apropiat. Această metodă este mai bună decât metoda bazată pe **adâncimea absolută** deoarece permite o mobilitate mai mare.

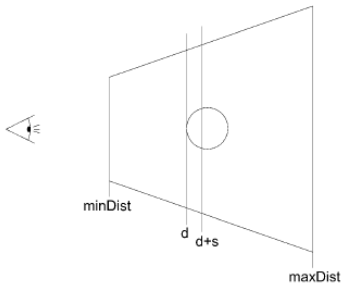


Fig. X.8 Distanța relativă [32]

2. Reducerea zgomotului din imagine

Pentru a reduce zgomotul din imagine se pot aplica transformările morfologice dilatare și eroziune pe imaginea 2D (inchidere) $\Rightarrow$ netezire contur

3. Clasificarea punctelor ce aparțin mâinii

Avem o matrice binară ce conține forma mâinilor. Fiecare punct din matrice ne indică dacă pixelul respectiv aparține mâinii sau nu. Pixelii ce aparțin mâinii trebuie împărțiți în:

- pixeli de contur
- pixeli de interior

Un pixel este de interor dacă fiecare pixel vecin (pixeli de sus, jos, din stânga și din dreapta) aparțin sunt pixeli valizi. Un pixel aparține conturului mâinii dacă unul din pixeli vecini nu este valid (nu este de interior) [14].

4. Diferențierea mâinilor și găsirea conturului

Pentru a crește eficiența calculării centrului palmei avem nevoie ca pixelii ce aparțin conturului mâinii să fie sortați. Pixeli sortați vor fi păstrați într-o listă de puncte. Pentru calcularea conturului mâinilor se va folosi algoritmul *Turtle* descris în [33]. Pseudocodul algoritmului este următorul.

```
1: function TurtleAlgorithm(startPoint, contour, valid)
2:         currPoint := startPoint
3:         dir :=0
4:         lastPoint := Point(-1,-1)
5:         while True do
6:                 if currPoint ≠ lastPoint   then
```

```
7:                          list.add(currentPoint)
8:                          lastPoint := currentPoint
9:                          contour[currPoint.Y][currPoint.X] := False
10:             else
11:                         dir := (dir + 4 - 1)
12:             end if
        #Schimbă direcția
13:             if dir = 0 then
14:                         currPoint.X := currPoint.X + 1
15:             end if
16:             if dir = 1 then
17:                         currPoint.Y := currPoint.Y +1
18:             end if
19:             if dir = 2 then
20:                         currPoint.X := currPoint.X - 1
21:             end if
22:             if dir = 3  then
23:                         currPoint.Y := currPoint.Y - 1
24:             end if
        # Termină bucla dacă s-a ajuns la punctul de început
25:             if currPoint = startPoint then
26:                         break
27:             end if
28:     end while
29:     return list
30: end function
```

De fiecare dată când un punct este adăugat conturului unei mâini, acel punct este marcat ca fiind vizitat. După terminarea găsirii conturului unei mâini, aplicația caută după puncte de contur pentru a calcula conturul sortat al altei mâini. Dacă toate punctele de contur au fost vizitate, nu mai există alte mâini în imagine.

5. Alocarea punctelor interioare

Următorul pas este alocarea punctelor interioare mâinii corespunzătoare.

- calcularea figuri dreptunghice ce va conține (circumscrie) fiecare punct din interiorul mâinii
- aplicarea unui alg. de region filling aplicat pe contur

6. Găsirea centrului palmei

Centrul palmei împreună cu vârfurile degetelor sunt niște puncte foarte importante. Acestea pot fi folosite pentru calcularea altor informații relevante care pot ajuta de exemplu la identificarea de gesturi.

Centrul palmei reprezintă centrul celui mai mare cerc care poate fi desenat în interiorul palmei. Pentru a calcula centrul palmei, se calculează distanțele minime dintre punctele interioare și punctele conturului mâinii. Punctul care corespunde maximului acestor distanțe reprezintă centrul palmei.

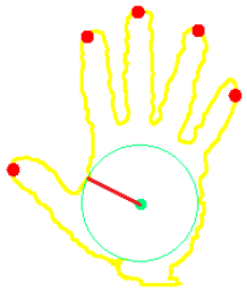


Fig. X.9. Calcularea centrului palmei [34]

Pentru a spori eficiența acestui algoritm vom considera următoarele:

- când se calculează distanța minimă a unui punct interior față de fiecare punct de pe contur, dacă minimul curent este mai mic ca maximul curent, putem fi siguri că acest punct nu este centrul, si nu mai este nevoie să calculăm distanța pentru restul punctelor din contur.
- știind că punctele interioare și cele de contur sunt foarte apropiate unele de altele, putem crește performanța prin executarea algoritmului pe 1 din N puncte consecutive. Eroarea care apare este neglijabilă, iar eficiența este crescută de $1/N^2$.

7. Găsirea vârfurilor degetelor

Pentru găsirea vârfurilor degetelor se va folosi algorimul **k-curvature** descris în [35]. Ideea principală este ca pentru fiecare punct ce aparține conturului P(i), avem punctele P(i-k) și P(i+k). Cu aceste 3 puncte vom genera 2 vectori și vom calcula unghiul minim format. Vectori sunt formați de P(i-k) - P(i) și P(i+k) - P(i). Dacă unghiul este mai mic decât o valoare α, atunci punctul P(i) reprezintă varful unui deget. Valori potrivite sunt k=22 și α = 40.

Pentru a evita găsirea unui vârf fals reprezentat de o vale, vom compara distanța dintre centrul palmei si punctul găsit, cu distanța dintre centrul palmei și cele două puncte ajutătoare. Dacă prima distanță este mai mică, avem un vârf fals.

8. Alocarea punctelor mainii într-un spațiu 3D

- Corespondenta 2D ↔ 3D a senzorului ⇒ aloca punctelor palmei coordonate 3D

X.3.2. Recunoașterea posturilor (gesturilor statice)

- înregistrarea de gesturi statice

- recunoașterea acestora.

Înregistrarea și încărcarea de gesturi statice / posturi

Pentru înregistrarea și încărcarea gesturilor statice fluxul de operații este următorul :

1. Se detecteaza pozitiile punctelor de interes de pe fața, corpul sau mâinile unei persoane, în funcție de tipul de gest ce se dorește a fi înregistrat.
2. se crează un set de referință (antrenare) pentru fiecare gest care se dorește a fi recunoscut. Se înregistrează K instanțe ale valorilor vectorul de trăsături (atribute) pentru fiecare tip de gest.
3. pentru fiecare tip de gest se păstrează informații legate de anumite atribute:
    a. pentru mâini avem următoarele atribute (coordonate 3D) : *Finger_1*, *Finger_2*, *Finger_3*, *Finger_4*, *Finger_5*, *palm_center*.
    b. pentru corp avem următoarele atribute : *HandLeft_X*, *HandLeft_Y*, *HandLeft_Z*, *WristLeft_X*, *WristLeft_Y*, *WristLeft_Z*, *ElbowLeft_X*, *ElbowLeft_Y*, *ElbowLeft_Z*, *ElbowRight_X*, *ElbowRight_Y*, *ElbowRight_Z*, *WristRight_X*, *WristRight_Y*, *WristRight_Z*, *HandRight_X*, *HandRight_Y*, *HandRight_Z*.
    c. pentru față avem următoarele atribute : *BrowLower*, *BrowRaiser*, *JawLower*, *LipCornerDepressor*, *LipRaiser*, *LipStretcher*.
4. se centrează și apoi normalizează vectorul de trăsături în funcție de un anumit punct :
    a. pentru corp se centrează în funcție de punctul de mijloc dintre umeri și se normalizează în funcție de distanța dintre umeri.
    b. pentru mâini se centrează în funcție de mijlocul palmei și se normalizează în funcție de distanța dintre centrul palmei și cot.
    c. fața folosește ca și trăsături/atribute unitățile de animație (AU) oferite de framework-ul de urmărire a feței - se consideră că acestea sunt deja normalizate.
5. se salvează pe disc într-un fișier de tip *arff* gesturile înregistrate.
6. se antreneaza un model de clasificare (Naive Bayes [36]) folosind unealta WEKA [37].
7. se încarcă modelul de clasificare creat mai devreme **(cate un model pt. postura corporala / fata / degete !!!)**.

Recunoașterea de gesturi statice / posturi

1. Se inregistreaza un gest de test (vezi pasii 1-4 de mai sus)
2. se apelează clasificatorul din Weka (**model pt. postura corporala SAU fata SAU degete)** ce primeste un set de atribute pentru testare și returnează un vector cu procentele de potrivire. În vectorul respectiv pentru fiecare gest existent în memorie se returnează un procent de potrivire.
3. se alege procentul maxim din acest vector.
4. dacă acest procent trece de un anumit prag, atunci a fost găsită o postură/un gest static, altfel nu s-a putut clasifica setul de atribute.

X.3.3. Antrenarea clasificatorilor

Pt exempleul considerat, s-au înregistrate **198 instante** temporale pentru fiecare din cele **8 posturi corporale -** acest număr poate crește prin modificarea unei liste constante ce conține etichetele (clasele) posibile:

Left_Hand_Up, Left_Hand_To_The_Left, Left_Hand_To_The_Right, Right_Hand_Up, Right_Hand_To_The_Left, Right_Hand_To_The_Right, Both_Hands_Up, Both_Hands_Down.

Atributele pentru o postura corporala **(18)** sunt numere reale și sunt reprezentate de etichetele :

HandLeft_X, HandLeft_Y, HandLeft_Z, WristLeft_X, WristLeft_Y, WristLeft_Z, ElbowLeft_X, ElbowLeft_Y, ElbowLeft_Z, ElbowRight_X, ElbowRight_Y, ElbowRight_Z, WristRight_X, WristRight_Y, WristRight_Z, HandRight_X, HandRight_Y, HandRight_Z.

Clasificatorii au fost antrenați folosind unealta Weka. În final s-a ales clasificatorul cel mai bun pentru fiecare categorie de gesturi, după ce rezultatele returnate de acestea au fost evaluate statistic.

Tabel X.4. Comparație între rezultatele clasificatorilor testați pentru posturile corporale

Clasificator / **Atribute**	**NaiveBayes**	**LWL clasificator Lazy**
Instanţe	1584	1584
Atribute	18	18
Modul de test	10-fold cross-validation	10-fold cross-validation
Instanţe clasificate corect	1584 100 %	1581 99.9369 %
Instanţe clasificate incorect	0 0 %	1 0.0631 %
Kappa statistic	1	0.9993
Media erorii absolute	0	0.0745
Rădăcina pătratică a mediei erorii	0	0.1415
Eroarea relativă absolută	0 %	34.0365 %
Rădăcina pătrată a erorii relative	0 %	42.7787 %

Tabel X.5 acuratețea detaliată a clasificatorului Bayes Naive pentru posturile corporale

	TP Rate	**FP RATE**	**Precision**	**Recall**	**F-Measure**	**ROC Area**	**Class**
	1	0	1	1	1	1	@Left_Hand_Up
	1	0	1	1	1	1	@Left_Hand_To_The_Left
	1	0	1	1	1	1	@Left_Hand_To_The_Right
	0.995	0	1	0.995	0.997	0.995	@Right_Hand_Up
	1	0	1	1	1	1	@Right_Hand_To_The_Left
	1	0.001	0.995	1	0.997	1	@Right_Hand_To_The_Right
	1	0	1	1	1	1	@Both_Hands_Up
	1	0	1	1	1	1	@Both_Hands_Down
Weighted Avg.	0.999	0	0.999	0.999	0.999	0.999	

Pentru posturile faciale există **198 instante** pentru **5 etichete (gesturi faciale)** :
Neutral, Yawn, Frowny_Face, Happy, Raise_Eyebrows.

Atributele pentru posturile **(6)** faciale suntnumere reale și sunt reprezentate de etichetele :
BrowLover, BrowRaiser, JawLower, LipCornerDepressor, LipRaiser, LipStretcher.

X.6. Comparație între rezultatele clasificatorilor testați pentru posturile faciale

Clasificator / **Atribute**	**NaiveBayes**	**LWL clasificator Lazy**
Instanţe	990	990
Atribute	6	6
Modul de test	10-fold cross-validation	10-fold cross-validation
Instanţe clasificate corect	982 99.1919 %	980 98.9899 %
Instanţe clasificate incorect	8 0.8081 %	10 1.0101 %
Kappa statistic	0.9899	0.9874
Media erorii absolute	0.0079	0.113
Rădăcina pătratică a mediei erorii	0.0554	0.2152
Eroarea relativă absolută	1.437 %	35.32.39 %
Rădăcina pătrată a erorii relative	13.8426 %	53.7905 %

Tabel X.6 acuratetea detaliată a clasificatorului Bayes Naive pentru posturi faciale

	TP Rate	**FP RATE**	**Precision**	**Recall**	**F-Measure**	**ROC Area**	**Class**
	0.985	0.006	0.975	0.985	0.98	0.999	@Neutral
	1	0	1	1	1	1	@Yawn
	1	0	1	1	1	1	@Frowny_Face
	0.99	0.004	0.985	0.99	0.987	1	@Happy
	0.99	0	1	0.985	0.992	1	@Raise_Eyebrows
Weighted Avg.	0.992	0.002	0.992	0.992	0.992	1	

X.3.4. Testarea metodei de recunoastere a gesturilor statice

Scenariu de test:

S-au testat aceleași posturi de câte 10 ori de fiecare persoană, pentru 2 persoane diferite. Fiecare subiect a stat în poziția respectivă timp de 10 secunde pentru a considera recunoașterea posturii ca fiind validă.

Au fost obținute următoarele matrici de confuzie:

Tabel X.7 Matrice de confuzie/acurateț̦e gesturi statice corporale - Realitate scenariu 1

a	**b**	**c**	**d**	**e**	**f**	**g**	**h**		Corect	Incorect	Procent
17	1	2	0	0	0	0	0	**a**	17	5	77
2	16	1	0	0	0	0	1	**b**	16	2	88
2	1	15	0	0	0	0	2	**c**	15	3	83
0	0	0	18	1	0	1	0	**d**	18	1	94
0	0	0	1	17	1	0	1	**e**	17	2	89
0	0	0	0	1	16	1	2	**f**	16	1	94
1	0	0	0	0	0	19	0	**g**	19	2	90
0	0	0	0	0	0	0	20	**h**	20	6	77
22	18	18	19	19	17	21	26	**Total**	138	22	86

unde a = *@Left_Hand_Up* , b = *@Left_Hand_To_The_Left*, c = *@Left_Hand_To_The_Right* , d = *@Right_Hand_Up* , e = *@Right_Hand_To_The_Left* , f= *@Right_Hand_To_The_Right* , g = *@Both_Hands_Up*, h = *@Both_Hands_Down*

Tabel X.8 Matrice de confuzie/acurateț̦e gesturi statice faciale - Realitate scenariu 1

a	**b**	**c**	**d**	**e**		Corect	Incorect	Procent
15	1	2	1	1	**a**	15	4	78
0	17	1	1	1	**b**	17	1	94
1	0	16	2	1	**c**	16	5	76
1	0	2	16	1	**d**	16	4	80
2	0	0	0	18	**e**	18	4	81
19	18	21	20	22	**Total**	82	18	82

unde a = *@Neutral*, b = *@Yawn* , c = *@Frowny_Face* , d = *@Happy* , e = *@Raise_Eyebrows*

Bibliografie

M.C. Giuroiu, **T. Marita**, Gesture Recognition Toolkit Using a Kinect Sensor, *Proceedings of the 11-th IEEE International Conference on Intelligent Computer Communication and Processing* (ICCP2015), Cluj-Napoca, Romania, Sept. 3-5, 2015, p. 317 – 324, ISBN: 978-1-4673-8200-7, DOI: 10.1109/ICCP.2015.7312678 (download)

20. Microsoft Kinect for Windows, http://www.microsoft.com/en-us/kinectforwindows/discover/features.aspx, (cited June 2012)

21. Face Tracking, http://msdn.microsoft.com/en-us/library/jj130970.aspx (cited Sept.. 2013)

22. Kinekt gesture SDK, http://kinectgesture.codeplex.com/, http://kinectdtw.codeplex.com/documentation (cited Jan. 2013)

23. Sait Celebi, Ali S. Aydin, Talha T. Temiz, Tarik Arici, Gesture Recognition Using Skeleton Data with Weighted Dynamic Time Warping, http://mll.sehir.edu.tr/pub/VISAPP2013.pdf, (cited Jan. 2013)

24. P. Senin, Dynamic Time Warping Algorithm Review, Information and Computer Science Department, University of Hawaii at Manoa, Honolulu, USA, December 2008 (cited Jan. 2013)

25. D. Dervinis, Head Orientation Estimation using Characteristic Points of Face, Electronics And Electrical Engineering, T116 - Medicine Technology, No. 8(72), 2006, pp. 61-64.

26. CANDIDE-3 - A parameterized face, http://www.icg.isy.liu.se/candide/ (cited march 2013)

27. O. Patsadu, Human gesture recognition using Kinect camera, Computer Science and Software Engineering (JCSSE), 2012 International Joint Conference on, May 30 2012-June 1 2012, Bangkok, Thailand, pp. 28 – 32.

28. Yi Li, Hand gesture recognition using Kinect, Software Engineering and Service Science (ICSESS), 2012 IEEE 3rd International Conference on, 22-24 June 2012, Louisville, KY, USA, pp. 196 – 199.

29. Face Shift, http://www.faceshift.com/

30. Creation of 3D Human Avatar using Kinect, Asian Transactions on Fundamentals of Electronics, Communication & Multimedia (ATFECM) (ATFECM ISSN: 2221-4305), Vol. 1, Issue 5, Jan. 2012.

31. A Tutorial on Dynamic programming http://www.avatar.se/lectures/molbioinfo2001/dynprog/dynamic.html (cited June 2013)

32. F. Trapero Cerezo 3D Hand and Finger Recognition using Kinect, Universidad de Granada (UGR), Spain (cited May 2013)

33. MARCOS A. G., DE PISÓN ASCACÍBAR F. J. M., ESPINOZA A. V. P., ELÍAS F. A., LIMAS M. C., MERÉ J. O., GONZÁLEZ E. V.: Técnicas y algoritmos básicos de visión artificial 443. Universidad de la Rioja. Servicio de publicaciones, 2006

34. Stefan Stegmueller. Hand and finger tracking with Kinect depth data, 2011. http://candescentnui.codeplex.com

35. TRIGO T. R., PELLEGRINO S. R. M.: An analysis of features for hand-gesture classification. 17th International Conference on Systems, Signals and Image Processing (2010)

36. Naïve Bayes Classifier, http://www.cs.ucr.edu/~eamonn/CE/Bayesian%20Classification%20withInsect_examples.pdf

37. WEKA, Weka 3: Data Mining Software in Java, http://www.cs.waikato.ac.nz/ml/weka/

38. Marius-Cristian GIUROIU, Unealtă de recunoaștere a gesturilor folosind un senzor kinect, Lucrare de licență, Catedra de Calculatoare, UTCN, 2013.