FACULTY OF ENGINEERING AND ARCHITECTURE

Live actieherkenning met de Kinect sensor in Python

Bert De Saffel
Student number: 01614222

Supervisors: Prof. dr. ir. Peter Veelaert, Prof. dr. ir. Wilfried Philips
Counsellors: ing. Sanne Roegiers, ing. Dimitri Van Cauwelaert

Master's dissertation submitted in order to obtain the academic degree of
Master of Science in Information Engineering Technology

Academic year 2018-2019

GHENT UNIVERSITY

FACULTY OF ENGINEERING AND ARCHITECTURE

Live actieherkenning met de Kinect sensor in Python

Bert De Saffel
Student number: 01614222

Supervisors: Prof. dr. ir. Peter Veelaert, Prof. dr. ir. Wilfried Philips
Counsellors: ing. Sanne Roegiers, ing. Dimitri Van Cauwelaert

Master's dissertation submitted in order to obtain the academic degree of
Master of Science in Information Engineering Technology

Academic year 2018-2019

GHENT UNIVERSITY

Voorwoord

Inhoudsopgave

Hoofdstuk 1

Inleiding

Menselijke actieherkenning is het proces dat op een automatische manier (I) detecteert dat een persoon een bepaalde actie uitvoert en (II) herkennen welke actie dit is. Voorbeelden van zulke acties zijn lopen, stappen, zwaaien, springen, bukken, enz. Actieherkenning kent tal van toepassingen zoals

1.1 Probleemstelling

Actieherkenning kan opgesplitst worden in twee onderdelen: actiedetectie en de actieherkenning zelf. Voor actieherkenning is er al uitbundig onderzoek gedaan naar allerlei verschillende manieren om effectief acties te herkennen. Vaak wordt er gebruik gemaakt van histogrammen [1], [2], [3], [4], _ToDo: andere vaak gebruikte dingen zoeken

Actieherkenning en actiedetectie blijft moeilijk te realiseren in een real-time scenario, waarbij snelle beslissingen gemaakt moeten worden. Het doel van deze masterproef is om live classificatie van acties mogelijk te maken.

1.2 De Kinect

De Kinect (figuur 1.1) is origineel ontworpen als manier om de gebruiker zelf als controller te beschouwen. De Kinect bevat een kleurencamera, dieptesensor, _ToDo: eens experimenteren met kinect . Door de goedkope technologie wordt de Kinect vaak gebruikt bij onderzoek naar menselijke actieherkenning. De kleurencamera en dieptesensor kunnen elk bijdragen tot actieherkenning. Enerzijds kunnen de kleurencamera en dieptesensor gecombineerd worden om RGB-D data te bekomen. Anderzijds kan vanuit het dieptebeeld een skelet gegenereerd worden via de methode van Shotton et al. [5], die reeds ingebouwd zit in de Kinect.

Voor deze masterproef wordt de tweede versie van de Kinect gebruikt, die te zien is op figuur 1.1b.

1.3 Structuur

Deze scriptie _ToDo: structuur, pas doen wanneer voldoende inhoud

(a) De originele Kinect sensor, ontwikkeld voor de Xbox 360 in 2010.

(b) De tweede iteratie van de Kinect sensor, specifiek gemaakt voor Xbox One en uitgebracht in 2013.

Figuur 1.1: Twee versies van de Kinect sensor.

Hoofdstuk 2

Methodologie

Ieder persoon heeft een eigen interpretatie van een bepaalde actie. Er kunnen verschillen zijn in snelheid, de positie relatief tot het hele lichaam en _ToDo: nog iets? . Elk van deze variaties in een algoritme steken is dan ook onbegonnen werk. Daarom maakt elk actieherkenningsalgoritme op één of andere manier gebruik van machine learning. Op basis van training data wordt een classifier getraind die kan voorspellen tot welke klasse een nieuwe observatie behoort. Observaties bij een Kinect kunnen RGB beelden of dieptebeelden zijn.

Zulke observaties worden getransformeerd naar *features*. Dit zijn meetbare eigenschappen of karakteristieken van het object dat geobserveerd wordt. Deze eigenschappen moeten bovendien voldoende onderscheidend zijn zodat het mogelijk is om de observatie te klasseren. Een feature tracht de originele observatie te reduceren tot bruikbare informatie om op een eenvoudigere manier classificatie uit te voeren. Een *feature vector* vormt een n-dimensionale wiskundige vector van features. Elke dimensie van deze vector is een individuele feature en vormt de *feature space*. Via bestaande features kunnen er nieuwe features aangemaakt worden via *feature construction*. Het proces om een observatie om te vormen tot een feature vector wordt gerealiseerd met een *feature descriptor*.

Een *classifier* verwacht als input zo een feature vector. Het is de taak van een classifier om te bepalen tot welke klasse een nieuwe observatie behoort. In het geval van actieherkenning is de klasse een bepaalde actie, zoals zwaaien, bukken of springen. Een classifier zal bij het bepalen van een klasse ook een zogenaamde *score* geven. Dit is de waarschijnlijkheid dat de voorspelde klasse correct is. Een eenvoudige *lineaire classifier* berekent de score op basis van een lineaire combinatie door het kruisproduct te nemen tussen de feature vector en een speciale gewichten-vector, specifiek voor die klasse en gebaseerd op de training data. De voorspelde klasse is dan die met de hoogste score. Er bestaan zowel *supervised* als *unsupervised* classificatiemodellen. Beiden maken gebruik van een leerverzameling; een collectie van voorbeelden. Voor een supervised model wordt het gewenste resultaat meegegeven aan elk object in de leerverzameling. Bij een unsupervised model is dit niet zo, maar er is wel een algemeen idee van wat er moet aangeleerd worden.

Een supervised model wordt vaak toegepast op actieherkenning: de leerverzameling bevat videobeelden waarin acties door personen worden uitgevoerd, samen met de gelabelde klasse. Voor actieherkenning speelt het tijdelijke aspect een grote rol. Het is daarom dan ook minder interessant om slechts voor één frame de juiste actie te classificeren. Wel interessant is om voor een

verzameling van frames na te gaan welke actie deze frames voorstellen.

_ToDo: eigen inbreng vanaf nu

- Basisidee: *key frames* = kan zeker voordeel brengen.
  - Welke acties herkennen?
  - Wat is 'te weinig verschil'? Zie bv [6].
  - Wanneer gebruikte frames weggooien? Ik beslis welke frames niet opgenomen worden, dus er is vrij veel bias.
  - Studie hidden markov model → zie 4.3
- Waarom?
  - Live actieherkenning vereist snelle classificatie
  - Verlagen van computationele kost
  - Op voorwaarde dat bepalen van keyframes sneller is dan gewoon elke frame in beschouwing te nemen.
  - Wat is live actieherkenning?
    * Er is geen default pose
    * Er is niet altijd een actor in beeld. Een actor is een persoon waarvan de skeletinformatie beschikbaar is, dus als de kinect correct het skelet kan bepalen van een persoon.
    * Vanaf dat een actor een actie uitvoert, moet deze vroeg genoeg herkend kunnen worden ($<$ 1 seconde, liefst sneller)
    * De classificatie moet ook kunnen omgaan met het tijdsaspect van de uitgevoerde actie.
- Classificatiemodel pas vastleggen nadat verschillende mogelijkheden getest zijn op dataset.
  - Support vector machines
  - ensemble methoden
  - _Opmerking: nog geen prioriteit

_Opmerking: misschien kijken welke features het snelst zijn en sowieso die gebruiken?

2.1 Implementatie

Om enigszins het aantal geschreven code tot een minimum te houden wordt er gekozen om Python te gebruiken in combinatie met scikit-learn.

(voor mezelf, dingen die ik zeker nog op laptop moet uitvoeren:)

- `python -m pip install --upgrade pip`
- `python -m pip install -U matplotlib`

- `pip install -U scikit-learn`
- Zorgen dat tkinter gecheckt is bij installatie (eventueel installatie opnieuw uitvoeren met MODIFIY)

Hoofdstuk 3

Literatuur

- Bron [7]
  - Actieherkenning en actiedetectie systeem voor temporally untrimmed videos door de combinatie van motion en appearance features.
    - Motion feature = Fisher vector representatie met dense trajectories
    - Appearance feature: deep convolutional neural network
- Bron [8]
  - Actieherkenning = het herkennen van een actie binnen een goed gedefinieerde omgeving
  - Actiedetectie = het herkennen en lokaliseren van acties(begin, duratie en einde) in de ruimte en de tijd
  - training set = wordt gebruikt om classifier te trainen
  - validation set = optioneel, bevat andere data dan de training set om de classifier te optimaliseren
  - testing set = testen van de classifier (performance)
  - Drie manieren om dataset op te splitsen in deze drie sets:
    - voorgedefinieerde split: De dataset wordt opgesplitst in twee of drie delen zoals de auteurs van die dataset dat vermelden
    - n-voudige cross-validatie: Verdeeld de dataset in n gelijkvoudige stukken. Hierbij worden er (n-1)/n percentage van de videos gebruikt om te trainen, en dan de overige 1/n om te testen. Dit proces wordt n keer herhaald, zodat elke video éénmaal gebruikt werd voor te testen
    - leave-one-out cross-validatie: aangewezen methode indien testdata personen zijn. 1 persoon wordt als testset beschouwd. De overige $n-1$ personen is de training set.

  - om actieklasse te bepalen = features extraheren en in classifier steken → classifier bepaalt actieklasse
  - Temporally untrimmed video = delen van de video bevatten GEEN ENKELE actie. Variaties van dezelfde actie kan op hetzelfde moment voorkomen
  - THUMOS challenge:
  - 2015 → slechts één team heeft detection challenge geprobeerd
  - Classificatietaak: de lijst van acties geven die in een lange, niet getrimde video voorkomen
  - Detectietaak: ook de lijst van acties geven PLUS de plaats in tijd waar ze voorkomen
- Bron [5] gaat eerder over hoe het skelet bepaalt wordt
  - voorstel van een methode om op een accurate manier de 3D posities van de joints te bepalen, vanuit slechts één dieptebeeld, zonder temporale informatie
  - Het bepalen van lichaamsdelen is invariant van pose, lichaamsbouw, kleren, etc...
  - Kan runnen aan 200 fps
  - Wordt effectief gebruikt in de Kinect software (onderzoeksteam is van Microsoft)
  - Een dieptebeeld wordt gesegmenteerd in verschillende lichaamsdelen, aangegeven door een kleur, op basis van een kansfunctie; Elke pixel van het lichaam wordt apart behandeld en gekleurd. Een verzameling van dezelfde kleuren wordt een joint
  - Aangezien tijdsaspect weg is, is er enkel interesse in de statische poses van een frame. Verschillen van pose in twee opeenvolgende frames is miniscuul zodat die genegeerd worden
- Bron [9] (pre-kinect era)
  - Actieherkenning met behulp van reeksen van dieptebeelden
  - Gaan ervan uit dat efficiënte tracking van skeletbeelden nog niet mogelijk is. (is gepubliceerd zelfde jaar dat Kinect beschikbaar was, 2010)
  - Hun oplossing is dus niet gebaseerd op het tracken van de skeletbeelden
- Bron [10]
  - Probleem: output van de actiecategorie EN de start en eind tijd van de actie.
  - Ze beweren dat actieherkenning reeds goed opgelost is, maar niet actiedetectie. Hun definities zijn:
    * Actieherkenning: De effectieve actieherkenning indien het systeem weet wanneer hij moet herkennen
    * Actiedetectie: een langdurige video, waarbij de start en stop van een actie niet gedefinieerd zijn = untrimmed video (videos waarbij er meerdere acties op hetzelfde moment kunnen voorkomen, alsook een irrelevante achtergrond). sluit heel goed aan op onze masterproef

- Uitdaging in bestaande oplossingen: groot aantal onvolledige actiefragmenten. Voorbeelden:
  - Bron [11]:
    - maakt gebruik van **untrimmed classificatie**: de top $k = 3$ (bepaalt via cross-validation) labels worden voorspelt door globale video-level features. Daarna worden frame-level binaire classifiers gecombineerd met dynamisch programmeren om de activity proposals (die getrimmed zijn) te genereren. Elke proposal krijgt een label, gebaseerd op de globale label.
  - Bron [12]:
    - Spreekt over de onzekerheid van het voorkomen van een actie en de moeilijkheid van het gebruik van de continue informatie
    - Pyramid of Score Distribution Feature (PSDF) om informatie op meerdere resoluties op te vangen
    - PSDF in combinatie met Recurrent Neural networks bieden performantiewinst in untrimmed videos.
    - Onbekende parameters: actielabel, actieuitvoering, actiepositie, actielengte
    - Oplossing? Per frame een verzameling van actielabels toekennen, gebruik makend van huidige frame actie-informatie en inter-frame consistentie = PSDF
- De moeilijkheid is: start, einde en duur van de actie te bepalen.
- Hun oplossing is **Structured Segment Network**:
  - input: video
  - output: actiecategorieën en de tijd wanneer deze voorkomen
  - Drie stappen:
    1. Een "proposal method", om een verzameling van "temporal proposals", elk met een variërende duur en hun eigen start en eind tijd. Elke proposal heeft drie stages: *starting, course* en *ending*.
    2. Voor elke proposal wordt er STPP (structured temporal pyramid pooling) toegepast door (1) de proposal op te splitsen in drie delen; (2) temporal pyramidal representaties te maken voor elk deel; (3) een globale representatie maken voor de hele proposal.
    3. Twee classifiers worden gebruikt: herkennen van de actie en de "volledigheid"van de actie nagaan.

- Bron [13]
  - Temporal action detection = moet enerzijds detecteren of al dan niet een actie voorkomt, en anderzijds hoelang deze actie duurt, wat een uitdaging is bij untrimmed videos.
  - Veel moderne aanpakken gaan als volgt te werk: eerst wordt er klasse-onafhankelijke proposals gegenereerd door

- Bron [14]
  - Sliding window: laatste 30 frames bijhouden in buffer om hoge zekerheid van classificatie te voorzien; met majority voting de actie bepalen die het meeste voorkomt.
  - Classifier: random forests. Beslissingsbomen aanmaken via ID3 algoritme
- Bron [6]
  - Depth-based action recognition.
  - *key frames* worden geproduceerd uit skeletsequenties door gebruik te maken van de joints als **spatial-temporal interest points (STIPs)**. Deze worden gemapt in een dieptesequentie om een actie sequentie te representeren. De contour van de persoon wordt per frame bepaald. Op basis van deze contour en de tijd worden features opgehaald. Als classifier gebruiken ze een *extreme learning machine*
  - Voordeel van key frames: ze bevatten de meest informatieve frames. Twee methodieken om de key frames op te halen:
    1. **Interframe difference**: een nieuwe key-frame wordt gekozen als het verschil tussen twee frames een bepaald threshold overschrijft.
    2. **Clustering**: groeperen van frames die op elkaar lijken op basis van low-level features. Uit die groep wordt dan de keyframe genomen, die het dichtst bij het centrum van dat cluster ligt.
  - Zij gebruiken het 'opgenomen verschil': Een positie van een joint $P_{i,j}$ met i het frame index en j de joint index, kan gelijkgesteld worden als $P_{i,j} = x_{i,j}, y_{i,j}, z_{i,j}$

    Het opgenomen verschil is dan:

    $$D_i = \sum_{j=1}^{n} ||P_{i,j} - P_{i-1,j}||^2$$

    met $|| \cdot ||$ de euclidische afstand en n het aantal joints.
  - key frames worden dan gekozen op basis van maximum of minimum D_i binnen een sliding window. Een probleem: D_i is vrij laag voor de eerste en laatste aantal frames. De key frames worden dus eerder gecentraliseerd en kan de sequentie niet accuraat bepaalt worden. Stapsgewijze oplossing:
    1. Voor een video met N frames: neem de som van D_i van $i = 2$ tot $i = N$:

       $$D_N = \sum_{i=2}^{N} D_i$$

    2. Bepaal een aantal key frames K en bereken het gemiddelde van incrementen:

       $$D_{avg} = D_N / K$$

3. Voor $i = 2$ tot $i = L$ wordt het verschil berekent:

$$W_L = D_L - k * D_{avg}, k \in K$$

zodat er een verzameling W_L is. Het minimum van deze set wordt de key frame.

- Features op basis van contour
- Actieherkenning met neurale netwerken (EXTREME LEARNING)
- **Samenvatting:**
  - Actionherkenningsmethode voor kinect.
  - Features op basis van menselijke contour van een keyframe uit een dieptebeeld. Als constraint is er het temporaal verschil.
  - 'multi-hidden layer extreme learning machine' voor classificatie

- Bron [15]
  - Een bepaalde actie kan onderverdeeld worden in een sequentie van poses.
- Bron [3]
  - HMM laten toe om de temporale evolutie te modelleren.
  - De **feature set** en **emission probability function** moeten goed gedefinieerd zijn.
  - Gegeven een set C van actie classes $\Lambda = \lambda^1 ... \lambda^C$, zoek de klasse λ^* die de kans op $P(\lambda|O)$ maximaliseert met $O = \{o_1 ... o_T\}$ de frame observaties.
  - Een HMM voor elke actie. Classificatie voor een observatiereeks O wordt uitgevoerd door het model te pakken met de hoogste waarschijnlijkheid:

$$\lambda^* = \arg \max_{1 \leq c \leq C} \left[P(O|\lambda^c) \right]$$

  - Twee verschillende features gebruikt:
    - Projection histograms
    - Shape descriptors

Hoofdstuk 4

Machine learning

4.1 Features

- Een **feature** is een individueel, meetbare eigenschap of karakteristiek van een object dat geobserveerd wordt.
- Eigenschappen:
  - Informatief: de informatiewinst van de feature moet hoog zijn
  - Discriminative: op basis van de feature moet het eenvoudig zijn om het onderscheid te maken tussen de verschillende klassen
  - Onafhankelijk: De feature op zich mag van geen andere feature of meetwaarde van dezelfde feature afhangen.
- Een **sparse feature discriptor** heeft een variabel aantal features. Een **dense feature descriptor** heeft een vast aantal features.
- **Feature extraction** ($\equiv$ dimensionality reduction) is het verzamelen van features uit ruwe data zodat deze kunnen gebruikt worden als feature vector bij een classifier.
- Een **feature vector** is een n-dimensionale vector van numerieke features.
- De **feature space** ($\equiv$ vectorruimte) beschrijft de ruimte waarin de features zich bevinden. (bv 3 verschillende features = $\mathcal{R}^3$)
- **Feature construction** is het maken van nieuwe features op basis van reeds bestaande features. De mapping is een functie ϕ, van $\mathcal{R}^n$ naar $\mathcal{R}^{n+1}$, met f de geconstrueerde feature op basis van bestaande features, bv $f = x_1/x_2$.

$$\phi(x_1, x_2, ..., x_n) = (x_1, x_2, ...x_n, f)$$

4.2 Classifier

- Identificeren tot welke klasse een nieuwe observatie behoort, gebaseerd op een training set waarvan de klassen wel gekend zijn.

- **Lineaire classifiers** geven aan elke klasse k een score op basis van de combinatie van de feature vector met een gewichtenvector met het scalair product. De gekozen klasse is dan die met de hoogste score. Eenvoudiger geschreven:

$$score(X_i, k) = \beta_k \cdot X_i$$

  - X_i = de feature vector voor instantie i
  - β_k = de gewichtenvector voor klasse k

- ToDo: later onderzoeken
- **Support Vector machines**
- **Random forests**
- **Boosting**

4.3 Hidden Markov Model

Bron [16]

4.3.1 Markov model

- Markov process = stochastisch process met volgende eigenschappen:
  - Het aantal toestanden $S = \{s_1, s_2, ...s_{|S|}\}$ is eindig.
  - Observatie van een sequentie over de tijd $\vec{z} \in S^T$

    Voorbeeld:

    $S = \{sun, cloud, rain\}$ met $|S| = 3$ en $\vec{z} = \{z_1 = S_{sun}, z_2 = S_{cloud}, z_3 = S_{cloud}, z_4 = S_{rain}, z_5 = S_{cloud}\}$ met $T = 5$.
  - Markov Assumpties:
    1. Een toestand is enkel afhankelijk van de vorige toestand (**Markov Property**).

$$P(z_t|z_{t-1}, z_{t-2}, ..., z_1) = P(z_t|z_{t-1})$$

    2. De waarschijnlijk is constant in de tijd

$$P(z_t|z_{t-1}) = P(z_2|z_1); t \in 2...T$$

  - Beginstaat $z_0 \equiv s_0$.
  - Transitiematrix $A \in \mathbb{R}^{(|S|+1)\times(|S|+1)}$, met $A_{ij} = P(i \to j)$, :

$$A = \begin{array}{r} \\ s_0 \\ s_{sun} \\ s_{cloud} \\ s_{rain} \end{array} \begin{array}{cccc} s_0 & s_{sun} & s_{cloud} & s_{rain} \\ 0 & .33 & .33 & .33 \\ 0 & .8 & .1 & .1 \\ 0 & .2 & .6 & .2 \\ 0 & .1 & .2 & .7 \end{array}$$

- Twee vragen die een markov model kunnen oplossen:
  1. Wat is de kans op een bepaalde toestandenvector $\vec{z}$?

$$P(\vec{z}) = \prod_{t=1}^{T} A_{z_{t-1}z_t}$$

     Stel $\vec{z} = \{z_1 = S_{sun}, z_2 = S_{cloud}, z_3 = S_{rain}, z_4 = S_{rain}, z_5 = S_{cloud}\}$ dan is $P(\vec{z}) = 0.33 \times 0.1 \times 0.2 \times 0.7 \times 0.2$

  2. Gegeven $\vec{z}$, hoe worden best de parameters van A benaderd zodat de kans op $\vec{z}$ maximaal is.

$$A_{ij} = \frac{\sum_{t=1}^{T}\{z_{t-1} = s_i \wedge z_t = s_j\}}{\sum_{t=1}^{T}\{z_{t-1} = s_i\}}$$

     De maximale kans om van staat i naar j te gaan is het aantal transities van i naar j gedeeld door het totaal aantal keer dat we in i zitten. Met andere woorden: Hoeveel % zaten we in i als we van j komen.

4.3.2 Hidden Markov Model

- HMM = Veronderstelt een markov process met verborgen toestanden
- Bij een HMM: de staat is niet zichtbaar, maar de output is wel zichtbaar.
- Formeel:
  - Sequentie van geobserveerde outputs $x = \{x_1, x_2, ..., x_T\}$ uit een alfabet $V = \{v_1, v_2, ..., v_{|V|}\}$
  - Er is ook een sequentie van staten $z = \{z_1, z_2, ...z_T\}$ uit een alfabet $S = \{s_1, s_2, ...s_{|S|}\}$, maar deze zijn niet zichtbaar.
  - Transitiematrix A_{ij} wel bekend.
  - Kans dat een bepaalde output gegenereerd wordt in functie van de verborgen toestanden:

$$P(x_t = v_k | z_t = s_j) = P(x_t = v_k | x_1, ..., x_T, z_1, ..., z_T) = B_{jk}$$

    Matrix B geeft de waarschijnlijkheid dat een verborgen toestand s_j de output v_k teruggeeft.

- Vaak voorkomende problemen die opgelost kunnen worden met HMM:
  - Gegeven de parameters en geobserveerde data, benader de optimale sequentie van verborgen toestanden.
  - Gegeven de parameters en geobserveerde data, bereken de kans op die data. → Wordt het 'decoding' probleem (**Viterbi Algoritme**) genoemd en wordt gebruikt bij continue actieherkenning.
  - Gegeven de geobserveerde data, benader de parameters van A en B.

- Het **decoding probleem**: `http://jedlik.phy.bme.hu/~gerjanos/HMM/node8.html`
  - Zoek de meest waarschijnlijke reeks van toestanden $\vec{z} \in S^T$ voor een verzameling van observaties $\vec{x} \in V^T$.
  - Hoe 'meest waarschijnlijke toestandensequentie' definiëren. Een mogelijke manier is om de meest waarschijnlijke staat s_t voor x_t te berekenen, en alle q_t dier daar aan voldoen te concateneren. Andere manier is **Viterbi algoritme** die de hele toestandensequentie met de grootste waarschijnlijkheid teruggeeft.
  - Hulpvariabele:

    $$\delta_t(i) = \max_{q_1,q_2,...q_{t-1}} p\{q_1, q_2, ...q_{t-1}, q_t = i, o_1, o_2, ...o_{t-1}|\lambda\}$$

    die de hoogste kans beschrijft dat een partiele observatie en toestandensequentie tot $t = t$ kan hebben, wanneer de huidige staat i is.

Hoofdstuk 5

Conclusie

Woordenlijst

HMM Hidden Markov Model. 13

Lijst van figuren

Lijst van tabellen

Bibliografie

[1] L. Xia, C. Chen, and J. Aggarwal, "View invariant human action recognition using histograms of 3d joints," in Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on. IEEE, 2012.

[2] C. Chen, "Action recognition using 3d histograms of texture and a multi-class boosting classifier," IEEE Transactions on Image Processing, 06 2017.

[3] R. Vezzani, D. Baltieri, and R. Cucchiara, "Hmm based action recognition with projection histogram features," 2010.

[4] M. Mendoza and N. Perez de la Blanca, "Hmm-based action recognition using contour histograms," 06 2007, pp. 394–401.

[5] J. Shotton, A. Fitzgibbon, A. Blake, A. Kipman, M. Finocchio, B. Moore, and T. Sharp, "Real-time human pose recognition in parts from a single depth image." IEEE, June 2011, best Paper Award. [Online]. Available: https://www.microsoft.com/en-us/research/publication/real-time-human-pose-recognition-in-parts-from-a-single-depth-image/

[6] S. Liu and H. Wang, "Action recognition using key-frame features of depth sequence and elm," (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 8, No. 10, 2017, 2017.

[7] L. Wang, Y. Qiao, and X. Tang, "Action recognition and detection by combining motion and appearance features," 2014.

[8] S. Min Kang and R. Wildes, "Review of action recognition and detection methods," 10 2016.

[9] W. Li, Z. Zhang, and Z. Liu, "Action recognition based on a bag of 3d points." IEEE, June 2010, pp. 9–14. [Online]. Available: https://www.microsoft.com/en-us/research/publication/action-recognition-based-bag-3d-points/

[10] Y. Zhao, Y. Xiong, L. Wang, Z. Wu, X. Tang, and D. Lin, "Temporal action detection with structured segment networks," 2017.

[11] G. Singh and F. Cuzzolin, "Untrimmed video classification for activity detection: submission to activitynet challenge," arXiv preprint arXiv:1607.01979, 2016.

[12] J. Yuan, B. Ni, X. Yang, and A. Kassim, "Temporal action localization with pyramid of score distribution features," 06 2016, pp. 3093–3102.

[13] J. Huang, N. Li, T. Zhang, G. Li, T. Huang, and W. Gao, "Sap: Self-adaptive proposal model for temporal action detection based on reinforcement learning," 2018. [Online]. Available: https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16109

[14] F. Deboeverie, S. Roegiers, G. Allebosch, P. Veelaert, and W. Philips, "Human gesture classification by brute-force machine learning for exergaming in physiotherapy," in 2016 IEEE Conference on Computational Intelligence and Games (CIG), Sept 2016, pp. 1–7. [Online]. Available: http://ieeexplore.ieee.org/document/7860414/

[15] S. Carlsson and J. Sullivan, "Action recognition by shape matching to key frames," Workshop on Models versus Exemplars in Computer Vision, Jan. 2001.

[16] D. Ramage, "Hidden markov models fundamentals," 2007.