

Inhoudsopgave

1	Inleiding	1
1.1	Probleemstelling	1
1.2	De Kinect	2
1.3	Structuur	2
2	Methodologie	3
2.1	Implementatie	4
3	Features	6
3.1	Features uit kleurenbeelden	6
3.2	Features uit dieptebeelden	7
3.2.1	Histograms of 3D Joints (HOJ3D)	8
3.2.2	Covariance Descriptors on 3D Joint Locations (COV3DJ)	8
3.2.3	Local Occupancy Pattern (LOP)	9
3.2.4	EigenJoints	10
3.2.5	Sequence of the Most Informative Joints (SMIJ)	11
3.2.6	Lie group	13
3.2.7	Grassmann manifold	14
3.2.8	Rollende rotaties	14
3.2.9	3DHoTs	16
4	Classificatie	17
4.1	Leren	17
4.2	Directe classificatie	18
4.2.1	k -nearest neighbours	18
4.2.2	Lineaire classifiers	18
4.3	Temporale modellen	18
4.3.1	Generatieve modellen	19
4.3.2	Discriminerende modellen	19
4.4	Key frames	20
5	Evaluatie	21
5.1	Preprocessing	21
5.2	Classifiers	22
5.3	Segmentatie	23
5.4	Herkenning	23

6	notities	24
6.1	papers	24
6.2	Machine learning	26
6.2.1	Features	26
6.2.2	Classifier	27
6.2.3	Hidden Markov Model	27
	Bibliografie	31

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 fysiotherapie [1], lichaamsanalyse [2], mens-robot interactie [3] en teleoperatie [4]. In deze masterproef wordt actieherkenning gecombineerd met teleoperatie, meer specifiek voor rijdende voorwerpen, zoals een robot of een slimme fiets. Een operator kan dan door enkel gebaren uit te voeren het voorwerp besturen. Belangrijk hierbij is het real-time aspect; een te trage herkenning van de actie zorgt voor een late reactie waardoor er potentieel ongevallen kunnen plaatsvinden.

1.1 Probleemstelling

Men is reeds in staat om voor zowel kleurenbeelden als dieptebeelden actieherkenning toe te passen op een betrouwbare manier. Echter blijft het moeilijk om dit te realiseren in een real-time scenario, waarbij snelle beslissingen gemaakt moeten worden. Een groot probleem is de onzekerheid van de actie.

In de literatuur worden er vaak methoden ([5], ...) beschreven die acties herkennen op video's waarop slechts één actie uitgevoerd wordt. De actielabel wordt ook maar nadien toegekend, als de video reeds gedaan is. Een betere aanpak zou de actielabel toekennen op het moment dat de actie bezig is, op een video waarbij meerdere acties kunnen uitgevoerd worden op hetzelfde moment. In de literatuur worden dit *untrimmed video's* genoemd. In zulke video's zijn de actielabels, actiepositie en actieduur onbekende parameters.

De THUMOS Challenge ¹ werd georganiseerd in 2015 en was een initiatief om op een competitieve manier het probleem van actieherkenning in untrimmed video's te onderzoeken. Ze bieden een dataset aan met untrimmed video's die scenario's uit het echte leven bevatten.

Het doel van deze masterproef is om eerst te onderzoeken welke methoden er al bestaan op vlak van actieherkenning in untrimmed video's en hoe deze geoptimaliseerd kunnen worden zodat ze efficiënt werken op de skeletdata van een Kinect. In hoofdstuk 2 wordt de uitwerking beschreven.

¹<http://www.thumos.info>



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

1.2 De Kinect

De Kinect (figuur 1.1) werd initieel ontworpen voor gebruik met de Xbox 360, als een manier om de gebruiker zelf als spelcontroller te beschouwen. Dit is een grote vordering in vergelijking met bijvoorbeeld de Wii Remote, waarbij de gebruiker een controller moet hebben om acties uit te voeren. Door de combinatie van de diepte- en kleurenbeelden die de Kinect kan leveren kunnen er skeletbeelden gegenereerd worden. Voor elke frame kan de Kinect tot maximaal 25 skeletjoints genereren, voor maximaal zes personen, die drie-dimensionale coördinaten voorstellen van voornamelijk gewrichten van het lichaam. Voorbeelden van locaties waarop joints komen zijn onder andere het hoofd, de schouders, de ellebogen, de polsen, enz. Dit heeft een groot voordeel: het detecteren van de low-level features uit de kleuren- en dieptebeelden om de skeletjoints te bepalen wordt overgelaten aan de Kinect zelf. Sinds de opkomst van de Kinect is het onderzoek naar actieherkenning met behulp van skeletjoints weer op gang geschoten [1], [5], [?], [6], [3].

Ondanks het succes van de Kinect heeft Microsoft toch besloten om het product niet meer als entertainmentproduct te verkopen, maar wel als een tool die onderzoekers en ontwikkelaars kunnen helpen. Daarom laten ze de oude Kinect achterwegen en maken ze een nieuwe versie die de naam *Azure Kinect* draagt. Deze Kinect is rechtstreeks verbonden met het Azure platform van Microsoft. In deze masterproef wordt de tweede versie van de Kinect gebruikt (figuur 1.1b).

1.3 Structuur

In hoofdstuk 2 wordt de algemene methodologie beschreven die gebruikt wordt om deze masterproef te realiseren, waarom deze gebruikt wordt en wat het beoogde eindresultaat is. Verder wordt er in hoofdstuk 3 en hoofdstuk 4 een overzicht gegeven van bestaande methoden en welke features en classifiers zij gebruiken. In hoofdstuk **ToDo: hoofdstuk uitwerking, resultaten, conclusie**

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 zelfs de lichaamsbouw van die persoon. Elk van deze variaties in een algoritme steken is dan ook onbegonnen werk. Daarom maakt elk modern 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. In deze masterproef wordt de skeletinformatie van de Kinect beschouwd als observatie, maar andere observaties zijn ook mogelijk zoals de ruwe kleuren- of dieptebeelden.

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 heeft als doel 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 alle features samen vormt de *feature space*. Als voorbeeld van een feature vector zouden de drie-dimensionale coördinaten van elke skeletjoint gekozen kunnen worden. De feature vector voor een frame zou dan, in het geval van 25 joints, 75 dimensies bevatten. 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 *feature detectors* en *feature descriptors*. In functie van computervisie bepaalt een feature detector de locaties waar eventueel interessante pixels kunnen zijn. Een hoekdetector zal de coördinaten van pixels geven waar er hoeken zijn. Een feature descriptor omschrijft de lokale regio rond elk van deze gevonden pixels, zoals bijvoorbeeld de ruwe pixelwaarden in een bepaald bereik.

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 tussen de feature vector en een speciale gewichtenvector, specifiek voor die klasse en die gebaseerd is 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 waaruit het systeem moet leren. 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.

In deze masterproef wordt er gebruik gemaakt van de skeletdata die door de Kinect genereert wordt. Dit is een verzameling van joints waarbij elke joint gekenmerkt wordt door een unieke index, zijn drie-dimensionale coördinaten en zijn relatieve quaternionen. Deze skeletdata kan op allerlei manieren gemanipuleerd worden om nuttige feature vectors te construeren. Er wordt een onderscheid gemaakt tussen *joint-based* en *body-based* features. Joint-based features zien de joints als een verzameling van punten waarbij de joints onafhankelijk van elkaar beschouwd worden ([?], [7]), via een vast assenstelsel gelokaliseerd worden ([5]) of via de relatieve posities tussen elk paar joints gekenmerkt worden ([8], [9]). Body-based features zien het skelet als een geheel van vaste lichaamsdelen die onderling verbonden zijn met elkaar. Deze methoden focussen zich op verbonden paren van lichaamsdelen en modelleren de temporale evolutie met behulp van de hoeken tussen deze lichaamsdelen ([10], [11], [1]).

_ToDo: In deze masterproef wordt X en Y gebruikt omdat Z.

Tabel 2.1 geeft een overzicht van de herkende acties en wat hun semantische waarde is. Er wordt een eigen dataset gecreëerd die deze x acties bevat die door y verschillende personen worden uitgevoerd.

2.1 Implementatie

Er wordt gekozen om de software te implementeren met Python. Python biedt een rijk aanbod van *machine learning tools* die het ontwikkelen van dergelijke applicaties sterk vereenvoudigen. Om de Kinect aan te spreken wordt er gemaakt van *PyKinectV2*. Dit is een Application Programming Interface (API) geïmplementeerd die bindings beschikbaarstelt om de Kinect vanuit Python aan te spreken.

Actie	Betekenis
1. Gestrekt rechterarm met handpalm naar de camera gericht	Stop huidige actie
2. Achteruit zwaaien met de rechterarm	Verplaats in de richting van de gebruiker (vooruit)
3. Vooruit zwaaien met de rechterarm	Verplaats in de tegenovergestelde richting van de gebruiker (achteruit)
4. Naar rechts zwaaien met de rechterarm	Verplaats naar links, vanuit het perspectief van robot (dus naar rechts voor de operator)
5. Naar links zwaaien met de rechterarm	Verplaats naar rechts, vanuit het perspectief van robot (dus naar links voor de operator)
6. Gestrekt linkerarm met handpalm naar de camera gericht	Stop huidige actie
7. Achteruit zwaaien met de linkerarm	Verplaats in de richting van de gebruiker (vooruit)
8. Vooruit zwaaien met de linkerarm	Verplaats in de tegenovergestelde richting van de gebruiker (achteruit)
9. Naar rechts zwaaien met de linkerarm	Verplaats naar links, vanuit het perspectief van robot (dus naar rechts voor de operator)
10. Naar links zwaaien met de linkerarm	Verplaats naar rechts, vanuit het perspectief van robot (dus naar links voor de operator)
11. Cirkel tekenen	Rotatie rond de as uitvoeren

Tabel 2.1: De herkende acties samen met hun semantische waarde.

Hoofdstuk 3

Features

Een belangrijk aspect bij menselijke actieherkenning is het extraheren van bepaalde informatie. Om menselijke actieherkenning toe te passen is er een omzetting nodig van de ruwe beelden tot een vorm zodat gelijkaardige uitvoeringen van éénzelfde actie door dezelfde structuur kan beschreven worden. Aangezien dat het om menselijke actieherkenning gaat, is het enkel nuttig om informatie over de persoon die de actie uitvoert bij te houden. Een camera die aan het filmen is genereert frames. Men zou voor elke frame een nuttige beschrijving kunnen zoeken zodat enkel informatie overblijft van de persoon zelf. Zo een beschrijving wordt een *feature vector* genoemd. Deze *feature vector* kan verschillende vormen aannemen, maar elke vorm heeft de volgende gelijkens: de *feature vector* is een rij van getallen waarbij elk getal een ander attribuut beschrijft. Als voorbeeld nemen we skeletdata. De Kinect sensor kan voor elke frame op basis van de ruwe kleuren- en dieptebeelden een skeletbeeld genereren met maximum 25 punten. Elk punt kan geïdentificeerd worden door zijn x, y, en z coördinaten. Per frame kan er dan een *feature vector* f opgebouwd worden met 75 waarden, die de x, y en z beschrijven voor elk van de punten.

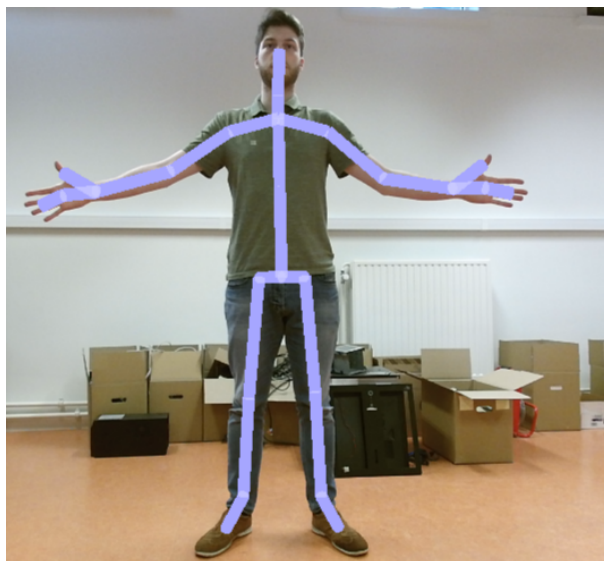
$$f = (x_1 \ y_1 \ z_1 \ x_2 \ y_2 \ z_2 \ \dots \ x_{25} \ y_{25} \ z_{25})$$

Dit is slechts één van de vele mogelijke *feature vectors* die kunnen opgesteld worden. Dit hoofdstuk verkent de mogelijkheden van *features* voor zowel kleuren- als dieptebeelden.

3.1 Features uit kleurenbeelden

De extractie van features bij kleurenbeelden kan in twee categorieën onderverdeeld worden: *globale feature extraction* en *lokale feature extraction* [12]. Bij een globale aanpak wordt een persoon eerst gelokaliseerd met behulp van *background subtraction* of *tracking* gevolgd door het encoderen van de interesseregio's. Background subtraction kan enkel met statische camera's uitgevoerd worden. Er wordt een referentiefraam bijgehouden die enkel statische informatie bevat. Het verschil tussen een nieuwe frame en de referentiefraam kan nieuwe informatie opleveren. Deze representatie kan veel informatie bevatten, maar door de nood aan background subtraction of tracking zijn zulke methoden gevoelig aan ruis.

Een lokale aanpak tracht deze problemen te vermijden door eerst lokale interessepunten, in de literatuur Spatio-Temporal Interest Points (STIP) genoemd, te bepalen. Deze interessepunten



Figuur 3.1: De 25 skeletjoints. De Kinect stelt onder andere de drie-dimensionale coördinaten en de relatieve quaternionen ter beschikking.

kunnen zowel het temporale als het ruimtelijke aspect modelleren. Rond deze interessepunten worden patches berekent, afhankelijk van gekozen parameters. De verzameling van zulke patches is de representatie.

feature detectors: ~~_ToDo:~~ harris3D [?] ~~_ToDo:~~ cuboid [13] ~~_ToDo:~~ Hessian [14] ~~_ToDo:~~ dense trajectory [15]

feature descriptors: ~~_ToDo:~~ cuboid descriptor: [13] ~~_ToDo:~~ hog/hof [?]]

3.2 Features uit dieptebeelden

De algoritmen die bruikbaar zijn op kleurenbeelden kunnen niet toegepast worden op dieptebeelden.

Literatuur over feature extraction op dieptebeelden levert vaak algoritmen [5], [8], [9] op die gebruik maken van skeletbeelden gegenereerd met methode [16] waarop de skelettracker van de Kinect op gebaseerd is. De skeletbeelden van de Kinect geven de drie-dimensionale coördinaten van 25 punten, *joints* genoemd, die belangrijke kenmerken van het menselijk lichaam voorstellen. Al deze joints worden weergegeven op figuur 3.1.

Het gebruik van de Kinect is geen vereiste om actieherkenning met dieptebeelden uit te voeren. [17] stelt elke frame voor als een verzameling van 3D punten, geëxtraheerd uit de silhouetten dat de dieptebeelden geven en maken gebruik van een Hidden Markov Model (HMM) om het temporale aspect te modelleren. [18]

In volgende onderdelen worden een aantal belangrijke feature descriptors beschreven.

3.2.1 Histograms of 3D Joints (HOJ3D)

Het werk van [5] toont aan hoe een histogram van drie-dimensionale punten aan real-time actieherkenning kan doen. Ze transformeren het skeletbeeld gegenereerd door de Kinect om in bolcoördinaten. Als oorsprong wordt de heup joint genomen. De horizontale referentievectoren α wordt parallel met de grond genomen door de oorsprong. De verticale referentievectoren θ staat loodrecht op α en gaat ook door de oorsprong. De drie-dimensionale ruimte wordt opgesplitst in 84 deelruimten. Elke joint zal zich dan in één van die 84 deelruimte bevinden. Een histogram wordt opgemaakt door elke joint te wegen in 8 naburige deelruimten via een Gaussische functie:

$$p(X, \mu, \Sigma) = \frac{1}{(2\pi)^{n/2} |\Sigma|^{1/2}} e^{-\frac{1}{2}(X-\mu)^T \Sigma^{-1} (X-\mu)}$$

Hierbij is μ de mediaanvector en Σ de identiteitsmatrix. Voor zowel Θ als α wordt de kansfunctie apart uitgerekend. Stel Ω de cumulatieve distributiefunctie van de normaalverdeling, dan wordt de kans dat een joint met locatie (μ_α, μ_θ)

Dit levert de Histogram of 3D Joints (HOJ3D) descriptor op. **ToDo: nog verder**

3.2.2 Covariance Descriptors on 3D Joint Locations (COV3DJ)

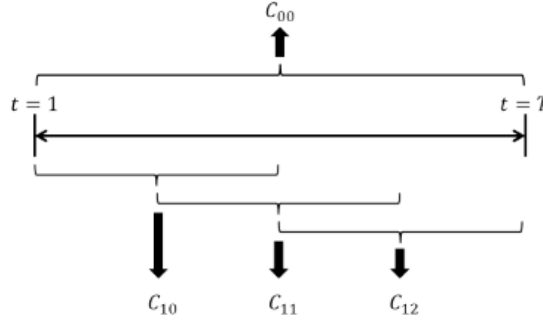
De methode van [19] maakt gebruik van covariantiematrices. Een covariantiematrix voor een verzameling van N willekeurige variabelen is een $N \times N$ matrix waarvan de elementen de covariantie bevatten tussen elk paar variabelen. Als $\mathbf{X} = (X_1, X_2, \dots, X_N)$, een vector met N random variabelen, dan wordt een element $K_{X_i X_j}$ van de covariantiematrix K gedefinieerd als:

$$K_{X_i X_j} = E[(X_i - E[X_i])(X_j - E[X_j])]$$

Zo een matrix bevat informatie over de gezamenlijke kans $P(X_j) \cdot P(X_i|X_j)$. Het eerste gebruik van zo een matrix is in het werk van [20] met als doel een regio van een afbeelding te beschrijven.

Elke joint i kan voorgesteld worden door zijn drie-dimensionale coördinaten voor een frame t : $p_i^{(t)} = (x_i^{(t)}, y_i^{(t)}, z_i^{(t)})$. De concatenatie van alle joints voor frame t is een vector \mathbf{S} , met $3K$ elementen: $\mathbf{S} = (x_1, \dots, x_K, y_1, \dots, y_K, z_1, \dots, z_K)$, hierbij is K het aantal joints dat beschikbaar is op één frame. Voor \mathbf{S} kan nu de covariantiematrix berekend worden, maar de kansverdeling is niet gekend, zodat de steekproefcovariantie wordt gekozen. De resulterende matrix is symmetrisch ten opzichte van de hoofddiagonaal, zodat enkel de bovendrehoek in het geheugen beschikbaar moet zijn. Voor $K = 25$ is het aantal resulterende elementen in de bovendrehoek gelijk aan 2850. In vergelijking met [19] waarbij ze $K = 20$ nemen, is het resultaat 1830. Vijf extra joints zorgt al voor een toename van 1020 elementen in de matrix.

Deze descriptor, Covariance of 3D Joints (Cov3DJ) genoemd, bevat de locaties van de verschillende joints, afhankelijk van elke andere joint tijdens een actie. De temporale informatie wordt beschreven als een hiërarchie van zulke descriptors. Het niveau l duidt de diepte in de hiërarchie aan waarbij $l = 0$ de top van de hiërarchie is. Elk niveau bevat $2^l - 1$ descriptors die elk $\frac{T}{2^l}$ frames bevatten van de sequentie. Voor $l = 1$ zullen er drie descriptors gegenereerd worden die elk $T/2$ frames zullen modelleren waarbij T het totaal aantal frames is. Dit wordt grafisch weergegeven op figuur 3.2.



Figuur 3.2: De temporale constructie van de descriptor. C_{li} is de i -de descriptor op niveau l .

Bron: Figuur 2 in [19].

Elke descriptor op elk niveau bevat nog steeds hetzelfde aantal elementen (1830 voor $K = 20$ of 2850 voor $K = 25$), zodat de lengte van de totale descriptor nu gelijk is aan het aantal descriptors maal het aantal elementen. Voor $K = 20$ en $K = 25$ bedraagt de lengte van de totale descriptor voor figuur 3.2 respectievelijk 7320 en 11 400 elementen.

3.2.3 Local Occupancy Pattern (LOP)

Deze methode [8] berekent allereerst de drie-dimensionale afstand van elke joint i tot elke andere joint j : $\mathbf{p}_{ij} = \mathbf{p}_i - \mathbf{p}_j$. De feature vector voor elke joint i wordt dan:

$$\mathbf{p}_i = \{\mathbf{p}_{ij} | i \neq j\}$$

Aanvullend aan deze feature wordt een Local Occupancy Pattern (LOP) gedefinieerd. Deze feature geeft de lokale bezettingsgraad aan; een maat om aan te geven in hoeverre een joint door een ander object gehinderd wordt. Op frame t wordt er een puntenwolk gegenereerd op basis van het dieptebeeld. Voor elke joint j wordt zijn lokale regio gepartitioneerd in een $N_x \times N_y \times N_z$ raster. Elke cel van dit raster bevat $S_x \times S_y \times S_z$ pixels. Voor elke cel c_{xyz} wordt de som van de punten genomen die zich in die cel bevinden. De sigmoïdefunctie $\delta(x) = \frac{1}{1+e^{-\beta x}}$ wordt toegepast op deze som om zo de feature o_{xyz} te bekomen:

$$o_{xyz} = \delta\left(\sum_{q \in c_{xyz}} I_q\right)$$

- **Inleiding:** Ze ontwikkelen een manier om intra-klasse variantie op te lossen met dieptebeelden + skeletbeelden via [16]. Ze geven ook nieuwe features die voor dieptedata werken. Ze vermelden dat features sensorafhankelijk zijn. Bij kleurenbeelden wordt bijvoorbeeld STIP [21] gebruikt om interessepunten te vinden, en gebruiken Histograms of Optical Flow (HOF) [?] of Histograms of Oriented Gradients (HOG) [22]. Voor dieptebeelden werken zo features niet.

Om het temporale aspect te modelleren worden HMM [7] gebruikt, of Conditional Random Field (CRF) [23]. Andere manier is Dynamic Temporal Warping (DTW) [24], maar dat vereist een goede metriek om twee frames te vergelijken. Voor cyclische acties lijkt DTW ook niet goed. Daarom gebruiken zij Fourier Temporal Pyramid (FTP).

- **Methode:**

1. Op basis van dieptedata en 3D joint posities bouwen zij een nieuwe feature LOP op. Elke 3D joint wordt geassocieerd met zo een LOP.
2. Zij gebruiken niet individuele joints, maar gebruiken de relatieve positie. Een joint i heeft 3 (genormaliseerde) coördinaten $p_i(t) = (x_i(t), y_i(t), z_i(t))$ op een frame t .

Ze berekenen $\mathbf{p}_{ij} = \mathbf{p}_i - \mathbf{p}_j$, de relatieve positie van een joint i met elke andere joint j . Op die manier wordt er voor elke joint i de feature bepaalt:

$$\mathbf{p}_i = \{\mathbf{p}_{ij} | i \neq j\}$$

Tussen elk paar joints berekenen is overbodig, ze introduceren daarom actionlet mining om enkel de joints te selecteren die nuttig zijn voor de classificatie.

3. De bijkomende feature, LOP dient om de lokale diepte voor de joints te modelleren. Ze construeren een drie-dimensionale puntenwolk rond een bepaalde joint.

Voor frame t kan voor elke joint j een puntenwolk opgebouwd worden als volgt:

- (a) De lokale regio van j wordt gepartitioneerd in een $N_x \times N_y \times N_z$ grid. Elke bin in de grid heeft (S_x, S_y, S_z) pixels.
- (b) Het aantal punten voor t dat in elke bin b_{xyz} zit wordt geteld. Er wordt een sigmoid normalisatiefunctie toegepast om de feature o_{xyz} te bekomen voor b_{xyz} . De *local occupancy information* voor die bin wordt dan:

$$o_{xyz} = \delta \left(\sum_{q \in bin_{xyz}} I_q \right)$$

met $I_q = 1$ als er zich een punt bevindt op locatie q en 0 anders en $\sigma(x) = \frac{1}{1+e^{-\beta x}}$

- (c) Voor elke frame t zijn er twee features: De 3D posities $\mathbf{p}_i[t]$ en de LOP features $\mathbf{o}_i[t]$. Via FTP wordt de temporale dynamiek gemodelled.

- **Gebruikte datasets:**

- MSRDailyActivity3D Dataset
- CMU MoCap Dataset

- **Toekomst:**

- Gebruik van actionlets om meer complexere acties te begrijpen.

3.2.4 EigenJoints

Het werk van [9] introduceert een nieuwe soort joint, de *EigenJoint*. Ze maken gebruik van de drie-dimensionale positieverschillen tussen elk paar van joints, zoals bij LOP, en extraheren drie features voor elke frame c : de *posture feature* f_{cc} , de *motion feature* f_{cp} en de *offset feature* f_{ci} . Deze drie features worden geconcateneerd om de feature $f_c = (f_{cc}, f_{cp}, f_{ci})$ te bekomen. Deze

feature wordt genormaliseerd via Principal Component Analysis (PCA) **ToDo: uitleggen PCA**. Elke frame i bevat N joints: $X_i = \{x_1^i, x_2^i, \dots, x_N^i\}$

De *posture feature* beschrijft de statische postuur dat een persoon aanneemt. Voor elke joint wordt de drie-dimensionale afstand berekent tussen elk paar van joints voor de huidige frame c :

$$f_{cc} = \{x_i^c - x_j^c | i, j = 1, 2, \dots, N; i \neq j\}$$

De dynamiek wordt gemodelleerd met de *motion feature* die de drie-dimensionale afstand berekent tussen elke joint van de huidige frame c met elke andere joint van de vorige frame p :

$$f_{cp} = \{x_i^c - x_j^p | x_i^c \in X_c; x_j^p \in X_p\}$$

Tot slot wordt nog de *offset feature* gedefinieerd, die de algemeen dynamiek modelleert door de drie-dimensionale afstand te berekenen tussen elke joint van de huidige frame c met elke andere joint van de initiële frame i :

$$f_{ci} = \{x_i^c - x_j^i | x_i^c \in X_c; x_j^i \in X_i\}$$

De verzameling van deze drie feature vectoren wordt f_c genoemd. Er wordt een lineaire normalisatie uitgevoerd zodat elk attribuut in f_c zich in het bereik $[-1, +1]$ bevindt zodat f_{norm} bekomen wordt. Het aantal dimensies wordt vrij hoog; voor $N = 25$ zorgen f_{cc} , f_{cp} en f_{ci} respectievelijk voor 300, 625 en 625 jointbewerkingen op. Elke bewerking genereert drie attributen, $\delta x, \delta y, \delta z$, zodat de totale dimensie van f_c gelijkgesteld moet worden aan $(300 + 625 + 625) \times 3 = 4650$. In vergelijking met [9] waarbij ze $N = 20$ gebruiken, is de totale dimensie 2970.

3.2.5 Sequence of the Most Informative Joints (SMIJ)

Het werk van [10] vertrekt van de observatie dat verschillende personen een actie op diverse manieren kunnen uitvoeren, maar dat altijd dezelfde joints gebruikt worden om die actie uit te voeren. Ze berekenen de *relative informativeness* van alle joints in een temporale window tijdens een actie. Een joint kan bijvoorbeeld belangrijk zijn als het de hoogste verandering in hoek heeft.

Allereerst berekenen ze de hoeken tussen elk paar ledematen die met elkaar verbonden zijn met een joint. De tijdsreeks van deze hoeken wordt als temporale data beschouwd. De vector \mathbf{a}^i bevat de tijdsreeks van de hoeken voor joint i voor T frames. Een actie kan dan gezien worden als de verzameling van zulke vectoren:

$$A = [\mathbf{a}^1 \mathbf{a}^2 \dots \mathbf{a}^J]$$

Hierbij is J het aantal joints zodat A een $T \times J$ matrix is. Uit A kan het gemiddelde, de variantie en de maximale hoeksnelheid berekent worden voor elke joint. **ToDo: verder**

- **Inleiding:** Ze vermelden dat elke persoon een actie op een verschillende manier kan uitvoeren, maar dat zij toch dezelfde joints gebruiken om die actie uit te voeren. Ze berekenen de *relatieve informativeness* van alle joints in een temporale window tijdens een actie. Een joint is bv belangrijk als het de hoogste variantie heeft door de grootste verandering in hoek.

- **Methode:**

1. Ze berekenen de hoek tussen twee ledematen en gebruiken de tijdsreeks van deze hoeken als motion data. De vector \mathbf{a}^i bevat de tijdsreeks van de hoeken voor joint i voor T frames.
2. Een actie kan dan gezien worden als de verzameling van alle vectoren

$$A = [\mathbf{a}^1 \ \mathbf{a}^2 \ \dots \ \mathbf{a}^J]$$

J is het aantal joints, dus A is een $T \times J$ matrix.

3. Uit A kan bijvoorbeeld het gemiddelde, variantie en maximale hoeksnelheid berekent worden voor elke joint, hier aangeduid met de operatie \mathcal{O} , die een bewerking uitvoert op een vector en een scalaire waarde teruggeeft:

$$\mathcal{O}(\mathbf{a}) : \mathcal{R}^{|\mathbf{a}|} \mapsto \mathcal{R}$$

4. Een actiesequentie kan nu als feature beschreven worden

$$\mathbf{F} = \{\mathbf{f}_1, \mathbf{f}_2, \dots, \mathbf{f}_{N_s}\}$$

met

$$\mathbf{f}_k = [\mathcal{O}(\mathbf{a}_k^1) \ \mathcal{O}(\mathbf{a}_k^2) \ \dots \ \mathcal{O}(\mathbf{a}_k^J)]$$

Deze worden gesorteerd en zo wordt de SMIJ feature bekomen:

$$SMIJ = \{\{idof(sort(\mathbf{f}_k), n)\}_{k=1, \dots, N_s}\}_{n=1, \dots, N}$$

In other words, the SMIJ features represent an action sequence by encoding the set of N most informative joints at a specific time instant (by rank-ordering and keeping the top-ranking N joints) as well as the temporal evolution of the set of the most informative joints throughout the action sequence (by preserving the temporal order of the top-ranking N joints).

- **Gebruikte datasets:**

- /
- HDM05 database
- MSR Action3D dataset

- **Toekomst:**

- **Geen.** Ze geloven dat hun methode het meest ideale is voor praktische toepassingen.

3.2.6 Lie group

Groepentheorie kan ook gebruikt worden bij actieherkenning. De Lie groep SO_3 kan de relatieve drie-dimensionale rotaties tussen elk paar van joints representeren. Deze representatie vormt dan een punt in de Lie groep $SO_3 \times \dots \times SO_3$. Een actie is dan een curve in diezelfde Lie groep.

ToDo: verder

Bron [6]

- **Inleiding:** Nieuwe representatie voor skelet dat de 3D verhoudingen tussen lichaamsdelen expliciet modelleert, gebruik makend van rotaties en translaties. Zij maken een allereerst een onderscheid tussen *joint-based* en *body-based* actieherkenning
 - Joint-based: Het menselijk skelet is gewoon een verzameling van punten. Aantal verschillende methoden:
 - * Gebruik maken van enkel de joints posities [19], [7]
 - * Gebruik maken van een assenstelsel [5]
 - * Paarsgewijze relatieve joint posities [8], [9]
 - Body-based: Het menselijk skelet is een graaf van joints. [10], [25]
- **Methode:**
 1. Een skeletjoint kan met een ander skeletjoint beschreven worden met behulp van rotaties en translaties. M.a.w: de *relatieve geometrie* kan beschreven worden tussen twee punten.
 2. Zulke rotaties en translaties in drie dimensies maken deel uit van de *speciale Euclidische groep* $SE(3)$.
 3. De relatieve geometrie tussen twee joints is een punt in $SE(3)$. Het hele skelet is een punt in $SE(3) \times \dots \times SE(3)$.
 4. Acties kunnen nu gerepresenteerd worden door een curve in $SE(3) \times \dots \times SE(3)$. Op die curves wordt dan classificatie uitgevoerd.
 5. De groep $SE(3) \times \dots \times SE(3)$ is niet vlak, zodat traditionele classificatiemethoden zoals SVM analyse niet direct werken. Daarom wordt de groep eerst gemapt op zijn algebra
- **Gebruikte datasets:**
 - MSR-Action3D dataset
 - UTKinect-Action dataset
 - Florence3D-Action dataset
- **Toekomst:**
 - Zij bepalen de relatieve positie tussen elk paar van joints, maar elke actie wordt maar gekarakteriseerd door een specifieke verzameling van joints. Ze zoeken een manier om automatisch te bepalen welke verzameling van joints samen horen tijdens een actie.

Wordt opgelost in [8] met de actionlet mining

- Zij hebben dit ook maar uitgetest op één enkele persoon. Ze willen dit ook uitbreiden naar een multi-persoon model.

3.2.7 Grassmann manifold

Bron [26]

- **Inleiding:**
- **Methode:**
 1. De 3D joints worden geëxtraheerd als een tijdreeks. Via het autoregressive and moving average model wordt de dynamiek gemodelleerd. De subspace over de kolommen van de observability matrix is een punt op een grassmann manifold.
 2. Via de 'Riemannian geometry' van dit manifold kan het classificatieprobleem opgelost worden door Control Tangent ruimten te leren die het gemiddelde van elke klasse modelleren.
 3. Elke observatie wordt geprojecteerd op alle CTs om een Local Tangent Bundle representatie op te bouwen. Dit dient als input voor een SVM classifier.
- **Gebruikte datasets:**
 - MSRAction3D
 - UTKinect
 - UCFKinect
- **Toekomst:**
 - Uitbreiden zodat 'human behaviour recognition' mogelijk is
 - Extra features gebruiken zodat mens-object interactie kan herkend worden

3.2.8 Rollende rotaties

- **Inleiding:** Ze vermelden ook het verschil tussen joint based en body based:
 - **Joint based:**
 - * individuele joints [27], [28], [19]
 - * Pairwise joints [8], [29], [30], [9]
 - * Assenstelsel [5], [31]
 - **Part based:**
 - * Hoeken: [10], [11], [32]
 - * Bio-inspired : [25]
 - * relative geometry : [6]
- **Uitleg over de groepen:**

– SO_n

De speciale orthogonale groep (rotatiegroep) SO_n is een groep gevormd door de verzameling van $n \times n$ matrices R die voldoen aan

$$R^T R = R R^T = I_n$$

met $|R| = 1$. De elementen van SO_n bewerken punten in R^n via matrix-vector vermenigvuldiging:

$$SO_n \circ R^n \rightarrow R^n, R \circ p = Rp$$

• **Methode:**

1. Ze representeren een 3D skelet door de relatieve rotaties tussen alle paren van joints. Deze 3D rotaties zijn lid van de Lie group SO_3 , dus de representatie is een punt in de Lie groep $SO_3 \times \dots \times SO_3$. Door de relative 3D rotaties te gebruiken is de representatie schaalinvariant en halveert de feature dimensie, vergeleken met [6].
2. Een actie is dan een curve in de Lie groep $SO_3 \times \dots \times SO_3$. Ze berekenen een verwachte curve, en gebruiken DTW om elke andere curve te warpen naar die curve. De DTW berekeningen zijn gebaseerd op de kwadratische euclidische afstand in de Lie algebra. Ze hebben het ook geprobeerd met geodisic distance maar da macheert niet.
3. Het uitrollen van de Lie group $SO_3 \times \dots \times SO_3$ over zijn Lie algebra $so_3 \times \dots \times so_3$ samen met elke verwachte vurve. Al de acties worden 'unwrapped' op de Lie algebra. De rolling map voor een gegeven rolling curve kan bekomen worden (theorem 2 in paper)
4. Elke unwrapped actiecurve wordt een feature vector door alle temporale samples samen te voegen en te classificeren volgens een one-vs-all lineaire SVM classifier.

• **Gebruikte datasets:**

- FLorence3D dataset
- MSRPairs dataset
- G3D dataset

• **Toekomst:**

- Rolling map kan voor elke Riemannian manifold gebruikt worden. Zij willen het zeker uitproberen op andere zaken zoals Grassmann manifold en de manifold
- Zij gebruiken enkel skelet features. Indien deze gecombineerd kunnen worden met dieptebeelden zou de accuracy verhoogd kunnen worden.
- Zij gebruiken ook de kwadratische afstand. Het zou kunnen geoptimaliseerd worden door 'transported square root vector field' [33] te gebruiken aangezien deze invariant is voor temporale warping.

3.2.9 3DHoTs

Bron [34]

- **Inleiding:** Een nieuwe low-cost descriptor: 3D histograms of texture (3DHoTs) die features uit dieptebeelden haalt. Dit histogram wordt opgebouwd door de dieptebeelden te projecteren op drie orthogonale cartesische vlakken
- **Methode:**
 - 1.
- **Gebruikte datasets:**
 - MSRAAction3D dataset
 - MSRGesture3D dataset
 - UTD-MHAD dataset
 - MSRActivity3D
- **Toekomst:**
 - Uitbreiden zodat het ook werkt voor object herkenning en het opzoeken van images

Hoofdstuk 4

Classificatie

Op het moment dat een feature vector opgebouwd is voor een individuele frame of een verzameling van frames is het actieherkenningprobleem gereduceerd tot een classificatieprobleem. De bekomen feature vector wordt als input aan een classifier gegeven die dan tracht de juiste klasse toe te kennen op basis van de leerverzameling. Een eenvoudige classifier zou een gelijkaardigheidsfunctie $g(\mathbf{u}, \mathbf{v})$ die nagaat hoe dicht twee *feature vectoren* \mathbf{u} en \mathbf{v} op elkaar lijken. Dit is echter niet voldoende:

4.1 Leren

Ongeacht welke classifier gebruikt wordt, moet het eerst leren wat het moet herkennen. Daarom wordt er een leerverzameling \mathcal{L} opgebouwd. Deze verzameling bevat voorbeelden, en in het geval van *supervised learning* ook de klasse (of label) die bij het voorbeeld hoort. Bij actieherkenning bestaat de leerverzameling uit frames en de bijhorende klasse voor elke frame die de corresponderende actie voorstelt. Niet elk voorbeeld uit de leerverzameling wordt gebruikt om te leren: de leerverzameling wordt opgesplitst in een *training set* en een *testing set*. De *training set* wordt gebruikt tijdens de leerfase. In deze fase tracht de classifier optimale parameters te vinden om nieuwe voorbeelden te classificeren. De *testing set* wordt gebruikt na de leerfase om na te gaan hoe goed de classifier scoort en indien nodig wordt de leerfase opnieuw uitgevoerd met andere parameters of *feature vectoren*. Na de leer- en validatiefase komt de gebruiksfase, zodat de classifier kan gebruikt worden. De leerverzameling kan op verschillende manieren opgesplitst worden:

1. Er kan door de auteurs van de leerverzameling een voorgedefinieerde splitsing vastgelegd worden. Dit is de minst flexibele methode en wordt ook sterk afgeraden aangezien er een hoge *bias* kan zijn.
2. *Leave-p-out cross-validation* beschouwt p voorbeelden als de testing set en de overige voorbeelden als training set. Hierbij worden alle mogelijke combinaties uitgevoerd, zodat er $\binom{n}{p}$ keer het model getraind en gevalideert moet worden, met n de lengte van de hele leerverzameling. Het eenvoudigste geval komt voor bij $p = 1$, waardoor er $\binom{n}{1} = n$ validaties zijn, en wordt *leave-one-out cross-validation* genoemd. Deze methode is slechts aangewezen voor $p = 1$ aangezien deze methode exponentieel gedrag vertoont, waarbij n^p iteraties nodig zijn.

3. *k-fold cross-validation* verdeelt de hele leerverzameling in k gelijke groepen. De training set bevat dan $k - 1$ van deze groepen en de testing set bevat de overige groepen. Dit proces wordt k keer herhaald, zodat elk groep zeker eens in de testing set zit. De uitvoeringstijd is hier lineair want in het slechtste geval, voor $k = n$, is de methode exact gelijk aan *leave-one-out cross-validation*.

Het grote verschil tussen *k-fold cross-validation* en *Leave-p-out cross-validation* is dat de eerste slechts een benadering is van de tweede (voor $k \neq 1$). Bij actieherkenning speelt de uitvoeringstijd een minder grote rol. Een typische leerverzameling die gebruikt wordt bij actieherkenning bestaat uit een aantal acties, die uitgevoerd wordt door verschillende personen. Elke persoon heeft zijn eigen manier om een bepaalde actie, binnen de beperkingen, uit te voeren. Daarom is het nuttig om de leerverzameling op te splitsen door een variant van *Leave-p-out cross-validation*, *Leave-one-subject-out cross-validation*. Deze manier verloopt gelijkaardig aan *leave-one-out cross-validation*, maar hier worden de acties van één persoon als testing set beschouwd, terwijl de acties van alle andere personen de training set opmaken.

Voor actieherkenning kunnen drie algemene methoden beschreven worden:

1. Classificeren zonder de temporale dimensie expliciet te modelleren, *directe classificatie*;
2. Classificeren waarbij de temporale dimensie wel gemodelleerd wordt, *temporale modellen*;
3. Algemene classificatie zonder de actie te modelleren, *actiedetectie*.

4.2 Directe classificatie

Wanneer het temporale aspect verworpen wordt zijn er maar twee mogelijkheden: alle geobserveerde frames vervatten in één enkele representatie of actieherkenning uitvoeren op elke individuele frame.

4.2.1 *k*-nearest neighbours

Een voorbeeld van zo een algoritme is *k*-nearest neighbours. Deze methode maakt gebruik van de afstand van de geobserveerde feature vector tot elke andere feature vector uit de leerverzameling. Uit de k dichtste burens wordt dan de klasse genomen die het meest voorkomt. Deze operatie vraagt voor een grote leerverzameling veel rekenkost omdat elke feature vector vergeleken moet worden. De classificatie zal meer tijd vragen naarmate de leerverzameling groter is.

4.2.2 Lineaire classifiers

4.3 Temporale modellen

Bij temporale modellen zijn er twee grote klassen te onderscheiden: *generatieve* en *discriminerende* modellen. Een generatief algoritme modelleert hoe een input x wordt gegenereerd om zo een actieklass y toe te kennen en maakt gebruik van de gezamenlijke kansverdeling $P(x, y) = P(x|y)P(y)$. Een generatief model zal dus de kansverdeling van de leerverzameling modelleren en zal bij een nieuwe observatie **ToDo: wat zal het doen?**. Een discriminerend model zal de kans $P(y|x)$ direct modelleren zodat een directe mapping van x op y beschikbaar is. De onderliggende kansverdelingen worden dan ook niet berekend. Er bestaat een grote dis-

cussie over welk van deze twee modellen nu de voorkeur krijgen. Er is aangetoond [35] dat discriminerende modellen de voorkeur genieten.

4.3.1 Generatieve modellen

Een markovketen is een speciaal soort automaat die gebruikt kan worden bij het modelleren van temporale processen. Elke markovketen kent een aantal staten $S = \{s_1, \dots, s_n\}$ en een $n \times n$ transitiematrix T . Deze matrix bevat de waarschijnlijkheid om van één staat naar een andere staat te gaan. Een markovketen gaat uit van twee aassumpties:

1. Een verandering van toestand hangt enkel af van de vorige toestand. Dit wordt ook de *Markov eigenschap* genoemd.
2. De observatie behorend bij een toestand is onafhankelijk van elke andere observatie.

Een markovketen kent ook een uitvoeralfabet $A = \{a_1, \dots, a_k\}$ en een $n \times k$ uitvoermatrix U . Deze matrix bevat de waarschijnlijkheden dat een bepaalde staat s_i een uitvoer a_j genereert.

Een HMM is een uitbreiding van een Markov model waarbij de staat van elke toestand nu verborgen is. De toestanden zijn hier de verschillende fasen van een bepaalde actie. Voor een verzameling met n klassen $\Lambda = \lambda^1 \dots \lambda^n$ en een verzameling met k observaties $O = \{o_1 \dots o_k\}$, moet de juiste klasse λ geselecteerd worden die de kans op $P(\lambda|O)$ maximaliseert.

$$\lambda = \arg \max_{1 \leq i \leq n} P(O|\lambda^i)$$

4.3.2 Discriminerende modellen

Conditional Random Fields

Een HMM gaat ervan uit de observaties onafhankelijk zijn van elkaar, wat niet altijd het geval is. Een CRF is een voorbeeld van een discriminerend model. Een discriminerende classifier houdt rekening met meerdere observaties in de tijd en is geschikt voor de classificatie van een reeks van observaties. Een CRF gaat ervan uit dat de volgorde van observaties wel degelijk een impact heeft op de betekenis van deze observaties.

Dynamic Time Warping

DTW is een algoritme dat de gelijkenis tussen twee sequenties bestudeert, die verschillend kunnen zijn in snelheid. In actieherkenning lost dit het probleem van verschillen in actiesnelheid op. Een persoon die trager of sneller zwaait dan een persoon in de leerverzameling, kan via DTW toch herkend worden. Veronderstel twee verzamelingen van feature vectoren $X = \{x_1, x_2, \dots, x_N\}$ en $Y = \{y_1, y_2, \dots, y_M\}$, een kostfunctie $c(x, y)$ die de kost bepaald tussen twee feature vectoren en $p = \{p_1, \dots, p_L\}$ met $p_l = (n_l, m_l)$

Een DTW heeft bepaalde restricties:

- $p_1 = (1, 1)$ en $p_L = (N, M)$.
- $n_1 \leq n_2 \leq \dots \leq n_L$ en $m_1 \leq m_2 \leq \dots \leq m_L$.

4.4 Key frames

In plaats van elke frame te classificeren, zou een actie kunnen voorgesteld worden door *key frames*. Dit is een selectie van frames die een actie voldoende kunnen voorstellen zodanig dat verschillende acties nog steeds onderscheidbaar zijn en variaties van dezelfde actie hetzelfde gelabeld worden. In de literatuur bestaan er diverse manieren om zulke key frames te bepalen. De methode van [36] berekent een verschil op basis van de joints die de Kinect beschikbaar stelt. De methode van [37] maakt gebruik van randdetectie om silhouette te bekommen en zoekt een match tussen voorgedefinieerde silhouetten die een key frame van een actie voorstelt. Andere methoden maken ook gebruik van voorgedefinieerde exemplaren [38], [39],

Hoofdstuk 5

Evaluatie

5.1 Preprocessing

Om goede resultaten te verkrijgen bij actieherkenning, is het noodzakelijk om allereerst de gegevens te normaliseren. Belangrijke factoren zijn onder andere de positie van de persoon ten opzichte van de Kinect camera in zowel de horizontale, verticale en diepterichting, de schaal van de persoon zoals de lengte van de armen en benen, en de oriëntatie van de persoon ten opzichte van de camera. Voor de normalisatie is er een *referentiejoint* nodig om te voorkomen dat er gewerkt wordt met absolute waarden. Deze rol wordt vervuld door de *Spine Base Joint* (\mathbf{J}_{sb}) aangezien deze de enigste joint is die geen kind is van een andere *joints*. Een *joint* p is een ouder van een andere *joint* c als er een pad bestaat vanuit \mathbf{J}_{sb} naar c waarbij de voorlaatste knoop op dit pad p is. Figuur ?? toont de boomstructuur die deze relaties voorstelt.

De volgende drie onderdelen bespreken de verschillende stappen van de *preprocessing* fase.

Plaats-invariantie

Om ervoor te zorgen dat een persoon zich eender waar kan bevinden in het bereik van de Kinect camera, zonder invloed te hebben op de classificatie, wordt een translatie uitgevoerd zodat \mathbf{J}_{sb} zich in de oorsprong bevindt. Voor elke joint \mathbf{J}_i wordt zijn bijhorende puntvector \mathbf{P}_i aangepast via volgende bewerking:

$$\mathbf{P}'_i = \mathbf{P}_i - \mathbf{P}_{sb}$$

Rotatie-invariantie

Een persoon kan verschillende oriëntaties aannemen ten opzichte van de camera, maar toch dezelfde actie uitvoeren. Elke joint \mathbf{J}_i bevat een *quaternion* $\mathbf{Q}_i = b\mathbf{i} + c\mathbf{j} + d\mathbf{k} + w$ die de oriëntatie codeert van die joint. Aangezien \mathbf{J}_{sb} de referentiejoint is, zou deze de *reeële quaternion* moeten bedragen. Dit is de *quaternion* waarbij $w = 1$ en $b = c = d = 0$. Dit is mogelijk via het product van tussen \mathbf{Q}_{sb} en zijn conjugatie $\mathbf{Q}_{sb}^* = -b\mathbf{i} - c\mathbf{j} - d\mathbf{k} + w$. Voor elke joint \mathbf{J}_i moeten nu twee bewerkingen uitgevoerd worden, enerzijds op de quaternion van de joint en anderzijds op de drie-dimensionale coördinaten, waarop reeds een translatie is op uitgevoerd.

De eerste bewerking wijzigt de quaternion \mathbf{Q}_i door het product $\mathbf{Q}_i \mathbf{Q}_{sb}^*$ te nemen:

$$\mathbf{Q}''_i = \mathbf{Q}_i \mathbf{Q}_{sb}^*$$

De multiplicatie van twee quaternionen noemt men het *Hamilton product*, maar de bewerking volgt gewoon de distributieregel.

Enkel de oriëntaties aanpassen is niet voldoende. Ook moeten de drie-dimensionale coördinaten aangepast worden om de juiste vorm van het skelet te behouden. Dit komt neer op het roteren van de puntvector \mathbf{P}'_i via de rotatie beschreven door \mathbf{Q}_{sb} . Dit vertaalt zich in

$$\mathbf{P}''_i = \mathbf{Q}_{sb} \mathbf{P}'_i \mathbf{Q}_{sb}^*$$

Schaal-invariantie

_ToDo: toon figuur waarbij verschillende aanpassingen te zien zijn

5.2 Classifiers

De gebruikte dataset bestaat uit x acties. De *classifier* kan voor elk frame één van deze acties toekennen. Om de classifier te evalueren zou er een $x \times x$ frequentietabel opgesteld kunnen worden waarbij zowel de rijen als de kolommen gelabeld worden met de namen van de acties. De *classifier* heeft dan een correcte classificatie uitgevoerd als de rij en kolom overeenkomen. Een vereenvoudigd voorbeeld is te zien op tabel 5.1

	zwaaien	bukken	springen
zwaaien	5	2	0
bukken	1	11	4
springen	0	0	4

Tabel 5.1: Een voorbeeld van een 3×3 frequentietabel.

Het is echter moeilijk om hieruit eenvoudig relevante statistieken te berekenen. Het is daarom interessanter om de informatie van die tabel te aggregeren voor elke klasse om een zogenaamde *confusion matrix* te bekomen. Op die manier wordt een binair classificatiemodel gesimuleerd. Tabel 5.2 toont een voorbeeld van een *confusion matrix* voor de klasse **bukken**.

	bukken	niet-bukken
bukken	11 (TP)	5 (FP)
niet-bukken	2 (FN)	9 (TN)

Tabel 5.2: Een

Er zijn nu 4 statistieken beschikbaar. Ten eerste is er het aantal keer dat de *classifier* de actie 'bukken' heeft herkend terwijl het effectief bukken was. Dit wordt een True Positive (TP) genoemd. Ten tweede is er het aantal keer dat de *classifier* herkent heeft dat de persoon niet aan het bukken was en dat het werkelijk zo niet was. Dit wordt een True Negative (TN) genoemd. Deze twee statistieken tonen aan wanneer de *classifier* een juiste classificatie heeft gedaan voor deze specifieke actie. De overige twee statistieken zijn de False Positive (FP) en False Negative (FN) die respectievelijk het aantal keer aanduiden dat het bukken is terwijl het niet zo was, en dat het niet bukken is terwijl het wel zo was. *_ToDo: dit moet voor elke klasse uitgevoerd worden*

Aan de hand van deze vier waarden kunnen interessante statistieken berekend worden. Een populaire evaluatiemaat is het gebruik van de *precision* en *recall* statistieken, gedefinieerd als:

$$precision = \frac{TP}{TP + FP} \quad recall = \frac{TP}{TP + FN}$$

De *precision* bepaalt de kans dat een classificatie correct is als de classifier een positief resultaat geeft, terwijl de *recall* de kans bepaalt dat de classificatie een positief exemplaar als positief zal classificeren).

5.3 Segmentatie

Een real-time systeem moet een manier hebben om de continue stroom van invoergegevens nuttig op te splitsen in korte stukjes die het classificeren van de actie betrouwbaarder maakt. Men zou segmentatie kunnen weglaten en de stroom van gegevens frame per frame kunnen bekijken, maar dan is informatie van voorgaande frames verloren.

De segmentatie heeft als doel een buffer van acties op te bouwen. Hier is er een analogie bij spraakherkenning: de segmentatie bij dergelijke herkenningssystemen proberen bijvoorbeeld fonemen, of meer algemeen, woorden te segmenteren.

5.4 Herkenning

Hoofdstuk 6

notities

6.1 papers

Actieherkenning met skeletdata

- Bron [40]
 - 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 [41]:
 - 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 [42]:
 - 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 performantie-winst 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 [36]
 - 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

6.2 Machine learning

6.2.1 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 descriptor** 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 $f = x_1/x_2$.

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

6.2.2 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**

6.2.3 Hidden Markov Model

Bron [43]

Markov model

- Markov process = stochastisch process met volgende eigenschappen:
 - Het aantal toestanden $S = \{s_1, s_2, \dots, s_{|S|}\}$ is eindig.

- Observatie van een sequentie over de tijd $\mathbf{z} \in S^T$

Voorbeeld:

$S = \{sun, cloud, rain\}$ met $|S| = 3$ en $\mathbf{z} = \{z_1 = S_{sun}, z_2 = S_{cloud}, z_3 = S_{cloud}, z_4 = S_{rain}, z_5 = S_{cloud}\}$ met $T = 5$.

- Markov Assumpties:

1. Een toestand is enkel afhankelijk van de vorige toestand (**Markov Property**).

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

2. De waarschijnlijk is constant in de tijd

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

- Beginstaat $z_0 \equiv s_0$.

- Transitie matrix $A \in \mathbb{R}^{(|S|+1) \times (|S|+1)}$, met $A_{ij} = P(i \rightarrow j)$, :

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

- Twee vragen die een markov model kunnen oplossen:

1. Wat is de kans op een bepaalde toestanden vector \mathbf{z} ?

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

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

2. Gegeven \mathbf{z} , hoe worden best de parameters van A benaderd zodat de kans op \mathbf{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 transitie 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.

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 $x = \{x_1, x_2, \dots, x_T\}$ uit een alfabet $V = \{v_1, v_2, \dots, v_{|V|}\}$
 - Er is ook een sequentie van staten $z = \{z_1, z_2, \dots, z_T\}$ uit een alfabet $S = \{s_1, s_2, \dots, s_{|S|}\}$, maar deze zijn niet zichtbaar.
 - Transitie matrix A_{ij} wel bekend.
 - Kans dat een bepaalde output gegenereerd wordt in functie van de verborgen toestanden:

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

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 $\mathbf{z} \in S^T$ voor een verzameling van observaties $\mathbf{x} \in V^T$.
 - 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 die daar aan voldoen te concateneren. Andere manier is **Viterbi algoritme** die de hele toestandensequentie met de grootste waarschijnlijkheid teruggeeft.
 - Hulpvariabele:

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

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

Bibliografie

- [1] F. Deboeverie, S. Roegiers, G. Allebosch, P. Veelaert, and W. Philips, “Human gesture classification by brute-force machine learning for exergaming in physiotherapy,” in IEEE Conference on Computational Intelligence and Games, CIG, 2016.
- [2] M. Devi, S. Saharia, and D. D.K.Bhattacharyya, “Dance Gesture Recognition: A Survey,” International Journal of Computer Applications, vol. 122, no. 5, pp. 19–26, 2015.
- [3] K. Li, J. Wu, X. Zhao, and M. Tan, “Real-Time Human-Robot Interaction for a Service Robot Based on 3D Human Activity Recognition and Human-mimicking Decision Mechanism,” feb 2018. [Online]. Available: <http://arxiv.org/abs/1802.00272>
- [4] I. Ajili, M. Mallem, and J.-y. Didier, “Gesture recognition for humanoid robot teleoperation,” in 2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN). IEEE, aug 2017, pp. 1115–1120. [Online]. Available: <http://ieeexplore.ieee.org/document/8172443/>
- [5] L. Xia, C.-c. Chen, and J. Aggarwal, “View Invariant Human Action Recognition Using Histograms of 3D Joints,” CVPR 2012 HAU3D Workshop, pp. 20–27, 2012.
- [6] R. Vemulapalli, F. Arrate, and R. Chellappa, “Human action recognition by representing 3D skeletons as points in a lie group,” Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 588–595, 2014.
- [7] F. Lv and R. Nevatia, “Recognition and segmentation of 3-D human action using HMM and multi-class AdaBoost,” Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 3954 LNCS, pp. 359–372, 2006.
- [8] J. Wang, Z. Liu, Y. Wu, and J. Yuan, “Mining Actionlet Ensemble for Action Recognition with Depth Cameras,” 2012.
- [9] X. Yang and Y. L. Tian, “EigenJoints-based action recognition using Naïve-Bayes-Nearest-Neighbor,” in IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, 2012.
- [10] F. Ofli, R. Chaudhry, G. Kurillo, R. Vidal, and R. Bajcsy, “Sequence of the Most Informative Joints (SMIJ): A new representation for human skeletal action recognition,” in 2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, vol. 25, no. 1. IEEE, jun 2012, pp. 8–13. [Online]. Available: <http://ieeexplore.ieee.org/document/6239231/>

- [11] E. Ohn-Bar and M. M. Trivedi, "Joint angles similarities and HOG2for action recognition," IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, pp. 465–470, 2013.
- [12] R. Poppe, "A survey on vision-based human action recognition," Image and Vision Computing, vol. 28, no. 6, pp. 976–990, 2010.
- [13] P. Dollar, V. Rabaud, G. Cottrell, and S. Belongie, "Behaviour Recognition via Sparse Spatio-Temporal Features," 2005.
- [14] G. Willems, T. Tuytelaars, and L. V. Gool, "An efficient dense and scale-invariant spatio-temporal interest point detector," 2008.
- [15] H. Wang, A. Kl, C. Schmid, L. Cheng-lin, H. Wang, A. Kl, C. Schmid, L. C.-l. A. Recognition, and A. Kl, "Action Recognition by Dense Trajectories," Cvpr'11, 2011.
- [16] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, and M. Finocchio, "Real-time human pose recognition in parts from single depth images," CVPR, 2011.
- [17] W. Li, Z. Zhang, and Z. Liu, "Action recognition based on a bag of 3d points." IEEE, jun 2010, pp. 9–14. [Online]. Available: <https://www.microsoft.com/en-us/research/publication/action-recognition-based-bag-3d-points/>
- [18] J. Wang, Z. Liu, J. Chorowski, Z. Chen, and Y. Wu, "Robust 3d action recognition with random occupancy patterns," 2012.
- [19] M. E. Hussein, M. Torki, M. A. Gawayyed, and M. El-saban, "Human Action Recognition Using a Temporal Hierarchy of Covariance Descriptors on 3D Joint Locations," IJCAI International Joint Conference on Artificial Intelligence, 2013.
- [20] O. Tuzel, F. Porikli, and P. M. Vision, "Zusammenfassung: Region Covariance: A Fast Descriptor for Detection and Classification (2006)," European conference on computer, no. 2006, 2006. [Online]. Available: https://link.springer.com/chapter/10.1007/11744047_{-}45
- [21] Laptev and Lindeberg, "Space-time interest points," in Proceedings Ninth IEEE International Conference on Computer Vision. IEEE, 2003, pp. 432–439 vol.1. [Online]. Available: <http://ieeexplore.ieee.org/document/1238378/>
- [22] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," in 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), vol. 1. IEEE, 2015, pp. 886–893. [Online]. Available: <http://ieeexplore.ieee.org/document/1467360/>
- [23] L. Han, X. Wu, W. Liang, G. Hou, and Y. Jia, "Discriminative human action recognition in the learned hierarchical manifold space," Image and Vision Computing, vol. 28, no. 5, pp. 836–849, 2010.
- [24] M. Müller and T. Röder, "Motion Templates for Automatic Classification and Retrieval of Motion Capture Data," SCA, 2006.
- [25] R. Chaudhry, F. Ofli, G. Kurillo, R. Bajcsy, and R. Vidal, "Bio-inspired Dynamic 3D Discriminative Skeletal Features for Human Action Recognition," in 2013 IEEE Conference on Computer Vision and Pattern Recognition Workshops. IEEE, jun 2013, pp. 471–478. [Online]. Available: <http://ieeexplore.ieee.org/document/6595916/>

- [26] R. Slama, H. Wannous, M. Daoudi, and A. Srivastava, "Accurate 3D action recognition using learning on the Grassmann manifold," Pattern Recognition, vol. 48, no. 2, pp. 556–567, 2015. [Online]. Available: <http://dx.doi.org/10.1016/j.patcog.2014.08.011>
- [27] M. A. Gowayyed, M. Torki, M. E. Hussein, and M. El-Saban, "Histogram of Oriented Displacements (HOD): Describing trajectories of human joints for action recognition," IJCAI International Joint Conference on Artificial Intelligence, no. August, pp. 1351–1357, 2013.
- [28] J. Gu, X. Ding, S. Wang, and Y. Wu, "Action and gait recognition from recovered 3-D human joints," IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 2010.
- [29] J. Wang, Y. Wu, S. Rd, and E. Il, "Learning Maximum Margin Temporal Warping for Action Recognition," 2013.
- [30] P. Wei, N. Zheng, Y. Zhao, and S.-c. Zhu, "Concurrent Action Detection with Structural Prediction," in 2013 IEEE International Conference on Computer Vision, no. 1. IEEE, dec 2013, pp. 3136–3143. [Online]. Available: <http://ieeexplore.ieee.org/document/6751501/>
- [31] Zhanpeng Shao and Y. F. Li, "A new descriptor for multiple 3D motion trajectories recognition," in 2013 IEEE International Conference on Robotics and Automation. IEEE, may 2013, pp. 4749–4754. [Online]. Available: <http://ieeexplore.ieee.org/document/6631253/>
- [32] Jaeyong Sung, C. Ponce, B. Selman, and A. Saxena, "Unstructured human activity detection from RGBD images," in 2012 IEEE International Conference on Robotics and Automation. IEEE, may 2012, pp. 842–849. [Online]. Available: <http://ieeexplore.ieee.org/document/6224591/>
- [33] J. Su, S. Kurttek, E. Klassen, and A. Srivastava, "Statistical analysis of trajectories on Riemannian manifolds: Bird migration, hurricane tracking and video surveillance," The Annals of Applied Statistics, vol. 8, no. 1, pp. 530–552, mar 2014. [Online]. Available: <http://projecteuclid.org/euclid.aoas/1396966297>
- [34] B. Zhang, Y. Yang, and C. Chen, "Action recognition using 3d histograms of texture and a multi-class boosting classifier," IEEE Transactions on Image Processing, 2017.
- [35] A. Ng and M. Jordan, "On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes," 2002.
- [36] 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.
- [37] S. Carlsson and J. Sullivan, "Action recognition by shape matching to key frames," Workshop on Models versus Exemplars in Computer Vision, jan 2001.
- [38] D. Weinland and E. Boyer, "Action recognition using exemplar-based embedding," 26th IEEE Conference on Computer Vision and Pattern Recognition, CVPR, no. May, 2008.
- [39] A. Fathi and G. Mori, "Human pose estimation using motion exemplars," Proceedings of the IEEE International Conference on Computer Vision, pp. 1–8, 2007.

- [40] Y. Zhao, Y. Xiong, L. Wang, Z. Wu, X. Tang, and D. Lin, “Temporal Action Detection with Structured Segment Networks,” 2017.
- [41] G. Singh and F. Cuzzolin, “Untrimmed Video Classification for Activity Detection: submission to ActivityNet Challenge,” arXiv preprint arXiv:1607.01979, 2016.
- [42] J. Yuan, B. Ni, X. Yang, and A. Kassim, “Temporal Action Localization with Pyramid of Score Distribution Features,” 2016, pp. 3093–3102.
- [43] D. Ramage, “Hidden Markov Models Fundamentals,” 2007. [Online]. Available: <http://see.stanford.edu/materials/aimlcs229/cs229-hmm.pdf>