

Human Pose Estimation: Toepassing

Isaac venus
Stan Vanhecke
Mathieu Vanooteghem

KU Leuven Kulak, Wetenschap & Technologie

Titularis: Koen Van Den Abeele
Begleider: Jens Goemare
Academiejaar 2020 – 2021

Inhoudsopgave

1 Theoretische achtergrond	4
1.1 HPE, revolutionair?	4
1.2 Openpose	5
1.2.1 werking	5
1.3 Neurale netwerken	6
1.3.1 Opbouw van een neuraal netwerk	6
1.3.2 Trainen van een neuraal netwerk	6
2 Toepassingen	8
2.1 Toepassing 1: opvolgen van revalidatie na schouderoperatie	8
2.1.1 Bepalen van de hoek tussen arm en lichaam	8
2.1.2 Voorlopige resultaten	9
2.1.3 Voorlopige conclusies	11
2.2 Toepassing 2: fietspositie bepalen	11
2.2.1 Principe	11
2.2.2 Algoritme voor wijzigen van de zadelhoogte	12
2.2.3 Algoritme voor wijzigen van de stuurpenlengte	13
2.2.4 Conclusies	15
2.3 Toepassing 3:	17
2.3.1 Richtlijnen voor een goede squat	17
3 Vakintegratie	19
4 Planning	20

Inleiding

Aan materiaal in de medische wereld hangt steeds een stevig prijskaartje. Wij vroegen ons af hoe we deze kost kunnen verlichten en verschillende medische toepassingen beter beschikbaar kunnen maken voor de gewone burger. Daarom maken we gebruik van technologie die iedereen ter beschikking heeft: een laptop of een smartphone. In dit verslag richten wij ons op *human pose estimation* (HPE), of lichaamspositiebepaling. We gebruiken hiervoor de open-source HPE software Openpose die via neurale netwerken de positie van een persoon schat op een foto en de coördinaten van verschillende knooppunten in het menselijk lichaam teruggeeft waarmee we dan berekeningen kunnen uitvoeren. In dit verslag zullen we er twee uitwerken: het opvolgen van de revalidatie na een schouderoperatie en het bepalen van de optimale fietspositie om blessures te voorkomen. In beide gevallen kunnen we met behulp van HPE een goede en goedkope oplossing aanbieden die gemakkelijk toegankelijk is voor iedereen.

Hoofdstuk 1

Theoretische achtergrond

1.1 HPE, revolutionair?

Human pose estimation is de laatste tijd geëvolueerd naar een heel gebruiksvriendelijke vorm. Vroeger waren hiervoor hoogtechnologische opstellingen met mensen in speciale pakken en veel rekenkracht nodig. Nu zijn een foto en een laptop voldoende om de posities van zelfs meerdere personen in hetzelfde beeld tegelijkertijd te bepalen. Dit allemaal door de gigantische vooruitgang op vlak van artificiële intelligentie en neurale netwerken.

Een 3D *motion capture* pak zorgt wel nog altijd voor een preciezere lichaamspositiebepaling dan de schatting die Openpose maakt. Daarom wordt dit nog altijd gebruikt bij grote producties zoals deze van films en games (zie Figuur 1.1).



Figuur 1.1: Bij grote productiehuizen wordt de positie bepaald door middel van een 3D motion capture pak (links), maar tegenwoordig is HPE een veel goedkopere en gemakkelijkere oplossing (rechts). (afbeeldingen van Pinterest en beyondminds.ai)

1.2 Openpose

Over het algemeen zijn er 2 manieren om menselijke poses te schatten. Ten eerst heb je de top-down aanpak. Hierbij maak je eerst gebruik van een personen detector en schat je dan voor elke persoon apart hun pose. Deze methode heeft enkele minpunten namelijk dat de runtime evenredig is met het aantal personen en als een persoon in de eerste stap niet herkend is, is er geen mogelijk om de fout toch nog goed te maken. Een bottom-up aanpak daarentegen heeft deze problemen niet. Deze methode bepaalt alle belangrijke knooppunten en probeert dan de punten op de juiste manier aan elkaar te linken zodat het de poses van verschillende mensen vormt. Openpose is een voorbeeld van zo'n bottom-up aanpak en is een van de meest efficiënte neurale netwerken voor positiebepaling. Openpose was de eerste open-source library voor realtime lichaams-, voet-, hand- en gezichts- detectie. Het maakt gebruik van een “two-branch multistage CNN”, heel kort betekent dit het volgende: CNN staat voor convolutioneel neuraal netwerk, “two-branch” wijst op het feit dat het CNN twee takken heeft en multistage wil zeggen dat het netwerk meerdere keren op elkaar is gestapeld.

1.2.1 werking

Het systeem neemt als input een kleurfoto met grootte $w \times h$ en geeft als output voor elke persoon op de foto de 2D locaties van zijn lichaamsdelen. Eerst wordt een eerste voorspelling gedaan van de set van 2D vectorvelden (\mathbf{L}), de *part affinity fields* (PAF's), die de graad van associatie tussen de lichaamsdelen vastleggen en van de 2D betrouwbaarheidskaarten (\mathbf{S}) van de lichaamsdelen. Dit zijn respectievelijk de eerste en de tweede tak van het CNN. Deze eerste voorspelling wordt dan door het netwerk verbeterd. Vervolgens wordt deze verbeterde voorspelling samen met de initiële voorspelling nog eens gevoerd aan het netwerk. Na zes keer is het resultaat goed genoeg en wordt het samengevoegd om de uitendelijk output te geven. Omdat het netwerk meerdere keren doorlopen wordt noemt men dit een *multistage* CNN. De volledige uitleg van het netwerk en hoe men van de heatmaps en PAF's tot het uiteindelijk resultaat komt vind je in het origineel artikel.

De set \mathbf{L} bestaat uit C vectorvelden, één per connectie tussen twee knooppunten, $\mathbf{L} = (\mathbf{L}_1, \mathbf{L}_2, \dots, \mathbf{L}_C)$. Elke $\mathbf{L}_c \in \mathbb{R}^{w \times h \times 2}$, met $c \in \{1 \dots C\}$. De PAF's zijn velden die de graad van associatie representeren tussen de verschillende lichaamsdelen. Er wordt gestart met een set van connecties tussen twee punten, dit zijn de lijnen op figuur... Voor elke connectie wordt dan zo'n veld opgesteld. Het is eigenlijk een tensor met in elke cel een 2d-vector die de richting van de connectie aangeeft, bijvoorbeeld van de rechterschouder naar de rechter elleboog. De set $\mathbf{S} = (\mathbf{S}_1, \mathbf{S}_2, \dots, \mathbf{S}_j)$ bestaat dus uit \mathbf{J} betrouwbaarheidskaarten, één per lichaamsdeel en $\mathbf{S}_j \in \mathbb{R}^{w \times h}$, $j \in \{1 \dots J\}$. Deze worden heel duidelijk voorgesteld met heatmaps. Hierop zie je voor elke positie de kans dat het lichaamsdeel zich daar bevindt. Indien er meerdere mensen op de foto staan dan zal je meerdere punten zien met een hoge kans, zie figuur...

Zoals eerder gezegd worden uiteindelijk de betrouwbaarheidskaarten en PAF's samengevoegd om als output de 2D knooppunten van alle mensen op de foto te geven.

1.3 Neurale netwerken

Een (artificieel) neuraal netwerk of ANN is een netwerk geïnspireerd op een biologisch neuraal netwerk zoals dat voorkomt in hersenen, met als doel deze iets te doen leren. Het ANN is opgebouwd uit artificiële neuronen die een neuron in een brein moeten voorstellen. Al de neuronen zijn verbonden met andere neuronen en kunnen signalen uitsturen en ontvangen. In een ANN zijn de signalen getallen en kan elk neuron een bepaalde bewerking uitvoeren op een binnenkomend signaal om deze verder te sturen. Elke connectie heeft ook een gewicht dat bepaalt hoe sterk het signaal is dat hij uitstuurt. Dit gewicht past zich aan als het neuraal netwerk leert.

1.3.1 Opbouw van een neuraal netwerk

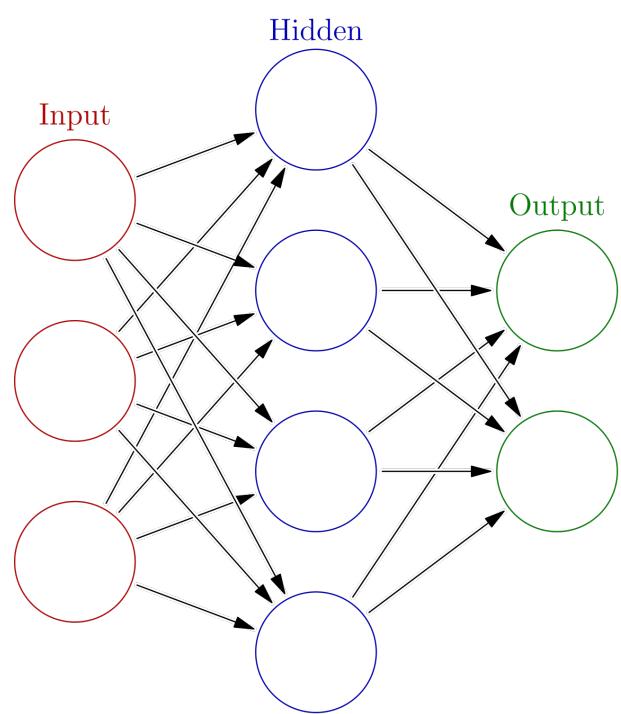
Neuronen Zoals er hierboven staat, is een artificieel neuron geïnspireerd op een neuron uit een brein. Elk neuron heeft een of meerdere inputs en één output die kan verzonden worden naar meerdere andere neuronen. De input van een neuron kan ofwel komen van de data die in het neuraal netwerk wordt gestoken of van andere neuronen. De output van de output neuronen is wat uit het programma komt. Om de output van een neuron te berekenen neem je de gewogen som van de inputs, met als gewicht de waarden van de connecties. Daarbij wordt ook nog eens een bias of vertrekking opgeteld.

Connecties Om te kunnen leren moet het natuurlijk mogelijk zijn om de data door te geven van het ene neuron naar het andere. Dit verloopt via een “connectie” die een bepaald gewicht heeft dat de sterkte en daarmee ook het belang van een bepaalde connectie weergeeft.

Lagen Een neuraal netwerk bestaat uit meerdere lagen van neuronen, meer bepaald de *input layer*, *hidden layers* en *output layer* 1.2. Zoals de namen wel duidelijk maken geef je de data aan de *input layer* en komen de berekende waarden uit de *output layer*. Daartussen zijn er eventueel *hidden layers*. Tussen twee lagen zijn er meerdere manieren van connecties tussen de neuronen mogelijk.

1.3.2 Trainen van een neuraal netwerk

Bij het programmeren van een neuraal netwerk weet je natuurlijk niet wat de gewichten zijn van alle connecties. Je moet het model dus doen leren. Dit doe je door de gewichten (over meerdere leercycli) aan te passen zodat de output zo goed mogelijk past bij het gewenste resultaat. Er zal altijd een bepaalde fout op de output zitten dus het model gaat nooit perfect zijn. Het trainen van een model is gedaan als de fout op de output niet meer verkleint en dus een minimum heeft bereikt. Het aanpassen van de gewichten gebeurt met een zekere *learning rate*. Dit bepaalt de grootte van de correcties die gebeuren bij de gewichten. Bij een grote *learning rate* kan je model sneller klaar zijn met leren, maar is er meer kans op een grotere fout op de output. Een kleine *learning rate* zal het trainen vertragen maar je uiteindelijke resultaat zal beter zijn. Openpose is getraind met twee datasets [2]: de COCO dataset, die bestaat uit meer dan 100.000 beelden en meer dan 1.000.000 *keypoints*, en de MPII dataset.



Figuur 1.2: Voorbeeld van een neuraal netwerk. (afbeelding van wikipedia.org)

Hoofdstuk 2

Toepassingen

Zoals eerder al gezegd zullen we ons focussen op toepassingen in de medische wereld waarop HPE kan toegepast worden. Wij kunnen dan een goedkope oplossing ontwikkelen voor die toepassingen. Hierbij testen we mogelijke problemen en analyseren we of dit een goede optie is. We proberen dit dan allemaal op een methodische manier uit te werken, waarbij we een specifiek geval steeds breder gaan bekijken.

Een eerste toepassing is het opvolgen van de revalidatie na een schouderoperatie. Met een foto gemaakt met een gsm kunnen we een analyse uitvoeren van de beweging van de schouder na de operatie en dan het doorheen de revalidatie opvolgen.

Een tweede en uitgebreidere toepassing is het analyseren van de positie op de fiets. Veel amateurwielrenners hebben een slechte houding op de fiets en een professionele *bikefitting* kost al snel een paar honderd euro. Via lichaamspositiebepaling kunnen wij met een simpele foto de positie bepalen op de fiets en dan correcties voorstellen. Dit is een goedkope oplossing die voor een zeer breed publiek inzetbaar is. Voor professionele wielrenners zal dit wel waarschijnlijk niet voldoende zijn, maar voor een amateur kan dit een goedkoop alternatief zijn.

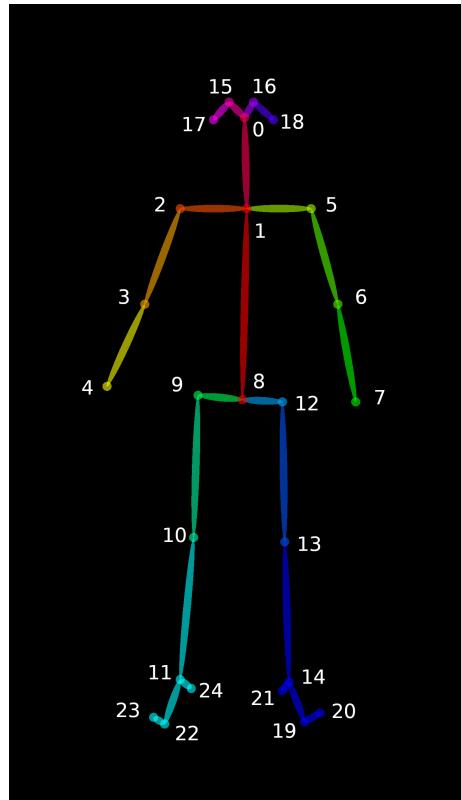
2.1 Toepassing 1: opvolgen van revalidatie na schouderoperatie

2.1.1 Bepalen van de hoek tussen arm en lichaam

We willen meten hoe ver een persoon zijn arm kan roteren voor de opvolging van de revalidatie na een schouderoperatie. Hiervoor kunnen we de hoek tussen het opperarmbeen en een ander lichaamsdeel bepalen. Op figuur 2.1 komt dat dus neer op de hoek bepalen tussen de lijnstukken [32] en [21] of tussen de lijnstukken [15] en [65]. Als we Openpose gebruiken om de positie te schatten van een persoon op een foto krijgen we als output de coördinaten van de verschillende knooppunten. Met deze coördinaten hebben we voldoende informatie om de hoek te berekenen en zo objectieve informatie te krijgen over het verloop van de revalidatie.

Bij een foto vanuit vooraanzicht kunnen we berekenen hoe ver de patiënt

zijn arm zijwaarts omhoog kan brengen. Bij een foto genomen vanuit zijaanzicht kunnen we berekenen hoe ver hij de arm vooruit kan omhoog steken. Het is wel belangrijk dat de foto altijd vanuit dezelfde positie wordt getrokken omdat er anders variatie kan zijn op de hoek.

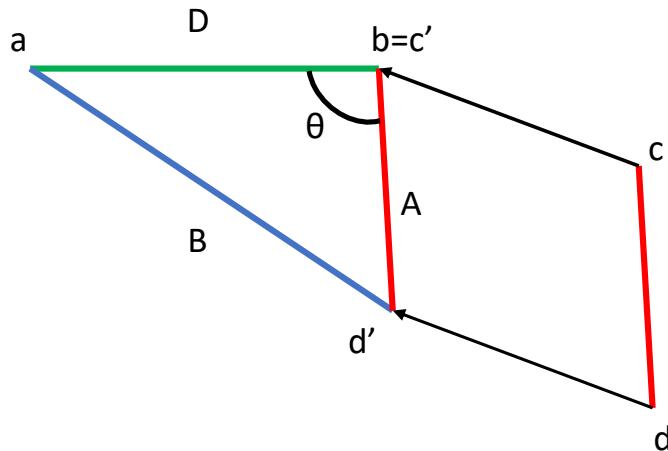


Figuur 2.1: Voorstelling van de positie bepaald via Openpose (afbeelding van openpose)

2.1.2 Voorlopige resultaten

Via Openpose krijgen we de coördinaten van alle punten die openpose kan herkennen. Hiermee kunnen we dan verder onderzoek doen. Op dit moment zijn we in staat om hoeken tussen bepaalde lichaamsdelen te berekenen met als doel objectieve gegevens te bekomen tijdens bijvoorbeeld een revalidatie na een ongeval.

Wiskundige achtergrond Om de hoek tussen twee lichaamsdelen te berekenen gebruiken we de cosinusregel. Hier volgt een klein beetje duiding over hoe we die precies gebruiken (zie figuur 2.2).



Figuur 2.2: Hoek tussen 2 lichaamsdelen

We stellen a, b de coördinaten die het eerste lichaamsdeel afbakenen en c, d de coördinaten van het tweede lichaamsdeel. De hoek tussen de lichaamsdelen is dan de θ van op de figuur. We kunnen gewoon de cosinusregel gebruiken om de hoek te bepalen als we c op b leggen. We krijgen dan

$$B^2 = A^2 + D^2 - 2 \cdot A \cdot D \cos \theta$$

en dus

$$\cos \theta = \frac{A^2 + D^2 - B^2}{2 \cdot A \cdot D}$$

met

$$A = \sqrt{(d_x - c_x)^2 + (d_y - c_y)^2}$$

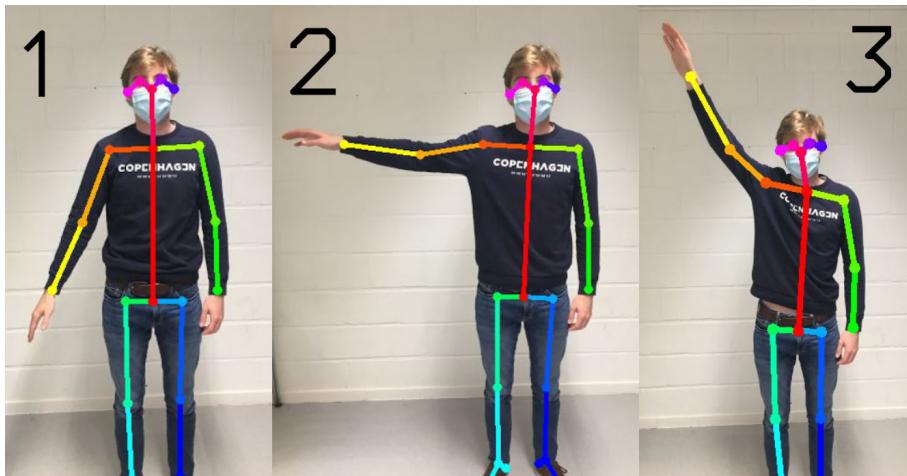
$$B = \sqrt{(b_x - a_x + d_x - c_x)^2 + (b_y - a_y + d_y - c_y)^2}$$

$$D = \sqrt{(b_x - a_x)^2 + (b_y - a_y)^2}$$

Als we dus de 4 coördinaten weten kunnen we de hoek bepalen.

Voorbeeld Een voorbeeld bij het bepalen van een hoek tussen de bovenarm en de romp is afbeelding 2.3. Als we de hoek berekenen tussen de ruggengraat (rood) en het opperarmbeen (licht oranje) krijgen we als waarden:

- Foto 1: 21.63 graden
- Foto 2: 78.06 graden
- Foto 3 130.00 graden



Figuur 2.3: Voorbeeld bij het bepalen van een hoek.

2.1.3 Voorlopige conclusies

Door deze toepassing weten we dat het relatief eenvoudig is om met Openpose zinvolle berekeningen te doen. Door de coördinaten die gegeven worden als output kunnen we gemakkelijk programma's schrijven die hiermee werken. Doordat Openpose werkt in 2D moeten de foto's recht worden genomen. Als de hoek die we willen berekenen schuin afgebeeld staat op de inputfoto komt de berekening niet overeen met de werkelijke waarde van de hoek.

2.2 Toepassing 2: fietspositie bepalen

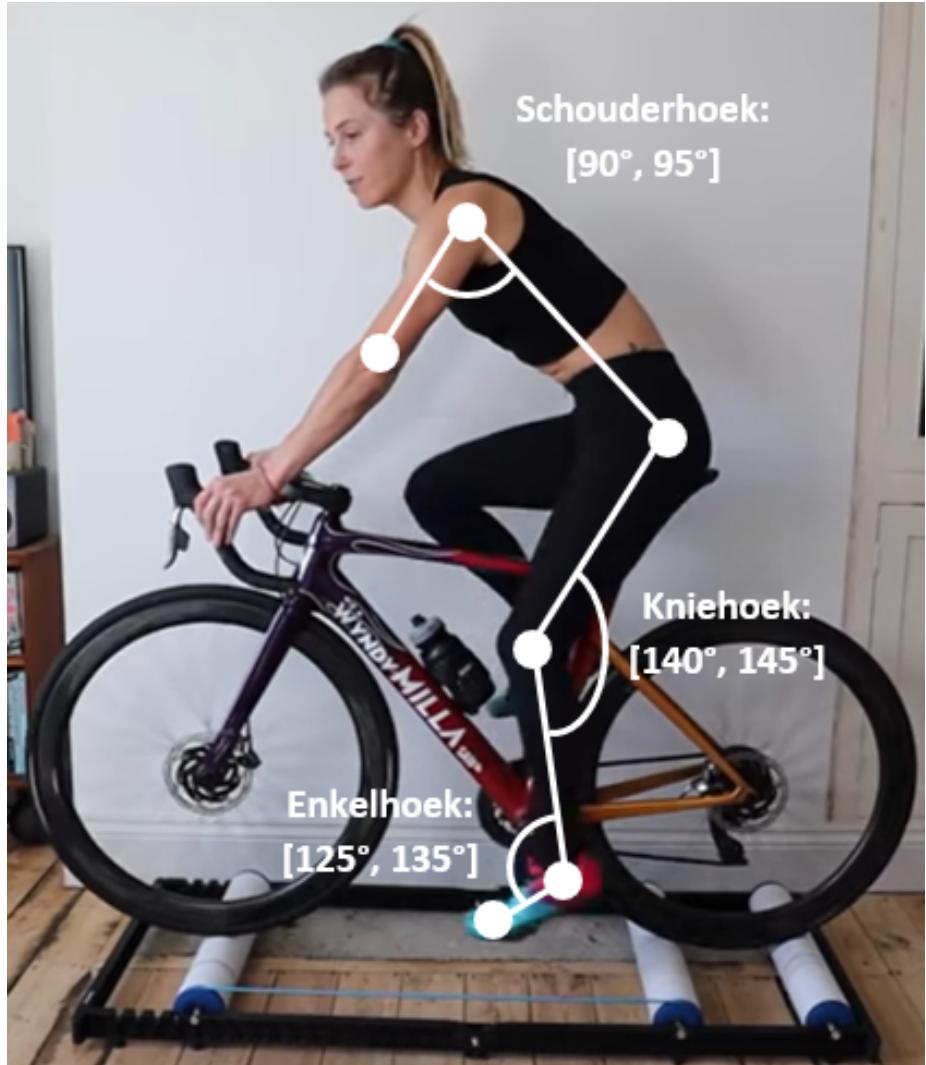
Voor de tweede toepassing willen we een goedkoper alternatief bieden voor een bikefitting. Hierbij worden er foto's getrokken van de persoon op de fiets en kunnen we dan op basis van de positie bepaald met Openpose eventuele correcties uitvoeren op de fiets.

Een *bikefit* is eigenlijk het analyseren en eventueel aanpassen van de positie op de fiets. Het belangrijkste doel daarvan is het voorkomen van blessures. Bij wielrenners die aan competitie doen heeft een *bikefit* ook als doel om een zo aerodynamisch mogelijke positie op de fiets aan te nemen. Ook kan een betere positie ervoor zorgen dat men een groter vermogen kan leveren op de pedalen. Alleen maar voordelen dus.

2.2.1 Principe

Op Figuur 2.4 staan enkele hoeken die voor een persoon als optimaal gezien worden voor de positie op een koersfiets. Deze hoeken kunnen wel wat variëren afhankelijk van hoe flexibel de persoon is. Met behulp van Openpose berekenen we deze hoeken met als input een foto of video van de wielrenner. Het is belangrijk dat de foto genomen wordt vanuit een zo loodrecht mogelijk zijaanzicht en dat de wielrenner het stuur vasthoudt bij de *shifters*. Ideaal staat wordt de foto

getrokken op ongeveer de helft van de hoogte van de fietser als die zit. Als de berekende hoeken te veel afwijken van wat voorop gesteld wordt, dan kunnen we een verbetering voorstellen. Dit kan een verhoging van het zadel zijn of een verandering van de lengte van de stuurstangen.



Figuur 2.4: Belangrijkste hoeken voor een optimale positie op de fiets (hoeken zijn gebaseerd op presentatie over bikefitting door Wolf Performance, bijgevoond door onze begeleider Jens Goemaere)

2.2.2 Algoritme voor wijzigen van de zadelhoogte

We zullen beginnen met een algoritme te schrijven voor het veranderen van één parameter, namelijk de zadelhoogte. Voornamelijk de kniehoek zal variëren als gevolg van een wijziging in zadelhoogte. Er zal ook een klein effect hebben op

de schouderhoek maar dit laten we hier voor het gemak achterwege. De mate waarin je de zadelhoogte kan wijzigen hangt af van de lengte van de zadelpen. Op de meeste zadelpennen staat de minimale diepte waarmee de pen in het frame moet zitten. Indien dit niet zo is, is de vuistregel dat de zadelpen minstens 90 mm diep in het frame moet zitten. Zet je zadel dus zeker niet hoger dan dat. Als je dit niet doet, bestaat de kans dat de zadelpen breekt. Eerst berekenen we de kniehoek. We kijken initieel hoe groot die is. Indien de hoek tussen 140 ° en 145 ° ligt (zie Figuur 2.4), wordt er geen voorstel gedaan voor een wijziging van de zadelhoogte. Indien de hoek buiten dit interval ligt, berekent het algoritme met hoeveel pixels de y-coördinaat van het knooppunt van de heup moet wijzigen om de hoek in dit interval te krijgen. Vooraf wordt door het programma de lengte van bijvoorbeeld het bovenbeen gevraagd in cm. Via openpose kunnen we dan het aantal pixels berekenen tussen het knooppunt van de knie en de heup. Op die manier kunnen we de eenheid pixels overzetten naar cm. De output is dan het aantal cm dat het algoritme voorstelt om het zadel te wijzigen. Een positief getal betekent verhogen, een negatief verlagen.

Wiskundige redenering en implementatie Het programma veronderstelt een aantal constanten; natuurlijk de lengten van het boven- en onderbeen, punt B en de x-coördinaat van A die te zien zijn op figuur 2.5. Met deze veronderstellingen kunnen we de hoek van de knie θ kiezen en de y-coördinaat van A berekenen om dan de verandering van de hoogte van het zadel te bepalen.

We kunnen de lengte van c bepalen met de gewenste hoek θ .

$$c^2 = a^2 + b^2 - 2 \cdot a \cdot b \cdot \cos(\theta)$$

Voor deze lengte geldt ook dat:

$$c = \sqrt{(A_x - B_x)^2 + (A_y - B_y)^2}$$

Waaruit volgt dat

$$A_y = \sqrt{c^2 - (A_x - B_x)^2} + B_y$$

Dit kunnen we dan vergelijken met de hoogte van de heup en dan een correctie voorstellen van $heup_y - A_y$.

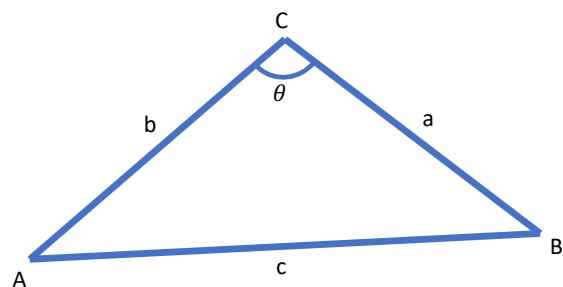
2.2.3 Algoritme voor wijzigen van de stuurpenlengte

Nu kijken we naar de invloed van de lengte van de stuurpen. Dit heeft enkel invloed op de schouderhoek. Het probleem is dat als we de lengte van de stuurpen willen aanpassen, we een nieuwe stuurpen moeten kopen. Er zijn ook bepaalde grenzen aan de lengte van een stuurpen. De minimale lengte is 40 mm, korter is bijna onmogelijk. Het maximum ligt rond de 140 mm, groter kan nog maar dat is heel uitzonderlijk. De output van het algoritme is het aantal cm dat het stuur naar voor of naar achter moet.

Wiskundige redenering en implementatie We willen graag de verandering in de x-coördinaat van de pols (B_x) berekenen om de gewenste hoek θ



Figuur 2.5: Verduidelijkende figuur bij het bepalen van de zadelhoogte. Afbeelding van redbull.com



Figuur 2.6: Verduidelijkende figuur bij het bepalen van de stuurpenlengte.

tussen de rug en de arm te bekomen. Hiervoor gebruiken we weer de cosinusregel om de lengte tussen de heup en de pols te bepalen als deze in de goede positie staat.

$$c^2 = a^2 + b^2 - 2 \cdot a \cdot b \cdot \cos(\theta)$$

Met a de lengte van de rug, b de lengte van de arm (schouder tot pols) en c de lengte tussen de heup en de pols. We weten ook dat

$$c^2 = (B_x - A_x)^2 + (B_y - A_y)^2$$

Waaruit volgt dat

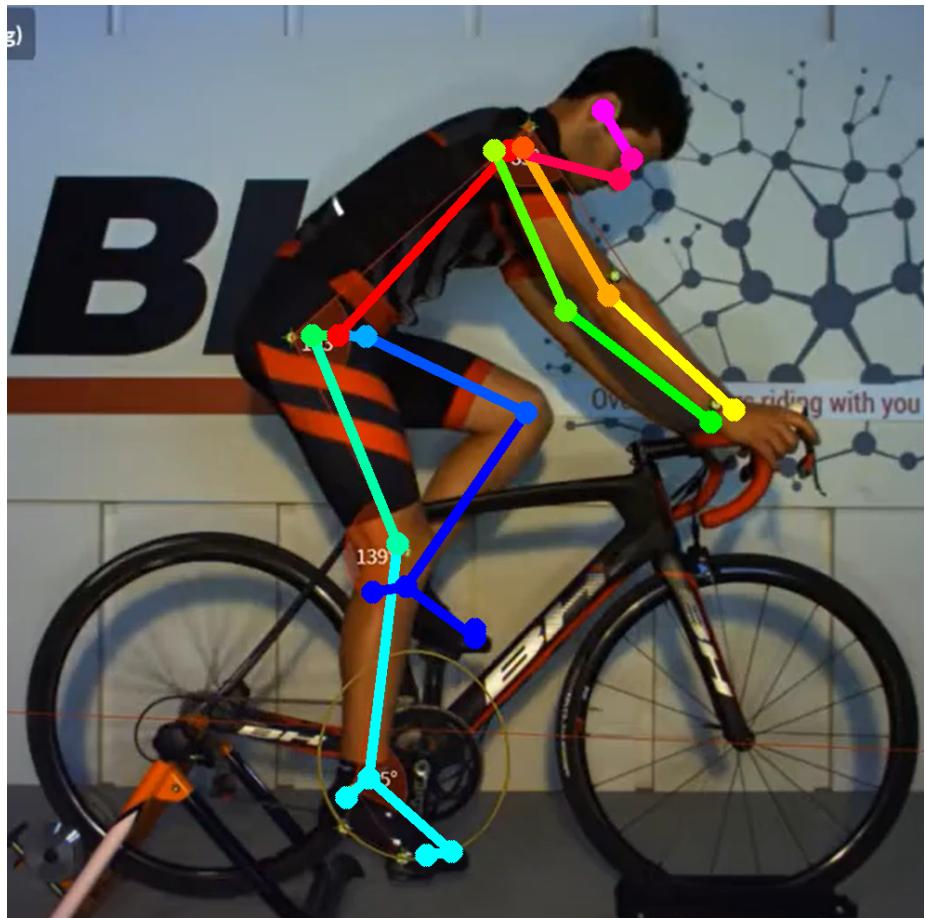
$$B_x = \pm \sqrt{c^2 - (B_y - A_y)^2} + A_x$$

We weten nu welke x-coördinaat de pols zou moeten hebben om de hoek θ te bekomen tussen de rug en de arm. Hiermee kunnen we een wijziging in de positie van het stuur voorstellen. Of je een + of - moet nemen hangt af van de richting waarin de persoon op de fiets kijkt.

2.2.4 Conclusies

invloed van de resolutie Om de invloed van de resolutie van de genomen foto na te gaan, zijn we gestart met een foto van 1500 op 1500 pixels als referentie. We hebben deze foto dan herschaald naar foto's van lagere en hogere resoluties. Bij foto's die een hogere resolutie hadden dan 500x500 bleven de hoeken tamelijk stabiel met een maximale spreiding van 2° . Bij foto's met een nog kleinere resolutie werd dit al snel 7° . Dit is een veel te grote fout voor een bikefitting. Als we dan keken naar de afwijking van de voorgestelde verhoging/verlaging van het zadel kwamen we op een veel grotere spreiding. Bij foto's van een resolutie hoger dan 1000x1000 bedroeg de afwijking 2 tot 3 mm, maar bij lagere resoluties liep dit op tot 5 à 6 mm. Dit is een veel te grote foutenmarge voor een bikefitting waar juist die enkele millimeters bepalend kunnen zijn.

Openpose maakt slechts een schatting van de positie Openpose kiest niet altijd de keypoints juist op de gewrichten omdat het slechts een schatting maakt van de pose. Hierdoor kom je vaak een andere waarde uit voor een hoek dan de werkelijke waarde. Dit vormt een probleem aangezien we de optimale zadelhoogte baseren op werkelijke hoeken. We hebben online een foto van een professionele bikefitting gevonden en ons programma daarop laten lopen. Op Figuur 2.2.4 is duidelijk te zien dat een afwijking van de keypoints een grote invloed heeft op de aanpassingen die ons algoritme voorstelt. Zo ligt het juiste knooppunt van de heup verder naar achter en van de schouder meer naar boven. Ook dat van de arm ligt meer schuin naar boven. Al deze afwijkingen van de keypoints geschat door Openpose zorgen voor fouten op ons algoritme en de aanpassingen die voorgesteld worden.



Figuur 2.7: foto van professionele bikefitting waarop we Openpose laten lopen hebben om het verschil te analyseren. (screenshot van <https://www.youtube.com/watch?v=CTYmJi6zH6oabchannel=STTSsystems>)

Omzetten van pixels naar centimeter De output van Openpose zijn de coördinaten van de keypoints. De eenheid van deze coördinaten is echter pixels en is dus relatief ten opzichte van de resolutie van de foto. Om over te gaan naar een absolute eenheid vroegen we als input de lengte van het dijbeen die we dan ook kunnen berekenen in openpose en zo bepalen hoeveel pixels er in een cm gaan. Het grote probleem is nu dat je nooit op dezelfde manier de lengte van je dijbeen zal meten zoals openpose die berekent. Als je dus van bij de start al te maken hebt met fouten zal je dus nooit de juiste waarde kunnen uitkomen.

Probleem bij bepalen van de schouderhoek De ideale schouderhoek is 90° . Openpose ziet de rug echter als een recht stuk terwijl er in het echt wel enige buiging in zit. Daardoor kan de hoek in het echt rond de 90° liggen terwijl Openpose een compleet andere waarde zal geven.

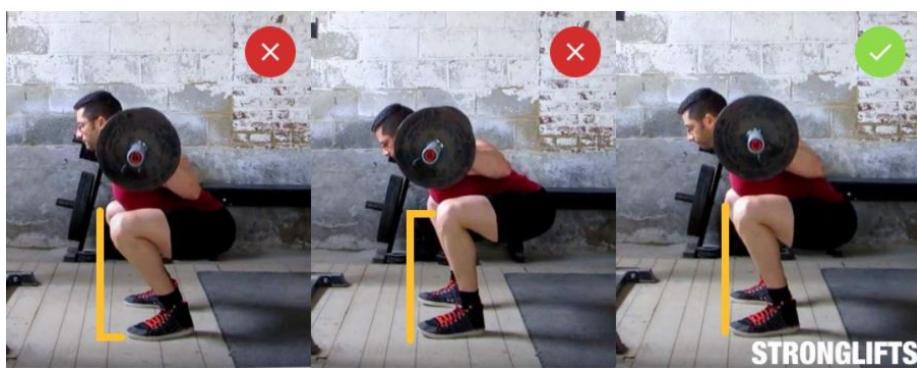
Besluit Bikefitting is een heel precieze wetenschap. Het gaat om die laatste millimeters waarmee de parameters aangepast worden. We hebben echter gemerkt dat openpose hier niet de goede toepassing voor is om de redenen die hierboven opgeliist staan. We concluderen dus dat een bikefitting programma gebaseerd op openpose veel te omslachtig is en dat je waarschijnlijk een beter en sneller resultaat zal hebben indien je de zadelhoogte op het gevoel aanpast in de wetenschap dat je knie nog lichtjes gebogen moet zijn als je voet beneden staat.

2.3 Toepassing 3:

Als derde toepassing maken we een programma dat helpt om op de correcte manier een squat uit te voeren. Dit is meer geschikt voor Openpose omdat het niet aankomt op millimeters. Als input wordt een filmpje van een aantal squats gevraagd. Er wordt dan per squat een score gegeven op basis van een aantal richtlijnen voor het uitvoeren van een goeie squat.[1]

2.3.1 Richtlijnen voor een goeie squat

Met Openpose kunnen we twee parameters goed bepalen. Ten eerste controleren we de positie van de knie. Deze moet zich op het laagste punt van de squat recht boven de tenen bevinden zoals op Figuur 2.8. Daarnaast moeten ook de heup en de knie zich op dezelfde hoogte bevinden op het laagste punt van de squat zoals op Figuur 2.3.1. Bij deze twee voorschriften komt het niet aan op millimeters en dus ideaal voor Openpose. Andere richtlijnen zoals een rechte rug kunnen we moeilijk of niet testen met Openpose aangezien de rug daar altijd een rechte is.



Figuur 2.8: Op het laagste punt van de squat moet de knie zich recht boven de tenen bevinden. (afbeelding van [1])



Figuur 2.9: Op het laagste punt van de squat moeten de heup en knie zich op dezelfde hoogte bevinden. (afbeelding van [1])

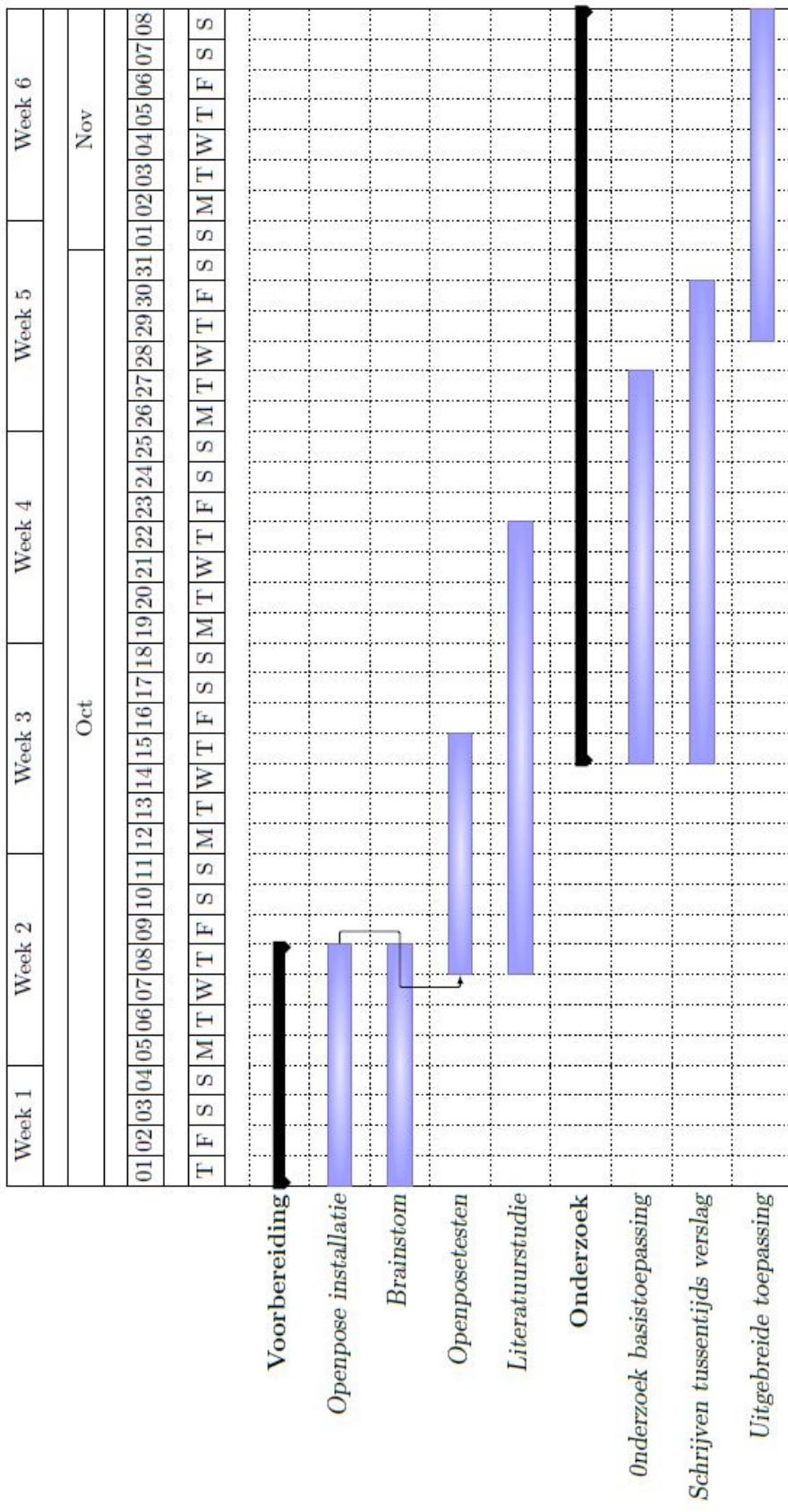
Hoofdstuk 3

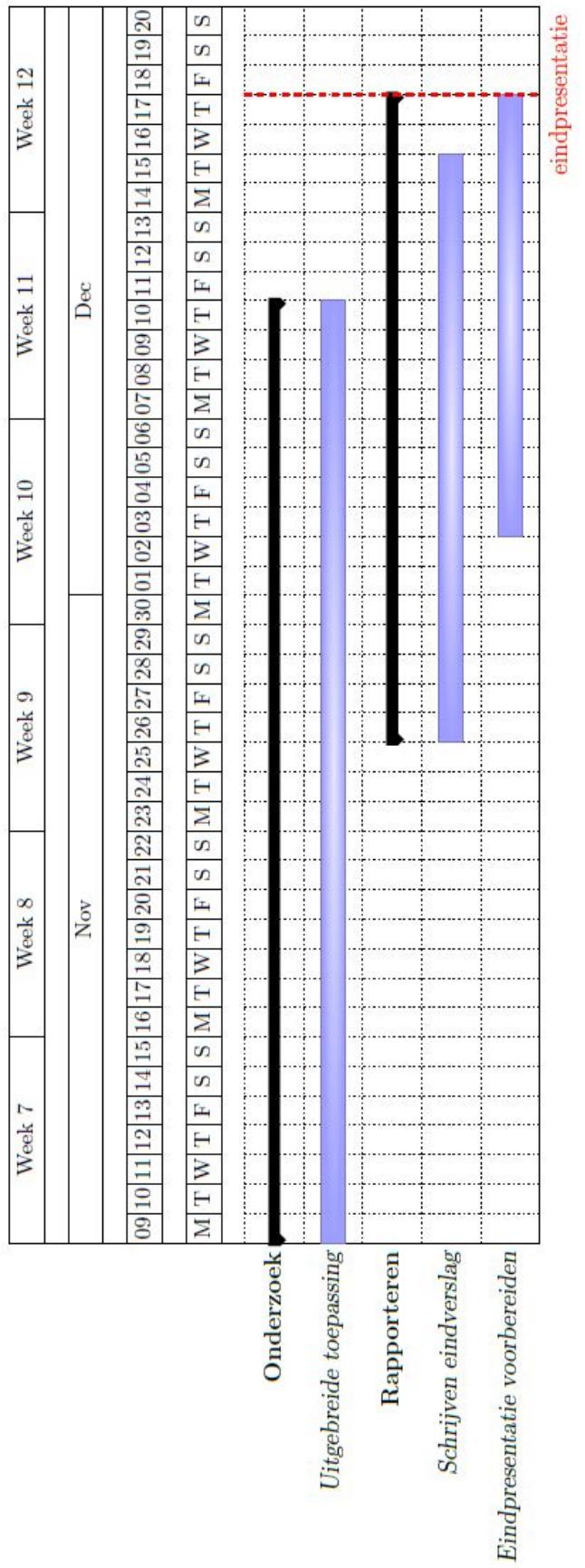
Vakintegratie

Het is natuurlijk belangrijk om dit project een plaats te geven binnen onze opleiding ingenieurswetenschappen. We bekijken welke vakken uit de eerste 3 semesters van onze bachelor het meest gebruikt worden tijdens dit project. De belangrijkste link is met het vak beginselen van programmeren. In dit vak leerden we werken met Python en verworven we inzicht in het programmeren. Wat heel handig is, want tijdens ons project maken we veelvuldig gebruik van Python. Elk zelfgemaakte programma is geschreven in Python want Openpose heeft een uitstekende Python API. Verder maakt wiskunde ook een groot deel uit van ons project, want het berekenen van hoeken of afstanden kan niet zonder de wiskunde. We gebruiken hierbij kennis uit verschillende wiskundige vakken zoals, Analyse & calculus en Lineaire algebra. Als laatste kunnen we Statistiek ook nog linken aan dit project aangezien Openpose enkel een schatting geeft van de lichaamspositie. Het werkt met *heatmaps* en kiest per knooppunt dan het coördinaat met de hoogste kans. We gebruiken zelf geen statistische technieken, maar het helpt ons wel bij het begrijpen van Openpose.

Hoofdstuk 4

Planning





Bibliografie

- [1] URL https://stronglifts.com/squat/#Squat_Technique.
- [2] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

Taakverdeling & verantwoordelijkheden

October 29, 2020

Taakverdeling	Isaac	Mathieu	Stan
Theoretische achtergrond	X	X	X
Toepassingen	X	X	X
Vakintegratie		X	X
Vergaderverslagen	X		
Planning		X	
Programmeren	X	X	X

Verantwoordelijkheden	Isaac	Mathieu	Stan
Teamleider			X
Notulist	X		
Verslag		X	
Programmeren	X		

KU Leuven Kulak
Wetenschap & Technologie
Etienne Sabbelaan 53, 8500 Kortrijk
Tel. +32 56 24 60 20
isaac.venus@student.kuleuven.be

