Ristomatti Leivo

Koneoppimiskehyksen käyttäminen askelanalyysisovelluksessa

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikka

Insinöörityö

24.4.2023

Tiivistelmä

Tekijä:	Ristomatti Leivo
Otsikko:	Koneoppimiskehyksen käyttäminen askelanalyysisovelluksessa
Sivumäärä:	25 sivua
Aika:	24.4.2023
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Tieto- ja viestintätekniikka
Ammatillinen pääaine:	Ohjelmistotuotanto
Ohjaajat:	Tutkijaopettaja Aarne Klemetti

Kävelyn aikana tapahtuvista nivelten liikkeistä ja niistä mitatuista nivelkulmien muutoksista on mahdollista tehdä monenlaisia terveyteen ja lihaskuntoon liittyviä havaintoja. Markkinoilta on saatavilla erilaisia ammattilaistason laitteistoja ja ohjelmistoja tähän tarkoitukseen. Nämä laitteistot ja ohjelmistot ovat kuitenkin hintavia ja usein vaativat erillisiä laitteisto- ja ohjelmistoasennuksia.

Tämän insinöörityön tavoitteena oli kehittää ReactJS-kirjaston avulla internetselaimissa toimiva verkkosovellus, joka mittaa MediaPipe-ohjelmistokehyksen Pose-komponentin avulla nivelkulmat ennalta nauhoitetuista videotiedostoista. Onnistuneen videoanalysoinnin jälkeen sovellus piirtää tulokset kuuteen kuvaajaan.

Yhtenä tavoitteista oli saada sovellus helposti saataville. Videotiedoston suosituksena on 60 kuvaa per sekunti ruutunopeus, ja että se on kuvattu sivulta päin koehenkilöön nähden. Suositusten mukaisten videoiden kuvaaminen on täten mahdollista nykyaikaisilla älypuhelimilla. Valmis sovellus laitettiin konttiin Dockerin ja OpenShiftin avulla helposti kaikkien saataville. Moderneilla laitteistoilla ja verkkoselaimilla sovelluksen käyttäminen ei siten vaadi minkäänlaista asentamista, vaan pelkkä sovelluksen verkkosivun avaaminen riittää.

Sovelluksen avulla haluttiin myös saada mahdollisimman toistettavia ja luotettavia mittaustuloksia, ja tunnistaa askeleen eri vaiheet mahdollisimman tarkasti.

Lopputuloksena saatiin toimiva sovellus, joka antaa johdonmukaisia mittaustuloksia ja myös tunnistaa askelvaiheet luotettavasti.

Avainsanat: MediaPipe Pose, React.js,Gait analysis, Docker, OpenShift

---

Abstract

Author:	Ristomatti Leivo
Title:	Utilizing ML-Framework in Gait Analysis Application
Number of Pages:	25 pages
Date:	2 May 2023
Degree:	Bachelor of Engineering
Degree Programme:	Information Technology
Professional Major:	Software Engineering
Supervisor:	Aarne Klemetti, Researching Lecturer

Human gait analysis offers the means for obtaining valuable information about a person´s health and physical attributes. While commercial solutions for this purpose are available, they are often expensive and require both software and hardware installation.

The primary goal of the study was to create a web application, which utilizes ReactJS library and the Pose solution of the MediaPipe framework. The application analyzes a pre-recorded video file and obtains the joint coordinates of the test subject performing a walking exercise. After a successful measurement the results of six different joint angles are calculated and plotted into separate graphs.

One of the goals was to make the application highly available. The requirements of the videos for analyzation are such that they can be recorded by any modern smartphone. The application was deployed to the OpenShift environment and thus made available for virtually anyone. The only requirements are recent hardware, internet access and a compatible internet browser.

The result is a working application which produces repeatable results and successfully recognizes the key events in human gait cycle.

Keywords: MediaPipe Pose, React.js,Gait analysis, Docker, OpenShift

Sisällys

Lyhenteet

AI: *Artificial Intelligence* tietokoneohjelma, joka pystyy tekemään älykkäinä pidettäviä toimintoja.

CNN: *Convolutional Neural Network* koneoppimisessa käytetty neuroverkkotyyppi kuvien ja visuaalisen tiedon käsittelyyn.

RNN: *Recurrent Neural Network* koneoppimisessa käytetty neuroverkkotyyppi aikasarjojen ja liikkeen käsittelyyn.

IMU: *Inertial Measurement Unit* laite kiihtyvyyden, kulmanopeuden ja magneettikentän mittaamiseen.

MP4: MPEG-4 Part 14 säiliömuoto multimedialle.

OGG: säiliötiedostomuoto multimedialle. Kehittäjä Xiph.Org Foundation.

WebM: Googlen kehittämä säiliötiedostomuoto multimedialle.

HTML: *Hypertext Markup Language* standardoitu merkintäkieli hyperlinkkejä sisältävän tekstin kuvaamiseen.

PaaS: *Platform-as-a-service* sovellusalusta pilvipalveluna.

DOM: *Document Object Model* HTML-rakenteen kuvaaminen puumuodossa.

CUDA: *Compute Unified Device Architecture* Nvidian kehittämä alusta ja ohjelmointirajapinta grafiikkasuoritinlaskentaan.

1 Johdanto

Ihmisen kävelystä on mahdollista havaita ihmisen terveyteen ja lihaskuntoon liittyviä asioita, ja jopa joitain sairauksia (1). Teknologian kehittyminen on mahdollistanut syväoppimiseen perustuvien ratkaisujen toteuttamisen, joiden avulla on mahdollista tehdä näitä havaintoja laitteistoja käyttäen myös reaaliajassa. Kun suoritusta tarkastellaan koneellisesti, on mahdollista tallentaa tulokset myöhempää tarkastelua ja esimerkiksi vertailua varten. Laskentatehojen kasvamisen myötä nämä ratkaisut, jotka ovat aiemmin vaatineet kalliit erikoislaitteet ja -sovellukset, ovat viime vuosina tulleet kaikkien ulottuville.

Tässä insinöörityössä haluttiin kehittää verkkosovellus ja samalla selvittää, onko nykyaikaisilla laitteistoilla ja web-kehitysmenetelmillä mahdollista saada myös askelanalysointi kaikkien ulottuville. Valmis sovellus haluttiin myös saada verkkoon helposti loppukäyttäjien saataville.

Tässä raportissa luodaan ensin katsaus käytettyihin menetelmiin ja teknologioihin. Tämän jälkeen tarkastellaan sovelluksen toimintaa ja ominaisuuksia ja lopuksi käydään läpi saadut tulokset ja jatkokehitysideat.

2 Liikeanalyysi

2.1 Koneoppiminen

Koneoppimisella tarkoitetaan sellaisten algoritmien ja mallien kehittämistä, joiden avulla tietokone voi suorittaa tehtäviä ilman, että sille täytyy muodostaa jokaiselle tehtävälle omat erilliset säännöt. Koneoppimista pidetään yleisesti tekoälyn (engl. AI) alana. Koneoppimisisalgoritmit ja -mallit kykenevät usein sopeutumaan ja oppimaan saatavilla olevan tiedon perusteella.

Koneoppimisen yleisimmät tyypit ovat

- ohjattu oppiminen
- ohjaamaton oppiminen
- vahvistusoppiminen.

Ohjatussa oppimisessa mallille annetaan syötteet ja oikeat vastaukset, jolloin malli oppii löytämään säännönmukaisuudet haluttujen lopputulosten saavuttamiseksi. Jos mallille annetaan pelkät syötteet ilman oikeita vastauksia, puhutaan ohjaamattomasta oppimisesta. Vahvistusoppimisen tapauksessa mallia opetetaan antamalla sille joko positiivista tai negatiivista vahvistusta sen perusteella, onko lopputulos halutun kaltainen. (2.)

2.2 Syväoppiminen

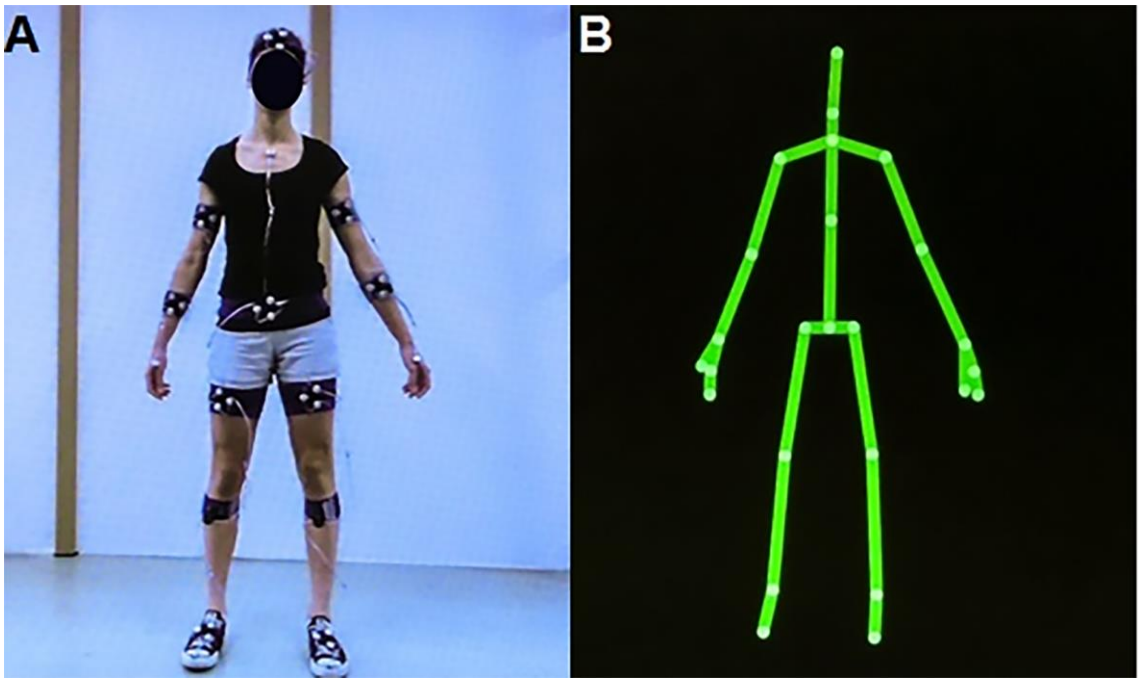
Syväoppiminen (engl. Deep Learning) on yksi koneoppimisen menetelmä. Syväoppiminen perustuu neuroverkkoihin eli matemaattisiin malleihin, jotka jäljittelevät ihmisen aivojen toimintaa. Syväoppiminen mahdollistaa monimutkaisten mallien toteutuksen, jotka pystyvät paremmin tunnistamaan esimerkiksi kohteiden liikkeitä ja niiden vaikutuksia eri muuttujiin.

Kehonliikkeiden analysoinnissa käytetään yleisesti konvoluutio-neuroverkkoja (CNN) tai rekurrentteja neuroverkkoja (RNN). Näistä CNN on erityisen soveltuva kuvien ja videoin käsittelyyn. RNN-neuroverkon vahvuus on tiedon käsittely sen tapahtuma-aika huomioon ottaen, jolloin se soveltuu hyvin liikkeen analysointiin.

Kehonliikettä analysoivia malleja on mahdollista vahvistaa käyttämällä erillisen mittauslaitteen keräämää tietoa. Yksi tällainen laite on inertiamittausyksikkö (engl. IMU), joka voi mitata muun muassa kiihtyvyyttä ja kulmanopeutta. Mittauslaitteella kerätty data syötetään koneoppimismallille koulutusvaiheessa. (2.)

2.3 Asennon arviointi

Asennon arviointi (engl. Pose Estimation) on kehittynyt vuosien saatossa merkkipohjaisista (engl. Marker) ratkaisuista (ks. kuva 1) merkittömiin (engl. Markerless) syväoppimista hyödyntäviin ratkaisuihin (3).



Kuva 1. Marker-pohjainen Pose Estimation -järjestelmä (4).

Merkitön liikeanalyysi mahdollistaa täysin uudenlaiset sovellukset verrattuna merkkipohjaisiin ratkaisuihin. Merkitön analyysi soveltuu käytettäväksi esimerkiksi urheilussa, terveydenhuollossa ja peleissä. Tietokoneiden ja mobiililaitteiden laskentatehojen kasvun myötä nämä sovellukset ovat tulleet kaikkien ulottuville (3).

Kuva 2. Lightweight OpenPose -esimerkki (5).

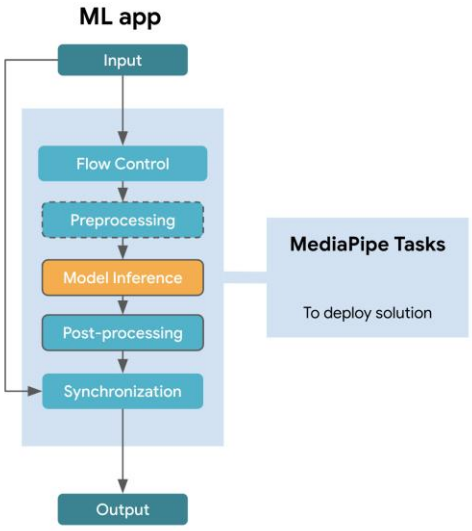
Syväoppimiseen perustuvat mallit kehittyvät edelleen ja saatavilla on monia eri ratkaisuja, kuten OpenPose, PoseDetection, DensePose ja AlphaPose (6).

2.4 Google MediaPipe

Googlen kehittämä MediaPipe on avoimen lähdekoodin ohjelmistokehys, joka mahdollistaa reaaliaikaisen merkittömän liikeanalyysin moniin erilaisiin käyttökohteisiin, kuten koko kehon asennon arviointiin, käden seurantaan ja kasvojen tunnistamiseen. MediaPipen tarjoama tunnistus pohjautuu myös Googlen kehittämään BlazePose-malliin.

MediaPipen toimintakaaviosta voidaan nähdä koneoppimista käyttävän sovelluksen toimintaperiaate. Malli saa syötteen, tekee vuonohjauksen ja syötteen esikäsittelyn. Tämän jälkeen malli tarjoaa rajapinnan analyysin soveltamiseen halutulla tavalla. Kun syöte on onnistuneesti analysoitu, suoritetaan jälkikäsittely, syötteiden synkronointi ja annetaan valmis tulos

käytettäväksi sovellukseen.



Kuva 3. MediaPipen toimintakaavio (7).

MediaPipen tarjoamat eri ratkaisut voidaan jaotella seuraaviin kategorioihin

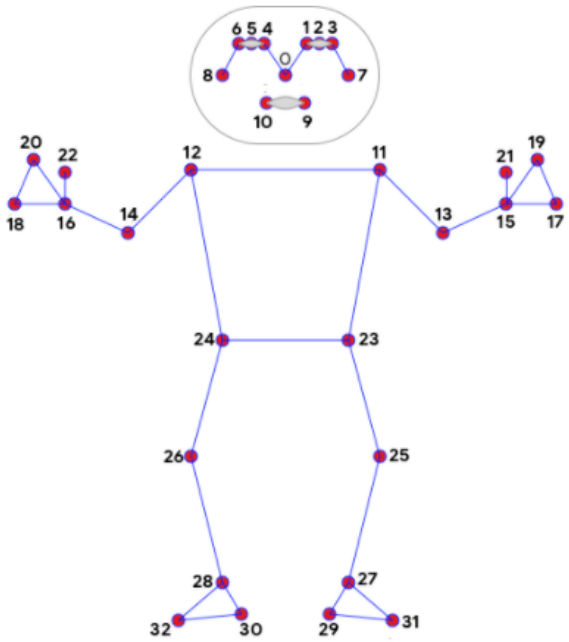
- esineen tunnistus
- kuvien luokittelu
- käden tunnistus
- kehon tunnistus
- kuvien upottaminen
- kuvien segmentointi
- tekstin luokittelu
- tekstin upottaminen
- äänen luokittelu.

MediaPipe on saatavilla useille eri alustoille ja kielille, kuten Android, iOS, Python ja JavaScript ja C++. (7.)

2.5 MediaPipe Pose

Yksi MediaPipen tarjoamista ratkaisuista on Pose, joka on reaaliaikainen koko kehon tunnistamiseen kykenevä ratkaisu. Malli tunnistaa ihmisen kehosta

yhteensä 33 eri pistettä ja tarjoaa niistä koordinaatit joko 2D- tai 3D-muodossa. Koordinaatit ovat suhteutettuna joko piirretyn kuvan pikseleihin tai 3D-koordinaattien tapauksessa lantion keskipisteessä sijaitsevaan origoon (8).



0. nose
1. left_eye_inner
2. left_eye
3. left_eye_outer
4. right_eye_inner
5. right_eye
6. right_eye_outer
7. left_ear
8. right_ear
9. mouth_left
10. mouth_right
11. left_shoulder
12. right_shoulder
13. left_elbow
14. right_elbow
15. left_wrist
16. right_wrist
17. left_pinky
18. right_pinky
19. left_index
20. right_index
21. left_thumb
22. right_thumb
23. left_hip
24. right_hip
25. left_knee
26. right_knee
27. left_ankle
28. right_ankle
29. left_heel
30. right_heel
31. left_foot_index
32. right_foot_index

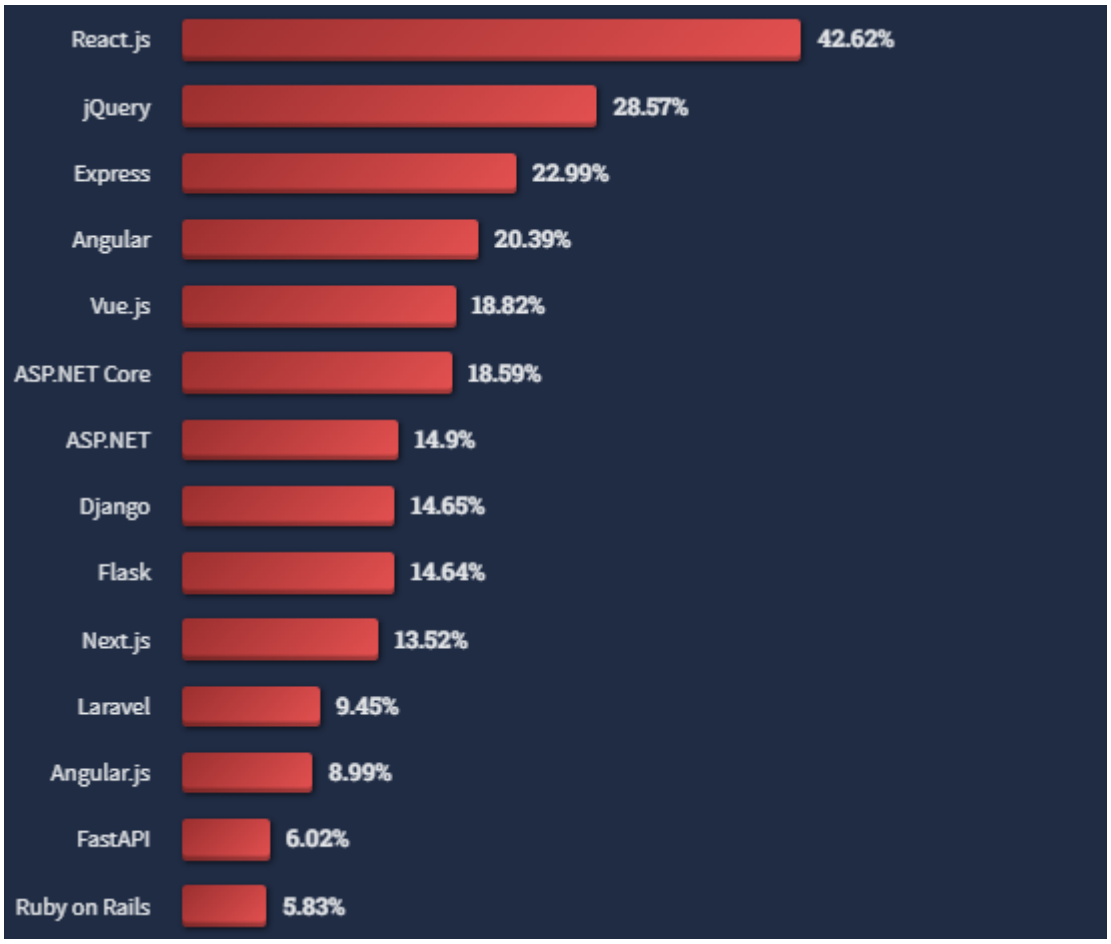
Kuva 4. MediaPipen koordinaatit (8).

MediaPipen Pose myös mahdollistaa käyttäjälleen monien tunnistukseen liittyvien parametrien manuaalisen määrittelemisen, joilla voidaan esimerkiksi painottaa tunnistuksen tarkkuutta tai nopeutta (1). Palaamme parametrien määrittelyyn myöhemmin raportissa.

3 ReactJS

ReactJS, yleiseltä nimeltään React, on Facebookin vuonna 2013 kehittämä avoimen lähdekoodin JavaScript-kirjasto frontendin kehitykseen. React kehitettiin ennen kaikkea helpottamaan dynaamisten verkkosivujen luomista. Reactin ytimessä on virtuaalinen DOM, jonka tehtävä on mahdollistaa muuttuvien sivujen ja näkymien nopea renderöinti loppukäyttäjälle, päivittämällä vain ne osat sivusta, joilla muutos tapahtuu (9).

React on ylivoimaisesti suosituin kirjasto frontendin kehittämiseen StackOverflow-sivuston vuonna 2022 suorittaman kyselyn mukaan (ks. kuva 5) (9).



Kuva 5. ReactJS:n suosio verrattuna muihin kirjastoihin 2022 (10).

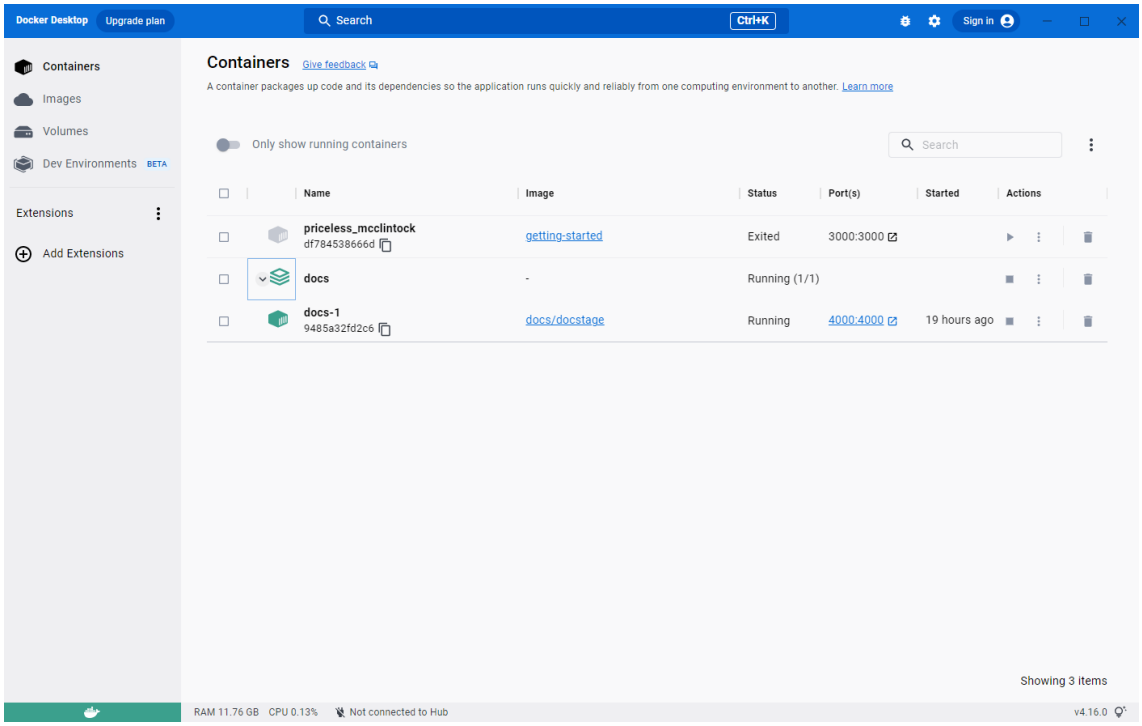
Reactin periaate on komponenttiperustainen kehittäminen. Sovellukset ja niiden tarjoamat näkymät koostuvat pienemmistä dynaamisista komponenteista, jotka liitetään yhteen ylemmissä (engl. Parent) komponenteissa. Toiminnallisuuksien ja näkymien pilkkominen omiin komponentteihinsa selkeyttää ja nopeuttaa sovellusten kehittämistä, testaamista ja ylläpitoa (11).

4 Konttiteknologia

Konttiteknologia mahdollistaa sovellusten eristämisen toisistaan ja isäntäjärjestelmästä ja toimii vaihtoehtona kokonaisten tietokoneiden virtualisoimiselle. Virtuaalikoneisiin verrattuna kontit ovat kevyempiä ja nopeampia käynnistää. Yksittäinen kontti sisältää itse sovelluksen ja kaikki sen vaatimat riippuvuudet, kuten kirjastot, ajurit ja määrittelytiedostot. Tämä mahdollistaa sovelluksen toiminnan takaamisen riippumatta järjestelmästä, jossa sitä käytetään (12).

4.1 Docker

Docker on suosittu avoimen lähdekoodin konttiteknologia. Docker pohjautuu Linuxin LXC-konttiteknologiaan ja sisältää lisäominaisuuksia, kuten Docker Enginen ja Docker Hubin. Docker Engine vastaa konttien suorittamisesta isäntäkoneella. Docker Hub on puolestaan palvelu, jonka avulla kontitettuja sovelluksia voi helposti tallentaa ja jakaa pilvessä (12).



Kuva 6. Docker Desktop -hallintasovelluksen säiliö näkymä (13)

Docker tarjoaa suosituimmille käyttöjärjestelmille Docker Desktop -käyttöliittymän säiliöiden ja ajoympäristöjen hallintaan. (ks. Kuva 6).

4.2 OpenShift

OpenShift on Red Hatin kehittämä avoimen lähdekoodin konttipohjainen sovellusalusta palveluna (PaaS) -ratkaisu sovellusten kehittämistä, jakelua ja hallintaa varten. OpenShift perustuu Kubernetes-klusterienhallintaan ja käyttää Docker-konttiteknologiaa. OpenShift mahdollistaa muun muassa sovellusten automaattisen skaalauksen ja päivityksen (engl. Rolling Update). (15.)

Red Hat tarjoaa OpenShiftistä myös riisuttua versiota nimeltään OKD, josta puuttuu suurin osa yritystoimintaan suunnatuista toiminnallisuuksista ja ominaisuuksista (15).

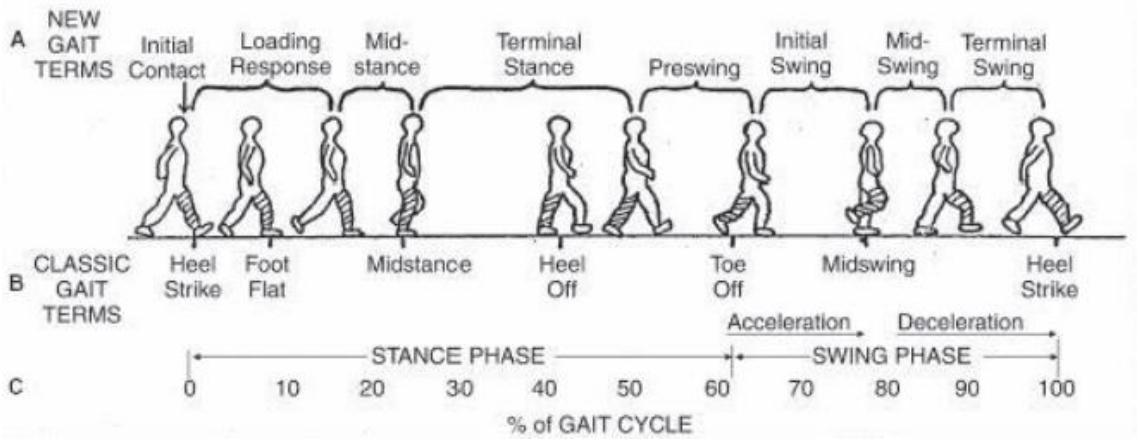
5 Askelanalysointisovellus

Sovellus toteutettiin ReactJS-kirjastoa käyttäen. Sovellus on todettu toimivaksi Google Chrome-, Mozilla Firefox- ja Microsoft Edge-verkkoselaimilla.

Sovellukselle annetaan videotiedosto kävelysuorituksesta, joka on kuvattu sivusta koehenkilöön nähden. Sovellus analysoi videon ja tallentaa suorituksen aikana Posella saadut koordinaattitiedot taulukkoon.

Videon päättymisen jälkeen sovellus laskee tallennettujen taulukoiden arvoista kolme eri nivelkulmaa per jalka. Saadut tulokset uudelleennäytteistetään ja piirretään sitten kuuteen kuvaajaan.

Laskettavat nivelkulmat ovat lonkan, polven ja nilkan koukistus- ja ojennuskulmat. Sovellus tunnistaa automaattisesti askeleiden alku-, loppu- ja heilahdusvaiheet. Heilahdusvaiheella tarkoittaa hetkeä, jolloin tukijalka nousee alustasta ilmaan (ks. kuva 7) (16).



Kuva 7. Askelsyklin vaiheet (16).

Jalan tukivaihe alkaa, kun kantapää osuu alustaan ja päättyy, kun varpaat irtoavat alustasta. Tätä seuraa heilahdusvaihe, jonka päättyminen aloittaa uuden askeleen.

5.1 Posen käyttöönotto ReactJS-sovelluksessa

Pose tuodaan React-sovellukseen import-käskyllä. Tämän lisäksi tarvitaan videolähde, joka on tämän sovelluksen tapauksessa videotiedosto, mutta se voisi yhtä hyvin olla web-kameran kuva. Jos halutaan piirtää Posen tunnistamat pisteet videokuvan päälle käyttäjän nähtäville, tarvitaan myös JavaScriptin Canvas-elementti.

Alustuksen yhteydessä Poselle annetaan videolähde parametrina ja jokaisen onnistuneen tunnistuksen jälkeen Pose kutsuu funktiota onResults(). Sovelluksen tunnistusdatan tallentaminen tapahtuu onResults()-funktion sisällä. Tallennettua dataa jälkikäsitellään (engl. Postprocess) sovelluksessa runsaasti tallennuksen jälkeen.

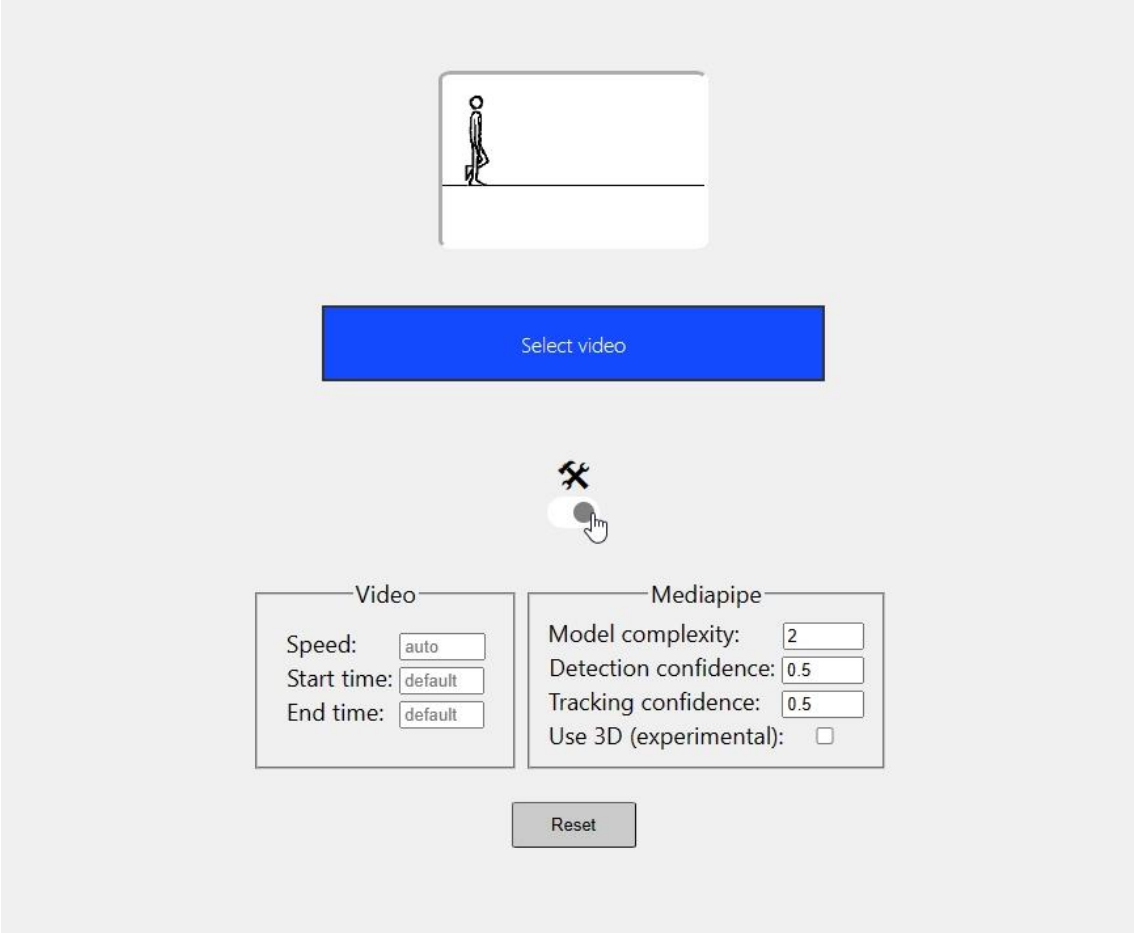
```
import { Pose } from "@mediapipe/pose"

const pose = new Pose({
      locateFile: (file) => {
        return `https://cdn.jsdelivr.net/npm/@mediapipe/pose/${file}`
      },
    })

    pose.setOptions({
      modelComplexity: modelComplexity,
      smoothLandmarks: true,
      enableSegmentation: false,
      smoothSegmentation: false,
      minDetectionConfidence: detectionConfidence,
      minTrackingConfidence: trackingConfidence
    })
    setPose(pose)
    pose.onResults(onResults)

     …
    function onResults(results) {
```

Esimerkkikoodi 1. Posen käyttöönotto JavaScriptillä.

Käyttöönotto on hyvin samankaltainen kuin muillakin ohjelmointikielillä, esimerkiksi Pythonilla (17).

5.2 Aloitusnäkymä

Sovelluksen aloitusnäkymässä on tiedostonvalintapainike ja mahdollisuus määritellä erilaisia analysointiin liittyviä asetuksia (ks. kuva 8).



Kuva 8. Sovelluksen aloitusnäkymä.

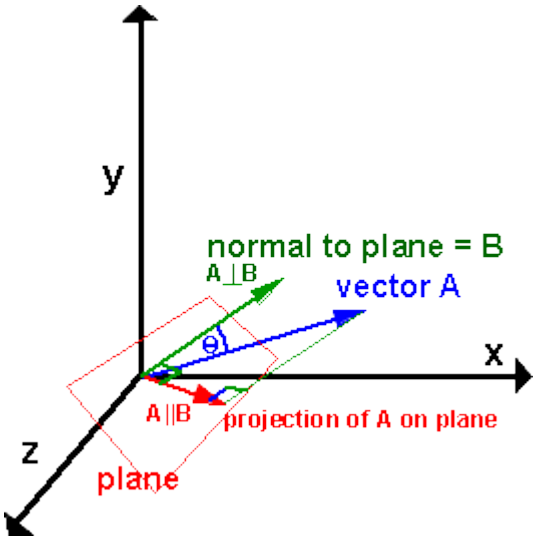
Koska sovellus analysoi ainoastaan videotiedostoja eikä esimerkiksi reaaliaikaista web-kamerakuvaa, on analysointi mahdollista ajaa nopeutetusti koneen laskentatehojen niin salliessa. Sovellukseen on tästä syystä kehitetty automaattinen kalibrointitoiminnallisuus, joka testaa, kuinka monta näytettä Pose-malli kykenee tunnistamaan tietyssä ajassa, ja säätää videon toistonopeuden saadun tuloksen mukaan. Nopeus on määriteltävissä asetusvalikosta myös manuaalisesti, kuten on myös analysoinnin haluttu aloitus- ja lopetushetki videossa.

5.3 Asetukset

Muut asetukset liittyvät Pose-mallin ominaisuuksiin ja niitä voidaan halutessa määritellä käsin, mikäli oletusasetuksilla ei saada halutun kaltaisia tuloksia.

Model Complexity -asetus määrittelee käytettävän mallin monimutkaisuuden. Korkeampi arvo tyypillisesti nostaa sekä tarkkuutta että tunnistukseen kuluvaa aikaa. Detection confidence -asetuksella voidaan määrittää raja-arvo, jonka mukaan tunnistus tulkitaan onnistuneeksi ja tuloksia aletaan muodostamaan. Tracking confidence -parametri liittyy liikkeentunnistukseen ja on vaikutukseltaan samankaltainen Detection confidencen kanssa sillä erolla, että korkeampi arvo myös tyypillisesti nostaa tunnistukseen kuluvaa aikaa. (7.)

Use 3D -valinta ottaa käyttöön sovelluksessa kokeelliselle asteelle jääneen toiminnallisuuden, joka mahdollistaa viistosta kuvatun videon analysoimisen suorittamalla 2D-tasoheijastuslaskennan saaduille 3D-koordinaateille. Tasoheijastuksen periaate on havainnollistettu kuvassa 8.



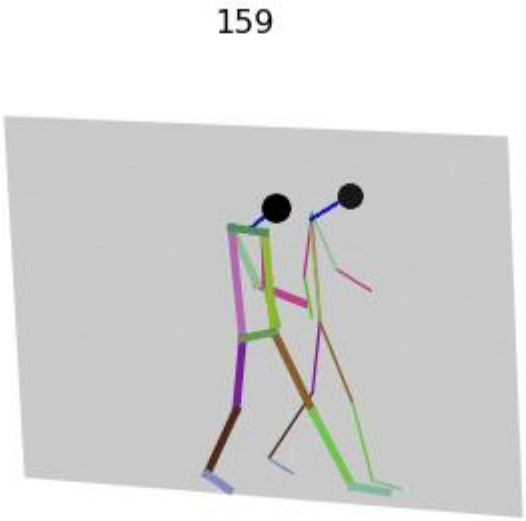
Kuva 9. Esimerkki tasoheijastuksen toimintaperiaatteesta (18).

Sovellukseen toteutettiin koodi, joka suorittaa 2D-tasoheijastuksen 3D-koordinaateista tallennetulle datalle. Tason määrittävä normaalivektori muodostettiin lantion vasemman ja oikean pisteen välille.

```
//normaali vektori
  n = [leftHipX - rightHipX, leftHipY - rightHipY, leftHipZ -
rightHipZ]

  //apuvektori
  v = [1, 0, 0]

  e1 = mathjs.cross(v, n)
  e1 = mathjs.divide(e1, mathjs.norm(e1))

  e2 = mathjs.cross(e1, n)
  e2 = mathjs.divide(e2, mathjs.norm(e2))

  A = [e1, e2]

  vLeftHip1 = [(rightShoulderX + leftShoulderX) / 2 - (rightHipX +
leftHipX) / 2, (rightShoulderY + leftShoulderY) / 2 - (rightHipY +
leftHipY) / 2, (rightShoulderZ + leftShoulderZ) / 2 - (rightHipZ +
leftHipZ) / 2]
  vLeftHip2 = [leftKneeX - leftHipX, leftKneeY - leftHipY, leftKneeZ -
leftHipZ]

  vLeftHip1_proj = mathjs.multiply(A, vLeftHip1)
  vLeftHip2_proj = mathjs.multiply(A, vLeftHip2)

const getHipAngle = (side, direction) => {

  if (side && direction) { // left & forward
    return (Math.atan2(vLeftHip1_proj[0], vLeftHip1_proj[1]) -
Math.atan2(vLeftHip2_proj[0], vLeftHip2_proj[1])) * 180 / Math.PI -
180
}
```

Esimerkkikoodi 2. Näyte vasemman lonkkakulman tasoheijastuslaskennan toteuttavasta koodista

Koodi saatiin toimimaan, mutta tasoheijastuksesta plotatussa kuvasta (ks. kuva 10) on nähtävissä, ettei Pose pystynyt luotettavasti arvioimaan esimerkiksi pään todellista sijaintia.



Kuva 10. Alkuperäiset ja tasoheijastuksen avulla saadut koordinaatit plotattuna Pythonin matplotlib-kirjastoa käyttäen. Yläpuolella näkyvä luku 159 on tulostaulukon senhetkinen indeksi.

Pose epäonnistuu arvioimaan luotettavasti koordinaattien sijainteja syvyyssuunnassa (Z-axis). Tämän seurauksena tasoheijastuksella saatujen koordinaattien avulla lasketut nivelkulmat jäävät käytännössä liian loiviksi.

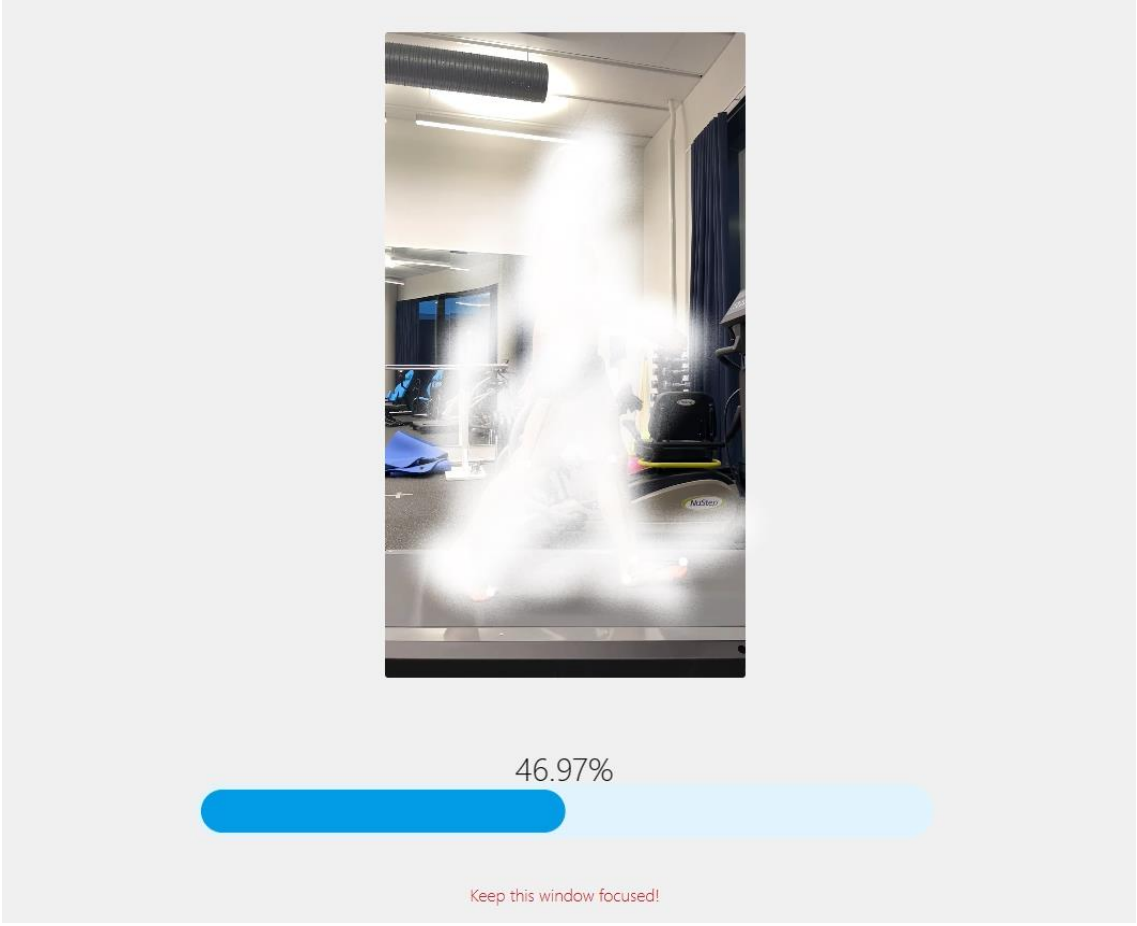
Vaikka tasoheijastusta käyttämällä saadut tulokset eivät osoittautuneet kovinkaan käyttökelpoisiksi, pystyttiin niiden avulla saamaan parempi käsitys Pose-mallin 3D-koordinaattien luotettavuudesta.

5.4 Analysointivaihe

Sovellus tukee tiedostomuotoja MP4, WebM ja OGG. Rajoitus perustuu HTML5 <video> -elementin rajoituksiin (19). Kun käyttäjä on valinnut haluamansa tiedoston, aloittaa sovellus automaattisesti alkukalibroinnin ja siirtyy sen jälkeen analysoimaan videota.

Analysointivaiheen aikana sovellus tallentaa taulukkoon koordinaatit tähdäten noin 80–100 näytteeseen per sekunti. Lopullinen näytemäärä riippuu saatavilla olevasta laskentatehosta. Vähintään 60 kuvaa per sekunti sisältävät videot ovat suositeltuja ja sovellus kykenee hyödyntämään videoita aina 100 kuvaa per sekunti taajuuteen saakka.

Sovelluksella on mahdollista analysoida myös alle 60 kuvaa per sekunti sisältäviä videoita, mutta tämä vaikuttaa negatiivisesti askeltunnistuksen tarkkuuteen.



Kuva 11. Sovelluksen analysointivaihe näkymä. Kuvaa on käsitelty koehenkilön yksityisyyden suojaamiseksi.

Käyttäjälle ilmoitetaan analysoinnin aikana punaisella varoitustekstillä, että internetselaimen ikkuna on pidettävä valittuna. Vaatimus liittyy Windows-käyttöjärjestelmän tapaan allokoida laskentaresursseja internetselaimelle. Jos

käyttäjä avaa kesken analysoinnin toisen sovelluksen, ei internetselain saa tarpeeksi prosessoriaikaa, jolloin analysointi ei onnistu ja käyttäjälle näytetään virheilmoitus.

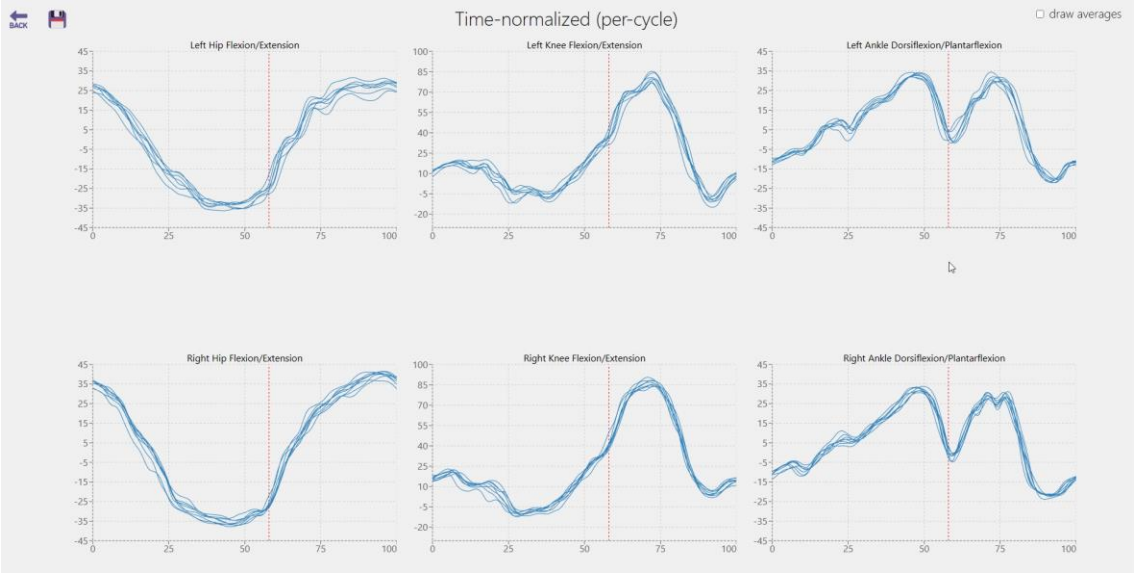
5.5 Tulokset-näkymä

Kun videon analysointi saadaan suoritettua onnistuneesti loppuun, suorittaa ohjelma varsinaiset laskelmat. Tallennetuista koordinaattitiedoista päätellään ensin algoritmien avulla koehenkilön kävelysuunta. Kävelysuunta saadaan selville yksinkertaisesti vertaamalla jalan varpaiden ja kantapään koordinaattien sijaintia keskenään.

Tämän jälkeen tunnistetaan askeleiden alku-, loppu- ja heilahdusvaihekohdat. Askelsyklin vaiheiden tunnistamiseen on olemassa monia erilaisia tapoja (20). Sovelluksessa päädyttiin ratkaisuun, jossa uuden askeleen alkukohdaksi valitaan hetki, jolloin kantapää on mahdollisimman etäällä keskivartalosta ja keskivartalon etupuolella.

Seuraavaksi sovellus laskee nivelkulmat ja saadut tulokset uudelleennäytteistetään (engl. Resample) niin, että jokaiselta askeleelta on olemassa sama määrä näytteitä. Uudelleennäytteistetyt taulukot viedään Recharts-komponentille, joka piirtää ne käyttäjän nähtäville graafeihin.

Kaikki edellä mainittu datan käsittely tapahtuu omassa tiedostossaan DataPostprocess.js.



Kuva 12. Tulokset-näkymä.

Tuloksia voi tarkastella kuvassa 11 näkyvään tapaan piirtämällä saadut kulmat per askel. Kuten graafeista voi nähdä, askeltunnistusalgoritmi tunnistaa askeleiden eri vaiheet yhtenäisesti. Punainen pystykatkoviiva havainnollistaa tukivaiheen päättymishetkeä. Sivun oikeasta yläkulmasta voidaan valita ”draw averages”, jolloin tuloksista näytetään painotetut keskiarvot yksittäisten askelten tulosten sijasta.

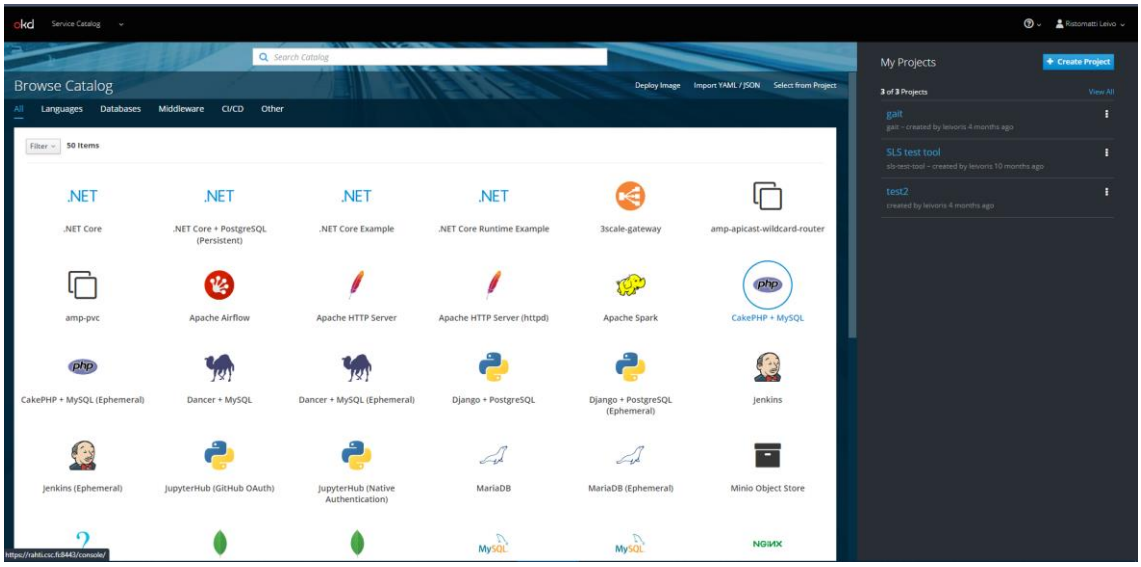
Sovellus toimii täysin selaimessa, eikä sisällä minkäänlaista backendiä tai tietokantaa. Tuloksista on kuitenkin halutessa mahdollista tallentaa ruutukaappaus painamalla vasemmassa yläkulmassa sijaitsevaa levykkeen kuvaa. Kuvatiedosto nimetään automaattisesti tallennushetken päivämäärän ja kellonajan mukaan.

5.6 Sovelluksen vieminen konttiin

Valmis sovellus haluttiin saada helposti kaikkien saataville. Yksi ratkaisu tähän oli viedä sovellus OpenShiftin avulla virtualisoituun verkkopalvelinympäristöön. Tässä tapauksessa käytössä oli CSC:n Rahti-palvelu, joka on OpenShift OKD:tä käyttävä konttiorkesterointiympäristö.

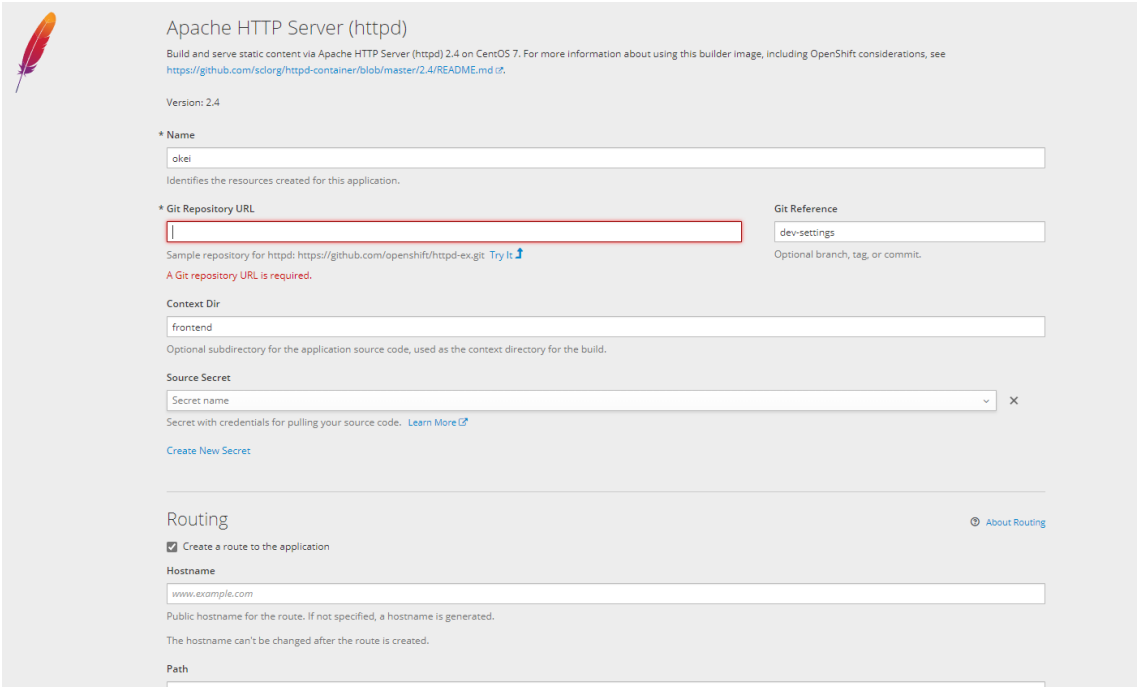
5.7 OpenShiftin määrittely

Sovellus ei sisällä erillistä backendiä tai tietokantaa, joten sen määrittely OpenShiftillä oli kohtuullisen suoraviivaista.



Kuva 13. Rahti-palvelun OpenShift-kataloginäkymä.

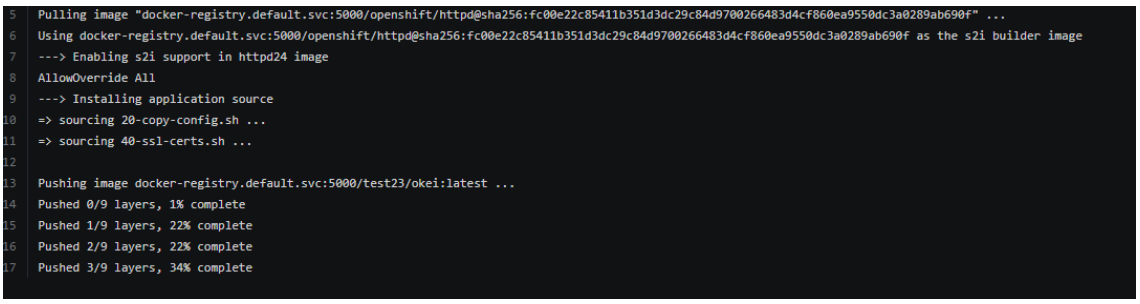
Näkymän oikeasta yläkulmasta valittiin ”Create project” ja annettiin projektille nimi ja lyhyt kuvaus. Tämän jälkeen luotu projekti valittiin listasta ja avautuvasta näkymästä painettiin ”Browse Catalog” -painiketta. Listasta valittiin ”Apache HTTP Server (httpd)” ja painettiin next-painiketta.



Kuva 14. OpenShiftin määrittely. Apache HTTP Serverin asetukset.

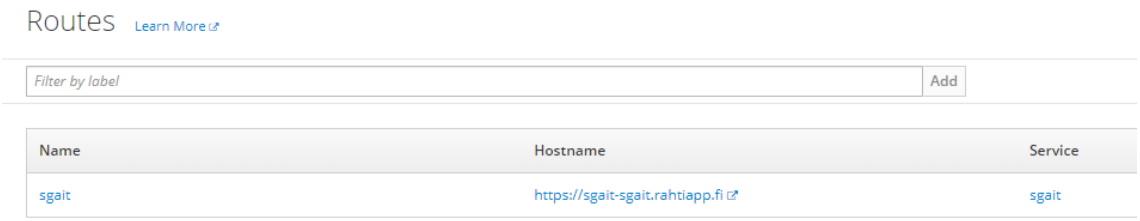
Sovellukselle annettiin haluttu nimi ja ”Git Repository” -kohtaan sovelluksen GitHub-osoite. Git Reference -kenttään syötettiin haluttu tuotantohaara (engl. Branch). Context Dir -kenttään laitettiin sovelluksen tapauksessa ”frontend/build”, koska käännetty tuotantoversio sovelluksen frontendistä löytyy tuosta kansiosta. Valittu metodi siis vaatii, että sovellus on viety GitHubiin julkiseen säilöön (engl. Repository).

Tämän jälkeen painettiin Create -painiketta, jolloin OpenShift alkaa alustamaan konttiympäristöä (ks. kuva 15).



Kuva 15. Luodun kontin alustus OpenShift-ympäristössä.

Alustuksen jälkeen OpenShift loi automaattisesti reitityksen (engl. Route) sovellukselle, ja sovellus on nyt verkossa kaikkien saatavilla (ks. kuva 16).



Kuva 16. OpenShiftin reititysnäkymä, jossa on nähtävissä sovelluksen julkinen osoite.

OpenShiftiin vieminen voi olla parhaimmillaan erittäin suoraviivaista, jos valmiit mallinteet (engl. Template) tarjoavat sovelluksen kanssa yhteensopivan ratkaisun. Aina näin ei ole, vaan käyttäjä törmää yhteensopivuusongelmiin käytössä olevien ohjelmistojen versioiden vuoksi, jolloin ainoaksi vaihtoehdoksi voi jäädä konttien lisääminen ympäristöön käsin. Kun lähtee rakentamaan kokonaisuutta pala palalta frontend, backend ja tietokanta kerrallaan ja määrittelee näiden näkyvyydet niin, että ne pystyvät kommunikoimaan tarvittaessa keskenään ja näkyvät ulkomaailmaan, avautuu käyttäjälle näkymä OpenShiftin konttiorkesteroinnin pohjalla olevaan monimutkaiseen järjestelmään.

6 Tulokset

Sovellus saatiin toimimaan hyvin ja luotettavasti. Saman kävelysuorituksen analysoiminen useita kertoja muodosti jokaisella ajokerralla keskenään hyvin samankaltaiset tulokset. Silmämääräisesti tulokset näyttävät oikean suuntaisilta, kun niitä vertaa ammattilaislaitteistoilla saatuihin mittaustuloksiin samankaltaisista suorituksista. Tästä voidaan varovaisesti päätellä, että sovellus onnistuu tehtävässään hyvin.

Joissain olosuhteissa, kun koehenkilön vaatetus ja videolla näkyvän taustan väri ovat tarpeeksi samankaltaiset, on Pose-mallilla vaikeuksia tunnistaa kehoa luotettavasti, ja analysointi hajoaa. Näitä tapauksia varten ohjelmaan on toteutettu ominaisuus, joka tarkistaa tulokset sen varalta, ovatko ne järkeviä vai ei. Tarpeen tullen käyttäjälle ilmoitetaan, että mittaus ei onnistunut ja palaa alkunäkymään.

Eräs isompi ongelma, joka sovellusta kehittäessä tuli vastaan, oli Pose-mallin tarjoamien koordinaattipisteiden sijainti. Malli ei tarjoa pistettä, joka sijaitsisi koehenkilön varpaiden kohdalla, vaan lähin piste sijaitsee henkilön päkiän seudulla. Tämä tekee tarkasta nilkkakulman tuloksen saamisesta lähes mahdotonta. Tämä johtuu tarkemmin siitä, miten jalkapohja taipuu askeltaessa ja kuinka nilkkakulma lasketaan.

Sovelluksen saatavuuden suhteen onnistuttiin hyvin. Jos tietokoneesta löytyy erillinen näytönohjain, toimii sovellus mainiosti esimerkiksi kannettavalla tietokoneella. MediaPipe osaa hyödyntää näytönohjainten Cuda-ytimiä niiden ollessa saatavilla (21). Cuda-ytimien hyödyntäminen vaikuttaa Posen suorituskykyyn merkittävästi. Ilman erillistä näytönohjainta mallin suorituskyky jää alhaiseksi, mikä vaikuttaa negatiivisesti saatuihin tuloksiin. Jos videota joudutaan toistamaan erittäin hitaasti, on olemassa riski, että askeltunnistus hajoaa.

Sovellus saatiin myös vietyä onnistuneesti OpenShift-konttiympäristöön. Tässä tapauksessa käyttöönottotapa oli hyvin suoraviivainen ja hyvä esimerkki siitä, miten helppoa konttiympäristön käyttäminen parhaimmillaan voi olla.

7 Yhteenveto

Vaikka tulokset näyttävät lupaavilta ja sovellus vaikuttaa toimivan hyvin, jäi varsinainen vertailu ammattilaisjärjestelmään tekemättä. Vaikka Pose Estimation on nimensä mukaisesti aina vain arvio, eikä täysin tarkkoihin tuloksiin ole ainakaan toistaiseksi mahdollista päästä (22), olisi sovelluksen

tulosten vertaileminen esimerkiksi Markereita käyttävän ammattilaiskäyttöön tarkoitetun järjestelmän tuloksiin mielenkiintoista.

Jälkeenpäin tarkastellen ohjelman koodia olisi mahdollista refaktoroida huomattavasti selkeämpään suuntaan ja hyödyntää tehokkaammin ReactJS:n tarjoamia lukemattomia ominaisuuksia ja kirjastoja. Refaktoroinnin tarve johtuu ennen kaikkea siitä, että ReactJS:n käyttö opeteltiin vasta sovellusta kehittäessä.

Jos mietitään sovelluksen jatkokehitystä, yksi idea olisi tehdä sovellukseen tietokanta, joka tallentaa tulokset myöhempää tarkastelua varten. Tämä mahdollistaisi sovellukseen myös sellaisen ominaisuuden implementoinnin, jossa henkilön tulosten kehitystä voisi seurata vertaamalla eri hetkillä kerättyjä tuloksia keskenään.

Lähteet

1 Shahid ym. 2012. A study on human gait analysis. Association for Computing Machinery.

2 Goodfellow ym. 2016. Deep Learning. Verkkoaineisto. <https://www.deeplearningbook.org/>. Luettu 30.4.2023

3 Tekin ym. 2016 Structured prediction of 3D human pose.

4 Geerse ym. 2015. Kinematic Validation of a Multi-Kinect v2 Instrumented 10-Meter Walkway for Quantitative Gait Assessments. PLoS ONE 10(10). Verkkoaineisto. <https://doi.org/10.1371/journal.pone.0139913>. Luettu 18.4.2023.

5 Deep Vision Processing. 2022. Verkkoaineisto. GitHub. <https://github.com/cansik/deep-vision-processing>. Päivitetty 8.12.2022. Luettu 18.4.2023.

6 Walia, Mrinal. 2022. A Comprehensive Guide on Human Pose Estimation. Verkkoaineisto. Analytics Vidhya. <https://www.analyticsvidhya.com/blog/2022/01/a-comprehensive-guide-on-human-pose-estimation/>. Päivitetty 10.2.2022. Luettu 19.4.2023.

7 MediaPipe Pose. 2022. Verkkoaineisto. Google GitHub. <https://github.com/google/mediapipe/blob/master/docs/solutions/pose.md > Päivitetty 3.4.2023. Luettu 18.4.2023.

8 Cochard, David 2021 BlazePose: A 3D Pose Estimation Model. Verkkoaineisto. Medium.com <https://medium.com/axinc-ai/blazepose-a-3d-pose-estimation-model-d8689d06b7c4>. Luettu 19.4.2023.

9 Describing the UI. 2023. Verkkoaineisto. React.dev. <https://react.dev/learn/describing-the-ui>. Luettu 22.4.2023.

10 2022 Developer Survey. 2022. Verkkoaineisto. Stackoverflow.com. Verkkoaineisto. <https://survey.stackoverflow.co/2022/#technology-most-popular-technologies>. Luettu 20.4.2023.

11 Quick Start. 2023. Verkkoaineisto. React.dev. <https://react.dev/learn>. Luettu 20.4.2023.

12 Merkel, Dick. 2014. Docker: lightweight Linux containers for consistent development and deployment. Linux J. 2014, 239, Article.

13 Overview. 2023. Verkkoaineisto. docs.docker.com. <https://docs.docker.com/desktop/use-desktop/> Luettu 20.4.2023

14 Pahl, Claus & Jamshidi, Pooyan. 2016. Microservices: A Systematic Mapping Study. 137-146. 10.5220/0005785501370146.

15 Red Hat OpenShift vs. OKD. 2022. Verkkoaineisto. www.redhat.com. <https://www.redhat.com/en/topics/containers/red-hat-openshift-okd>. Luettu 19.4.2023.

16 Korotkin, Dmitry & Artem, Kuznetocv. 2013. Inertial Measuremenbt System for Human Gait Analysis. BodyNets 2013.

17 Anwar, Iqra. 2021. MediaPipe Python Tutorial [How to Install + Real-Time Hand Tracking Example]. Verkkoaineisto. Luettu 22.4.2023

18 Baker, Martin. 2023. Maths - Projection of lines on planes. Verkkoaineisto. EuclideanSpace. <https://www.euclideanspace.com/maths/geometry/elements/plane/lineOnPlane/>. Luettu 21.4.

19 HTML <video> Tag. 2023. Verkkoaineisto. w3schools.com. <https://www.w3schools.com/tags/tag_video.asp>. Luettu 22.4.2023.

20 Vu ym. 2020. A Review of Gait Phase Detection Algorithms for Lower Limb Prostheses. Sensors 20, no. 14: 3972. Verkkoaineisto. <https://doi.org/10.3390/s20143972> Luettu 19.4.2023.

21 GPU Support. 2023. Verkkoaineisto. developers.google.com. <https://developers.google.com/mediapipe/framework/getting_started/gpu_support>. Luettu 23.4.2023.

22 Needham ym. 2021. The accuracy of several pose estimation methods for 3D joint centre localisation. Sci Rep 11, 20673. Verkkoaineisto. <https://doi.org/10.1038/s41598-021-00212-x> Luettu 23.4.2023.