



Antti Nyholm

MediaPipe-liitännäisen implemen- tointi ja käyttö pelimoottorissa

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikka

Insinöörityö

19.5.2022

Tiivistelmä

Tekijä: Antti Nyholm
Otsikko: MediaPipe-liitännäisen implementointi ja käyttö pelimootorissa
Sivumäärä: 43 sivua
Aika: 19.5.2022

Tutkinto: Insinööri (AMK)
Tutkinto-ohjelma: Tieto- ja viestintätekniikka
Ammatillinen pääaine: Pelisovellukset
Ohjaajat: Lehtori Antti Laiho
Asiakkaan edustaja Aarne Klemetti

Unity-pelimootorin liitännäinen MediaPipe tarjoaa erilaisia koneoppimista hyödyntäviä ratkaisuja raajojen, kasvojen ym. seuraamiseen. Insinöörityössä MediaPipe-liitännäinen asennettiin Unity-pelimootoriin ja hyödynnettiin MediaPipen Pose-ratkaisua tehdyn sovelluksen pääominaisuuden luomiseen.

Insinöörityön tavoitteena oli luoda pienin toimiva tuote sovelluksesta, joka kerää tietoa polven kulmasta ja säilyttää siitä saatavaa tietoa eri muodoissa. Polven kulmasta saatava tieto on tärkeää esim. loukkaantumisriskien arvioinnissa.

Insinöörityössä tehty sovellus laskee polven kulman, kun kuvattava henkilö suorittaa yhden jalan kyykyn, ja tallentaa kyykkäykset videolle ja siitä saatavat polven kulmien tiedot tekstitiedostoon ja tietokantaan, jotta niihin voidaan päästä käsiksi myöhemmin.

Insinöörityössä syntynyt sovellus onnistuu tehtävissään suurimmilta osin hyvin. Kulman arvioinnissa on pientä vaihtelua, silloin kuin seisotaan suorassa, mutta tulokset silloin, kun yhden jalan kyykkyä suoritetaan, olivat järkeviä. Toisaalta sovellusta ei testattu kuin yhdellä henkilöllä. Pienten ongelmien jälkeen datan tallennus saatiin toimimaan kaikissa halutuissa muodoissa.

Insinöörityössä syntynyt sovellus on myös hyvä esimerkki siitä, kuinka Unity-pelimootoria voidaan hyödyntää muuhunkin kuin perinteisiin pelisovelluksiin, sekä oiva esimerkki Unityllä luodusta sovelluksesta, joka käyttää hyödykseen MediaPipe-liitännäistä.

Avainsanat: MediaPipe, koneoppiminen, Unity, polven kulma

Abstract

Author:	Antti Nyholm
Title:	The implementation and use of MediaPipe in a game engine.
Number of Pages:	43 pages
Date:	19 May 2022
Degree:	Bachelor of Engineering
Degree Programme:	Information and communications technology
Professional Major:	Game Applications
Supervisors:	Antti Laiho, Senior Lecturer Aarne Klemetti, Customer representative

MediaPipe offers a variety of machine learning solutions for tracking limbs, faces, etc. The purpose of this bachelor's thesis was to create a Minimum Viable Product of an application that collect information about the angle of the knee and store the information obtained from it in various formats. Information from the knees angle is important, for example, in assessing the risk of injury.

The application made during this bachelor's thesis calculates the angle of the knee when the person being in front of the camera performs a single-leg squat and saves the squats on video, and the resulting knee angle data into a text file and database for later access.

First, we'll go through a bit of the basics of machine learning and MediaPipe, as well as what different features and solutions MediaPipe offers to developers.

In this bachelor's thesis we will also go through the different ways in which the MediaPipe Unity plug-in can be installed, as well as a bit about how the MediaPipe Pose solution can be used in Unity to calculate the angle of the knee and how to access the coordinates of Pose's landmarks.

The application born from this bachelor's thesis is also a good example of how the Unity game engine can be used for non-traditional gaming applications, as well as a great example of an application created with Unity that takes advantage of the MediaPipe plugin.

Keywords: MediaPipe, Machine Learning, Unity, Angle of Knee

Sisällys

Lyhenteet

1	Johdanto	1
2	Koneoppiminen ja MediaPipe-liitännäinen	2
2.1	Koneoppimisen historia	2
2.2	Koneoppimisen periaatteet	3
2.3	MediaPipe-liitännäinen	10
2.4	MediaPipe-Unity-liitännäisen ominaisuudet	12
2.5	MediaPipea vastaavia ratkaisuja	21
3	Polven kuntoutus	21
3.1	Polven valgus- ja varusikulma	21
3.2	Polvivammojen syntymekanismit	22
3.3	Biomekaniikan perusteet	22
4	Polven kulman laskeminen Unityn MediaPipe-liitännäisellä	23
4.1	Sovelluksen tavoitteet	23
4.2	Sovelluksen tekemiseen käytetyt työkalut	23
4.3	MediaPipe-liitännäisen asennus Unityyn	24
4.4	MediaPipe Posen kordinaattien paikantaminen ja polven kulman laskeminen	25
4.5	Datan tallennus eri muodoissa	28
5	Projektin tulokset	37
5.1	Projektin haastavimmat osuudet	37
5.2	Vahvuudet ja heikkoudet	37
5.3	Parannusehdotuksia	39
6	Yhteenveto	39
	Lähteet	41

Lyhenteet

- 2D: Two-dimensional eli kaksiulotteinen. Sisältää kaksi ulottuvuutta eli pituuden ja leveyden.
- 3D: Three-dimensional eli kolmiulotteinen. Sisältää kolme ulottuvuutta eli pituuden, leveyden ja syvyyden.
- 8K: Kuvatarkkuus, jossa on noin 8 000 kuvapistettä vaakasuunnassa. $7\,680 \times 4\,320$ kuvapisteen 8K on korkein kuvatarkkuus, joka on määritetty Rec. 2020-ultrateräväpiirtotelevisiostandardissa.
- API: Application programming interface eli ohjelmointirajapinta. Määritelmä, jonka mukaan eri ohjelmat voivat tehdä pyyntöjä ja vaihtaa tietoja keskenään. Toimii rajana komponenttien tai moduulien välillä ohjelmoitavassa järjestelmässä.
- RGB: RGB-värimallin nimi juontuu sen päävärien englanninkielisistä nimistä: red, green ja blue. Se on väritila, jossa eri värejä muodostetaan sekoittamalla keskenään punaisen, vihreän ja sinisen väristä valoa.

1 Johdanto

Insinööritö käsittelee polven kulman laskentaa Unity-pelimootorilla hyödyntäen MediaPipe-liitännäistä ja polven kulmasta saatavan tiedon tallentamista. Polven kulmaa voidaan hyödyntää esimerkiksi polven loukkaantumisriskin arvioimiseen. Insinööritön tavoitteena on saada aikaan pienin toimiva tuote (engl. Minimum Viable Product) sovelluksesta, joka suorittaa mainitut toiminnot. Insinööritö tehtiin osana Metropolia Ammattikorkeakoulun virtuaalikuntoutusprojektia.

Insinööritön raportti on jaettu neljään eri osaan. Ensimmäinen osio käy läpi yleistietoa koneoppimisesta, kuten sen historiaa ja sen perusperiaatteita sekä MediaPipen perusteita ja sitä, millaisia erilaisia ratkaisuja MediaPipe tarjoaa.

Toisessa osiossa käsitellään polven valgus- ja varuskulmia ja niiden merkitystä, polvivammojen syntymekanismeja sekä hieman biomekaniikan perusteita ja polven biomekaniikkaa.

Kolmannessa osiossa käydään tarkemmin läpi polven kulman laskentasovelluksen tavoitteita ja työkaluja niiden saavuttamiseen sekä MediaPipen erilaisia asennustapoja ja sitä, kuinka MediaPipea hyödynnetään sovelluksessa ja kuinka ja missä muodoissa dataa tallennetaan.

Viimeisessä osiossa käydään läpi, kuinka hyvin sovellus onnistuu tai epäonnistuu sille asetetuissa tavoitteissa sekä hieman sitä, minkälaisia ongelmia projektin aikana ilmeni. Lopuksi käydään vielä läpi, miten sovellusta voisi parantaa ja kehittää eteenpäin tulevaisuudessa.

2 Koneoppiminen ja MediaPipe-liitännäinen

2.1 Koneoppimisen historia

Koneoppimisella on pitkä historia, joka alkaa jo 1950-luvulta. Alan Turing kysyi vuonna 1950 julkaistussa "Computing Machinery and Intelligence" -tutkimuksessa, osaavatko koneet ajatella (engl. Can machines think). Julkaisun osassa "Imitation Game" Turing kuvailee kolme osallistujaa testissä, jonka tavoitteena on vakuuttaa, että tekoäly pystyy imitoimaan ihmistä vakuuttavasti. Testin ensimmäinen osallistuja on ns. tuomari, jonka rooli on kirjoittaa pääteohjelmaan ja "jutella" muiden osallistujien kanssa. Testin muut osallistajat ovat toinen ihminen ja tietokone, joka yrittää esittää ihmistä. Molemmat, toinen ihminen ja tietokone, vastaavat tuomarille, joka valitsee, tuleeko vastaus tietokoneelta vai ihmiseltä, ja jos tuomari ei pysty johdonmukaisesti erottamaan ihmisen vastauksia tietokoneen vastauksista tietokone voittaa (1). Kyseinen testi elää edelleen kuvassa 1 nähtävissä olevan Loebner-palkinnon muodossa, jossa kilpailijat pyrkivät toteuttamaan tekoälyllä toimivan chatbotin, joka pystyy esittämään ihmistä.



Kuva 1. Loebner-palkinto (2).

Vuonna 1959 Arthur Samuel määritteli koneoppimisen tutkimusalaksi, jossa tietokoneelle annetaan kyky oppia ilman eksplisiittistä ohjelmointia. Työskennellessään IBM:llä Samuel kehitti yhden ensimmäisistä itseoppivista tietokoneohjelmista. Samuel keskittyi tammipeliin, alfa-beetakarsintaan ja minimax-strategiaan (1).

Teoksen Machine Learning (McGraw-Hill, 1997) kirjoittajan Tom M. Mitchellin määritelmää koneoppimisesta siteerataan usein:

A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P , if its performance at tasks in T , as measured by P , improves with the experience E .

Toisin sanoen, kun tietokone suorittaa useita tehtäviä, tietokoneen havaintojen tulisi johtaa suorituskyvyn kasvuun (1).

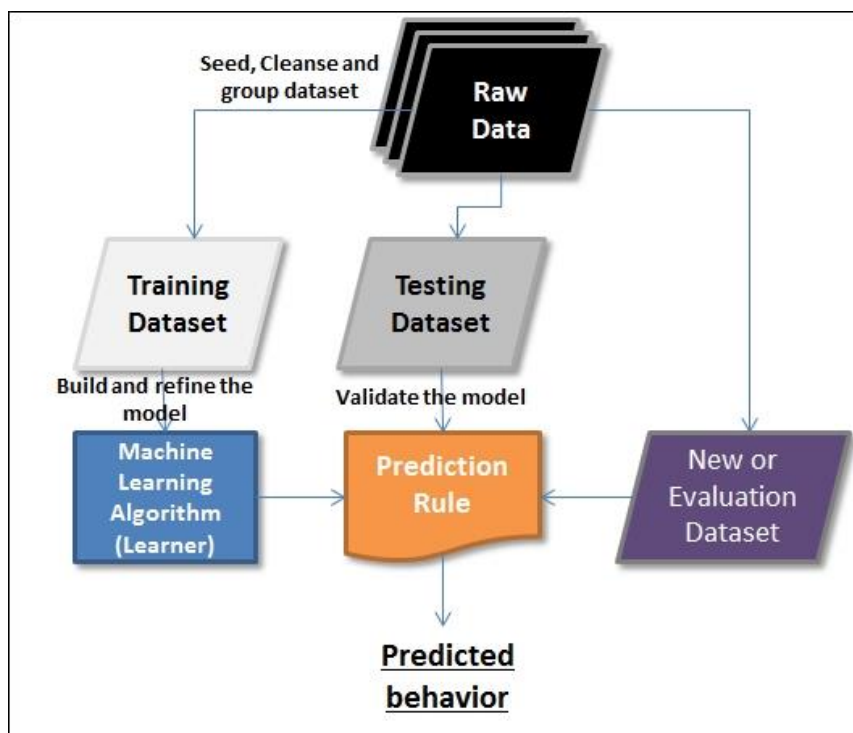
2.2 Koneoppimisen periaatteet

Yksinkertaisesti ilmaistuna koneoppimisella tarkoitetaan ohjelmaa, joka käyttää aiempia tietoja tai havaintoja ennakoimaan tai päättämään tulevia toimintoja. Toisin sanoen yksi älykkään järjestelmän pääpiirteistä on sen kyky oppia.

Seuraavaksi muutamia huomiota siitä, kuinka oppimisongelma voidaan määrittää. Ensimmäiseksi kannattaa määritellä se, mitä ohjelman tulisi oppia, ja opittavan asian tarpeellisuus. Seuraavaksi kannattaa määritellä tiedon vaatimukset ja tiedon lähteet. Lopuksi tulisi määritellä käytetäänkö tietoja kokonaisuudessaan vai vain osaa tiedoista.

Yleisesti koneoppimisen suorittamiseen tarvitaan kahdentyyppisiä tietojoukkoja. Ensimmäinen tietojoukko yleensä luodaan manuaalisesti. Siinä syöttötiedot ja odotetut lähtötiedot ovat valmiiksi saatavilla ja valmisteltuina. Toisessa tietojoukossa käyttäjällä on syöttötiedot ja kiinnostus odotettavissa olevien lähtötietojen ennustamiseen. Tyypillisesti koneoppimisen suorittamisessa on kolme eri vaihetta. Ne on havainnollistettu kuvassa 2. Ensimmäinen vaihe on

harjoitteluvaihe, jossa harjoitustietoja käytetään mallin harjoittamiseen yhdistämällä annettu syöte odotettuun lähtöön. Tämän vaiheen tuloksena saadaan aikaan itse oppimismalli. Toinen vaihe on validointi- ja testausvaihe, jossa mitataan, kuinka hyvä opetettu oppimismalli on, ja arvioidaan mallin ominaisuuksia, kuten virhemittauksia, muistamista, tarkkuutta ym. Tämä vaihe käyttää validointitietojoukkoa, jonka tuloksena saadaan kehittynyt oppimismalli. Vaihe kolme on sovellusvaihe, jossa malliin sovelletaan reaali maailman tietoja, joista tulokset on johdettava (3).



Kuva 2. Koneoppimisen suorittamisen kolme eri vaihetta (3).

Data on koneoppimisen pääasiallinen oppimisen lähde. Data voi olla missä tahansa muodossa, ja sitä voi vastaanottaa millä tahansa tiheydellä. Datat voivat olla minkäkokoisia tahansa. Koneoppimisen kontekstissa suurten tietojoukkojen käsittelyyn on olemassa joitain uusia tekniikoita, jotka ovat kehittyneet ja joita kokeillaan. On myös enemmän Big Datan näkökulmia, kuten rinnakkaiskäsittely, hajautettu tallennus ja suoritus (3).

Koneoppimisen kontekstissa data voi olla joko merkitsemätöntä dataa (engl. unlabeled data) tai merkittyä dataa (engl. labeled data). Merkitsemätön data on yleensä tietojen raakamuoto. Tämän muodon tiedoilla ei yleensä ole liitteenä olevaa selitystä merkitykselle. tällaista dataa voivat olla mm. suoratoistettavat videot, ääni, valokuvat ja twiitit. Merkitsemättömästä datasta tulee merkittyä dataa, kun siihen liitetään merkitys, kuten "tunniste" (engl. tag) tai "etiketti" (engl. label), joka on tarpeellinen ja pakollinen merkityksen tulkitsemiseksi ja määrittelymiseksi. Esimerkiksi valokuvassa nämä tunnisteet voivat olla sen sisältämiä tietoja, kuten eläin, puu, jne. Oppimismalleja voidaan soveltaa sekä merkittyyn dataan että merkityksettömään dataan. Käyttämällä merkittyjen ja merkitsemättömien tietojoukkojen yhdistelmää voidaan johtaa tarkempia malleja. Yleisesti ohjattu oppiminen ottaa käyttöön merkittyä dataa ja ohjaamaton oppiminen merkitsemätöntä dataa. Puoli-ohjatut oppimis- ja syväoppimistekniikat soveltavat merkittyjen ja merkitsemättömien tietojen yhdistelmää monin eri tavoin tarkkojen mallien luomiseksi.

Koneoppimisalgoritmi on rakennettu ratkaisemaan ongelmia, joita kutsutaan tehtäviksi (engl. tasks). Tehtävässä on tärkeää mitata sen suorituskyky, joka tässä yhteydessä tarkoittaa laajuutta tai luottamusta, jolla ongelma ratkaistaan. Kun erilaisia algoritmeja ajetaan eri tietojoukoissa, saadaan aikaan erilaisia malleja (engl. models). On tärkeää, ettei näin luotuja malleja verrata toisiinsa vaan sen sijaan mitataan tulosten johdonmukaisuutta eri aineistojen ja eri mallien kanssa (3).

Kun koneoppimisongelma on ymmärretty selkeästi, voidaan keskittyä siihen, mitkä tiedot ja algoritmit ovat merkityksellisiä tai sovellettavia. Käytettäviä algoritmeja on useita, ja ne on ryhmitelty joko oppimisalikenttien (kuten valvottu, valvomaton, vahvistus, puolivalvottu tai syvä) tai ongelmaluokkien (kuten luokittelu, regressio, klusterointi tai optimointi) mukaan. Algoritmeja sitten sovelletaan iteratiivisesti eri tietosarjoihin, josta kaapataan uuden tiedon myötä kehittyneet tulokset.

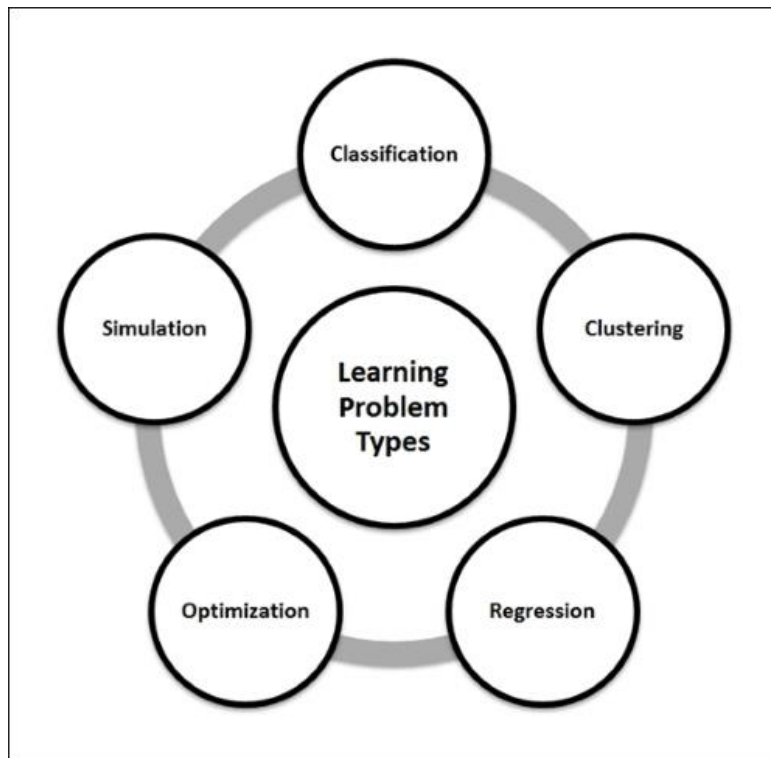
Mallit ovat keskeisiä kaikissa koneoppimisen toteutuksissa. Mallit kuvaavat järjestelmässä havaittua dataa, ja ne ovat tietojoukkoon käytettyjen algoritmien tulos. Yleisesti malleja sovelletaan uusiin tietokokonaisuuksiin, jotka auttavat malleja oppimaan uutta käyttäytymistä ja myös ennustamaan uutta käyttäytymistä. Koneoppimisalgoritmeja on laaja valikoima, ja niitä voidaan soveltaa halutun ongelman ratkaisemiseen.

Erittäin korkealla tasolla on seuraavasti luokiteltavia malleja: Loogiset mallit ovat luonteeltaan algoritmisia. Ne auttavat johtamaan sääntöjoukkoa ajamalla algoritmeja iteratiivisesti. Esimerkki loogisesta mallista on mm. päätöspuu. Geometriset mallit käyttävät käsitteitä, kuten viivoja, etäisyyksiä ja tasoja hyväkseen. Yleensä tällaiset mallit toimivat suurilla tietomäärillä. Yleensä lineaariset muunnokset auttavat vertailemaan erilaisia koneoppimismenetelmiä. Todennäköisyysmallit ovat tilastollisia malleja, ja ne käyttävät tilastollisia tekniikoita hyväkseen. Todennäköisyysmallit perustuvat strategiaan, joka määrittelee kahden muuttujan välisen suhteen ja tämä suhde voidaan johtaa varmasti, koska tämä edellyttää satunnaisten taustaprosessin käyttöä. Useissa tapauksissa kokonaistietojen osajoukkoa voidaan harkita tiedon prosessointiin (3).

Koneoppimisprojekteja toteutettaessa voi kohdata monenlaisia epäjohtonmuokaisuuksia, kuten alisovitettu malli (engl. Under-fitting), jossa malli ei ota riittävästi huomioon informaatiota mallintaakseen tarkasti todellista dataa. Vastaa- vasti ylisovitettu (engl. Over-fitting) malli on myös riski, koska malli saattaa kuvata kohinaa oikeiden suhteiden kuvaamisen sijaan. Epävakaa data (engl. Data instability) saattaa johtua manuaalisesta virheestä tai asiaankuuluvien tietojen virheellisestä tulkinnasta, mikä saattaa johtaa datan vääristymiseen, joka lopulta päättyy virheelliseen malliin. Siksi on olemassa vahva tarve prosessille korjata tai käsitellä inhimillisiä virheitä, jotka voivat johtaa virheellisen mallin rakentamiseen. Ennalta arvaamattomat tietomuodot (engl. Unpredictable data formats) johtavat ongelmiin silloin, kuin järjestelmään tuleva uusi data tulee muodossa, jota koneoppimisjärjestelmä ei tue. Tämän tapahtuessa on vaikea sanoa, toimiiko malli hyvin sen saatua uutta dataa, ellei ole tehtynä mekanisme tämän käsittelemiseksi (3).

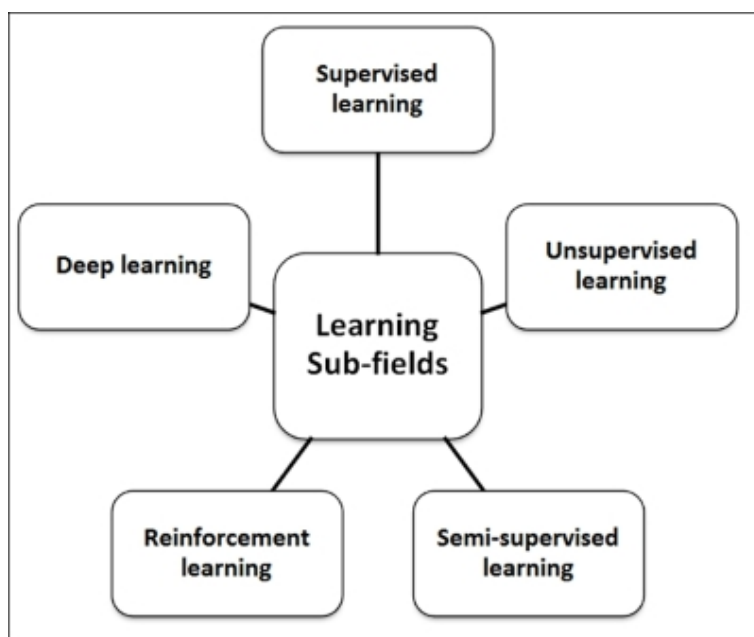
Käytännön esimerkkejä koneoppimisesta ovat esim. roskapostin tarkastus, luotokorttipetosten havaitseminen, numeroiden tunnistus, puheentunnistus, kasvojen tunnistus, tuotesuositus tai asiakassegmentointi, osakekauppa ja sentimenttianalyysi (3).

Kuvassa 3 voidaan nähdä erilaisia koneoppimisen ongelmien tyyppejä. Luokittelu on tapa tunnistaa tietojoukon ryhmittelytekniikka. Luokittelun avulla voidaan kohde- tai lähtöattribuutin arvosta riippuen luokitella tietojoukko kuuluvaksi johonkin luokkaan. Tämä tekniikka auttaa tunnistamaan datan käyttäytymismalleja. Klusteroinnissa data-analyytikolle annetaan vain joitain tietoja. Näiden tietojen avulla data-analyytikon odotetaan löytävän mielenkiintoisia kuvioita, jotka voivat auttaa johtamaan älykkyyttä. Ennustaminen, prognoosi tai regressio on sen tunnistamista, miten asiat tapahtuisivat tulevaisuudessa. Nämä tiedot ovat peräisin aiemmasta kokemuksesta tai tiedosta. Tapauksissa, jossa dataa ei ole tarpeeksi, tulevaisuus on määriteltävä regression avulla. Ennuste- ja prognoositulokset esitetään aina yhdessä epävarmuus- tai todennäköisyysasteen kanssa. Tätä ongelmatyyppin luokittelua kutsutaan myös säännön purkamiseksi. Näiden lisäksi myös simulaatio ja optimointi ovat oppimisongelmien tyyppejä (3).



Kuva 3. Erilaisia koneoppimisen ongelmien tyyppejä (3).

Induktio-niminen prosessi on yksi koneoppimisen keskeisistä käsitteistä. Monet oppimisen alakentät käyttävät induktioprosessia mallien rakentamiseen. Induktiivisessa oppimisessa käytetään yhden kokeen tuloksia seuraavan koesarjan suorittamiseen ja mallin kehittämiseen iteratiivisesti tietystä tiedosta. (3.) Kuvas-
sassa 4 on joitain alakenttiä, joilla koneoppimisalgoritmeja voidaan luokitella.



Kuva 4. Koneoppimisalgoritmien alakenttiä (3).

Ohjattu oppiminen (engl. supervised learning) on koneoppimisalgoritmien alakenttä, jossa toimitaan tunnetun odotuksen mukaan ja siitä, mitä määriteltävästä tiedosta on analysoitava. Ohjatussa toimimisessa syötettäviä tietojoukkoja kutsutaan ”merkityiksi” tietojoukoiksi. Tällaiset algoritmit keskittyvät määrittämään tulo- ja lähtöattribuuttien välisen suhteen käyttävät tätä suhdetta spekulatiivisesti luodakseen tulosten uusille syötetietopisteille. Ohjaamattomassa oppimisessä (engl. unsupervised learning) ei ole asetettu mitään spesifiä tavoitetta ratkaistavaksi. Esimerkki ohjaamattomasta oppimisesta on klusterointi, jossa ei lähdetä liikkeelle tietyistä tavoitteista. Puoliohjatulla oppimisella (engl. semi-supervised learning) käytetään sekä merkittyjä että merkitsemättömiä tietoja mallien oppimiseen. On tärkeää, että merkitsemättömille tiedoille on asianmukaiset oletukset, sillä sopimattomat oletukset voivat mitätöidä mallin. Puoliohjattu oppiminen saa motivaationsa ihmisen oppimistavasta. Vahvistusoppiminen (engl. reinforcement learning) on koneoppimisalgoritmi, jossa keskitytään maksimoimaan tuloksesta saatavat palkkiot. Vahvistusoppimisessa on tärkeintä, että malli on vastuussa sellaisten päätösten tekemisestä, joista mallia ajoittain palkitaan. Toisin kuin ohjatussa oppimisessa, tulokset eivät ole välittömiä ja saattavat edellyttää vaiheiden suorittamista, ennen kuin lopullinen tulos näkyy.

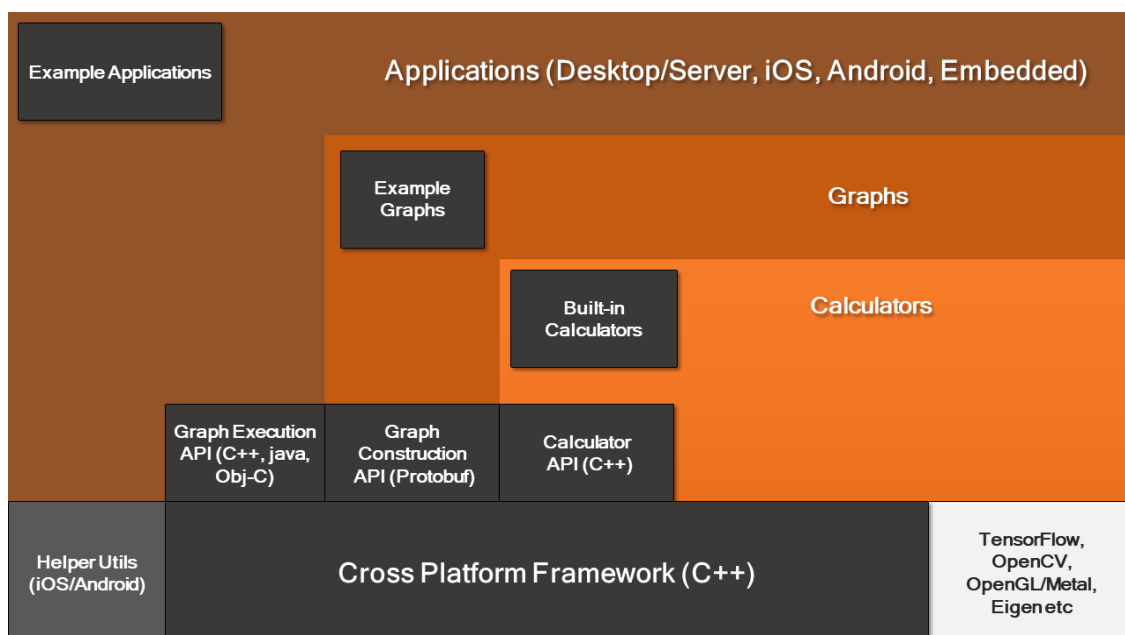
Ihanteellisesti tällainen algoritmi luo sarjan päätöksiä, jotka auttavat saavuttamaan korkeimman palkkion tai hyödyn (3).

Syväoppiminen (engl. deep learning) on koneoppimisen osa-alue, joka keskittyy yhdistämään koneoppimisen ja tekoälyn. Syväoppiminen käsittelee monimutkaisempien hermoverkkojen rakentamista ongelmien ratkaisemiseksi, jotka on luokiteltu puolivalvotun oppimisen piiriin ja toimivat tietojoukoilla, joissa on vähän merkittävää dataa. Esimerkkejä syväoppimisesta ovat mm. konvoluutioverkostot, rajoitettu Boltzmann-kone (engl. Restricted Boltzmann Machine (RBM)), syvän uskon verkosto (engl. Deep Belief Networks (DBN)) ja pinotut automaattiset kooderit (engl. Stacked Autoencoders) (3).

2.3 MediaPipe-liitännäinen

MediaPipe on Googlen julkaisema ohjelmistokehys koneoppimistiedonsiirtokanavien (engl. pipeline) rakentamiseen aikasarjadataan (engl. time-series data), kuten videon, äänen jne., käsittelyyn.

MediaPipen kirjastosisältää MediaPipen ohjelmistokehyksen ja ratkaisut. Kuvasta 5 voidaan nähdä MediaPipen kirjaston komponentit.



Kuva 5. MediaPipe-kirjaston komponentit (4).

MediaPipen ohjelmistokehys on kirjoitettu C++:lla, Javalla ja Obj-C:llä, ja se koostuu seuraavista API:eista:

- laskin API (C++)
- graafin rakennussovellusliittymä (Protobuf)
- graph Execution API (C++, Java, Obj-C).

MediaPipen tiedonsiirtokanavia kutsutaan kuvaajiksi. Kuvaaja koostuu reunojen yhdistämisestä solmuista, ja MediaPipe kuvaajan sisällä solmuja kutsutaan laskimiksi ja reunoja kutsutaan jonoiksi. Jonot kuljettavat sarjan paketteja, joissa on nousevat aikaleimat (4).

Laskimet ovat erityisiä laskentayksiköitä, jotka on kirjoitettu C++:lla. Näihin laskentayksiköihin on määritetty prosessoitavia tehtäviä. Datapaketit (videokehys tai äänisegmentti) tulevat sisään ja lähtevät laskimen porttien kautta. Laskimen alustuessa laskin ilmoittaa paketin hyötykuorman tyyppin, joka kulkee portin läpi. Kuvaajaa suorittaessa otetaan käyttöön viitekehyksen Open-, Process- ja

Close-menetelmiä laskimessa. Open-menetelmä käynnistää laskimen, minkä jälkeen Process-menetelmää suoritetaan toistuvasti, kunnes paketti saapuu. Paketin saapumisen jälkeen Close-menetelmä sulkee prosessin, kun koko kuvaaja on ajettu (4).

MediaPipen laskintyytit voidaan ryhmitellä neljään eri luokkaan:

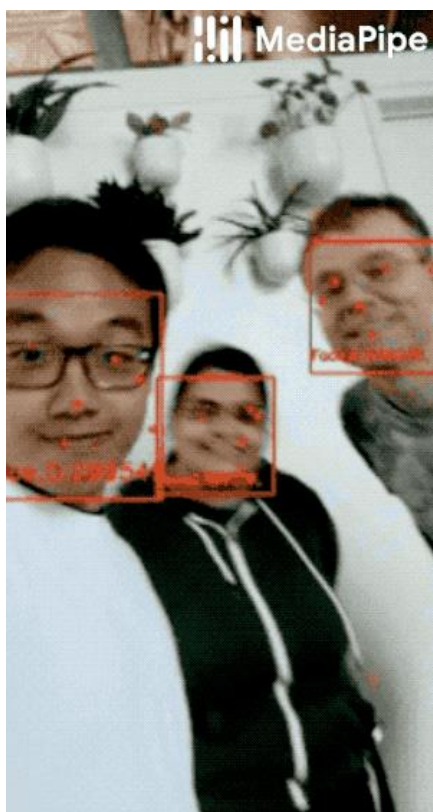
1. Esikäsitteilylaskurit (engl. Pre-processing calculators) ovat kuvan ja median käsitteilylaskimia.
2. Pääteilmälaskimet (engl. Inference calculators) mahdollistavat natiivi liittämissen Tensorflow- ja Tensorflow Lite -sovelluksiin koneoppimisjohtopäätöksiä varten.
3. Jälkikäsitteilylaskurit (engl. Post-processing calculators) suorittavat koneoppimisen jälkikäsitteilytehtäviä, kuten havaitsemista, segmentointia ja luokittelua.
4. Hyödyllisyyslaskurit (engl. Utility calculators) suorittavat viimeisiä tehtäviä, kuten kuvan huomautuksia (engl. image annotation).

MediaPipen laskurin sovellusliittymillä voi myös kirjoittaa mukautettuja laskimia (4).

2.4 MediaPipe-Unity-liitännäisen ominaisuudet

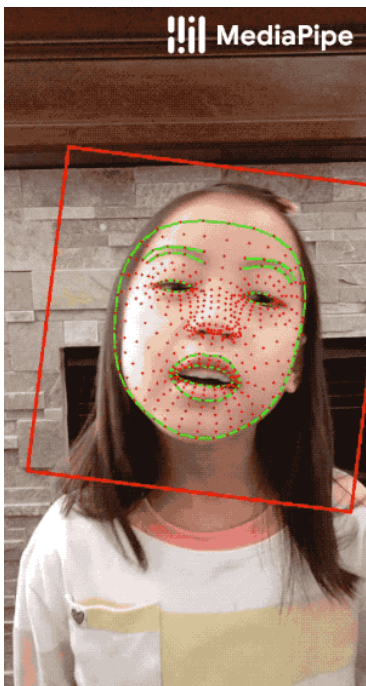
MediaPipe tarjoaa viisitoista erilaista ratkaisua, joista kaksitoista on toiminnallisia Unity-pelimoottorin-liitännäisellä.

MediaPipe Face Detection (kuva 6) on nopea kasvojentunnistusratkaisu. Se sisältää 6 maamerkkiä (kuvassa 6 nähtävät punaiset pallot) sekä tuen useiden kasvojen tunnistamiseen. Se perustuu BlazeFaceen, joka on kevyt ja hyvän suorituskyvyn omaava mobiilialustoille räätälöity kasvojentunnistin (5).



Kuva 6. MediaPipe Face Detection -esimerkki (5).

MediaPipe Face Mesh -ratkaisu (kuva 7) arvioi 468 3D-kasvojen maamerkkiä reaaliajassa. Se toimii myös mobiililaitteilla. Se pystyy päättämään 3D-kasvojen pintaa hyödyntäen koneoppimista. Se vaatii vain yhden kameran sisääntulon ilman erillistä syvyysanturia. Tämä ratkaisu sisältää myös Face Transform -moduulin, joka kattaa eron kasvojen maamerkkiarvioinnin ja hyödyllisten reaaliaikaisten lisätyn todellisuuden (AR) sovellusten välillä. Se arvioi kasvojen muutoksia metrisessä 3D-tilassa maamerkkien avulla (6).



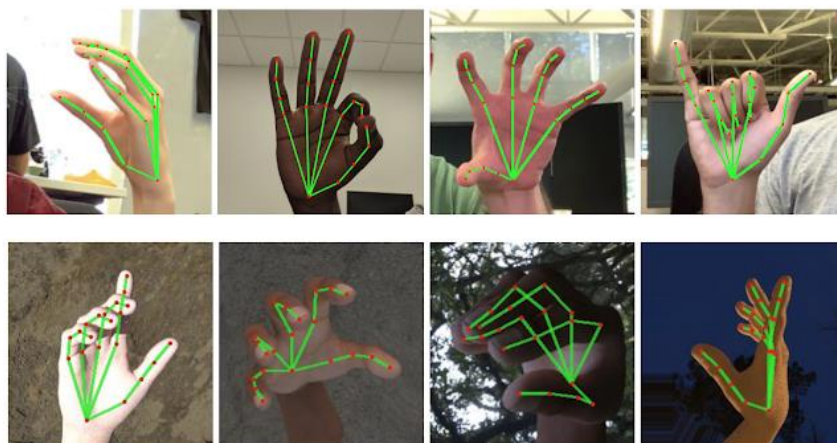
Kuva 7. MediaPipe Face Mesh -esimerkki (6).

MediaPipe Iris (kuva 8) on koneoppimisratkaisu tarkkaan iiriksen estimointiin. Se pystyy seuraamaan iiriksen, pupillien ja silmien ääri viivoja koskevia määrittejä yhdellä RGB-kameralla reaaliajassa ilman erikoislaitteita. Vaihtelevien valo-olosuhteiden ja tukosten, kuten hiusten tai ihmisten silmien siristelyn, vuoksi iiriksen seuranta on haastava tehtävä ratkaista varsinkin mobiililaitteilla. Iris-seurantaa voidaan käyttää myös kameran ja käyttäjän etäisyyden arvioimiseen (7).



Kuva 8. Esimerkki MediaPipe Iris -ratkaisusta (7).

MediaPipe Hands (kuva 9) on käsien ja sormien seurantaratkaisu, joka käyttää koneoppimista päättämään 21 käden 3D-maamerkkiä yhdestä ruudusta. Se saavuttaa reaaliaikaisen suorituskyvyn myös matkapuhelimella ja kykenee hahmottamaan usean käden samanaikaisesti (8).



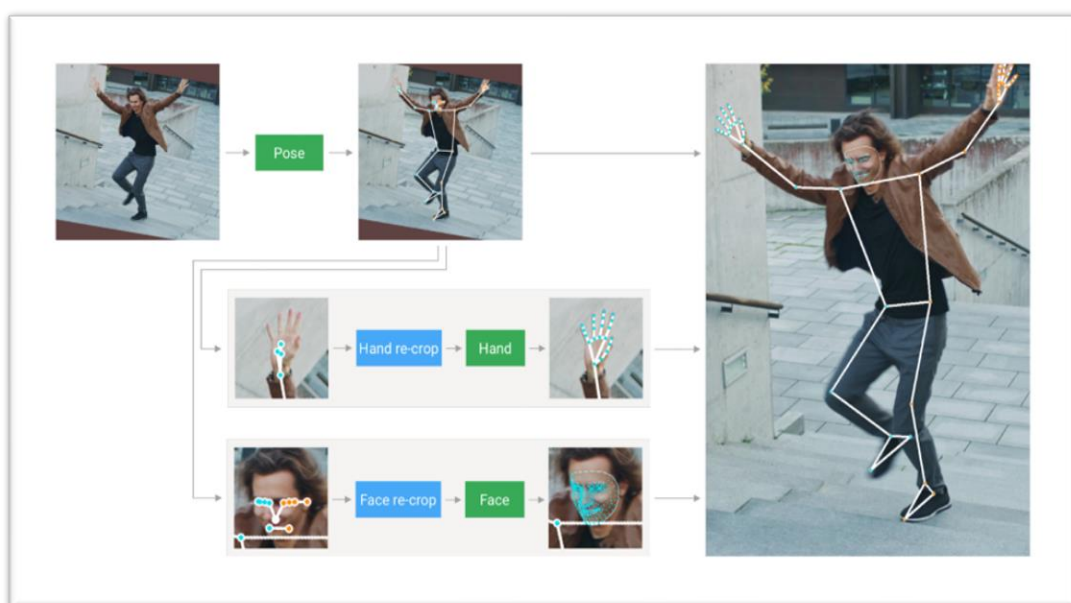
Kuva 9. MediaPipe Hands -ratkaisun esimerkki (8).

MediaPipe Pose (kuva 10) on koneoppimISRatkaisu korkean tarkkuuden vartaloasennon seurantaan. Se päättelee 33 3D-maamerkkiä ja taustan segmentointimaskia koko kehossa RGB-videokehysistä hyödyntäen MediaPipen BlazePose-tutkimusta. Ihmisen asennonarviointia videosta käytetään erilaisissa sovelluksissa, kuten fyysisten harjoitusten määrittämisessä, viittomakielen tunnistamisessa ja koko kehon eleohjauksessa (9).



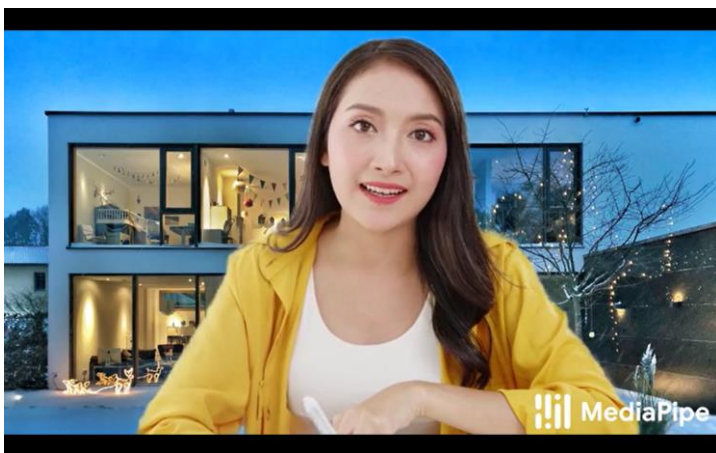
Kuva 10. MediaPipe Pose -esimerkki (9).

MediaPipe Holistic (kuva 11) yhdistää samanaikaisesti ihmisen asennon, kasvojen ja käsien seuraamisen reaaliaikaisesti. Sitä voidaan käyttää erilaisiin sovel-
luksiin, kuten kunto- ja urheiluanalyysi, eleiden ohjaus ja viittomakielen tunnista-
minen ja lisätty todellisuus (10).



Kuva 11. MediaPipe Holisticin tiedonsiirtokanavien yleiskatsaus (10).

MediaPipe Selfie Segmentation (kuva 12) jaottelee näkyvillä olevat ihmiset taustasta. Se toimii reaaliajassa niin kannettavassa tietokoneessa kuin älypuhelimissa. Käyttökohteita ovat selfie-efektit ja videoneuvottelut, joissa henkilö on lähempänä kuin kaksi metriä kamerasta (11).



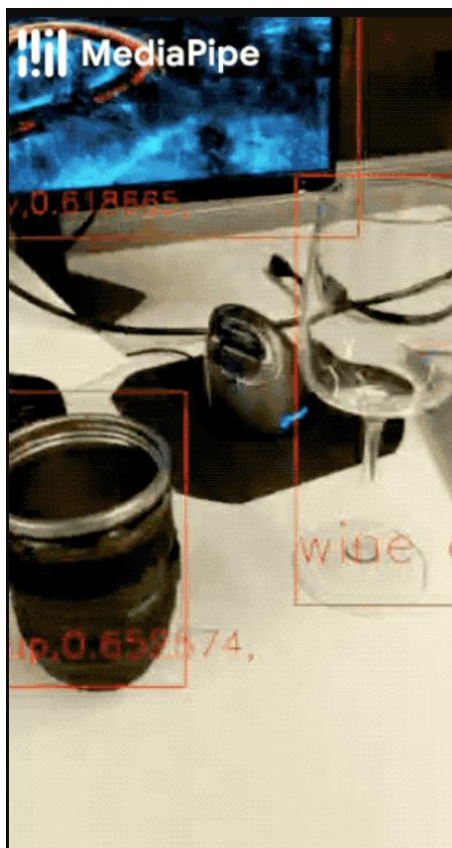
Kuva 12. Esimerkki MediaPipe Selfie Segmentationista (11).

MediaPipe Hair Segmentation (kuva 13) erittelee hiukset muusta päästä. Sen avulla voi esimerkiksi testata, miltä käyttäjän hiukset näyttäivät eri värisinä (12).



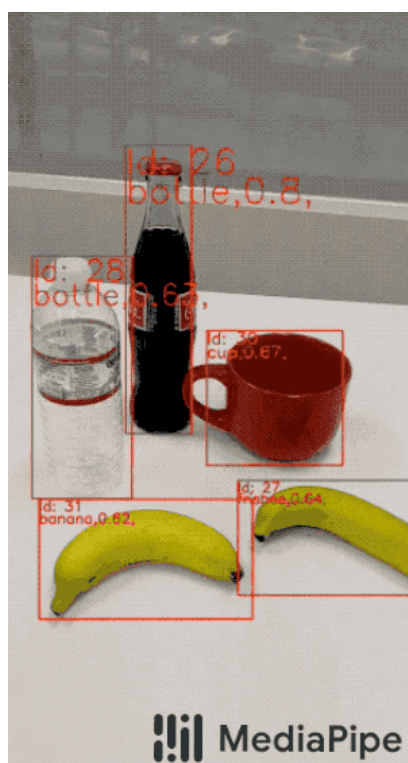
Kuva 13. MediaPipe Hair Segmentation -esimerkki (12).

MediaPipe Object Detection (kuva 14) paikantaa ja merkitsee erilaisia objekteja (13).



Kuva 14. MediaPipe Object Detection -esimerkki (13).

MediaPipe Box Tracking (kuva 15) on laatikonseurantaratkaisu, joka käyttää kuvakehyksiä video- tai kameravirrasta. Se luo aloituslaatikoiden sijainnit aikaleimoilla, jotka osoittavat seurattavat 2D-alueet, ja sitten laskee seurattujen laatikoiden sijainnit jokaiselle kehykselle. Tämä ratkaisu koostuu kolmesta pääkomponentista, jotka ovat liikeanalyysikomponentti, virtauspakkauskomponentti ja laatikonseurantakomponentti (14).



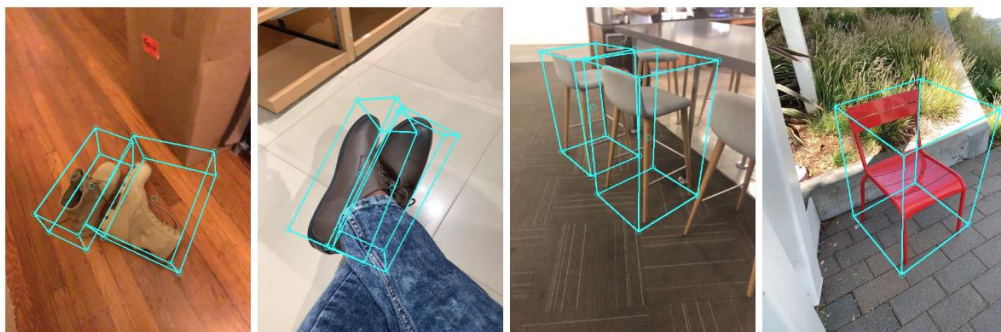
Kuva 15. Laatikon seuranta yhdistettynä koneoppimis pohjaiseen objektien tunnistukseen (14).

MediaPipe Instant Motion Tracking (kuva 16) perustuu MediaPipe Box Tracking -ratkaisuun ja tarjoaa AR-seurannan laitteissa ja alustoissa ilman alustusta tai kalibrointia. Tämän ratkaisun avulla voi helposti sijoittaa virtuaalista 2D- ja 3D-sisältöä staattisille tai liikkuville pinnoille. Tämä tarkoittaa, että ne voivat olla saumattomasti vuorovaikutuksessa todellisen ympäristön kanssa (15).



Kuva 16. MediaPipe Instant Motion Tracking -esimerkki (15).

MediaPipe Objectron (kuva 17) on reaaliaikainen 3D-objektien tunnistusratkaisu jokapäiväisille esineille. Ratkaisu havaitsee kohteet 2D-kuvissa ja arvioi niiden asennot koneoppimismallin avulla. 3D-ennustuksella voidaan kaapata kohteen koko, sijainti ja suunta maailmassa, mikä johtaa erilaisiin sovelluksiin robotiikassa, itseohjautuvissa ajoneuvoissa, kuvien haussa ja lisätyssä todellisuudessa. Objektien erilaiset muodot ja ulkonäkö ovat haastavia ongelmia, kun yritetään tunnistaa 3D-kohteita 2D-kuvista. (16.)



Kuva 17. MediaPipe Objectron -esimerkki (16).

2.5 MediaPipea vastaavia ratkaisuja

On olemassa myös muita MediaPipen kaltaisia ratkaisuja, kuten esimerkiksi OpenPose, AlphaPose ja InData Labs Pose Estimation.

OpenPose (linkki: <https://github.com/CMU-Perceptual-Computing-Lab/open-pose>) on reaaliaikainen monen henkilön asennontunnistuskirjasto. Se pystyy havaitsemaan yhteensä 135 avainpistettä. Se on COCO 2016 Keypoints Challenge voittaja. Sen ovat luoneet Ginés Hidalgo, Yaser Sheikh, Zhe Cao, Yaad-hav Raaj, Tomas Simon, Hanbyul Joo ja Shih-En Wei (17).

AlphaPose (linkki: <https://github.com/MVIG-SJTU/AlphaPose>) on myös reaaliaikainen monen henkilön asennontunnistuskirjasto. Sen ovat luoneet Hao-Shu Fang, Jiefeng Li, Yuliang Xiu, Ruiheng Chang ja Cewu Lu (18).

InData Labs Pose Estimation (linkki: <https://indatalabs.com/services/human-pose-estimation>) on tekoälyllä toimiva asennonarviointitekniikka mukautetuilla ominaisuuksilla. Sitä voidaan hyödyntää terveydenhuolto- ja urheilusovelluksissa, valvontateollisuudessa ja lisätyn todellisuuden tehosteissa (19).

3 Polven kuntoutus

3.1 Polven valgus- ja varuskulma

Yksi mahdollinen altistava tekijä kontaktittomille polvivammoille on tavanomaista suurempi polven valguskulma toiminnan aikana. Kun polvi asettuu valgus (sisäänkääntynyt)- tai varus (uloskääntynyt) -asentoon toiminnan aikana, se asettaa nivelen vähemmän vakaaseen asentoon ja tekee siitä alttiimman vaurioille. Tätä varten tarvitaan päteviä klinisiä seulontatyökaluja sellaisten henkilöiden tunnistamiseen, joilla on suurempi polven valgus-kulma, jotta suuren riskin omaavat henkilöt voidaan suunnata vammojen ehkäisyohjelmiin toiminnallisen polven valgus-liikkeen korjaamiseksi. Tutkimuksissa on todettu, että mediaalisen polven siirtymän omaavilla henkilöillä on merkittävästi suurempi valguskulma (keskiarvo = 12,86) verrattuna tutkittuun kontrolliryhmään (keskiarvo =

6,08). Yhden jalan kyykky on yleinen toiminnallisen liikkeen arviointi, jota voidaan käyttää sellaisten henkilöiden tunnistamiseen, joilla on korkean riskin biomekaanisia malleja, jotka voivat asettaa heidät suurempaan riskiin loukkaantua (20).

3.2 Polvivammojen syntymekanismit

Yleinen äkillisten polvivammojen aiheuttaja on nivelen vääntyminen, jolloin vaurio tyypillisimmin kohdistuu nivelsiteisiin. Polven eturistisidevammoista peräti 70–80 % sattuu ilman kontaktia. Äkillinen polvivamma tapahtuu tyypillisesti tilanteessa, jossa liikehallinnan pettämisen seurauksena polvi painuu äkillisesti sisäänpäin (valgus-liike) ja samaan aikaan tapahtuu voimakas säären kiertyminen joko sisäänpäin tai ulospäin suhteessa reiteen.

Polven rasitusvammat ovat yleensä seurausta yksipuolisesta, paljon toistoja sisältävästä ja liian tiheästi samankaltaisena toistuvasta harjoittelusta, mutta myös suoritustekniikkavirhe tai rakenteellinen poikkeavuus saattaa olla vamman taustalla, kuten myös nopeat muutokset harjoittelun intensiteetissä tai määrässä tai harjoitusolosuhteissa (esim. urheilualustan muutokset). Yleisin rasitusvamman kohdekudos on lihas-jänneyksikkö kiinnityskohtineen (21).

3.3 Biomekaniikan perusteet

Biomekaniikalla tarkoitetaan elävän ruumiin liikkuvuutta ja lihasten, luiden, jänneiden ja nivelsiteiden yhteistoimintaa liikkuvuuden tuottamiseksi. Se on osa laajempaa kinesiologian alaa, jossa keskitytään erityisesti liikkeen mekaniikkaan (22). Perinteisesti biomekaniikka tutkii ihmisen tuottamia voimia ja liikettä sekä liikuntabiologiassa hermostollista säätelyä, jolla näitä voimia ja liikkeitä hallitaan. Biomekaniikan tutkijat etsivät vastauksia kysymyksiin, kuten esimerkiksi miten hermolihaskäyttäytyminen adaptoituu liikunnassa ja eri olosuhteissa, miten hermolihaskäyttäytyminen toimii liikkeessä ja eri lihastyötavoilla tai millainen on optimaalinen suoritustekniikka eri urheilulajeissa (23).

Polven biomekaniikkaa tarvitaan päivittäisiin aktiviteetteihin, kuten kävelyyn ja portaiden kiipeämiseen. Tällaisten aktiviteettien aikana polvi kokee jopa 2,7–4,9-kertaisesti kehon painon. Polvi on samanaikaisesti tarkoitettu kestäämään kuormitusta ja auttamaan kehoa liikkumaan. Polvessa on myös mekanismi, joka vähentää pystyssä pysymiseen tarvittavaa voimaa. Viime aikoina biolääketieteen tekniikan alalla on kehitetty menetelmiä, joilla voidaan korjata trauman tai vain heikentymisen johdosta syntyneitä ongelmia (24).

4 Polven kulman laskeminen Unityn MediaPipe-liitännäisellä

4.1 Sovelluksen tavoitteet

Opinnäytetyön tavoitteena oli luoda pienin toimiva tuote Unity-sovelluksesta, joka MediaPipe-liitännäistä hyödyntäen mittaa ja tallentaa tietoa polven kulmasta silloin, kun mitattava kohde kyykistyy yhdellä jalalla. Kyykistys-suoritukselta tulisi saada tietoa polven kulman asteista, jotta voidaan tarkkailla, meneekö polvi vinoon suorituksen aikana. Suorituksesta saatua tietoa tulee tallentaa niin tietokantaan tekstin muodossa kuin videona tehdystä suorituksesta. Tallennettua tietoa tulisi myös pystyä lukemaan sovelluksen sisältä.

4.2 Sovelluksen tekemiseen käytetyt työkalut

Polven kulman laskemiseen ja siitä saatavan tiedon tallentamiseen tarvitaan muutamia erilaisia työkaluja Unity-pelimoottorin lisäksi.

MediaPipen ratkaisusta valituksi tuli MediaPipe Pose, koska se sisältää tarvittavat 3D-maamerkit polvien, nilkkojen ja lonkkien seuraamiseen. Toisena varteen otettavana vaihtoehtona olisi ollut MediaPipe Holistic, mutta se sisältää naamalle ja käsille enemmän maamerkkejä, jotka eivät ole tarpeellisia tavoitteiden näkökulmasta ja mahdollisesti saattaa vaikuttaa ohjelman suoritukseen.

RockVR Video Capture on Unity-liitännäinen, jonka avulla voidaan kaapata videota ja ääntä suoraan Unity-sovelluksesta. Sen ominaisuuksia ovat mm.

videon tallennus mistä tahansa kamerasta, usean kameran kaappaus samanaikaisesti, kuvankaappauksen resoluutio jopa 8K, stereovideokaappaus ja 360 asteen videokaappaus (25). Sovelluksessa sitä käytetään videonkaappaamiseen silloin, kun polven kulman tietoja kerätään kyykistymissuorituksen aikana.

Unisave on Unity-pelimoottorin kanssa toimiva taustajärjestelmä. Se yhdistää palvelimen, tietokannan, viitekehiksen ja alustan samaan pakettiin (26).

Unisavea käytetään sovelluksessa sisäänkirjautumiseen, käyttäjän rekisteröintiin, tiedon tallentamiseen ja lukemiseen ArangoDB-tietokannasta.

4.3 MediaPipe-liitännäisen asennus Unityyn

Työssä käytettiin hyväksi Homuler-nimimerkin GitHubiin lataamaa Unity MediaPipe -liitännäistä (linkki: <https://github.com/homuler/MediaPipeUnityPlugin>). Liitännäisen tavoitteena on kääntää MediaPipen ominaisuudet C++:sta C#:lle, jotta niitä voidaan käyttää natiivisti Unityn sisällä. MediaPipe Unity -liitännäinen voidaan asentaa käytettäväksi Linuxiin, Windowsiin tai Maciin. Tässä dokumentissa keskitytään liitännäisen asennukseen Windows-käyttöjärjestelmään.

MediaPipe Unity -liitännäinen voidaan asentaa kolmella eri tavalla. Ensimmäinen tapa on asentaa liitännäinen käyttäen Docker Windows Containeria. Docker Windows Container -asennuksessa täytyy ottaa huomioon, että se käyttää Windowsin Hyper-V-backend-ominaisuuksia, mikä tarkoittaa, ettei se toimi, jos laitteistossa, johon asennus tehdään, on käytössä Windowsin kotiversio, esimerkiksi Windows 10 Home -käyttöjärjestelmä. Toisena vaihtoehtona on asentaa liitännäinen käyttämällä Docker Linux Containeria. Docker Linux Container -asennuksella voidaan rakentaa kirjastoja vain Androidille, joten jos tavoitteena on luoda Windows-sovellus, täytyy käyttää jotain muuta asennustapaa. Kolmantena vaihtoehtona on tehdä manuaalinen Windows-asennus. Ennen kuin manuaalinen asennus voidaan aloittaa, täytyy asentaa muutama muu ohjelma. Ensimmäisenä tarvitaan Python 3.9.0 tai uudempi, Bazelisk ja NuGet. Muita asennettavia ohjelmia tai työkaluja ovat MSYS2, Visual Studio C++ Build Tools 2019, Visual Studio WinSDK, Numpy ja OpenCV, jos asennuksen tekijä haluaa

yhdistää OpenCV:n dynaamisesti. Jos asennettu OpenCV-versio on muu kuin 3.4.16, muokataan OpenCV-versio siihen, mikä versio on asennettuna `third_party/opencv_windows.BUILD-` ja `WORKSPACE-`tiedostoihin (27).

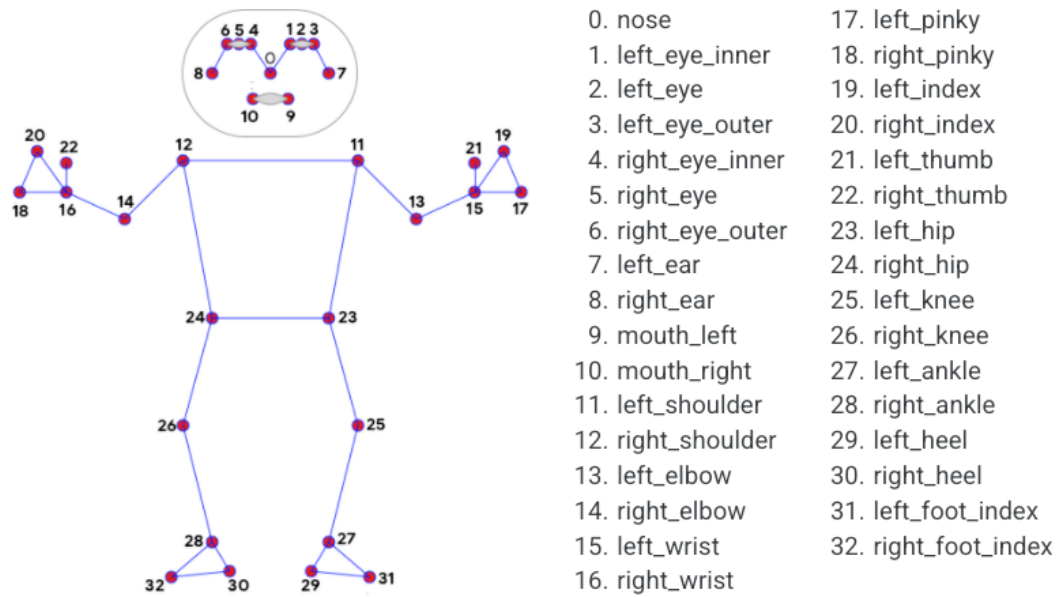
Aluksi yritettiin asentaa liitännäistä kaikilla mahdollisilla tavoilla, mutta ilman tulosta. Koneen vaihdon myötä yritettiin ensimmäiseksi asentaa liitännäinen Linux Containerilla, mutta kun Docker ei saanut yhteyttä palvelimeen, tehtiin onnistunut manuaalinen Windows-asennus.

4.4 MediaPipe Posen kordinaattien paikantaminen ja polven kulman laskeminen

MediaPipe Unity -liitännäisen onnistuneen asentamisen jälkeen voidaan aloittaa MediaPipe Posen koodin tutkiminen, jotta voidaan paikantaa, missä skriptissä MediaPipe Pose käsittelee maamerkkejä, jotta niiden koordinaatteihin päästään käsiksi. PoseWorldLandmarkListAnnotationController-nimisestä skriptistä löytyy seuraava muuttuja:

```
private IList<Landmark> _currentTarget;
```

Tämä muuttuja on lista, joka sisältää MediaPipe Posen käyttämät maamerkit. Maamerkkien löydyttyä voidaan paikantaa kuvassa 18 esiintyvän mallin avulla, mitä maamerkkejä halutaan käyttää.



Kuva 18. MediaPipe Posen 33 maamerkkiä (9).

Kun halutut maamerkit on löydetty, niiden koordinaatit voidaan määrittää omiin muuttujiin esimerkiksi alla olevalla esimerkikoodin 1 lailla.

```

public void SetValues() //Sets X, Y, Z values for knees, hips and ankles
{
    R_HipX = _currentTarget[24].X;
    R_HipY = _currentTarget[24].Y;
    R_HipZ = _currentTarget[24].Z;

    R_KneeX = _currentTarget[26].X;
    R_KneeY = _currentTarget[26].Y;
    R_KneeZ = _currentTarget[26].Z;

    R_AnkleX = _currentTarget[28].X;
    R_AnkleY = _currentTarget[28].Y;
    R_AnkleZ = _currentTarget[28].Z;

    L_HipX = _currentTarget[23].X;
    L_HipY = _currentTarget[23].Y;
    L_HipZ = _currentTarget[23].Z;

    L_KneeX = _currentTarget[25].X;
    L_KneeY = _currentTarget[25].Y;
    L_KneeZ = _currentTarget[25].Z;

    L_AnkleX = _currentTarget[27].X;
    L_AnkleY = _currentTarget[27].Y;
    L_AnkleZ = _currentTarget[27].Z;

    HipR = new Vector3(R_HipX, R_HipY, R_HipZ);
    KneeR = new Vector3(R_KneeX, R_KneeY, R_KneeZ);
    AnkleR = new Vector3(R_AnkleX, R_AnkleY, R_AnkleZ);

    HipL = new Vector3(L_HipX, L_HipY, L_HipZ);
    KneeL = new Vector3(L_KneeX, L_KneeY, L_KneeZ);
    AnkleL = new Vector3(L_AnkleX, L_AnkleY, L_AnkleZ);
}

```

Esimerkkikoodi 1. Metodi, joka määrittää haluttujen maamerkkien koordinaatit omiin muuttujiin.

Kun koordinaattien arvot on määritetty, niitä voidaan käyttää esimerkiksi kulmien laskemiseen. Esimerkiksi polvien kulmat voidaan laskea esimerkkikoodin 2 mukaisesti.


```

angleR = Mathf.Rad2Deg * (Mathf.Atan2(R_AnkleY - R_KneeY, R_AnkleX -
R_KneeX) - Mathf.Atan2(R_HipY - R_KneeY, R_HipX - R_KneeX));

    if(angleR < 0)
    {
        angleR += 360;
    }

    angleL = Mathf.Rad2Deg * (Mathf.Atan2(L_AnkleY - L_KneeY,
L_AnkleX - L_KneeX) - Mathf.Atan2(L_HipY - L_KneeY, L_HipX -
L_KneeX));

    if(angleL < 0)
    {
        angleL += 360;
    }

```

Esimerkkikoodi 2. Polven kulman laskeminen.

4.5 Datan tallennus eri muodoissa

Polven kulman laskemisesta saatavaa tietoa tallennetaan kolmella eri tavalla: 1. tekstitiedostona, 2. videon muodossa ja 3. tietokantaan tallennettavat tiedot.

Datan tallentamisessa tekstitiedostoon täytyy aluksi luoda uusi tekstitiedosto, jos samalla nimellä ei löydy aikaisempaa tekstitiedostoa. Jos samalla nimellä oleva tekstitiedosto löytyy, data lisätään siihen. Esimerkkikoodi 3 luo uuden tekstitiedoston.

```

private void WriteOnClick()
{
    //Debug.Log(name.text);

    if (!Directory.Exists(Application.streamingAssetsPath + "/Text-
Files/" + name.text + ".txt"))
    {
        _ = Directory.CreateDirectory(Application.streamingAssetsPath
+ "/TextFiles/");

        string txtDocumentName = Application.streamingAssetsPath +
"/TextFiles/" + name.text + ".txt";

        if (!System.IO.File.Exists(Application.streamingAssetsPath +
"/TextFiles/" + name.text + ".txt"))
        {
            File.WriteAllText(txtDocumentName, name.text + "\n\n");
        }

        filePath = txtDocumentName;
        PoseWorldLandmarkListAnnotationController.path = filePath;
        DataBase.GetComponent<HomeController>().Username(name.text);
        usedName = name.text;

        if (_Visited == false)
        {
            _Visited = true;
            PoseWorldLandmarkListAnnotationController._Visited = false;
            SceneManager.LoadScene("Start Scene", LoadSceneMode.Addi-
tive);
        }

        else
        {
            PoseWorldLandmarkListAnnotationController._Visited = false;
            SceneManager.LoadScene("Pose Tracking", LoadSceneMode.Addi-
tive);
        }
    }

    else
    {
        //open text file
        Debug.Log("File allready exist!");
        StreamReader reader = new StreamReader(Application.stream-
ingAssetsPath + "/TextFiles/" + name.text + ".txt");
        Debug.Log(reader.ReadToEnd());
        showData.text = reader.ReadToEnd();
        reader.Close();
    }
}
}

```

Esimerkkikoodi 3. Metodi tekstitiedostojen luomiseen.

Sen jälkeen kuin tekstitiedosto on luotu, sovellus siirtyy sceneen, jossa polven kulman laskenta suoritetaan. Esimerkkikoodi 4 kirjoittaa saadun datan tekstitiedostoon.

```

public void AverageAngles() //Counts average angles & writes data
{
    if (loaded == true)
    {
        if (counter < 3)
        {
            RightAngles.Add(angleR);
            averageR = 0;
            sumR = 0;

            for (var i = 0; i < RightAngles.Count; i++)
            {
                sumR += RightAngles[i];
                //Debug.Log (sumR);
            }

            //Debug.Log (sumR + "/" + RightAngles.Count);

            averageR = sumR / RightAngles.Count;

            UI_AngleR.text = averageR.ToString();

            StreamWriter writer = new StreamWriter(path, true);
            writer.WriteLine("R" + RightAngles.Count.ToString() + ": " +
angleR);
            writer.Close();

            dataAverage = averageR.ToString();
            dataName = "R" + RightAngles.Count.ToString();
            //data = true;
            angleList.Add("R" + RightAngles.Count.ToString() + ": " +
angleR);

            barchartList.Add(angleR);
        }

        else if (counter > 2 && counter < 6)
        {
            LeftAngles.Add(angleL);

            averageL = 0;

            sumL = 0;

            for (var i = 0; i < LeftAngles.Count; i++)
            {
                sumL += LeftAngles[i];
            }

            averageL = sumL / LeftAngles.Count;

            UI_AngleL.text = averageL.ToString();

            StreamWriter writer = new StreamWriter(path, true);

```

```

        writer.WriteLine("L" + LeftAngles.Count.ToString() + ": " +
angleL);
        writer.Close();

        dataAverage = averageL.ToString();
        dataName = "L" + LeftAngles.Count.ToString();
        //data = true;
        angleList.Add("L" + LeftAngles.Count.ToString() + ": " + an-
gleL);
        barchartList.Add(angleL);
    }

    if(counter == 2)
    {
        averageR = sumR / RightAngles.Count;

        UI_AngleR.text = averageR.ToString();

        StreamWriter writer = new StreamWriter(path, true);
        writer.WriteLine("Right Angle Average: " + averageR);
        writer.Close();

        dataAverage = averageR.ToString();
        dataName = "Right Angle Average: ";
        //data = true;
        angleList.Add("Right Angle Average: " + dataAverage);
    }

    if (counter == 5)
    {
        averageL = sumL / LeftAngles.Count;

        UI_AngleL.text = averageL.ToString();

        StreamWriter writer = new StreamWriter(path, true);
        writer.WriteLine("Left Angle Average: " + averageL);
        writer.Close();

        dataAverage = averageL.ToString();
        dataName = "Left Angle Average: ";
        //data = true;
        angleList.Add("Left Angle Average: " + dataAverage);
        barchartList.Add(averageL);
    }

    //Debug.Log("averag2: " + averageR);

    StreamReader reader = new StreamReader(path);
    //Print the text from the file
    //Debug.Log(reader.ReadToEnd());
    reader.Close();

    }
}

```

Esimerkkikoodi 4. Metodi, joka kirjoittaa polven kulmasta saadun datan teksti-
tiedostoon.

Aluksi yritettiin käyttää RockVR-videokaappausta sen komponentin ominaisuuksia hyödyntäen, mutta tämä ei toiminut halutulla tavalla vaan videonkaappaus jäi usein liian lyhyeksi. Lopulta päädyttiin tekemään esimerkkikoodi 5, joka manuaalisesti aloittaa ja lopettaa videonkaappauksen.

```
public class StartRec : MonoBehaviour
{
    public GameObject rec;
    private float counter;
    bool visited;
    // Start is called before the first frame update
    void Start()
    {
        rec.GetComponent<RockVR.Video.VideoCaptureCtrl>().StartCapture();
        counter = 0;
        visited = false;
    }

    // Update is called once per frame
    void Update()
    {
        counter += Time.deltaTime;

        if(counter > 75 && visited == false)
        {
            rec.GetComponent<RockVR.Video.VideoCaptureCtrl>().StopCapture();
            visited = true;
        }
    }
}
```

Esimerkkikoodi 5. Koodi, joka manuaalisesti aloittaa ja lopettaa videonkaappauksen.

Tämän jälkeen videonkaappaus toimi moitteettomasti.

Tallennettaessa dataa tietokantaan käyttäjän täytyy ensiksi joko kirjautua sisään tai rekisteröityä sähköpostiosoitteella. Tämän jälkeen annetaan tiedoston nimi, joka voi esimerkiksi olla mitattavan henkilön nimi. Esimerkkikoodi 6 ajetaan, kun tiedoston nimi on valittu ja nappia painettu.

```

public void RenameUser(string name)
{
    var player = Auth.GetPlayer<PlayerEntity>();

    player.name = name;

    player.FillWith(player);

    player.Save();
}

```

Esimerkkikoodi 6. Metodi, joka muuttaa käytössä olevan nimen.

Esimerkkikoodi 6 muuttaa käytössä olevan nimen, jonka alle kerättävät tiedot tallennetaan. Tämän jälkeen edetään sceneen, josta varsinaiset polven kulman arvot kerätään. Esimerkkikoodi 7 joko luo uuden listan tai etsii listan annetulla nimellä, joka kuuluu käyttäjälle, joka on kirjautunut sisään.

```

public void CreateList(string name, List<string> angles)
{
    var player = Auth.GetPlayer<PlayerEntity>();

    var existingList = DB.TakeAll<ListEntity>().Filter(l =>
l.owner.email == player.email)
        .Filter(l => l.patient == name).
        Get();

    try
    {
        if (existingList[0].owner.email == player.email)
        {
            existingList[0].angles.AddRange(angles);
            existingList[0].Save();
        }
    }

    catch
    {
        var list = new ListEntity()
        {
            owner = player,
            patient = name,
            angles = angles,
        };

        list.Save();
    }

    player.Save();
}

```

Esimerkkikoodi 7. Palvelinmetodi, joka luo uuden listan tai etsii annetun listan.

Tietokannassa olevaa tietoa voidaan myös lukea silloin, kun se kuulu sille käyttäjälle, jona kirjauduttiin sisään, ja lukunappia on painettu. Lukunapin painaminen pyöräyttää esimerkkikoodin 8.

```
private void ReadOnClick()
{
    builder.Clear();

    OnFacet<HomeFacet>
    .Call<ListEntity>(nameof(HomeFacet.ReadList), name.text)
    .Then((ListEntity) =>
    {
        //var player = Auth.GetPlayer<PlayerEntity>();

        for (int i = 0; i < ListEntity.angles.Count; i++)
        {
            builder.AppendLine(ListEntity.angles[i].ToString());
        }

        showData.text = "\n\n" + ListEntity.patient.ToString() +
        "\n\n" + builder.ToString() + "\n\n";

        if (builder.ToString() == "\n\n" + "Either the file doesn't
        exist or access is denied!" + "\n\n")
        {
            ListEntity.Delete();
        }
    })
    .Done();
}
```

Esimerkkikoodi 8. Metodi datan lukemista varten.

Esimerkkikoodi 8 tekee palvelinmetodikutsun, joka käynnistää esimerkkikoodin 9, joka onnistuessaan tulostaa halutun tiedoston datan UI-elementtiin käyttäjän luettavaksi.


```

public ListEntity ReadList(string patient)
{
    var player = Auth.GetPlayer<PlayerEntity>();

    var existingList = DB.TakeAll<ListEntity>().Filter(l => l.patient
== patient).Get();

    List<string> error = new List<string>();

    Debug.Log(existingList[0].owner.email.ToString());

    try
    {
        if(existingList[0].owner.email != player.email)
        {
            Debug.Log("No Access");
            error.Add("\n\n" + "Either the file doesn't exist or access is
denied!" + "\n\n");

            var list = new ListEntity()
            {
                owner = player,
                patient = patient,
                angles = error,
            };

            return list;
        }

        if (existingList[0].patient.ToString() == patient && exist-
ingList[0].owner.email == player.email)
        {
            Debug.Log("Existing");
            return existingList[0];
        }
    }

    catch
    {
        Debug.Log("List");
        error.Add("\n\n" + "Either the file doesn't exist or access is
denied!" + "\n\n");

        var list = new ListEntity()
        {
            owner = player,
            patient = patient,
            angles = error,
        };

        return list;
    }

    return existingList[0];
}

```

Esimerkkikoodi 9. Palvelinmetodi, joka tulostaa halutun tiedoston datan UI-elementtiin.

5 Projektin tulokset

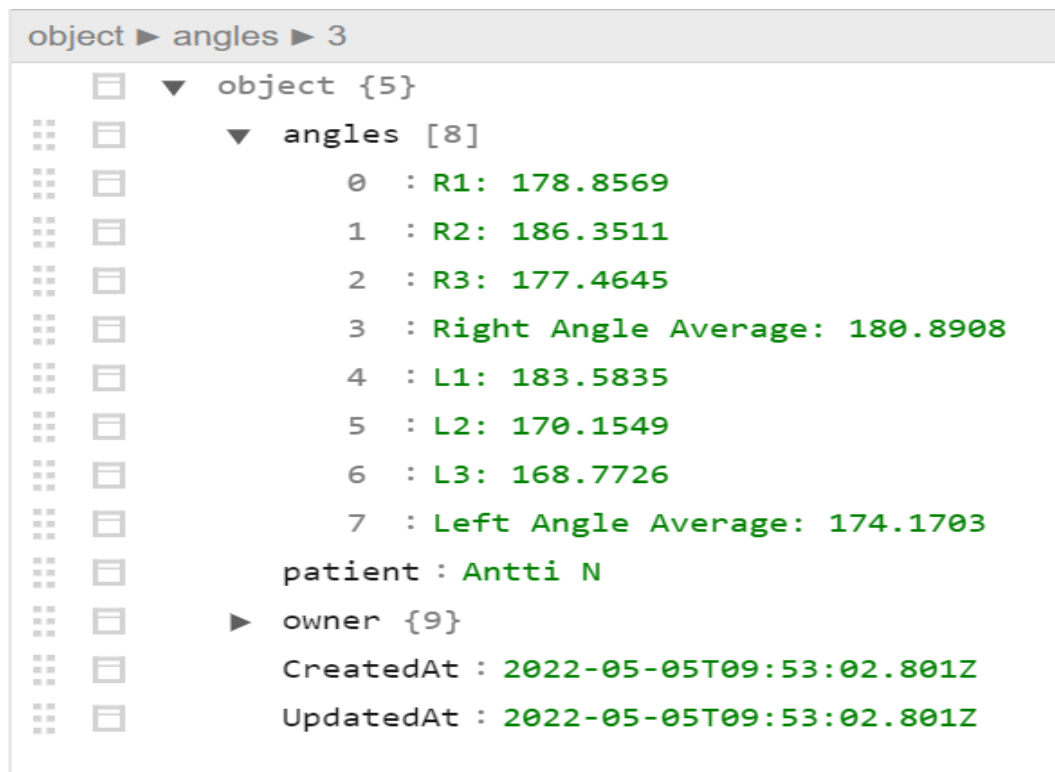
5.1 Projektin haastavimmat osuudet

Ihan projektin alussa oli haasteita asennuksen kanssa, mutta koneen vaihdon jälkeen asennus sujui hyvin. Oikean kaavan löytäminen polven kulman laskemiseen tuotti myös vaikeuksia, mutta hieman kyselemällä saatiin oikea laskukaava. Tietokannan kanssa oli myös hieman haasteita, varsinkin, kun Yritettiin rajata sitä, kuka pystyy lukemaan kenellekin rekisteröityjä tietoja. Ongelma lopuksi ratkesi käyttämällä try-catch-lauseketta pelkän if-else-lausekkeen sijaan.

5.2 Vahvuudet ja heikkoudet

Kokonaisuudessaan projektina tehty pienin toimiva tuote Unity-sovelluksesta onnistui. Se laskee polven kulman melko tarkasti, vaikka kulman arvioinnissa silloin, kun mitattavissa oleva henkilö seisoo suorassa, saattaa olla muutaman asteen vaihtelua. Polven kulman mittaus yhden jalan kyykyssä antoi järkeviä lukuja (kuva 19). Lukujen tulisi olla 170–190 asteen välillä terveen polven omaavilla kyykkääjillä. Jos asteet menevät alle 170 asteen tai yli 190 asteen, testatun henkilön polven kulma vääntyy joko liikaa ulos- tai sisäänpäin. Yksi heikkouksista oli testaajien määrä, koska opinnäytetyön puitteissa sovellusta testasi vain

tekijä itse.



Kuva 19. Polven kulman tuloksia tietokannasta.

Silloin tällöin MediaPipe Pose saattaa myös epäonnistua arviomaan kuvassa olevan henkilön raajojen sijainnin, mikä aiheuttaa virheitä koodissa, mutta ei onneksi kaada sovellusta. Nämä virheet myös ainakin osittain johtuvat siitä, että mitattava henkilö ei ole kokonaan nähtävissä kuvassa. Sovellus myös vaatii aika paljon tilaa koneen ja mitattavan henkilön välille.

Datan tallentaminen tekstitiedostoon ja tietokantaan toimivat kuten niiden pitää, ja myös tietokannasta tiedon lukeminen onnistuu halutulla tavalla. Videotallennuksessa oli aluksi välillä pieniä ongelmia. Sovellus onnistui käynnistämään videotallenteen, mutta ainakin välillä videotallennus jäi lyhyemmäksi kuin tallenteen ajaksi määritelty pituus, mutta sen jälkeen kuin koodattiin video alkamaan ja loppumaan skriptin eikä komponentin kautta, videotallennus toimii moitteettomasti.

Myös ohjelman silmukointi ei onnistu. Riippuen siitä kumpaa tapaa käytetään scenen vaihdokseen, törmätään ongelmiin. Jos scenen vaihdos tehdään muodossa `LoadSceneMode.Single`, koko ohjelma kaatuu silloin, kun yritetään mennä `MediaPipe`-ominaisuuksiin uudelleen. Silloin, jos käytetään scenen vaihtamiseen muotoa `LoadSceneMode.Additive`, vanhat `mediaPipe Pose` -ominaisuudet jäävät taustalle käyntiin ja muiden scenejen UI-elementit eivät enää toimi normaalisti.

5.3 Parannusehdotuksia

Ensimmäisenä parannusehdotuksena on luonnollisesti korjata jäljelle jääneet pienet ohjelmointivirheet, kuten sovelluksen silmukointi. Myös kunnollisen sähköpostiosoitteen verifiointin luominen tietokantaan tallentamista varten voisi olla hyödyllinen toiminto. Olisi myös hyvä, jos yhden jalan kyykyn aikana liian suureksi kasvanut kulma visualisoidaisiin jotenkin suorituksen aikana. Sovellus voisi myös tarkastaa sen, onko tehty kyykky tarpeeksi syvä, jotta suoritus voidaan hyväksyä. Myös sovelluksen käyttöliittymäelementtien ulkonäössä on parantamisen varaa. Tämän jälkeen sovellusta olisi hyvä testata isommalla määrällä testaajia ja kerätä enemmän palautetta, minkä jälkeen voidaan mahdollisesti tehdä vielä enemmän parannuksia sovelluksen toimintoihin.

6 Yhteenveto

Insinööriyön tavoitteena oli luoda Unity-sovelluksesta pienin toimiva tuote, joka laskee polven kulman ja tallentaa siitä saatua tietoa. Sovellus suorittaa sille asetetut toiminnot pääasiassa hyvin, vaikka muutamia parannuksia voisi vielä tehdä.

Projektissa tehdyn sovelluksen tuloksista voidaan päätellä, että `MediaPipe Pose` -ratkaisulla saadaan suhteellisen tarkasti arvioitua polven kulmaa yhden jalan kyykyn aikana, koska mitatut tulokset olivat järkeen käypiä.

Projekti toimi myös hyvänä esimerkkinä siitä, millaisilla eri tavoilla dataa voidaan tallentaa ja miten datan tallennus on toteutettu tekstitiedoston, videokaappauksen ja tietokannan suhteen.

Sovellus toimii myös oivana esimerkkinä siitä, miten Unity-pelimoottoria voidaan käyttää muuhunkin kuin perinteisiin pelisovelluksiin hyötysovelluksien muodossa. Sovellus toimii myös esimerkkinä siitä, kuinka MediaPipen ratkaisuja voidaan käyttää hyödyksi Unity-sovelluksessa. Projekti jatkuu vuoden 2022 loppuun asti.

Lähteet

- 1 Bell, Jason. 2014. Machine Learning: Hands-On for Developers and Technical Professionals. E-kirja. John Wiley & Sons.
- 2 Loebner Prize. Verkkoaineisto. University of Exeter. <<https://loebner.exeter.ac.uk/media/universityofexeter/emps/maths-computer-sci/loebner/gold.jpg>>. Luettu 19.5.2022.
- 3 Gollapudi, Sunila. 2016. Practical Machine Learning. E-kirja. Packt Publishing.
- 4 Introduction to MediaPipe. 2022. Verkkoaineisto. Learnopencv. <<https://learnopencv.com/introduction-to-mediapipe/>>. Päivitetty 1.3.2022. Luettu 19.5.2022.
- 5 MediaPipe Face Detection. Verkkoaineisto. Google Github. <https://google.github.io/mediapipe/solutions/face_detection>. Luettu 19.5.2022.
- 6 MediaPipe Face Mesh. Verkkoaineisto. Google Github. <https://google.github.io/mediapipe/solutions/face_mesh>. Luettu 19.5.2022.
- 7 MediaPipe Iris. Verkkoaineisto. Google Github. <<https://google.github.io/mediapipe/solutions/iris>>. Luettu 19.5.2022.
- 8 MediaPipe Hands. Verkkoaineisto. Google Github. <<https://google.github.io/mediapipe/solutions/hands>>. Luettu 19.5.2022.
- 9 MediaPipe Pose. Verkkoaineisto. Google Github. <<https://google.github.io/mediapipe/solutions/pose>>. Luettu 19.5.2022.
- 10 MediaPipe Holistic. Verkkoaineisto. Google Github. <<https://google.github.io/mediapipe/solutions/holistic>>. Luettu 19.5.2022.
- 11 MediaPipe Selfie Segmentation. Verkkoaineisto. Google Github. <https://google.github.io/mediapipe/solutions/selfie_segmentation.html>. Luettu 19.5.2022.
- 12 MediaPipe Hair Segmentation. Verkkoaineisto. Google Github. <https://google.github.io/mediapipe/solutions/hair_segmentation.html>. Luettu 19.5.2022.

- 13 MediaPipe Object Detection. Verkkoaineisto. Google Github. <https://google.github.io/mediapipe/solutions/object_detection>. Luettu 19.5.2022.
- 14 MediaPipe Box Tracking. Verkkoaineisto. Google Github. <https://google.github.io/mediapipe/solutions/box_tracking.html>. Luettu 19.5.2022.
- 15 MediaPipe Instant Motion Tracking. Verkkoaineisto. Google Github. <https://google.github.io/mediapipe/solutions/instant_motion_tracking.html>. Luettu 19.5.2022.
- 16 MediaPipe Objectron. Verkkoaineisto. Google Github. <<https://google.github.io/mediapipe/solutions/objectron.html>>. Luettu 19.5.2022.
- 17 Boesch, Gaudenz. 2022. A Guide to OpenPose in 2022. Verkkoaineisto. Visio.ai. <<https://viso.ai/deep-learning/openpose/>>. Luettu 19.5.2022.
- 18 AlphaPose. Verkkoaineisto. Machine Vision and Intelligence Group. <<https://mvig.sjtu.edu.cn/research/alphapose.html>>. Luettu 19.5.2022.
- 19 Human Pose Estimation and Analysis Software Development. Verkkoaineisto. InData Labs. <<https://indatalabs.com/services/human-pose-estimation>>. Luettu 19.5.2022.
- 20 Mauntel, Timothy C.; Frank, Barnett S.; Begalle, Rebecca L.; Blackburn, J. Troy & Padua, Darin A. 2014. Kinematic Differences Between Those with and without Medial Knee Displacement During a Single Leg Squat. Verkkoaineisto. Journal of Applied Biomechanics. <https://www.researchgate.net/publication/263815157_Kinematic_Differences_Between_Those_With_and_Without_Medial_Knee_Displacement_During_a_Single_Leg_Squat>. Luettu 19.5.2022.
- 21 Polvi. Verkkoaineisto. Terve urheilija. <<https://terveurheilija.fi/urheiluvammojen-ennaltaehkaisy/polvi-polvivammat/>>. Luettu 19.5.2022.
- 22 Mitä biomekaniikka tarkoittaa? Verkkoaineisto. Julinse. <<https://fi.julinse.com/mitae-biomekaniikka-tarkoittaa/>>. Luettu 19.5.2022.
- 23 Biomekaniikka. 2021. Verkkoaineisto. Jyväskylän yliopisto. <<https://www.jyu.fi/sport/fi/biomekaniikka>>. Luettu 19.5.2022.
- 24 Mohammadi, Hadi. 2018. Biomechanics. E-kirja. intechOpen.

- 25 RockVR Video Capture. Verkkoaineisto. Unity Store. <<https://assetstore.unity.com/packages/tools/video/video-capture-75653>>. Luettu 19.5.2022.
- 26 Unisave. Verkkoaineisto. <<https://unisave.cloud/>>. Luettu 19.5.2022.
- 27 Nishida, Junrou. 2022. Installation Guide. Verkkoaineisto. homuler/MediaPipeUnityPlugin. <<https://github.com/homuler/MediaPipeUnityPlugin/wiki/Installation-Guide>>. Luettu 19.5.2022.