TIETO- JA SÄHKÖTEKNIIKAN TIEDEKUNTA

Niko Pirttiaho
Arttu Räty
Matias Virtala

KASVOJA SEURAAVAN NÄKÖJÄRJESTELMÄN TOTEUTUS ROBOTTIPÄÄHÄN

Kandidaatintyö
Tietotekniikan tutkinto-ohjelma
Kesäkuu 2020

Pirttiaho N., Räty A., Virtala M. (2020) Kasvoja seuraavan näköjärjestelmän toteutus robottipäähän. Oulun yliopisto, Tietotekniikan tutkinto-ohjelma, 41 s.

TIIVISTELMÄ

Työssä kehitettiin robottiin kytketty koneoppimisjärjestelmä, joka käsittelee reaaliajassa kameralta tulevaa videosyötettä. Järjestelmää käytettiin havaitsemaan, jäljittämään ja tunnistamaan ihmisten kasvoja, sekä hallitsemaan kameraa (pään ja silmien suuntaus) tarpeenmukaisella tavalla. Kasvojentunnistukseen hyödynnettiin OpenCV:n Haar-ominaisuuksiin pohjautuvaa kaskadiluokittelijaa. Tunnistettua henkilöä seurattiin suhteessa robotin sijaintiin koordinaatistossa, joka helpottaa tunnistuksien käyttöä ja käsittelyä. Henkilön tunnistamiseen koulutettiin Tensorflow-kirjastoa käyttäen autoenkooderityyppinen neuroverkko, jolla enkoodataan kasvoista helposti muistettava ja matemaattisesti vertailtavissa oleva vektori. Tuotettu kokonaisuus onnistui kohtuullisella tarkkuudella ihmisen silmänliikkeitä imitoiden tunnistamaan ja seuraamaan kasvoja. Kyseessä oli usean kandidaattivaiheen opiskelijan laajempi yhteistyöprojekti, jossa eri ryhmät toteuttavat robotille eri toiminnallisuuksia. Ryhmien tuottamat ominaisuudet yhdistettiin käyttäen ROS (Robot Operating System) -nimistä järjestelmää.

Avainsanat: tekoäly, koneoppiminen, robotiikka, kasvojentunnistus, konenäkö, ROS, Haar, autoenkooderi, OpenCV, InMoov, neuroverkko, syväoppiminen

Pirttiaho N., Räty A., Virtala M. (2020) Implementation of a Face Tracking System for a Robothead. University of Oulu, Degree Programme in Computer Science and Engineering, 41 p.

ABSTRACT

In this work we developed a machine learning system for a robot used to process real time video feed of a camera. The system is used to detect, track and recognize faces and control a camera accordingly (head and eye movement). We used OpenCV cascade classifier which uses Haar-features for facial recognition. Recognitions are mapped to a coordinate system relative to the robot which helps the usage and processing of the detections. An autoencoder based solution was trained using Tensorflow-library for facial recognition by encoding an easily mathematically comparable vector from the faces. The produced system was able to imitate human eye movement with reasonable accuracy and track faces. The project was a part of larger collaboration between other bachelor's degree students on this project course. Each group developed a specific functionality for the robot. All of the functionalities developed by each group were combined using Robot Operating System.

Keywords: AI, machine learning, robotics, face recognition, machine vision, ROS, Haar, autoencoder, OpenCV, InMoov, neural network, deep learning

SISÄLLYSLUETTELO

ALKULAUSE

Kiitämme työn ohjaajaa Teemu Tokolaa ja professori Juha Röningiä. Kiitämme myös muita sulautettujen ohjelmistojen projektikurssin opetukseen osallistuneita.

Oulussa 11. kesäkuuta 2020

Niko Pirttiaho
Arttu Räty
Matias Virtala

LYHENTEIDEN JA MERKKIEN SELITYKSET

AI	Artificial Intelligence
CV	Computer Vision
ML	Machine Learning
NN	Neural Network
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
RISC	Reduced Instruction Set Computer
ROS	Robot Operating System
USB	Universal Serial Bus
I/O	Input/Output
HD	High Definition
RGB	Red-Green-Blue
GDPR	General Data Protection Regulation
FoV	Field of View
VR	Virtual Reality
LBP	Local Binary Pattern
JSON	JavaScript Object Notation
PCA	Principal Component Analysis

1. JOHDANTO

Teknologian kehityksen myötä on erilaisten robotiikkaa ja konenäköä hyödyntävien järjestelmien kysyntä lisääntynyt suuresti. Erityisesti teollisuudessa on kasvava tarve automaattiselle prosessinvalvonnalle ja kokoonpanolle, ja tietoliikenteessä data-analyysille. Näihin tarpeisiin kehitetyiltä laitteilta vaaditaan kykyä suorittaa monia yleensä ihmisille tyypillisiä toimenpiteitä. Esimerkiksi ympäristön visuaalinen havainnoiminen ja päätösten tekeminen on perinteisillä keinoilla mahdotonta koneille. Tähän ratkaisuna voidaan käyttää tekoälyä ja täten automatisoida prosesseja, jotka vaativat esimerkiksi esineiden tunnistusta ja päätöksen tekoa. Aikaisemmin tällaista suorituskykyä on ollut mahdollista odottaa ainoastaan ihmiseltä. Tekoälyjärjestelmät pystyvät jo nykyään tehtäviin, jotka ovat ihmisten kykyjen saavuttamattomissa, mutta niitä käytetään myös ihmisten työskentelyn tukena [1]. Tekoälyn soveltamisen ansiosta koneiden kyvyt, tehokkuus ja tarkkuus ovat parantuneet huomattavasti, mutta kehittymisen varaa vielä on [2].

Konenäkö on nykyään yksi tärkeimmistä tekoälyyn liittyvistä alalajeista ja sen kehittämiseen on käytetty ja käytetään edelleen suuria määriä resursseja. Konenäön kehityksen tavoitteena on biologisen eliön näön simulointi ja tällä tavoin ymmärryksen luominen kuvasta. Tämän toteuttaminen on hyvin vaativaa ja monet konenäkösovellukset liittyvät vain tiettyjen asioiden tunnistamiseen kuvasta.

Konenäköä hyödynnetään useilla arkielämän osa-alueilla ja teollisuudessa. Monet sovellukset olisivat perinteisillä toteutustavoilla mahdottomia. Teollisuudessa konenäköä käytetään koneiden tekemien prosessien parantamiseen, ihmisten toiminnan helpottamiseen ja aikaisemmin ihmisten tekemien vaiheiden automatisointiin [3].

Arkipäivän käyttökohteita konenäölle ja esineen tunnistukselle ovat esimerkiksi älypuhelimissa ja tietokoneissa käytetty lukituksen avaus kasvojen tunnistuksella ja itseajo-ominaisuudet moderneissa autoissa, kuten automaattinen jarrutus, pysäköinti ja kaistavahti sekä itseajavien autojen autopilotti. Viime vuosina konenäkö ja kasvojentunnistus on noussut esille myös joukkovalvonnassa, joka erityisesti Kiinassa on edennyt erittäin kattavaksi. Esimerkiksi alkuvuodesta 2020 koronaviruksen leviämisen estämiseksi Kiinassa julkaistiin puhelinsovellus nimeltään "close contact detector". Sovellus pystyi varoittamaan käyttäjiä tartuntariskistä, jos he olivat olleet tartunnan saaneiden ihmisten lähettyvillä [4, 5].

Yksi konenäön käyttökohteista on robotit ja androidit. Androidit ovat robotteja, jotka on luotu muistuttamaan ihmistä sekä ulkonäöltään, että käytökseltään. Tässä työssä esitetään yksi toteutus robotin näköjärjestelmästä, jolla pyritään reaaliaikaisesta kuvasta saatujen tietojen perusteella tuottamaan luonnollisen oloiset silmien liikkeet. Tämä on tärkeää, sillä Takashi Minaton tekemässä tutkimuksessa havaittiin, että ihmiset kokivat androidien silmien liikkeet helpommin häiritseväksi kuin esimerkiksi suun liikkeet [6].

Tässä työssä tutkittiin kasvojen paikallistamista ja tunnistamista reaaliaikaisesta kameran kuvasta käyttäen konenäköä. Tätä tietoa käytettiin robotin silmien ohjaamiseen, joista toiseen kamera on sijoitettu. Kasvotietoja säilöttiin väliaikaisesti aikaisempien henkilöiden tunnistamiseen uudelleen.

2. TAUSTA

Tässä luvussa esitellään työhön liittyviä käsitteitä, teknologioita ja ohjelmistoja erityisesti tekoälyyn ja kasvojentunnistukseen liittyen. Tekoälyyn kuuluvia keskeisiä käsitteitä ovat neuroverkot ja syväoppiminen, joita hyödynnetään esimerkiksi puheen-, esineen- ja kasvojentunnistuksessa. Neuroverkkojen idea pohjautuu biologisiin hermoverkkoihin ja termillä syväoppiminen tarkoitetaan monikerroksisia neuroverkkoja, joilla voidaan saavuttaa erittäin monimutkaisia toimintoja suorittavia tekoälysovelluksia. Muita tärkeitä aihealueeseen liittyviä käsitteitä ovat autoenkooderi ja kaskadiluokittelijat, joiden avulla erilaiset tunnistusalgoritmit toimivat. Autoenkooderia sekä kaskadiluokittelijaa käytetään datasta yhtäläisyyksien löytämiseen esimerkiksi kasvojentunnistuksessa.

Tekoälykehitykseen on luotu useita vapaasti käytössä olevia työkaluja, kuten OpenCV, jota tässä työssä käytettiin kasvojentunnistukseen sekä kasvojen seuraamiseen. OpenCV hyödyntää kohteiden tunnistuksessa Haar-ominaisuuksiin perustuvia kaskadiluokittelijoita.

Robotin eri komponenttien kehittämiseen käytettiin Robot Operating System-järjestelmää, joka mahdollistaa useiden eri toimintoja tarjoavien komponenttien kehityksen ja yhdistämisen toimivaksi kokonaisuudeksi.

Tekoälyyn ja erityisesti kasvojentunnistukseen liittyy lisäksi laissa säädettyjä rajoitteita koskien erilaisen datan keräämistä ihmisistä. Tämä data voi sisältää esimerkiksi kuvia, videoita, tekstiä tai äänitallenteita, joiden keräämisen katsotaan rikkovan ihmisen yksityisyyden suojaa. Nämä seikat on syytä huomioida sovelluksissa, jotka vaativat kyseisten tallenteiden hyödyntämistä ja säilömistä. [7]

2.1. Robotiikka

Robotiikka on robottien tutkimukseen ja kehittämiseen keskittynyt tekniikan ala. Sen avulla kehitetään robotteja eri tarkoituksiin, kuten esimerkiksi teollisuudessa tuotantoon ja testaamiseen. Erilaisia robotteja on yksinkertaisista teollisuuden robottikäsistä aina autonomisiin tekoälyä soveltaviin ja ihmistä muistuttaviin androideihin asti.

Robotit ovat koneita, jotka pystyvät suorittamaan automaattisesti monimutkaisia tehtäviä. Robotit koostuvat aktuaattoreista eli toimilaitteista ja sensoreista eli ilmaisimista. Toimilaitteiden avulla robotti pystyy toimimaan fyysisessä maailmassa. Toimilaitetta ovat esimerkiksi moottorit, joiden avulla robotti voi liikkua. Sensoreiden avulla robotit keräävät ympäristöstään tietoa, jonka avulla ne voivat tehdä päätöksiä [8]. Esimerkiksi tässä työssä esitelty robottipää koostuu kamerasta ja servomoottoreista. Kameran tuottamasta videosta saadaan tietoa ympäristöstä, jonka avulla robotti kykenee automaattisesti kääntämään katsettaan halutulla tavalla.

Robotteja, jotka käyttäytyvät kuin ihmiset ja näyttävät ihmiseltä, sanotaan androideiksi. Jotta ihmiset kokevat kanssakäymisen androidin kanssa mieluisaksi, sen käytöksen on oltava sopiva sekä tilanteeseen että sen ulkonäköön. Esimerkiksi, jos androidi on ulkonäöltään hyvin realistisen ihmisen näköinen, mutta sen käytös on hyvin yksinkertaista ja konemaista, androidi koetaan oudoksi ja häiritseväksi. Takashi Minaton tutkimuksen mukaan ihmiset kiinnittävät erityisesti huomiota androidien

silmien käytökseen, joten robottipäätä kehittäessä on syytä harkita, kuinka robotin silmien tulisi käyttäytyä. [6]

Työssä käytetty robotin pää perustuu InMoov-projektiin, joka on ensimmäinen avoimen lähdekoodin 3D-tulostettava ihmisen kokoinen robotti. InMoov on ranskalaisen Gael Langevinin henkilökohtainen projekti joka alkoi tammikuussa 2012 ensimmäisestä avoimen lähdekoodin prosteesikädestä ja on myöhemmin johtanut muihin projekteihin, kuten Bionico. [9]

2.2. Silmien käyttäytyminen

Ihmiset vaihtavat katseen kohdetta niin sanotuilla sakkadisilla silmänliikkeillä. Sakkadisille silmänliikkeille tyypillistä on nopea kiihdytys liikkeen alkaessa ja pysähtyessä, jopa 40 000 deg/s^2. Sakkadisten silmänliikkeiden huippunopeus vaihtelee välillä 400 - 600 deg/s ja liike kestää yleensä noin 30 - 120 ms. Pituudeltaan sakkadinen liike on 1 - 40 astetta, ja siihen sisältyy tyypillisesti päänliikettä, mikäli liike on suurempi kuin 30 astetta. Jos sakkadinen liike tapahtuu reaktiona johonkin tapahtumaan, liikkeen alkuun on tapahtuman jälkeen noin 100 - 300 ms viive. [10]

Ihmiset pyrkivät pitämään katseensa tarkasti kohteessa niin sanotuilla mikrosakkadiliikkeillä. Mikrosakkadiliikkeet ovat suuruudeltaan asteen luokkaa ja tapahtuvat jopa 30 ms välein.

Ihmiset seuraavat hitaasti liikkuvia kohteita tasaisella silmänliikkeellä, jonka nopeus vaihtelee välillä 1-30 deg/s. Nämä silmänliikkeet ovat nopeudeltaan ja kiihtyvyydeltään rajoittuneita, eivätkä pysty sakkadimaisiin nopeuksiin. Tasaiset seurantaliikkeet vaativat yleensä liikkuvan katselun kohteen, jotta ne voidaan toteuttaa. Samanlaista liikehdintää käytetään myös kompensoimaan pään liikettä, kun ihminen katsoo jotain kohdetta. Katse pysyy siis silloin kohteessa pään liikkeistä huolimatta. [10]

2.3. Tekoäly

1800-luvulla teollisen vallankumouksen aikana koneita valmistettiin fyysisen työn vähentämiseksi. 1900-luvulla tietotekniikan edistyessä alettiin tuottaa laitteita, joita voitiin hyödyntää normaalisti aivotyötä vaativiin tehtäviin, mistä alkoi tekoälyn kehitys. Nykyään eletään tekoälyn aikakaudella, jossa kaikkialta löytyy tekoälyn sovelluskohteita, kuten internet, pilvilaskenta ja jakamispalvelut. [11]

1950-luvulla Alan Turing esitteli testin, jonka tarkoituksena on selvittää, onko koneella ihmisen älykkyyttä. Testi tunnetaan laajalti Turingin testinä. Testin määritelmien mukaan, jos koneen reaktiot ja toimet ovat samat kuin ihmisen, koneela ajatellaan olevan tietoisuutta ja älykkyyttä. Kokeessa ihmishaastattelija kysyy kysymyksiä sekä koneelta että ihmiseltä. Tietyn aikamäärän sisällä haastattelijan tulee päätellä mitkä vastaukset ovat peräisin koneelta, ja mitkä ihmiseltä. Sarjalla tällaisia testejä määritettiin koneen älykkyyden taso [11, 12, 8].

Nykyään tekoälyä hyödyntäviä sovelluksia on kaikkialla internetistä älykkäisiin rakennuksiin, ja tekoälyyn liittyviä tutkimuksia tehdään lähes kaikissa yliopistoissa. Tekoälyn kehitys on siitä huolimatta ollut mutkikas matka täynnä kiistoja ja

väärinymmärryksiä. 80-luvun alkupuolella matemaattinen logiikka ja symbolinen järkeily olivat tekoälyn kulmakiviä ja ohjelmointikieliä, kuten LISP ja PROLOG, suosittiin maailmanlaajuisesti. Japanilaisten alulle laittama tekoälyjärjestelmä nimeltään Fifth-Generation Computer Systems oli paljon odotettu edistys alalla, mutta sekään ei kyennyt kommunikoimaan ihmisten kanssa luonnollisella tavalla tai simuloimaan ihmisen tajuntaa [11].

2.4. Neuroverkot

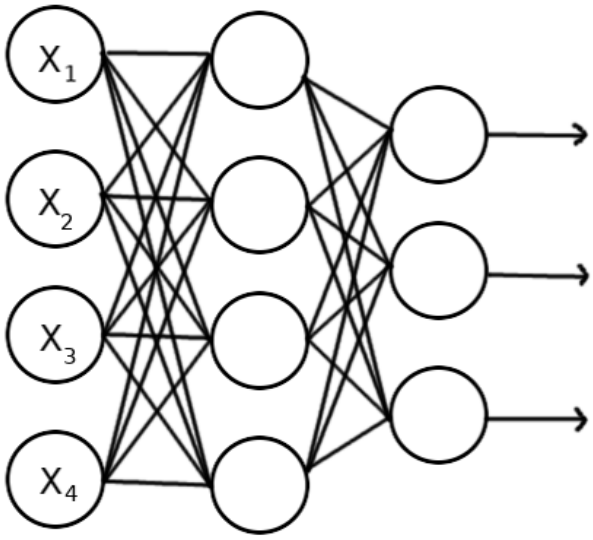
Tekoälyn kehitys kiihtyi myös neuroverkkotutkimuksien johdosta. 1940-luvun alkupuolella esitettiin ensimmäinen keinotekoinen neuroverkkometodi (ANN). Vuonna 1986 David E. Rumelhart ja kumppanit esittelivät *Back-Propagation*-algoritmin eli takaisinkytkentäalgoritmin monikerrosverkkoihin, joista tuli neuroverkkojen kulmakivi [11]. Takaisinkytkennän avulla neuroverkko oppii yhdistämään tietyt syötteet tiettyihin vasteisiin. Algoritmin ideana on säätää neuroverkon kytkentöjä iteratiivisesti, kunnes tietyllä syötteellä päästään mahdollisimman lähelle haluttua vastetta. Algoritmin alussa neuroverkon kytkennöille ja syötteille annetaan satunnaiset arvot. Syöte ajetaan verkon läpi ja verrataan kuinka kaukana saavutettu vaste oli halutusta vasteesta. Tämän eron perusteella muodostetaan virhevektori, joka ajetaan verkossa takaisin päin ja korjataan kytkentöjen arvoja virhevektorin perusteella. Tätä toistetaan niin kauan, että jokainen syöte tuottaa vasteen, joka on mahdollisimman lähellä haluttua vastetta. [13, 14, 15, 8]

Neuroverkot löytävät kouluttamisessa yhteyksiä, joita sääntöpohjaisella ohjelmoinnilla ei välttämättä huomata ottaa huomioon. Tämä tekee niistä hyödyllisiä, mutta haittapuolena on, että verkkoon muodostuneita sääntöjä on vaikea ymmärtää muuttamatta niitä erilaisiin muotoihin [16]. Tätä verkon ominaisuutta kutsutaan selitettävyydeksi ja se on tärkeä esimerkiksi lakiasioissa, koska Euroopan Unionin GDPR säätää, että järjestelmien päätökset täytyvät olla selitettävissä ja perusteltavissa. Mikäli verkon toiminta täytyy perustella, verkosta täytyy saada luotettavasti pääteltyä tapa, jolla tulokseen päädyttiin.

Neuroverkot ovat tapa esittää epälineaarisia hypoteeseja dataan sopivilla parametreilla. Neuroverkon neuronit ovat laskentayksiköitä, joilla on sisääntuloja, ulostuloja ja aktivointifunktio. Neuroni valitsee ulostulon sisääntulojen perusteella aktivointifunktion määräämällä tavalla. Neuroverkot muodostuvat useiden yksittäisten neuronien kytkennöistä. Esimerkiksi kolmekerroksinen neuroverkko voi koostua syötekerroksesta, piilotetusta kerroksesta ja vaste-kerroksesta. Neuroverkon kerroksia sanotaan piilotetuiksi, jos niiden arvoja ei käytetä opetuksessa [17, 13, 14]. Oheisessa kuvassa 1 on esitetty yksinkertainen kolmekerroksinen neuroverkko, jossa vasemmalla on syötekerros, keskellä piilotettu kerros ja oikealla vastekerros.

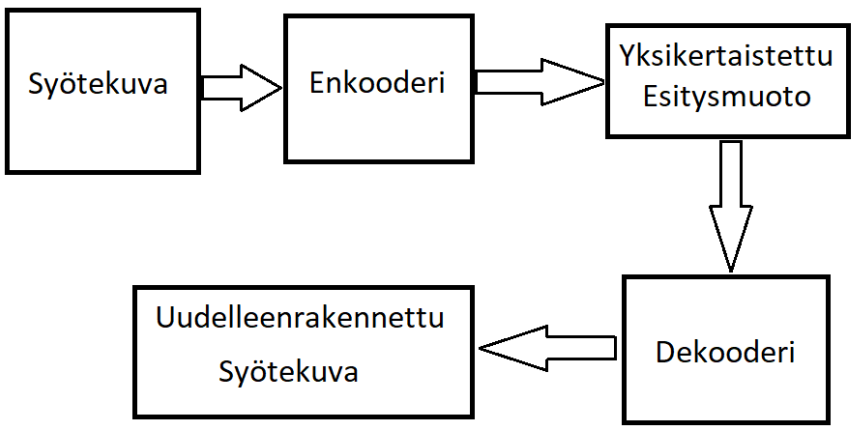
2.4.1. Autoenkooderi

Yksi tekoälyn tehokkaimpia työkaluja on valvottu oppiminen, jota on hyödynnetty esimerkiksi konenäkö- ja puheentunnistussovelluksissa. Valvotun oppimisen hyödyntäminen oli ennen varsin työlästä monissa sovelluksissa, sillä se vaati



Kuva 1. Yksinkertainen neuroverkko

tunnistettavien piirteiden manuaalisen määrittämisen algoritmille. Mallin piirteiden manuaaliseen määrittämiseen on aiemmin käytetty paljon aikaa, mutta nykyään piirteiden automaattiseen oppimiseen on olemassa työkalu, autoenkooderi [17]. Autoenkooderi on neuroverkko, jota käytetään vähentämään syötetyn tiedon ulottuvuuksia ja muodostamaan siitä tällä tavalla yksinkertaisempi havainnollistus. Toiminta on havainnollistettu kuvassa 2. Autoenkooderi koulutetaan muodostamaan syötetylle tiedolle mahdollisimman hyvä vastine käyttäen haluttua määrää vektoreita [18]. Tätä yksinkertaistettua muotoa voidaan käyttää esimerkiksi kasvontunnistuksessa vertailemaan löydettyjä kasvoja tietokantaan [19] [20].



Kuva 2. Autoenkooderin toiminta

2.4.2. Syväoppiminen

Syväoppiminen on tekoälyn alalaji, jolla toteutetaan suuria, tarkkaan data-analyysiin pystyviä neuroverkkoja. Syväoppiminen on suuresta määrästä dataa löydettävien yhteyksien avulla annetuista syötteistä päätösten muodostamista [21, 15]. Syväoppimisessa suuri neuroverkko siis löytää syötetystä materiaalista yhteyksiä ja antaa sen perusteella tuloksia. Neuroverkot koulutetaan käyttäen testidataa, joka sisältää syötteen sekä sitä vastaavat odotetut tulosteet. Koulutuksessa neuroverkko saa syötteen ja antaa tulosteen, jota verrataan odotettuun tulosteeseen. Neuroverkko pisteytetään sen mukaan, kuinka hyvin sen antamat tulosteet vastaavat odotettuja tulosteita. Pisteytyksen jälkeen neuroverkko muuttaa itseään asetettujen painotusten mukaan suuntaan, jolla saataisiin oikeat tulosteet. Neuroverkot ovat näin ollen myös verrattavissa biologiseen oppimiseen, joka itseasiassa onkin niiden toiminnan lähtökohta.

2.5. Konenäkö

Konenäöllä tarkoitetaan tiedon muuntamista kuvasta tai videosta, joko päätökseen tai uuteen esitystapaan. Koska näkeminen on ihmisille luonnollista, on helppoa ajatella, että konenäön tehtävät olisivat yksinkertaisia, mutta ongelma on itseasiassa varsin haasteellinen. Jopa ihminen on helposti harhautuva esimerkiksi väärän keskittymisen kohteen takia. Konenäköjärjestelmissä tietokone vastaanottaa numeromatriiseja kameralta ilman sen suurempia automaatioita, kuten kuvan tarkennusta. Kaksiulotteisesta näkökulmasta on erittäin haasteellista tunnistaa kolmiulotteisia kohteita, sillä eri kulmasta sama kohde voi olla täysin erinäköinen. Käytännön järjestelmissä tieto asiayhteydestä on yleensä erittäin hyödyllistä. Esimerkiksi etsittäessä jalkapalloa sisätiloissa, sitä ei tarvitse etsiä mahdottomista paikoista, kuten katosta. [22]

Konenäköä voidaan soveltaa esimerkiksi hedelmäpuussa kasvavien hedelmien laskemiseen. A. Paynen artikkelin mukaan 2000-luvun alussa hedelmiä havaittiin konenäöllä pääasiassa kahdella tavalla: joko etsimällä kuvasta pyöreitä ja soikeita muotoja tai tietyllä alueella pikseleiden värin, intensiteetin ja syvyyden perusteella. Nykyään hedelmiä havaitaan sekä värin että muodon avulla luokitellen, mikä on havaittu tehokkaammaksi kuin vain yhden piirteen perusteella luokittelu. Valvotun oppimisen kuvan prosessointimenetelmien on huomattu antavan parempia tuloksia kuin yksinkertaisempien menetelmien, mutta ne vaativat korkeamman laskentatehon ja tarkan opetusmateriaalin. Artikkelin mukaan hedelmien havaitsemisessa suurimpia haasteita ovat muutokset valaistuksessa ja hedelmien kasaantuminen [23].

Kuvan hankinnassa tärkeitä seikkoja ovat valaistus, kamera, kameran linssi ja tietokoneen ja kameran rajapinta. Valaistuksella huolehditaan, että kuvattavan kohteen tärkeät piirteet saadaan selvästi näkyviin. Linsseillä tuotetaan terävä kuva kameralle ja voidaan säätää kuvan ominaisuuksia, kuten näkökentän kokoa. Kamera muuntaa kuvan videosignaaliksi ja tietokoneen ja kameran rajapinnassa esimerkiksi USB-väylällä tietokone ottaa vastaan videosignaalin ja muuntaa sen kuvaksi tietokoneen muistiin [3].

2.6. Kasvojentunnistus

Turvallisuuteen liittyvät huolenaiheet, tarve henkilöllisyyden varmentamiseen sekä kiinnostus kasvojen analyysiin ja mallintamiseen ovat viime aikoina lisänneet kiinnostusta kasvojentunnistukseen. Automaattista kasvontunnistusta on tehty 1960-luvulta asti, mutta kaikkia ongelmia ei ole vieläkään ratkaistu. Automaattisella kasvontunnistuksella on monia käyttökohteita, esimerkiksi turvallisuus- ja viihdesektoreilla. Kasvontunnistuksessa pyritään kaksiulotteisesta kuvasta tunnistamaan kolmiulotteiset kasvot, jotka ovat alttiina valoisuuden, asennon ja ilmeen muunnoksille. Kasvojentunnistusjärjestelmä koostuu yleensä neljästä komponentista: kasvojen havaitseminen, kohdistus, piirteiden erittely ja vastaavuuksien löytäminen. Kasvojenhavaitsemiskomponentti erottaa kasvot kuvan taustasta. Jos kasvoja etsitään videokuvasta, saatetaan tarvita erillinen komponentti kasvojen jäljittämiseen. Kasvojenkohdistuskomponentilla pyritään saamaan tarkempi tieto kasvojen sijainnista ja koosta kuin havaitsemisvaiheessa. Kasvojen piirteet, kuten suu, nenä ja silmät paikannetaan ja normalisoidaan geometristen ominaisuuksien suhteen. Tämän jälkeen kerätään tiedot hyödyllisistä kasvonpiirteistä, joilla voidaan erottaa kasvot toisistaan. Lopuksi on mahdollista verrata saatuja kasvonpiirretietoja tietopankissa olevaan dataan ja yrittää löytää vastaavuuksia [24].

Kasvontunnistuksessa on edistytty viime vuosina konvoluutioneuroverkkojen (CNN) ansiosta. Kasvojentunnistus jaetaan tyypillisesti kasvojentunnistukseen ja -verifiointiin. Tunnistuksella tarkoitetaan kasvojen yhdistämistä tiettyyn henkilöllisyyteen ja verifioinnilla varmistetaan, että kasvot kuuluvat tietylle henkilölle.

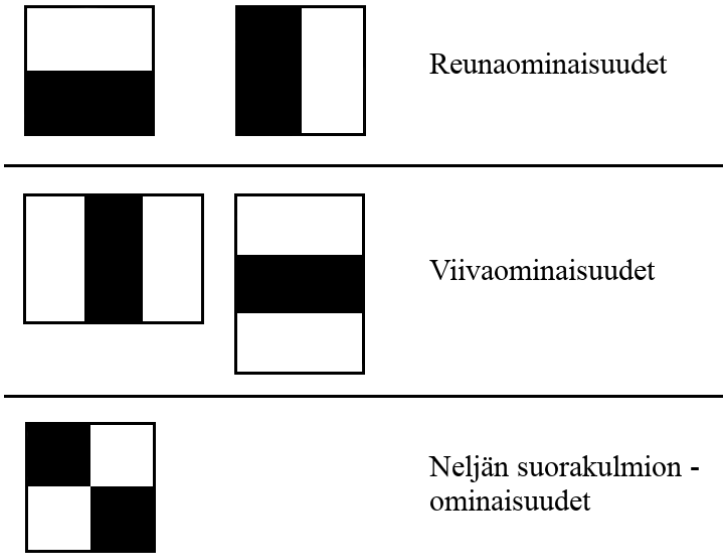
Kasvojen tunnistuksen testauksessa voidaan käyttää joko avoimen tai suljetun aineiston testausta. Suljetun aineiston testauksessa kaikki testihenkilöt on aiemmin määritelty opetusmateriaalissa. Suljetun aineiston testaus on siis käytännössä luokitteluongelma. Avoimen aineiston testauksessa testihenkilöt ovat täysin toiset kuin mitä käytettiin opetukseen, mikä on tunnistuksen kannalta haasteellisempaa, mutta myös lähempänä käytännön tilannetta. Avoimen aineiston testauksessa pyritään oppimaan selkeitä kasvonpiirteitä. Se on metriikan oppimisongelma, jossa pyritään mittamaan samankaltaisuutta [25, 26].

Viime vuosina avoimen aineiston tunnistusta on pyritty kehittämään asetelmiin, joissa on tarve hylätä tuntemattomia kohteita testauksen aikana. Vaikka neuroverkkoja on mahdollista kouluttaa luokittelemaan epäkiinnostavat kohteet esimerkiksi luokkaan muut, on mahdotonta kouluttaa neuroverkko tunnistamaan kaikki mahdolliset kohteet. Siksi on syytä kehittää työkaluja tunnistukseen, jossa otetaan huomioon aidosti tuntemattomat kohteet [27].

2.6.1. Haar-ominaisuudet ja -luokittelija

Nopeassa esineentunnistuksessa, ja täten myös kasvojentunnistuksessa, käytetään hyväksi Paul Violan ja Michael Jonesin kehittämää ns. Haar-ominaisuuksiin perustuvaa algoritmia. Nämä Haar-ominaisuudet ovat hyvin yksinkertaisia kahdesta, kolmesta tai neljästä suorakulmiosta muodostuvia ikkunoita. Käytännössä kuvasta ominaisuuksien löytäminen tapahtuu käymällä kuva läpi käyttämällä sopivan kokoista havaintoikkunaa, jossa Haar-ominaisuuksia etsitään. Mikäli tämän havaintoikkunan

sisällä löydetään halutun kohteen kanssa vastaavat ominaisuudet, luokitellaan tämä osa kuvasta kyseiseksi kohteeksi. Kuvassa 3 on esitetty erilaiset Haar-ominaisuudet.

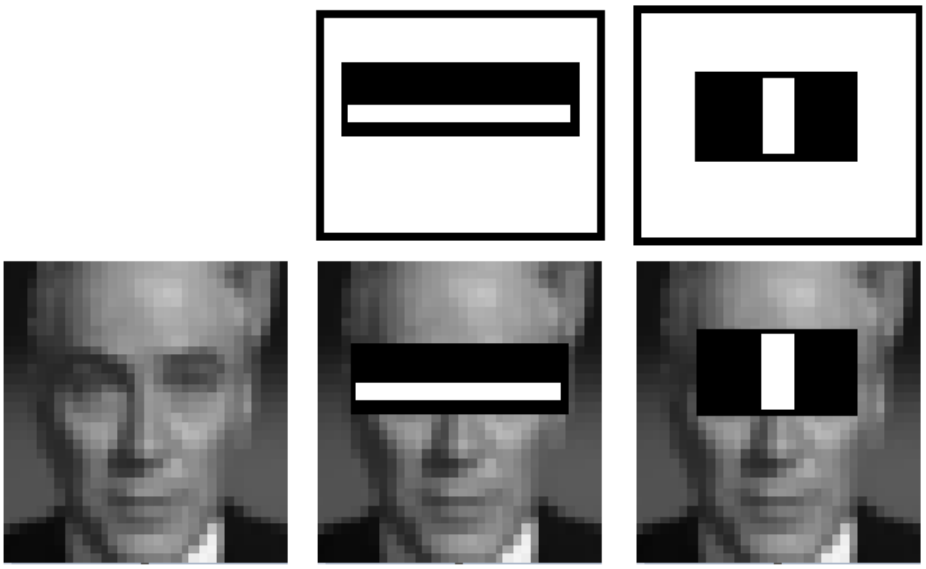


Kuva 3. Haar-ominaisuudet

Kahden suorakulmion ominaisuudet eli reunaominaisuudet löytyvät vähentämällä valkoisen suorakulmion sisälle jäävien pikseliarvojen keskiarvo mustan suorakulmion sisälle jäävien keskiarvosta. Mitä suurempi erotus on, sitä todennäköisemmin kuvan osa vastaa kyseistä ominaisuutta. Samaan tapaan viivaominaisuuksissa lasketaan uloimpien suorakulmioiden sisälle jäävien pikseleiden keskiarvot ja vähennetään keskimmäisen sisälle jäävien keskiarvosta. Neljän suorakulmion ominaisuudet taas löytyvät kahden diagonaalin erotuksella [28].

Kasvojentunnistuksessa hyödynnettäviä Haar-ominaisuuksia ovat esimerkiksi silmien ja otsan raja (reunaominaisuus), silmien väli (viivaominaisuus) sekä suun ja leuan raja (reunaominaisuus). Oheisessa kuvassa 4 on havainnollistettu, kuinka nämä kyseiset kasvonpiirteet vastaavat eri Haar-ominaisuuksia. Kasvojen kuvat on skaalattu 32×32 pikselin kokoisiksi käsittelyä varten. Kasvot ovat peräisin tekijänoikeudettomasta elokuvasta *Plan 9 From Outer Space*.

Algoritmi käy havaintoikkunan sisällä läpi mahdolliset ominaisuudet aloittaen reunaominaisuuksista. Jos havaintoikkunan sisältä ei löydy kyseistä ominaisuutta, algoritmin mukaan se ei myöskään sisällä kasvoja. Täten havaintoikkunaa voidaan siirtää kuvan sisällä ja aloittaa etsiminen uudestaan, kunnes reunaominaisuuksia löydetään. Tämän jälkeen etsitään mahdolliset viivaominaisuudet. Mikäli niitä ei löydetä, siirretään havaintoikkunaa jälleen ja aloitetaan etsiminen edellä mainitulla tavalla. Kun kaikki tarvittavat Haar-ominaisuudet ovat löytyneet havaintoikkunan sisältä, tulkitaan tämä kohta kuvasta kasvoiksi. Tunnistuksen tarkkuus riippuu luokittelijan (Cascade Classifier) tarkkuudesta.



Kuva 4. Kasvoista löytyviä Haar-ominaisuuksia (kasvot tekijänoikeudettomasta elokuvasta *Plan 9 From Outer Space*)

2.6.2. Integraalikuvat

Haar-ominaisuuksien löytäminen vaatii suuria määriä yhteenlaskuja ja jo 24×24 kokoisen havaintoikkunan sisällä etsittäviä ominaisuuksia voi olla jopa yli 200 000. Yhteenlaskujen helpottamiseksi tehdään havaittavasta kuvasta ns. integraalikuva, jossa jokainen pikseli sisältää alkuperäisen kuvan vastaavan pikselin vasemmalla ja yläpuolella olevien pikseliarvojen ja näistä muodostuvan suorakulmion sisälle jäävien pikseliarvojen summan [28, 29].

5	8	5	3	2	6
2	5	8	8	4	1
5	7	7	1	9	5
7	8	1	3	5	9
4	4	2	5	7	9
7	4	1	2	3	4

5	13	18	21	23	29
7	20	33	44	50	57
12	32	52	64	79	91
19	47	68	83	103	124
23	55	78	98	125	155
30	66	90	112	142	176

Kuva 5. Alkuperäinen kuva (vas) ja sen integraalikuva (oik)

Kuvassa 5 on esitetty satunnaisia pikseliarvoja sisältävä kuva ja näistä muodostettu integraalikuva. Jos esimerkiksi halutaan laskea punaisen suorakulmion sisälle jäävien pikseleiden summa, voidaan se laskea käyttämällä integraalikuvassa vihreällä merkittyjä summia. Tällöin lasku on:

$$125 + 33 - 50 - 78 = 30$$

joka vastaa kaikkia yksittäisiä yhteen laskettuja suorakulmion sisälle jääviä pikseliarvoja:

$$1 + 9 + 3 + 5 + 5 + 7 = 30$$

Tällä tavoin yhden suorakulmion tapauksessa laskutoimituksien määrä saadaan pudotettua kolmeen sen sijaan, että jokainen pikseliarvo laskettaisiin erikseen yhteen, jolloin laskutoimituksia voisi olla isojen kuvien tapauksessa satoja.

2.7. OpenCV ja Tensorflow

Nykyään konenäön ja muiden tekoälysovellusten kehittämistä ja toteuttamista on helpotettu julkisesti saatavilla olevilla työkaluilla, kuten Googlen kehittämä Tensorflow ja alun perin Intelin kehittämä OpenCV. OpenCV on avoimen lähdekoodin konenäkötyökalu, jonka alkuperäisenä tarkoituksena oli nopeuttaa konenäön ja tekoälyn kehitystä tarjoamalla hyvä alusta alalla työskenteleville. Sitä on mahdollista ohjelmoida käyttäen C, C++, Python, sekä Java-ohjelmointikieliä. OpenCV:n tavoite on laskennallinen tehokkuus ja reaaliaikaiset sovellukset [22].

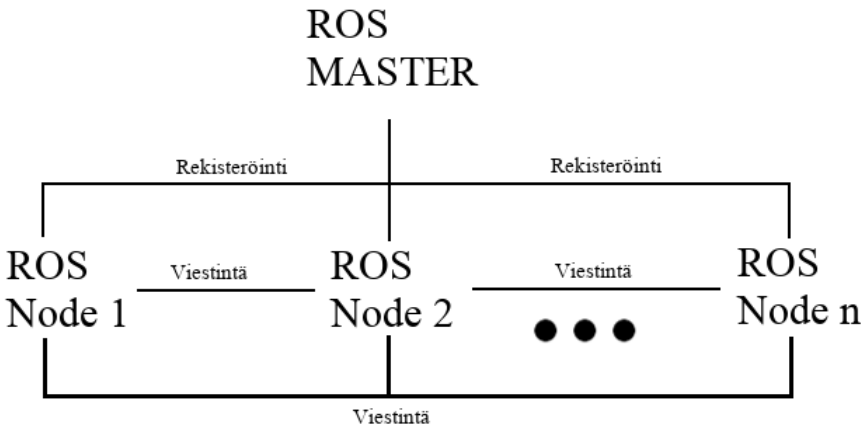
Intel Research aloitti vuonna 1999 avoimen standardin luomisen konenäön kehityksen edistämiseksi. Vuonna 2012 tämän kehittämistä jatkoi OpenCV Foundation. OpenCV on alun perin kirjoitettu C++-ohjelmointikielellä, mutta se toimii myös Python-ohjelmointikiellä, jota tässä työssä käytettiin. Pythonille asennettuna OpenCV vaatii NumPy-kirjaston, joka sisältää numeerisia työkaluja, kuten funktioita ja erilaisia taulukoita. Itsessään OpenCV sisältää myös yli 500 kirjastoa [30]. OpenCV:n mukana tulee valmiiksi mallinnettuja Haar-ominaisuuksiin sekä LBP-malleihin pohjautuvia kaskadiluokittelijoita, joiden avulla voidaan tunnistaa, esimerkiksi kasvoja, kehoja tai rekisterikilpiä. Lisäksi OpenCV on yhteensopiva Googlen kehittämän Tensorflowin kanssa, joten sitä voidaan hyödyntää syväoppimisessa, esimerkiksi mallintamalla erilaisia luokittelijoita.

Tensorflow on Googlen vuonna 2015 julkaisema koneoppimiskirjasto, joka on luotu helpottamaan mallien tekemistä korkean tason käyttöliittymillä. Tässä työssä henkilöntunnistamiseen käytetyn neuroverkkoratkaisun toteutuksessa käytettiin Tensorflown Keras käyttöliittymää. Tensorflow helpotti kehitystä esimerkiksi yksinkertaistamalla näytönohjaimen tai prosessorin tehokasta käyttöä neuroverkon kouluttamiseen ja tällä tavalla nopeutti prosessia [31, 32].

2.8. Robot Operating System

Robot Operating System (ROS) on avoimen lähdekoodin järjestelmä, jonka avulla tietokoneella voidaan ohjata robotin eri osia. ROS-järjestelmässä kommunikaatio eri osien välillä perustuu tilaus- ja julkaisujärjestelmään, jossa komponentit lähettävät tietoa toisille komponenteille näiden toimiessa tilaajina kyseisille tiedoille. Viestit kulkevat aihetunniste (engl. topic) tietoväylässä, joka perustuu nimettömään tilaus- ja julkaisujärjestelmään. Järjestelmä on siinä mielessä nimetön, että lähettävät ja vastaanottavat komponentit eivät tiedä, mihin tieto lähetetään, tai mistä se on peräisin, vaan viestien ohjauksesta huolehtii isäntäkomponentti. ROS-järjestelmässä isäntäkomponentti (engl. master component), johon muut komponentit ovat rekisteröityneet, vastaa liikenteestä muiden komponenttien välillä. Isäntäkomponentti pitää kirjaa, mitkä komponentit ovat tietyn aiheen tilaajia, ja pystyy tilaustietojen

perusteella kertomaan lähettäjille, mille komponenteille tieto täytyy lähettää [33, 34]. Kuvassa 6 havainnollistettu yksinkertainen ROS-verkko.



Kuva 6. ROS-rakenne-esimerkki

Yksinkertaisten viestien lisäksi jokin komponentti voi olla palveluntarjoaja, esimerkiksi numeroiden yhteenlaskua varten. Muut komponentit voivat lähettää palveluntarjoajalle viestejä, joihin ne saavat vastauksen. Esimerkiksi lähettämällä kokonaislukuja yhteenlaskupalvelulle palvelu lähettää takaisin numeroiden summan.

ROS tarjoaa joustavuutta, sillä kaikkien komponenttien ei tarvitse olla samanlaisia laitteita, kuten tietokoneita, eikä laitteiden tarvitse olla edes samasta arkkitehtuurista. ROS tarjoaa alustan, jossa komponenttien verkkoon voi kuulua useita erilasia laitteita, esimerkiksi tietokone, älypuhelin ja Arduino [33].

2.9. Tietosuoja

Vuoden 2018 toukokuussa voimaan tullut Euroopan Unionin säätämä General Data Protection Regulation (GDPR) säätää uusia sääntöjä ihmisten yksityisten tietojen käytöstä ja säilömisestä kaikissa järjestelmissä, joita jäsenmaiden kansalaiset käyttävät. Tietokannoista, joihin kerätään ihmisten henkilökohtaisia tietoja, täytyy ilmoittaa kerättävän tiedon sisältö, ja antaa vaadittaessa käyttäjän hankkia kaikki hänestä kerätty tieto itselleen. Käyttäjällä on myös oikeus pyytää poistamaan kaikki hänestä kerätty tieto järjestelmästä [7]. Tämä säädös täytyy ottaa huomioon, kun käytetään kasvontunnistusta, joka tallettaa tunnistustietonsa tietokantaan.

Tässä työssä tallennetaan kasvotietoja väliaikaisesti välimuistissa pidettyyn jonoon. Järjestelmän tallentamista tiedoista täytyy myös ilmoittaa käyttäjille, esimerkiksi kyltillä.

3. TOTEUTUS

Kurssin tavoitteena oli valmistaa ihmismäinen robottipää yhteistyössä muiden ryhmien kanssa Sulautettujen järjestelmien ohjelmistojen projekti -kurssilla. Kukin ryhmä toteutti oman toiminnon kokonaisuutta varten. Ryhmät tekivät toiminnot käyttäen ROS Robot Operating System -ohjelmistokehystä, joka varmistaa osien yhteensopivuuden [34]. Ryhmän tehtävä oli toteuttaa robotin silmät, niiden ohjaus ja niihin liittyvä ohjauslogiikka.

Tässä työssä esitellään toteutus, jossa käytettiin konenäkemiseen robotin silmiin sijoitettua Microsoft LifeCam HD-3000 -kameraa, jota ohjataan kahdella Dynamixel XL-320 -servomoottorilla. Järjestelmän mekaniikan kehitys ja testaus tehtiin käyttäen Arduino Uno -kehitysalustaa.

3.1. Sovellusympäristöt

Tavoitteena oli, että ryhmien yhdessä kehittämä robotti kykenisi luontevaan kanssakäymiseen ihmisen kanssa vähintään minuutin kerrallaan. Järjestelmä muistaa ihmiset, joiden kanssa se on keskustellut päivän aikana, jos siihen on annettu lupa. Päivän jälkeen kasvotiedot pyyhitään muistista. Kasvotietojen ansiosta järjestelmä pystyy paremmin ylläpitämään luontevaa keskustelua ihmisen kanssa, esimerkiksi tilanteessa, jossa keskustelutoveri siirtyy pois järjestelmän näkökentästä. Tällaisissa tilanteissa järjestelmä ei luule henkilöä toiseksi tämän palatessa takaisin näköpiiriin, ja pystyy jatkamaan keskustelua. Järjestelmä pystyy myös muistamaan henkilöt, jotka palaavat pidemmän ajan jälkeen takaisin keskustelemaan järjestelmän kanssa. Kasvotietojen avulla pystytään myös paremmin selviämään tilanteista, joissa ihmiset yrittävät saada järjestelmän epäonnistumaan, esimerkiksi harhauttamalla järjestelmän luulemaan samaa henkilöä toiseksi ja aloittamaan keskustelun alusta. Taulukossa 1 on esitetty muutama käyttötilanne robotin päälle.

Koska robotin toiminnot sisältävät kasvojentunnistuksen ja siten myös henkilötietojen tallennuksen, on syytä ottaa huomioon ihmisten tietoturvaoikeudet (esimerkiksi Euroopan Unionin GDPR). Vaikka kasvotietoja käytetään havaitsemaan, onko kyseistä henkilöä nähty aikaisemmin, ihmisillä on oikeus tietää mitä tietoja heistä tallennetaan ja on tallennettu. Henkilön niin pyytäessä, on hänestä poistettava kaikki hänestä tallennettu tieto järjestelmästä, joka voidaan säädösten mukaan poistaa, ja lisäksi hänellä on oikeus saada kopio kaikista hänen tiedoistaan.

Käytännön tilanteessa robotti voisi esimerkiksi kysyä henkilöltä suostumusta kasvotietojen tallennukseen; ilmoittaen samalla, että robotti ei pysty kanssakäymiseen ilman suostumusta, koska se tarvitsee kasvotietoja toimiakseen suunnitellulla tavalla. Käytännön tilanteessa tämä toiminta tuottaisi tosin haasteita, mikäli keskustelun aikana henkilö haluaa, että hänen kasvotietonsa unohdetaan, alkaisi robotti heti unohdettuaan tunnistamaan samaa henkilöä uutena henkilönä tallentaen kasvotiedot jälleen. Mahdollinen ratkaisu tällaisen tilanteen välttämiseksi olisi pyytää henkilöä poistumaan robotin näkökentästä, minkä jälkeen tiedot voidaan poistaa.

Taulukko 1. Käyttötapaukset

Käyttötapaus	Kuvaus
Tervehtiminen	Tervehtii ohikulkijoita ja houkuttelee vierailijoita pysähtymään. Jos vierailija pysähtyy tutkimaan robottia, robotti voi demonstroida useita toimintojaan.
Keskusteminen	Kykenee selviytymään yksinkertaisesta noin minuutin mittaisesta keskustelusta, kysymään yksinkertaisia kysymyksiä ja vastaamaan niihin. Pystyy myös seuraamaan keskustelua katsomalla aina puhujaan päin. Robotti pystyy myös muistamaan henkilöt, joiden kanssa se on keskustellut, jos se on saanut siihen luvan ja pystyy siten uudelleen tapaamisen sattuessa toimimaan eri tavalla kuin ensitapaamisessa.
Mököttäminen	Välttelee katseita reaktiona esimerkiksi törkeisiin kommentteihin.
Toimeton	Kääntelee katsetta satunnaisiin paikkoihin ja seuraa ohikulkijoita.
Etäkäyttö	Käyttäjä pystyy virtuaalilasien avulla näkemään robotin silmien läpi ja katselemaan ympärilleen. Lisäksi käyttäjälle voi näyttää robotin havaitsemat kasvot ja niiden arvioidun etäisyyden.

3.2. Laitteisto

Pääosa ohjelmiston suorituksesta tapahtuu Raspberry Pi -minitietokoneella, jolla voidaan suorittaa sekä konenäköön, tunnistukseen, että ohjauskomentoihin liittyvä logiikka. Työssä käytetty malli on Raspberry Pi 4 Model B, joka sisältää neliytimisen ARM Cortex-A72 -suorittimen, neljä gigatavua keskusmuistia, kaksi USB 2.0 ja USB 3.0 -porttia, Gigabit Ethernet -portin sekä 2,4/5 GHz 802.11b/g/n/ac langattoman yhteyden [35].

Microsoft LifeCam HD-3000 on tietokoneeseen tarkoitettu USB 2.0 -väylän kautta käytettävä videokamera. HD-3000 kuvaa 720p 16:9 kuvasuhteen videota, eli 1280×720 pikseliä, 30 Hz taajuudella [36]. Kameralta voidaan lukea data OpenCV-kirjastoa hyväksi käyttäen. Videosta otetaan puskuriin talteen kuvia kaksiulotteisena taulukkona, jossa jokainen alkio sisältää pikselin RGB-arvot. Kuvia pidetään puskurissa haluttu määrä, millä varmistetaan jatkuvuus käsittelyssä.

Kameroiden liikutteluun käytetyt XL-320 -servomoottorit ovat niin sanottuja älyservoja (smart servo), jotka käyttävät kommunikointiin asynkronista sarjaliikennettä. Servoilla on 1024 asentoa ja käännös laidasta laitaan on 300 astetta, jolloin yhden askeleen pituus on noin 0,29 astetta. Servojen valmistajan sivulta löytyy ohjauslogiikan muistialueen käsittelyyn tarkat ohjeet, joiden avulla myös Arduinolle on tehty useita kirjastoja helpompaa ja nopeampaa ohjelmointia varten. Ohjausparametrien vastaanottamisen lisäksi XL-320 -servoista löytyy myös ominaisuus lähettää dataa ohjainlaitteelle. Tämä data sisältää esimerkiksi servon asentotiedot, pyörimisnopeuden, kuorman, jännitteen ja lämpötilan. [37]

Servomoottorien hallintaan valittiin Arduinon kehittämä kehitysalusta. Arduino on avoimen lähdekoodin projekti, jossa on kehitetty useita alustoja mikrokontrollereille. Nämä yhdessä projektia varten kehitetyn ohjelmiston kanssa mahdollistavat yksinkertaisen ja helpon prototyyppien valmistamisen [38]. Kehistysalustana käytetty Arduino Uno sisältää Atmelin kehittämän 8-bittisen ATmega328P AVR RISC -pohjaisen mikrokontrollerin, jossa on 32 kilotavua flash-muistia ja 23 yleiskäyttöistä I/O-nastaa [39].

Servomoottoreiden ohjauksen helpottamiseksi on käytettävissä Dynamixelin Arduino-kortti (Arduino shield). Se sisältää valmiit liitännät servomoottoreille ja piirit kaksisuuntaista sarjakommunikointia varten. Arduino-kortin käyttö mahdollisti myös Dynamixelin valmiiden Arduino-kirjastojen käytön.

3.3. Ohjelmisto

Kuvanprosessointi toteutettiin käyttäen Python-ohjelmointikieltä ja OpenCV- ja Tensorflow-kirjastoa. Kuva saadaan kameralta käyttäen OpenCV-kirjaston sisäänrakennettuja funktioita USB web-kameran käyttämiseen. Kuvan analysointi tehtiin käyttäen ensin neuroverkoilla toteutettua Haar-luokittelijaa, joka löytää kuvasta kasvot. Tämän jälkeen kameran kuvasta leikattiin alue, josta kasvot oli tunnistettu. Kasvot enkoodataan vektorimuotoon käyttäen Tensorflow-kirjaston avulla koulutettua autoenkooderia. Saatua vektoria voidaan sitten vertailla aikaisempiin tunnistettuihin kasvoihin ja tunnistaa henkilö.

Kasvojentunnistuksen kaskadiluokittelijana päädyttiin käyttämään OpenCV-kirjaston valmiiksi koulutettua *haarcascade_frontalface_default* [40]. Oman luokittelijan kouluttaminen olisi vaatinut hyvän koulutusmateriaalin, jonka puuttuessa päädyttiin käyttämään valmista ratkaisua.

Autoenkooderin kouluttamiseen tarvittiin luokiteltua kuva-aineistoa ihmisten kasvoista. Materiaali jaettiin koulutus- ja testausmateriaaliin. Kuvista valittiin ainoastaan kohdat, joissa on ihmisten kasvoja. Jotta kuvia voidaan käyttää luotettavasti neuroverkon kouluttamiseen, täytyy ne ensin skaalata samankokoisiksi ja pikseleiden arvot normalisoida. Kuvat muutettiin lisäksi mustavalkoisiksi ja 32×32-kokoisiksi. Materiaalina käytettiin tekijänoikeudettomista elokuvista [41] tunnistettuja kasvoja, jotka tunnistettiin käyttämällä samaa kaskadiluokittelijaa kuin kasvojentunnistuksen toteutuksessa. Tekijänoikeudettomat elokuvat ovat julkista omaisuutta ja kaikkien vapaassa käytössä [42]. Kerätyssä, noin 2500 kasvon aineistossa, samat henkilöt esiintyvät useasti. Kasvoista noin 2350 käytettiin kouluttamiseen ja 150 testaamiseen. Materiaalin kerääminen käyttämällä samaa luokittelijaa kuin toteutuksessa varmistaa samalla koulutuksessa käytettyjen ja käytännössä saatujen kuvien rajaamisen vastaavuuden.

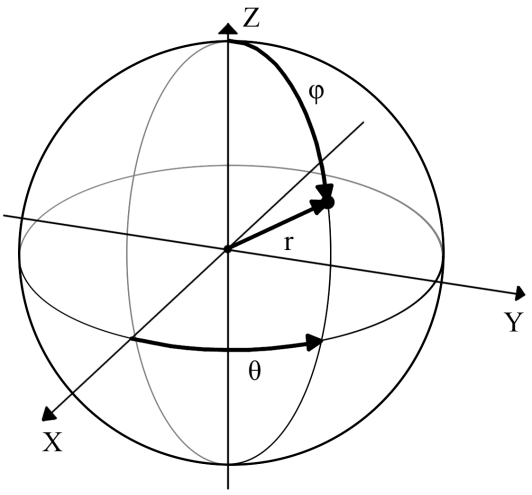
3.4. Toteutuksen teoria

Silmien osoittamiseksi haluttuun kohteeseen täytyy laskea kulma, jolla kameraa täytyy kääntää kahdella akselilla. Kuvassa 8 on esitetty kulman laskeminen kameran kuvan pisteestä suhteuttamalla kohta ensin kuvan leveyteen ja sitten kameran tunnettuun

näkökentän laajuuteen (FoV). Käytetyn Microsoft LifeCam HD3000 diagonaalinen näkökentän laajuus on noin 69 astetta.

Kuvasta voidaan myös arvioida etäisyyttä tunnistettuihin kasvoihin, mikäli tiedetään kuinka leveät ihmisen kasvot keskimäärin ovat. Joseph W. Youngin tekemän tutkimuksen mukaan kasvojen leveys poskiluiden kohdalta mitattuna on keskimäärin 13 senttimetriä naisilla ja 14 senttimetriä miehillä neljän senttimetrin keskihajonnalla [43]. Työssä käytetty kasvojentunnistusalgoritmi rajaa kuvasta hieman kasvoja leveämmän alueen, joten sitä on syytä kompensoida hieman kasvojen leveyden arviossa. Leveyden arviona voidaan näin ollen käyttää 15 senttimetriä ilman, että suuria virheitä etäisyyden arvioinnissa syntyy. Virhettä syntyy enemmän esimerkiksi kasvojen koon arvioinnissa kuvasta. Etäisyyden absoluuttinen tarkkuus ei ole tärkeä tässä käyttötarkoituksessa, koska sitä hyödynnetään ainoastaan yhdessä samalla tavalla kerättyjen havaintojen kanssa.

Pään ja silmien kääntökulmia, havainnon kulmaa suhteessa kameran osoittamaan suuntaan ja havainnon arvioitua etäisyyttä käytetään sijoittamaan havainto pallokoordinaatistoon, jossa kamera sijaitsee origossa. Sijainti pallokoordinaatistossa määritellään sen etäisyyden ja vaaka- ja pystysuuntaisten kiertokulmien avulla. Pallokoordinaatisto on havainnollistettu kuvassa 7.



Kuva 7. Pallokoordinaatisto suhteessa silmiin

Pallokoordinaatisto auttaa pitämään kirjaa ympärillä olevista ihmisistä, yhdistämään päällekkäiset havainnot ja arvioimaan havaintojen realistisuutta. Sijoittamalla havainnot koordinaatistoon, joka ei ole sidottu kameran asentoon, mahdollistetaan useamman kohteen muistaminen yhtäaikaisesti, sekä helpotetaan näiden seuraamista tilassa. Pitämällä yllä tietoa edellisistä havainnoista ja yhdistämällä niihin uusia havaintoja voidaan vähentää uusien identifiointien tarvetta ja tarkentaa arviota sijainnista.

Havainto sijoitetaan koordinaatistoon laskemalla havainnon alueen keskikohdan perusteella kulmat suhteessa kameran suuntaan ja etäisyys kamerasta suhteuttamalla havainnon koko kuvassa oletettuun kokoon ja tiedettyyn näkökentän laajuuteen. Tämän jälkeen saatu kulma suhteutetaan silmien ja pään kääntökulmien avulla suhteelliseksi robottiin. Uusien tunnistuksien käsittelyyn tehtiin funktio, joka arvioi

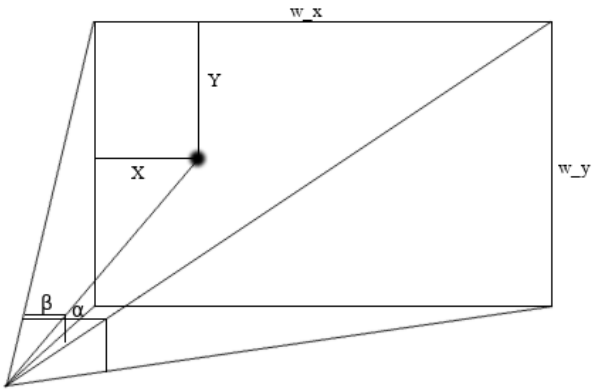
aikaisempien tunnistuksien sijainnin ja yhdistää uuden siihen määriteltyjen ehtojen täyttyessä.

Pallokoordinaatiston toiminnalle on tärkeää, että kameran näkökenttä ja servojen kulmat ovat tarkkoja. Silmien kääntyessä virheellinen näkökenttä tai kääntösuhde aiheuttaa tunnistuksen sijoittamisen väärään sijaintiin, mikä häiritsee tunnistuksen yhdistämistä aikaisempiin.

Kuvassa 8 on havainnollistettu laskenta havainnon kulmille suhteessa kameran suuntaan käyttäen kaavaa (1), joka on johdettu kuvan sijainnista pikseleissä suhteessa kuvan kokoon. Tämä laskenta tehdään molemmille akseleille kuvassa.

$$\alpha = \tan(x \times \arctan(FoV)), \tag{1}$$

missä α on havainnolle saatu kulma, x on havainnon keskikohta $[0, w_n]$ valitulla akselilla normalisoituna välille $[-1, 1]$ ja FoV on kyseisen akselin näkökentän laajuus, jossa w_n on valitun akselin leveys pikseleinä: w_x tai w_y.



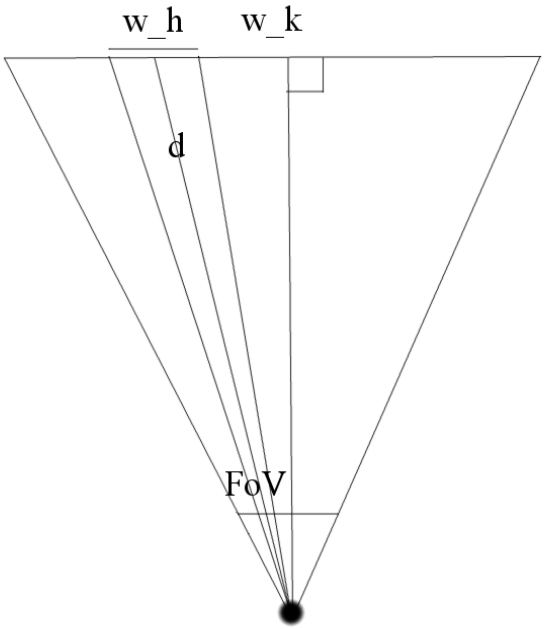
Kuva 8. Kulman laskeminen kameran kuvasta

Havainnon etäisyys lasketaan käyttämällä kaavaa (2), joka suhteuttaa havainnon koon kuvan kokoon ja tämän näkökenttään, havainnollistettuna kuvassa 9. Laskennassa käytetään kasvojen leveyttä, koska se on luotettavampi arvio kuin korkeus.

$$d = \frac{w_f}{\arctan(\frac{w_h}{w_k \times FoV}) \times \arccos(\alpha_x)}, \tag{2}$$

missä d on arvoitu etäisyys, w_f kasvojen oletettu leveys, w_h havainnon leveys ja w_k kuvan leveys.

Silmien suuntaaminen toteutettiin suhteuttamalla haluttu kulma servojen askeliksi käyttämällä kaavaa (3), jossa x_s on suhteutettu arvo, x_h haluttu kulma ja i kokeellisesti tai laskennallisesti määritelty kerroin servon kääntymisen vaikuttamisesta silmien kulmaan. Kaava suhteuttaa ensiksi halutun kulman voimanvälityssuhteella ja keskittää seuraavaksi kulman servon 300 asteen alueen keskelle lisäämällä arvoon 150 astetta. Lopuksi näin laskettu servolle haluttu kulma muutetaan sen käyttämiin 8-bitin arvoisiin askeliin kuvaamaan 0-300 astetta.



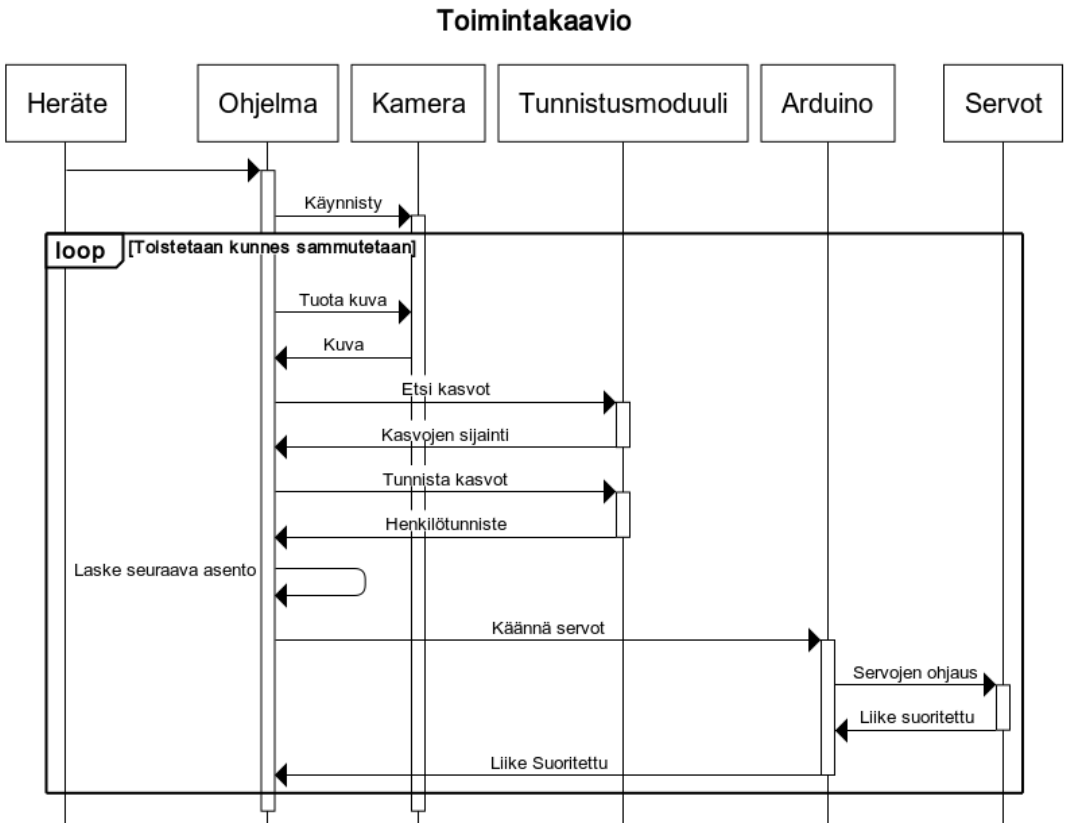
Kuva 9. Etäisyyden laskeminen

$$x_s = \frac{(150 + (x_h \times i))}{(300/1024)} \tag{3}$$

3.5. Kokonaisuus

Järjestelmän käynnistyessä kamera alkaa tuottaa videokuvaa, josta etsitään kasvoja tunnistusmoduulilla. Tunnistusmoduuli palauttaa kasvojen sijainnin kuvassa ja pallokoordinaatistossa. Jos kuvasta löydettiin kasvot, halutessa voidaan määrittää kasvoista henkilötunniste. Kun tiedetään, missä kohdassa kuvaa kasvot ovat, voidaan laskea, miten silmiä ohjaavia servoja täytyy kääntää, jotta kasvot saadaan keskelle kuvaa. Tämän jälkeen ohjaustiedot lähetetään rajapinnan kautta arduinolle, joka ohjaa servomoottoreita suorittamaan halutun liikkeen ja prosessi alkaa alusta. Järjestelmän toiminta on havainnollistettu alla olevassa toimintakaaviossa 10.

Jotta järjestelmä imitoisi hyvin ihmisen silmien liikettä, viiveet eri toimintojen suorittamiseen täytyy saada realistiselle tasolle. Esimerkiksi viive silmien liikuttamiselle reaktiona johonkin, on noin 100-300 millisekunttia. Servojen nopeus riittää hyvin ihmisen silmien nopeuden simuloimiseen ja servojen vasteaika saadaan hyvin pieneksi. Usean korjausliikkeen tekeminen nopeasti on vaikea saavuttaa tällä tavalla ja komentoja, joita ei ehditty suorittaa joudutaan karsimaan pois välistä viiveen pitämiseksi hallinnassa. Suurimmaksi ongelmaksi suorituskyvyssä nousee kuvan analysoiminen kasvontunnistuksessa, joka vaatii huomattavasti suorituskykyä suuren pikselimäärän takia. Testeissä tämä saatiin laskettua optimoinnilla noin 200 millisekuntiin yhtä kuvaa kohti. Eli vaste tapahtumaan kuvassa saadaan vastaamaan hyvin ihmisen reaktioaikaa.



Kuva 10. Järjestemän toimintaa kuvaava toimintakaavio

ROS-moduulia käyttäen toteutettiin rajapinta työssä toteutettujen ominaisuuksien käyttämiseksi. Moduuli toimii palvelimena ja vastaa JSON-muotoisilla viesteillä annettuihin komentoihin. Tämä mahdollistaa ryhmien toteuttamien toiminnallisuuksien yhdistämisen monimutkaisempien käyttötarkoitusten saavuttamiseksi. Taulukossa 2 on esitelty toteutetuille toiminnallisuuksille suunnitellun rajapinnan käskyt ja niiden käyttö.

Rajapinta hoitaa yhteyden kameraan ja arduino-korttiin, joka kommunikoi servojen kanssa. Rajapinta tekee automaattisesti tarvittavat yksikkömuunnokset kumpaankin suuntaan, jotta ohjauslogiikan laskut voidaan hoitaa yksinkertaisesti asteina.

Taulukko 2. Rajapinnan funktiot ja niiden vastaukset

Komento	detect_from_feed	eyes	recognize_person
Komennon selite	Etsi videosta kasvoja	Hae tai määritä silmien suunta asteina	Tunnista henkilö
Parametrit	[]	[]/[x,y]	[detection_id]
Tyyppi	Haku	Haku/Määritys	Haku
Onnistunut palaute	[{"detection_id":0, "pi_coord":[0,0], "sp_coord":[0,0,0]}]	{eye_position: [x,y]}	{person_id:0}
Poikkeus/ virhe	{error:"Example"}	{error:"Example"}	{error:"Example"}

4. TESTAUS

Seuraavaksi esitellään kehityksen aikana kohdatut ongelmat, ongelmien ratkaisut sekä toiminnan testauksia ja niiden tuloksia. Käydään läpi ongelmat ja ratkaisut, joita kohdattiin Rasperry Pi:n, Arduinon ja servomoottoreiden välisessä kommunikaatiossa sekä servomoottoreiden ohjauslogiikan implementaatiossa. Lopuksi esitellään testejä, joiden avulla mitattiin ja parannettiin järjestelmän suorituskykyä.

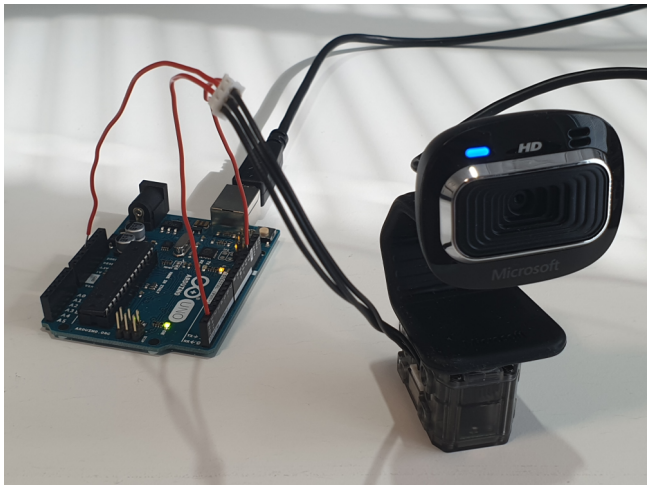
4.1. Kommunikointi

Kasvojentunnistuksen ja servomoottoreiden ohjausmoduuleita testattiin Raspberry Pi 4:lla. Kyseistä alustaa käyttämällä ohjelman toiminnassa ilmaantui huomattavaa viivettä, sillä ohjelman yhden syklin prosessointi vei liikaa aikaa, jolloin toiminta jäi jälkeen reaaliajasta. Viiveen takia kameran syöte oli noin sekunnin jäljessä reaaliaikaa ja siten myös silmiä ohjaavien moottoreiden ohjaus ei pysynyt reaaliajassa. Toiminnan nopeuttamiseksi kamerasta tulevan kuvan resoluutiota pudotettiin, jolloin prosessoitavan tiedon määrää saatiin pienemmäksi. Päädyttiin käyttämään resoluutiota 640×480, jolloin kamerasta tuleva kuva ja moottoreiden ohjaus vaikutti reaaliaikaiselta. Resoluution pienentäminen liikaa vaikeuttaa kasvojen tunnistamista pidemmiltä etäisyyksiltä.

Aluksi Arduino ohjasi servomoottoreita lähettämällä hyvin nopeasti komentoja. Kun servomoottori ei pystynyt suorittamaan tai vastaanottamaan käskyjä tarpeeksi nopeasti, se jäi jälkeen. Puskurin täyttyessä myös tietokoneen ja Arduinon välinen kommunikointi jäi odottamaan käskyjen prosessointia ja ohjelman suoritus jumittui lähes täysin. Arduinon ja servomoottoreiden välisen sarjakommunikoinnin vasteaika haluttiin mahdollisimman pieneksi ja yksittäisten käskyjen merkitys oli pieni, joten puskuri tyhjennettiin ennen jokaista käskyä. Tämä tyhjentää suorittamatta jääneet komennot ja jättää suoritettavaksi ainoastaan viimeisimmän käskyn.

4.2. Ohjauslogiikka yhdellä akselilla

Servomoottoreiden ohjauslogiikkaa testattiin aluksi vain vaaka-akselilla väliaikaisella testauskokoonpanolla, jota on havainnollistettu oheisessa kuvassa 11. Testausta varten kamera kiinnitettiin yhteen servomoottoriin, jolloin kameraa voitiin kääntää. Kameroita pyrittiin ohjaamaan niin, että kuvasta löydetyt kasvot saadaan kuvan keskelle. Testissä pyrittiin selvittämään ohjauslogiikan toimivuus ja tarkkuus, sekä tekemään parannuksia tulosten perusteella. Ensimmäisissä testeissä ohjauslogiikka onnistui kääntämään kameran kohti kasvoja, mutta se yliarvioi vaadittavan liikkeen suuruuden, jolloin kameraa jouduttiin kääntämään hiukan vastakkaiseen suuntaan yliarvioinnin kompensoimiseksi.



Kuva 11. Kameran kääntämisen testikokoonpano

4.3. Ohjauslogiikka kahdella akselilla

Tämän InMoov-kokoonpanon molempia silmiä ohjataan kahdella servolla, joista toinen kääntää silmiä vaakasuunnassa ja toinen pystysuunnassa. Kyseinen kokoonpano on esitetty kuvassa 12. Vaaka-akselilla käytetään väännön välitystapaa, joka käytännön testeillä määritetysti liikuttaa silmiä kuusi kertaa hitaammin verrattuna servon kääntökulmaan. Silmän käännös reunasta reunaan on pituudeltaan noin 30 astetta, joka vastaa 180 asteen moottorin pyörähdystä. Alustavissa testeissä kasvoja onnistuttiin seuraamaan kahdella akselilla tyydyttävällä tarkkuudella. Kasvojen keskityksessä havaittiin taas pientä yliarviointia, mutta pienellä hakemisella kasvot saatiin kuvan keskelle. Esimerkki kuvassa käytetyt kasvot ovat peräisin tekijänoikeudettomasta elokuvasta *Charade*.

Kuvassa ylärivillä on esitetty robotin silmien asento ja niiden alapuolella robotin näkemä kuva. Silmien kääntyminen on havainnollistettu lyhyesti seuraavassa kuvassa 13. Kuvan vasemmanpuoleisissa paneeleissa silmät on suunnattu oikealle ja robotin näkökentän vasemmalle puolelle on tuotu kasvot, jolloin robotin silmät kääntyvät kohti kasvoja. Keskimmäisessä paneelissa on esitetty välivaihe, jossa silmät ovat siirtymässä kohti kasvoja ja lopuksi oikean puoleisessa paneelissa silmien katse on keskitetty kasvoihin. Kamera sijaitsee kuvassa punaisen ympyrän sisällä.

Uutena ongelmana ilmeni servojen ohjauksen ja lasketun sijainnin ylläpitäminen äärirajoilla, mikä aiheutti silmien jumiutumisen ääriasentoon. Tämä johtui toistaiseksi yksisuuntaisesta kommunikoinnista servon ja Arduinon välillä, joten servon sijainnista jouduttiin pitämään ohjelmallisesti kirjaa. Tässä tapahtui virhettä rajoilla, koska silmiä pyrittiin kääntämään, ja ohjelma antoi lasketun kulman muuttua yli servoille ohjelmallisesti rajoitetun liikealueen. Tämä aiheutti jatkuvasti lisääntyvän virheen laskennalliseen kulmaan.

4.4. Ohjauslogiikka- ja kommunikointiongelmien ratkaisut

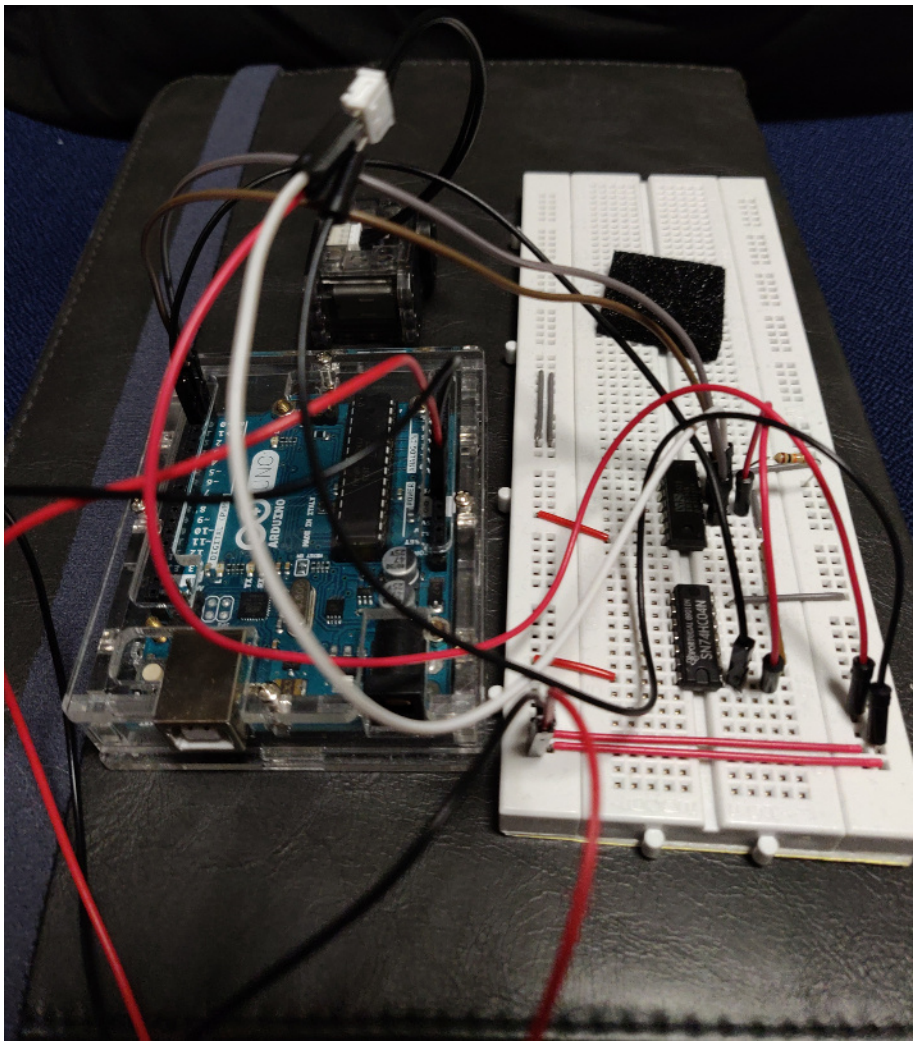
Silmän liikkeen äärirajoilla tapahtuvan jumiutumisen estämiseksi kokeiltiin toteutusta, jossa servomoottorit ilmoittavat jokaisen saamansa käskyn jälkeen oman sijaintinsa.

Kuva 12. InMoov-kokoonpano

Kuva 13. Silmien kääntyminen

Tätä varten tarvittiin erillinen puskuripiiri, jonka piirikaavio löytyy servojen manuaaleista. Puskuripiirin rakentamiseen käytettiin SN74HC04N-invertteripiiriä, kolmitilaista puskuripiiriä DM74126N, 10 kΩ vastusta sekä kytkentäjohtoja. Testikokoonpano on esitetty kuvassa 14. Tämän piirin lisäksi kaksisuuntainen kommunikaatio vaati toisen sarjaportin käyttämistä ohjainkäskyjä antavan laitteen ja Arduinon välillä, sillä Arduinon yksi sarjaportti oli varattu servojen ohjaamiseen. Tästä muodostui ongelmia Arduino Unoa käytettäessä, sillä kyseinen kehitysalusta sisältää vain yhden sarjaportin. Ongelman ratkaisemiseksi on Arduinolle luotu SoftwareSerial-

kirjasto, jonka avulla voidaan simuloida sarjaportin toimintaa myös muilla Arduinon I/O-nastoilla. Tämän toteutuksen avulla onnistuttiin saamaan asentotieto servoilta, vaikkakin osaksi virheellisenä. Kyseinen ongelma oli myös Arduino-kirjaston tekijän tiedostama ja sille ei toistaiseksi ollut keksitty ratkaisua.



Kuva 14. Kaksisuuntaisen sarjakommunikaation testikokoonpano

Toinen mahdollinen ratkaisu oli käyttää Dynamixel Arduino-korttia, joka sisältää valmiit piirit ja liitännät kaksisuuntaisen sarjaliikenteen hoitamiseksi. Kortin käytön myötä oli mahdollista hyödyntää Robotiksen Dynamixel2Arduino-kirjastoa. Arduino Unoa käytettäessä järjestelmä kuitenkin vaati jälleen SoftwareSerialin käyttöä, joka puolestaan vaatisi erillisen USB-sarja-adapterin. Järjestelmää testattiin myös Arduino Mega -kehitysalustalla sen sisältämien neljän sarjaportin vuoksi. Tällä tavoin sarjaliikenne toimi moitteetta kaksisuuntaisena, mutta servoille komentoja antavat funktiot eivät toimineet yhdessä Dynamixel2Arduino-kirjaston funktioiden kanssa.

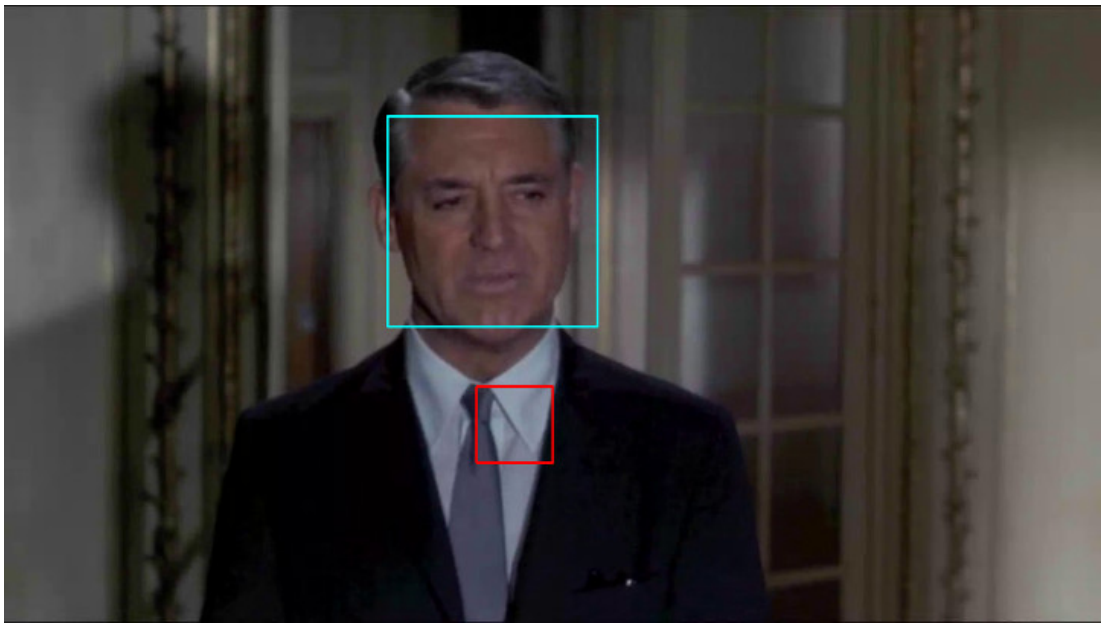
Ongelma servojen rajakäyttäytymisessä ratkaistiin rajoittamalla silmien kulmat jo ennen servojen kulmiksi muuntamista. Tämä pitää tarkasti kirjaa silmien sijainnista, eivätkä laskennallinen kulma ja servojen kulma epäsynkronoidu kääntyvyyden rajalle tultaessa. Tätä varten määriteltiin kokeellisesti maksimit silmien kääntymiselle, niitä

vastaavat kääntymiskulmat sekä servojen ohjausarvot. Tästä saatiin kertoimet, joilla voidaan laskea kuinka paljon servoja täytyy kääntää halutun kulman saavuttamiseksi.

4.5. Kasvontunnistus

Kasvontunnistusalgoritmin suorituskykyä testattiin eri minNeighbors- ja scaleFactor-parametrien arvoilla, syöttämällä algoritmille sama testiaineisto ja mittaamalla algoritmin aidot ja väärät positiiviset tunnistukset sekä aidot ja väärät negatiiviset tunnistukset. Tunnistusten oikeellisuuden lisäksi mitataan myös algoritmin suoritusaikaa. Testit suoritettiin pöytäkoneella, joka on huomattavasti Raspberry Pi:tä tehokkaampi, mutta testit auttavat siitä huolimatta valitsemaan optimaaliset arvot testattaville parametreille. Algoritmi tulee käyttäytymään samalla tavalla myös Rasperry Pi:llä, mutta suoritusaika tulee olemaan hitaampi.

Aito positiivinen tunnistus tarkoittaa sitä, että algoritmi on tunnistanut kasvot kuvasta oikein. Väärä positiivinen tunnistus puolestaan sitä, että algoritmi tekemä tunnistus ei sisällä kasvoja. Aito negatiivinen tunnistus tarkoittaa sitä, että algoritmi jättää oikein tunnistamatta kasvot, kun niitä ei ole. Aitoja negatiivisia tässä tapauksessa on järjetöntä laskea, sillä niitä on mahdollista olla missä tahansa kohdassa kuvaa. Väärä negatiivinen tarkoittaa sitä, että algoritmi ei onnistu löytämään kuvasta kasvoja, vaikka sen pitäisi. Oheisessa kuvassa 15 havainnollistettu aito positiivinen havainto ja väärä positiivinen havainto. Aito positiivinen havainto on sinisellä laatikolla rajattu ihmisen kasvo ja väärä positiivinen havainto on punaisella laatikolla rajattu virheellinen tunnistus.



Kuva 15. Aito ja väärä positiivinen tunnistus - Kuvakaappaus tekijänoikeidettomasta elokuvasta *Charade*

Testiaineisto tuotettiin tekijänoikeudettomasta elokuvasta *Charade*, ottamalla lyhyestä keskustelukohtauksesta 84 kpl HD 720p kuvaa käsittelyyn. Kuvista laskettiin manuaalisesti kasvot, jotka algoritmien pitäisi pystyä tunnistamaan. Kuvista laskettiin

yhteensä 145 kasvoa. Testien tulokset on kirjattu oheiseen taulukkoon 3. Taulukossa minNeighbors- ja scaleFactor-parametrien arvot ovat omalla rivillään ja mitatut arvot sarakkeittain. Aidot positiiviset ja väärät negatiiviset ovat ilmoitettuina prosentteina suhteessa aineiston kokoon eli 145 kasvoon. Väärät positiiviset ovat ilmoitettuna havaintomääränä, sillä mahdollisten väärien tunnistuksien määrää ei ole määritetty. Algoritmin suoritusaika testattiin Windows 10 -käyttöjärjestelmällä *Intel Core* i7-6700K -suoritinta käyttäen ja se on ilmoitettu taulukossa sekunteina.

Kuva 16. Esimerkkejä testimateriaalista

Taulukossa 3 havainnollistetaan parametrien minNeighbors ja scaleFactor vaikutusta tunnistusten määrään, oikeellisuuteen ja ohjelman suoritusaikaan. Valitsemalla parametreille arvoiksi minNeighbors=5 ja scaleFactor=1,3 maksimoidaan aidot positiiviset tulokset saaden silti huomattavasti pienempi väärien positiivisten tulosten määrä. Tässä yhteydessä tärkeä suoritusaika saadaan tällä tavalla puolitettua verrattuna toiseen hyvään kokoonpanoon, minNeighbor=10 ja scaleFactor=1,1, menettämällä vain 9 prosenttiyksikköä aitoja positiivisia tunnistuksia ja yhtä aikaa laskemalla vääriä positiivisia 55%.

Raspberry Pi 4:lla testiparametreja minNeighbors=5 ja scaleFactor=1,3 käyttämällä testitunnistusten suorittamiseen kului 17,44 sekuntia eli yhdestä kuvasta tehtäviin tunnistuksiin kului keskimäärin noin 208 millisekuntia.

Testiaineisto sisälsi tarkoituksella paljon kuvia, joissa kasvot eivät olleet suoraan kameraa päin. Tämän takia pienimmälläkin testiparametrin arvolla vain 43% kasvoista tunnistettiin oikein. Jos kaikki aineiston kasvot olisivat suorassa kameraa kohti, oikeiden tunnistusten määrä olisi huomattavasti suurempi. Testiaineiston kuvia on havainnollistettu oheisessa kuvassa 16.

Säätämällä minNeighbors-parametria voidaan vaikuttaa oikeiden ja väärien tunnistusten määrään. Kaskadiluokittelija käy kuvan ensin läpi pienimmällä mahdollisella havaintoikkunalla, jonka jälkeen havaintoikkunaa kasvatetaan scaleFactor-parametrin mukaan. Kun kuva on käyty läpi käyttäen kaikkia eri

havaintoikkunan kokoja, voidaan minNeighbors-parametrin arvolla suodattaa pois ne havainnot, joilla ei ole tarpeeksi eri kokoisilla havaintoikkunoilla tehtyjä tunnistuksia.

Taulukko 3. Kasvontunnistuksen parametrien testaus

Parametrit	Aidot Positiiviset	Väärät Positiiviset	Väärät Negatiiviset	Suoritus-aika
minNeighbors=5 scaleFactor=1.1	43% (63/145)	106	57% (82/145)	9,43s
minNeighbors=10 scaleFactor=1.1	40% (58/145)	27	60% (87/145)	9,31s
minNeighbors=15 scaleFactor=1.1	33% (48/145)	10	67% (97/145)	9,21s
minNeighbors=5 scaleFactor=1.2	37% (53/145)	34	63% (92/145)	6,41s
minNeighbors=10 scaleFactor=1.2	30% (43/145)	4	70% (102/145)	6,30s
minNeighbors=15 scaleFactor=1.2	24% (35/145)	0	76% (110/145)	6,27s
minNeighbors=5 scaleFactor=1.3	31% (45/145)	12	69% (100/145)	4,75s
minNeighbors=10 scaleFactor=1.3	22% (32/145)	0	78% (113/145)	4,65s
minNeighbors=15 scaleFactor=1.3	17% (25/145)	0	83% (120/145)	4,65s
minNeighbors=5 scaleFactor=1.4	27% (39/145)	5	73% (106/145)	3,66s
minNeighbors=10 scaleFactor=1.4	19% (27/145)	0	81% (118/145)	3,60s
minNeighbors=15 scaleFactor=1.4	10% (15/145)	0	90% (130/145)	3,55s
minNeighbors=5 scaleFactor=1.5	27% (39/145)	7	73% (106/145)	3,60s
minNeighbors=10 scaleFactor=1.5	18% (26/145)	0	82% (119/145)	3,53s
minNeighbors=15 scaleFactor=1.5	9% (13/145)	0	91% (132/145)	3,45s

4.6. Henkilöntunnistus

Henkilöntunnistukseen suunniteltiin autoenkooderipohjainen ratkaisu käyttäen Tensorflow-kirjastoa ja sen Keras-rajapintaa. Autoenkooderi koulutettiin muutamasta tekijänoikeudettomasta elokuvasta kerätyllä aineistolla, jonka koko oli noin 2500 kasvoa. Koulutuksessa päädyttiin käyttämään Keras:n mukana tulevaa Adam-optimointifunktiota, 0,1 opetusnopeutta ja 30 opetusjaksoa. Opetusjaksojen lisääminen yli 30 vaikutti aiheuttavan ylisovittamista pienen aineiston takia.

Koulutuksen jälkeen autoenkooderin henkilöntunnistuksen tarkkuutta testattiin kasvoilla, jotka eivät esiinny koulutus- tai testiaineistossa. Testeihin valittiin samasta henkilöstä muutamia kuvia, joissa henkilöllä oli eri ilmeitä ja taustan tai kuvan valaistus oli erilainen. Lisäksi testeihin valittiin myös yksi kuva täysin eri henkilöstä, jotta voitiin mitata ja vertailla tulosten poikkeamia.

Koulutettua enkooderia käyttäen kasvoista muodostettiin yksinkertaistetut muodot, joita sitten vertailtiin toisiinsa. Yksinkertaistetusta muodosta voidaan dekoodata uudelleenrakennettu versio alkuperäisestä. Kahden yksinkertaistetun muodon erojen perusteella voitiin päätellä, olivatko kyseessä saman henkilön kasvot. Oheisessa kuvassa 17 on havainnollistettu alkuperäiset kuvat ylimmällä rivillä, 16-ulotteisesta vektorista uudelleenrakennetut kuvat keskirivillä ja 32-ulotteisesta vektorista uudelleenrakennetut kuvat alimmalla rivillä. Kasvot kuvissa 17 ja 18 ovat peräisin tekijänoikeudettomasta elokuvasta *Check and Double Check*.

Kuva 17. Alkuperäiset kuvat ylärivillä, autoenkooderin 16- ja 32-ulotteiset rekonstruktiot alla

Autoenkooderilla muodostetut yksinkertaistetut esitysmuodot ovat 32-ulotteisia vektoreita, joiden eroja mitattiin laskemalla vektoreiden välinen euklidinen etäisyys kaavalla (5). Vektorilla (4) on havainnollistettu 16 ulottuvuuden enkooderilla laskettu yksinkertainen esitysmuoto kasvoille. Taulukossa 4 on esitetty muutamien yksinkertaistettujen muotojen etäisyydet. Etäisyydet on laskettu kuvassa 17 esitetyistä kasvoista muodostetuista vektoreista, joista ensimmäinen on referenssikuvan vektori, johon muita vertaillaan.

Tuloksista nähdään, että etäisyys kahden samasta henkilöstä otetun kuvan välillä on huomattavasti pienempi, kuin etäisyys kahden eri henkilöistä olevien kuvien välillä. Nähdään myös, että eri vektoriulottuvuuksilla mitatut etäisyydet muuttuvat lähes samassa suhteessa. Eri valaistus aiheuttaa suurimman eron kahden saman henkilön etäisyyteen, mutta ero on silti huomattavasti pienempi kuin eri henkilöiden välinen. Yksinkertaistetun esitysmuotovektorin ulottuvuuksien määrää kasvattamalla

Taulukko 4. Autoenkooderin eri ulottuvuusmäärän testaus

Syötetty kuva	Etäisyys (16 ulottuvuutta)	Etäisyys (32 ulottuvuutta)
Referenssi	0	0
Samannäköinen kuva	8,8	282,3
Sama henkilö, eri ilme	9,4	253,8
Sama henkilö, eri valaistus	19,2	500,5
Eri henkilö	29,7	805,5

saadaan huomattavasti paremman näköiset uudelleenrakennetut kuvat, eikä tarvittavan tallennustilan määrä kasva kohtuuttomasti. Tarvittaessa ulottuvuuksien määrää voidaan helposti pudottaa.

Enkooderilla muodostettu 16-ulotteinen vektori:

$$\begin{bmatrix} 0{,}28 & -10{,}64 & 2{,}75 & 5{,}74 & 1{,}06 & 4{,}93 & -9{,}16 & -8{,}81 & 15{,}85 & 6{,}21 & -6{,}29 & 0{,}99 & 1{,}58 & 10{,}11 & -7{,}32 & -2{,}94 \end{bmatrix} \quad (4)$$

Euklidisen etäisyyden laskemiseen käytetty kaava (5), jossa a ja b ovat vertailtavat vektorit, d laskettu etäisyys ja n vektorien ulottuvuuksien määrä

$$d = \sqrt{\sum_{i=0}^{n-1} (b[i] - a[i])^2} \quad (5)$$

Seuraavaksi kasvojen väliselle etäisyydelle määritettiin kokeellisesti raja-arvo, jonka alapuolella olevat arvot määritetään samaksi henkilöksi ja yläpuolella olevat eri henkilöksi. Testiaineistoksi valittiin neljä henkilöä, joista jokaisesta on kaksi kuvaa. Testiaineisto on esitetty kuvassa 18, jossa keskirivillä on käytetyt kasvot, yläpuolella kasvolle annettu nimi ja alapuolella kasvojen rekonstruktio.

Kuva 18. Testikuvat henkilöntunnistukseen ja niiden rekonstruktiot

Testissä jokaisesta kasvosta muodostettiin vektori, jonka etäisyys mitattiin kaikkiin muihin kasvoista muodostettuihin vektoreihin. Testien tulokset on kirjattu taulukkoon 5. Taulukossa yhdellä rivillä on henkilön etäisyydet muihin henkilöihin. Solut, joissa on ilmoitettu etäisyys samaan henkilöön, on korostettu väreillä.

Tuloksien perusteella valittiin raja-arvoksi 600, koska etäisyydet samaan henkilöön ovat sen alapuolella ja eri henkilöihin sen yläpuolella. Poikkeuksena on henkilö kuvissa 6 ja 7, koska kuvissa henkilön kasvot ovat suuntautuneet eri lailla.

Taulukko 5. 32 ulotteisen autoenkooderin testaus

	1	2	3	4	5	6	7	8
1	0	269	832	805	826	739	844	1049
2	269	0	752	775	813	693	795	953
3	832	752	0	254	1122	918	611	634
4	805	775	254	0	1132	941	650	720
5	826	813	1122	1132	0	564	1260	1088
6	739	693	918	941	564	0	1134	968
7	844	795	611	650	1260	1134	0	704
8	1049	953	634	720	1088	968	704	0

Raja-arvon määrittämisen jälkeen henkilöntunnistuksen tarkkuutta mitatattiin suuremmalla aineistolla. Testissä yksi henkilö valittiin vertailun kohteeksi, josta muodostetusta vektorista laskettiin etäisyys muihin ja määriteltiin etäisyyden perusteella, ovatko vertailussa olevat henkilöt samat. Vertailujen jälkeen laskettiin manuaalisesti, kuinka monta oikeaa ja väärää tunnistusta algoritmi tuotti. Testiaineistona oli 45 kasvoa, joista 11 oli referenssihenkilön kasvoja ja loput 34 olivat eri henkilöitä. Testien tulokset on ilmoitettu taulukossa 6. Tuloksien perusteella algoritmi ei tunnista väärää henkilöä samaksi kovin herkästi, mutta ei aina onnistu tekemään oikeaa tunnistusta samasta henkilöstä erityisesti silloin, kun henkilön kasvot eivät ole suoraan kameraa kohti.

Taulukko 6. Autoenkooderin ennustustarkkuuden testaaminen

	Positiivinen	Negatiivinen
Positiivinen	8 (Aito Positiivinen)	0 (Väärä Positiivinen)
Negatiivinen	3 (Aito Negatiivinen)	34 (Väärä Negatiivinen)

Raspberry Pi 4 suoritti tämän 45 kasvon lataamisen, enkoodaamisen ja etäisyyksien vertailemisen 3,2 sekunnissa, eli keskimäärin 70 ms/kuva. Tämän viiveen takia henkilöntunnistus kannattaa suorittaa erillisessä säikeessä. Henkilöntunnistuksella ei ole luonnollisen käyttäytymisen kannalta yhtä suurta tarvetta pienelle vasteajalle kuin silmien liikkeellä, joten se voidaan suorittaa pienemmällä prioriteetillä.

5. POHDINTA

Tässä luvussa analysoidaan työn tuloksia ja verrataan tuloksia aluksi asetettuihin päämääriin. Työn toimivuutta ja tehokkuutta verrataan muihin vastaaviin töihin. Lopuksi pohditaan mitä työssä olisi voitu tehdä paremmin ja kuinka kehitystä voitaisiin jatkaa.

5.1. Tulokset

Työn tavoitteena oli kehittää robotin näköjärjestelmä, joka etsii reaaliaikaisesta videosta kasvoja ja tuottaa videosta saatujen tietojen perusteella luonnollista silmien liikettä. Asetetut tavoitteet saavutettiin kohtalaisen hyvin, mutta joidenkin toiminnallisuuksien kehitystä olisi voitu jatkaa. Robotti pystyy löytämään näkökentästä luotettavasti kasvot, keskittämään katseensa niitä kohti ja seuraamaan niitä. Silmien liikkeet ovat sakkadisia, eli robotti kääntää katsetta yhdellä nopealla liikkeellä nykyisestä sijainnista kohteeseen. Henkilöntunnistusta testattiin ja sen todettiin pystyvän erottelemaan ihmiset melko hyvin toisistaan.

5.2. Jatkokehitys

Suorituskyvyn parantamiseksi voitaisiin jatkossa käyttää tehokkaampaa tietokonetta, kuten kannettavaa tietokonetta tai verkon kautta yhdistää suurempaan laskentatehoon, kuten pöytäkoneeseen tai palvelimeen. Suuremmalla laskentateholla mahdollistettaisiin suuremman resoluution käyttäminen, mikä mahdollistaisi paremman tunnistamisen kauemmas ja prosessoinnin kannalta raskaampien parametrien käyttämisen, mikä taas vähentäisi vääriä positiivisia. Taulukossa 3 esitetyistä parametreista voitaisiin käyttää esimerkiksi minNeighbors=10 ja scaleFactor=1.1. Tämä lisäisi aitojen positiivisten tulosten määrää, mutta myös vääriä negatiivisia. Näitä vääriä positiivisia tunnistuksia voitaisiin vähentää muuttamalla prosessi vaatimaan useampia tunnistuksia ennen hyväksymistä aidoksi, vähentäen vääriä positiivisia havaintoja. Ylimääräisen laskentatehon ansiosta voitaisiin silti säilyttää viive alle halutun. Kasvojentunnistusta kuvasta voitaisiin myös parantaa kouluttamalla oma kaskadiluokittelija. Tämä ei vaikuttaisi suorituskykyyn, mutta kasvojentunnistus voitaisiin saada toimimaan luotettavammin käyttämällä koulutusmateriaalia, joka on lähempänä käyttötarkoitusta.

Henkilöntunnistuksen luotettavuutta voisi parantaa kouluttamalla autoenkooderi suuremmalla koulutusaineistolla, jossa esiintyy paljon enemmän uniikkeja kasvoja useilla erilaisilla kasvonpiirteillä, kuten esimerkiksi eri kokoisia ja muotoisia neniä. Tässä työssä käytetyssä koulutusaineistossa esiintyi varsin vähän esimerkiksi parrakkaita miehiä, jolloin parroista aiheutuu jonkin verran ongelmia autoenkooderille. Henkilöntunnistamista voitaisiin myös parantaa jättämällä huomiotta vektorien etäisyyden laskemisessa arvoja, jotka eivät liity ihmisten kasvojen ominaisuuksiin, kuten tausta. Tarpeeksi pitkälle koulutetulla autoenkooderilla voitaisiin myös ottaa huomioon ihmisen ilme. Mahdollinen vaihtoehto autoenkooderille olisi myös pääkomponenttianalyysi (PCA). Tällä saataisiin eriteltyä kasvojen kannalta

olennaiset ulottuvuudet ja vähennettyä näin esimerkiksi ympäristön aiheuttamia vääriä tunnistuksia.

Kameran kuvaa saadaan Raspberry Pi 4:llä luettua vajaa 10 kuvaa sekunnissa. Tämä voisi olla suurempi nopeampaa ja pienempiviiveistä tunnistamista varten. Suurempi kameran virkistystaajuus vähentäisi myös silmien liikkeestä kuvaan aiheutuvaa sumentumista. Nykyisen kameran näkökenttä on noin 60 astetta horisontaalisesti, kun taas ihmisen näkökenttä on yli 180. Näkökenttä voisi olla laajempi, jotta se vastaisi realistisemmin ihmisen näkökenttää. Tämä helpottaisi ihmisten löytämistä. Tätä varten tarvittaisiin hieman uudelleensuunnittelua päässä, sillä silmät ovat syvällä ja jo nykyisellä kokoonpanolla silmäkuoppien reunat rajaavat näkökenttää silmien kääntyessä äärirajoille molemmilla akseleilla.

Kahden kameran avulla voitaisiin simuloida stereonäköä ja saada syvyystietoa ympäristöstä. Tätä tietoa voitaisiin käyttää esimerkiksi virheellisten kasvohavaintojen karsimiseen etäisyyden perusteella. Ympäristön syvyystietoa voitaisiin käyttää myös haluttaessa esimerkiksi liikkumiseen.

Tässä työssä toteutetuilla ominaisuuksilla toteutettiin listatuista käyttötapauksista ohikulkijoiden seuraaminen. Yhdessä näiden ja muiden ryhmien kehittämien komponenttien ja ohjauslogiikan avulla voidaan toteuttaa käyttötapauksia, joista esiteltiin esimerkkejä edellisessä luvussa. Esimerkiksi kuulojärjestelmäkomponentin avulla pystyttäisiin toteuttamaan toiminto seurata puhujaa katseella.

Silmien liikkeiden luonnollisuutta olisi voitu kehittää paremmaksi. Toistaiseksi robotin silmien liikkeet ovat pääasiassa sakkadimaisia, mutta esimerkiksi kohdetta seuratessa silmien olisi syytä liikkua sulavasti ja tasaisesti. Jatkossa voitaisiin kehittää toiminnallisuus, jonka avulla robotin katse pysyy kohteessa pään liikkeistä huolimatta. Silloin robotti pystyy esimerkiksi nyökkäämään keskustelun aikana ilman, että silmät kääntyvät pään mukana ja katsekontakti katkeaa.

6. YHTEENVETO

Tässä työssä esiteltiin OpenCV:n avulla InMoov-robotin päälle kehitetty konenäkösovellus. Sovellus käytti OpenCV:n sisältämiä valmiita kaskadiluokittelijoita ja silmänä toimivan kameran toimintalogiikka pohjautui ihmisen silmän liikkeisiin sekä sovelluksen vaatimiin eri tarpeisiin. Havaittujen kasvojen seurannan tueksi esiteltiin pallokoordinaatistoa hyödyntävä malli.

Järjestelmä pystyy löytämään kasvot reaaliaikaisesta videosta ja seuraamaan katseellaan kasvoja sakkadimaisilla liikkeillä. Järjestelmä pystyy tunnistamaan kasvot ja erottamaan ne toisistaan niille annettujen pallokoordinaatistosijaintien avulla. Kasvoista tallennetaan myös autoenkooderilla muodostettu tunniste, jota voidaan käyttää myöhemmin kasvojen tunnistamiseen muodostamalla kasvoista kyseinen tunniste ja vertaamalla sitä tallennettuihin tunnisteisiin. Järjestelmää käytetään ROS-rajapinnan kautta sille suunnitelluilla komennoilla.

7. VIITTEET

[1] Gollapudi S. (2019) Artificial Intelligence and Computer Vision. Apress, Berkeley, CA. URL: https://doi.org/10.1007/978-1-4842-4261-2_1.

[2] Yu Haibin;Liu J.L.Z.Y.D. (2019) Intelligent robotics and applications : 12th International Conference, ICIRA 2019, Shenyang, China, August 8-11, 2019, Proceedings. Part VI. Lecture Notes in Artificial Intelligence; Lecture notes in computer science;LNCS sublibrary. SL 7, Artificial intelligence, Springer, Cham. URL: http://link.springer.com/10.1007/978-3-030-27529-7.

[3] Steger C., Ulrich M. & Wiedemann C. (2017) Machine Vision Algorithms and Applications. Wiley. URL: https://books.google.fi/books?id=2JY9DwAAQBAJ.

[4] BBC, China launches coronavirus 'close contact detector' app. URL: https://www.bbc.com/news/technology-51439401.

[5] Feng C., China launches coronavirus 'close contact detector' in effort to reassure public over health risks. URL: https://www.scmp.com/tech/apps-social/article/3050054/china-launches-coronavirus-close-contact-detector-effort-reassure.

[6] Minato T., Shimada M., Ishiguro H. & Itakura S. (2004) Development of an android robot for studying human-robot interaction. Teoksessa: B. Orchard, C. Yang & M. Ali (toim.) Innovations in Applied Artificial Intelligence, Springer Berlin Heidelberg, Berlin, Heidelberg, ss. 424–434.

[7] IntersoftConsulting (2018), Gdpr info. URL: https://gdpr-info.eu/.

[8] Russell S.J. (2016) Artificial intelligence: a modern approach. Prentice Hall Series in Artificial Intelligence, Pearson Education, Harlow, third edition. global edition p.

[9] Inmoov, Inmoov home page. URL: http://inmoov.fr/.

[10] Sheena D. & Young L.R. (1975) Survey of eye movement recording methods. Behavior Research Methods & Instrumentation URL: https://doi.org/10.3758/BF03201553.

[11] Deyi L. & Yi D. (2017) Artificial Intelligence with uncertainty. CRC Press.

[12] Piccinini G. (2000) Turing's rules for the imitation game. Minds and Machines URL: https://www.researchgate.net/profile/Gualtiero_Piccinini/publication/251383110_Turing's_Rules_for_the_Imitation_Game/links/59342fabaca272fc5541fb7f/Turings-Rules-for-the-Imitation-Game.pdf.

[13] Ooyen A.V. & Nienhuis B. (1992) Improving the convergence of the back-propagation algorithm. Neural Networks 5, ss. 465 – 471. URL: http://www.sciencedirect.com/science/article/pii/0893608092900087.

[14] Siregar S.P. & Wanto A. (2017) Analysis of artificial neural network accuracy using backpropagation algorithm in predicting process (forecasting). International Journal Of Information System and Technology URL: http://ijistech.org/ijistech/index.php/ijistech/article/view/4.

[15] LeCun Y., Bengio Y. & Hinton G. (2015) Deep learning. Nature URL: https://doi.org/10.1038/nature14539.

[16] Osipov (2019) Artificial Intelligence. Springer International Publishing. URL: http://sfx.nelliportaali.fi/nelli28b?url_ver=Z39.88-2004&ctx_ver=Z39.88-2004&ctx_enc=info:ofi/enc:UTF-8&rfr_id=info:sid/sfxit.com:opac_856&url_ctx_fmt=info:ofi/fmt:kev:mtx:ctx&sfx.ignore_date_threshold=1&rft.object_id=4100000009606108&svc_val_fmt=info:ofi/fmt:kev:mtx:sch_svc&.

[17] Ng A. (2011), Sparse autoencoder. URL: http://ailab.chonbuk.ac.kr/seminar_board/pds1_files/sparseAutoencoder.pdf.

[18] Hinton G.E., Krizhevsky A. & Wang S.D. (toim.) (2011) Transforming Auto-encoder. Department of Computer Science, University of Toronto. URL: http://www.cs.toronto.edu/~fritz/absps/transauto6.pdf.

[19] Shi H. (2018) Bidirectional long short-term memory variational autoencoder. University of Oulu. URL: http://urn.fi/urn:nbn:fi-fe201902266257.

[20] Wilson R., Hancock E.R., Bors A. & Smith W.A.P. (toim.) (2013) Computer Analysis of Images and Patterns : 15th International Conference, CAIP 2013, York, UK, August 27-29, 2013 : proceedings. Part II. Lecture notes in computer science, LNCS sublibrary, Springer, Heidelberg. URL: http://link.springer.com/10.1007/978-3-642-40246-3.

[21] Kelleher J.D. (2019) Deep Learning. The MIT Press, Place of publication not identified. URL: https://www.books24x7.com/marc.asp?bookid=147440.

[22] Kaehler A. & Bradski G. (2016) Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library. O'Reilly Media. URL: https://books.google.fi/books?id=SKy3DQAAQBAJ.

[23] Qureshi W., Payne A. & Walsh K. (2017) Machine vision for counting fruit for mango tree canopies. Precision Agriculture URL: https://doi.org/10.1007/s11119-016-9458-5.

[24] Li S.Z. & Jain A.K. (2011) Handbook of face recognition. Springer.

[25] Liu W., Wen Y., Yu Z., Li M., Raj B. & Song L. (2017) Sphereface: Deep hypersphere embedding for face recognition. Teoksessa: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), ss. 212 – 214.

[26] Jain L.P., Scheirer W.J. & Boult T.E. (2014), Multi-class open set recognition using probability ofinclusion. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.437.793&rep=rep1&type=pdf.

[27] Bendale A. & Boult T.E. (2016) Towards open set deep networks. Teoksessa: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), ss. 1–10.

[28] Viola P. & Jones M. (2001) Rapid object detection using a boosted cascade of simple features. Teoksessa: Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001, nide 1, IEEE, nide 1, ss. I–I.

[29] OpenCV (2019), Opencv documentation. URL: https://docs.opencv.org/.

[30] Cicolani J. (2018) Beginning Robotics with Raspberry Pi and Arduino : Using Python and OpenCV. Apress, Place of publication not identified. URL: https://www.books24x7.com/marc.asp?bookid=141433.

[31] Tensorflow (2020), Tensorflow website. URL: https://www.tensorflow.org/about.

[32] Abadi M., Agarwal A., Barham P., Brevdo E., Chen Z., Citro C., Corrado G.S., Davis A., Dean J., Devin M., Ghemawat S., Goodfellow I., Harp A., Irving G., Isard M., Jia Y., Jozefowicz R., Kaiser L., Kudlur M., Levenberg J., Mané D., Monga R., Moore S., Murray D., Olah C., Schuster M., Shlens J., Steiner B., Sutskever I., Talwar K., Tucker P., Vanhoucke V., Vasudevan V., Viégas F., Vinyals O., Warden P., Wattenberg M., Wicke M., Yu Y. & Zheng X. (2015), TensorFlow: Large-scale machine learning on heterogeneous systems. URL: https://www.tensorflow.org/, software available from tensorflow.org.

[33] ClearRobotics. URL: http://www.clearpathrobotics.com/assets/guides/ros/IntroToTheRobotOperatingSystem.html#what-is-ros.

[34] ROS (2020), About robot operating system. URL: https://www.ros.org/about-ros/.

[35] RasberryPi (2020), Rasberry pi 4 model b. URL: https://www.raspberrypi.org/products/raspberry-pi-4-model-b/.

[36] Microsoft (2020), Microsoft lifecam hd-3000. URL: https://www.microsoft.com/accessories/en-us/products/webcams/lifecam-hd-3000/t3h-00011.

[37] Dynamixel (2020), Xl-320 - introduction. URL: http://emanual.robotis.com/docs/en/dxl/x/xl320/.

[38] Arduino (2020), Arduino - introduction. URL: https://www.arduino.cc/en/Guide/Introduction.

[39] Microchip.com (2020), Atmega328p. URL: https://www.microchip.com/wwwproducts/en/ATMEGA328P.

[40] OpenCV (2019), Opencv github haarcascade. URL: https://github.com/opencv/opencv/blob/master/data/haarcascades/haarcascade_frontalface_default.xml.

[41] PublicDomainMovies, Public domain movies. URL: http://publicdomainmovie.net/feature_movies.

[42] Prattlibrary, What is public domain. URL: https://www.prattlibrary.org/research/tools/index.aspx?cat=105&id=5661.

[43] Young J.W. (1993), Head and face anthropometry of adult u.s. civilians. URL: https://apps.dtic.mil/docs/citations/ADA268661.