TIETO- JA SÄHKÖTEKNIIKAN TIEDEKUNTA

Anssi Meisalmi
Sampo Niittyviita
Mikael Martinviita

Kasvojentunnistussovelluksen haasteet lisätyn todellisuuden sulautetussa järjestelmässä

Kandidaatintyö

Tietotekniikan tutkinto-ohjelma

Toukokuu 2017

Anssi Meisalmi, Sampo Niittyviita, Mikael Martinviita (2017) Kasvojentunnistussovelluksen haasteet lisätyn todellisuuden sulautetussa järjestelmässä. Oulun yliopisto, tietotekniikan tutkinto-ohjelma. Kandidaatintyö, 40 s.

TIIVISTELMÄ

Tutkimuksessa tarkastellaan reaaliaikaisen lisätyn todellisuuden kasvojentunnistussovelluksen haasteita sulautetuissa järjestelmissä. Merkittävimmät haasteet liittyvät reaaliaikaisuuden toteuttamiseen säilyttäen samalla hyvä kuvanlaatu ja sovelluksen tarkoituksenmukaisuus. Työtä varten kehitetty kasvojentunnistussovellus testattiin Raspberry Pi -tietokoneella. Tarkkailun kohteena olivat sovelluksen suorituskyky ja tunnistustarkkuus, sekä Raspberry Pi:n kaltaisen tietokoneen soveltuvuus sovelluksen alustana.

Testit havainnollistavat sovelluksen toteutukseen vaikuttavien ratkaisujen merkitystä sovelluksen suorituskykyyn. Nämä vaatimukset täyttävä reaaliaikainen lisätyn todellisuuden sovellus tarvitsee tarkkaa kuvankäsittelyä, hyvää kuvanlaatua sekä mobiilisuutta. Sulautettu järjestelmä tarvitsee optimointia, reaktiivisuutta ja pientä kokoa. Kehitetty sovellus tunnistaa videokameran kuvasta ihmisen kasvot, piirtää kasvojen päälle grafiikkaa ja toistaa tehostetun kuvan reaaliaikaisesti käyttöliittymän kautta. Sovellus hyödyntää Raspberry Pi -kosketusnäyttöä ja Raspberry Pi -kameraa, sekä tietokonenäköä varten koottua kirjastoa OpenCV.

Sovellus ja alusta yhdessä eivät täyttäneet kaikkia suorituskyvyn vaatimuksia. Metrin etäisyydellä sovellus kykeni 99,9 %:n tunnistustarkkuuteen alhaisellakin resoluutiolla, mutta etäisyyden kasvaessa tunnistustarkkuus pieneni lähes täysin lineaarisesti kolmeen metriin, jossa se ei enää kyennyt tunnistamaan yhtään kasvoa kuvamateriaalista. Parhaimmillakin olosuhteilla ruudunpäivitysnopeus valitettavasti ei riittänyt toistamaan kolmeakymmentä kuvaa sekunnissa. Pienimmällä 144p resoluutiolla se oli kaksikymmentä kuvaa sekunnissa ja suurimmalla 720p resoluutiolla se oli kolme kuvaa sekunnissa. Raspberry Pi ei testien perusteella sovi reaaliaikaisen lisätyn todellisuuden sovelluksen alustaksi.

Avainsanat: Raspberry Pi 2, OpenCV, tietokonenäkö, kasvojentunnistus, suorituskyky, FPS

Anssi Meisalmi, Mikael Martinviita, Sampo Niittyviita (2017) Face Recognition Application in Augmented Reality Embedded System. University of Oulu, Degree Programme in Computer Science and Engineering. Bachelor's Thesis, 40 p.

ABSTRACT

In this research, the challenges of real time augmented reality application are inspected, while software is developed for embedded systems. The most significant challenges consider simultaneous implementation of real time application, high image quality and purposefulness of the software. For the purpose experimentation in this research, a face recognition software was developed and tested with Raspberry Pi 2 single board computer. Under examination were performance and face recognition accuracy of the application.

The experiments illustrate the effect on the performance of the software for the implementation choices' of the software. A real time augmented reality software, which passes these criteria, requires accurate image processing, high image quality and mobility. Embedded systems, on the other hand, require optimization, reactivity and small size. The developed software recognizes faces from image produced by the Raspberry Pi's video camera, draws graphic on the faces and plays the enhanced image in real time from the user interface. The software utilizes Raspberry Pi touchscreen and Raspberry Pi camera, as well as a computer vision library called OpenCV.

In the research, the software does not pass an acceptable level of performance using Raspberry Pi 2 single board computer. In one meter's distance, the software achieved face recognition accuracy of 99.9 %, yet, while increasing the distance, the face recognition accuracy diminished nearly in a linear fashion until three meters, in which the face recognition accuracy was zero. Additionally, while the image size grew larger, FPS was reduced exponentially. Even with the best conditions the FPS was low, for the smallest resolution, 144p, the FPS was 20 and for the largest, 720p, resolution the FPS was three.

Keywords: Raspberry Pi 2, OpenCV, computer vision, performance, FPS

SISÄLLYSLUETTELO

ALKULAUSE

Tämän kandidaatintyön kirjoittamisen mahdollistamisesta haluamme kiittää ohjaajiamme prof. Juha Röningiä ja Teemu Tokolaa arvokkaista neuvoista ja ohjeista.

Oulu, Toukokuu 3. 2017

Anssi Meisalmi, Sampo Niittyviita, Mikael Martinviita

LYHENTEIDEN JA MERKKIEN SELITYKSET

AR	Augmented reality, lisätty todellisuus
CPU	Central processing unit, keskusyksikkö
FPS	Frame rate, Frames per second, kuvataajuus, ruudunpäivitysnopeus
HD	High Definition, teräväpiirto tarkkuus
HMD	Head-mounted display, päässä pidettävä näyttölaite
IoT	Internet of Things, esineiden internet
LBP	Local Binary Patterns, paikalliset binääri mallit
RAM	Random-access memory, keskusmuisti
RP	Raspberry Pi
VR	Virtual reality, virtuaalinen todellisuus
ES	Embedded system, sulautettu järjestelmä

1. JOHDANTO

Lisätty todellisuus (engl. *augmented reality, AR)* yhdistää todellisen ja virtuaalisen maailman lisäämällä todelliseen kuvaan virtuaalisisältöjä, kuten kuvaa tai ääntä. Sitä voidaan käyttää viihteeseen tai työkaluna tunnistusta vaativissa tehtävissä. Teknologian tarjoamien sovelluskohteiden perusteella lisättyä todellisuutta on kannattavaa käyttää mobiililaitteissa, jolloin sovellusten ajoon tarvitaan joko älypuhelin tai sulautettu järjestelmä. Lisätty todellisuus vaatii järjestelmältä usein reaaliaikaisuutta, liikuteltavuutta, yleistä käytön mukavuutta, kuvankäsittelyä sekä verkkoviiveen hallintaa.

Älypuhelimet ovat ideaalisia käsikäyttöisiä AR-sovelluksia varten. Päässä pidettäviä sovelluksia varten on kuitenkin mielekkäämpää tutkia sulautetun järjestelmän käyttämistä lisätyn todellisuuden sovelluksen alustana. Tämä kuitenkin lisää omat vaatimuksensa lisätyn todellisuuden sovellusten vaatimusten lisäksi, kuten järjestelmän optimoinnin, reaktiivisuuden ja pienen kokoon. Näiden haasteiden yhdistämistä hankaloittaa yhden elementin muuttamisen vaikutus toisen elementin toimintaan, välillä haitallisesti. Tätä ongelmaa olisi mahdollista rajata käyttämällä valmiita yhden piirilevyn tietokoneita sovelluksen alustana. Tällöin ainakin järjestelmän resurssit, hinta ja koko ovat selvillä, mutta suorituskyky on sovelluskohteesta ja sen toteutuksesta riippuvainen.

Raspberry Pi:n (RP) kaltainen halpa, mutta suhteellisen tehokas yhden piirilevyn tietokone tarjoaa monia erilaisia käyttömahdollisuuksia erityisesti harrastusmielessä. RP:n prosessointivoima ja muisti rajoittavat sen soveltuvuutta yleiskäyttöön tarkoitettuna tietokoneena tablet-tietokoneiden tai älypuhelimien tapaan. RP kuitenkin suoriutuu hyvin sulautettuna järjestelmänä, jolloin sille annetaan yksi tehtävä, jota se suorittaa mahdollisimman tehokkaasti. RP:n soveltuvuutta muiden yhden piirilevyn tietokoneiden tapaan on tutkittu IoT:n sovelluksen alustana [1]. RP:stä kehitetyt sulautetut järjestelmät useimmiten liittyvät konenäköön ja erityisesti objektien, kuten kasvojen tunnistamiseen [2], tai enemmän IoT:n teemaiseen kodin tarkkailusovellukseen [3].

Lisätyn todellisuuden kasvaessa yhä useammalle alueelle on hyvä tutkia, voiko markkinoilta löytyvästä valmiista yhden piirilevyn tietokoneesta kehittää tehokas lisätyn todellisuuden sovellus. Vaikka Raspberry Pi:llä on aiemmin tehty kasvojentunnistussovelluksia, niin tutkimuksia, jotka koskisivat lisätyn todellisuuden aiheuttamia haasteita sovelluksen suorituskykyyn ei tämän työn yhteydessä löydytty.

Haasteiden tunnistamisen lisäksi tutkimuksessa havainnollistetaan, kuinka yksinkertaisen lisätyn todellisuuden sovelluksen suorituskykyyn vaikuttaa kuvan kokoon ja objektin etäisyyden muuttaminen. Tarkoituksena on esittää sovelluksen kehitysvaiheessa tehtyjen yksittäisten ratkaisujen vaikutus lopulliseen suorituskykyyn. Lisäksi voi päätellä, että millaisiin lisätyn todellisuuden sovelluksiin Raspberry Pi:n kaltaista tietokonetta voisi mahdollisesti käyttää. Olettaen sovelluksilla olevan seuraavat tehtävät: reaaliaikainen objektin tunnistus ja videokuvan tehostaminen ja toistaminen.

2. LISÄTTY TODELLISUUS

Lisätty todellisuus voidaan jakaa laitteiston, sovelluskohteiden ja prosessien mukaan. Laitteisto koostuu sensoreista, prosessoreista ja näytöistä, jotka voidaan vielä ottaa käyttöön päässä pidettävillä laitteilla, kädessä pidettävillä laitteilla ja projektoreilla. Laitteiston sensoreihin kuuluvat muun muassa valosensorit, äänisensorit ja gps-sensorit. Sensorit yhdessä käyttömenetelmän kanssa luovat perustan sovelluksiin, joita laitteistolla voidaan käyttää.

Lisätyn todellisuuden sovellusmahdollisuudet ovat laajenneet eri alueille teknologian kehittyessä. Nimittäin aiemmin live-kuvaan voitiin liittää informaatiota ja projisoida kuva eteenpäin, mutta nykyään tehokas kuvankäsittely ja pienet laitteistokomponentit mahdollistavat kuvien analysoimisen ja käsittelemisen helposti. Toistaiseksi sovelluskohteita on mm. pelialalla, teollisuudessa, lääketieteessä ja arkkitehtuurissa, mutta tulevaisuudessa tehokkaammat laitteet voivat mahdollistaa uusia menetelmiä, ja nykylaitteiden levinneisyys kasvattaa kevyiden sovellusten suosiota.

Tehostetun kuvan luomisprosessi on kuin konenäön prosessi: sensoreiden syöte esikäsitellään, segmentoidaan, tunnistetaan ja jälkikäsitellään. Lähes kaikilla lisätyn todellisuuden sovelluksilla on kuvan luomisprosessi lähes sama ennen jälkikäsittelyä. Jälkikäsittely määrittää sovelluksen käyttökohteen, erottaen esimerkiksi muistikuormaa helpottavat sovellukset tunnistusta helpottavista sovelluksista.

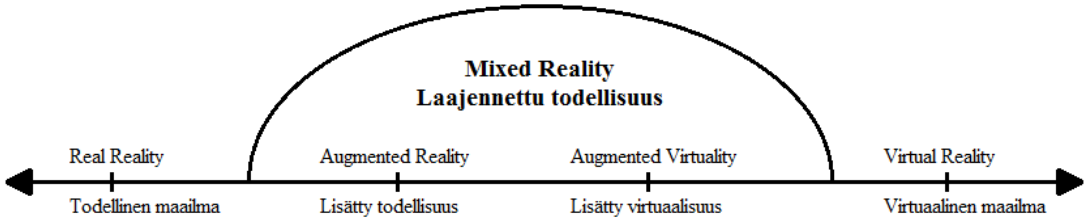
2.1. Tausta

Moni on voinut tutustua lisättyyn todellisuuteen esimerkiksi pelien kautta, sillä mobiilipelien kehitykseen ja pelaamiseen lisätty todellisuus soveltuu erittäin hyvin. Varsinkin kun useimmilla ihmisillä on älypuhelin, joka pystyy suorittamaan lisätyn todellisuuden sovelluksia vaivatta. Viihteen lisäksi lisättyä todellisuutta voi hyödyntää vähentämään ihmisen muistin kuormitusta sekä yksinkertaisia tehtäviä suorittaessa. Esimerkiksi ajoneuvon ajaminen helpottuu, kun lisäinformaatio nopeuksista, varoituksista, säästä ja onnettomuuksista on koska tahansa kuskin luettavissa.

Muunlaisia sovelluskohteita on tunnistusta ja erottelua vaativat tehtävät, jolloin tietty kohde korostetaan muusta ympäristöstä työn helpottamiseksi tai -mahdollistamiseksi. Huomattava asia lisätyn todellisuuden tulevaisuudessa on esineiden internet (engl. Internet of Things, IoT). Tämä tarkoittaisi, että laitteet eivät vain vaikuttaisi keskenään, mutta myös ihmisen kanssa, antaen käyttäjälle reaaliaikaista tietoa niiden toiminnasta ja tehtävistä. Oikean käyttöliittymän avulla ihminen voisi ohjata esineiden toimintaa etäältä esimerkiksi pelkästään katseensa avulla.

Yleisesti käytetyssä määritelmässä lisätystä todellisuudesta on kolme ominaisuutta: todellisen ja virtuaalisen yhdistäminen, interaktiivisuus sekä kolmiulotteinen ilmaisu [4]. Lisätty todellisuus yhdistää todellisen ja virtuaalisen maailman. Se on yksi laajennetun todellisuuden (engl. *mixed reality*) alakategoria ja sijaitsee kuvassa 1 esitetyssä todellisuus - virtuaalisuus jatkumossa lähempänä todellista maailmaa, kuin

virtuaalista maailmaa [5]. Lisätyssä todellisuudessa näkymään todellisuudesta tuodaan tietokoneen tuottamaa virtuaalista dataa, eli esimerkiksi live-kuvaa ihmisestä voidaan käsitellä piirtämällä hattu pään päälle, jolloin tehostetun- ja todellisen kuvan ainoana erona on virtuaalinen hattu [5]. Augmentoiduille teknologioille teknologisen kehityksen vauhti on huima sen eri osa-alueilla; kannettavien laitteiden graafinen suorituskyky, laskentateho ja sensorien monipuolinen käyttö yhdessä muiden spesifikaatioiden kanssa kehittyvät jatkuvasti. Tähän on vaikuttanut suuryritysten, kuten Microsoftin, Googlen ja Facebookin, tekemät suuret sijoitukset lisätyn todellisuuden tutkimukseen ja kehitykseen vuodesta 2009 eteenpäin [6].



Kuva 1. Todellisuus - virtuaalisuus jatkumo.

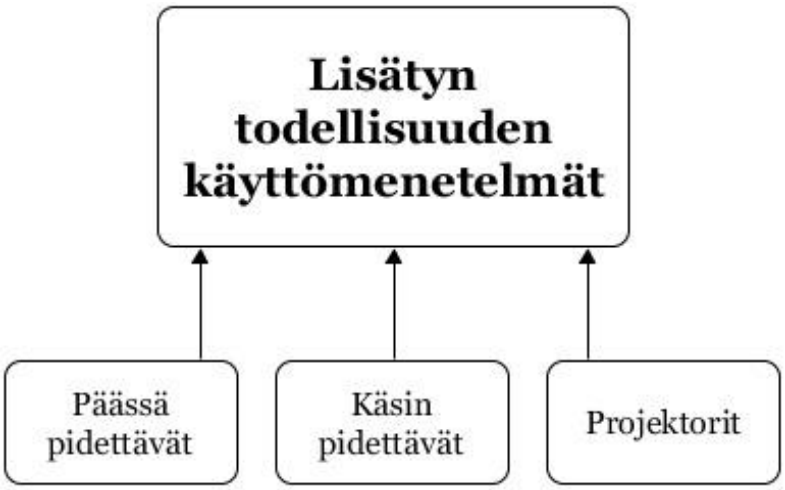
2.2. Laitteisto

Lisätty todellisuus koostuu ainakin kolmesta peruslaitteistokomponentista: sensoreista, prosessoreista ja näytöistä. Prosessorien järjestelmäarkkitehtuurivaihtoehtoja on useita. Keskeisimmät vaihtoehdot ovat: tietokoneet, tietokoneet yhteydessä ulkoiseen serveriin, kannettavat laitteet, kannettavat laitteet yhteydessä ulkoiseen serveriin sekä web ja pilvipohjaiset applikaatiot. Suurin rajoittava tekijä eri vaihtoehdoille on verkon saantiviive (network latency). Tämä tulee esille erityisesti kannettavilla laitteilla, joilla on heikompi suorituskyky, ja tämän johdosta tarvitsevat usein yhteyden taustajärjestelmään tiedon käsittelyä ja kuljetusta varten [7 p.82, 90].

Todellisuuden havaitsemiseen käytetään erilaisia sensoreita monenlaisiin käyttötarkoituksiin, muun muassa tiedonkeruuseen ympäristöstä ja paikan määrittämiseen. Tyypillisesti sensorit havaitsevat valoa, ääniaaltoja tai elektromagneettisia aaltoja, joiden tuottamaa informaatiota lisätyn todellisuuden sovellus käyttää tarkoituksiinsa. [7 p.82, 90]

Todellisuuden kuvantamiseen voidaan käyttää kolmea eri menetelmää kuvan 2 mukaisesti. Päässä pidettäviä näyttöjä eli HMD (engl. *Head-Mounted Displays),* kädessä pidettäviä näyttöjä, esimerkiksi mobiililaitteet, sekä projektorit. Suurimpia haasteita näissä on ollut käyttäjän katseen seuranta ja sen vaikutus virtuaalisen datan kohdistamisessa. Esimerkiksi jos näytölle tuotetaan näkyvän rakennuksen nimi tiettyyn kohtaan ja käyttäjä nostaa katsettaan, ilman että rakennus katoaa kuvasta, kyseinen nimi ei saisi vaihtaa paikkaa. Toinen suuri haaste varsinkin HMD:n kanssa on miten mahdollistaa käyttäjän interaktio [8]. Yksi viimeaikaisista ratkaisuista tähän ongelmaan tuotti Google Glass, joka julkaistiin ensimmäisen kerran vuonna 2012, jossa käyttäjällä oli kosketuspainikkeita ja ääniohjaus joilla pystyi reagoimaan

tuotetun näkymän kanssa [9]. Yksi ratkaisu sovelluksen ohjaamiseen on esittää käyttöliittymässä painikkeita, joita käyttäjä pystyy sormellaan osoittamaan kameralle valitakseen haluamansa toiminto [10].



Kuva 2. Lisätyn todellisuuden käyttömenetelmät.

2.3. Sovelluskohteet

Lisätyllä todellisuudella on toistaiseksi monia eri sovelluskohteita. Teoriassa sitä voisi käyttää kaikkeen ihmisen tekemiseen. Ensimmäiset paljon käytetyt sovelluskohteet olivat lähinnä informaation lisäämistä kuvasyötteeseen. Nykyään sovelluksia löytyy esimerkiksi pelialalta, lääketieteestä, teollisuudesta, arkkitehtuurista ja sotateollisuudesta. Sovellusten käyttötarkoitus vaihtelee alasta riippuen.

Ensimmäisiä ja yksinkertaisimpia sovelluskohteita olivat esimerkiksi urheilutapahtumasta lähetettävään live-kuvaan liitetty tieto ottelun tilanteesta tai urheilijan henkilökohtaisia suoritustilastoja [11]. Ensimmäisiä HMD-sovelluksia olivat EyeTap, jonka ensimmäisiä versioita syntyi jo 1980-luvulla. EyeTap on silmälasien tapainen laite joka kuvaa silmälle tulevan näkymän, lähettää sen tietokoneelle tehostettavaksi ja lopulta heijastaa prosessoidun kuvan takaisin silmälle.

HMD:n ja yleisesti lisätyn todellisuuden sovellusten yleistymiseen on vaikuttanut komponenttien pienentyminen, jolloin laitteita on saatu pienempään, kevyempään ja käyttäjäystävällisempään kokoon [12]. Toinen tärkeä sovellus on ensimmäisiä lisätyn todellisuuden järjestelmiä oleva Virtual Fixture, joka kehitettiin 1990-luvulla [13]. Järjestelmä auttaa kauko-ohjattavan robotin käyttöä. Kyseisessä järjestelmässä käyttäjälle näytetään esimerkiksi HMD:n avulla mihin kohtiin pitää osua tai mitä kohtia pitää välttää. Siinä järjestelmä auttaa parantamaan ihmisten suorituskykyä suorissa ja kauko-ohjattavissa tehtävissä sensoreiden avulla.

Vuodesta 2015 lähtien peliteollisuus on astunut lisätyn todellisuuden alueelle. *SpecTrek* on vuonna 2015 julkaistu laajennetun todellisuuden peli, joka sisältää lisätyn todellisuuden toteutuksen yhtenä pelin ominaisuuksista. Pelissä pelaajat jahtaavat

haamuja, jotka ilmestyvät tiettyihin koordinaatteihin. Pelaajan pitää siirtyä kyseiseen paikkaan ja napata haamu, joka ilmestyy kameralla katsottaessa ja napata se [14]. Samalla periaatteella toimiva *Pokemon Go* nousi maailmanlaajuisesti kuluttajien tietämykseen vasta 2016 [15].

Lääketieteessä lisätyn todellisuuden sovelluksia käytetään suurimmaksi osaksi harjoitteluun. Useita erilaisia simulaattoreita ja virheiden tarkkailijoita erilaisten leikkausten suorittamiseen on kehitetty [16]. Suunnittelussa ja tuotannossa lisätyn todellisuuden käyttö on kasvavassa asemassa. 3D-prototypointi ja useita kerroksia sisältävän tuotteen eritteleminen eri kerroksiin ovat jo toteutettuja ja tutkinnan alla olevia käyttökohteita. Sovelluksia löytyy esimerkiksi robotiikasta, missä mm. robotille etsitään turvallisia liikeratoja [17].

Armeijan käytössä HMD teknologia on ollut jo pitkään. Lentäjien kypärien visiireissä on näytetty tietoa lentokoneen tai helikopterin tärkeistä tiedoista tai aseiden tähtäin on näytetty visiirissä ja näin parantanut tarkkuutta. Visiiriin on myös heijastettu lentäjän näkökentän ulkopuolella oleva reaalimaailma luoden lentäjälle täydellisen näkökentän [18].

Samanlaista sovellus tyyppiä on siirtynyt myös autoteollisuuteen, jossa auton tuulilasiin heijastetaan nopeus ja navigointiohjeet. Kehittyneimmissä järjestelmissä tuulilasiin merkataan näkyviä jalankulkijoita, annetaan informaatiota ja palautetta ajolinjasta ja tyylistä. Tekniikasta käytetään nimeä "heads-up display" eli HUD [19].

Arkkitehtuurissa ja turismissa lisätyn todellisuuden sovelluksia on myös käytössä. Arkkitehtuurista suunnittelua voidaan sovellusten avulla tehdä paikan päällä, jos tarpeena on lisätä rakenteita aikaisempaan tai virtuaalinen malli voidaan heijastaa suunnitteluhuoneessa missä sitä voidaan käsin muokata ja tehdä helposti yhteistyönä. Turismissa sovellukset voivat kertoa mielenkiintoista historiallista tietoa paikoista joita ihmiset katselevat [20, 21].

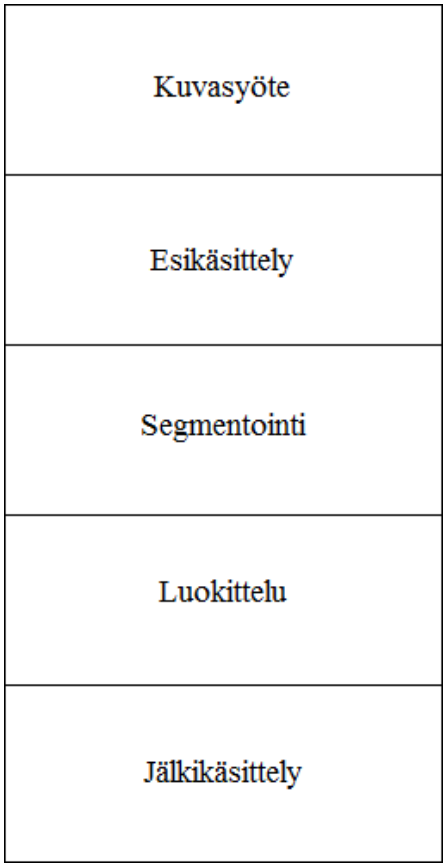
Yksi projektorilla toteutettu sovellus on *SixthSense*, jossa käyttäjä pystyy vuorovaikuttamaan sovelluksen kanssa käden liikkeillä. Projektori on pieni päällä pidettävä laite, joka heijastaa virtuaalisen datan käyttäjän eteen. Käyttäjä pystyy esimerkiksi lähentämään seinälle heijastettua dokumenttia loitontamalla käsiään [22].

Opetuksessa ja kirjallisuudessa lisättyä todellisuutta hyödynnetään herättämällä kirjassa näkyvät kuvat ja tekstit eloon luoden niistä 3D-mallin, jonka käyttäjä näkee laitteen läpi. Malliin pystytään lisäämään myös ääntä, mikä tekee luku- ja opiskelukokemuksessa hauskempaa ja mielenkiintoisempaa. [23]

Tulevaisuudessa lisättyä todellisuutta voidaan käyttää mittaamiseen ja tehokkaaseen mittausdatan esittämiseen. Myös älypuhelinten ja tablettien tuominen ja aseman vahvistuminen lisätyn todellisuuden laitteina, tulevat muuttamaan sovelluksia yhä helppokäyttöisemmiksi ja lähemmäksi arkielämää. [24]

2.4. Kuvankäsittelyprosessi

Lisätyn todellisuuden sovellusten tehostetun kuvan tuottamiseen vaaditaan kameralla kaapatun kuvan käsitteleminen. Kuvankäsittelyprosessi on muiden konenäön sovellusten kanssa samanlainen ja se seuraa samoja askelia kuin digitaalinen kuvankäsittely. Kuvankäsittelyprosessiin kuuluvat ainakin seuraavat vaiheet: alkuperäisen kuvan lukeminen, esikäsittely, kuvan segmentointi, objektien luokittelu ja jälkikäsittely (Kuva 3) [25].



Kuva 3. Kuvankäsittelyn vaiheet tunnistuksessa.

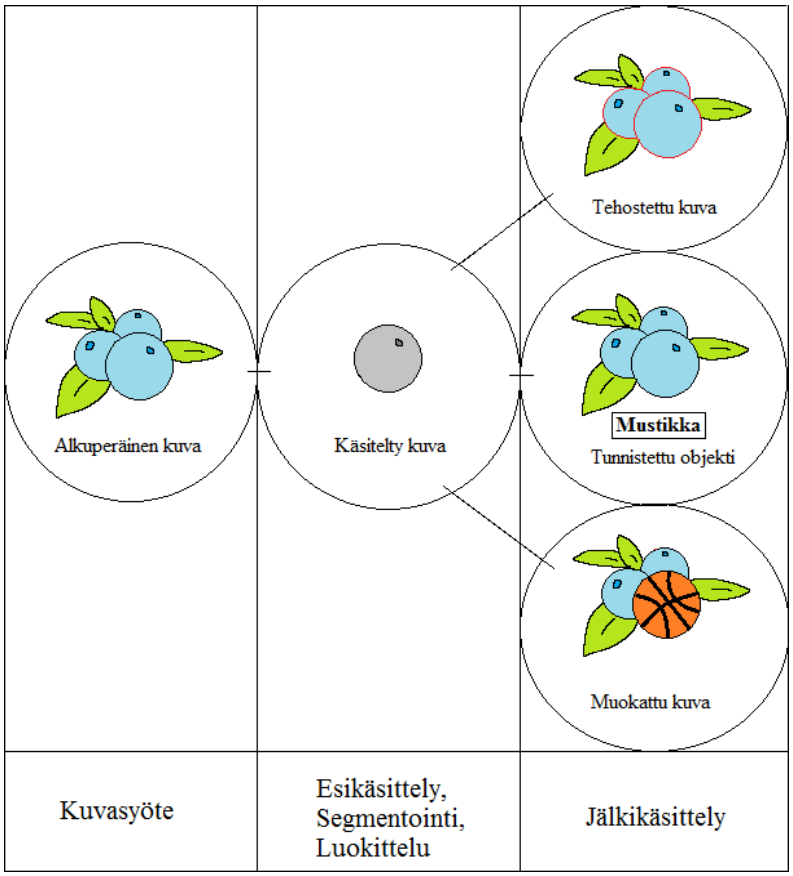
Alkuperäinen kuva saadaan systeemin sensoreiden kautta, jotka tallentavat esimerkiksi valon tai äänen kaksi- tai kolmiulotteisena kuvana. Kuvaa voi sitten esikäsitellä paikkatasossa tai taajuustasossa [26 p. 1-215]. Esikäsittelyssä kuvaa käsitellään prosessoinnin helpottamiseksi. Tarkoituksena on saada kuva käsiteltävään muotoon, eli poistaa kuvasta haitalliset häiriöt, muokata kuvan intensiteettiä, terävöittää tai tasoittaa kuvaa. Vastaavasti äänenkäsittelyssä ihmispuheen histogrammin tasoittaminen parantaa puheentunnistusta, kun värihistogrammin käsitteleminen parantaa kuvan tunnistusta [27].

Esikäsiteltyyn kuvaan voidaan suorittaa seuraavaksi segmentointi. Segmentointi on prosessi, joka jakaa kuvan eri rakenneosiin, kuten taustaan ja objekteihin [28]. Kuvan segmentoinnin kaksi päämetodia ovat kerrospohjainen segmentointi, missä objektit

jaetaan muodon ja syvyyden perusteella, ja lohkopohjainen segmentointi, jolloin kuva erotellaan värin tai reunojen perusteella [28]. Siinä tapauksessa, kun esillä olevia objekteja on useita, voidaan suorittaa halutun objektin tai objektien tunnistus. Objektin tunnistamiseen on kehitetty erilaisia menetelmiä; kuten reunojen-, kulman- ja alueen tunnistusta [29]. Iso ongelma objektin tunnistuksessa on objektin ääriviivojen erittely objektin tekstuurista. Yksi mahdollinen ratkaisu on yhdistää kulmat ja objektin kiinnostuspisteet [30].

Etenkin sellaisissa AR-sovelluksissa, joissa tietyn objektin päälle piirretään reaaliajassa, voidaan sijoittaa oikeaan maailmaan haluttu erikoisuus eli marker. Markerin tunnistukseen voi käyttää muun muassa seuraavia menetelmiä:

1. QRCode, joka on kaksidimensionaalinen viivakoodi,
2. Kalman filter,
3. ennustetaan markerin liikettä dynaamisessa ympäristössä,
4. kynnystäminen, epätavallisen värinen marker on helppo erottaa muusta kuvasta värin perusteella tai
5. erityisiä algoritmeja markerin seuraamiseen kuten Natural Feature Tracking [31].



Kuva 4. Jälkikäsittely.

Jälkikäsittely vaihtelee sovelluksen käyttötarkoituksesta riippuen. Jälkikäsittely on esimerkiksi kuvan tehostamista, objektin tunnistamista ja kuvan muokkaamista, kuten

kuvassa 4 on esitetty. Eristetyn objektin kuvanlaatua voi vielä parantaa ja esimerkiksi piirtää huonossa valaistuksessa olevan todellisen objektin ääriviivat tehostettuun kuvaan. Markerin tunnistaminen piirteiden avulla vaatii neljän tehtävän suorittamista [32]:

1. piirteiden tunnistus,
2. piirteiden vertaus,
3. muutosmallin estimointi ja kuvanmuunnos sekä
4. uudelleennäytteistys.

Tämän jälkeen tehostettuun kuvaan lisätään tunnistetusta objektista informaatiota tai vain kirjataan tietokantaan uusi merkintä havaitusta objektista. Toisissa sovelluksissa voidaan piirtää tehostettuun kuvaan markerin paikalle valmiiksi luotu digitaalinen kuva tai -animaatio.

3. SULAUTETUT JÄRJESTELMÄT

Sulautetut järjestelmät erotellaan yleiskäyttöisistä tietokoneista niiden käyttötarkoituksen, tunnusmerkkien ja tiukkojen suunnitteluvaatimusten perusteella. Tunnusmerkkejä sulautetuilla järjestelmillä on yksi tehtävä, tiukat rajoitukset ja reaaliaikaisuus. Sulautettuja järjestelmiä suunniteltaessa joudutaan tekemään valintoja tehon, koon, hinnan ja suorituskyvyn välillä. Nykyisistä sulautetuista järjestelmistä yksi tunnetuin ja käytetyin on Rasperry Pi. RP:tä on myös käytetty useissa kasvojentunnistussovelluksissa.

3.1. Määritelmä

Tietokonejärjestelmät ovat nykypäivänä yleinen näky. Lähes kaikki omistavat tietokoneen tai älypuhelimen ja useimmat käyttävät niitä päivittäin. Toinen tietokonetyyppi, jota ihmiset varmasti käyttävät päivittäin on sulautetut järjestelmät. Sulautettuihin järjestelmiin kuuluvat pääosin kaikki tietokonejärjestelmät, jotka eivät ole pöytätietokoneita, kannettavia tietokoneita, älypuhelimia, tablet-tietokoneita tai palvelintietokoneita.

Tämä merkitsee, että arkipäivän esineet, kuten laskimet, kamerat, mikroaaltouunit, termostaatit, pesukoneet, printterit, elektroniset maksujärjestelmät, hälytysjärjestelmät ja lähes kaikki autoissa olevat ominaisuudet kuuluvat sulautettuihin järjestelmiin. Voi väittää, että kaikissa elektronisissa laitteissa on tai tulee olemaan jonkin asteinen sulautettu tietokonejärjestelmä. Vaikka pöytävalossa tai vedenkeittimessä ei vielä olekaan varsinaista tietokonejärjestelmää ohjaamassa toimintaa, niin IoT voi muuttaa tämänkin asian tulevaisuudessa.

3.2. Tunnusmerkit

Sulautetuilla järjestelmillä on keskenään muutama yhteinen erityispiirre [33 p. 5-8].

1. *Yksi tehtävä*: sulautettu järjestelmä suorittaa yhtä ohjelmaa toimintansa aikana. Tämä piirre näkyy hyvin siinä, että funktiolaskimella suoritetaan laskutoimituksia ja e-lukulaitteella luetaan sähköisiä kirjoja, mutta älypuhelimella voi tehdä nämä molemmat tehtävät vaivatta.

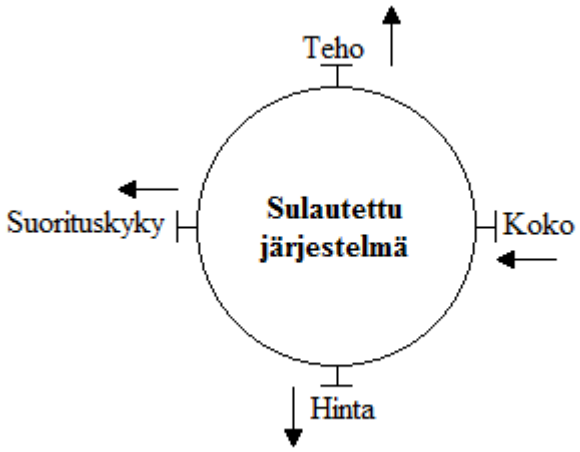
2. *Tiukat rajoitukset*: kaikilla tietokonejärjestelmillä on omat suunnitteluvaatimuksensa, mutta sulautetuilla järjestelmillä ne ovat erityisen tiukat. Sulautetun järjestelmän on usein tasapainoteltava neljän rajoituksen kanssa: hinta, koko, suorituskyky ja teho. Sulautettujen järjestelmien on lähes poikkeuksetta oltava halpoja ja pieniä. Silti niiden on oltava tarpeeksi tehokkaita prosessoimaan dataa reaaliajassa ja säilyttämään pitkä käyttöaika yhdellä paristolla tai akun latauskerralla.

3. *Reaktiivinen ja reaaliaikainen*: monien sulautettujen järjestelmien on jatkuvasti reagoitava niiden ympäristössä tapahtuviin muutoksiin ja laskettava näiden muutosten aiheuttamat seuraukset järjestelmän tilaan. Esimerkiksi valvontajärjestelmän on oltava jatkuvasti aktiivinen ja valmis reagoimaan ei-toivottuihin muutoksiin ollakseen hyödyllinen tehtävässään.

3.3. Suunnitteluvaatimukset

Sulautettua järjestelmää suunniteltaessa aina tärkeintä on sille annetun tehtävän onnistunut toteuttaminen. Vaikeaa tästä kuitenkin tekee se, että sulautetuilla järjestelmillä on joukko muitakin suunnitteluvaatimuksia, jotka on myös täytettävä. Tästä syystä on tärkeää heti suunnittelun alkuvaiheessa etsiä parhaita ratkaisuja ja optimoida laitteistoa, ohjelmistoa ja itse järjestelmän toteutusprosessia. Kehitykseen ja markkinointiin kuluvan ajan ja materiaalien hinnan lisäksi itse lopullisen järjestelmän suorituskyky, teho ja koko rajaavat suunnittelupäätöksiä merkittävästi. Kaiken tämän lisäksi on myös huolehdittava, että lopullinen systeemi ei vahingoitu käytön aikana ja että tietoturva on kunnossa.

Nämä vaatimukset ovat usein toisistaan riippuvaisia siinä mielessä, että esimerkiksi laitteiston tehoa lisäämällä sen hinta kasvaa, ja suorituskykyä kasvattamalla usein kokokin kasvaa (Kuva 5) [33 p. 7].



Kuva 5. Sulautettujen järjestelmien suunnitteluarvojen tasapainottelu - korottamalla yhtä toinenkin voi kasvaa.

Osan vaatimuksista voi ratkaista esimerkiksi käyttämällä valmiiksi kehitettyjä piirilevyjä järjestelmän laitteistona. Seurauksena on, että laitteiston hinnan, koon ja tehon ollessa kiinnitettyinä valmiiksi, voi kehittäjätiimi keskittyä itse ohjelmiston suorituskykyyn ja sen optimointiin.

3.4. Raspberry Pi sulautettuna järjestelmänä

Valmiita piirilevyjä, kuten Raspberry Pi:tä, käytettäessä järjestelmän ketteryysaste on erittäin korkea, koska tietokoneet ovat nimenomaan suunniteltu laajaan alaan sovellusmahdollisuuksia. Huonona puolena on kuitenkin, että ne eivät aina sovellu juuri tiettyyn tehtävään. Tietokoneiden on myös mahdollista olla huonosti optimoituja tarkoitettua tehtävää varten, jolloin ne ovat joko tarpeettoman tehokkaita tai päinvastoin. Ongelmaksi voi myös nousta järjestelmän markkinointimahdollisuudet, koska valmiit piirilevyt voivat olla liian kalliita massatuotantoon. Valmiit piirilevyt soveltuvatkin parhaiten harrastelijoiden käyttöön.

Raspberry Pi on suosittu yhden piirin tietokonealusta, jota oli jo myyty yli kymmenen miljoonaa kappaletta vuonna 2016 [34]. Raspberry Pi laitteet ovat Raspberry Pi Foundation -hyväntekeväisyysjärjestön valmistamia edullisia tietokonealustoja. Raspberry Pi Foundation on julkaissut säännöllisin väliajoin sekä aivan uusia että myös päivitetettyjä versioita vanhoista malleistaan. RP alustoista on viisi erillistä mallia: Raspberry Pi Zero, Raspberry Pi Model 1 A+, Raspberry Pi Model 1 B+, Raspberry Pi 2 Model B sekä Raspberry Pi 3 Model B [35]. RP oli alunperin suunnattu kaikenikäisille lapsille mahdollistamaan uudet sukupolvet oppimaan tietotekniikasta Isossa-Britanniassa tuomalla edullisen tietokonealustan markkinoille [36].

Raspberry Pi Foundation julkaisee tiheään uusia versiota vanhoista malleista sekä aivan uusia malleja. Ensimmäisen RP mallin julkaisun 29.2.2012 ja vuoden 2017 vaihteen välillä, RP laitteita on julkaistu 12 eri mallia, joista on myynnissä viisi eri mallia. Loput seitsemän julkaisua ovat olleet aiempia malleja korvaavia malleja. Vanhempia RP:n malleja on parannettu tehokkaimmilla osilla samalla säilyttäen RP:n eri mallien hinnan ennallaan.

RP:n myynnissä olevat eri mallit poikkeavat toisistaan prosessointitehonsa, muistinsa ja sisäänrakennettujen I/O-osiensa puolesta, joiden johdosta virrankulutus, hinta ja koko vaihtelevat suurestikin eri mallien välillä, ja siten ne soveltuvat erilaisiin käyttötarkoituksiin. Sisäänrakennettuina uudemmissa malleissa on mm. ethernet-portti, wifi ja bluetooth. Kaikissa RP-malleissa on sama lukumäärä pinnejä: 40 kappaletta. Aiemmin myynneissä olleissa RP-malleilla on ollut pienempi lukumäärä pinnejä, mutta näistä on julkaistu uudemmat versiot useammalla pinnien lukumäärällä. Taulukossa 1 on tarkemmin esitettyinä teknisiä tietoja RP-laitteista [35, 37].

Taulukko 1. Raspberry Pi-mallien ominaisuuksien vertailu

Laite	Hinta	CPU	RAM	GPIO-määrä	Sähkö-virta	Ethernet-portti	Wifi, Blue-tooth	USB portit
RP Zero	$5	Single Core, 1 Ghz	512 MB	40	160 mA	No	No	1
RP 1 Model A+	$20	Single Core, 700 MHz	512 MB	40	200 mA	No	No	1
RP 1 Model B+	$25	Single Core, 700 MHz	512 MB	40	700 mA	Yes	No	4
RP 2 Model B	$35	Quad Core, 900 MHz	1 GB	40	800 mA	Yes	No	4
RP 3 Model B	$35	Quad Core, 1.2 GHz	1 GB SDRAM, 400 MHz	40	800 mA	Yes	Built in	4

3.5. Valmiiden piirilevyjen sovelluksia

Raspberry Pi:lle on tehty aikaisemmin kasvojentunnistussovelluksia. Sovellukset voidaan jakaa kahteen kategoriaan: pysyvään kuvaan ja liikkuvaan kuvaan tapahtuvaan tunnistukseen. Kumpaa tapaa sovellus käyttää riippuu sovelluksen käyttötarkoituksesta.

Nikisins ym. (2015) tekivät kasvojentunnistussovelluksen Raspberry Pi 1 B+ mallille [38]. Kuvankokona he käyttivät 320x240 pikseliä, jolla tunnistus sovellus toimi noin 9 FPS nopeudella. Heidän tunnistus algoritminsa toimi noin 99,3 % tarkkuudella. Toinen esimerkki sovellus on Patil ja Shuklan (2014) tekemä läsnäoloa tarkkaileva systeemi, joka tunnisti luokassa olevat henkilöt Raspberry Pi:hin kiinnitetyn kameran avulla [39]. Heidän tekemänsä järjestelmä käytti Viola-Jonesin algoritmia eli Haar-like-piirteitä kasvojen tunnistamiseen.

Janard ja Marurngsith (2015) tutkivat Raspberry Pi:tä käyttävien telemaattisesti läsnäolevien robottien kasvojentunnistusta ja niiden nopeutta [2]. He saivat keskiarvolliseksi tulokseksi 16,7 FPS nopeuden 320x240 pikselin kuvakoolla.

4. SOVELLUSTEN SUORITUSKYKY JA OPTIMOINTI

Sovelluksen suorituskykyyn vaikuttaa sen suoritusympäristö ja toteutuksessa tehdyt ratkaisut. Sovellus käyttäytyy eri tavalla eri ympäristöissä. Lisäksi saman tehtävän voi tehdä eri vaihtoehdoilla, joista osa ovat tehokkaampia kuin toiset.

Ohjelmiston suoritusympäristö jakautuu käytetyn laitteiston tehoon, verkkoyhteyden laatuun ja turvallisuuteen. Ympäristöön kuuluvat myös laitteistossa ajettava käyttöjärjestelmä ja esimerkiksi verkkosovellusten eri selaimet.

Sovellusta optimoidaan ratkaisemalla yksi pullonkaula kerrallaan. Optimoinnin helpottamiseksi sovelluksen voi jakaa sovelluspalvelimeen, verkkopalvelimeen ja tietokantapalvelimeen. Eri palvelimista tunnistetaan mahdollinen pullonkaula, eristetään se esimerkiksi profilointityökalulla ja toteutetaan parempi ratkaisuvaihtoehto, joka parantaa sovelluksen suorituskykyä. Vaihtoehtoisesti sovellus voidaan kehittää alusta loppuun suorituskyky tarkkailun kohteena, jolloin sen suorituskyvyn tiedetään riittävän suunniteltuun tehtävään.

4.1. Sovelluksen suoritusympäristö

Sovelluksen suoritusympäristö jaetaan laitteistoon, verkkoon ja turvallisuuteen. Laitteisto merkitsee tietokonetta joko yleiskäyttöisenä- tai sulautettuna järjestelmänä. Tietokoneen ytimeen, emolevyyn, liitetään käyttötarkoituksesta riippuen ajoon vaadittavat komponentit kuten virtalähde, prosessori, muisti, lukulaitteet ja näyttölaitteet, joilla kaikilla on vaikutus koko järjestelmän tehokkuuteen. Suoritusympäristön resursseilla tarkoitetaan tiedon prosessointiin, siirtämiseen, vastaanottamiseen ja tallentamiseen vaadittavia komponentteja.

Sulautetut järjestelmät, kuten pesukoneet tai mikroaaltouunit, suunnitellaan täyttämään yksi tietty tehtävä mahdollisimman tehokkaasti. Sulautetuissa järjestelmissä on usein rajattu siihen liitettävien komponenttien koko, teho, hinta ja täten myös suorituskyky. Laitteiden liitännäisiin voivat kuulua esimerkiksi yksinkertainen ohjauspaneeli ja äänisummeri.

Sen sijaan yleiskäyttöiset tietokoneet pystyvät suorittamaan useita erilaisia ohjelmia samanaikaisesti. Yleiskäyttöisiin tietokoneisiin voi luokitella pöytätietokoneet ja älypuhelimet. Näiden tietokoneiden suorituskyky voi vaihdella laidasta laitaan esimerkiksi käyttäjän tarpeiden mukaan. Voi olla, että pöytätietokoneessa tietty verkkoselain toimii moitteettomasti, kun taas kannettavassa tietokoneessa sama versio selaimesta tökkii ja jumiutuu puhtaasti laitteiston tehon perusteella.

Tietokoneiden komponenttien yhteistoiminnan mahdollistava varusohjelma, kuten BIOS (engl. Basic Input-Output System), ja erityisesti yleiskäyttöisen tietokoneen käyttöjärjestelmä rajoittavat tietokoneella ajettavia sovelluksia. Käyttäjä voi itse vapaasti asentaa haluamansa käyttöjärjestelmän, mutta laitteiston ollessa identtinen, jotkin sovellukset toimivat vain tietyllä käyttöjärjestelmällä. Saman yhtiön tuottama ohjelmistotuote voi olla myös paljon paremmin optimoitu toimimaan tietyssä käyttöjärjestelmässä paremmin kuin toisessa, vaikka molemmissa voi ohjelmistoa

käyttää. Sovelluksen voi katsoa olevan hyvin optimoitu, kun se suoriutuu tehokkaasti suoritusympäristöstä riippumatta.

Verkkosovellusten suorituskykyyn vaikuttaa erityisesti käytössä olevan verkon nopeus. Erämaassa langattoman tietoliikenneverkon ollessa heikko on mahdollista, että verkkosovellus ei toimi ollenkaan, mutta kaupunkiin siirryttäessä voi sovellus taas toimia tarkoituksenmukaisesti.

Käyttöympäristön turvallisuus on merkittävää siinä mielessä, että tietokoneen käyttäjän tai ohjelman ollessa epätoivottu voi syntyä haastava ympäristö normaalille toiminnalle. Epäturvallisessa ympäristössä hyökkääjän on mahdollista kaapata informaatiota ja tietokoneen resursseja etäältä aiheuttaen esimerkiksi systeemin heikentyneen suorituskyvyn.

4.2. Optimointi

Sovelluksen yksi tärkeimmistä ominaisuuksista käyttäjän kannalta on käyttöliittymän suorituskyky, eli kauanko aikaa kuluu käyttäjän syötteen ja palautteen välillä ja kuinka nopeasti sovellus suorittaa kunkin tehtävänsä. Sovelluksen prosessiketjun hitain komponentti toimii pullonkaulana, mikä rajoittaa koko prosessin suoritusaikaa. Pullonkaula voi olla esimerkiksi tehoton arkkitehtuuri, tiedonsiirron hidas kulku tai huono algoritmi. Käyttöliittymän suorituskykyyn vaikuttaa sovelluksen suoritusympäristö ja sovelluksen toteutuksessa tehdyt ratkaisut. Sovellusta voi optimoida muuttamalla tai parantamalla sovelluksen suorituskykyyn vaikuttavia ratkaisuja. Sovelluksen koko vaihtelee yksittäisestä ohjelmasta suuriin ohjelmistoihin. Suurissa ohjelmistoissa on usein erillinen sovelluspalvelin ja tarvittaessa myös tietokantapalvelin tai web-palvelin, jotka voidaan erottaa toisistaan optimoimisen helpottamiseksi.

Sovelluksen optimointiin vaadittavat tehtävät voidaan jakaa kolmeen askeleeseen: **tunnista, eristä ja ratkaise**. Käsittelemällä yksi pullonkaula tai ongelma kerrallaan päästään aina lähemmäksi tehokkaampaa järjestelmää.

4.2.1. Tunnista

Pullonkaulojen, oireiden ja ongelmien löytäminen tapahtuu usein normaalin käytön tai erilaisten testien avulla. Käytännön esimerkkinä uuden toiminnon lisääminen ohjelmistoon vaikuttaakin koko sovelluksen tavalliseen suoritukseen ilman selvää syytä.

Sovelluksen suorituksen aikana esimerkiksi prosessorin korkea käyttöaste, prosessorin suuri odotusjono tai prosessorin tapahtumat kasvavilla vasteajoilla ovat merkkejä pullonkaulasta. Syynä oireisiin voivat olla ohjelmiston tehoton arkkitehtuuri tai heikko toteutus, jotka ilmenevät muun muassa objektien tarpeettomana luomisena ja tuhoamisena, ohjelmiston huonona skaalautuvuutena, tehottomina algoritmeina tai liiallisina kyselyinä ja tarpeettoman suuren tiedon siirtelynä yhteistä väylää pitkin. Käyttöliittymässä sovelluksen huono arkkitehtuuri ja heikko toteutus näkyvät

esimerkiksi kuvan epätasaisena päivityksenä, kun lenkin tehottomat osat hidastavat odotettua toimintaa.

Oireilevan tietokantapalvelimen merkkejä voivat olla korkea prosessorin käyttöaste, suuret jonot ja paljon levyhakuja. Syynä tietokannan esittämiin oireisiin yleensä ovat tehoton SQL (engl. Structured Query Language), eli tietokannan kyselykielen käyttäminen tarpeettomiin hakuihin ja muutoksiin, huonot tietokannan indeksit tai huono tietokantatuotteen parametrointi. Ollakseen tehokas on tietokannan lähetettävä pyydetyt tiedot pyydettyyn paikkaan mahdollisimman nopeasti. Käyttöliittymässä tietokantapalvelimen vajaatoiminta vaikuttaa esimerkiksi käyttäjän tekemiin hakuihin ja tulosten palautusaikoihin, hakutulosten osuvuuteen hakusanan perusteella ja jopa hakutulosten epätoivottuun paljouteen tai vähyyteen.

Verkkopalvelimessa ongelmat saattavat ilmetä matalana prosessorin käyttöasteena silloin, kun verkkovirheitä ilmenee tai havaitaan korkea verkon kuormitus. Näiden syynä voivat olla usein verkkolaitteiden huono konfigurointi, ISP-palveluiden (engl. Internet Service Provider) ongelmat ja luvattua huonompi verkon nopeus tai muusta syystä johtuva hidas verkkoyhteys. Käyttöliittymässä verkko-ongelmat aiheuttavat aina siirrettävän tiedon pidennettyä odottelua tai heikentynyttä laatua, kun osa tiedosta katoaa.

4.2.2. Eristä

Ongelmien eristämisestä hankalan tekee ohjelmiston suunnitteluun-, koodiin- ja suoritusympäristöön tehtyjen ratkaisujen yhteinen vaikutus. Eristämistä helpottaa tarkkailemalla järjestelmän osien toimintaa erikseen mahdollisimman yksityiskohtaisesti. Esimerkiksi tarkkailemalla eri koodinpätkien aiheuttamaa vaikutusta suoritusympäristön resursseihin voi eristää pullonkaulan lähteen.

Yksinkertaisin tapa analysoida koodin suorituskykyä on selvittää suoritettavien lohkojen suoritusajat ja etsiä ketjun hitain lenkki suoritusajan perusteella. Suurin osa sovelluksen suorituskykyongelmista usein on seurausta sovelluksen kehitysvaiheessa tehdyistä laiskoista ratkaisuista. Ensimmäinen löydetty ratkaisu harvoin tuottaa parhaan mahdollisen suorituskyvyn. Kehitysvaiheen huono suunnittelu ja dokumentointi vaikeuttavat sovelluksen eliniässä myöhemmin tapahtuvaa muuttamista ja optimointia.

Toinen tapa eristää ongelmia on käyttää järjestelmän analysointiin tarkoitettuja profilointityökaluja. Profilointityökalujen tehtävä on paljastaa järjestelmän resurssien muutokset ajon aikana. Työkaluja on kehitetty sopiviksi tietyille käyttöjärjestelmille tai ohjelmointikielille.

4.2.3. Ratkaise

Parhaiden ratkaisujen löytäminen vaatii työkalujen tarkoituksenmukaisen käytön ja tulosten analysoimisen. Hyödyllinen tapa etsiä ratkaisuja olisi listata useampi vaihtoehto ja vertailla niiden hyötyjä tai haittoja keskenään, sekä pohtia ratkaisun muutoksen vaikutusta lopulliseen järjestelmään.

Sovelluksen suorituskykyä voi lähes aina nostaa parantamalla ympäristön laatua esimerkiksi lisäämällä muistia, parantamalla verkkoyhteys tai jakamalla sovellus useampaan prosessiin. Tämä menettelytapa ei kuitenkaan ole paras ratkaisu, koska sovelluksen muilla käyttäjillä ei aina ole mahdollisuutta parantaa ympäristön laatua. Huomio on siirrettävä koodiin ja etsittävä parempia ratkaisuja toteuttaa sama toiminto tai vaatimus.

Eräs parantamisen kohde on monesti suuria tietojoukkoja käsittelevät algoritmit. Algoritmien yksi merkittävimmistä ominaisuuksista on niiden aikavaatimus eli kompleksisuus. Algoritmin kompleksisuus tarkoittaa sen suoritusajan, toisin sanoen silmukan toistoihin kuluvan ajan, suhteellista kasvua syötteen kokoon verrattuna. Vakiokompleksisella algoritmilla (O(1)) suoritukseen kestää syötteen koosta riippumatta tietyn vakion verran aikaa. Lineaarisessa algoritmissa (O(N)) syötteen kasvaessa käsittelyaika kasvaa tasaisesti. Niissä tapauksissa joissa kompleksisuus on eksponentiaalisesti kasvava ($O(N^x)$, kun x >= 2), algoritmin suoritusaika kasvaa nopeasti syötteen kokoa suurennettaessa. Sovelluksissa jatkuvasti toistuvien algoritmien aikakompleksisuuden tietäminen on tärkeää, koska ne voivat helposti aiheuttaa sovellukseen pullonkaulan esimerkiksi suuria videokokoja käsiteltäessä.

4.3. Vaihtoehtoiset menetelmät

Erityisesti sulautettuja järjestelmiä kehitettäessä on mahdollista ottaa käyttöön kehitysmenetelmä, jonka avulla on helppoa keskittyä suorituskyvyn vaatimuksiin ja suunnitella samalla kehitysprosessi. Näin tekemällä voi varmistaa esimerkiksi suoritustehon, viiveen ja muistin riittävä toiminta kehityksen alusta loppuun asti. Yksi merkittävä suorituskykyvaatimuksiin kohdistunut ohjelmistotuotannon menetelmä on Software Performance Engineering (SPE). SPE:n olennaiset aktiviteetit ovat seuraavat:

1. Ongelmien identifiointi. Identifioidaan kaikki suorituskykyvaatimuksiin liittyvät tekijät ja niihin liittyvät mahdolliset ongelmat.
2. Vaatimusten määrittely ja analyysi. Luodaan järjestelmästä erilaisia järjestelmäkarttoja, käyttötapausmalleja ja aktiviteettikaavioita, jotta niiden avulla voidaan luoda suorituskykytestejä.
3. Suorituskyvyn ennakointi. Suunnittelusta ja arkkitehtuurista tehtyjen mallien avulla pyritään ennakoimaan resurssien välistä vuorovaikutusta.
4. Suorituskykytestaus. Käytetään testidataa selvittääkseen järjestelmän suorituskyky normaalissa käytössä sekä kovassa stressissä.
5. Ohjelmiston ylläpito ja kasvattaminen. Pyritään ennustamaan järjestelmään tehtävien potentiaalisten muutosten ja lisäysten vaikutus.
6. Täysi järjestelmän analyysi. Analysoidaan lopullista kokonaisuutta. [40]

5. TESTIALUSTA

Testikokoonpano toimii reaaliaikaisena lisätyn todellisuuden sovelluksena, jonka alustana toimii sulautettu järjestelmä. Testausalusta koostuu laitteistosta (Raspberry Pi), kasvojentunnistussovelluksesta ja testiaineistosta. Tietokoneen lisälaitteisiin kuuluvat kamera ja kosketusnäyttö.

Kasvojentunnistussovellukseen on kehitetty yksinkertainen käyttöliittymä, joka mahdollistaa piirrosmaskin valinnan, tehostetun kuvan reaaliaikainen esittämisen, kuvakaappauksen ja sovelluksen sulkemisen. Käytön aikana sovellus tunnistaa ennalta määrätyn etäisyyden rajoissa ihmiskasvoja esimerkiksi metrin tai kahden metrin etäisyydeltä, sekä piirtää kasvojen päälle maskin.

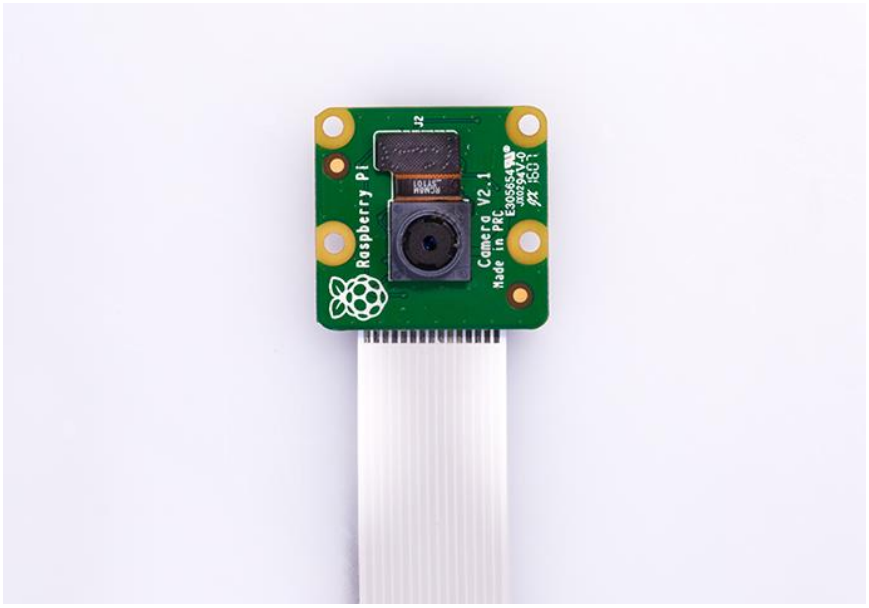
Testiaineisto on jaettu kahta testiä varten kahteen osaan. Ensimmäinen osa aineistosta koostuu videoista, joista kaikista on tehty useampi versio erikokoisilla resoluutioilla. Toisessa osassa aineistosta on yhdellä resoluutiolla kuvattu ihmiskasvot ennalta määrätyin välimatkoin.

5.1. Laitteisto

Raspberry Pi 2 B on yhden piirilevyn tietokone, jossa on 900 MHz neliydinprosessori ja yksi GB keskusmuistia. Tietokoneessa on VideoCore IV 3D -grafiikkasuoritin, joka tukee OpenGL ES (*engl. Embedded Systems*), sulautettuja järjestelmiä varten kehitetty ohjelmointirajapinta tietokonegrafiikan tuottamiseen. Tietokoneeseen voi liittää lisälaitteita neljän USB -portin kautta, tai erillisen näyttölaitteen HDMI -portin avulla, tai äänilaitteen 3,5mm jakkiliittimellä. Tietokoneeseen voi myös liittää lisämuistia Micro SD -korttipaikan avulla, ja yhdistää tietokoneen internetiin Ethernet -portilla (Kuva 6). [35]

Kuva 6. Raspberry Pi 2 model B.

Tietokoneeseen on liitettynä Raspberry Pi kamera (engl. Raspberry Pi Camera V2) moduuli, jolla kuvaaminen tapahtuu. Kamerassa on kahdeksan megapikselin sensori ja se yhdistetään piirilevyyn CSI porttiin, jossa tiedon vaihto nopeus on 4 Gbps. Kamera pystyy tallentamaan HD tasoista videota jopa 1080p30 tai 720p60 resoluutio*kuvataajuus. (Kuva 7) [35]



Kuva 7. Raspberry Pi Camera V2.

Raspberry Pi Kosketusnäyttö (engl. Raspberry Pi Touch Display) on seitsemäntuumainen ja sen natiiviresoluutio on 800x480 pikseliä. Näyttö saa voimansa Raspberry Pi:stä ja näyttö liitetään Raspberry Pi:hin erillisen kaapelin avulla. Näyttö mahdollistaa tietokoneen käytön ilman erillisiä osoitin tai syöttölaitteita näytölle piirrettävän näppäimistön avulla. [35]

5.2. Sovellus

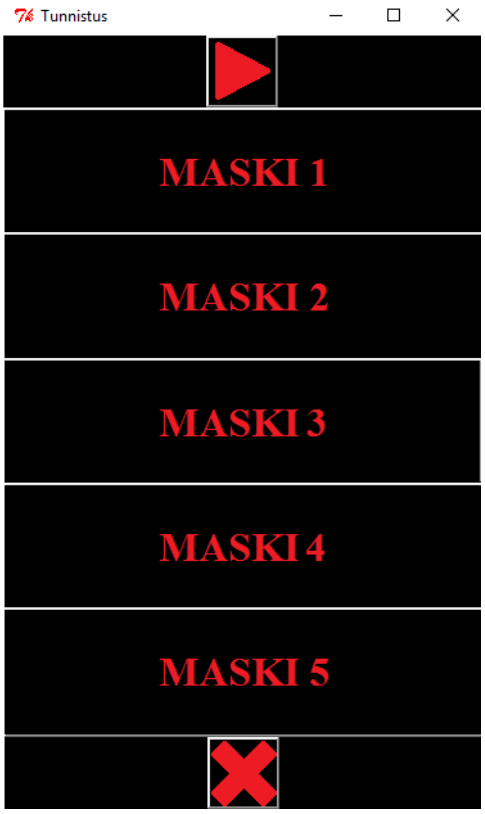
Tutkimusta varten kehitettiin kasvojentunnistussovellus Python -ohjelmointikielellä. Kasvojentunnistussovelluksen tarkoituksena oli luoda reaaliaikainen lisätyn todellisuuden sovellus suorituskyvyn ja sen haasteiden arviointiin. Sovellus tunnistaa kasvot videokuvasta ja piirtää kasvojen päälle erilaisia käyttäjän valitsemia maskeja. Videokuva kaapataan laitteistoon liitetyn kameran avulla, ja näytetään tehostettu kuva alustaan liitetyllä kosketusnäytöllä.

Sovelluksen käyttöliittymässä on kosketuspainikkeet tunnistuksen aloitusta, lopetusta sekä kuvakaappausta varten (Kuva 8). Tehostettu kuva suoratoistetaan ohjelman ikkunaan. Sovelluksen graafinen käyttöliittymä (engl. graphical user interface, GUI) on toteutettu Pythonin standardikirjastoon kuuluvalla Tkinter-

moduulilla. Tunnistusprosessi käyttää avoimen lähdekoodin kirjastokokoelman OpenCV moduulia objdetect (Object Detection) kasvojen tunnistamiseen.

Kasvojen tunnistaminen hyödyntää OpenCV:n tarjoamaa valmiiksi koulutettua LBP (engl. Local Binary Patterns) kaskadi-aineistotiedostoa. Tunnistamiseen olisi myös mahdollista hyödyntää haar-kaskadeja paremman tarkkuuden saavuttamiseksi, mutta tällöin sovellus olisi myös huomattavasti hitaampi.

Tunnistuksen jälkeen tunnistettujen kasvojen päälle piirretään tietokoneen muistissa kuvina olevia maskeja, jotka skaalataan kasvojen koon mukaiseksi. Käyttäjä voi ennen tunnistusta valita piirrettävän maskin. Prosessin nopeuttamiseksi sovellus on jaettu kahteen prosessiin, jossa ensimmäisessä pyörii videokuvan kaappaaminen, tunnistus ja tehostaminen, ja toisessa itse käyttöliittymä ja uusimman kuvan esittäminen.



Kuva 8. Tunnistussovelluksen päävalikko.

LBP:ssä kuva jaetaan pienempiin alueisiin. Alueen sisällä jokaiselle pikselille lasketaan uusi arvo pikselin kahdeksaa naapuripikseliä hyväksi käyttäen. Naapuripikselit kynnystetään laskettavan pikselin arvolla seuraavalla periaatteella: arvoa pienemmät pikselit ovat nolla ja suuremmat yksi. Näistä muodostuu kahdeksanbittinen luku, joka on tarkasteltavan pikselin arvo.

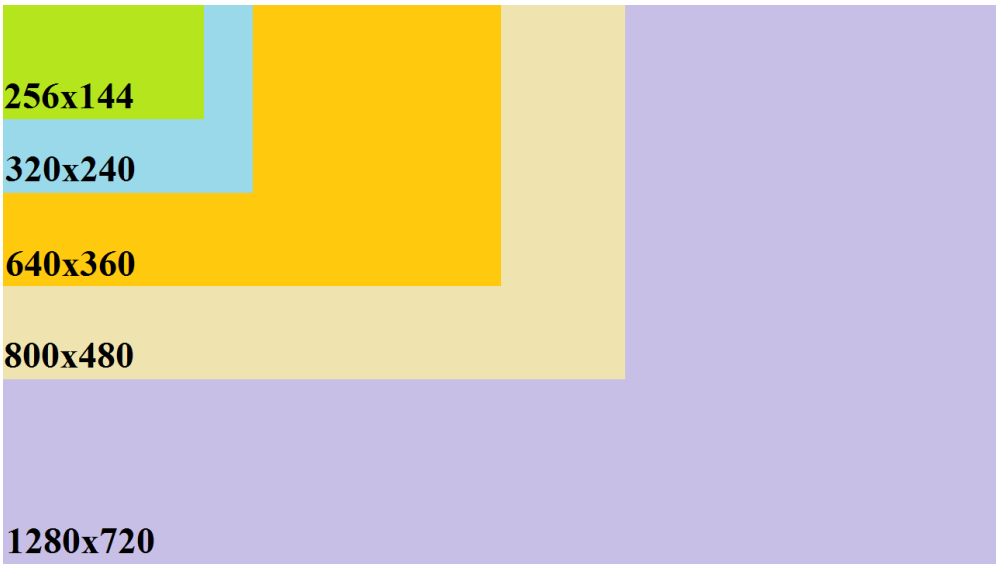
Kun alueen kaikille pikseleille on laskettu uusi arvo, alueesta tehdään histogrammi. Koko kuvan LBP-histogrammi muodostetaan lisäämällä nämä pienempien alueiden histogrammit yhdeksi jonoksi. LBP-histogrammi esittää kuvassa esiintyviä piirteitä [41].

Etsittävien piirteiden määrää on pienennetty ja täsmennetty juuri kasvojen tunnistamiseen kouluttamalla aineisto käyttämällä positiivisia kuvia, joissa on kasvot, ja negatiivisia kuvia, joissa ei ole kasvoja. Tunnistusta on nopeutettu vielä tekemällä aineistosta kaskadi luokittelija (eng. cascade classifier) eli jakamalla tunnistettavat piirteet eri tasoihin. Näin jos ensimmäisellä tasolla löytyvää piirrettä ei löydy kuvasta, sen prosessoiminen voidaan heti hylätä. [42]

5.3. Aineisto

Suorituskykytestejä varten valittiin neljä noin minuutin pituista videota, joissa kolmessa näkyy yhdet ihmiskasvot videon alusta loppuun, ja yhdessä videossa ei näy ihmiskasvoja ollenkaan. Ihmiskasvot sisältävissä videoissa on nähtävissä ihminen pitämässä puhetta tai haastattelun kohteena. Näissä kasvot tekevät normaalia liikehdintää välillä kohdentuen kameraan hieman eri kulmassa. Videomateriaalissa itse kamera ei liiku ja kasvojen taustalla olevalla alueella ei tapahdu muutoksia tai liikehdintää. Kasvottomassa videossa näkyy välillä vaihtelevaa videokuvaa metsämaisemista, joissa kamera on ison osan liikkeessä, jolloin kuva kokee enemmän muutoksia verrattuna kasvollisiin videoihin. Testien tuloksista voi täten havainnoida kasvojen määrän ja erilaisten taustojen vaikutuksen tunnistusmenetelmän suorituskykyyn.

Jokaisesta videosta on viisi erikokoista versiota, joista kolme ovat alhaisia piirtotarkkuuksia (*engl. Low-definition, LD*), yksi standardipiirtoa (*Standard-definition (SD*) ja yksi teräväpiirtoa (*High-definition, HD*). Testejä varten valitut piirtotarkkuudet painottuvat alhaisiin tarkkuuksiin siitä syystä, että oletuksen mukaan pienemmät videokoot tuottaisivat mukavamman AR-sovelluksen käyttökokemuksen. Nimittäin laitteiston ja näytön pieni koko ja ohjelman nopea prosessointi mahdollistavat sovelluksen käytön käsikäyttöisenä, tai päässä pidettävänä alustana, mikä on AR-sovelluksille tärkeää. Kuvassa 9 havainnollistetaan resoluutioiden kokoerot. Resoluutio siis muuttaa kuvan tarkkuutta siinä mielessä, että saman kohteen piirtämiseen käytetään enemmän pikseleitä, jolloin kohde piirretään yksityiskohtaisemmin.



Kuva 9. Testivideoiden resoluutioerot.

Testatessa sovelluksen tunnistuksen maksimitunnistusetäisyyden rajoittamisen vaikutusta suorituskykyyn oli käytössä Raspberry Pi:n oma kamera. Kameralla olisi mahdollista kuvata HD-laatuista materiaalia, mutta aiemmissa testeissä sen selvisi olevan epäkäytännöllistä käyttää lopulliseen sovellukseen, joten testit suoritettiin LD piirtotarkkuudella. Testin videot kuvattiin Raspberry Pi:n kameralla käyttäen resoluutiota 360x240 (24 fps). Testiä varten kuvattiin viisi videota, joissa testaaja istuu kasvot kohti kameraa minuutin ajan. Videot eroavat toisistaan testaajan etäisyyden perusteella, testaajan ollessa ensimmäisessä videossa metrin etäisyydellä kamerasta aina viiteen metriin asti. Testivideoita vastaavat kuvat on esitetty Kuvassa 10, jossa kasvoille on piirretty mustat aurinkolasit eri etäisyyksillä. Kuvan 10 tarkkuus ei kuitenkaan vastaa testivideoissa käytettyjä tarkkuuksia.

Kuva 10. Kasvojen etäisyys kamerasta.

5.4. Testausmenetelmät

Testien tavoitteena on havainnollistaa järjestelmän toteutuksessa tehtyjen ratkaisujen suhteellista vaikutusta lopullisen sovelluksen suorituskykyyn. Erityisesti testit antavat

suuntaa Raspberry Pi:n suorituskyvyn tasosta vastaavissa reaaliaikaisissa kuvankäsittelysovelluksissa. Tutkimuksessa esitellyt muuttuvat elementit ovat käsitellyn tiedon määrän muuttaminen eli kuvakoon muuttaminen ja tunnistusalgoritmin tunnistusajan rajoittaminen, eli kasvoja pienempien objektien hylkääminen tunnistuksessa. Ympäristön ja taustan vaikutusta ei otettu erityisemmin huomioon, mutta se pyrittiin pitämään normaalina. Tällaisia elementtejä ovat esimerkiksi taustan väri, värikylläisyys ja valaistus.

5.4.1. Kuvakoon vaikutus suorituskykyyn ja tunnistukseen

Raspberry Pi:n kuvasuhde on 15:9, mutta videomateriaalin kuvasuhteeksi valittiin kuitenkin 16:9, sillä molemmat kuvasuhteet ovat hyvin lähellä toisiaan ja 16:9 on yleisemmin käytetty kuvasuhde kuin 15:9. Kuvasuhteen 16:9 resoluutioina käytettiin 256x144, 320x240, 640x360 ja 1280x720 pikseliä, lisäksi käytettiin kuvasuhteen 15:9 800x480 pikselin resoluutiota, sillä se on Raspberry Pi:n näytön oletusresoluutio. Jokaiselle kuvalle jokaista resoluutiota kohden laskettiin kuvantunnistuksen onnistumisen todennäköisyys, korrelaatio ja keskimääräinen kuvaruutujen käsittelynopeus.

Testin tarkoituksena on selvittää, kuinka paljon sovelluksessa käsiteltävien kuvien koon pienentäminen vaikuttaa sovelluksen toimintaan. Erityisesti testissä mitataan sovelluksen ruudunpäivityksen muutoksia ja sovelluksen tunnistustarkkuutta pienilläkin resoluutioilla. Testin tulosten perusteella voi päätellä, että onko mahdollista säilyttää hyvä kuvanlaatu, tarkka tunnistusprosentti ja korkea ruudunpäivitysnopeus.

Testausolosuhteiden vakioimiseksi käytettiin neljää noin minuutin pituista videota, joista kolmesta esiintyvät yhdet kasvot ja suhteellisen liikkumaton tausta. Neljännessä videossa ei esiinny ollenkaan kasvoja, vaan kirjava metsämaisema. Tunnistusalgoritmia varten valittu tunnistettavien objektien minimikoot ovat esitettynä Taulukossa 2.

Taulukko 2. Tunnistettavien objektien minimikoot.

Resoluutiot	1280x720	800x480	640x360	320x240	256x144
Minimikoko	100,100	64,64	50,50	29,29	20,20

Objektien minimikoot kuvaavat objektin leveys*korkeus-pikselialuetta, jota pienemmät objektit poistetaan käsittelystä. Minimikoot ovat suhteellisesti yhtä suuret verrattuina toisiinsa. Suhteellinen minimikoko valittiin yhteisesti aineistoon sopivaksi, kun kaikissa testivideoissa tunnistettiin kasvot ongelmitta.

Tulosten mittaaminen toteutettiin kirjoittamalla jokaisen ruudun prosessointi- eli päivitysnopeus ja tunnistettujen kasvojen lukumäärä lokiin ylös. Ruudunpäivitysnopeuksista laskettiin sen jälkeen keskiarvo, josta saatiin kyseisen videon ruudunpäivitysnopeus.

5.4.2. Etäisyyden rajoittamisen vaikutus suorituskykyyn ja tunnistukseen

Etäisyystestien tarkoituksena on selvittää; missä määrin sovelluksen suorituskyky kasvaa sovelluksen tunnistusetäisyyttä rajoitettaessa. Tutkittavien objektien minimikokoarvoja rajatessa tunnistusprosessin käsittelee vähemmän kohteita, joten kuvataajuuden tulisi kasvaa jossain määrin.

Testitilanne suoritetaan pitämällä kamera liikkumattomana ja osoittaen samaan suuntaan. Näin kuvien tausta säilyy samana ja ainoa muuttuja on tunnistettavien kasvojen etäisyys. Tunnistustulokseen merkittävässä roolissa on myös kasvojen valaistus, joten testivideoita kuvatessa pidetään kasvot riittävän valaistuksen kohteena.

Videokuvan laatuna käytetään 320x240 pikseliä 24 ruudunpäivitystä sekunnissa, koska aiemmassa testissä sen huomattiin tuottavan sopivan tasapainon tuotettavan tehostetun kuvan ruudunpäivityksessä ja tunnistustarkkuudessa.

Kaikilla etäisyyksillä pidetään sama suhteellinen tunnistettavan kasvojen koko. Pienillä etäisyyksillä ruudunpäivityksen tulisi kasvaa sillä perusteella, että kuinka suurikokoisia kasvoja/objekteja algoritmi kuvasta etsii. Algoritmi näin hylkää taustasta löydettyjä mahdollisia piirteitä helpommin, lyhentäen kuvan prosessointiin kuluvaa aikaa. Kasvojen etäisyys kamerasta mitataan tarkasti mittanauhalla ja kasvojen satunnainen heiluminen pyritään minimoida etäisyyden vakioimiseksi.

Lopulliset mittaustulokset saadaan ajamalla kuvatut testivideot kehitetyn tunnistussovelluksen lävitse käyttäen Raspberry Pi -alustaa. Video käsitellään yksi kuva kerrallaan, tallentaen samalla jokaiseen kierrokseen kuluva todellinen aika. Samalla tallennetaan tieto kasvojen määrästä, jotka algoritmi mahdollisesti tunnistaa videon kuvasta.

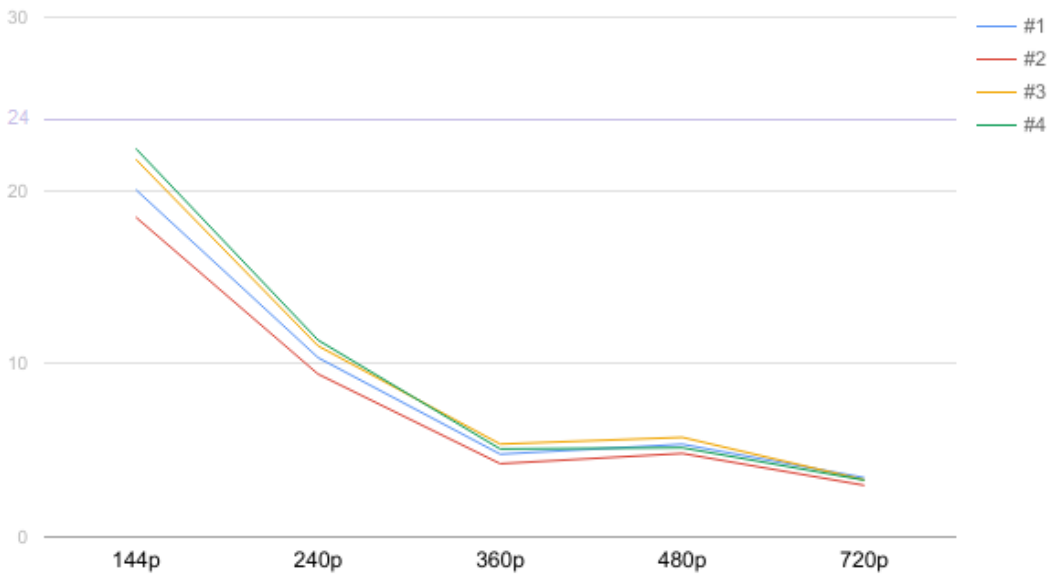
6. TULOKSET

Testit koostuivat kahdesta eri osiosta; kuvakoon sekä etäisyyden testeistä. Testien aikana tarkasteltiin videon kuvantunnistuksen käyttäytymistä, siitä kuinka hyvin ja milloin tunnistus onnistui tai epäonnistui. Tällä pyrittiin havaitsemaan videomateriaalin ominaisuuksista vaikuttavia tekijöitä kasvojentunnistuksen toteutumiseen, esimerkiksi kasvojen kallistuminen suhteessa kameraan.

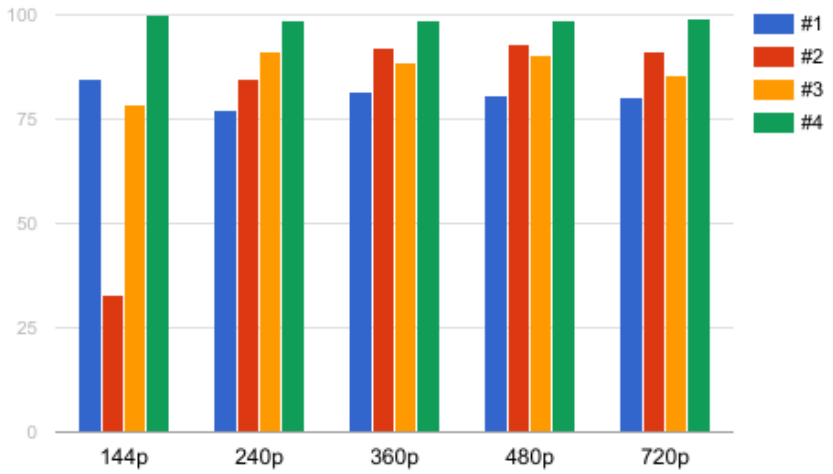
Testaustuloksina saatiin jokaista yksittäistä videoiden kehystä kohden, kehyksen prosessointiin kulunut aika sekä löydettyjen kasvojen lukumäärä. Tämän aineiston perusteella laskettiin kaikille videoille keskimääräinen nopeus, kokonaiskesto, tunnistusprosentti, varianssi, korrelaatio sekä keskipoikkeama.

6.1. Kuvakoon testi

Kuvakoon testissä muutettava elementti oli kuvan koko eli resoluutio. Videot 1-3 esittävät yhdet kasvot ja Video 4 ei sisältänyt kasvoja ollenkaan. Testissä mitattiin kuvataajuutta eli ruudunpäivitysnopeutta, sekä tunnistustarkkuutta. Testin tulokset on koottu kaavioon 11 ja 12 sekä taulukkoon 3. Kaaviossa 11 on korostettuna tavoitteena ollut, myös elokuvateollisuudessa käytetty ruudunpäivitysnopeus 24 FPS.



Kaavio 11. Resoluution vaikutus kuvataajuuteen.



Kaavio 12. Resoluution vaikutus tunnistustarkkuuteen.

Testin tuloksista voidaan sanoa, että videokuvan resoluution kasvaessa ruudunpäivitystaajuus laskee eksponentiaalisesti. Käyrässä on kuitenkin poikkeamia. Ruudunpäivitystaajuustaajuus pienenee resoluution pienentyessä 480p:stä 360p:hen. Vaikka kasvojen sisältyvyys videossa vaikutti hiukan ruudunpäivitysnopeuteen, kaikki videot käyttäytyvät samalla tavalla resoluution pienentymiseen nähden.

Kaaviosta 12 nähdään resoluution vaikutus algoritmin kasvojen tunnistustarkkuuteen. Video neljä ei sisältänyt kasvoja, joten sen tunnistustarkkuus on lähes 100 % kaikilla resoluutioilla. Kolmen muun videon välillä on paljon eroa tunnistustarkkuudessa. Erityisesti videon kaksi tunnistustarkkuuden romahtaminen pienimmällä resoluutiolla viittaa videon sisällöllä olevan enemmän vaikutusta tarkkuuteen. Tunnistusprosentteja ei ole korjattu väärien positiivisten osalta, koska siihen vaikuttaa algoritmin koulutusaineisto enemmän kuin resoluutio. Tunnistustarkkuus näyttää kuitenkin pysyvän samalla tasolla resoluutiosta riippumatta.

Tuloksista on havaittavissa prosessointiajan kasvua resoluution kasvaessa suuremmaksi, tällöin myös algoritmin läpikäymä pisteiden lukumäärän kasvaa. Kuvakoko ei kuitenkaan voi yksistään selittää nopeutta, sillä 800x480 pikselin resoluutiolla on suurempi pikselien lukumäärä kuin 640x320 pikselin resoluutiolla, ja siitä huolimatta se suoriutuu kasvojen tunnistuksesta nopeammin. Raspberry Pi 2:n natiivikuvasuhde on 800x480 pikseliä, tämän kyseisen resoluution tai saman kuvasuhteen videomateriaalin toistaminen voi johtaa nopeampaan kuvankäsittelyyn kevyemmän kuvankäsittelyprosessin seurauksena.

Eri videomateriaalin välillä samoilla resoluutioilla oli pieniä poikkeamia toisiinsa nähden. Tämän voi olettaa johtuvan videomateriaalin ominaisuuksista, kuten:

enemmän yksityiskohtia sisältävä tausta, sekä itse kuvattu henkilö ja tämän liikehdintä.

Pienimmällä, 256x144 pikseliä, resoluutiolla oli havaittavissa suurta varianssia kehysten läpikäynnissä. Pienin suhteellinen varianssi oli videolla 2; 38,89 %, ja suurin kasvottomalla ja suurimman kuvakompleksisuuden omaavalla videolla 4; 227,25 %. Vastaavasti video 2 oli hitain kasvojentunnistuksessa ja video 4 taas nopein.

Kasvollisilla videoilla kasvojen liikehdintä, sekä osittainen kasvojen peittäminen kädellä vaikutti tuottavan ongelmia kasvojen tunnistamiselle. Videolla 2 resoluutiolla 256x144 pikseliä, kasvojentunnistus prosentti jäi 38,89 %:iin, kun muilla tämän videon resoluutioilla, sekä eri videoilla tätä ei tapahtunut. Tämän mahdolliseksi aiheuttajaksi arvioitiin joko mikrofonia, joka peitti pienen osan kasvoista videon keston ajan, videolla esiintyneen henkilön kasvonpiirteitä tai molempien yhteisvaikutusta.

Taulukko 3. Resoluutiotestien kootut tulokset.

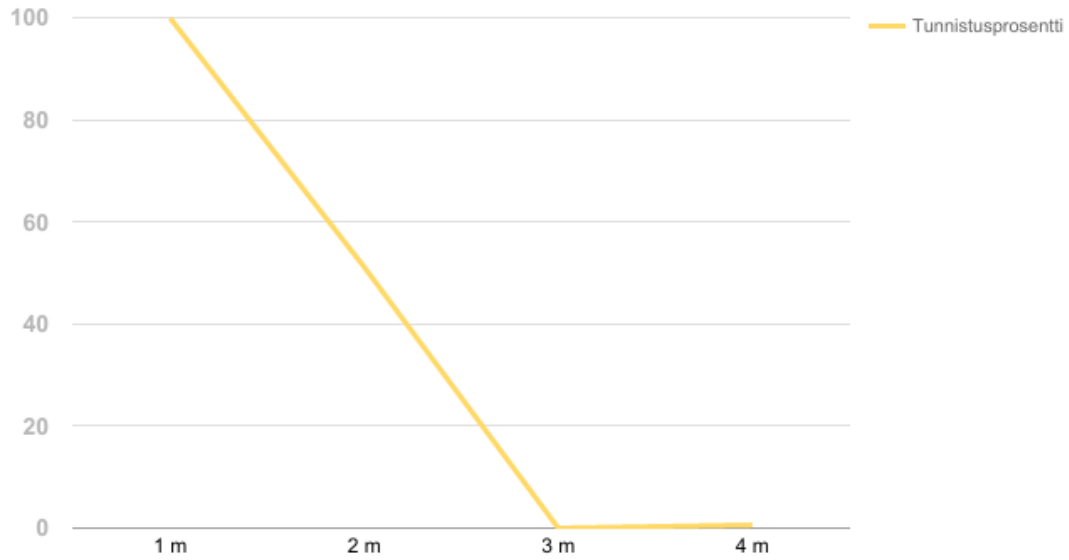
Kuvataajuus (FPS)						
Resoluutiot	256x144	320x240	640x360	800x480	1280x720	Keskiarvo
Video#1	20,037	10,351	4,775	5,347	3,418	8,786
Video#2	18,455	9,404	4,227	4,825	2,983	7,979
Video#3	21,785	11,028	5,357	5,754	3,291	9,443
Video#4	22,406	11,361	5,082	5,146	3,281	9,455
Korrelaatio %						
Resoluutiot	256x144	320x240	640x360	800x480	1280x720	Keskiarvo
Video#1	−4,25	−9,63	−4,39	−4,58	0,69	−4,43
Video#2	−1,98	−2,10	−1,59	−0,03	−1,37	−1,41
Video#3	−7,18	−10,37	−2,08	−6,62	−2,71	−5,79
Video#4	−1,86	−7,30	−5,79	−4,99	−3,16	−4,62
Varianssi %						
Resoluutiot	256x144	320x240	640x360	800x480	1280x720	Keskiarvo
Video#1	53,23	2,63	2,61	6,01	1,00	13,09
Video#2	38,89	0,72	1,42	4,60	2,24	9,58
Video#3	51,04	2,54	5,00	1,58	0,87	12,21
Video#4	227,25	89,46	46,33	16,74	0,68	76,09

Keskipoikkeama %						
Resoluutiot	256x144	320x240	640x360	800x480	1280x720	Keskiarvo
Video#1	16,30	5,04	7,39	10,60	5,41	8,95
Video#2	14,52	2,77	5,80	9,77	8,67	8,31
Video#3	15,31	4,80	9,66	5,25	5,15	8,03
Video#4	31,85	28,06	30,19	18,03	4,56	22,54
Kasvojentunnistus %						
Resoluutiot	256x144	320x240	640x360	800x480	1280x720	Keskiarvo
Video#1	84,48	77,25	81,36	80,65	80,44	80,84
Video#2	32,66	84,70	92,20	93,12	91,57	78,85
Video#3	78,39	91,10	88,59	90,45	85,63	86,83
Video#4	0,05	1,40	1,45	1,31	0,73	0,99

6.2. Etäisyystesti

Toisessa testissä muutettava elementti on tunnistettavan kohteen etäisyys. Testissä tunnistettava kohde oli paikallaan neljällä eri etäisyydellä kamerasta, joista mitattiin tunnistusprosentti sekä tunnistettavan kohteen etäisyyden vaikutusta kuvataajuuteen minuutin ajanjakson yli. Resoluutio vakioitiin testissä 360x240 pikseliin. Etäisyyden kasvaessa tunnistettavan kohteen koko pienenee kuva-alueeseen nähden ja tarkempaa eli olettaen hitaampaa tarkistusta tehdään. Testin tulokset on koottu kaavioihin 13 ja 14.

Tunnistustarkkuus pienenee lähes täydellisen lineaarisesti etäisyyden kasvaessa. Metrin etäisyydellä tunnistustarkkuus on 99,9 %, kahdessa metrissä se on 51,1 % ja kolmessa metrissä se laskee 0 %:iin. Tunnistustarkkuuteen vaikuttaa kuitenkin paljon valaistus ja tausta. Jos taustassa on paljon pieniä yksityiskohtia ja kuvioita, suurilla etäisyyksillä kasvojen ollessa saman kokoisia näiden kanssa, tunnistustarkkuus kärsii voimakkaasti. Kuvataajuus pysyy lähes samana tunnistettavan objektin etäisyydestä riippumatta. Yhdessä ja kahdessa metrissä kuvataajuus on noin 14 FPS. Kolmessa ja neljässä metrissä vastaavasti 12-13 FPS välissä. Kuvataajuus on kuitenkin kaikilla etäisyyksillä korkeampi kuin kuvankoon testissä käytetyissä videoissa samalla resoluutiolla.



Kaavio 13. Tunnistettavan kohteen etäisyyden vaikutus tunnistustarkkuuteen resoluutiolla 360x240.

Kaavio 14. Tunnistettavan kohteen etäisyyden vaikutus kuvataajuuteen resoluutiolla 360x240.

7. ANALYYSI

Testit havainnoivat hyvin, kuinka pelkästään kahden eri elementin muuttaminen vaikuttaa olennaisesti sovelluksen suorituskykyyn. Analyysissä käsitellään näiden elementtien, kuvankoon ja tunnistusetäisyyden, vaikutusta. Näiden lisäksi käsitellään olosuhteiden vaikutusta testituloksiin.

7.1. Kuvakoon muutos

Kuvakoolla on suuri vaikutus reaaliaikaisten sovelluksen käytettävyyteen sovelluksen ruudunpäivitysnopeudessa ja kuvantarkkuudessa. Sulautetuissa järjestelmissä valitettavasti harvoin voi valita molemmat vaatimukset, jolloin vaatimuksista riippuen toista korostetaan enemmän ja valitaan sopiva ratkaisu vaatimusten väliltä.

Kuvatun videon koolla on selvä vaikutus sovelluksen suorituskykyyn. Syynä suorituskyvyn alenemiseen suuremmilla kuvakoilla on käsiteltävän tiedon määrä. Pienempää pikseli määrää on nopeampi prosessoida, siirtää ja muuttaa, joten ruudunpäivitys on alustasta riippumatta suurempi pienillä kuvakoilla.

Tunnistusalgoritmin tunnistustarkkuus kasvaa kuvakoon myötä. Suurikokoista kuvakokoa käytettäessä siis sovellus tuottaa parhaan lopputuloksen, mutta hitaammin. Tästä syystä on mielekästä valita mieluiten sellainen resoluutio, joka palauttaa sovelluksesta riippuvan ruudunpäivityksen ja tarkkuuden suhteen. Toiset sovellukset eivät esimerkiksi vaadi reaaliaikaisuutta, jolloin kuvanlaadun voi vapaasti pitää tarkkana.

Testeissä saatu FPS on yllättävän alhainen. Se aiheuttaa suuria rajoituksia sovelluskohteille. Vaikka tunnistusalgoritmi toimii monisäikeisenä, ei Rasperry Pi pysty toimimaan siedettävällä nopeudella toteuttaakseen lisätyn todellisuuden sovellusten vaatimaa reaaliaikaisuutta. Testaussovelluksessa pullonkaula on sen käyttöliittymä. Kehityksen aiemmassa vaiheessa ilman käyttöliittymää sovelluksen suorituskyvyn huomattiin olevan jopa 60 % parempi. Jatkokehityksen kannalta olisi siis syytä etsiä parempi ratkaisu toteuttaa käyttöliittymän toiminnallisuus.

7.2. Tunnistusetäisyyden muutos

Tunnistusalgoritmin käsittelemien objektien koolla on suora vaikutus sovelluksen ruudunpäivitykseen ja tunnistustarkkuuteen, etenkin pienillä kuvankoilla. Riittävän pienillä kuvankoilla tunnistustarkkuus heikkenee. Koska kehitetty sovellus vaatii reaaliaikaisuutta sulautetussa järjestelmässä ja ruudunpäivitysnopeutta on korostettava yli muiden vaatimusten, tunnistustarkkuuden ei pitäisi antaa vääriä positiivisia arvoja.

Tunnistettavan objektin etäisyydellä ei huomattu olevan suurta vaikutusta kuvataajuuteen sen jälkeen, kun sovellus jaettiin eri prosesseihin. Testissä käytetty aineisto tuotti paremman FPS nopeuden kuin saman resoluution aineisto kuvankoon testissä. Aineistolla on siis suuri vaikutus algoritmin ja sovelluksen toimintaan.

Pienellä kuvakoolla tunnistusprosentin havaittiin laskevan huomattavasti tunnistettavan objektin etäisyyden kasvaessa kamerasta. Tämä ongelma oletettavasti poistuisi kuvan resoluutiota kasvattaessa, mutta tällöin kuvankäsittelyyn kuluva aika laskisi alle toivotun rajan.

Sovelluksen tunnistusetäisyyttä pystyttäisiin kasvattamaan, jos tunnistettava kohde olisi yksinkertaisempi. Tällöin tunnistettavan kohteen piirteet olisivat selkeämmät eikä sekoittumista muihin objekteihin tapahdu. Kasvot ovat kuitenkin suhteellisen kompleksiset verrattuna esimerkiksi koripalloon, jolloin pienelläkin kuvakoolla koripallon tunnistaminen olisi mahdollista yksinkertaisen muodon perusteella.

7.3. Olosuhteet

Sovellusta testatessa tuli selväksi käytetyn tunnistusmenetelmän puutteet, kuten kasvojen asento kameraan nähden, tarpeellisen valaistuksen ylläpitäminen sekä tunnistuskohteen taustan monimutkaisuus.

Sovelluksen suoritusalustalla on lisäksi oma vaikutuksensa suorituskykyyn. Esimerkiksi nopeammalla prosessorilla kuvakokoa voisi kasvattaa samalla säilyttäen korkean ruudunpäivityksen.

Kasvojen ollessa selvästi kallellaan ei tunnistusmenetelmä tunnista kasvoja kuvasta, mikä aiheuttaa osan aineiston alhaisemmista tunnistusprosenteista. Tunnistus ei onnistu heikossa valaistuksessa, vaan on välillä tarpeellista osoittaa kasvoja lisävalolla. Vaihtuva ja monimutkainen tausta aiheuttaa vain pienen laskun suorituskyvyssä, mutta vaikutus kasvaa tunnistettavan kohteen etäisyyden kasvaessa.

Nämä ongelmat ovat suurimmaksi osaksi korjattavissa paremmalla tunnistusalgoritmin opetuksella ja selkeämmillä tunnistettavilla objekteilla. Ympäristön tuottamat ongelmat kuitenkin luovat rajoitteita käyttöympäristöille ja -kohteille. Esimerkiksi ulkona tapahtuva tunnistus voi olla haastavaa. Myös jos tunnistuksessa halutaan saavuttaa maksimaalinen etäisyys, esimerkiksi liikennemerkkien tai jalankulkijoiden tunnistus autossa, on tällä menetelmällä saatu hyvä tunnistusetäisyys liian pieni kyseisiin sovelluksiin.

8. YHTEENVETO

Lisätyn todellisuuden sovelluksen yksi tärkeimpiä osia on objektin tunnistus, joten tutkimuksessa on keskitytty selvittämään tunnistuksen suorituskyky tietyllä laitteistolla, Raspberry Pi:llä. Seurauksena voi arvioida lisätyn todellisuuden sovelluksen toteuttamisen mahdollisuuksia tällä sulautetulla järjestelmällä. Tutkimuksessa keskityttiin objektintunnistuksessa kasvojentunnistukseen. Sen suoriutumista ja optimaalisia asetuksia lähdettiin tutkimaan kuvan koon ja tunnistettavan objektin etäisyyden avulla.

Testien tuloksissa näkyy kuvanlaadun ja käyttötarkoituksen muuttamisen vaikutus suorituskykyyn. Kuvan koon kasvaessa ruudunpäivitysnopeus pieneni eksponentiaalisesti liian pieneksi ollakseen reaaliaikainen. Ruudunpäivitysnopeus oli parhaimmillaan 20 FPS pienimmällä 144p resoluutiolla, mikä ei ole siltikään yltänyt toivottuun 30 FPS nopeuteen. Tunnistustarkkuus pysyi noin 80 %:ssa suuremmilla resoluutioilla, joten lähietäisyyksillä alhaisiakin resoluutioita käytettäessä tunnistustarkkuus säilyy hyvänä.

Etäisyydellä huomattiin olevan suuri vaikutus tunnistustarkkuuteen 240p resoluutiolla. Metrin etäisyydellä saatiin 100 % tunnistustarkkuus, mutta kun etäisyys kasvoi kahteen metriin, pieneni tunnistustarkkuus 50 %:iin. Tunnistustarkkuuden heikkeneminen etäisyyden kasvaessa on seurausta testissä käytetyn kuvan resoluution pienestä koosta. Ympäristön olosuhteilla huomattiin lisäksi olevan yhä suurempi vaikutus tunnistustarkkuuteen etäisyyden kasvaessa.

Yksi keskeisimmistä haasteista lisättyä todellisuutta sovellettaessa sulautetussa järjestelmässä reaaliaikaisesti on hyvän kuvanlaadun toteuttaminen. Reaaliaikaisuus on saavutettavissa yksinkertaisemmissa tehtävissä, mutta ei kasvojentunnistuksessa. Tarkkaa tunnistusta haettaessa suuri kuvakoko asettaa liian monia hidasteita datan käsittelyssä.

Löydösten perusteella reaaliaikainen objektintunnistus Raspberry Pi:llä ei sovellu erityisen monimutkaisiin tehtäviin. Objektin tulisi täyttää erittäin suuri osa kuvasta ja olla mahdollisimman yksinkertainen hyväksyttävän suorituskyvyn saavuttamiseksi. Testattu järjestelmä toteuttaa joko hyvän kuvanlaadun tai reaaliaikaisuuden vaatimukset, mutta ei molempia. Tämä tarkoittaa, että sovelluksen käyttötarkoitus on rajattava pienelle sovellusalueelle ollakseen käytettävä. Valinta kahden ominaisuuden välillä onkin tyypillistä sulautetuille järjestelmille. Mahdollisesti toimiva tarkoitus liittyisi yksinkertaisten hahmojen tunnistukseen siinä tapauksessa, kun reaaliaikaisuus on ydinvaatimuksena.

9. LÄHTEET

[1] Kruger CP & Hancke GP. (2014) Benchmarking Internet of Things Devices. Proceedings - 2014 12th IEEE International Conference on Industrial Informatics, INDIN 2014. : 611-616.

[2] K. Janard & W. Marurngsith. (2015) Accelerating Real-Time Face Detection on a Raspberry Pi Telepresence Robot. Fifth International Conference on the Innovative Computing Technology (INTECH 2015). : 136-141.

[3] Vujovic V & Maksimovic M. (2015) Raspberry Pi as a Sensor Web node for home automation. Comput Electr Eng 44: 153-171.

[4] Azuma, Ronald T. (1997) A Survey of Augmented Reality. Presence: Teleoperators and virtual environments 6.4: 355-385.

[5] Milgram, P., & Kishino, F. (1994). A taxonomy of mixed reality visual displays. IEICE TRANSACTIONS on Information and Systems, 77(12), 1321-1329.

[6] Arth, Clemens, et al. (2015) The History of Mobile Augmented Reality Developments in Mobile AR over the last almost 50 years. p.23

[7] Craig AB. (2013) Understanding Augmented Reality: Concepts and Applications. : Newnes.

[8] Azuma R, Baillot Y, Behringer R, Feiner S, Julier S & MacIntyre B. (2001) Recent advances in augmented reality. IEEE Comput Graphics Appl 21(6): 34-47.

[9] Google Glass [Online] URL: https://www.google.com/glass/help/ (20.11.2016).

[10] Seichter H, Grasset R, Looser J & Billinghurst M. (2009) Multitouch Interaction for Tangible User Interfaces. ISMAR. : 213-214.

[11] Chi H, Kang S & Wang X. (2013) Research trends and opportunities of augmented reality applications in architecture, engineering, and construction. Autom Constr 33: 116-122.

[12] Mann S, Fung J, Aimone C, Sehgal A & Chen D. (2005) Designing EyeTap digital eyeglasses for continuous lifelong capture and sharing of personal experiences. Alt.Chi, Proc.CHI 2005 .

[13] Rosenberg LB. (1993) Virtual Fixtures: Perceptual Tools for Telerobotic Manipulation. Virtual Reality Annual International Symposium, 1993., 1993 IEEE. : IEEE: 76-82.

[14] Sood R. (2012) Pro Android Augmented Reality. : Apress.

[15] Business Wire (2016) Pokémon GO Exceeds 500 Million Downloads Worldwide. [Online] URL: http://www.businesswire.com/news/home/20160907006800/en/Pok%C3%A9mon-Exceeds-500-Million-Downloads-Worldwide (20.11.2016).

[16] Barsom EZ, Graafland M & Schijven MP. (2016) Systematic review on the effectiveness of augmented reality applications in medical training. Surg Endosc : 1-10.

[17] Nee A, Ong SK, Chryssolouris G & Mourtzis D. (2012) Augmented reality applications in design and manufacturing. CIRP Annals-Manufacturing Technology 61(2): 657-679.

[18] Moshell M. (1993) Three views of virtual reality: virtual environments in the US military. Computer 26(2): 81-82.

[19] Soro A, Rakotonirainy A, Schroeter R & Wollstdter S. (2014) Using Augmented Video to Test in-Car User Experiences of Context Analog HUDs. Adjunct Proceedings of the 6th International Conference on Automotive User Interfaces and Interactive Vehicular Applications. : ACM: 1-6.

[20] Wang X. (2009) Augmented reality in architecture and design: potentials and challenges for application. International Journal of Architectural Computing 7(2): 309-326.

[21] Yovcheva Z, Buhalis D & Gatzidis C. (2012) Smartphone augmented reality applications for tourism. e-Review of Tourism Research (eRTR) 10(2): 63-66.

[22] Mistry P & Maes P. (2009) SixthSense: A Wearable Gestural Interface. ACM SIGGRAPH ASIA 2009 Sketches. : ACM: 11.

[23] Grasset R, Dnser A & Billinghurst M. (2008) Edutainment with a Mixed Reality Book: A Visually Augmented Illustrative Childrens' Book. Proceedings of the 2008 international conference on advances in computer entertainment technology. : ACM: 292-295.

[24] Daponte P, De Vito L, Picariello F & Riccio M. (2014) State of the art and future developments of the Augmented Reality for measurement applications. Measurement 57: 53-70.

[25] Bali A & Singh SN. (2015) A Review on the Strategies and Techniques of Image Segmentation. 2015 Fifth International Conference on Advanced Computing & Communication Technologies. : IEEE: 113-120.

[26] Gonzalez RC & Woods RE. (2002) Digital image processing.

[27] De La Torre A, Peinado AM, Segura JC, Prez-Crdoba JL, Bentez MC & Rubio AJ. (2005) Histogram equalization of speech representation for robust speech recognition. IEEE Transactions on Speech and Audio Processing 13(3): 355-366.

[28] Zaitoun NM & Aqel MJ. (2015) Survey on Image Segmentation Techniques. Procedia Computer Science 65: 797-806.

[29] Bali A & Singh SN. (2015) A Review on the Strategies and Techniques of Image Segmentation. 2015 Fifth International Conference on Advanced Computing & Communication Technologies. : IEEE: 113-120.

[30] Vacchetti L, Lepetit V & Fua P. (2004) Combining Edge and Texture Information for Real-Time Accurate 3d Camera Tracking. Mixed and Augmented Reality, 2004. ISMAR 2004. Third IEEE and ACM International Symposium on. : IEEE: 48-56.

[31] Rajappa S & Raj G. (2016) Application and Scope Analysis of Augmented Reality in Marketing using Image Processing Technique. Cloud System and Big Data Engineering (Confluence), 2016 6th International Conference. : IEEE: 435-440.

[32] Zitova B & Flusser J. (2003) Image registration methods: a survey. Image Vision Comput 21(11): 977-1000.

[33] Vahid F & Givargis T. (2002) Embedded System Design: A Unified Hardware/Software Introduction. : John Wiley & Sons New York, NY.

[34] David Meyer. (2016) This $35 Computer Just Passed a Major Sales Milestone. [Online] URL: http://fortune.com/2016/09/08/raspberry-pi-10-million/ (20.1.2017).

[35] Raspberry Pi Products. (2016) [Online] URL: https://www.raspberrypi.org/products/ (20.1.2017).

[36] Raspberry Pi Foundation. (2013) Trustees' report and financial statements for the year ended 31 december 2012. [Online] URL: https://www.raspberrypi.org/files/about/RaspberryPiFoundationReport2012.pdf (1.2.2017).

[37] Raspberry Pi 3, Pi 2, B+, A+ Comparison Chart (2017) [Online] URL: https://www.element14.com/community/docs/DOC-68090?ICID=rpimain-crosspromo-bullet (3.2.2017).

[38] Nikisins O, Fuksis R, Kadikis A & Greitans M. (2015) Face recognition system on raspberry pi. Proc.of ICIPCE 15.

[39] Patil A & Shukla M. (2014) Implementation Of Classroom Attendance System Based On Face Recognition In Class. International Journal of Advances in Engineering & Technology 7(3): 974.

[40] M. Woodside, G. Franks & D. C. Petriu. (2007) The Future of Software Performance Engineering. Future of Software Engineering, 2007. FOSE '07. : 171-187.

[41] Ahonen T, Hadid A & Pietikainen M. (2006) Face description with local binary patterns: Application to face recognition. IEEE Trans Pattern Anal Mach Intell 28(12): 2037-2041.

[42] OpenCV Python Tutorials. Face Detection using Haar Cascades (2016) [Online] URL: http://docs.opencv.org/trunk/d7/d8b/tutorial_py_face_detection.html (28.12.2016).