TAMPEREEN TEKNILLINEN YLIOPISTO

AKI AHONEN

KASVOANIMAATION KAAPPAUSJÄRJESTELMÄ INDIE-PELIKEHITTÄJILLE

Diplomityö

Tarkastaja: Prof. Tommi Mikkonen
Tarkastaja ja aihe hyväksytty Tieto- ja sähkötekniikan tiedekuntaneuvoston kokouksessa 5.12.2012

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Tietotekniikan koulutusohjelma
AHONEN, AKI: Kasvoanimaation kaappausjärjestelmä indie-pelikehittäjille
Diplomityö, 58 sivua
Kesäkuu 2013
Pääaine: Ohjelmistotuotanto
Tarkastaja: Prof. Tommi Mikkonen
Avainsanat: animaatio, liikkeenkaappaus, liikkeenseuranta, OpenCV

Tässä diplomityössä tutustutaan liikkeenkaappaukseen perustuvaan kasvoanimaatioon ja esitellään yksinkertainen ja halpa järjestelmä, joka suorittaa liikkeenkaappauksen nykyaikaisen älypuhelimen videokameran ja liikesensorien avulla. Entistä realistisempi 3D-grafiikka vaatii rinnalleen entistä realistisempaa animaatiota todellisuuden tunnun säilyttämiseksi. Pelissä esiintyvien ihmismäisten hahmojen kannalta tämä on tarkoittanut vaiheittaista siirtymistä käsin tuotetusta animaatiosta kohti liikekaapattua animaatiota, jossa animaatio tuotetaan todellisen ihmisen liikkeistä. Viime vuosina liikekaapatun kasvoanimaation määrä on lähtenyt nousuun suuren budjetin tuotannoissa, ja monenlaisia kasvonliikkeiden kaappaukseen erikoistuneita tuotteita on ilmestynyt markkinoille.

Kasvonliikkeiden kaappauksessa yleisin käytetty menetelmä on asettaa näyttelijän kasvoille selkeästi erottuvia merkkipisteitä ja kuvata kasvoja videokameroilla useista eri suunnista. Merkkipisteiden liikettä seurataan optisesti videokuvan perusteella ja järjestelmän kalibroinnilla tuotetaan järjestelmälle tietoa, jonka perusteella eri kameroiden kuvista voidaan laskea yksittäisten merkkipisteiden kolmiulotteiset sijainnit. Kaappaukseen on toki muitakin menetelmiä, joista subjektiivisesti luonnollisimman tuloksen on viime vuosina antanut merkkipisteetön liikkeenkaappaus, jossa kaapattiin koko kasvojen 3D-malli sekä sen tekstuurit suoraan näyttelijän kasvoilta. Kyseinen peli oli L.A. Noire vuodelta 2011.

Tässä diplomityössä tarkastellaan kasvoanimaation kaappausta indie-pelikehittäjän näkökulmasta. Työn kannalta indie-kehittäjiksi lasketaan kaikki, jotka eivät vaadi lopputulokselta täydellisyyttä, ja joilla on käytössään hyvin rajalliset resurssit, niin rahalliset kuin ajallisetkin. Niinpä liikkeenkaappauksen lähdemateriaalin taltioimiseen valjastettiin yksittäinen nykyaikainen älypuhelin, jonka videokameralla tallennettiin kuvaa näyttelijän kasvoista ja jonka liikesensoreilla kaapattiin hänen päänliikkeitään. Puhelin on kiinnitetty kypäratelineeseen, jotta se liikkuu näyttelijän pään mukana. Itse liikkeenseuranta ja sensoridatan jälkikäsittely suoritetaan työpöytätietokoneella, ja puhelimen vastuulla on vain lähdemateriaalin tallentaminen. Kehittäjän aikaresurssien säästämiseksi järjestelmä suunniteltiin mahdollisimman pitkälle automatisoiduksi.

Toteutetun järjestelmän perusteella nykyaikaisen älypuhelimen ja avoimen OpenCV-tietokonenäkökirjaston avulla on todellakin mahdollista kaapata laadukasta kasvoanimaatiota indie-kehittäjän tarpeisiin nähden. Päänliikkeet jouduttiin kuitenkin laitteistorajoitusten takia kaappaamaan ainoastaan kiihtyvyyssensorin avulla, mikä rajoitti lopputuloksen tarkkuutta verrattuna järjestelmään, joka pystyisi hyödyntämään gyroskooppiakin. Kehitettävää on myös vielä niin tehokkuuden, luotettavuuden kuin animaation luonnollisuudenkin saralla, mutta potentiaalia järjestelmällä on runsaasti.

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Information Technology
AHONEN, AKI: Facial Motion Capture System for Indie Game Developers
Master of Science Thesis, 58 pages
June 2013
Major: Software engineering
Examiner: Prof. Tommi Mikkonen
Keywords: animation, motion capture, motion tracking, OpenCV

This master's thesis examines motion captured facial animation and presents a simple and cheap system that performs facial motion capture, using only a modern smart phone's video camera and motion sensors. As 3D graphics have become more and more realistic, they have also started to require more and more realistic animation to preserve the naturalness of the graphics, even in motion. For any human-like in-game characters, this has meant a gradual transition from manually produced animation to motion captured animation, which is captured from the movements of a real human being. In recent years, motion captured facial animation has become popular in big budget productions, and many kinds of facial motion capture systems have appeared on the market.

The most popular method for capturing facial motion is to put distinct markers on an actor's face and shoot the actor's face on video from many different directions. The movements of the markers are then tracked optically from the video footage and combined with the system's calibration data in order to calculate the three dimensional coordinates of each maker. There are, of course, several other methods to capture facial motion, one of which utilized a marker-less motion capture system that captured the actor's entire face into a stream of 3D meshes and their textures. Subjectively, this system produced some of the most natural looking facial animation seen in games to this day. The game that used this system was L.A. Noire, released in 2011.

This master's thesis is made from the viewpoint of an indie game developer. For the purposes of this thesis, an indie developer is anyone who does not require perfection from the results and who has very limited financial and human resources. Thus, the system harnesses the features of a modern smart phone to capture the footage for the motion capture. The phone's camera captures the actor's face on video while her head movements are captured with the phone's motion sensors. The phone is attached to a helmet rig that keeps the device fixed to the actor's moving head. The motion tracking and the processing of the sensor data is later handled on a desktop computer. In order to reduce the amount of work for the indie developer, the system is designed to be as automated as possible.

The system created for this thesis provides clear evidence that the combination of a modern smart phone and the OpenCV computer vision library makes it possible to capture facial animation of good enough quality for the requirements of an indie game developer. However, due to hardware restrictions, head movements were captured using only the phone's accelerometer, which limits the quality of the resulting animation when compared to a system that can also utilize a gyroscope's measurements. Improvements are needed in other areas as well but the system clearly has plenty of potential.

ALKUSANAT

Tämä diplomityö syntyi henkilökohtaisesta kiinnostuksestani ja aiemmista aiheeseen liittyvistä kokeiluistani. Tarinavetoiset pelit ovat olleet minulle tärkeitä, joten indiepelikehittäjänä koin tietysti tarvetta monipuolisemmalle tarinankerronnalle. Minulle se tarkoitti ensisijaisesti mahdollisuutta pystyä syventämään pelin hahmoja monipuolisemman hahmoanimaation avulla. Työn aihe pyöri päässäni reilun parin vuoden ajan, ennen kuin marraskuun lopulla 2012 rohkenin alkaa työstää sitä diplomityöksi. Tänään, noin kuusi kuukautta myöhemmin, se on viimein valmis.

Suuri kiitos itsevarmuuden nostattamisesta ja kaiken kaikkiaan työn laadunvalvonnasta kuuluu työn ohjaajana ja tarkastajana toimineelle professori Tommi Mikkoselle. Hänen kannustava ja työni aiheesta kiinnostunut olemuksensa oli erittäin tärkeää myös motivaation ylläpitämiseksi. Kiitoksensa ansaitsevat myöskin isäni, Veikko Ahonen, joka kasasi työssä käytetyn kypärätelineen ja avusti sen jatkokehityksessä, sekä muu kotiväki ja ystäväpiiri, joiden kiinnostus aihetta kohtaan kannusti tekemään hyvää jälkeä. Kiitos kaikille!

Helsingissä 9.5.2013

Aki Ahonen
akiahonen@gmail.com

SISÄLLYS

1 JOHDANTO

Tässä työssä tutustutaan liikkeenkaappauksella toteutettuun kasvoanimaatioon ja esitellään yksinkertainen, nykyaikaisen älypuhelimen käyttöön perustuva järjestelmä, jonka avulla indie-kehittäjätkin pystyisivät tuottamaan peleihinsä tavallista realistisempaa kasvoanimaatiota vastaamaan teknologian kehityksen myötä kasvaneeseen peligrafiikan realismiin. Laadukkaalla animaatiolla pystytään tehostamaan pelin draamaa ja mahdollistetaan elokuvamaista kuvallista kerrontaa, minkä ansiosta pelaajien on aiempaa helpompi muodostaa tunnesiteitä pelin hahmoihin ja saavuttaa siten parempi pelikokemus. Uskottavuuden nimissä entistä realistisemman näköisten hahmojen on myös käyttäydyttävä realistisesti.

Useissa suuremman budjetin tuotannoissa ollaankin viime vuosina siirrytty käyttämään liikkeenkaappausta ihmishahmojen animointiin. Liikkeenkaappauksessa animaatio kaapataan suoraan todellisen näyttelijän kehonliikkeistä, jolloin pystytään tuottamaan hyvin luonnollista animaatiota nopeasti verrattuna perinteiseen, manuaalisesti tehtävään animaatioon. Liikkeenkaappauksen käyttö on kuitenkin pääasiassa jäänyt kehonliikkeiden kaappaamisen tasolle ja kasvot on jätetty perinteisten menetelmien varaan, kenties lisätyömäärän välttämisen takia. Liikekaapattu kasvoanimaatio on kuitenkin aivan viime vuosina alkanut yleistyä suuren budjetin tuotannoissa.

Tämän diplomityön esittelemän järjestelmän avulla harrastelijat ja ammattimaiset indie-pelikehittäjät pystyvät valjastamaan liikekaapatun kasvoanimaation ja päänliikkeet ehostamaan omaa peliään. Järjestelmän suunnittelussa on otettu huomioon oleellisimmat indie-kehittäjien rajoitukset: Järjestelmän on oltava edullinen eikä se saa vaatia mitään erityislaitteistoa toimiakseen. Erityislaitteiston sijaan järjestelmä käyttää nykyaikaista älypuhelinta kaappaamaan videokuvaa näyttelijän kasvoista sekä seuraamaan näyttelijän päänliikkeitä puhelimen liikesensoreilla. Lisäksi järjestelmän toiminnan on oltava tehokasta, luotettavaa ja mahdollisimman pitkälle automatisoitua tuottavuuden maksimoimiseksi. Viimeisenä, mutta hyvin oleellisena vaatimuksena, lopputuloksena saatavan animaation on oltava riittävän luonnollista. Järjestelmä ei siis pyri kilpailemaan kalliiden tai edes keskihintaisten liikkeenkaappausjärjestelmien kanssa, vaan pyrkii tarjoamaan mahdollisimman edullisen, mutta silti hyvän, vaihtoehdon.

Tässä diplomityössä tutustutaan yleisesti liikkeenseurannan ja -kaappauksen historiaan ja menetelmiin ja selvitetään tarkemmin, miten nimenomaan älypuhelimen valjastaminen tähän käyttötarkoitukseen onnistuu. Liikkeenkaappaus jakautuu tämän työn toteutuksen osalta optiseen, videokuvaan perustuvaan liikkeenseurantaan ja liikesensorien datan perusteella tehtävään päänliikkeiden arvioimiseen. Puhelin vastaa lähdemateriaa-

lin, videokuvan ja sensoridatan, tallentamisesta, ja varsinainen datan käsittely suoritetaan tavallisella, Windows-käyttöjärjestelmää käyttävällä tietokoneella.

Teksti jakaantuu johdannon lisäksi neljään lukuun. Luvussa 2 esitellään työn taustoja aloittaen työn pohjana ja inspiraationa toimineesta edellisestä prototyypistä ja jatkaen liikkeenkaappauksen ja -seurannan teoriaan ja päänliikkeiden arvioimiseen älypuhelimen liikesensorien avulla. Luvussa 3 tutustutaan tarkemmin työssä toteutettuun järjestelmään, sille asetettuihin vaatimuksiin ja toteutuksen oleellisimpiin osiin. Luvussa 4 arvioidaan toteutetun järjestelmän toimintaa ja soveltuvuutta käyttötarkoitukseensa, siis sitä, täyttääkö se sille asetetut vaatimukset. Lopulta luvussa 5 esitetään yhteenveto työn tuloksista.

2 TAUSTAA

Aloitetaan tutustumalla tämän työn ja työssä oleellisten tekniikoiden ja laitteiden taustoihin. Käsiteltävät aihepiirit ovat aiemmin toteutettu kasvoanimaatioprototyyppi (kohta 2.1), liikkeenseuranta ja -kaappaus yleisesti (kohta 2.2) sekä videokuvaan perustuva liikkeenseuranta tarkemmin esiteltynä (kohta 2.3), päänliikkeiden arviointi (kohta 2.4), joka keskittyy elektronisin sensorein tapahtuvaa liikkeenseurantaan, ja lopulta varsinainen 3D-mallin animointi (kohta 2.5).

2.1 Työn perusta

Tämän työn juuret juontuvat vuonna 2010 viikossa kehitettyyn lyhyeen peliin nimeltä By the Numbers[1]. Peli sijoittuu poliisiaseman kuulusteluhuoneeseen, jossa pelaajan ohjastama, sarjamurhia selvittävä rikostutkija haastattelee nuoren tytön kaappauksen silminnäkijää. Muista ansioistaan vähäpätöinen peli sai Internetin seikkailupeliyhteisöissä yllättävän paljon huomiota teknisen toteutuksensa ansiosta. Sisällöllisesti peli on pelkkää dialogivalintojen tekemistä, mutta koukku piileekin siinä, miten pelin hahmot, rikostutkija ja silminnäkijä, on animoitu.

Sen lisäksi, että kaikki pelin dialogi äänitettiin, se myös kuvattiin teräväpiirtovideokameralla. Näyttelijöiden kasvoille piirrettiin merkkipisteitä, joiden liikkeet kaapattiin videokuvasta Adoben After Effects -ohjelmiston liikkeenseurantatoiminnolla (engl. motion tracking). Tuloksena saadun merkkipistedatan perusteella kasattiin vektorialgebran avulla kaksiulotteinen malli kasvoista. Pelissä tämä kaksiulotteinen malli piirrettiin ohjelmallisesti kaksiulotteisena vektorigrafiikkana, täytettyinä kolmioina mallin määrittävien pisteiden välillä (kuva 2.1). Näin saatiin aikaan yksinkertaista, näyttelijän ilmeilyä seuraavaa kasvoanimaatiota, johon monet pelaajat ihastuivat.

Järjestelmässä oli kuitenkin puutteensa. Ensinnäkin näyttelijää kuvaava kamera oli paikoillaan kolmijalan päällä, joten näyttelijät eivät saaneet kääntää päätään juuri ollenkaan; muutoin kasvojen merkkipisteet olisivat hävinneet kuvasta tai niiden väliset suhteet olisivat vääristyneet. Toiseksi kuvamateriaalin käsittely oli äärimmäisen aikaa vievää. Videotehosteohjelmisto After Effectsillä merkkipisteiden seuraaminen oli hankalaa: Jokaista videoleikettä varten kunkin merkkipisteen sijainti, koko ja etsintäalue jouduttiin määrittelemään käsin. Varsinaista automaattista liikkeenseurantaakin joutui valvomaan jatkuvasti, sillä mikäli ohjelmisto kadotti hetkellisesti jonkin merkkipisteistä, seuranta keskeytyi ja käyttäjä joutui käsin siirtämään merkkipisteen oikealle paikalleen. Osasyyl-

1 http://www.adventuregamestudio.co.uk/site/games/game/1335/

lisiä merkkipisteiden katoamiseen olivat näyttelijän pään kääntyily sen kieltämisestä huolimatta ja ohjelmiston merkkipisteiden etsintäalueet. Etsintäalueet rajoittavat kunkin merkkipisteen etsinnän tietylle etäisyydelle edellisestä havainnosta ja siten nopeuttavat liikkeenseurantaa. Se kuitenkin edellyttää, että käyttäjä on onnistunut määrittelemään seuranta-alueet juuri oikeanlaisiksi, jotta piste ei ehdi kadota nopeassa liikkeessä.

Kuva 2.1. *Vektorigrafiikalla muodostetut kasvot.*

Tässä työssä toteutetun järjestelmän on tarkoitus korjata näitä puutteita. Materiaalin käsittelyä nopeutetaan ja automatisoidaan. Pään kääntelyn rajoite käännetään eduksi kaappaamalla myös päänliikkeet, jolloin animaatioon saadaan lisää eloa. Lisäksi animaatio tuotetaan 2D-vektorigrafiikan sijaan 3D-malleille, tarkemmin sanoen niitä ohjaaville luurankomalleille. Seuraavissa kohdissa tutustutaan menetelmiin, joilla näitä tavoitteita on mahdollista saavuttaa.

2.2 Liikkeenseuranta ja -kaappaus

Liikkeenseuranta on nimensä mukaisesti tehtävän kannalta oleellisten kohteiden sijainninmuutosten seuraamista jonkin lähdemateriaalin perusteella. Lähdemateriaali voi olla monenlaista: optista, kuten tavallinen videokuva, sähkömagneettisuuteen tai radioaaltoihin perustuvaa tai vaikkapa yksittäisten pisteiden itsensä suorittamaa omien sijainninmuutostensa seurantaa erilaisin liikkeisiin reagoivien sensorien avulla. Seurattavat kohteet puolestaan voidaan paikantaa joko kahdessa tai kolmessa ulottuvuudessa.

Kaksiulotteisesta paikantamista on esimerkiksi videokuvaan perustuva liikkeenseuranta (engl. video tracking), jossa seurataan kuvamateriaalissa olevien merkkipisteiden tai muiden korkeakontrastisten alueiden liikkeitä. Esimerkiksi televisio- ja elokuvateollisuudessa tätä hyödynnetään vaikkapa liittämään kuvaan objekteja, jotka liikkuvat kuvan mukana, vakauttamaan kuvan tärinää tai jopa ratkaisemaan kameran liikkeen videokuvassa [57]. Tässä työssä kasvonliikkeitä seurataan nimenomaan videokuvan pohjalta kahdessa ulottuvuudessa.

Kolmiulotteista paikantamista voidaan puolestaan hyödyntää kolmiulotteisten objektien tai hahmojen liikkeenkaappaukseen (engl. motion capture, mocap; myös performance capture, etenkin kun kasvojenkaappaus sisältyy prosessiin), jolloin esimerkiksi ihmishahmon 3D-malli saadaan liikkumaan oikean ihmisen liikkeiden mukaisesti. Tätä on jo vuosia käytetty monissa elokuvissa ja peleissä tietokoneella luotujen hahmojen kehonliikkeiden kaappaamiseen, mutta kasvoanimaatioissa se on toistaiseksi ollut suhteessa vähäistä; kasvot on usein animoitu manuaalisesti referenssivideon pohjalta avainkuvatekniikalla (engl. keyframe) [59]. Liikkeenkaappaus rajoittuu yleensä tietylle kaappausalueelle (engl. capture volume), jota ympäröivät esimerkiksi kamerat, jotka kuvaavat kaikkea kaappausalueella liikkuvaa.

Seuraavaksi tarkastellaan tarkemmin liikkeenseurantaa pääasiassa viihdeteollisuuden ja liikkeenkaappauksen näkökulmasta. Lyhyen historiakatsauksen jälkeen esitellään erilaisia seurantatekniikoita, joita liikkeenkaappauksessa voidaan käyttää. Lopuksi keskitytään vielä nimenomaan kasvojen liikkeenkaappaukseen, joka on tämän työn ydinasiaa.

2.2.1 Historiaa

Videokuvaan perustuva liikkeenseuranta syntyi Seymourin [57] mukaan alkujaan Yhdysvaltain puolustusministeriön ajatuksesta kehittää ohjusten ohjausjärjestelmiä. Viihdeteollisuudessa liikkeenseurantaa alettiin hyödyntää seuraamalla jotakin liikkuvassa kuvassa näkyvää kohdetta. Kuvatehosteissa (engl. visual effects, VFX) ensimmäisiä sovelluskohteita olivat National Geographicille tehdyt mainokset, joissa liikkeenseurannalla vakautettiin taustana käytetyn videokuvan liikettä ja tärinää. FFT-muunnokseen (Fast Fourier Transform [67]) perustuvan, yhden pisteen seuraamista tukevan järjestelmän toteuttivat Tom Brigham ja J.P. Lewis vuonna 1985. [57]

Ajan kuluessa ja teknologian ja algoritmien kehittyessä liikkeenseurannan käyttökohteet lisääntyivät. Seurattavien pisteiden tuettu maksimimäärä kasvoi hiljalleen, mikä mahdollisti jo kahdella pisteellä kuvan kääntymisen ja skaalautumisen määrittämisen. Pisteitä lisäämällä ja kirjaamalla niiden todelliset fyysiset etäisyydet toisistaan pystyttiin lopulta jopa ratkaisemaan, miten videokuvan tallentanut kamera liikkuu kuvassa. Kun kameran liike on selvillä, kolmiulotteisten graafisten elementtien liittäminen videokuvaan helpottuu huomattavasti. [57] Useissa tapauksissa liikkeenseurantaa varten kuvattavaan ympäristöön tai kohteeseen liitetään korkeakontrastisia merkkipisteitä (engl. tracking marker), jotka erottuvat selkeästi muusta kuvasta ja ovat siten helpompia seurata. Kuitenkin jo vuonna 1991 kehitettiin yksi ensimmäisistä merkkipisteettömistä järjestelmistä Star Trek: The Next Generation -televisiosarjan tarpeisiin [57].

Nykyteknologialla voidaan jo seurata jokaista videokuvan pikseliä tai ainoastaan helposti havaittavia, kuvan korkeakontrastisia kulmakohtia, joissa sekä kuvan X- että Y-akselin suuntainen derivaatta muuttuu jyrkästi [3, s. 317]. Tällaista koko videokuvaan kohdistuvaa liikkeenseurantaa kutsutaan Seymourin mukaan optiseksi vuoksi (engl. optical flow). Optisen vuon käyttökohteina on yksinkertaisemman liikkeenseurannan tavoin

esimerkiksi kameran liikkeen selvittäminen, mutta lisäksi sillä voidaan vaikkapa pyrkiä leikkaamaan liikkuvat kohteet paikallaan olevasta taustasta. [56] Kuvatehosteteollisuuden ulkopuolisia sovelluskohteita ovat esimerkiksi videokuvan pakkaaminen [56] ja autojen kuskia avustavat näköjärjestelmät [22].

Liikkeenkaappaus on eräs tapa hyödyntää liikkeenseurantaa. Siinä liikkeenseurannan avulla seurataan elävän olion tai kappaleen liikkeitä ja siirretään ne tietokoneella luodulle oliolle sen animoimiseksi. Pelialalla liikkeenkaappaus lieneekin yleisin liikkeenseurannan käyttökohde. Menachen [40] ja Sturmanin [63] mukaan liikkeenkaappaus sai viihdeteollisuudessa alkunsa 1900-luvun alkupuoliskolla piirrosanimaatioissa. Kyseessä oli rotoscope-tekniikka, jossa kuvattiin oikeita ihmisiä ja eläimiä filmille ja kaapattiin niiden liikkeet yksinkertaisesti asettamalla paperi tai animaatiokalvo filmikuvan päälle ja piirtämällä läpi näkyvä kuva uudelleen piirrosanimaation vaatimalla tyylillä. [40; 63] Kuuluisimpia esimerkkejä tämän tekniikan käytöstä lienevät Walt Disneyn Lumikki ja seitsemän kääpiötä (1937) ja jotkin Ralph Bakshin animaatiot kuten Taru sormusten herrasta (1978).

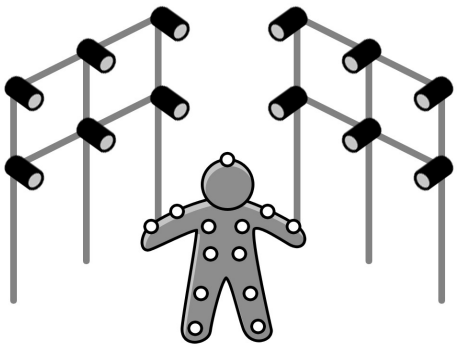
Tietokoneilla suoritettava automatisoitu liikkeenkaappaus puolestaan otti ensiaskelensa 1970–80-lukujen taitteessa, alkujaan lääketeollisuuden käytössä arvioimaan ihmiskehon liikkeitä ja havaitsemaan niissä epätavallisuuksia. Sieltä teknologia loikkasi viihdeteollisuuden puolella ensiksi yliopistojen tutkimusprojekteihin, sitten robotiikalla ohjattujen nukkehahmojen ja lopulta tietokoneella luotujen hahmojen animointiin. Eräs ensimmäisiä tietokoneella luotuja ja liikekaapattuja hahmoja oli Muppet-hahmojen täyteisessä The Jim Henson Hour -televisiosarjassa Pacific Data Images -yrityksen avulla vuonna 1988 toteutettu Waldo C. Graphic, jota ohjattiin nukkea muistuttavalla ohjauslaitteella kuten fyysistäkin Muppettia. [40]

Peliteollisuudessa liikkeenkaappaus alkoi vastaavaan tapaan rotoscope-tekniikalla toteutetussa grafiikassa, kuten pelissä Prince of Persia vuodelta 1989, jolla on kunnia olla Guinnessin ennätystenkirjassa ensimmäisenä liikkeenkaappausta hyödyntävänä pelinä [20]. 3D-hahmojen liikkeenkaappaus sen sijaan saapui kuviin, kun 3D-grafiikkakin alkoi yleistyä peleissä; esimerkiksi vuonna 1995 ilmestyneessä pelissä Highlander: The Last of the MacLeods käytettiin liikkeenkaappausta yksinkertaisten 3D-hahmojen reaaliaikaiseen (siis ei esirenderöityyn) animointiin [35]. Nykyisin liikkeenkaappausta voisi jo pitää pikemminkin sääntönä kuin poikkeuksena ihmishahmojen animoinnissa. Eräitä viime vuosien kiinnostavia teknologisia esityksiä ovat olleet esimerkiksi pelit L.A. Noire ja Far Cry 3, joissa kummassakin kaapattiin kehonliikkeiden lisäksi myös kasvoanimaatiota, vaikkakin hyvin erilaisilla tekniikoilla.

2.2.2 Liikkeenkaappaustekniikoita

Seuraavaksi esitellään erilaisia tekniikoita, joilla liikkeenkaappausta on mahdollista toteuttaa. Tekniikoiden kuvaukset perustuvat Alberto Menachen liikkeenkaappauksesta kertovaan kirjaan [40] keskittyen ihmiskehon liikkeisiin kasvonliikkeiden sijaan.

Menache jakaa järjestelmät kolmeen kategoriaan perustuen siihen, mihin kaappauksen kohteet (engl. capture source, yksittäinen merkkipiste oli se sitten käytännössä millainen tahansa) ja niitä tarkkailevat sensorit on sijoitettu. Ensimmäisessä kategoriassa, ulkoa sisään (engl. outside-in), ulkoiset sensorit keräävät dataa näyttelijän keholle asetetuista kaappauskohteista. Tällaisia järjestelmiä ovat esimerkiksi kameroihin perustuvat järjestelmät. Toisessa kategoriassa, sisältä ulos (engl. inside-out), keholle asetetaan sensoreita, jotka keräävät dataa ulkoisista kohteista. Tällaisia ovat esimerkiksi sähkömagneettiset järjestelmät, joissa sensorit liikkuvat ulkoisesti generoidussa sähkömagneettisessa kentässä. Viimeisessä kategoriassa, sisältä sisään (engl. inside-in), sekä kohteet että sensorit on asetettu keholle. Tällaisia ovat esimerkiksi sähkömekaaniset tai inertiaasut, joiden sensorit ovat potentiometrejä, goniometrejä tai kiihtyvyyssensoreita ja gyroskooppeja, ja joiden kohteet ovat kehon niveliä. [40] Seuraavaksi esitellään keskeisimpiä näiden kategorioiden edustajia.



Kuva 2.2. *Esimerkki optisesta liikkeenkaappausjärjestelmästä.*

Optinen järjestelmä (kuva 2.2) koostuu yleensä 8–32 kamerasta, jotka kaappaavat heijastavia merkkipisteitä 30–2000 kuvan sekuntinopeudella ja jopa 16 megapikselin tarkkuudella. Kussakin kamerassa on tyypillisesti oma valolähteensä, jotta pallomaiset, heijastavalla materiaalilla päällystetyt merkkipisteet heijastaisivat valoa kameran suuntaan. Käytetty valo on yleensä joko punaista tai infrapunaista. Merkkipisteet voivat myös olla aktiivisia ja tuottaa oman valonsa eri taajuuksisilla LED-pulsseilla, mikä mahdollistaa myös merkkipisteiden automaattisen yksilöinnin [42]. Optisen järjestelmän kalibrointi tapahtuu liikuttamalla tunnetun kokoista, merkkipisteillä varustettua kappaletta, kuten kuutiota tai sauvaa. Tällöin kunkin kameran näkemän kuvan ja kappaleen mittojen perusteella voidaan laskea kameroiden sijainnit. Jotta pisteen kolmiulotteinen sijainti saataisiin selville, on sen oltava vähintään kahden kameran näköpiirissä. Tästä syystä kameroita on oltava riittävästi varmistamaan, että jokaisen pisteen sijainti pystytään aina laskemaan. [40]

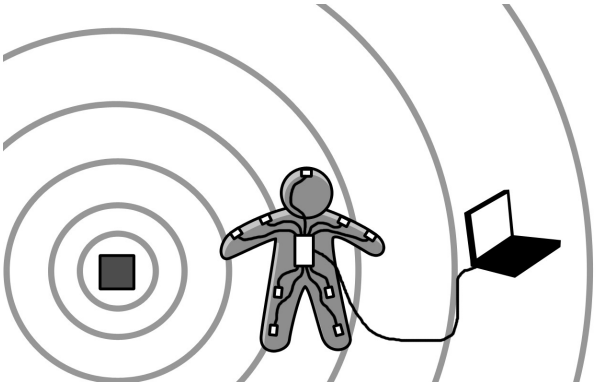
Kaapatun kuvamateriaalin jälkikäsittely aloitetaan suodattamalla kunkin kameran kuvaa niin, että jäljelle jää ideaalisesti ainoastaan merkkipisteet. Tämä onnistuu yksinker-

taisimmillaan vertaamalla kuvapisteryhmien kirkkausarvoa johonkin raja-arvoon. Seuraavaksi määritetään kunkin pisteen kaksiulotteiset koordinaatit kunkin kameran kuvassa, mikä voi vaatia käyttäjän toimia eri merkkipisteiden tunnistamiseksi. Lopulta nämä useasta kamerasta saadut pisteen koordinaatit yhdistettynä kameroiden sijainteihin tuottavat lopulliset kolmiulotteiset koordinaatit merkkipisteelle. Optisen järjestelmän hyödyiksi mainittakoon datan tarkkuus, korkea näytteenottotaajuus (jopa 2 000 kuvaa sekunnissa), suuri kaappausalue, mahdollisuus seurata erittäin monia pisteitä, ja helppo merkkipistekonfiguraation muutettavuus. Vastaavasti heikkouksia ovat materiaalin jälkikäsittelyn vaatima työmäärä, kallis laitteisto ja tarve sille, että kameroilla on näköyhteys merkkipisteisiin. [40]

On myös olemassa optisia järjestelmiä, joissa ei käytetä lainkaan merkkipisteitä, vaan liikkeenseuranta tapahtuu analysoimalla kuvassa tai kuvissa tapahtuvaa liikettä edellä mainitun optisen vuon tavoin. Merkkipisteetön liikkeenkaappaus on ollut viime vuosikymmenen aikana suosittu tutkimuksen aihe. Stanfordin yliopiston Markerless Motion Capture -projektin kotisivut [61] tarjoavat selkeän kuvauksen erään tällaisen liikkeenkaappausjärjestelmän vaiheista. Ensimmäinen vaihe on luonnollisesti kerätä kuvamateriaalia, tässä tapauksessa kahdeksasta kaappausalueen ympärille sijoitellusta videokamerasta. Seuraavaksi liikkuva kohde irrotetaan taustasta kunkin kameran videokuvassa ja tiedot liikkuvista kuvapisteistä yhdistetään, jolloin saadaan kolmiulotteinen esitys liikkuvalle kohteelle, siis kolmiulotteinen pistejoukko. Tähän pistejoukkoon pyritään sovittamaan ihmiskehon malli. Malliin liitetty tieto kunkin nivelen keskipisteistä auttaa lopulta ratkaisemaan ihmisen luurangon liikkeet, joilla voidaan animoida varsinaista 3D-mallia. Järjestelmän suunnittelussa on keskitytty terveydenhuollon sovelluskohteisiin kävelyn analysoinnista urheilusuorituksiin ja loukkaantumisten estämiseen. [61] Lisätietoa on saatavilla järjestelmään liittyvissä julkaisuissa [8; 9; 43; 44]. Muita näkökulmia aiheeseen tarjoavat esimerkiksi lähteet [1; 29; 50]. Eräs mielenkiintoinen kuluttajakäytössäkin oleva järjestelmä on myös Microsoftin Kinect, joka hahmottaa syvyysnäkymän ja tunnistaa ihmishahmot ja niiden liikkeet infrapunaprojektorin ja kahden kameran avulla [10].

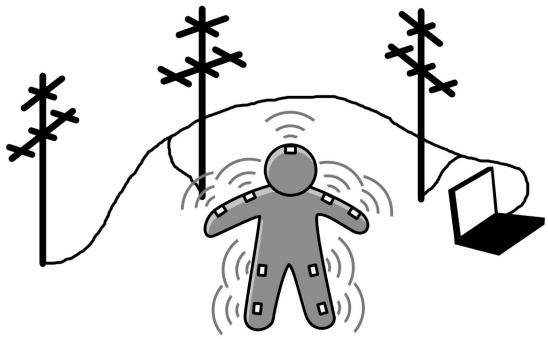
Sähkömagneettinen järjestelmä (kuva 2.3) koostuu Menachen [40] mukaan vastaanottimista, jotka mittaavat sijaintejaan suhteessa läheiseen lähettimeen. Vastaanottimet sijoitetaan näyttelijän keholle ja yhdistetään johdoilla ohjausyksikköön. Vastaanottimia on yleensä 11–18 kappaletta, mutta nykyaikaisimmissa järjestelmissä niitä voi olla jopa 90 kappaletta, ja niiden näytteenottotaajuus voi olla jopa 144 näytettä sekunnissa. Järjestelmän lähetin generoi matalataajuuksisen sähkömagneettisen kentän, jonka vastaanottimet havaitsevat ja ilmoittavat mittauslukemansa ohjausyksikölle. Ohjausyksikkö suodattaa ja käsittelee vastaanottimilta saamansa datan ja välittää sen tietokoneelle, jolla ohjelmisto laskee kunkin vastaanottimen sijainnin ja asennon kolmessa ulottuvuudessa. Tällaisen järjestelmän hyötyjä ovat esimerkiksi reaaliaikaisuus, jälkikäsittelytarpeen jääminen pois, matalampi hinta kuin optisissa järjestelmissä, mahdollisuus kaapata useita

hahmoja kerralla ja se, että merkkipisteisiin on aina ”näköyhteys”. Haittapuolina sen sijaan ovat järjestelmän herkkyys lähellä oleville metalleille, johtojen tarve joissain tapauksissa, matalahko näytteenottotaajuus, pieni kaappausalue ja merkkipistekonfiguraation muuttamisen hankaluus. [40]



Kuva 2.3. *Esimerkki sähkömagneettisesta liikkeenkaappausjärjestelmästä.*

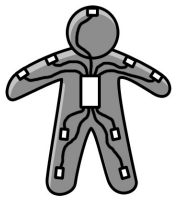
Radiotaajuuspaikannus (kuva 2.4) on tuoreempi, suosioltaan nousussa oleva tapa käyttää radioaaltoja sijainnin määrittämiseen. Valitettavasti nykyjärjestelmien tarkkuus ei vielä riitä kunnolliseen hahmoanimaatioiden kaappaukseen. Tunnetuin esimerkki tällaisesta järjestelmästä on maailmanlaajuinen paikannusjärjestelmä GPS (Global Positioning System), joka perustuu paikannettavan laitteen ja satelliittien väliseen tiedonvaihtoon. GPS:n tarkkuus ja näytteenottotaajuus ovat kuitenkin erittäin matalia. Tulevaisuudessa huomio saattaakin keskittyä paikalliseen paikannusjärjestelmään eli LPS:ään (Local Positioning System), joka mittaa sijaintidataa erittäin tarkasti ja korkealla näytteenottotaajuudella.



Kuva 2.4. *Esimerkki radiotaajuuspaikannusjärjestelmästä.*

Paikannus perustuu aikaan, joka lähetetyltä radiosignaalilta kestää kulkea lähettimeltä vastaanottimeen. Signaalit ovat kuitenkin äärimmäisen nopeita, mikä rajoittaa paikan-

nuksen tarkkuutta. Järjestelmässä lähettimet sijoitetaan keholle ja vastaanotinantenneja sijoitetaan kaappausalueen ympärille. Saatuaan signaalin antennit lähettävät tiedon signaalinkäsittely-yksikölle, joka laskee lähettimen sijainnin käyttäen apuna tunnettuja antennien sijainteja. LPS:n hyötyjä ovat reaaliaikaisuus, suuri kaappausalue sisällä tai ulkona, mahdollisuus pitää lähettimiä vaatteiden alla, valmistuksen ja käytön halpuus, automaattinen kalibrointi, laajennettavuus käyttämään jopa tuhansia lähettimiä ja lähettimien automaattinen tunnistettavuus niiden lähettämien tunnistesignaalien perusteella. Haittoja ovat mahdolliset häiriöt muiden radiosignaalien kanssa sekä se, että LPS-signaalit eivät läpäise metallisia pintoja. [40]



Kuva 2.5. *Esimerkki sähkömekaanisesta liikkeenkaappausjärjestelmästä.*

Sähkömekaaninen järjestelmä (kuva 2.5) pyrkii mittaamaan kaikkia ihmisen raajojen kiertymiä käyttäen potentiometrejä, gyroskooppeja tai muita liikekulmia mittaavia laitteita, jotka sijoitetaan nivelien kohdalle. Nykyaikaisimmat järjestelmät käyttävät näitä pienempiä ja tarkempia MEMS-inertiasensoreita (Microelectromechanical Systems), jotka kiinnitetään näyttelijän asuun. Järjestelmän etuja ovat laaja kaappausalue, halpa hinta, järjestelmän liikuteltavuus, reaaliaikaisuus, mahdollisuus kaapata useita hahmoja kerralla ja taattu ”näköyhteys” merkkipisteisiin. Haittoja ovat matala näytteenottotaajuus, kömpelyys laitemäärän takia, lukittu merkkipistekonfiguraatio ja se, että järjestelmä ei pysty määrittämään sensorien globaaleja sijainteja. Järjestelmä ei myöskään pysty tunnistamaan kaikkia pallonivelten, kuten olkapään, kiertymiä. [40]

2.2.3 Kasvonliikkeiden kaappaaminen

Kasvonliikkeiden seuranta poikkeaa hieman kehonliikkeiden seurannasta: tarkkuutta vaaditaan enemmän, sillä kasvoilla pienetkin liikkeet voivat olla merkityksellisiä, kaappausalue on huomattavasti pienempi eikä merkkipisteitä voi asettaa kaikkiin seurattaviin kasvojen alueisiin, kuten silmiin tai kieleen. Kasvoilla käytettävät merkkipisteet ovat myös tavallista pienempiä ja ne on aseteltu tiheästi, mikä on varmasti osasyynä siihen, että kasvot on perinteisesti ollut tapana kaapata kehonliikkeistä erillään [37, s. 137]. Tästä kuitenkin seuraa epäyhtenäisyyttä kehon- ja kasvonliikkeiden välille. Esimerkiksi edellä mainittu Far Cry 3 on ottanut asian huomioon ja kaapannut näyttelysuoritukset kokonaisuudessaan dialogeineen yhdellä kerralla [58]. Esimerkeiksi kasvonliikkeiden

kaappauksessa käytetyistä tekniikoista on valittu pelit Far Cry 3 ja L.A. Noire, jotka molemmat käyttävät optisia järjestelmiä, mutta joista toinen perustuu merkkipisteisiin ja toinen merkkipisteettömyyteen.

Alun perin Crytekin kehittämässä, mutta Ubisoftille myydyssä peliprojektissa Far Cry 3 vuodelta 2012 päätettiin tehdä täysi näyttelysuorituksen kaappaus perinteisellä merkkipisteisiin perustuvalla tekniikalla. Kehonliikkeet, mukaan lukien sormien liikkeet, kaapattiin optisella järjestelmällä, jolla pystyi kaappaamaan kerralla korkeintaan 6–7 näyttelijän liikkeet riippuen siitä, kuinka lähekkäin heidät on sijoiteltu, koska kameroiden näköyhteys merkkipisteisiin on säilytettävä. Kasvonliikkeet kaapattiin kaksiulotteisesti kypärään kiinnitetyllä infrapunakameralla ja samalla äänitettiin kypärämikrofonin avulla lopullinen dialogi. Kasvon- ja silmänliikkeet muunnettiin animaatioksi opettamalla järjestelmä tunnistamaan liuta ennalta määritettyjä ilmeitä ja katsesuuntia, vertaamalla kaappaustulosta näihin malleihin ja sekoittamalla keskenään 3D-mallille määriteltyjä vastaavia ilmeitä. [58] Subjektiivisesti arvioituna tällainen kasvonliikkeiden kaappaaminen ei ainakaan tämän pelin lopputuloksen perusteella tuota erityisen luonnollista ja ilmeikästä animaatiota, kenties koska ilmeet ovat ennalta määritettyjä ja niitä on rajoitettu määrä.

Australialainen pelikehittäjä Team Bondi otti hyvin erilaisen lähestymistavan vuonna 2011 julkaistun pelinsä L.A. Noire kasvoanimaatioihin. Tässäkin pelissä kehon liikekaappaus suoritettiin perinteisellä optisella tavalla, mutta kasvot kirjaimellisesti kaapattiin näyttelijöistä eikä minkäänlaisia merkkipisteitä käytetty [34]. Tämä Depth Analysis -yrityksen kehittämä MotionScan-järjestelmä mahdollistaa näyttelijän pään muuttamisen kolmiulotteiseksi malliksi kaikkine liikkeineen ja tekstuureineen. Järjestelmä koostuu 32 teräväpiirtoisesta videokamerasta, jotka on asetettu pareittain täsmällisesti ympäri tuolia, jolla näyttelijä istuu. Itse kaappaushuone on valaistu tasaisesti valkoisella valolla, jotta varjoja ei syntyisi näyttelijän kasvoille. Liikkumavaraa näyttelijällä ei ole paljoa, ainoastaan noin 50 cm verran, jotta hän ei katoa kaappausalueelta. [4] Kullakin 16 kameraparista generoidaan kolmiulotteinen osa kasvojen pinnasta hyödyntäen kameraparin stereonäkymän poikkeamia (engl. stereo reconstruction [53]). Näistä pintapaloista muodostetaan yhtenäinen pistejoukko, joka suodatetaan vähäkohinaiseksi ja sulavaksi, ja jonka päälle sovitetaan kolmioverkko (engl. mesh), jonka pelilaitteiden grafiikkapiirit osaavat piirtää näyttöruudulle. Kolmioverkkojen sarja teksturoidaan kaapatun videomateriaalin mukaisesti ja pakataan loppukäyttöä varten. [33]

Järjestelmän merkittävin rajoite lienee se, että kaapattu animaatio on pitkälti lopullista eikä sitä voi esimerkiksi siirtää toiselle hahmolle. Näyttelijän on oltava maskeerattu täsmälleen hahmonsa kaltaiseksi, koska sekä 3D-malli että sen pinnoite (tekstuuri) kaapataan automaattisesti hänen kasvoiltaan. Järjestelmää on myös vaikea kuvitella laajennettavaksi siten, että sekä kehon että kasvojen liikkeet pystyttäisiin kaappaamaan samanaikaisesti, ainakaan kun kaapattavana on useita hahmoja, jotka ovat keskenään vuorovaikutuksessa. Toisaalta järjestelmän selkeä etu on se, että se kaappaa lähestulkoon

kaikki näkyvät pään ja kasvojen liikkeet: jos näyttelijä työntää kielensä ulos, järjestelmän syvyysanalyysi generoi kolmioverkkoon kielen sellaisena kuin se todellisuudessakin on. Samalla tavoin ihonliikkeet ja rypyttymisetkin tallentuvat. Merkkipisteisiin perustuvalla järjestelmällä tämä on, jos ei mahdotonta, niin ainakin hyvin hankalaa.

Kaupallisia kasvoanimaation kaappausjärjestelmiäkin on monenlaisia, niin merkkipisteisiin perustuvia, kuten CaptiveMotion Embody [5], OptiTrack Expression [46] ja EasyCap Studio Facemotion [17], kuin merkkipisteettömiäkin, kuten Faceshift [18] ja Maskarad [38]. Hinnoiltaan nämä järjestelmät vaihtelevat sadoista euroista tuhansiin, kertamaksuisiin ja vuoden mittaisiin lisensseihin. Lisäksi suuremman budjetin tuotannoissa on mahdollisuus kääntyä koko liikkeenkaappausprosessin suorittavan yrityksen puoleen, jolloin hintaan saattaa sisältyä niin kaappaustilan ja -laitteiston vuokra kuin kaappausdatan käsittelykin.

Tässä työssä keskitytään optiseen, merkkipisteisiin perustuvaan kasvonliikkeiden kaappaukseen, joka suoritetaan yhdellä, kasvoja edestäpäin kuvaavalla kameralla käyttäen kaksiulotteista liikkeenseurantaa ja muuntamalla liikkeet 3D-mallin kasvojen luurakenteen liikkeiksi. Seuraavassa kohdassa tutustutaan hieman työssä käytettävän, videokuvaan pohjautuvan liikkeenseurannan toteuttamiseen analysoimalla tallennettuja kuvia.

2.3 Videokuvan liikkeenseuranta

Videokuvan pohjalta tapahtuva liikkeenseuranta jakaantuu kahteen päävaiheeseen: kiinnostavien kohteiden tunnistamiseen ja näiden kohteiden liikkeen mallintamiseen, jotta samat kohteet pystytään löytämään seuraavista kuvista. Ensimmäiseen vaiheeseen, kohteiden tunnistamiseen, on olemassa useita lähestymistapoja, jotka voi jakaa karkeasti kahteen kategoriaan. Ensimmäisessä kategoriassa pyritään erottelemaan kuvan kiinnostavat alueet, eli etuala, kuvan taustasta, eli taka-alasta. Toisessa kategoriassa kuvasta yritetään etsiä kiinnostavat kohteet vertaamalla niitä malleihin, jotka voivat olla kuvia tai muuta kuvaavaa dataa. Näitä kohteiden paikannustapoja sekä liikkeen mallintamista käsitellään seuraavaksi.

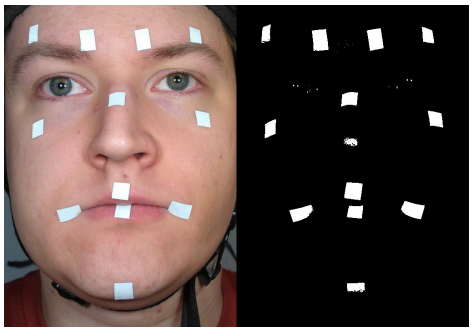
2.3.1 Etualan ja taustan erottelu

Etu- ja taka-alojen erottelua (engl. segmentation) voidaan lähestyä kahdelta kannalta: joko pyritään mallintamaan kuvan taustaa ja poistamaan se myöhemmistä kuvista tai yritetään erotella etuala sen perusteella, mitä tiedetään etualaan kuuluvien kohteiden ulkonäöstä. Näistä lähestymistavoista taustan mallintaminen on yleiskäyttöisempi siinä mielessä, että se ei teoriassa rajoita etualaksi kelpaavia kohteita; riittää että ne poikkeavat tarpeeksi taustasta. Kuvaava esimerkki taustan poistamisesta on kuvatehosteteollisuudessa paljon hyödynnettävä sinisen tai vihreän taustan poistaminen (engl. chroma keying) ja etualaksi lukeutuvien näyttelijöiden, esineiden ja lavasteiden säilyttäminen kuvassa. Vastakohtana lavasteiden hallittavissa oleville olosuhteille ovat esimerkiksi

valvontakamerajärjestelmät, jotka pyrkivät luonnonoloissa erottamaan etualan kohteita alati muuttuvasta taustasta. Tehtävä ei missään nimessä ole helppo. Toyama et al. [65] luettelevat ongelmakohtia, joista ideaalisen, taustaa mallintavan järjestelmän pitäisi selviytyä:

- Liikkuvia, taka-alaan lukeutuvia kohteita voidaan liikutella.
- Valaistus voi muuttua joko yllättäen tai hitaasti ajan kuluessa.
- Puut ja muu kasvillisuus voivat huojua tuulessa.
- Etualaan kuuluva kohde voi olla taustan kaltainen.
- Järjestelmällä ei välttämättä ole aikaa muodostaa mallia taustasta.
- Yhtenäisen värisen etualan kohteen sisäpikselien liikettä ei voida havaita.
- Liikkumaton etualan kohde voidaan tunnistaa taka-alaksi.
- Taka-alaan lasketun kohteen liikkuessa sekä se että sen takaa paljastuva alue havaitaan muuttuviksi.
- Etualan kohteiden synnyttämät varjot eroavat mallinnetusta taustasta. [65]

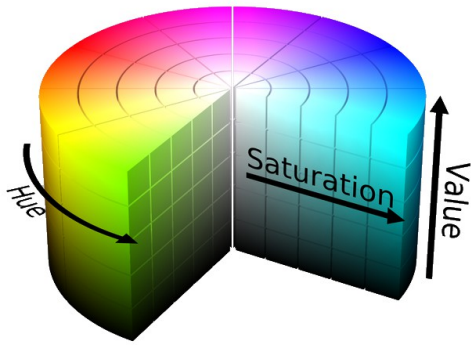
Tähän työhön taustan mallintaminen ja poistaminen ei ole järkevä ratkaisu, sillä etualaan halutaan sisältyvän vain merkkipisteet, eikä taka-ala (siis kasvot ja muu tausta) pysy liikkumattomana näyttelijän ilmeillessä ja käännellessä päätään. Käytännöllisempi tapa on hyödyntää tietoa käytössä olevien merkkipisteiden ulkonäöstä ja sisällyttää vain niiden kaltaiset kuva-alueet etualaan.



Kuva 2.6. *Etualan erottelu kuvapisteiden kirkkauden perusteella.*

Yksinkertaisimpia ja suoritusajaltaan nopeimpia tapoja etualan erotteluun ovat kuvapisteiden väriarvoihin kohdistuvat raja-arvovertailut (engl. thresholding): voidaan esimerkiksi rajata etualaksi kuvapisteet, jotka ovat enimmäkseen sinisiä, tai vaikkapa pisteet, joiden kirkkaus ylittää tietyn raja-arvon (kuva 2.6). Tällöin tietysti järjestelmä rajoittaa käytettäviä merkkipisteitä etualaksi sallittavien väriarvojen kaltaisiksi. Väreihin perustuva rajaus suoritetaan usein HSV-väriavaruudessa (Hue, Saturation, Value), jossa väriarvo esitetään värisävyn H, värikylläisyyden S ja valöörin (kirkkaus) V kolmikkona. Näistä H esitetään usein astelukuna välillä 0–360º ja S ja V prosenttilukuna 0–100 %. HSV-komponenttien voidaan ajatella muodostavan yksikkölieriön (säde ja korkeus yh-

den yksikön mittaisia), jonka korkeusakselilla ilmaistaan kirkkaus V, etäisyydellä lieriön korkeusakselista värikylläisyys S, ja jonka ympyräpinnalle jakautuvat eri värisävyt H, kuten kuvassa 2.7.

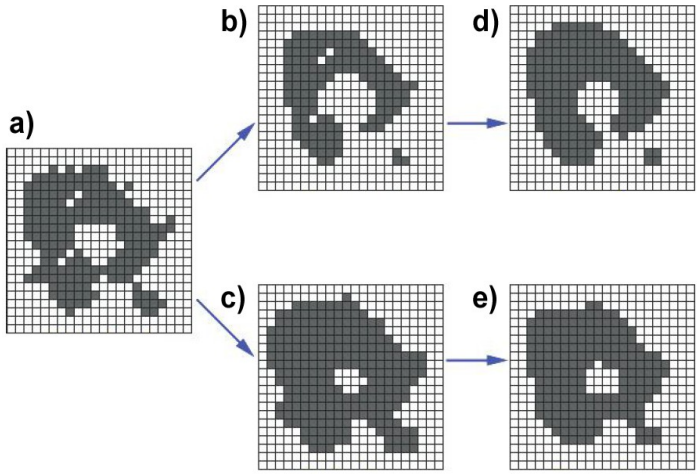


Kuva 2.7. *HSV-väriavaruus. Muokattu [24].*

Näin esimerkiksi sinisten kuva-alueiden rajaaminen onnistuu rajoittamalla värisävy vaikkapa välille 180–270º, syaanista punertavansiniseen. Kirkkaudella rajaaminen onnistuu vastaavasti kirkkauskomponenttiin vertaamalla. Ennen rajausta on väriarvot kuitenkin muunnettava HSV-väriavaruuteen, sillä kuva- ja videoformaateissa on käytössä monenlaisia väriavaruuksia, kuten RGB (Red, Green, Blue), jossa väriarvo jakaantuu punaiseen, vihreään ja siniseen komponenttiin, ja erilaiset YUV-värimuodot, joissa väri esitetään kirkkautena Y ja värikomponentteina U ja V [60].

Raja-arvorajauksen tuloksena saadaan yleensä binäärinen kuva, jossa kuhunkin kuvapisteeseen on tallennettu tieto siitä, kuuluuko se etu- vai taka-alaan. Joskus tämä binäärikuva vaatii vielä jälkikäsittelyä, jonka tavoitteena on muun muassa poistaa kohinaa, yhdistää pirstaloituneita alueita ja erottaa toisiaan lähellä olevia kohteita erillisiksi alueiksi [3, s. 115]. Tällaiseen jälkikäsittelyyn käytetään usein morfologisia, muotoon perustuvia, muunnoksia (engl. morphological transformation), joiden perusoperaatioita ovat kuluttaminen (engl. erosion) ja laajentaminen (engl. dilation). [41]

Nimensä mukaisesti kuluttaminen kutistaa etualan alueen pinta-alaa poistamalla pikseleitä alueen reunoilta. Laajentaminen toimii päinvastoin lisäämällä pikseleitä alueen reunoille. Operaatioissa käytetyn rakenne-elementin (engl. structuring element) koko ja muoto määräävät sen kuinka paljon pikseleitä lisätään tai poistetaan ja mistä kohdista. Yhdistämällä operaatioita saadaan operaatiot avaaminen (engl. opening), jossa ensin kulutetaan ja sitten laajennetaan, ja sulkeminen (engl. closing), jossa ensin laajennetaan ja sitten vasta kulutetaan. Näistä operaatioista avaaminen säilyttää alueen aukkoja ja karsii ulokkeita, kun taas sulkeminen tukkii aukkoja ja säilyttää ulokkeita. [41] Kuvassa 2.8 on havainnollistettu näitä operaatioita.



Kuva 2.8. *Morfologisia operaatioita: a) alkuperäinen kuva, b) kuluttaminen, c) laajentaminen, d) avaaminen, e) sulkeminen. Muokattu [51].*

Joskus on tarvetta erotella alueita toisistaan, jotta lähekkäin olevat etualan kohteet pystyttäisiin tunnistamaan erillisiksi kohteiksi ja seuraamaan niiden liikkeitä toisistaan riippumatta. Tällainen erottelu ei tämän työn kannalta ole tärkeää, sillä merkkipisteet ovat aina selkeästi toisistaan erillään. Mainittakoon kuitenkin, että erotteluun soveltuvia operaatioita, kuten watershed-algoritmi on esitelty esimerkiksi lähteissä [3; 51].

Etualan ja taustan erottelun viimeisenä vaiheena on vielä tunnistaa löydetyt, yhtenäiset etualan alueet. Tämä onnistuu etsimällä kunkin alueen reunaviivan (engl. contour). Reunaviiva esitetään luettelona pisteitä, jotka jollain tavoin esittävät alueen muotoa. Koska esitystapa on matemaattinen, voidaan löydetylle reunaviivalle esimerkiksi etsiä pienin mahdollinen suorakulmio, jonka sisään alue mahtuu, tai suorittaa sille muuta muotoanalyysiä. [3, s. 234–250] Yhtä lailla voidaan selvittää alueen keskipiste tai muu piste, jota käytetään liikkeenseurannassa kyseisen kohteen sijaintina.

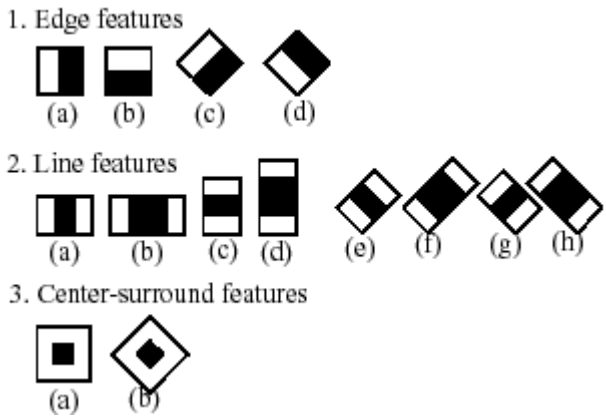
2.3.2 Malleihin perustuva etsintä

Monimutkaisempi mutta yleiskäyttöisempi tapa etualan kohteiden löytämiseen on käyttää etualan kohteiden etsimiseen ennalta määriteltyjä malleja. Malli voi olla esimerkiksi videokuvasta rajattu kohdealue (engl. region of interest, ROI), siis kuva tai useitakin kuvia, joita yritetään etsiä seuraavista kuvista. Malli voi myös olla yleisluontoisempi kuvaus etsittävästä kohteesta. Tällaista lähestymistapaa edustaa esimerkiksi Haar-tyyppisten piirteiden tunnistaminen.

Kuvapohjaista mallin sovittamista (engl. template matching) hyödyntävät tavalla tai toisella esimerkiksi aiemmassa prototyypissä käytetty kuvatehosteohjelmisto Adobe After Effects sekä Zdenek Kalalin väitöstyönään kehittämä oppiva TLD-järjestelmä (Tracking-Learning-Detection), toiselta nimeltään Predator [30; 31]. Kenties suoraviivaisin mallinsovitustapa on yksinkertaisesti verrata kiinnostavaa kuva-aluetta eli malli-

kuvaa toisen kuvan kanssa asettamalla se vuorollaan kunkin vertailukuvan kuvapisteen päälle. Mallikuva voidaan esimerkiksi asettaa niin, että mallin keskipiste tulee tarkasteltavan kuvapisteen kohdalle. Kunkin kuvapisteen kohdalla tallennetaan tieto siitä, kuinka hyvin mallikuva vastaa mallikuvan kokoista aluetta vertailukuvassa. Parhaan tuloksen saanut kuvapiste on tällöin todennäköisin mallikuvan sijainti. [3, s. 214–215] Varsinaiseen kuva-alueiden vertailuun on olemassa monia eri menetelmiä [3, s. 215–216], joita ei tässä käsitellä sen tarkemmin.

Toinen tapa etsiä tietyn mallin piirteet täyttäviä kohteita ovat harmaasävykuvilla operoivat Haar-luokittelijat (engl. Haar classifier). Menetelmän kehittivät alkujaan Viola & Jones [71]. Wilsonin & Fernandezin kuvauksen mukaan Haar-tyyppiset piirteet määritellään pikselien kirkkausarvojen sijaan muutoksina suorakulman muotoisten, vierekkäisten pikselijoukkojen välisessä kontrastiarvossa. Kontrastivaihteluilla erotellaan vaaleat alueet tummista. Koska pikselijoukkojen kokoa voidaan muuttaa, menetelmä skaalautuu hyvin eri kokoisten kohteiden tunnistamiseen. [70] Kuvassa 2.9 on esimerkkejä Haar-tyyppisistä piirteistä, joita yhdistelemällä pystytään muodostamaan malli monimutkaisemmistakin kohteista, kuten ihmiskasvoista.



Kuva 2.9. *Haar-tyyppisiä piirteitä [6].*

Haar-luokittimia yhdistellään vesiputousmaiseksi sarjaksi (engl. cascade), jonka kullakin askeleella rajataan pois kuvan kohtia, siis alikuvia, jotka eivät sisällä etsittävää kohdetta. Jos alikuva läpäisee koko luokittimien sarjan, on etsittävä kohde tunnistettu alikuvan alueelta. Mitä pidemmälle sarjassa edetään sitä enemmän piirteitä analysoidaan kullakin askeleella. Näin selkeästi kohdetta sisältämättömät alikuvat pystytään hylkäämään aikaisessa vaiheessa prosessin nopeuttamiseksi. [70]

Haar-luokittimien sarja muodostetaan kouluttamalla, mikä voi tehdä menetelmän käyttöönotosta työlästä. Kouluttamiseen tarvitaan kaksi kuvajoukkoa, joista toisessa on tunnistettavaksi tarkoitettu kohde (positiiviset kuvat) ja toisessa ei (negatiiviset kuvat). Kukin kuvassa esiintyvä kohde on määriteltävä kertomalla, millä kuvan alueella se sijaitsee. Wilsonin & Fernandezin kouluttamissa luokittimissa ihmisen silmille, nenälle ja

suulle käytettiin 5 000 negatiivista kuvaa ja 1 500 positiivista kuvaa. Luokittimilla silmät ja nenä löytyivät 93% ja 100% testitapauksista, kun taas suu löytyi vain 67% tapauksista. Kullakin luokittimella virheelliset tunnistukset vaihtelivat 23–29% välillä. [70] Jo tämän perusteella voidaan päätellä, että mallien tunnistamiseen perustuvat menetelmät ovat järkeviä pääasiassa yleiskäyttöisissä tilanteissa. Sen sijaan käytettäessä ennalta määrätynlaisia merkkipisteitä kontrolloidussa ympäristössä, kuten tässä työssä on tehty, on luotettavampi ja suorituskykyisempi ratkaisu aiemmin mainitut etualan kuva-alueiden erotteluun perustuvat menetelmät.

2.3.3 Kohteen liike

Kun seurattava kohde on pystytty tunnistamaan, täytyy sitä hakea myös seuraavista kuvista. Yksittäisen kuvan tapauksessa riittäisi, että kuvasta löydetään kaikki haetut kohteet, mutta kuvasarjassa on vielä lisäksi pystyttävä tunnistamaan, mihin edellisestä kuvasta tunnistettu kohde on seuraavassa kuvassa liikkunut. Yleisluontoisessa tapauksessa tämä ei aina ole yksinkertaista, jos esimerkiksi kaksi kohdetta lähestyvät toisiaan, ilmenevät yhdessä kuvassa päällekkäin yhtenä kohteena ja jatkavat sitten matkaansa toistensa ohi. Liikkeen määrittämisessä on suotavaa tehdä oletus lähdevideon kuvien ajallisesta ja paikallisesta jatkuvuudesta, mikä tarkoittaa että kukin kuva on ensinnäkin ajallisesti suurin piirtein yhtä etäällä toisistaan ja toiseksi kuvakulma ei missään vaiheessa hyppää toisaalle. Käytännössä siis videokuvan on oltava yhtenäinen otos, joka ei sisällä leikkauksia, olivat ne sitten ajallisia tai kamerasta toiseen siirtyviä.

Edellä jo mainittiin, miten määrittämällä etualaksi tunnistettujen alueiden reunaviivat voidaan laskea alueen sijainti käyttämällä esimerkiksi reunaviivan sisältämän, mahdollisimman pienen suorakulmion keskipistettä. Sen sijaan seurattaessa kuvan yleistä liikettä, kuten optisen vuon tapauksessa, voidaan saada tuloksena liuta kuvapisteitä, jotka ovat liikkuneet. Nämä kuvapisteet saattavat ollaan kerääntyneet tiiviimmiksi joukoiksi, esimerkiksi kun jokin kohde liikkuu liikkumattomana pysyvän taustan editse. Tällaisessa tapauksessa on usein tärkeää pystyä mallintamaan tällaisen liikkuvan pistejoukon liikettä. Yksinkertaisimmillaan voitaisiin ottaa pisteistä keskiarvo ja käyttää sitä kohteen sijaintina, mutta tässäkin tapauksessa ongelmaksi jää erillisten pistejoukkojen rajaaminen toisistaan.

Pistejoukkojen rajaamisessa voidaan käyttää hyväksi klusterointialgoritmeja, jotka pyrkivät havaitsemaan paikallisia keskittymiä datassa. Esimerkki tällaisesta algoritmista on K-keskiarvot (engl. K-means), joka pyrkii etsimään datasta halutun määrän keskittymien keskipisteitä arpomalla keskittymien alustavat sijainnit, liittämällä datapisteet lähimpiin keskittymiinsä ja siirtämällä keskittymien sijainteja niihin liittyvien datapisteiden keskelle iteratiivisesti, kunnes keskittymät eivät enää siirry [3, s. 479].

Liikkuvan kuvan tapauksessa, kun keskittymien sijainnit on jo havaittu, voidaan käyttää hieman vastaavaan tapaan toimivaa keskimääräissiirtymäalgoritmia (engl. mean-shift) [21]. Se perustuu rajattuun, paikalliseen hakuikkunaan ja tilastolliseen las-

kentaan, joka pyrkii keskittämään hakuikkunan pistejoukon tiheimpään kohtaan [3, s. 337–341]. Hakuikkunan alkusijaintina voidaan käyttää edellisestä kuvasta havaittua sijaintia (kenties yhdistettynä pistejoukon havaitun nopeuden mukaiseen ennusteeseen uudesta sijainnista), jolloin algoritmi siirtää tämän hakuikkunan lähellä olevan pistejoukkokeskittymän tiheimpään kohtaan. Näin voidaan arvioida kyseisen pistejoukon liikettä tiheimpien kohtien sijainnin muutoksella. Erityisesti eletunnistukseen soveltuva tapa on myös liikemalleihin (engl. motion template) perustuva seuranta, jossa kohteen yhdistetyistä silueteista muodostettua liikehistoriakuvaa analysoimalla voidaan arvioida, millainen liike on tapahtunut [3, s. 342–343].

Tällä tavoin tai mallipohjaisella etsinnällä suoraan saadut arviot seurattavan kohteen sijainnista eivät luonnollisesti ole aina tarkkoja johtuen kameran rajallisesta tarkkuudesta, kohinasta, kohteiden hetkellisistä katoamisista tai monista muista syistä johtuen. Niinpä onkin kannattavaa pyrkiä mallintamaan kohteiden liikettä matemaattisesti vuorotellen arvioimalla, mihin kohde on liikkumassa, ja sitten korjaamalla arviota kuvadataan pohjautuvilla mittaustuloksilla. Arviointivaiheessa käytetään hyödyksi historiatietoja, kuten kohteen edellistä sijaintia, nopeutta ja jopa kiihtyvyyttä. Tällaisia arvioinnin ja mittauksen sykliin perustuvia algoritmeja kutsutaan estimaattoreiksi. Niistä esimerkkeinä toimivat laajalti käytetty Kalman-suodin (engl. Kalman filter) sekä tiivistymisalgoritmi (engl. condensation). [3, s. 348–349]

Signaalinkäsittelyssä paljon käytetty Kalman-suodin [32] pohjautuu ajatukseen, että tietyin oletuksin voidaan aiemmin muodostetun mallin ja sen epävarmuuden sekä uuden mittaustuloksen ja sen epävarmuuden perusteella muodostaa uusi malli niin, että se on mahdollisimman oikeassa. Kalman-suotimen asettamat oletukset ovat, että mallinnettava järjestelmä on lineaarinen, siinä esiintyvä kohina ei ole aikariippuvaista, eli se on valkoista kohinaa, ja kohina noudattaa normaalijakaumaa [68], eli sitä pystytään mallintamaan keskiarvon ja kovarianssin [66] perusteella. [3, s. 350] Matemaattisesti ilmaistuna suodin laskee lähteen [3] mukaisesti päivityskertoimen

$$K = \frac{\hat{\sigma}_1^{\,2}}{\hat{\sigma}_1^{\,2} + \sigma_2^2}, \tag{2.1}$$

missä $\hat{\sigma}_1^{\,2}$ on tehdyn ennustearvion epävarmuus ja σ_2^2 on uuden mittaustuloksen epävarmuus. Tämän päivityskertoimen avulla voidaan laskea uusi ennustearvio

$$\hat{x}_2 = \hat{x}_1 + K\left(x_2 - \hat{x}_1\right), \tag{2.2}$$

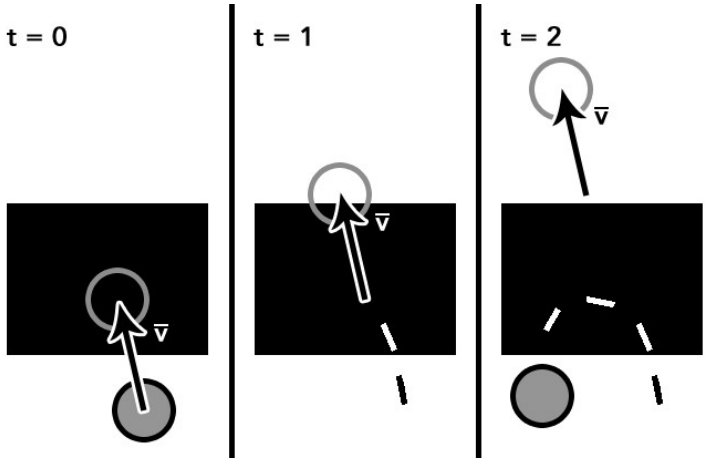
missä $\hat{x}_1$ on edellinen ennuste ja x_2 on uusi mittaustulos. Uuden ennusteen epävarmuus puolestaan saadaan kaavalla

$$\hat{\sigma}_2^{\,2} = \left(1 - K\right)\hat{\sigma}_1^{\,2}. \tag{2.3}$$

Alkuperäisenä ennusteena ja sen epävarmuutena käytetään luonnollisesti ensimmäistä mittaustulosta ja sen epävarmuutta. [3, s. 352–353]

Kun liikettä mallinnetaan tällä tavoin, esimerkiksi hetkellisesti näkyvistä katoava, vakionopeudella liikkuva kohde voidaan löytää uudelleen, kun se palaa näkyviin näkö-

esteen toiselta puolelta. Tämän toimiminen perustuu tietenkin siihen, että mittaustulosten epävarmuus on erittäin korkea, kun kohdetta ei löydetä, joten uuteen sijaintiarvioon käytetään pääasiassa ennustetta, joka vakionopeuden tapauksessa vastaa kohteen todellista liikettä. Tämän perusteella on helppoa huomata eräs Kalman-suotimen merkittävä puute: mitä tapahtuu, jos kappaleen nopeus, joko vauhti tai suunta tai molemmat, muuttuukin kohteen ollessa näkymättömissä? Silloin suotimen arviot eivät enää pidäkään paikkaansa ja kohde saatetaan kadottaa pysyvästi. Kuva 2.10 havainnollistaa suotimen toimintaa tällaisessa tilanteessa.

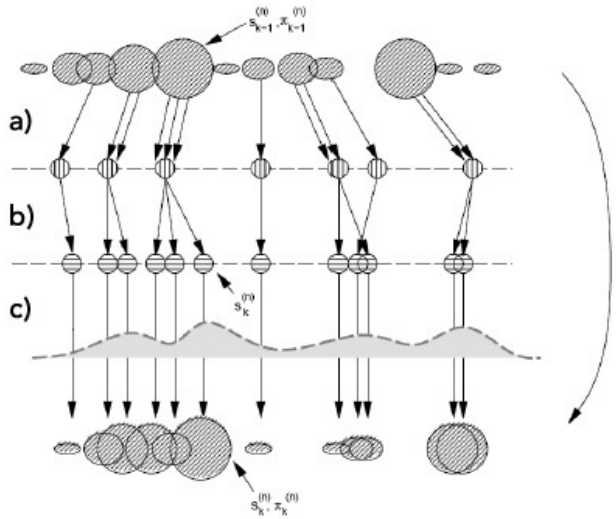


Kuva 2.10. *Kalman-suodin tekee virheellisen arvion, kun kohde muuttaa nopeuttaan näköesteen takana.*

Tätä Kalman-suotimien puutetta pyrkii paikkaamaan tiivistymisalgoritmi [27], joka on partikkelisuodin (engl. particle filter). Kun Kalman-suodin tuottaa yhden arvion kohteen uudelle sijainnille, tuottaa tiivistymisalgoritmi todennäköisyysjakauman tiheysfunktion (engl. probability density function): useita arvioita, joista kullekin määritetään todennäköisyys lähdemateriaaliin perustuvien havaintojen avulla. Näitä arvioita tai virallisemmin hypoteeseja, joista suodin pitää kirjaa, voidaan kutsua myös partikkeleiksi. [3, s. 364] Kuva 2.11 havainnollistaa algoritmin toimintaa, jota kuvaillaan seuraavaksi.

Algoritmin toiminta on sen kehittäneiden Isardin & Blaken [27] mukaan pääpiirteissään seuraavanlainen: Ensinnäkin seurattavia kohteita mallinnetaan käyrien, kuten kohteiden reunaviivojen, avulla. Itse algoritmi perustuu edellisestä kuvasta tuotetusta tiheysfunktiosta painoarvojen mukaan tehtävään näytteenottoon: mitä suurempi painoarvo partikkelilla on sitä todennäköisemmin ja useampaan kertaan se valitaan seuraavan iteraation (videon kuvan) näytejoukkoon, siis tiheysfunktioon. Algoritmin ennustavat vaiheet koostuvat kohteen dynamiikan (nykyisen tilan ja tilahistorian sekä seurattavien nykyisten kuvapiirteiden ja kuvapiirteiden historian) mukaan tapahtuvasta yhtenäisestä ajelehtimisesta (engl. drift) ja satunnaisesti suoritettavasta hajauttamisesta (engl. diffusion), jossa useaan kertaan valitut identtiset näytteetkin liikkuvat toisistaan poike-

ten tiheysfunktion partikkeliakselilla. Lopuksi käytetään kuvaan pohjautuvia havaintoja, joiden avulla määritetään uuden näytejoukon (jakauman) näytteiden painoarvot lopullisen todennäköisyysjakauman tuottamiseksi. Tämän jakauman näytteiden perusteella voidaan arvioida kohteen liikettä valitsemalla suurimmalla todennäköisyydellä (painoarvolla) paikkansa pitävän, kohteen uutta tilaa kuvaavan arvion (partikkelin). [27]



Kuva 2.11. *Tiivistymisalgoritmin todennäköisyyden tiheysfunktioihin perustuvan satunnaisnäytteistyksen vaiheet: a) ajelehtiminen, b) hajauttaminen ja c) havaintoperusteinen painottaminen. Muokattu [27].*

Näitä tai muita menetelmiä käyttäen saadaan lopulta mallinnettua seurattavan kohteen liike, eli saadaan liikkeenseurannan lopputulos. On kuitenkin selvää, että mikään edellä kuvatuista algoritmeista ei suoriudu tehtävästään olosuhteiden riittävästi huonontuessa, joten lähdemateriaalin laatuun on syytä panostaa, mikäli se vain on mahdollista. Seuraavassa, liikkeenseurantaosion päättävässä kohdassa käsitellään vielä lyhyesti asioita, joilla voi vaikuttaa suuresti liikkeenkaappauksen onnistumiseen.

2.3.4 Lähdemateriaalin optimointi

Jotta liikkeenseuranta onnistuisi mahdollisimman hyvin ja lopputuloksena saatava data olisi riittävän stabiilia, kannattaa kuvaushetkellä ja jo sitä ennen kiinnittää huomiota muutamiin asioihin. Keskeisin asioista on tietysti kuvauslaitteisto ja sen asetukset. Mikäli kuvattavassa kohteessa tapahtuu nopeita liikkeitä, kuten esimerkiksi suunliikkeet, on liikkeenseurannan kannalta tärkeää, että nopeasti liikkuvat merkkipisteet pysyvät kuvassa terävinä. Liike-epäterävyys (engl. motion blur) johtuu kameran valotusajasta (engl. exposure): mitä pitempi aika on sitä pidemmän matkan kohde ehtii liikkua yhden kuvan aikana, jolloin kuva suttaantuu liikkeen suunnassa. Toisaalta taas lyhyemmällä valotusajalla liike on terävämpää, mutta kameran kuvasensoriin pääsevän valon määrä vähenee, jolloin kuva pimenee. Vähenevää valon määrää voi tiettyyn pisteeseen asti

kompensoida avaamalla kameran aukkoa (engl. aperture), jolloin samassa ajassa pääsee enemmän valoa sensorille asti.

Aiemman prototyypin perusteella todettiin, että harrastelijakäyttöön suunnatulla videokameralla ja hyvällä valaisulla saadaan jo 1/100 sekunnin valotusajalla riittävän teräviä kuvia kasvonliikkeistä. Valitettavasti halvemmissa kameroissa, kuten älypuhelimien kameroissa, ei aina ole manuaalista valotusajan säätöä. Tällöin ainoa mahdollisuus on huijata laitteen automatiikka pienentämään valotusaikaa lisäämällä kuvattavan kohteen valaistusta. Hyvä ja halpa keino tähän on käyttää merkkipisteinä heijastavaa teippiä, joka valaistaan niin, että merkkipisteet hohtavat kameran näkökulmasta kirkkaina pakottaen kameran laskemaan valotusaikaansa. Tällöin myös merkkipisteet on erittäin helppo löytää muilta osin tummentuneesta kuvasta etsimällä kuvan kirkkaimmat pisteet.

Toinen laitteiston asettama rajoite tai mahdollisuus on kuvanopeuden (engl. frame rate) säätäminen: kuinka monta kuvaa tallennetaan sekunnissa. Mitä pienempi kuvanopeus on sitä pidemmän matkan merkkipisteet ehtivät liikkua kuvien välillä. Elokuvissa perinteinen kuvanopeus on 24 kuvaa sekunnissa, vaikkakin nykyään ollaan joissain 3D-tuotannoissa siirtymässä tuplattuun 48 kuvan sekuntinopeuteen liikkeen sulavoittamiseksi [12]. Vastaavasti televisiokuva Euroopassa ja muilla PAL-alueilla on 25 kuvaa sekunnissa, mutta kuvamäärään on usein sisällytetty lomitettuna 50 kuvaa sekunnissa [47]. Kasvonliikkeiden tapauksessa kelvollisia tuloksia saadaan jo 25 kuvan sekuntinopeudella ja interpoloimalla kuvien välisiä merkkipisteiden sijainteja, jotta animaatio olisi sulavaa jopa 60 kuvaa sekunnissa pyörivässä pelikuvassa. Suurempi kuvanopeus kuitenkin varmistaa, että äkkinäisetkin liikahdukset tallentuvat eivätkä jää kuvien väliin. Käytettävää kameraa valittaessa kannattaakin selvittää, onko kuvanopeuden muuttaminen ylipäänsä mahdollista.

Liikkeenseurannan onnistumista voi myös parantaa minimoimalla merkkipisteiden liikkeitä ja pyrkimällä varmistamaan, että ne eivät katoa kuvasta kesken otoksen. Kasvonliikkeitä taltioidessa tämä on suhteellisen helppoa toteuttaa toisin kuin esimerkiksi kehonliikkeitä kaapattaessa. Ratkaisuksi riittää esimerkiksi teline, joka pitää kameran aina suunnattuna kasvoja kohti liikuttelipa näyttelijä päätään miten tahansa. Näin varmistetaan merkkipisteiden näkyvyys, mutta asetetaan myös rajoitteita käytettävällä kameralaitteistolle: pieni koko ja keveys ovat valttia, jos näyttelijä joutuu kannattelemaan varren päässä olevaa kameraa niskoillaan.

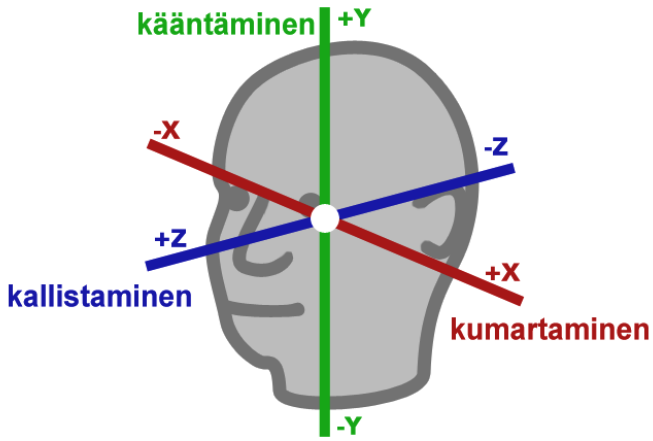
2.4 Päänliikkeiden arviointi

Nykyaikaisissa älypuhelimissa on monenlaisia sensoreita, joilla voi arvioida laitteen fyysistä asentoa tai liikettä. Lähestulkoon jokaisessa älypuhelimessa on kiihtyvään liikkeeseen, mukaan lukien painovoimaan, reagoiva kiihtyvyyssensori, jonka tavallisin käyttötapaus on arvioida, pitääkö käyttäjä puhelinta kädessään näyttö pystysuorassa vai vaakasuorassa, ja kääntää ruudulla näkyvä käyttöliittymä asentoon sopivaksi. Harvinai-

sempi, pääasiassa kalliimmista malleista löytyvä gyroskooppi puolestaan mittaa muutoksia laitteen asennossa eli sitä, kuinka laite kääntyy X-, Y- ja Z-akseliensa ympäri. Yleisyydeltään jossain näiden kahden välimaastossa oleva kompassi taas reagoi magneettikenttiin ja osoittaa maapallon magneettisen pohjoisnavan suunnan, mikäli muita magneettikenttiä ei ole aiheuttamassa häiriöitä. Kun älypuhelin liitetään liikkumaan näyttelijän pään mukana, voidaan näillä sensoreilla ja niiden yhteistoiminnalla, niin kutsutulla sensorifuusiolla, kaapata päänliikkeitä käytettäväksi 3D-mallin animointiin. Seuraavissa kohdissa tarkastellaan tarkemmin, miten tämä onnistuu, ja miten puuttuvia sensoreita voisi korvata optisen liikkeenseurannan avulla.

2.4.1 Päänliikkeet

Päänliikkeet voidaan ilmaista esimerkiksi kuvan 2.12 mukaisen koordinaatiston mukaisiin kiertymisiin paikallisten X-, Y- ja Z-akseleiden ympäri. Tässä työssä käytetään seuraavia nimityksiä erillisille kiertymille: pään kääntäminen tarkoittaa Y-akselin ympäri kiertymistä (engl. yaw), pään kallistaminen tarkoittaa Z-akselin ympäri kiertymistä (engl. roll) ja kumartaminen tarkoittaa X-akselin ympäri kiertymistä (engl. pitch).

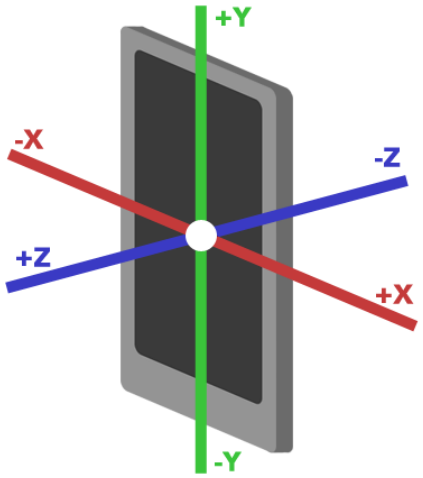


Kuva 2.12. *Pään kiertyminen eri akseleiden ympäri.*

Jotta 3D-hahmon päätä pystyttäisiin liikuttamaan oikein, täytyy älypuhelimen sensoreilla havaittu liike muuttaa kääntämisen, kallistamisen ja kumartamisen astelukemiksi. Aloittaen oletusasennosta, eli pään osoittaessa suoraan eteenpäin, nämä kiertymiset asetetaan hahmon päähän tietyssä järjestyksessä käyttäen koordinaatistona pään paikallista koordinaatistoa, jonka akselit kiertyvät jokaisen kierron mukana. Ensimmäisenä käännetään päätä (kierto Y-akselin ympäri), sitten kumarretaan (kierto X-akselin ympäri) ja lopulta kallistetaan (kierto Z-akselin ympäri). Näin saadaan pää haluttuun asentoon. Suurempi ongelma on kuitenkin selvittää, mikä haluttu asento on; siis miten esittää tosielämän mittaustulokset kääntämisen, kallistamisen ja kumartamisen komponentteina.

2.4.2 Asennon arviointi kiihtyvyyssensorilla

Älypuhelimissa yleisesti käytetyt kiihtyvyyssensorit mittaavat kiihtyvyyttä kolmen akselin suunnassa. Yleisimmillä älypuhelinalustoilla, iOS:llä, Androidilla ja Windows Phonella, nämä laitteen paikalliset akselit ovat kuvan 2.13 mukaiset [16; 54; 55]. Koska kiihtyvyyssensorit havaitsevat sekä lineaarisen liikkeen että painovoiman [48], esimerkiksi tasaiselle pöydälle, näyttö ylöspäin asetetun puhelimen kiihtyvyyssensori näyttäisi vektoriesityksenä lukemaa $[0,0,-g]$, missä g on gravitaatiokiihtyvyys; painovoima siis vetää kiihtyvyyssensorin lukemaa negatiivisen z-akselin suuntaan. Vastaavasti painottomassa tilassa tai vapaapudotuksessa lukema olisi nolla jokaisella akselilla, joten käytännössä edellisessä esimerkissä sensori mittasikin pöydän tukivoiman aiheuttaman, painovoiman vastaisen kiihtyvyyden. Kun tiedetään painovoiman suunta, voidaan määrittää, missä asennossa laite parhaillaan on. Tarkempi selvitys kiihtyvyyssensorin rakenteesta ja siitä, miten se havaitsee kiihtyvyyden ja painovoiman löytyy esimerkiksi lähteestä [48, s. 3].



Kuva 2.13. *Älypuhelimen paikalliset akselit.*

Pedleyn [48] johtamien laskukaavojen avulla voidaan kiihtyvyyssensorin mittaustuloksen perusteella ratkaista laitteen kallistus- ja kumarruskulmat, kun sensorin lineaarinen kiihtyvyys on nolla, eli mittauksessa esiintyy ainoastaan painovoiman vaikutus. Kääntymiskulman määrittäminen ei kiihtyvyyssensorilla onnistu. Tämä johtuu siitä, että koska mittaustuloksena saadun vektorin pituus on aina gravitaatiokiihtyvyyden suuruinen, voi vektorin pää olla missä tahansa kohtaa pallopintaa, jonka säde on gravitaatiokiihtyvyyden suuruinen. [48, s. 9] Vektori voidaan siis ilmaista pallokoordinaatistossa säteen ja kahden astelukeman avulla [69]. Niinpä kiihtyvyyssensorilla ei saada kolmea erillistä astelukemaa eikä siten pystytä määrittämään painovoima-akselin ympäri tapahtuvaa kiertymistä.

Toinen huomion arvoinen asia on, että koska eri akseleiden kiertämisjärjestyksellä on väliä, voidaan kallistus ja kumarrus määrittää jommassakummassa järjestyksessä. Kuten aiemmin mainittiin, tähän työhön valittu järjestys on kääntäminen, kumartaminen ja lopuksi kallistaminen. Niinpä lähteen [48] mukaisesti saadaan mittaustuloksena saadusta vektorista $\bar{G}_p=[G_{px}, G_{py}, G_{pz}]$ kallistuskulma laskemalla

$$\theta_{xyz}=atan\left(\frac{-G_{px}}{\sqrt{G_{py}{}^2+G_{pz}{}^2}}\right) \tag{2.4}$$

ja kumarruskulma laskemalla

$$\phi_{xyz}=atan2\left(G_{py}, G_{pz}\right), \tag{2.5}$$

missä *atan2*-funktio laskee arkustangentin

$$atan\left(\frac{G_{py}}{G_{pz}}\right) \tag{2.6}$$

huomioiden oikean yksikköympyrän neljänneksen parametrien merkkien perusteella. Toisin sanoen tavallisen arkustangentin antaessa tuloksensa radiaanivälillä $\left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$ antaakin *atan2* tuloksen radiaanivälillä $(-\pi, \pi]$ kattaen koko yksikköympyrän. [48]

Laskukaavoista on tärkeää huomata, että kaavassa 2.5 mittaustuloksen Y- ja Z-komponenttien ollessa nolla ei *atan2*-funktion sisältämän jakolaskun (2.6) tulosta ole määritelty. Tällaisessa tilanteessa mittaustuloksessa esiintyvä kohina on merkittävin elementti sekä jakolaskun nimittäjässä eikä jakajassa, jolloin kumarruskulman arvosta tulee epästabiili. Käytännössä ongelma johtuu siitä, että laitteen ollessa kyljellään kumartamisakseli on samansuuntainen kuin painovoima-akseli, jolloin kiihtyvyyssensori ei pysty havaitsemaan kumarrusliikettä. [48]

Ongelmaa voi korjata käyttämällä kumarruskulman laskemisessa mittaustuloksen X-akselin lukemaa seuraavasti:

$$\phi_{xyz}=atan\left(\frac{G_{py}}{\sqrt{G_{pz}{}^2+\mu G_{px}{}^2}}\right), \tag{2.7}$$

missä μ on jokin pienikokoinen kerroin, kuten 0,1 tai pienempi. Lähestyessä edellä mainittua ongelmatilannetta arvo G_{px} lähestyy gravitaatiokiihtyvyyden arvoa ja mittaustuloksen muut komponentit lähestyvät nollaa, jolloin kaavan tuloskin lähestyy nollaa. Nimittäjä ei siis enää X-arvon sisällyttämisen myötä voi olla nolla samaan aikaan kuin osoittajakin. Näin saatu tulos on kuitenkin vain arvio, jonka virheen saa määritettyä kaavalla:

$$\Delta\phi_{xyz}=atan\left(\frac{G_{py}\left(G_{pz}-\sqrt{G_{pz}{}^2+\mu G_{px}{}^2}\right)}{G_{py}{}^2+G_{pz}\sqrt{G_{pz}{}^2+\mu G_{px}{}^2}}\right). \tag{2.8}$$

Käytännössä laitteen liikkeet aiheuttavat kuitenkin vielä lisää virhettä näihin tuloksiin. [48]

Koska tässä työssä kiihtyvyyssensori sijaitsee kypärätelineen varren mittaisella etäisyydellä kiertymisliikkeiden keskipisteestä, voitaisiin ajatella, että kääntymisliikkeen arviointi onnistuisi tarkkailemalla laitteen kiihtyvyyttä X-akselin suunnassa. Tässä on muutamia merkittäviä ongelmia. Ensinnäkin esimerkiksi päätä kumarrettaessa X-akselin havaitsemaan kiihtyvyyteen alkaa vaikuttaa painovoima, joten se pitäisi jotenkin pystyä suodattamaan mittaustuloksista. Toinen ongelma on, että kääntymismäärän arvioimiseksi kiihtyvyyslukemalle joudutaan suorittamaan tuplaintegraatio: ensimmäisellä integraatiolla saadaan kiihtyvyyslukemasta nopeuslukema ja toisella integraatiolla sijaintilukema, joka voitaisiin muuntaa jollain tavoin kääntymislukemaksi. Tuplaintegraatio kasvattaa huomattavasti mittaustuloksen sisältämän kohinan vaikutusta ja aiheuttaa lopputuloksessa ajelehtimista (engl. drift), kun paikaltaan alkaneen ja paikalleen pysähtyvän liikkeen kiihtyvyyden integraali ei tuotakaan nollaa (nopeuden kasvua on tapahtunut yhtä paljon kuin nopeuden vähenemistäkin) kyseisellä aikavälillä [39]. Tällaista mittausakselin toiseen suuntaan painottuvaa virhettä (engl. bias) voitaisiin arvioida mittaamalla pitkän aikavälin keskiarvot kullekin akselille sekä positiiviseen että negatiiviseen suuntaan laitteen ollessa levossa [2].

2.4.3 Pyörimisliikkeen mittaaminen gyroskoopilla

Gyroskooppi mittaa laitteen kiertymisen kulmanopeuksia vastaavilta akseleilta kuin kiihtyvyyssensorikin (kuva 2.13). Ajelehtiminen on tässäkin ongelmana, vaikkakin kiihtyvyyssensorin tapausta vähemmässä määrin, sillä kiertymismäärän selvittämiseksi joudutaan ottamaan kulmanopeuksista ainoastaan yksittäinen integraali. Toinen ajelehtimista lisäävä ongelma on kalibrointivirhe, joka ilmenee mittauseroina kiertymisen eri nopeuksien ja kestojen välillä. [2] Gyroskooppi kuitenkin mahdollistaa myös kääntymisen mittaamisen kallistamisen ja kumartamisen lisäksi. Valitettavasti työn toteutusosuudessa käytetyssä testilaitteessa ei gyroskooppia ole, joten gyroskoopin käyttö esitellään tässä vain hyvin pintapuolisesti.

Gyroskoopin selkeä etu kääntymisen mittaamisen lisäksi on sen tarkkuus verrattuna kiihtyvyyssensorilla saatuihin kallistamisen ja kumartamisen määriin. Se reagoi nopeasti kiertymiseen reagoimatta kuitenkaan lineaariseen liikkeeseen tai gravitaatioon [14]. Integroinnin aiheuttaman ajelehtimisen vaikutuksia on kuitenkin vähennettävä jotenkin. Tässä kuviin astuu useamman sensorin tuottaman tiedon yhdistäminen (engl. sensor fusion), kuten esimerkiksi Ayubin et al. [2] kuvaamassa tavassa yhdistää matkapuhelimen sensoreita laitteen asennon määrittämiseksi. Kiihtyvyyssensorilla määritetty kallistus ja kumarrus eivät ajelehdi, joten käyttämällä näitä epätarkempia arvoja yhdessä gyroskoopin integraalin kanssa, voidaan ajelehtimista eliminoida hyödyntämällä gyroskoopin lukemia lyhyellä aikavälillä ja kiihtyvyyssensorin lukemia pitkällä aikavälillä. [2] Tällä tavoin järjestelmä reagoi liikkeisiin nopeasti, mutta ei kuitenkaan ala ajelehtia ajan kuluessa. Tämä tietofuusio voidaan käytännössä toteuttaa vaikkapa edellä mainitun

Kalman-suotimen [13] tai ali- ja ylipäästösuotimen yhdistelmän [15] avulla. Kääntymiskulman ajelehtimista ei luonnollisesti pystytä kiihtyvyyssensorin avulla korjaamaan.

2.4.4 Suunnan määrittäminen kompassilla

Gyroskoopin kääntymiskulman ajelehtimisen eliminoinnissa osoittautuu hyödylliseksi elektroninen kompassi. Kompassi muodostuu käytännössä kiihtyvyyssensorin ja magnetometrin yhteystyöstä. Magnetometri mittaa magneettikenttää (magneettivuon tiheyttä) mikrotesloina paikallisessa koordinaatistossa kolmen akselin suunnalta (kuva 2.13). Kiihtyvyyssensorilla arvioidun laitteen asennon perusteella saadaan määritettyä magneettivuon tiheys globaaleissa koordinaateissa. Näin kompassilla on mahdollista selvittää laitteen absoluuttinen kääntymiskulma mittaamalla maapallon magneettisen pohjoisnavan suunta. Yhdistämällä tämä tieto gyroskoopilla mitattuun kääntymiskulmaan saadaan ajelehtiminen tämänkin akselin suhteen kuriin. Kompassilla on kuitenkin yksi erittäin selkeä puute: magneettikentän vääristymät aiheuttavat mittausvirheitä. [2] Lisäksi kompassi reagoi kääntymiseen erittäin hitaasti.

Kuten huomataan, kullakin sensorilla on omat etunsa ja puutteensa eikä yksikään niistä itsellään tuota erityisen hyvää tulosta pitemmällä aikavälillä tarkasteltuna joko ajelehtimisen, muiden epätarkkuuksien tai hitauden osalta. Yhdessä käytettynä ne kuitenkin pystyvät kumoamaan toistensa puutteita, jolloin saadaan paras mahdollinen arvio laitteen asennolle. Eräs kuvaus näiden kolmen sensorin mittaustulosten fuusiosta älypuhelinkäytössä löytyy lähteestä [2].

2.4.5 Optiset menetelmät

Jos käytettävissä ei ole kiihtyvyyssensoria, gyroskooppia ja kompassia, mutta tämän työn tavoin käytössä on videota tallentava kamera, voidaan harkita optista liikkeenseurantaa täydentämään sensorien puutteita. Tässä työssä gyroskooppia ei pystytty käyttämään, joten muilla sensoreilla epätarkimmaksi jää pään kääntymisen määrittäminen, jota voidaan arvioida ainoastaan hitaan kompassin avulla. Seuraavaksi esitetään ajatuksia, joilla pään kääntymistä voisi pystyä mittaamaan.

Ensinnäkin on tehtävä oletus, että kameran näkemä kuva on rajattu niin, että kasvojen lisäksi näkyy sivuilla myös muuta taustaa. Tämän taustan liikettä voisi seurata esimerkiksi määrittämällä niiden alueiden optinen vuo, joilla tausta näkyy. Tässä ongelmaksi voi muodostua liian yksivärinen tausta, kuten valkoinen seinä, jonka liikettä on vaikea havaita. Optisesta vuosta havaitusta taustan vaakasuuntaisen liikkeen nopeudesta, esimerkiksi liikkuneiden pisteiden keskimääräisestä nopeudesta, voitaisiin sitten tehdä arvio siitä, mikä vastaava pään kääntymisnopeus on.

Tässäkin menetelmässä vaaditaan integrointi nopeudesta lopulliseen kääntymiskulmaan, mikä voi aiheuttaa ajelehtimista. Tätä puutetta voidaan gyroskooppien tavoin yrittää paikata kompassin kertoman absoluuttisen suunnan avulla. On kuitenkin huomioitava, että niin gyroskoopin kuin liikkeenseurannankin perusteella saadaan aina

näyttelijän pään koordinaatiston mukainen kääntyminen, kun taas kompassi osoittaa kääntymissuunnan globaalin koordinaatiston mukaan. Menetelmän hyötynä sen sijaan on, että kääntymisnopeudesta saadaan viiveetön arvio, eli videokuvien välisenä aikana tapahtunut liike tiedetään heti seuraavan kuvan perusteella, toisin kuin esimerkiksi hidasta kompassia käytettäessä. Mittaustuloksen parantamiseksi voidaan vielä valmistella kuvaustila niin, että taustana on jonkinlaista kuviointia, jonka optinen vuo on helppo määrittää. Schall et al. esittävät lähteessä [52] toisenlaisen, näkyvää ympäristöä kartoittavan menetelmän, jolla kameran avulla pyritään saamaan selville absoluuttinen kääntyminen ja korjaamaan sillä muiden sensorien ajelehtimista.

2.4.6 Suodatus

Erilaisten sensorien mittaustulosten ajelehtimisen vähentämiseksi ja muutenkin mittaustulosten stabilisoimiseksi satunnaisten häiriöiden ja kohinan varalta on mittaustuloksia suodatettava. Yleisin tapa tähän on käyttää yksinkertaista alipäästösuodinta. Alipäästösuodin päästää nimensä mukaisesti lävitseen signaalin matalat taajuudet, eli käytännössä hitaat liikkeet, ja karsii suurempia taajuuksia, kuten sensorin kohinasta aiheutuvaa mittaustuloksen heittelehtimistä. Ohjelmallisesti tällainen suodin voidaan toteuttaa esimerkiksi ottamalla signaalista liukuva keskiarvo (engl. moving average), eli määritellyllä aikavälillä sijaitsevien mittausten keskiarvo [2]. Koska alipäästösuodin tasoittaa signaalin nopeita muutoksia, se myös viivästyttää suurien ja äkkinäisten muutosten havaitsemista. Liukuvan keskiarvon tapauksessa ja tunnettaessa kaikki mittaustulokset etukäteen voidaan viivettä vähentää laskemalla keskiarvo käyttämällä sekä tulevia että menneitä mittaustuloksia nykyisen mittaustuloksen suodattamiseksi.

Päinvastoin toimiva ylipäästösuodin suodattaa hitaita muutoksia ja säilyttää nopeat. Tästä on hyötyä esimerkiksi hiljalleen kasautuvan ajelehtimisen suodattamisessa. [2] Yksinkertaisimmillaan ylipäästösuotimen toteuttaminen onnistuu vähentämällä alkuperäisestä signaalista siitä alipäästösuodatettu signaali. Eikä sovi unohtaa Kalmansuotimia, jotka ovat erittäin hyödyllisiä useiden signaalien yhdistämiseen, kuten edelläkin jo mainittiin.

2.5 3D-mallin kasvoanimaatio

Liikkeenkaappauksen tarkoituksena on nopeuttaa animointiprosessia korvaamalla ainakin osittain työläs avainkuviin perustuva animaatio. Avainkuva-animaatiossa animaattori luo 3D-mallin liikkeen määrittämällä aikajanalle avainkuvia, joissa malli on animaattorin määräämässä asennossa. Avainkuvien väliset kuvat saadaan interpoloimalla avainkuvien asentoja, jolloin muodostetaan ohjelmallisesti kaikki avainkuvien väliin jäävät kuvat. Tätä välikuvien täydentämistä kutsutaan myös tweenaamiseksi (engl. tweening, inbetweening). Käytännössä interpolointi, tai tweenaus, suoritetaan 3D-mallin vertekseille. 3D-mallit koostuvat kolmioverkoista (engl. mesh), jotka puolestaan koostu-

vat vertekseistä (tai solmuista) ja niitä kolmioiksi yhdistävistä kaarista (engl. edge). Seuraavaksi käsitellään menetelmiä, joilla voidaan reaaliaikaisesti muunnella 3D-mallin ulkonäköä animaation tuottamiseksi.

2.5.1 Luuanimaatio

Koska 3D-hahmon mallin verteksien animoiminen käsin olisi käsittämättömän työlästä, määritellään hahmolle yleensä luurankorakenne, joka huolehtii varsinaisen mallin verteksien liikuttamisesta. Samaa luurankoa voi myös käyttää useammalle hahmolle, jolloin yksi animaatio toimii niillä kaikilla. Mikäli luurangot ovat erilaisia, on animaatio kuitenkin ehkä mahdollista kohdistaa uudelleen (engl. retarget) eri luurangollekin [58].

Luurangossa voi esiintyä hierarkiaa samoin kuin elävien olentojenkin luurangoissa. Esimerkiksi reisiluun liikuttaminen liikuttaa samalla myös sääriluuta ja muita jalan luita, koska ne ovat kiinnittyneinä toisiinsa. Hierarkiassa alemman luun, kuten varvasluun, liikuttaminen ei luonnollisesti liikuta hierarkian ylempiä luita, kuten sääriluuta.

Kasvojen luurankomalli harvemmin rajoittuu luiden mallintamiseen muuten kuin matalapolygonisissa tai kasvoanimaatiota käyttämättömissä tarkoituksissa. Käytännössä leukaluu on ainoa varsinainen luu kasvoanimaatiossa, muut kasvojen liikkeet ovat pääasiassa lihasten liikuttamaa ihoa. Luurankomallin kannalta tällä ei ole merkitystä, sillä lopputulos on sama: mallin luiden liikuttaminen liikuttaa hahmomallin verteksejä. Se, mitä verteksejä mikäkin luu liikuttaa on luurankomallin määrittäjän päätettävissä.

Luurankomallin pinnoittamisessa (engl. skinning) kuhunkin luuhun liitetään luettelo vertekseistä, joita luun liike liikuttaa, sekä kunkin verteksin painokerroin, joka vaikuttaa siihen, kuinka paljon verteksi seuraa luun liikettä. Esimerkiksi painokertoimella 0,5 verteksi liikkuu puolet siitä mitä luu. Yksittäinen verteksi voi olla kerrallaan useamman luun ohjattavana, jolloin liike määräytyy eri luiden liikkeiden ja niihin määriteltyjen eri painokertoimien kesken. Reaaliaikaisissa animaatiojärjestelmissä, kuten pelimoottoreissa, saattaa olla rajoitettu sitä, kuinka moni luu voi vaikuttaa yhteen verteksiin [11].

Luurangon ohjaaminen liikkeenseurannalla havaittujen kasvonliikkeiden ja sensoreilla mitattujen päänliikkeiden avulla on suhteellisen suoraviivaista. Esimerkiksi pään kääntyminen onnistuu yksinkertaisesti kiertämällä niska- ja kalloluita haluttuun suuntaan. Sen sijaan liikkeenseurannalla arvioitu leuan liike on muunnettava leuan merkkipisteen liikkeestä leukaluun kiertämismääräksi. Muut kasvojen merkkipisteet pystyy tätä helpommin liittämään lähes suoraan kasvoluiden paikallisessa koordinaatistossa tapahtuvaksi kaksiulotteiseksi liikkeeksi.

2.5.2 Muodonmuutos

Muodonmuutokseen (engl. morphing) perustuvassa animaatiossa luodaan samalle mallille useita kohdemuotoja. Nämä muodot ovat erilaisia kasvonilmeiden ääriasentoja. Näiden muotojen välillä interpoloimalla ja eri ilmeitä yhdistelemällä eri suhteissa saadaan aikaan kasvoanimaatiota. [49] Liikkeenkaappausdatan käyttäminen tällaiseen ani-

maatiojärjestelmään ei ole erityisen luonnollista. Periaatteessa sen toteuttamiseksi pitäisi kaapatusta liikkeistä tunnistaa, mitä ilmeitä (muotoja) ja missä suhteissa kuvassa on parhaillaan näkyvillä, kuten pelissä Far Cry 3 [58]. Tällöin animaatio myös rajoittuu näihin määriteltyihin ilmeisiin, mikä voi saada lopputuloksen näyttämään epäluonnolliselta ja tönköltä. Tässä työssä on päätetty käyttää muodonmuutoksen sijaan luuanimaatiota, koska se mahdollistaa suoraviivaisemman muunnoksen liikkeenkaappauksen ja lopullisen animaatiodatan välille.

2.5.3 Tekstuurin animointi

Animaation ei välttämättä tarvitse rajoittua pelkkään 3D-mallin muuntelemiseen. Esimerkiksi vuonna 1998 julkaistussa Grim Fandangossa avitettiin matalapolygonisten hahmojen ilmeikkyyttä animoimalla luurankohahmojen kasvoja tekstuurien avulla. Tekstuurianimaatiossa yksinkertaisuussaan vaihdetaan näkyvillä olevaa tekstuuria tai tekstuurin osaa. Tämä on käytännössä täsmälleen samanlaista kuin 2D-grafiikan bittikarttakuviin pohjautuva animaatio, paitsi että kuva esitetään 3D-mallin pinnalla.

Tuoreempi esimerkki tekstuurianimaatiosta on edelläkin mainittu L.A. Noire. Siinä tekstuurit kaapattiin suoraan näyttelijän kasvoilta ja niitä toistettiin 3D-mallin pinnalla kuin videota, 30 kuvaa sekunnissa [33]. Näin saatiin aikaan hyvin luonnollinen lopputulos, jossa näkyi kaikki liike ryppyjä myöten. Suurimmaksi ongelmaksi voisi sanoa sen, että tilansäästön ja pelikonsolien vähäisen muistin puitteissa tekstuurivirta, mukaan lukien 3D-mallin animaatio, jouduttiin pakkaamaan 30–100 kilotavuun sekunnissa [33], mikä rajoittaa tekstuurien laatua ja vaikuttavuutta jonkin verran.

3 TOTEUTUS

Tämä luku keskittyy työn teknisenä osuutena suunniteltuun ja toteutettuun prototyyppijärjestelmään. Luvun aluksi esitellään järjestelmälle asetettuja tavoitteita (kohta 3.1) ja järjestelmän osat vastuualueineen (kohta 3.2). Seuraavaksi tarkastellaan kunkin järjestelmän osan teknisempää toteutusta aloittaen älypuhelimella suoritettavasta kaappaussovelluksesta (kohta 3.3) ja jatkamalla kaappausdatan käsittelystä vastaaviin liikkeenseurantasovellukseen (kohta 3.4) ja käsittelysovellukseen (kohta 3.5) sekä lopulta itse animaation esittävään animaatiokirjastoon (kohta 3.6).

3.1 Tavoitteet

Järjestelmän päätavoite on tarjota indie-pelikehittäjille käyttökelpoinen ratkaisu riittävän luonnollisen kasvoanimaation toteuttamiseen. Indie-pelikehittäjä voidaan tässä tapauksessa määritellä kehittäjäksi, jolla on rajalliset henkilöstö- ja aikaresurssit eikä varaa tai halua sijoittaa kalliisiin erikoislaitteisiin ja -järjestelmiin animaation toteuttamiseksi. Tämän perusteella voidaan johtaa täsmällisempiä tavoitteita, joiden toteutuessa myös päätavoite toteutuu.

Henkilöstö- ja aikaresurssien rajallisuus tarkoittaa, että mitä automatisoidumpi ja tehokkaampi järjestelmä on sitä parempi. Automaattisuuden toteutumiseksi vaaditaan järjestelmältä luotettavuutta, jotta käyttäjän aikaa ei kuluisi järjestelmän tekemien virheiden korjaamiseen. Ideaalisessa tapauksessa näyttelijän suoritus taltioidaan ja järjestelmä tuottaa sen perusteella automaattisesti 3D-mallille animaation. Käytännössä tämä on mahdotonta, sillä käyttäjän toimia vaaditaan ainakin seuraaviin tehtäviin: Kuvattu materiaali on ensin leikattava, eli on valittava paras näyttelysuoritus ja rajattava sen alku- ja loppupisteet sopivasti. Lisäksi on pystyttävä vaikuttamaan siihen, miten animaatio vaikuttaa 3D-malliin. Voidaan esimerkiksi haluta rajoittaa suun aukeamismäärää eri verran eri malleilla tai vaikkapa skaalata kulmakarvojen liikettä hienovaraisemmaksi. Tällaiset säädöt olisivat siis pääasiassa 3D-mallikohtaisia, mutta mikään ei tietysti estä niiden käyttämistä yksittäistenkin animaatioleikkeiden hienosäätöön tarpeen vaatiessa. Käyttäjän toimia vaaditaan myös järjestelmän kalibrointiin, jotta eri kuvauskerroilla taltioidusta materiaalista saadaan mahdollisimman yhtenäistä animaatiodataa. Saman 3D-mallin animaatioita voidaan nimittäin joutua kuvaamaan useana eri kertana, jolloin kamera saattaa olla hieman eri kohdassa tai suunnassa kuvauskerrasta riippuen. Voidaan siis vaatia, että yllä mainittujen tehtävien ollessa suorittuina järjestelmän on pystyttävä kä-

sittelemään samalla kalibroinnilla toimivat ja samalle 3D-mallille tarkoitetut materiaalileikkeet ilman käyttäjän toimia tai valvontaa.

Erikoislaitteiden tarve puolestaan voidaan ohittaa toteuttamalla järjestelmä niin, että se perustuu nykyaikaisen älypuhelimen käyttöön videokuvan ja päänliikedatan tallentajana. Nykypuhelimet pystyvät jo tallentamaan hyvälaatuista videokuvaa tähän tarkoitukseen riittävällä resoluutiolla ja kuvanopeudella. Lisäksi niissä on usein monenlaisia sensoreita, kuten kiihtyvyyssensori, joita voidaan käyttää laitteen fyysisen asennon määrittämiseen. Kun älypuhelin kiinnitetään niin, että se liikkuu näyttelijän pään mukana, kiihtyvyyssensori kertoo näyttelijän pään asennon.

Vaihtoehtona älypuhelimelle voisi olla erityisesti käyttötarkoitukseen suunniteltu laite, joka sisältää pienen teräväpiirtovideokameran ja kaikki tarpeelliset sensorit. Tällaisella erikoislaitteella järjestelmästä saisi varmasti merkittävästi paremman, koska suunnittelussa voisi vaikuttaa myös laitteiston ominaisuuksiin. Voitaneen kuitenkin olettaa, että lähes kenelle tahansa kynnys ostaa laadukas ja yleiskäyttöinen älypuhelin on huomattavasti pienempi kuin pelkästään tässä järjestelmässä käytettävän, vaikkakin todennäköisesti älypuhelinta jonkin verran halvemman, erikoislaitteen hankkiminen. On myös todennäköistä, että kehittäjä omistaa jo tarkoitukseen soveltuvan älypuhelimen.

Lopputuloksena saatavan animaation ”riittävää luonnollisuutta” on hankala määritellä tai arvioida objektiivisesti muuten kuin ehkä vertaamalla lopputulosta lähdevideoon. Niinpä luonnollisuuden tulee täyttää seuraavat subjektiiviset vaatimukset:

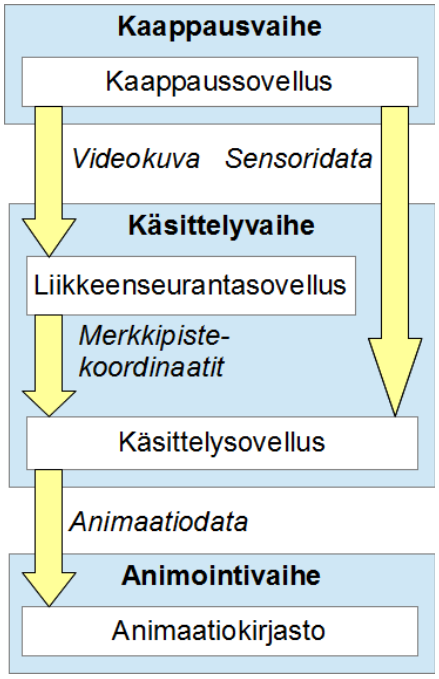
- Liikkeet eivät saa näyttää konemaisen töksähtäviltä, mutta eivät myöskään liian sulavilta. Lähdemateriaalista saatavan datan kohinasta aiheutuvat epätarkkuudet on myös pyrittävä eliminoimaan.
- Kasvot ja pää eivät saa vääntyä epäinhimillisiin asentoihin.
- Animaation, etenkin huultenliikkeiden, on pysyttävä ajassa ääniraidan kanssa riippumatta leikkeen pituudesta tai pelilaitteen prosessorin kuormituksesta.

Yksinkertaistettuna järjestelmän tulee siis tuottaa mahdollisimman luonnollinen lopputulos mahdollisimman automaattisesti, tehokkaasti ja luotettavasti, sillä pienillä pelikehittäjillä harvemmin on resursseja käsitellä suurta animaatiomäärää ja vielä harvemmin hienosäätää jokaista animaatiota käsin mieleisekseen. Järjestelmän on tarkoitus mahdollistaa laadukkaamman kasvoanimaation hyödyntäminen indie-peleissä vaatimatta kuitenkaan kohtuuttoman suuria työmääriä järjestelmän käyttämiseksi.

3.2 Järjestelmän osat ja niiden vastuualueet

Järjestelmä jakautuu kolmeen toiminnallisesti erilliseen osaan. Eri osat kuvastavat samalla järjestelmän toimintoketjun eri vaiheita: kaappausvaihe, jossa kerätään animaation pohjana toimiva lähdemateriaali; käsittelyvaihe, jossa lähdemateriaalista tuotetaan animoinnissa käyttökelpoista dataa; ja animointivaihe, jossa 3D-mallia animoidaan

edellisestä vaiheesta saadun animointidatan perusteella. Kuva 3.1 havainnollistaa järjestelmän osia sekä niiden käyttämiä syötteitä ja tuottamia tuloksia.



Kuva 3.1. *Järjestelmän rakenne.*

Kuvan mukaisesti järjestelmän kokonaisrakenne on liukuhihnamainen. Ajallisesti vaiheet eivät kuitenkaan käytännössä seuraa toisiaan yhtä läheisesti, vaan on hyvin todennäköistä, että ensimmäisessä vaiheessa kaapattua lähdemateriaalia käsitellään vasta viikkojen tai kuukausienkin kuluttua; ja itse animointihan tapahtuu vasta loppukäyttäjän laitteella peliä pelatessa. Niinpä vaiheiden tuottamat tulokset säilötään ulkoisiin tiedostoihin, minkä ansiosta järjestelmän osaset ovat helposti muokattavissa toisistaan riippumatta olettaen, että tulosten tiedostomuodoissa ei tapahdu muutoksia. Seuraavissa alakohdissa esitellään tarkemmin nämä vaiheet ja niiden vastuualueet.

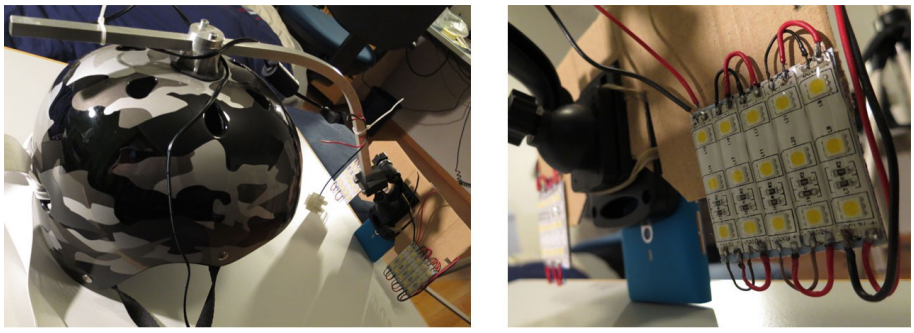
3.2.1 Kaappausvaihe

Ensimmäinen vaihe on kaapata lähdemateriaalia, jonka perusteella animaatio lopulta generoidaan. Lähdemateriaaliin sisältyy ensinnäkin videokuva näyttelijän kasvoista, joihin on aseteltu merkkipisteitä kasvonliikkeiden tunnistamista varten, ja toiseksi sensoridata, jonka perusteella näyttelijän päänliikkeitä pyritään tunnistamaan. Kutsutaan tätä vaihetta kaappausvaiheeksi.

Tämä osa järjestelmää toteutetaan mobiilisovelluksena älypuhelimelle, jossa on vähintään 720p-resoluution (1280x720) videokamera ja jonka kuvanopeus on ainakin 25 kuvaa sekunnissa. Lisäksi älypuhelimesta on löydyttävä kiihtyvyyssensori, joka löytyy-

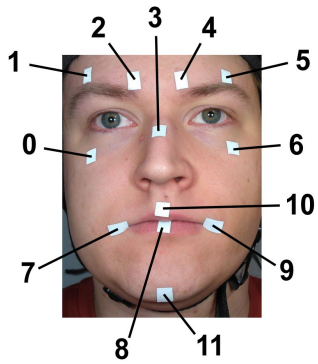
kin lähestulkoon jokaisesta älypuhelimesta. Mobiilisovellus tallentaa videokuvan ja sensoridatan, joita seuraava vaihe käyttää syötteinään.

Kasvoanimaation kaappaamiseksi kasvojen täytyy pysyä videokuvassa mahdollisimman vakaasti paikoillaan. Koska järjestelmän on sallittava ja tallennettava myös päänliikkeitä, on älypuhelin kiinnitettävä siten, että se pysyy jatkuvasti samassa kohdassa ja asennossa kasvoihin nähden. Tässä työssä ongelma on ratkaistu rakentamalla halvasta rullalautailukypärästä, matkapuhelintelineestä ja alumiinitangosta kuvan 3.2 mukainen teline, joka pysyy tukevasti paikoillaan ja seuraa päänliikkeitä minimaalisesti heilahdellen kääntymisnopeuden muuttuessa.



Kuva 3.2. *Kypäräteline kameralle.*

Älypuhelin kiinnitetään tukevasti matkapuhelintelineeseen ja varmistetaan kumilenkkien avulla, että se ei pääse putoamaan. Puhelimen ympärille on kiinnitetty ledivaloja kasvojen tasaisen valaisun saavuttamiseksi. Matkapuhelintelineen varsi on säädettävä, jotta älypuhelimen kameran ja valot saa helposti kohdistettua näyttelijän kasvoihin. Teline on liitetty alumiinitankoon, joka puolestaan on ruuvattu tukevasti kypärän päälle. Järjestelmän etupainoisuutta on pyritty tasoittamaan kypärän taakse ja näyttelijän vyöhön tai tuoliin kiinnitettävällä kuminauhalla, joka toimii vastapainona puhelimelle ja sen telineelle. Tämä kuitenkin lisää jo alkujaan melko raskaan telineen näyttelijän niskoille aiheuttamaa rasitusta, mutta se sallittakoon tämän prototyypin puitteissa.



Kuva 3.3. *Tunnistettavat kasvojen merkkipisteet.*

Järjestelmän on tunnistettava kuvassa 3.3 esitetyt kaksitoista kasvojen merkkipistettä. Ne on kokeilujen perusteella todettu riittäviksi tarpeeksi luonnollisen kasvoanimaation tuottamiseksi. Piste #3 toimii kontrollipisteenä ja eliminoi kameran tärähdyksiä pään liikkuessa, kun kaikki varsinaiset merkkipisteet ilmoitetaan suhteessa tämän kontrollipisteen sijaintiin. Järjestelmän suunnittelussa tulee huomioida se, että tätä merkkipistekonfiguraatiota (pisteiden määrää ja suurpiirteisiä sijainteja) voi tarvittaessa muuttaa mahdollisimman helposti kooditasolla.

3.2.2 Käsittelyvaihe

Seuraava vaihe ottaa kaappausvaiheen taltioiman lähdemateriaalin ja muuntaa sen 3D-mallin animoimiseen kelpaavaksi dataksi. Vaiheen suorittaa pelinkehittäjä Windows-pohjaisella tietokoneella. Kutsutaan tätä vaihetta käsittelyvaiheeksi.

Käsiteltävää lähdemateriaalia voi leikata käyttäjän erikseen tuottamalla leikkauslistalla, joka määrittää materiaalista valittujen yksittäisten otosten alkamis- ja päättymishetket sekä nimet. Leikkauslistan voisi tuottaa esimerkiksi videonleikkausohjelmistoon kirjoitettavalla skriptillä, mutta se on rajattu tämän työn ulkopuolelle. Ääniraidan on pysyttävä synkronoituna kaiken lähdemateriaalin kanssa leikkelystä riippumatta. Edellä mainitun kalibroinnin ja muiden animaatiodatan (hieno)säätöjen toteuttaminen on myöskin käsittelyvaiheen vastuulla.

Varsinainen lähdemateriaalin käsittely alkaa videokuvasta, josta käsitellään vain leikkauslistan mukaiset aikavälit. Kuvasta erotellaan järjestelmän kannalta oleellinen osa, siis etuala, josta pyritään erottelemaan kasvojen merkkipisteet ja yksilöimään ne, siis tunnistamaan mikä piste mikäkin on. Yksilöinti on tärkeää, sillä ei ole soveliasta, että kulmakarvapiste havaitaankin jossain vaiheessa leuan alueelta. Merkkipisteiden seuraamisessa on tärkeää varautua siihen, että ne voivat hetkellisesti kadota näkyvistä, vaikka ne ja kamera olisi kuinka huolellisesti aseteltu. Esimerkiksi alahuulen merkkipiste voi helposti jäädä piiloon huulen alle. Merkkipisteen kadotessa puuttuva data tulee täydentää interpoloimalla viimeisimmän havainnon ja seuraavan havainnon välillä.

Sensoridataa puolestaan käsitellään ja tarvittaessa eri sensorien dataa yhdistellään sellaiseksi, että sillä voidaan ilmaista näyttelijän pään kääntyminen X-, Y- ja Z-akseleiden suhteen. Lisäksi sekä merkkipiste- että sensoridataa rajoitetaan niin, että esimerkiksi pää ei käänny enempää kuin ihmisen pää todellisuudessa pystyisi kääntämään, vaikka lähdemateriaalin perusteella niin saattaisi käydäkin. Kaikki tuloksena saatava data suodatetaan vielä epätoivotun kohinan vaimentamiseksi.

3.2.3 Animointivaihe

Viimeinen vaihe pitää sisällään varsinaisen 3D-mallin animoimisen reaaliaikaisesti käsittelyvaiheesta saadun datan perusteella. Tämä vaihe suoritetaan pääasiassa vasta loppukäyttäjän pelatessa lopullista peliä omalla pelilaitteellaan, mutta sitä tarvitaan myös osana käsittelyvaihetta lopputuloksen esikatselun mahdollistamiseksi. Tässä vaiheessa

kriittistä on reaaliaikaisuus. Etenkin animaation on pysyttävä ajassa ääniraidan kanssa, jotta huultenliikkeet vastaisivat kuultavaa puhetta. Kutsutaan tätä vaihetta animointivaiheeksi.

3.3 Kaappaussovellus

Videon ja sensoridatan kaappaukseen käytettäväksi älypuhelimeksi valittiin Windows Phone 7.1 -rajapinnalla (markkinoinnissa käytetty versionumero 7.5, koodinimi Mango) ohjelmoitava Nokia Lumia 800, koska siinä oli käytettävissä olevista puhelimista paras videokuvanlaatu. Puhelimien kuvanlaatua arvioidessa otettiin huomioon silmämääräisesti yleinen kuvanlaatu, liikkeen terävyys (tai valotusaika) kunnollisessa valaistuksessa sekä kuvanopeuden vakaus.

Parhaan tuloksen saavuttamiseksi liikkeenseurantaa suoritettaessa päätettiin näyttelijän kasvoille aseteltavina merkkipisteinä käyttää heijastavan teipin palasia. Tällöin kuvaustilanteessa voidaan kytkeä päälle älypuhelimen kuvausvalo, joka valaisee teipinpalat ja saa ne hohtamaan videokuvassa valkoisena. Näin merkkipisteet erottuvat selkeästi kasvoista ja taustasta olettaen, että taustalla ei ole muita kirkkaita alueita, kuten toisia valonlähteitä.

Itse sovellus on rakenteeltaan ja toiminnoiltaan äärimmäisen yksinkertainen. Sovelluksen käyttöliittymänä toimii näytöllä esitettävä kameran näkemä kuva, jonka tallentaminen käynnistetään ja pysäytetään painamalla laitteen fyysistä kamerapainiketta. Tallennuksen alussa sovellus käynnistää älypuhelimen kiihtyvyyssensorin, jonka dataa puhelimen käyttöjärjestelmä välittää takaisinkutsumalla sovelluksen määrittämää tapahtumakäsittelijämetodia. Tapahtumakäsittelijä säilöö muistiin kunkin datapaketin aikaleiman sekä kaiken järjestelmän tarvitseman datan: kiihtyvyyssensorin tapauksessa X-, Y- ja Z-kiihtyvyyslukemat.

Koska tallennuksen aikana videokuvaa tallennetaan jatkuvasti laitteen flash-muistille, päätettiin sensoridata säilöä käyttömuistiin siihen hetkeen saakka, että tallennus pysäytetään. Tällöin sensoridata tallennetaan lopullisesti flash-muistille. Laitteessa on käyttömuistia 512 megatavua eikä sen käyttöjärjestelmä tue sovellusten moniajoa, joten muistin loppuminen tavallisessa kuvaustilanteessa, jossa otot tuskin koskaan kestävät muutamaa minuuttia pitempään, on hyvin epätodennäköistä.

Toteutuksen suurimmaksi ongelmaksi osoittautui Windows Phone 7.1 -rajapinnasta puuttuva tuki tallentaa 720p-resoluutioista videota. Käyttöjärjestelmän omalla kamerasovelluksella tämä kuitenkin on mahdollista. Asiaa tutkimalla selvisi lopulta, että puhelimen käyttöjärjestelmän GAC (Global Assembly Cache) pitää sisällään kirjastoja, joilla käyttöjärjestelmän kamerasovellus on toteutettu, mutta jotka eivät ole julkisesti saatavilla sovelluskehittäjille. Ne on kuitenkin mahdollista purkaa Windows Phone SDK:n sisältämästä emulaattorista, jolloin niiden tarjoamiin toimintoihin pääsee käsiksi vaikkapa Thomas Hounsellin tarjoamien ohjeiden avulla [23]. Tämän työn kannalta oleellinen

kirjasto oli *Microsoft.Phone.Media.Extended*, jonka paljastama *VideoCamera*-luokka tarjosi mahdollisuuden 720p-resoluutioisen videon tallentamiseen. Epävirallisten rajapintojen käyttäminen voisi hankaloittaa tai jopa estää sovelluksen saamisen Microsoftin sovelluskauppaan, mutta tätä järjestelmää ei onneksi olla julkaisemassa laajalle yleisölle eikä sitä etenkään ole tarvetta saada sovelluskaupan hyllylle.

Tiedonsiirto älypuhelimesta tietokoneelle, jolla datankäsittely suoritetaan, tapahtuu Windows Phone SDK:n sisältämän *IsolatedStorageExplorerTool*-sovelluksella. Sen avulla voi USB-yhteyden kautta siirtää sovelluksen yksityisen tallennustilan sisällön tietokoneelle. Koska kyse on prototyypistä, eikä tiedonsiirto ole järjestelmän kannalta kovin oleellinen osa, on se päätetty ajan säästämiseksi jättää toteuttamatta itse mobiilisovelluksesta.

3.4 Liikkeenseurantasovellus

Liikkeenseurantasovellus on käsittelyvaiheen osa, joka etsii videokuvasta näyttelijän kasvoille asetettuja merkkipisteitä, seuraa niiden liikkeitä ja tallettaa löydöksensä tiedostoon käsittelysovelluksen käytettäväksi. Sovellus on komentorivipohjainen eikä sitä ole tarkoitettu suoraan loppukäyttäjän suoritettavaksi, vaan sen suoritus laukaistaan käsittelysovelluksen kautta.

Sovelluksen toiminta rakentuu vahvasti alakohdassa 3.4.1 esiteltävän avoimen lähdekoodin OpenCV-tietokonenäkökirjaston tarjoamien toimintojen päälle. Sovelluksen eri suoritusvaiheita kuvataan seuraavissa alakohdissa. Ensimmäisenä suoritettavan järjestelmän kalibroinnin (alakohta 3.4.2) ja siihen sisältyvän merkkipisteiden yksilöimisen (alakohta 3.4.3) jälkeisen liikkeenseurannan voi jakaa suurpiirteisesti kahteen erilliseen vaiheeseen: merkkipisteiden etsimiseen ja löydettyjen pisteiden käsittelyyn. Merkkipisteiden etsiminen tapahtuu OpenCV-kirjaston avulla, kuten alakohdassa 3.4.4 on kuvattu. Kun kaikki potentiaaliset merkkipisteet on löydetty videokuvasta, pyritään seuraamaan merkkipisteiden liikettä videon kuvasta toiseen alakohdan 3.4.5 mukaisesti. Merkkipisteiden liikedata tallennetaan lopulta tiedostoon käsittelysovellusta varten.

3.4.1 OpenCV

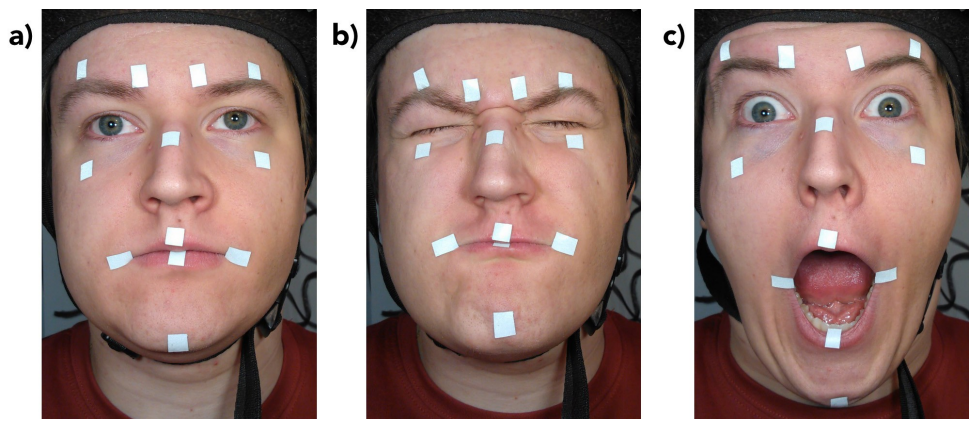
OpenCV on laajalti käytössä oleva tietokonenäkökirjasto (engl. Computer Vision), jonka tavoitteena on tarjota helppokäyttöinen perusta tietokonenäköä hyödyntävien sovellusten toteuttamiseksi. Kirjasto on keskittynyt erityisesti vastaamaan reaaliaikaisten sovellusten tarpeisiin, joten koodin tehokkuuteen ja optimointiin on käytetty paljon vaivaa. OpenCV on avointa lähdekoodia (erittäin vapaan käytön salliva BSD-lisenssi [45; 64]) ja kirjoitettu C:llä ja C++:lla, mutta sille on toteutettu myös rajapintoja muihin kieliin, kuten Pythoniin ja Rubyyn. Avoimuuden ansiosta OpenCV:n käyttäjä- ja kehittäjäkuntaan sisältyy niin monia suuryrityksiä (kuten Google, IBM ja Intel) kuin yliopistojakin (kuten Stanford ja MIT) ympäri maailman. [3, s. 1–2]

Kirjaston alulle panneen Gary Bradskin ja Adrian Kaehlerin mukaan OpenCV syntyi vuonna 1999 Intelin tutkimuslaboratoriossa yhteistyössä Software Performance Libraries -ryhmän ja Intelin venäläisten toteuttajien ja optimoijien kanssa. Ensimmäinen virallinen julkaisuversio 1.0 näki päivänvalon vuonna 2006. [3, s. 6–7] Nykyään projektin ylläpitovastuu on päätynyt Itseezille, joka on digitaalisen median älykkään käsittelyn tutkimus- ja kehitystyötä tekevä yritys [28]. Tässä työssä käytetty OpenCV-versio on heinäkuussa 2012 julkaistu 2.4.2, mutta tuoreempiakin versioita on jo tätä kirjoitettaessa saatavilla.

OpenCV:n kotisivujen mainostamasta yli 2500 optimoidusta algoritmista [45] ovat tämän työn kannalta oleellisia lähinnä kuvanmuokkaukseen ja kuva-alueiden ääriviivojen tunnistukseen liittyvät algoritmit. Käännettynä FFmpeg-tuen [19] kanssa OpenCV:llä pystyy lisäksi lukemaan monia erilaisia videotiedostomuotoja ja dekoodaamaan monenlaisia videokoodekkeja, jolloin älypuhelimella kuvatut MP4-videotiedostot (MPEG-4 Part 14) saadaan sellaisinaan käsiteltyä ilman tarvetta muuntaa niitä esimerkiksi pakkaamattomiksi AVI-tiedostoiksi (Audio Video Interleave).

3.4.2 Järjestelmän kalibrointi

Järjestelmän kalibrointia varten on kuvaushetkellä otettava kolme kalibraatioilmettä, joita kutsutaan oletusilmeeksi, minimiksi ja maksimiksi. Mikäli kamera liikkuu tai kääntyy suhteessa kasvoihin tai merkkipisteet vaihtuvat tai siirtyvät, on kalibrointi suoritettava uudelleen. Oletusilme (tai lepoilme, kuva 3.4a) vastaa käytettävän 3D-mallin oletusilmettä, joten kasvojen ollessa levossa näytetään 3D-malli alkuperäisellä ilmeellään. Minimi-ilmeessä (kuva 3.4b) on tarkoitus saada kaikki merkkipisteet niin lähelle kasvojen keskipistettä kuin on mahdollista. Maksimi-ilmeen (kuva 3.4c) tarkoitus on päinvastainen, eli merkkipisteet pyritään liikuttamaan mahdollisimman etäälle kasvojen keskipisteestä. Käyttäjän on valittava nämä yksittäiset ilmekuvat kaapatusta videomateriaalista ja tallennettava ne erillisiin kuvatiedostoihin, jotka kalibrointi ottaa syötteenään.



Kuva 3.4. *Kalibraatioilmeet: a) oletusilme, b) minimi ja c) maksimi.*

Kun kuvatiedostot on annettu kaappaussovelluksen kalibrointivaiheelle, käyttäjän on tarvittaessa säädettävä merkkipisteiden etsimisessä käytettävän raja-arvofunktion parametreja käyttöliittymän liukusäätimillä, jotta merkkipisteet saadaan mahdollisimman hyvin erottumaan muusta kuvasta. Tämän jälkeen käyttäjä osoittaa tuplaklikkaamalla kustakin ilmekuvasta järjestelmälle, mitkä havaituista kuva-alueista ovat liikkeenseurannassa käytettäviä merkkipisteitä. Kun tämä on suoritettu, sovellus yksilöi käyttäjän osoittamat merkkipisteet ja tallentaa kalibraatiodatan tiedostoon käytettäväksi myöhemmissä vaiheissa. Kalibraatiodata koostuu kunkin ilmeen merkkipisteiden sijainneista, pinta-aloista sekä raja-arvofunktion parametrien arvoista. Samaa kalibraatiodataa voidaan käyttää kaikkien niiden lähdemateriaalileikkeiden kanssa, joissa kamera ja merkkipisteet ovat samoilla paikoilla.

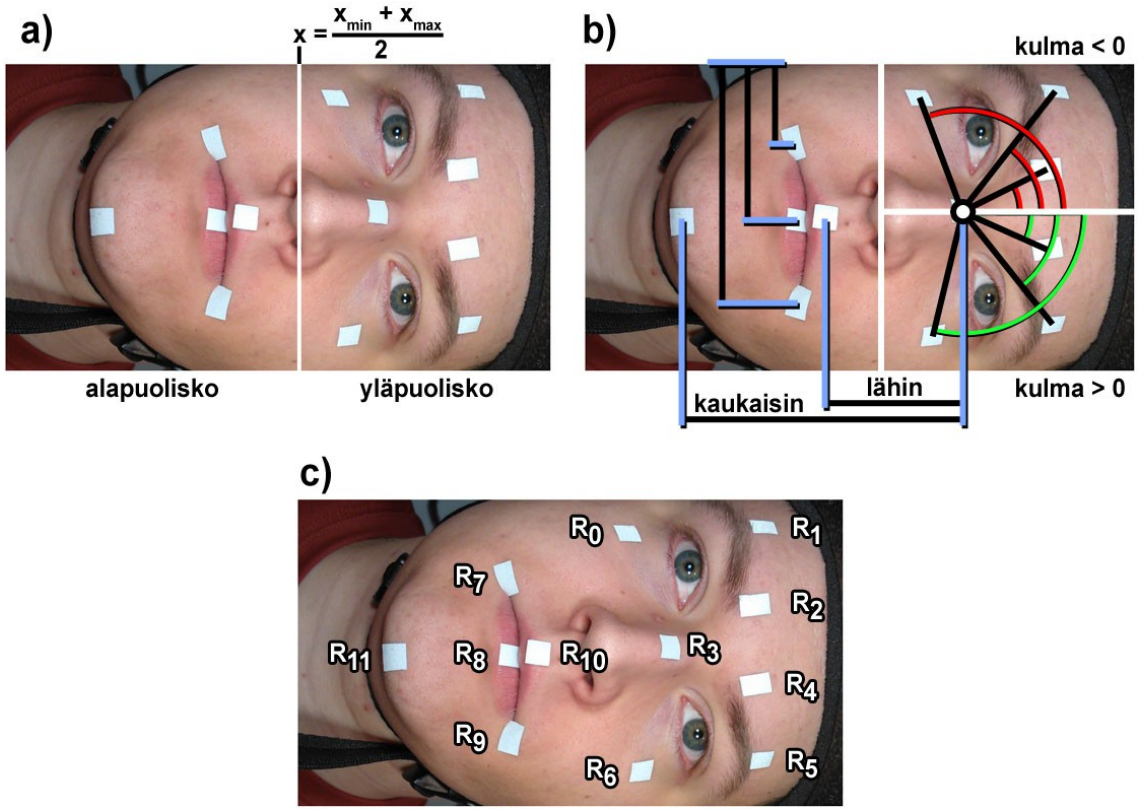
3.4.3 Merkkipisteiden yksilöiminen

Merkkipisteiden yksilöinti aloitetaan etsimällä kontrollipiste #3, joka on suurin piirtein pistejoukon keskellä. Tämä onnistuu yksinkertaisesti laskemalla pistejoukon X- ja Y-koordinaattien keskiarvot ja etsimällä pistejoukon piste, joka on lähinnä keskiarvoa. Lisäksi lasketaan pistejoukon X-koordinaattien minimin ja maksimin keskiarvo ja käytetään sitä jakamaan pistejoukko kasvojen ylä- ja alapuoliskoon niin, että keskiarvoa pienemmät (siis kuvassa vasemmalla olevat) pisteet kuuluvat alapuoliskoon ja loput yläpuoliskoon kuvan 3.5a mukaisesti. Kontrollipistettä lukuun ottamatta jokaiselle pisteelle P lasketaan vielä kulma suhteessa kontrollipisteeseen C nähden. Paikkavektorina (origo O) ilmaistuna se tarkoittaa vektorin $\vec{v} = \vec{OP} - \vec{OC}$ ja X-akselin välistä kulmaa, joka saadaan laskemalla

$$\alpha = atan2(v_y, v_x). \tag{3.1}$$

Kasvojen yläpuoliskon pisteet voidaan keskenään järjestää tämän astelukeman mukaan kasvavaan järjestykseen kuvan 3.5b mukaisesti. Näin saadaan kuvassa 3.5c esitetyn lopputuloksen R pisteet R_0–R_6.

Kasvojen alapuoliskon yksilöimiseksi astelukema ei kelpaa, sillä huulten ylä- ja alapisteet (#10 ja #8) sekä leukapiste (#11) ovat suurin piirtein linjassa toistensa ja kontrollipisteen kanssa. Niinpä niiden astelukemat suhteessa toisiinsa voivat vaihdella suunliikkeiden mukana. Onneksi ylähuulen piste sattuu olemaan kasvojen alapuoliskon pisteistä aina kontrollipistettä lähimpänä ja leukapiste puolestaan siitä kauimpana. Laskemalla pisteiden ja kontrollipisteen väliset etäisyydet voidaan siis yksilöidä nämä kaksi pistettä, jolloin jäljelle jäävät pisteet on helppo järjestää kasvavan Y-koordinaattiarvon perusteella. Näin saadaan lopputuloksen R pisteet R_7–R_{11}.



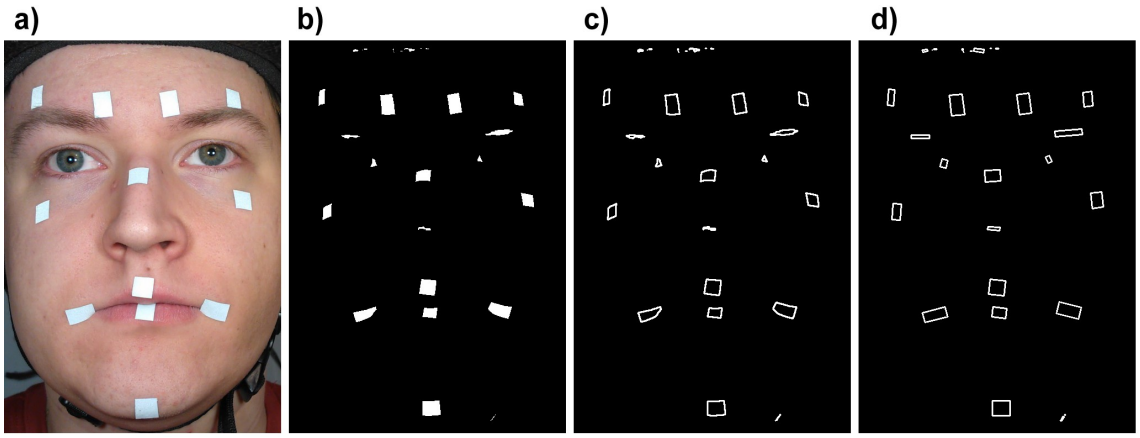
Kuva 3.5. *Merkkipisteiden yksilöimisen vaiheet: a) löydetyt pisteet jaettuna kasvojen ylä- ja alapuoliskoon, b) pisteiden järjestäminen kulman ja etäisyyden perusteella ja c) yksilöidyt pisteet.*

Merkkipisteiden yksilöinti siis olettaa, että se saa syötteenä täsmälleen varsinaisten merkkipisteiden lukumäärän verran potentiaalisia merkkipisteitä, eli tässä tapauksessa kaksitoista kappaletta. Yksilöinti suoritetaan kalibroinnin lisäksi myös jokaisen videokuvan liikkeenseurannasta saadulle tulokselle varmistamaan, että tuloksena saadut merkkipisteet on yksilöity oikein. Ennen varsinaista liikkeenseurantaa on kuitenkin etsittävä uudet potentiaaliset merkkipisteet, joista pyritään havaitsemaan varsinaiset merkkipisteet.

3.4.4 Merkkipisteiden etsiminen

Kuten aiemmin mainittiin, merkkipisteinä käytettiin heijastavaa teippiä, jolloin pisteet korostuvat kuvassa kirkkaina alueina. Merkkipisteiden löytämiseksi on siis pystyttävä erottamaan nämä kirkkaat alueet (etuala) muusta kuvasta (taka-ala). OpenCV-kirjasto tarjoaa tähän tarkoitukseen hyödyllisen raja-arvofunktion, *threshold*, jonka avulla kuvan voi muuntaa binääriseksi vertailemalla kunkin pikselin kirkkausarvoa annettuun raja-arvoon. Tällöin alkuperäisestä kuvasta 3.6a saadaan kuvan 3.6b tapainen kuva, jossa kunkin pikselin arvo on binäärinen: pikseli joko kuuluu etualaan (valkoinen) tai taka-alaan (musta). Jotta järjestelmä sallisi jonkin verran epätasaisuutta merkkipisteiden valaisussa,

on kuitenkin hyödyllisempää käyttää funktiota *adaptiveThreshold*, joka tekee raja-arvotarkastuksen suhteessa tarkasteltavan pikselin lähiympäristön kirkkausarvoihin. Näin merkkipisteet havaitaan lähiympäristönsä kirkkaimpina alueina.



Kuva 3.6. *Merkkipisteiden etsimisen vaiheet: a) alkuperäinen kuva, b) muunnos binääriseksi, c) havaittujen alueiden reunaviivat ja d) alueisiin sovitetut suorakulmiot.*

Seuraava tehtävä on löytää binäärisestä kuvasta valkoiset etualan alueet. Tämä onnistuu funktiolla *findContours*, joka etsii kuvassa esiintyvien yhtenäisten alueiden reunaviivoja. Tuloksena saatuihin, reunaviivojen määrittämiin alueisiin, joista on esimerkki kuvassa 3.6c, sovitetaan funktiolla *minAreaRect* mahdollisimman pieni käännetty suorakulmio, joka pitää alueen sisällään. Näin päästään kuvan 3.6d tilanteeseen. Käännetty suorakulmio on käännös OpenCV:n käyttämästä tietorakenteesta *RotatedRect*, mutta yleisesti käytetty termi vastaavanlaiselle suorakulmiolle on OBB (Oriented Bounding Box). OBB poikkeaa tavallisesta X- ja Y-akselien suuntaisesta suorakulmiosta (AABB, Axis-Aligned Bounding Box) nimensä mukaisesti sillä, että sitä voi lisäksi kääntää Z-akselin ympäri haluamaansa kulmaan.

Kun kunkin alueen kattava suorakulmio on löydetty, voidaan karsia löydetyistä alueista pois ne, jotka ovat liian pieniä ollakseen merkkipisteitä. Tällaiset pienet videokuvan valopisteet voivat olla esimerkiksi kuvausvalon heijastumia silmistä tai silmälaseista. Toisaalta ne voivat myös olla hetkellisesti pieniksi kutistuvia huulten merkkipisteitä. Niinpä sopivaksi rajapinta-alaksi valittiin eri arvoja kokeilemalla lopulta 25 neliöpikseliä, joka neliönmuotoisen alueen tapauksessa vastaisi viiden pikselin sivunpituutta. Tällä arvolla virheelliset valopisteet karsiutuivat suurimmaksi osaksi pois niin, että itse merkkipisteille jäi kuitenkin runsaasti varaa kutistua täydestä koostaan.

Kustakin rajapinta-alaa suuremmasta alueesta otetaan jatkokäsittelyä varten talteen alueen pinta-ala ja sijainti, joksi valittiin alueen X-koordinaattimaksimi ja keskipisteen Y-koordinaatti. Nämä arvot tallennetaan erilliseen tiedostoon, jotta merkkipisteitä ei tarvitsisi etsiä useampaan kertaan, vaikka merkkipisteiden yksilöimistä ja liikkeenseurantaa tarvitsisikin tehdä useammin. Merkkipisteiden etsimisen lopputuloksena saadaan siis

ainoastaan lista kussakin videon kuvassa olevista potentiaalisista merkkipisteistä, joita saattaa paikoin olla enemmän tai vähemmän kuin varsinaisia merkkipisteitä.

3.4.5 Merkkipisteiden liikkeenseuranta

Merkkipisteiden liikkeenseurannassa käytetään hyväksi kalibraatiodataa sekä tietoa merkkipisteiden edellisestä kuvasta havaituista sijainneista, joita ei kuitenkaan esimerkiksi ensimmäisen kuvan tapauksessa ole saatavilla. Kullekin merkkipisteelle pyritään löytämään merkkipisteiden etsimisestä saadusta pistejoukosta paras kandidaatti laskemalla kunkin pistejoukon pisteen painoarvo suhteessa kuhunkin haettavaan merkkipisteeseen. Painoarvon laskemisessa otetaan huomioon pisteeseen liittyvän alueen pinta-ala, etäisyys kalibraatiodatan merkkipisteen eri sijaintien välille muodostetuista janoista (minimistä oletukseen ja oletuksesta maksimiin) sekä etäisyys edellisestä kuvasta havaittuun merkkipisteen sijaintiin. Painoarvo pyrkii suosimaan oikeankokoisia sekä lähellä edellistä havaintoa ja kalibraatiodatan kuvaamaa liikerataa olevia pisteitä. Kokeilujen perusteella hyvä kaava laskea painoarvo w pisteelle P suhteessa merkkipisteeseen M on seuraavanlainen:

$$\begin{aligned} calibDistSq &= min(distSq(lsMinDef_M, P), distSq(lsDefMax_M, P)) \\ w &= (2 \cdot calibDistSq + distSq(M_{prev}, P)) \cdot |M_{areaDef} - P_{area}|, \end{aligned} \tag{3.2}$$

missä $lsMinDef_M$ on jana merkkipisteen minimi- ja oletusilmeiden sijaintien välillä, $lsDefMax_M$ on jana oletus- ja maksimi-ilmeiden sijaintien välillä, M_{prev} on merkkipisteen edellinen havaittu sijainti, $M_{areaDef}$ on merkkipisteen oletusilmeestä laskettu pinta-ala ja P_{area} on pisteen pinta-ala. Funktio *distSq* palauttaa annettujen pisteiden tai pisteen ja janan välisen etäisyyden neliön. Pinta-alavertailua voisi myös tehdä sen kalibraatioilmeen perusteella, jonka merkkipistettä lähimpänä tutkittava piste on, mutta se osoittautui ongelmalliseksi huulien alueella, jossa merkkipisteet voivat kutistua merkittävästi aiheuttaen virheellisiä tunnistuksia pienten kohina-alueiden kanssa.

Järjestämällä painoarvot kasvavaan järjestykseen voidaan löytää parhaat kandidaatit kunkin merkkipisteen uudelle sijainnille. Järjestyksessä ensimmäinen ja mahdollisesti lähin piste ei kuitenkaan välttämättä ole oikea, sillä nopeissa liikkeissä tarkkailtava piste P saattaa liikkua kuvan aikana kauas ja jokin toinen piste lähemmäs edellistä havaintoa. Voi myös tulla tilanteita, joissa sama potentiaalinen merkkipiste on paras kandidaatti kahdelle tai useammallekin merkkipisteelle. Olettaen, että vaihe kuitenkin tunnistaa oikeat merkkipisteet, mutta ainoastaan sekoittaa niitä keskenään, ratkeaa ensimmäinen ongelma yksilöimällä merkkipisteet edellä kuvatulla tavalla. Jälkimmäinen ongelma vaatii havaittujen pistekandidaattiduplikaattien käsittelyä.

Parhaiden kandidaattien valinta ja duplikaattien havaitseminen ja käsittely seuraavat toisiaan toistuvasti, kunnes duplikaatteja ei enää havaita. Parhaaksi kandidaatiksi valitaan piste, jota ei ole vielä merkitty lopullisesti valituksi, jonka painoarvo on mahdollisimman pieni ja jonka etäisyys edelliseen havaintoon ei ylitä sovellukselle parametrina

annettua maksimiliike-etäisyyttä. Käytettyyn testimateriaaliin sopiva maksimietäisyys oli 75 pikseliä.

Kun kullekin merkkipisteelle on valittu paras kandidaatti, etsitään mahdollisia duplikaatteja vertaamalla kunkin merkkipisteen parasta kandidaattia muiden merkkipisteiden parhaisiin kandidaatteihin. Mikäli sama piste on valittu useamman merkkipisteen parhaaksi kandidaatiksi, tarkistetaan, kumpaa edellistä merkkipistehavaintoa lähempänä se on, poistetaan se etäisemmästä ja merkitään lopullisesti valituksi lähempään. Tämä tarkastus voi alentaa myös jo lopullisesti valituksi merkityn pisteen takaisin tavalliseksi kandidaatiksi. Koska parhaaksi kandidaatiksi ei valita lopullisesti valituksi merkittyjä pisteitä uudelleen, päättyy algoritmin suoritus lopulta joko siihen, että duplikaatteja ei enää löytynyt tai että uusia kandidaatteja ei enää pystytty valitsemaan. Näin merkkipiste voi esimerkiksi näkyvistä kadotessaan jäädä ilman uutta havaintoa, kun sopivaa kandidaattia ei löytynyt. Tällöin havaintoon liitetään tieto, että merkkipistettä ei löydetty ja asetetaan havainnon sijainniksi edellinen havaittu sijainti, jolloin tuloksena saadaan aina etsittävän merkkipistemäärän verran havaintoja.

Lopuksi uudet merkkipistehavainnot yksilöidään ja lopputuloksena saadaan tiedosto, joka sisältää havaintotiedot kustakin merkkipisteestä kussakin videon kuvassa. Tätä tiedostoa hyödyntää käsittelysovellus, joka siistii saamaansa dataa ja tuottaa siitä puolestaan lopulliseen animaatioon käytettävää dataa.

3.5 Käsittelysovellus

Käsittelysovelluksen tarkoituksena on ottaa eri lähteistä, älypuhelimen sensoreista ja liikkeenkaappaussovelluksesta, saatu data ja yhdistää se lopulliseksi animaatiodataksi. Sovellus on toteutettu Windows-käyttöjärjestelmille C#-kielellä käyttäen Microsoftin .NET-sovelluskehyksen versiota 3.5. Sovellus noudattaa MVC-arkkitehtuuria (Model-View-Controller), joka jakaa sovelluksen rakenteen käsiteltävästä datasta ja siihen kohdistuvista operaatioista koostuvaan malliin, dataa visualisoivaan näkymään ja kontrolleriin, joka sisältää sovelluksen toimintalogiikan [41]. Seuraavissa alakohdissa tutustutaan tarkemmin sovelluksen toiminnan kannalta oleellisimpiin osiin: suodinarkkitehtuuriin ja animaatiodatan tuottamiseen.

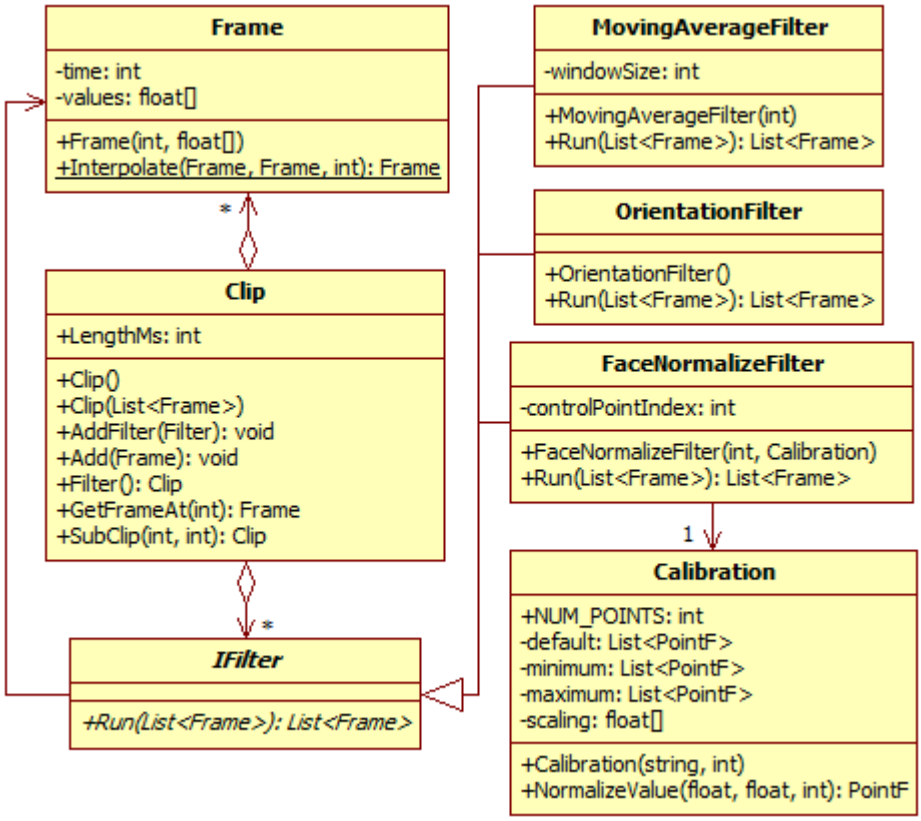
3.5.1 Suodinarkkitehtuuri

Käsittelysovelluksen keskeisin osa on sen suodinarkkitehtuuri. Arkkitehtuuri noudattaa suunnittelumallia strategia, jossa yhteisen rajapinnan taakse voidaan toteuttaa samalla tavoin käytettävää mutta vaikutuksiltaan eriävää toiminnallisuutta [62].

Kaikki sovelluksen käsittelemät datavirrat (liikkeenkaappausdata, sensoridata ja animaatiodata) kootaan kukin yhteen oman *Clip*-olioonsa. Nämä oliot koostuvat *Frame*-olioista, joista kukin kuvaa datavirran yhden tietyn ajanhetken arvoja. Yhteen ajanhetkeen liittyvien arvojen lukumäärää ei ole rajoitettu. Esimerkiksi kiihtyvyyssensorin ta-

pauksessa nämä arvot ovat sensorin X-, Y- ja Z-akselien mittauslukemat tietyllä ajanhetkellä. Aikayksikkönä sovelluksessa on käytetty millisekunteja kokonaislukuina. *Clip* olettaa, että *Frame*-oliot lisätään siihen aikajärjestyksessä. Datavirran tietyn ajanhetken arvo saadaan metodilla *GetValueAt*, joka hakee joko sen *Frame*-olion, joka sattuu juuri haetun ajanhetken kohdalle, tai yhden olion haetun ajan hetken kummaltakin puolelta, jolloin lopputulos saadaan *Frame*-luokan luokkafunktiolla *Interpolate*. Luokkafunktio suorittaa lineaarisen interpolaation [36] kahden *Frame*-olion data-arvojen välille ja palauttaa tuloksena haetun ajanhetken mukaisen *Frame*-olion.

Käytetty suunnittelumalli ilmenee suodinluokkien käytössä. Jokaisen *Clip*-olion sisältämää dataa voidaan suodattaa vapaasti lisäämällä olioon erilaisia *IFilter*-rajapinnan toteuttavia suotimia ja kutsumalla metodia *Filter*, joka suorittaa suodatuksen kutsumalla vuorollaan kunkin suodinolion *Run*-metodia ja palauttaa suodatetun *Clip*-olion. Suotimen *Run*-metodi ottaa parametrinaan listan *Frame*-olioita, joiden dataa se käsittelee omalla tavallaan ja palauttaa tuloksena suodatetut *Frame*-oliot. Arkkitehtuuria havainnollistaa kuva 3.7.



Kuva 3.7. *Suodinarkkitehtuurin luokkakaavio.*

Erilaisia suotimia sovellukseen toteutettiin kokeilumielessä useita, mutta lopulta käyttöön jäi ainoastaan kolme. *MovingAverageFilter* suorittaa datalle alipäästösuodatusta, eli karsii nopeita vaihteluja kuten kohinaa. Nimensä mukaisesti suodatus on toteutet-

tu liikkuvan keskiarvon avulla siten, että käsiteltävän ajanhetken keskiarvo lasketaan ottamalla suotimelle annetun ikkunakoon (*windowSize*) verran näytteitä (*Frame*) sekä menneistä että tulevista. Mikäli näytteitä ei ole saatavilla tarpeeksi, kuten esimerkiksi datavirran alussa tai lopussa, otetaan keskiarvoon mukaan niin monta näytettä kuin saadaan. Alipäästösuodatusta käytetään sekä liikkeenseurannan että päänliikkeiden datavirroissa.

OrientationFilter suorittaa kiihtyvyyssensorin datavirralle edellä kuvattujen kaavojen 2.4 ja 2.7 mukaisen muunnoksen kiihtyvyyslukemista pään kallistus- ja kumarruskulmiin. Kumarruskulma jouduttiin lisäksi muuntamaan radiaaniväliltä $\left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$ välille $(-\pi, \pi]$ käyttäen hyväksi tietoa siitä, että tangentin arvon ollessa negatiivinen, sijoittuu kulma yksikköympyrän toiselle tai neljännelle neljännekselle. Muunnos tehtiin alkuperäissignaalin Y- ja Z-arvojen merkkien perusteella. Laskukaavan X-arvon kertoimena käytettiin arvoa $\mu = 0{,}001$.

FaceNormalizeFilter pyrkii muuntamaan kasvojen liikkeenseurannan tuottaman merkkipisteiden sijaintidatan riittävän yhtenäiseksi riippumatta kameran ja merkkipisteiden sijainneista kuvassa. Apunaan se käyttää tietoa kontrollipisteen indeksistä ja kalibrointivaiheessa tuotetusta *Calibration*-luokan hallinnoimasta datasta. *Calibration*-olio luodaan antamalla sille kalibraatiodatatiedoston nimi ja kontrollipisteen indeksi. Tiedostosta luetaan kunkin kalibraatioilmeen merkkipisteiden sijainnit ja muunnetaan ne suhteellisiksi kontrollipisteiden sijaintiin nähden. Tämän jälkeen ilmeistä lasketaan kullekin merkkipisteelle skaalausarvo, joka on merkkipisteen maksimaalinen siirtymä oletusilmeestä. *FaceNormalizeFilter* muuntaa vastaavalla tavalla merkkipisteiden sijainnit suhteellisiksi kontrollipisteeseen nähden ja kutsuu sitten kullekin merkkipisteelle *Calibration*-olion *NormalizeValue*-metodia, joka muuttaa pisteen koordinaatit suhteellisiksi merkkipisteen sijaintiin oletusilmeessä ja jakaa tuloksen merkkipisteen skaalausarvolla. Koordinaattien muuntamisella suhteellisiksi kontrollipisteisiin nähden pyritään estämään kameran tärähdysten aiheuttamaa virhettä. Suhteellisuudella oletusilmeen merkkipisteiden sijainteihin pyritään vähentämään kameran sijainnin ja asennon ja merkkipisteiden sijaintien vaihtelusta aiheutuvia virheitä. Skaalausarvolla pyritään minimoimaan kameran ja kasvojen välisen etäisyyden vaihtelusta aiheutuvia virheitä.

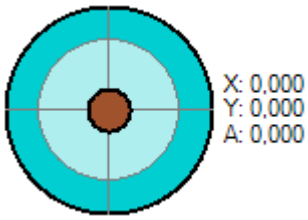
Datan suodatus tapahtuu käytännössä seuraavanlaisesti. Kiihtyvyyssensorin datavirtaa suodattaa ensin *MovingAverageFilter* ikkuna-arvolla kuusikymmentä ja sitten *OrientationFilter*, joka tuottaa lopulliset kulmalukemat. Liikkeenkaappauksen datavirtaa puolestaan suodattaa ensin *MovingAverageFilter* ikkuna-arvolla kaksi ja sitten *FaceNormalizeFilter*.

3.5.2 Muunnos animaatiodataksi

Kun data on suodatettu, alkaa viimeinen vaihe, jossa data muunnetaan lopulliseen muotoonsa animaatiodataksi. Tässä vaiheessa käyttäjän on suoritettava toinen kalibraatio

asettamalla kullekin animoitavalle luulle skaalausarvot, joiden mukaisesti datasta saatavat koordinaattiarvot skaalataan luiden koordinaattiarvoiksi. Tätä kalibraatiota ei resurssien puutteen vuoksi toteutettu sovelluksen käyttöliittymään, vaan skaalausarvot on kovakoodattu animaation hienosäätöä varten kehitettyihin *ControlStick*-säädinelementteihin (kuva 3.8), joista kukin vastaa yhtä animoitavaa luuta. Itse hienosäätöäkään ei sovellukseen vielä toteutettu, vaan säädinelementtien päätarkoitus on toistaiseksi havainnollistaa datan ja luiden välistä yhteyttä ja mahdollistaa luiden liikeratojen testaaminen sovelluksessa.

Lisäksi on suoritettava synkronointi kiihtyvyyssensorilta ja liikkeenkaappauksesta saadulle datalle. Kokeilujen perusteella havaittiin, että ainakin käytetyllä älypuhelimella videokuvan tallennus alkoi noin 500 millisekuntia kiihtyvyyssensoridatan tallennuksen jälkeen. Ottamalla tämä aikaero huomioon havaittiin lopputulos riittävän hyvin synkronoiduksi eikä resurssien rajallisuuden vuoksi päätetty käyttää enempää aikaa täsmällisemmän synkronointimenetelmän kehittämiseksi.



Kuva 3.8. *ControlStick-säädin.*

ControlStick-säädin toimii kuten monen nykyisen peliohjaimen analoginen tikkuohjain. Ruskeaa nuppia raahaamalla säätimen kaksiulotteista arvoa voi muuttaa mieleisekseen säätimeen asetettujen rajojen puitteissa. Säätimen ulkokehän säde on 1,5 ja sisemmän kehän säde 1,0. Normalisoidun kasvonliikedatan pitäisi sijoittua ulkokehän sisäpuolelle suurimman osan ajasta, mutta mikäli raja-arvo ylittyy, säädin skaalaa arvon niin, että se pysyy ulkokehän sisäpuolella. Yksi säädin sisältää tässä vaiheessa kaiken tarpeellisen tiedon säädintä vastaavan luun animaatiodatan tuottamiseksi. Lopullinen data syntyy kaikessa yksinkertaisuudessaan kertomalla säätimen arvo säätimeen asetetulla skaalausarvolla. Tämä data tallennetaan lopulta tiedostoon, josta animaatiokirjasto voi sen lukea ja käyttää sitä 3D-mallin animointiin.

3.6 Animaatiokirjasto

Tämän työn puitteissa animaatiokirjasto toteutettiin ainoastaan käsittelysovelluksen esikatseluksi C#-kielellä Microsoftin .NET-sovelluskehykseen. Kirjaston ytimenä toimii C++-pohjainen Irrlicht 3D-grafiikkakirjasto, joka vastaa 3D-mallin lataamisesta ja renderöinnistä sekä luurankopohjaisesta animaatiojärjestelmästä [25]. Siten animaatiokirjaston vastuulle jää ainoastaan luurangon luiden liikuttelu käsittelysovelluksen tuotta-

man animaatiodatan mukaisesti. Grafiikkakirjaston kanssa kommunikointiin käytettiin Irrlichtin .NET-käärettä, Irrlicht Lime [26].

Kirjaston toiminta on suoraviivaista. Alustusvaiheessa animoitavalta 3D-mallilta pyydetään viitteet kuhunkin animoitavaan luuhun (*BoneSceneNode*) ja talletetaan ne ja niiden alkuperäiset sijainnit ja asennot muistiin. Lisäksi 3D-mallin animointitapa asetetaan ohjelmalliseksi, sillä oletuksena animointi tapahtuu 3D-mallitiedostoon sisällytetyn animaation perusteella. Käsittelysovelluksen tuottama animaatiodata luetaan tiedostosta muistiin käsittelysovelluksen määrittelemään *Clip*-luokkaan, kun animaatio käsketään ladattavaksi.

Seuraavaksi jokaisella ruudunpäivityksellä päivitetään luiden sijainteja ja asentoja. Päivitysmetodi saa parametrinaan esitettävän ajanhetken aikaleiman, jonka perusteella animaatiodatasta interpoloidaan kyseistä ajanhetkeä vastaavat, 3D-mallin luille asetettavat muunnosarvot. Käytännössä leukaluu ja päätä liikuttava niskaluu animoidaan muuttamalla niiden asentoja ja kaikki muut luut animoidaan muuttamalla niiden sijainteja. Animointi tapahtuu suhteessa luiden alkuperäisiin sijainteihin ja asentoihin suoraan käsittelysovelluksesta saatavan animaatiodatan mukaisesti. Esimerkiksi leukaluu animoidaan laskemalla luun alkuperäisen asennon ja animaatiodatan mukaisen uuden asennon summa ja asettamalla luun asennoksi tämän yhteenlaskun tulos.

Animaatiokirjaston reaaliaikavaatimus on pyritty saavuttamaan minimoimalla kirjaston jokaisella ruudunpäivityksellä tapahtuva animaatiodatan käsittelyn määrä. Käytännössä ainoa tarvittava käsittely on lukuarvojen lineaarista interpolointia ja yhteenlaskuja luiden alkuperäiseen tilaan suhteutetun liikkeen määrittämiseksi. Animaation synkronointia ääniraidan kanssa ei toteutettu, mutta sen pitäisi onnistua helposti, kun animaatiokirjaston päivitysmetodille annetaan aikaleimaksi suoraan toistettavan äänileikkeen nykyinen aika. Äänileikkeestä erillisen aikalaskurin käyttö ei ole suositeltavaa, sillä ne eivät välttämättä pysy samassa ajassa keskenään esimerkiksi järjestelmän kuormituksen takia.

4 ARVIOINTI

Tässä luvussa arvioidaan toteutetun järjestelmän toimivuutta. Arviointiperusteina käytetään aiemmin mainittuja tavoitteita: automaattisuutta (kohta 4.1), suorituskykyä (kohta 4.2), luotettavuutta (kohta 4.3) sekä lopputuloksena saatavan animaation luonnollisuutta (kohta 4.4). Luvun päätteeksi luetellaan vielä projektin aikana vastaan tulleita jatkokehitysajatuksia (kohta 4.5).

4.1 Automaattisuus

Järjestelmän suunnittelussa huomioitu automaattisuus toteutui niin, että käyttäjän toimia vaaditaan lopulta ainoastaan kahdessa prosessin vaiheessa. Ensinnäkin jokaista kuvauskertaa kohden on liikkeenseurannalle suoritettava kalibrointi, jossa käyttäjä säätää liikkeenseurantaparametreja ja osoittaa kalibraatiokuvista merkkipisteet, joita järjestelmän tulisi seurata. Tämän lisäksi käyttäjän on kunkin hahmon kohdalla määriteltävä skaalauskertoimet, joilla liikkeenseurantadata muunnetaan varsinaisen 3D-mallin luiden liikedataksi. Myös hienosäätöjen tekeminen vaatisi käyttäjän toimia, mikäli se olisi toteutettu, mutta sitä ei olisikaan mielekästä yrittää automatisoida.

Ideaalisessa tapauksessa yhden hahmon animaatioiden tuottamiseen vaadittaisiin siis käyttäjän toimia kertaalleen luiden skaalauskertoimien määrittämiseen sekä jokaista kuvauskertaa kohden kertaalleen liikkeenseurannan kalibroimiseen. Käytännössä automaattisuuden kasvattaminen tästä alkaisi jo uhata järjestelmän luotettavuutta ja lopputuloksen laatua. Niinpä järjestelmän automaattisuuden tasoa voidaan pitää hyvänä.

4.2 Suorituskyky

Järjestelmän suorituskykyä mitattiin laskemalla kuhunkin liikkeenseurannan vaiheeseen kulunut aika käsiteltäessä 3 466 kuvan, eli reilun 115,5 sekunnin (30 kuvaa sekunnissa), pituista testiaineistoa. Testaukseen käytetyssä tietokoneessa oli 3,0 GHz taajuudella toimiva Intelin Core2 Duo E8400 -prosessori sekä kahdeksan gigatavua muistia. Liikkeenseuranta ei hyödynnä kaksiytimisen prosessorin tarjoamaa rinnakkaisuutta. Mittauksen tulokset on lueteltu taulukossa 4.1.

Taulukko 4.1. *Liikkeenseurannan suorituskyky.*

	Kulunut aika (s)	Ajankäyttö / materiaalin kesto (%)
Kohteiden etsiminen	90,01	77,93
Liikkeenmallinnus	0,87	0,75
Yhteensä	90,88	78,68

Tulosten perusteella järjestelmään toteutettu liikkeenseuranta voisi toimia reaaliaikaisenakin, sillä koko liikkeenseuranta suoritetaan ajallisesti alle 80 prosentissa käsiteltävän materiaalin kestosta. Tästä ajasta suurin osa kuluu kohteiden etsimiseen ja loput liikkeenmallinnukseen, joka on erittäin nopea vaihe. Kaiken kaikkiaan tulos on erittäin hyvä verrattuna edelliseen prototyyppiin, jossa Adoben After Effects -ohjelmistolla suoritettu liikkeenseuranta vei aikaa noin 1 700 % käsitellyn materiaalin kestosta. Nykyinen prototyyppi on siis jopa 21 kertaa edellistä nopeampi. On tietysti huomioitava, että edellisen prototyypin käyttämä aika sisältää myös käyttäjän tekemät korjaukset liikkeenseurannassa ilmenneisiin virheisiin toisin kuin uuden prototyypin mittaustulokset. Yksittäisen kuvan käsittely on siitäkin huolimatta paljon tehokkaampaa uudessa prototyypissä.

4.3 Luotettavuus

Järjestelmän luotettavuudessa kriittinen osa on liikkeenseurannan luotettavuus. Siis havaitaanko merkkipisteiden liikkeet oikein kuvasta toiseen siirryttäessä. Tämän työn puitteissa testaus rajattiin kattamaan liikkeenmallinnuksen luotettavuutta. Testissä käytettiin 1 000 kuvan testiaineistoa, mikä tarkoittaa yhteensä 12 000 merkkipisteen tunnistusta, kun seurattavia merkkipisteitä oli 12 kappaletta. Luotettavuutta mitattiin laskemalla, kuinka monta kertaa merkkipisteen tunnistus tehtiin väärin, siis merkkipisteen uudeksi sijainniksi valittiin väärän kohteiden etsintävaiheessa havaitun kohteen sijainti. Tulokset on esitetty taulukossa 4.2 merkkipisteittäin.

Kokonaisuutena ajatellen luotettavuus on hyvä. Ongelmia tuotti testimateriaalissa ainoastaan alahuulen merkkipisteen tunnistaminen. Syynä ongelmiin oli se, että merkkipiste sekoitettiin vähän väliä vaaleisiin hampaisiin. Tämä voitaisiin yleisluontoisesti korjata huolellisemmalla, merkkipisteiden valon heijastavuutta korostavalla valaisulla ja mahdollisesti liikkeenseurantaparametrien, siis raja-arvofunktion parametrien, hienosäädöllä. Tässä tietyssä tapauksessa auttaisi myös se, että hampaita tummennettaisiin jollain tavoin kuvausten ajaksi. Tässä kuitenkin ongelmaksi muodostunee se, että väriaineet liukenevat syljen mukana kuvausten edetessä, jolloin hampaat alkavat jälleen tuottaa virheitä, tai että hampaat jäävät pitkäksi aikaa värjätyiksi.

Käytännössä lopputulokseen vaikuttivat lisäksi ongelmat etsittävien kohteiden havaitsemisessa, sillä valaisun ongelmien takia jotkin merkkipisteet (pääasiassa kulmakarvojen pisteet) havaittiin suuremmiksi kuin ne todellisuudessa olivat. Tästä seurasi sitä, että merkkipisteen sijainti havaittiin hetkellisesti väärin. Tätä on kuitenkin vaikea laskea

järjestelmän syyksi, sillä huolelliset kuvaushetken valmistelut ovat oleellinen osa järjestelmän luotettavaa toimintaa.

Taulukko 4.2. *Liikkeenmallinnuksen luotettavuus.*

Merkkipiste	Virheellisiä tunnistuksia	Virheitä / tunnistuksia (%)
Kulmakarva oikea A	0	0
Kulmakarva oikea B	0	0
Kulmakarva vasen A	0	0
Kulmakarva vasen B	0	0
Poski vasen	0	0
Poski oikea	0	0
Kontrollipiste	0	0
Suu ylähuuli	0	0
Suu alahuuli	69	6,90
Suu vasen	0	0
Suu oikea	0	0
Leuka	0	0
Yhteensä	69	0,58

Prosenttimääräisesti tulokset voivat ensisilmäykseltä vaikuttaa hyviltä, mutta animaatiossa jokainen virhe voi näkyä häiritsevästi. Tässä tapauksessa virheitä aiheuttaneen merkkipisteen prosenttimäärä on kuitenkin suhteellisen suuri. Kyseisellä virheprosentilla sekunnin aikana havaittaisiin keskimäärin peräti kaksi virhettä, mikä on aivan liikaa. Siksi järjestelmää tulisikin jatkossa kehittää niin, että pystyttäisiin saavuttamaan virheetön lopputulos hampaiden kirkkaudesta riippumatta.

4.4 Animaation luonnollisuus

Jos jätetään huomioimatta liikkeenseurannassa esiintyneet virheet, voidaan järjestelmän tuottaman animaation luonnollisuutta arvioida subjektiivisesti järjestelmän tavoitteita esitelleen kohdan mukaisesti. Ensimmäinen vaatimus oli, että liikkeet eivät saa näyttää konemaisen töksähtäviltä, mutta eivät liian sulaviltakaan. Tämän vaatimuksen toteutumisessa suuri painoarvo on kohinansuodatuksen voimakkuudella. Liian voimakas suodatus sulavoittaa liikettä liikaa ja kadottaa hienovaraisia liikkeitä. Liian heikko suodatus sen sijaan jättää liikkeeseen häiritsevää kohinaa. Erilaisia suodatusarvoja kokeilemalla saatiin lopulta tulos, joka ei ollut liian sulavaa eikä myöskään häiritsevän kohinaista. Töksähtelyä liikkeeseen aiheuttivat lähinnä virheelliset merkkipisteiden tunnistukset, joten merkkipistekoordinaattien suhteuttaminen kontrollipisteeseen selvästikin toimi kelvollisesti. Tulosta voisi kuitenkin parantaa, jos käytettäisiin kahta kontrollipistettä, jolloin pystyttäisiin suodattamaan myös tärähtelyn aiheuttamaa kameran kääntymistä ja

etäisyysheilahduksia kameran kuvaussuunnan akselilla. Joka tapauksessa järjestelmän tuottama animaatio täytti tämän ensimmäisen vaatimuksen.

Toinen vaatimus oli, että kasvot ja pää eivät saa vääntyä epäinhimillisiin asentoihin. Tämän saavuttamisesta vastasi merkkipistedatan normalisointi ja *ControlStick*-säätimen suorittama arvon rajaaminen, jotka kumpikin pyrkivät pitämään merkkipisteiden sijaintiarvot järkevissä rajoissa. Kiihtyvyyssensorin tuottama asentodata vaikutti testien perusteella kiitettävän vakaalta, eikä sen rajaamiselle koettu tarvetta. Järjestelmän testaamiseen käytetyn materiaalin perusteella tämäkin vaatimus tavoitettiin.

Viimeinen vaatimus oli, että animaation on pysyttävä ajassa ääniraidan kanssa riippumatta laitteiston kuormituksesta. Tätä ei työssä varsinaisesti toteutettu, mutta animaatiokirjaston toteutuksessa otettiin huomioon reaaliaikavaatimus ja mahdollisuus ääniraidan kanssa synkronointiin. Tämä vaatimus siis saavutettiin teoriassa.

Yleisellä tasolla arvioiden lopputulos on hyvä. Se ei tietenkään pysty kilpailemaan kalliiden kaupallisten järjestelmien kanssa, mutta indie-kehittäjille sen laatu riittänee hyvin. Kasvonliikkeiden seuranta on riittävän tarkkaa ja reagoi kiitettävästi hienovaraisiinkin liikkeisiin. Päänliikkeiden kaappaus, joka jouduttiin jättämään laitteistorajoitteiden takia täysin kiihtyvyyssensorin varaan, toimii kelvollisesti, mutta lopputulosta olisi saanut parannettua rutkasti, jos gyroskooppia olisi pystytty hyödyntämään.

Järjestelmän kanssa on kuitenkin oltava tarkkana, ettei yritä käyttää sitä liian realistisiten hahmomallien animoimiseen, sillä tällöin havaitaan väkisinkin oudon laakson ilmiö (engl. uncanny valley). Yksinkertaisuudessaan ilmiö tarkoittaa sitä, että realistisuuden kasvaessa ihminen kokee hahmon luonnollisemmaksi, mutta realistisuuden lähestyessä todellisuutta luonnollisuuden tuntuisuus putoaa jyrkästi ja lopputulos alkaa vaikuttaa oudolta, kunnes todellisuus lopulta saavutetaan [7]. Toisin sanoen epärealistinen liike realistisen näköisellä hahmolla ei tuota hyvää lopputulosta.

4.5 Jatkokehitys

Lopuksi käsitellään lyhyesti ajatuksia järjestelmän jatkokehitykseen. Näistä nykyisiin ominaisuuksiin liittyviä ovat:

- Kypärätelineen keventäminen, jotta se ei väsyttäisi näyttelijän niskoja.
- Gyroskoopin ja kompassin sisällyttäminen päänliikkeiden kaappaukseen, jotta liikkeet pystyttäisiin kaappaamaan tarkemmin.
- Paremman sensori- ja liikkeenseurantadatan synkronointimenetelmän kehittäminen, jotta datavirrat ovat aina ajassa keskenään jopa puhelimesta riippumatta.
- Käyttöliittymätuen lisääminen *ControlStick*-säätimiin asetetuille luiden skaalausarvoille, koska nykyisellään arvot on kovakoodattu käsittelysovellukseen.
- Merkkipistekonfiguraation määrittelyn siirtäminen ulkoiseen konfiguraatiotiedostoon, koska konfiguraatio ja merkkipisteiden yksilöiminen on nykyisellään kovakoodattu liikkeenseurantasovellukseen.

- Liikkeenseurannan suorituskyvyn ja luotettavuuden kehittäminen.

Suorituskyvyn kehittämisessä huomio kannattaa edellä mainittujen mittaustulosten perusteella kiinnittää kuvaa analysoivan etsimisvaiheen nopeuttamiseen, esimerkiksi lisäämällä tuki rinnakkaisuudelle. Hyvin skaalautuvan rinnakkaisuuden toteuttamisen pitäisi olla suhteellisen yksinkertaista, sillä yksittäisen kuvan käsittely ei riipu muiden kuvien käsittelystä; riittää, että kuvat luetaan videotiedostosta järjestyksessä. Suorituskyvyn lisäämisestä ei kuitenkaan ole hyötyä, jos liikkeenseurannan luotettavuus on heikko.

Järjestelmän parantamiseksi tulisi kehittää myös uusia ominaisuuksia. Lopputuloksen kannalta tärkeää olisi pystyä kaappaamaan silmien ja silmäluomien liike. Tämä parantaisi animaation luonnollisuutta merkittävästi, sillä katse hakeutuu usein ihmiskasvoja katsoessa juuri silmiin. Silmänliikkeiden tunnistamisessa on kuitenkin omat haasteensa, sillä merkkipisteitä ei tällöin pystytä käyttämään.

Käytettävyyden parantamiseksi kaappaussovellukseen tulisi toteuttaa tiedonsiirtotoiminto, jolla videokuvan ja sensoridatan saisi langattomasti siirrettyä älypuhelimelta tietokoneelle. Siirron ei tarvitse olla reaaliaikaista kuvauksen kanssa, vaan riittää, että kuvaamisen päätyttyä kaikki tarpeelliset tiedostot voi siirtää helposti ja nopeasti tietokoneelle.

Lopputuloksen kannalta tärkeää olisi myös hienosäädön toteuttaminen. Hienosäädön avulla voitaisiin varmistua siitä, että vaikka liikkeenseuranta tekisikin silloin tällöin virheitä, ne pystyttäisiin käyttäjän toimesta korjaamaan helposti. Näin ongelmallista lähdemateriaalia ei välttämättä tarvitsisi heittää roskiin ja kuvata uudelleen, mikä säästäisi aikaa usealta tuotannon osapuolelta.

5 YHTEENVETO

Järjestelmän tuottama animaatio pystyy paikoitellen yllättävänkin hienovaraisiin eleisiin, mikä osoittaa selkeästi, että nykyaikainen älypuhelin, joka pystyy tallentamaan kuvaa vähintään 1280x720 kuvapisteen resoluutiolla ja 30 kuvan sekuntinopeudella, riittää kasvoanimaation liikkeenkaappaukseen hyvin indie-pelikehittäjien mittapuulla. Tätä tarkempi kuva ja tiheämpi näytetaajuus mahdollistavat hienovaraisemman ja täsmällisemmän animaation sekä pienempien ja siten useampien merkkipisteiden käytön, mutta indie-kehittäjille nämä parannukset eivät ole läheskään välttämättömiä.

Videokuvaan perustuvan liikkeenkaappauksen toteuttaminen on käytännössä erittäin helppoa ilmaisen ja avoimen OpenCV-tietokonenäkökirjaston avulla. Kirjaston tarjoamien monipuolisten kuvankäsittelytoimintojen avulla pystyy helposti tuomaan kiinnostavat kohteet esille huolella kuvatusta materiaalista. Kuvan analysoiminen onnistuu myös vaivatta, ja OpenCV:n tarjoamilla toiminnoilla on helppoa saada tietoa esimerkiksi seurattavien kohteiden sijainneista ja pinta-aloista. Näin ollen liikkeenkaappauksen toteuttamiseksi tarvitsee keskittyä pääasiassa mallintamaan seurattavien kohteiden liikettä, jotta samat kohteet pystytään löytämään seuraavistakin kuvista. Kun videomateriaali on huolella valaistu, suoriutuu OpenCV sen käsittelystä ja analysoinnista tehokkaasti ja luotettavasti. Järjestelmään toteutettu liikkeenseurantakin toimi testien perusteella hyvin, vaikkakin liikkeenmallinnus sekoitti paikoitellen lähekkäin olevia, kiinnostaviksi kohteiksi havaittuja kohteita keskenään. Suorituskyvyltään järjestelmä on edelliseen prototyyppiin verrattuna erittäin hyvä.

Päänliikkeiden kaappaaminen oli suoraviivaista, kun käytettävissä olevasta älypuhelimesta pystyttiin hyödyntämään ainoastaan kiihtyvyyssensoria laitteen asennon määrittämiseen. Pelkän kiihtyvyyssensorin perusteella tehty liikkeenkaappaus tuotti kelvollisen lopputuloksen, mutta eräs oleellisimmista jatkokehityksen kohteista olisi laajentaa järjestelmää hyödyntämään myös älypuhelimen gyroskooppia ja tarvittaessa kompassiakin.

Kaiken kaikkiaan tässä diplomityössä toteutettu prototyyppi antaa vahvoja todisteita älypuhelimeen pohjautuvan liikkeenkaappauksen toimivuudesta. Nykyisellään järjestelmä ei kuitenkaan täysin sovellu varsinaiseen tuotantokäyttöön, vaan se vaatisi kipeästi muutamia oleellisia parannuksia luotettavuuteen, käytettävyyteen ja animaation luonnollisuuteen. Kehitettävää siis riittäisi vielä paljon, jotta järjestelmän täysi potentiaali saataisiin valjastettua käyttöön. Nykyaikaisten älypuhelimien kamerat ovat niin laadukkaita, että mahdollisuus monipuolisen ja luonnollisen kasvoanimaation kaappaamiseksi on jo olemassa. Tarvitaan vain kehittäjiä, jotka tarttuvat haasteeseen.

LÄHTEET

[1] De Aguiar, E., Theobalt, C., Stoll, C. & Seidel, H.-P. Marker-less 3D Feature Tracking for Mesh-based Human Motion Capture. Proceedings of the 2nd Conference on Human Motion: Understanding, Modeling, Capture and Animation, Rio de Janeiro, October 20, 2007. Berlin 2007, Springer-Verlag. pp. 1–15.

[2] Ayub, S., Bahraminisaab, A. & Honary, B. A Sensor Fusion Method for Smart Phone Orientation Estimation. Proceedings of the 13th Annual Post Graduate Symposium on the Convergence of Telecommunications, Networking and Broadcasting, Liverpool, June 25–26, 2012. Liverpool 2012, Liverpool John Moores University.

[3] Bradski, G. & Kaehler, A. Learning OpenCV: Computer Vision with the OpenCV Library. Sebastopol 2008, O'Reilly Media. 555 pp.

[4] Broughall, N. How L.A. Noire Conquered the Uncanny Valley with a Tech Called MotionScan [WWW]. 17.12.2010 [Viitattu 11.2.2013]. Saatavissa: http://gizmodo.com/5714436/how-la-noire-conquered-the-uncanny-valley-with-a-tech-called-motionscan.

[5] CaptiveMotion [WWW]. [Viitattu 24.2.2013]. Saatavissa: http://www.captivemotion.com/.

[6] Cascade Classification – OpenCV 2.4.5.0 Documentation [WWW]. [Viitattu 5.5.2013]. Saatavissa: http://docs.opencv.org/modules/objdetect/doc/cascade_classification.html.

[7] Case, A. Uncanny Valley – Cyborg Anthropology [WWW]. 7.8.2012 [Viitattu 27.4.2013]. Saatavissa: http://cyborganthropology.com/Uncanny_Valley.

[8] Corazza, S., Muendermann, L., Chaudhari, A., Demattio, T., Cobelli, C. & Andriacchi, T. A Markerless Motion Capture System to Study Musculoskeletal Biomechanics: Visual Hull and Simulated Annealing Approach. Annals of Biomedical Engineering 34(2006)6, pp. 1019–1029.

[9] Corazza, S., Mündermann, L. & Andriacchi, T. A Framework for the Functional Identification of Joint Centers Using Markerless Motion Capture, Validation for the Hip Joint. Journal of Biomechanics 40(2007)15, pp. 3510–3515.

[10] Crawford, S. HowStuffWorks "How Microsoft Kinect Works" [WWW]. [Viitattu 24.2.2013]. Saatavissa: http://electronics.howstuffworks.com/microsoft-kinect.htm.

[11] Creating Animations for the Unreal Engine [WWW]. [Viitattu 3.3.2013]. Saatavissa: http://udn.epicgames.com/Three/CreatingAnimations.html.

[12] Dashwood, T. Everything You Ever Wanted to Know about HFR and The Hobbit [WWW]. 14.12.2012 [Viitattu 21.2.2013]. Saatavissa: http://www.dashwood3d.com/blog/high-frame-rate-hfr-and-the-hobbit/.

[13] Elmenreich, W. An Introduction to Sensor Fusion. Austria 2002, Institut für Technische Informatik, Vienna University of Technology.

[14] Esfandyari, J., De Nuccio, R. & Xu, G. Introduction to MEMS Gyroscopes [WWW]. 15.11.2010 [Viitattu 28.2.2013]. Saatavissa: http://www.electroiq.com/articles/stm/2010/11/introduction-to-mems-gyroscopes.html.

[15] Euston, M., Coote, P., Mahony, R., Kim, J. & Hamel, T. A Complementary Filter for Attitude Estimation of a Fixed-Wing UAV. IEEE/RSJ International Conference on Intelligent Robots and Systems, Nice, September 22–26, 2008. New York 2008, IEEE Press. pp. 340–345.

[16] Event Handling Guide for iOS: Motion Events [WWW]. 28.1.2013 [Viitattu 1.3.2013]. Saatavissa: http://developer.apple.com/library/ios/#documentation/EventHandling/Conceptual/EventHandlingiPhoneOS/motion_event_basics/motion_event_basics.html.

[17] Facemotion – Facial Motion Capture [WWW]. [Viitattu 24.2.2013]. Saatavissa: http://www.easycapstudio.com/.

[18] Faceshift [WWW]. [Viitattu 24.2.2013]. Saatavissa: http://www.faceshift.com/.

[19] FFmpeg [WWW]. [Viitattu 9.2.2013]. Saatavissa: http://www.ffmpeg.org/.

[20] First Motion-Capture Animation in a Video Game [WWW]. [Viitattu 23.2.2013]. Saatavissa: http://www.guinnessworldrecords.com/records-6000/first-motion-capture-animation-in-a-video-game/.

[21] Fukunaga, K. & Hostetler, L. The Estimation of the Gradient of a Density Function, with Applications in Pattern Recognition. IEEE Transactions on Information Theory 21(1975)1, pp. 32–40.

[22] Giachetti, A., Campani, M. & Torre, V. The Use of Optical Flow for Road Navigation. IEEE Transactions on Robotics and Automation 14(1998)1, pp. 34–48.

[23] Hounsell, T. Avoiding Reflection – How to Use InteropServices with the WP7 SDK [WWW]. 15.11.2010 [Viitattu 9.2.2013]. Saatavissa: http://thounsell.co.uk/windows-phone/avoiding-reflection-adding-the-interopservices-library-to-the-wp7-sdk/.

[24] HSV Color Solid Cylinder Alpha Lowgamma.png [WWW]. [Viitattu 25.2.2013]. Saatavissa: http://en.wikipedia.org/wiki/File:HSV_color_solid_cylinder_alpha_lowgamma.png.

[25] Irrlicht Engine – A Free Open Source 3D Engine [WWW]. [Viitattu 20.4.2013]. Saatavissa: http://irrlicht.sourceforge.net/.

[26] Irrlicht Lime [WWW]. [Viitattu 20.4.2013]. Saatavissa: http://sourceforge.net/projects/irrlichtlime/.

[27] Isard, M. & Blake, A. Condensation – Conditional Density Propagation for Visual Tracking. International Journal of Computer Vision 29(1998)1, pp. 5–28.

[28] Itseez: About Us [WWW]. [Viitattu 9.2.2013]. Saatavissa: http://itseez.com/index.php?page=about_us.

[29] Kaimakis, P. & Lasenby, J. Markerless Motion Capture with Single and Multiple Cameras. Proceedings of IEEE International Conference on Image Processing, Singapore, October 24–27, 2004. New York 2004, IEEE Press. pp. 2607–2610.

[30] Kalal, Z. TLD [WWW]. 2011 [Viitattu 21.2.2013]. Saatavissa: http://info.ee.surrey.ac.uk/Personal/Z.Kalal/tld.html.

[31] Kalal, Z., Mikolajczyk, K. & Matas, J. Face-TLD: Tracking-Learning-Detection Applied to Faces. 17th IEEE International Conference on Image Processing, Hong Kong, September 26–29, 2010. New York 2010, IEEE Press. pp. 3789–3792.

[32] Kalman, R.E. A New Approach to Linear Filtering and Prediction Problems. Transactions of the ASME – Journal of Basic Engineering D(1960)82, pp. 35–45.

[33] Kong, J. & Bao, O. Tech Focus: L.A. Noire's MotionScan Animation [WWW]. 1.6.2011 [Viitattu 11.2.2013]. Saatavissa: http://www.gamesindustry.biz/articles/digitalfoundry-tech-focus-depth-analysis-interview.

[34] L.A. Noire Trailer – Behind the Scenes [WWW]. 16.12.2010 [Viitattu 26.2.2013]. Saatavissa: http://www.youtube.com/watch?v=ZY7RYCsE9KQ.

[35] LaBarge, D.M. James "Purple" Hampton on Highlander. Jaguar Explorer 1(1997)2.

[36] Linear Interpolation. Encyclopedia of Mathematics [WWW]. [Viitattu 5.5.2013]. Saatavissa: http://www.encyclopediaofmath.org/index.php?title=Linear_interpolation&oldid=27068.

[37] Liverman, M. The Animator's Motion Capture Guide: Organizing, Managing, Editing. Hingham 2004, Charles River Media. 336 pp.

[38] Maskarad: Automatic Markerless Facial Performance Capture Software for Windows [WWW]. [Viitattu 24.2.2013]. Saatavissa: http://www.di-o-matic.com/products/Software/Maskarad/#page=overview.

[39] Meike, R. Location, Location, Location (Accelerometer) [WWW]. 10.11.2008 [Viitattu 20.2.2013]. Saatavissa: https://blogs.oracle.com/roger/entry/location_location_location_accelerometer.

[40] Menache, A. Understanding Motion Capture for Computer Animation. Second edition. Burlington 2011, Morgan Kaufmann. 267 pp.

[41] Morphology Fundamentals: Dilation and Erosion [WWW]. [Viitattu 26.2.2013]. Saatavissa: http://www.mathworks.se/help/images/morphology-fundamentals-dilation-and-erosion.html.

[42] Motion Capture Overview [WWW]. [Viitattu 23.2.2013]. Saatavissa: http://lightcrafttech.com/support/doc/introduction/motion-capture/.

[43] Muendermann, L., Corazza, S. & Andriacchi, T. The Evolution of Methods for the Capture of Human Movement Leading to Markerless Motion Capture for Biomechanical Applications. Journal of NeuroEngineering and Rehabilitation 3(2006)1.

[44] Mündermann, L., Corazza, S. & Andriacchi, T. Accurately Measuring Human Movement Using Articulated ICP with Soft-Joint Constraints and a Repository of Articulated Models. IEEE Conference on Computer Vision and Pattern Recognition, Minneapolis, June 17–22, 2007. New York 2007, IEEE Press. pp. 1–6.

[45] OpenCV: About [WWW]. [Viitattu 9.2.2013]. Saatavissa: http://opencv.org/about.html.

[46] OptiTrack – Expression – Facial Motion Capture Software [WWW]. [Viitattu 24.2.2013]. Saatavissa: http://www.naturalpoint.com/optitrack/products/expression/.

[47] Page, N. NTSC, PAL & Interlace Explained [WWW]. [Viitattu 21.2.2013]. Saatavissa: http://nickyguides.digital-digest.com/interlace.htm.

[48] Pedley, M. Tilt Sensing Using a Three-Axis Accelerometer. 2013, Freescale Semiconductor. 22 pp.

[49] Rabin, S. Introduction to Game Development. Second edition. 2009, Charles River Media. 1008 pp.

[50] Rosenhahn, B., Brox, T., Kersting, U.G., Smith, A.W., Gurney, J.K. & Klette, R. A System for Marker-less Motion Capture. Künstliche Intelligenz. 2006. pp. 45–51.

[51] Russ, J.C. The Image Processing Handbook. Sixth edition. Boca Raton 2011, CRC Press. 885 pp.

[52] Schall, G., Wagner, D., Reitmayr, G., Taichmann, E., Wieser, M., Schmalstieg, D. & Hofmann-Wellenhof, B. Global Pose Estimation Using Multi-Sensor Fusion for Outdoor Augmented Reality. 8th IEEE International Symposium on Mixed and Augmented Reality, Orlando, October 19–22, 2009. New York 2009, IEEE Press. pp. 153 –162.

[53] Seitz, S.M., Curless, B., Diebel, J., Scharstein, D. & Szeliski, R. A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithms. IEEE Computer Society Conference on Computer Vision and Pattern Recognition, New York, June 17–22, 2006. New York 2006, IEEE Press. pp. 519–528.

[54] Sensors for Windows Phone [WWW]. 1.2.2013 [Viitattu 1.3.2013]. Saatavissa: http://msdn.microsoft.com/en-us/library/windowsphone/develop/hh202968(v=vs.105).aspx.

[55] Sensors Overview [WWW]. [Viitattu 1.3.2013]. Saatavissa: http://developer.android.com/guide/topics/sensors/sensors_overview.html.

[56] Seymour, M. Art of Optical Flow [WWW]. 28.2.2006 [Viitattu 23.2.2013]. Saatavissa: http://www.fxguide.com/featured/art_of_optical_flow/.

[57] Seymour, M. Art of Tracking Part 1: History of Tracking [WWW]. 24.8.2004 [Viitattu 20.2.2013]. Saatavissa: http://www.fxguide.com/featured/art_of_tracking_part_1_history_of_tracking/.

[58] Seymour, M. Far Cry 3: Digital Survivors [WWW]. 17.12.2012 [Viitattu 24.2.2013]. Saatavissa: http://www.fxguide.com/featured/far-cry-3-digital-survivors/.

[59] Singer, G. The Two Towers: Face to Face with Gollum [WWW]. 17.3.2003 [Viitattu 23.2.2013]. Saatavissa: http://www.awn.com/articles/technology/two-towers-face-face-gollum.

[60] Soule, K. What is YUV? [WWW]. 26.6.2010 [Viitattu 25.2.2013]. Saatavissa: http://blogs.adobe.com/VideoRoad/2010/06/what_is_yuv.html.

[61] Stanford Markerless Motion Capture Project [WWW]. [Viitattu 23.2.2013]. Saatavissa: https://ccrma.stanford.edu/~stefanoc/Markerless/Markerless.html.

[62] Strategy Pattern [WWW]. [Viitattu 27.4.2013]. Saatavissa: http://www.oodesign.com/strategy-pattern.html.

[63] Sturman, D.J. A Brief History of Motion Capture for Computer Character Animation [WWW]. 13.3.1999 [Viitattu 20.2.2013]. Saatavissa: http://www.siggraph.org/education/materials/HyperGraph/animation/character_animation/motion_capture/history1.htm.

[64] The BSD 3-Clause License [WWW]. [Viitattu 9.2.2013]. Saatavissa: http://opensource.org/licenses/BSD-3-Clause.

[65] Toyama, K., Krumm, J., Brumitt, B. & Meyers, B. Wallflower: Principles and Practice of Background Maintenance. The Proceedings of the Seventh IEEE International Conference on Computer Vision, Kerkyra, September 20–27, 1999. New York 1999, IEEE Press. pp. 255–261.

[66] Weisstein, E.W. Covariance. From MathWorld – A Wolfram Web Resource [WWW]. [Viitattu 27.2.2013]. Saatavissa: http://mathworld.wolfram.com/Covariance.html.

[67] Weisstein, E.W. Fast Fourier Transform. From MathWorld – A Wolfram Web Resource [WWW]. [Viitattu 23.2.2013]. Saatavissa: http://mathworld.wolfram.com/FastFourierTransform.html.

[68] Weisstein, E.W. Normal Distribution. From MathWorld – A Wolfram Web Resource [WWW]. [Viitattu 27.2.2013]. Saatavissa: http://mathworld.wolfram.com/NormalDistribution.html.

[69] Weisstein, E.W. Spherical Coordinates. From MathWorld – A Wolfram Web Resource [WWW]. [Viitattu 1.3.2013]. Saatavissa: http://mathworld.wolfram.com/SphericalCoordinates.html.

[70] Wilson, P.I. & Fernandez, J. Facial Feature Detection Using Haar Classifiers. Journal of Computing Sciences in Colleges 21(2006)4, pp. 127–133.

[71] Viola, P. & Jones, M. Rapid Object Detection Using a Boosted Cascade of Simple Features. Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Kauai, December 8–14, 2001. New York 2001, IEEE Press. pp. I–511–518.