TAMPEREEN TEKNILLINEN YLIOPISTO

JUSSI NISKANEN

KONENÄKÖÖN PERUSTUVA ELEOHJAUS MOBIILILAITTEELLA

Diplomityö

Tarkastaja: Tommi Mikkonen
Aihe, tarkastaja ja kieli hyväksytty
Tieto- ja sähkötekniikan tiedekunnan
tiedekuntaneuvoston
kokouksessa 9.11.2011

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Tietotekniikan koulutusohjelma
NISKANEN, JUSSI: Konenäköön perustuva eleohjaus mobiililaitteella
Diplomityö, 54 sivua
Toukokuu 2012
Pääaine: Ohjelmistotuotanto
Tarkastajat: Professori Tommi Mikkonen, DI Jaakko Kuivanen
Avainsanat: eleohjaus, eleentunnistus, konenäkö, OpenCV

Eleiden käyttö on keskeinen osa ihmisten välistä kommunikointia. Onkin luonnollista, että eleiden käyttöä myös tietokoneiden ohjaamisessa on tutkittu jo 1960-luvulta lähtien. Kuitenkin vasta videopeliyhtiö Nintendon vuonna 2006 julkistama pelikonsoli Wii ja tietotekniikkayhtiö Applen vuonna 2007 julkistama iPhone-älypuhelin toivat eleohjatut sovellukset valtavirran tietoisuuteen.

Mobiililaitteissa kosketuseleohjaus on nopeasti kasvanut perinteisen näppäinohjauksen rinnalle, ja muut mobiililaitevalmistajat ovat myös ottaneet iPhonen kosketuselesanastoa laajalti käyttöönsä. Kosketusnäytön lisäksi mobiililaitteissa on runsaasti komponentteja, jotka mahdollistavat monipuolisten eleohjattujen sovellusten tekemisen.

Mobiililaitteissa pääasiallinen eleohjausmenetelmä on kosketusnäyttö ja käytännössä kaikki laitevalmistajat tarjoavat valmiin ohjelmointirajapinnan kosketuselesovellusten toteuttamista varten. Jos ohjelmoija haluaa kuitenkin tehdä esimerkiksi laitteen kameroita hyödyntävän konenäön avulla ohjattavan sovelluksen, on eleentunnistus toteutettava itse. Tämä voi johtaa ylimääräiseen työhön.

Tässä diplomityössä tarkastellaan eleohjauksen suunnittelua ja erilaisia eleohjausmenetelmiä mobiililaitteella. Teknisenä kontribuutiona toteutettiin yleiskäyttöinen konenäköön perustuva eleohjauskirjasto. Kirjaston tunnistamissa eleissä pyrittiin sovittamaan kosketusnäytöistä tutuiksi tulleita eleitä. Kirjaston toteuttamisessa käytettiin hyödyksi OpenCV-konenäkökirjastoa ja Qt-ohjelmistokehystä.

Eleohjauskirjaston toteuttaminen osoittautui teknisesti mahdolliseksi. Erityisesti mobiililaitteilla käytännön rajoitteiksi kuitenkin osoittautuivat prosessointitehon puute ja eletunnistuksen vaatima suuri virrankulutus. Poikkialustalliset toteutustyökalut kuitenkin mahdollistivat työn täysipainoisen kehittämisen PC-tietokoneella työpöytäympäristössä verkkokameran avulla. Kirjastoa voi myöhemmin paremmin käyttää mobiililaitteilla kun niiden prosessointitehot ja akkujen kapasiteetit kasvavat.

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Information Technology
NISKANEN, JUSSI: Computer Vision Based Gesture Control on a Mobile Device
Master's Thesis, 54 pages
May 2012
Major: Software engineering
Examiners: Professor Tommi Mikkonen, M. Sc. Jaakko Kuivanen
Keywords: gesture control, gesture recognition, computer vision, OpenCV

Gesturing is a natural and very important part of our daily communication with other humans. It is natural that using gestures to control computers has received academic interest since 1960s. However, gesture controlled applications were pushed to the mainstream only after video games corporation Nintendo had published the gaming console Wii in 2006 and consumer electronics corporation Apple had published the iPhone in 2007.

On mobile devices gesture control has quickly gained footstep besides the more conventional button oriented control scheme. Other mobile manufacturers have also adopted the touch gesture vocabulary from iPhone. Besides the touch display, mobile devices also incorporate several components that enable creating rich gesture controlled applications.

The touch screen still remains the primary gesture control method for mobile devices. All mobile manufacturers also provide mature and ready-to-use application programming interfaces for creating gesture controlled applications. However, should the programmer want to utilize computer vision in their gesture control scheme, they need to implement the gesture recognition themselves. This could cause lots of extra work.

This thesis studies the design of a gesture control interface and different gesture control methods available on a modern mobile device. A general purpose computer vision based gesture control library was implemented as a technical contribution. The gestures recognized by the library were adopted from the gesture vocabulary familiar from touch interfaces. The implementation of the library utilizes toolkits from the OpenCV computer vision library and Qt software framework.

Implementing the computer vision based library for a mobile device turned out to be technically feasible. However, lack of processing power and increased battery consumption, caused by the gesture recognition, limit the usage of the library on mobile devices. Cross platform tools enabled full fledged development work with a desktop PC and a web camera. Later on, after the overall processing power and battery capacity have increased, the library can be better utilized on a mobile device.

ALKUSANAT

Tein tämän diplomityön työskennellessäni Sasken Finland Oy:llä. Istuimme työn ohjanneen DI Jaakko Kuivasen kanssa kesällä 2011 pohtimaan sopivia aiheita, ja jostain syntyi ajatus PowerPoint -tyylisen esitysohjelman ohjaamisesta älypuhelimen kameran ja käsieleiden avulla. Ajatus aiheutti välittömästi positiivistä hytinää mielessä. Toteuttamista varten pitäisi opetella runsaasti uusia mielenkiintoisia asioita ja yhdistää niitä olemassa olevaan osaamiseeni. Seuraavan reilun puolen vuoden aikana tuo ajatus muovautui täksi työksi.

Haluan kiittää erityisesti Jaakkoa, jonka aktiivinen ideointi, organisointi ja kannustus kirjoitusprosessin aikana auttoivat työn ajallaan valmiiksi. Myös työn tarkastaja professori Tommi Mikkonen ansaitsee lämpimät kiitokseni. Hänen innostuksensa esitellessäni työn aihetta tarttui minuunkin.

Tampereella 30.3.2012

Jussi Niskanen
jussi.o.niskanen@gmail.com

SISÄLLYS

1. JOHDANTO

Eleiden käyttäminen on luonnollinen osa jokapäiväistä elämäämme. Esimerkiksi rytmitämme puhettamme huitomalla käsiämme kuin huomaamatta, osoitamme sormella kulkuohjeiden kysyjälle oikeaa suuntaa, ja levitämme käsiämme kun haluamme osoittaa pyydystetyn kalan suuruuden. Eleet ovat siis tärkeä tapa kommunikoida ja vaikuttaa ympäristömme kanssa. Tästä johtuen ei ole mitenkään ihmeellistä, että eleiden käyttäminen ohjelmistojen syötevälineenä on melko tutkittu aihepiiri. Viime vuosina eleohjaus on noussut suosituksi välineeksi myös kaupallisten sovellusten ohjausmenetelmänä erityisesti peliteollisuudessa ja kosketusnäytöllisissä matkapuhelimissa. Tästä huolimatta eleohjaus on vasta niin sanotusti lyömässä itseään läpi valtavirtaan. Eleohjauksen parista löytyy paljon mielenkiintoista tutkittavaa ja sen avulla voi toteuttaa runsaasti uudenlaisia sovelluksia.

Eleiden käyttö tietokoneiden ohjaamisessa ei ole kuitenkaan uusi ajatus; asiaa on tutkittu jo 60-luvulla! Käytännössä kuitenkin Applen iPhone-julkaisu vuonna 2007 esitteli kosketusnäytöllä toimivat eleet kuluttajalaitteisiin. Myös muut mobiililaitevalmistajat ovat sittemmin ottaneet käyttöön Applen kosketuselesanastoa omissa laitteissaan, joten se on alkanut tulla tunnetuksi laajalle yleisölle. Eräänä motivaationa tämän työn tekemiselle on ollut sovittaa tuota kosketuselesanastoa konenäköön perustuvaan eleohjaukseen.

Tässä diplomityössä tarkastellaan erilaisia eleohjausmenetelmiä yleisesti sekä erityisesti mobiililaitteissa. Lisäksi toteutetaan konenäköön perustuva eleohjauskirjasto, jota voi käyttää eleohjaukseen perustuvien sovellusten luomiseen. Tätä kirjastoa käytetään kursorisesti esitysohjelman ohjaamiseen matkapuhelimella. Tarkasteltavat ohjauseleet perustuvat pääosin iPhonen myötä valtavirtaan nousseisiin eleisiin ja työssä pyritään osittain myös selvittämään miten kyseisiä kosketusnäytölle tarkoittavia eleitä voisi hyödyntää konenäön avulla.

Eleiden käyttäminen mahdollistaa useita uusia, mielenkiintoisia ja luontevia ohjaustapoja, mutta eleohjausjärjestelmän rakentaminen alusta alkaen on haastavaa. Tätä kirjoittaessa kaikkien käytettävissä olevia valmiita toteutuksia ei ole paljon tarjolla. Tämä voi johtaa ylimääräiseen työhön tai kasvaneisiin kehityskuluihin kun järjestelmiä toteutetaan alusta alkaen ilman mahdollisuutta hyödyntää jo toteutettuja eleohjauskirjastoja.

Eleohjauksen toteuttamisessa on lukuisia tapoja, mutta tässä työssä painopiste on konenäköpohjaisen eleohjauksen tarkastelussa ja ellei muuta erikseen mainita tai asia ei kontekstista käy ilmi, eleohjauksella tarkoitetaan juuri konenäköpohjaista eleohjausta, jossa seurataan käyttäjän liikkeitä syötteinä.

Luvussa 2 esitellään eleohjauksen maailmaa, eräs prosessi hyvän elesanaston löytämiseksi ja eri menetelmiä toteuttaa eleohjaus. Luvussa 3 tarkastellaan eleohjausta nimenomaan mobiililaitteissa. Luku 4 käsittelee tyypillisestä konenäköpohjaisesta eleohjausjärjestelmästä löytyviä osia. Luvussa 5 esitellään työssä toteutettu järjestelmä, minkä onnistumista arvioidaan luvussa 6. Luku 7 esittää yhteenvedon työstä.

2. ELEOHJAUS

Eleohjausta hyödyntävissä käyttöliittymissä yleensä ottaen käyttäjä antaa syötteen vartalonsa tai sen osien avulla, kuten liikuttamalla raajojaan, koko kehoaan, silmiään tai vaikkapa käyttämällä ääntään. Tällaisia tekniikoita on pitkään käytetty lähinnä erityisryhmien käyttöä avustavina välineinä, mutta viime vuosina elekäyttöliittymiä on otettu käyttöön yhä enenevissä määrin myös tavallisten käyttäjien interaktiivisissa sovelluksissa perinteisen WIMP[1] -mallin ohella tai korvaajana. [10]

Eletunnistuksen tutkimus ja käyttö mitä moninaisimpiin tarkoituksiin ei ole mitenkään uusi asia. Varhaisimpia eleohjausta hyödyntäviä laitteita oli Ivan Sutherlandin suunnittelema Sketchpad vuodelta 1963. Sketchpadilla saattoi muokata olioita – esimerkiksi ottamalla kiinni, siirtämällä tai muuttamalla kokoa – käyttäen valokynää. Se oli käytännössä myös ensimmäinen graafista käyttöliittymää käyttävä sovellus. Populaarikulttuurissa puolestaan tietokonetta ohjattiin eleohjauksella jo vuonna 1965, kun Spock vaihtoi näytöllä näkyvää kuvaa pyyhkäisyeleellä Star Trekin pilottijaksossa The Cage[2]. [37]

2.1 Eleohjauksen suunnittelu

Tämä alakohta perustuu pitkälti Nielsenin et al. [39] kehittämään prosessiin intuitiivisten ja ergonomisten eleohjausta käyttävien käyttöliittymien suunnitteluun.

Aivan ensimmäisenä pitäisi ottaa huomioon, että eleohjauskäyttöliittymää ei ole useinkaan järkevää suunnitella korvaamaan täysin jo olemassa olevia WIMP-pohjaisia järjestelmiä. Sen sijaan eleohjauksen pitäisi olla perinteisen käyttöliittymän rinnalla tuomassa omilla vahvuuksillaan lisäarvoa käyttäjälle. Joissakin tapauksissa eleohjaus ei ole lainkaan hyvä ajatus. Esimerkiksi tarkkaa ohjausta vaativissa sovelluksissa kuten useiden suunnittelijoiden käyttämissä CAD -ohjelmissa[3] perinteinen hiiriohjaus on selvästi eleohjausta parempi. Sen sijaan epätarkempia linjoja suova esteettinen suunnittelu voisi olla mukavampaa kuvailemalla muotoja vapailla käsiliikkeillä kuin hiirellä muokkaamalla.

[1] Windows, Icons, Menus, Pointing device
[2] http://www.imdb.com/title/tt0059753/
[3] Computer Aided Design

Eleohjaussuunnittelun tärkeimpiä tehtäviä on valita käytettävä elesanasto, jolla tarkoitetaan käyttöliittymässä käytettävien eleiden joukkoa ja niiden nimiä. Nimeäminen voidaan tehdä joko kuvailemalla eleiden fyysistä liikettä (esimerkiksi “kaari”, “ympyrä” tai “neliö”) tai antamalla eleille semanttisia nimiä, jotka kuvaavat eleen tarkoitusta tai sitä mitä sillä koitetaan viestiä. Semanttiset nimet voivat olla melko kulttuurisidonnaisia. Tästä Nielsen et al. [39] antavat esimerkkinä peukalon ja etusormen muodostaman ympyrän; länsimaissa tämä käsitetään yleensä “O.K.”, mutta Japanissa ele tarkoittaa “rahaa”. Esimerkkejä tyypillisistä koneiden kanssa vuorovaikuttamisessa käytettävien eleiden nimistä ovat vaikkapa kokoa tai liikettä osoittavat “se on näin iso”, “pistä tuo tuohon” ja “kahvan kääntö”.

Elesanaston valinnassa voidaan käyttää joko kone- tai ihmiskeskistä lähestymistapaa. Konekeskisessä elesanastossa painotetaan helppoa ja tarkkaa tunnistettavuutta käytettävyyden ja (käyttäjän) mukavuuden kustannuksella. Esimerkkinä voisi olla vaikka käyttöliittymä, joka tarkkaillaan käyttäjän paikallaan olevaa kättä ja jossa tehdään asioita riippuen käyttäjän näyttämien sormien lukumäärästä. Näin on helppo tunnistaa nopeasti kuusi eri elettä (0-5 sormea), mutta kyseiset eleet ovat ergonomisesti raskaita. Kaikki käyttäjät eivät myös välttämättä edes kykene toistamaan kaikkia kyseisiä eleitä. Tällainen elesanasto ei luultavasti myöskään ole kosketuksissa varsinaisen sovellusalueen käsitteiden kanssa, joten sen käyttämisen oppiminen on vaikeaa.

Ihmiskeskisessä tavassa taas pyritään noudattamaan perinteisiä käytettävyyden periaatteita: opittavuutta, tehokkuutta, muistettavuutta, virheminimointia ja kattavuutta. Näin pyritään luomaan elesanasto, joka on helppokäyttöinen ja muistettava ja intuitiivinen. Eleiden pitäisi muistuttaa metaforisesti ja ikonisesti toimintoja, joita niillä on tarkoitus ohjata, ja niiden pitäisi olla ergonomisia usein toistettunakin. Pitää kuitenkin muistaa, että myös ihmiskeskisen elesanaston eleet on voitava olla tarkasti tunnistettavissa, jotta niitä voitaisiin käyttää.

Elesanaston suunnittelu jakautuu neljään osaan. Ensimmäisessä vaiheessa pyritään **tunnistamaan toiminnot**, joita eleillä sovelluksessa on tarkoitus ohjata. Tässä kannattaa pitää mielessä vastaavien sovellusten perinteisten käyttöliittyminen toiminnot ja suunnittelumenetelmät. Seuraavaksi **seurataan käyttäjiä** heidän kuvaillessaan, miten he suorittaisivat löydettyjä toimintoja (niin sanottu Ozin velho -tutkimus [25]). Tässä vaiheessa on tarkoituksena tunnistaa potentiaalisia eleitä käyttäjien nonverbaalisesta viestinnästä. **Eleitä löydetään** erottamalla eri käyttäjien usein ja johdonmukaisesti käyttämiä liikkeitä ja asentoja. Lopuksi näin tunnistettuja elesanastoja **arvotetaan** käyttämällä niitä testeissä, joissa annetaan pisteitä muun muassa sanaston arvattavuuden, muistettavuuden ja ergonomisuuden suhteen.

2.2 Eleohjausmenetelmiä

Eleohjausjärjestelmä voidaan toteuttaa käyttäen useita eri menetelmiä. Monesti näitä menetelmiä jaotellaan ihmislähtöisesti modaliteettien eli aistikanavien avulla. Yksi jaottelutapa on seuraava: visuaalisuuteen eli konenäköön, haptisuuteen eli kosketukseen ja liikkeeseen, vestibulaarisuuteen eli tasapainoon ja auditiivisuuteen eli ääneen perustuviin järjestelmiin.

2.2.1 Konenäköön perustuvat

Konenäkö voidaan määritellä toimenpiteenä, jossa näkymästä otetaan tietoa analysoimalla kyseisen näkymän kuvia [51]. Konenäöllä on lukuisia kohdealueita, kuten kaukokartoitus, radiologia, mikroskopia, teollisuuden laadunvalvonta, robotiikka, viihde yleisesti ja tietenkin eleohjaus.

Toisen määritelmän mukaan konenäkö on videokuvan tai yksittäisen kuvan muuntamista päätökseksi tai toiseen esitysmuotoon. Tällä muutoksella pyritään saavuttamaan jokin tietty tavoite. Syöte voi kuvan lisäksi sisältää kontekstuaalista tietoa, kuten “kamera on asetettu auton katolle” tai “etäisyysmittari osoittaa että metrin päässä on jokin kohde”. Päätös puolestaan voisi olla “tässä näkymässä on henkilö” tai “kuvassa näkyy 14 syöpäsolua”. Uuteen esitysmuotoon muokkaaminen voisi olla värikuvan muuntaminen harmaasävykuvaksi tai kameran aiheuttaman liikkeen tasoittaminen videosta. [5]

Konenäköön perustuvassa järjestelmässä syöte saadaan erilaisten kameroiden avulla. Tyypillisiä konenäköjärjestelmässä käytettäviä kameroita ovat esimerkiksi tietokoneeseen liitettävät yksinkertaiset ns. verkkokamerat, syvyystunnistavat kamerat, infrapunakamerat tai stereokamerat. Kameran syötteestä pyritään erottamaan käyttöliittymäsovelluksen kannalta oleelliset alueet ja tulkitsemaan niissä tapahtuvaa liikettä joko hetkellisesti tai pidemmällä aikavälillä. Mielenkiinnon kohteena voi olla esimerkiksi käden tai pään asento ja niiden liike tietyllä alueella.

Eletunnistukseen konenäköperustaisuus soveltuu mainiosti. Oleellisia huomioon otettavia rajoitteita kuitenkin myös runsaasti. Käyttäjä ei esimerkiksi voi sijoittautua täysin vapaasti, vaan hänen tulee pysyä kameran kuvausalueella. Etenkin julkisella alueella toimivan järjestelmän tulee sietää myös muita seurattavalla alueella olevia ihmisiä (joista kaikki eivät välttämättä ole käyttäjiä). Luotettava järjestelmä vaatii jatkuvaa kuvankäsittelyä, joten myös suorituskykyrajoitteet tulee ottaa huomioon.

Hyvin toteutettu eletunnistusjärjestelmä ei vaadi käyttäjältään paljon huomiota, jolloin

säästetty huomio voi jäädä muualle käytettäväksi. Esimerkiksi autossa voisi äänenohjausjärjestelmä toimia eleohjauksella, jolloin ajajan ei tarvitse siirtää katsettaan muusta liikenteestä soitinlaitteeseen. [16]

Tällä hetkellä yksi tunnetuimpia konenäköön perustuvia järjestelmiä on Microsoftin Xbox 360 pelikonsolin lisälaite Kinect, jonka eleohjaus perustuu infrapunalaserprojektoriin ja -kameraan. Projektori lähettää ympäristöönsä pseudosatunnaishajautettuja infrapunavalopisteitä, jotka kamera havaitsee, vertaa niiden siirtymiä tunnettuun kovakoodattuun infrapunapistekuvioon ja laskee tästä pisteiden etäisyyden kamerasta. [56]

2.2.2 Haptisesti toimivat

Haptisissa järjestelmissä syötteet vastaanotetaan liikettä tunnistavien sensorien ja eriasteiseen kosketukseen reagoivien ohjauslaitteiden ja pintojen välityksellä. Käytännössä nämä siis voidaan jakaa liiketunnistusjärjestelmiin ja kosketusjärjestelmiin.

Liiketunnistuspohjaiset eletunnistusjärjestelmät vaativat erillisen ohjauslaitteen, esimerkiksi peliohjaimen, puhelimen tai hansikkaan, jossa on kiihtyvyysanturi ja/tai gyroskooppi. Nykyään tällaisia järjestelmiä on runsaasti saatavilla. Esimerkiksi lukuisissa älypuhelimissa on löytyy vähintään kiihtyvyysanturit. Ensimmäinen liiketunnistuspohjaista eleohjausjärjestelmää käyttävä videopelijärjestelmä oli Nintendo Wii. Kiihtyvyysantureita tarkempaan tulokseen päästään sähköisillä gyroskoopeilla, mutta vielä vähän aikaa sitten ne olivat liian kalliita käytettäväksi kuluttajaelektroniikassa. Sittemmin tilanne on muuttunut ja markkinoille tulee koko ajan lisää laitteita, jotka käyttävät myös gyroskooppia apunaan. Kiihtyvyysantureihin perustuva eleohjaus voidaan toteuttaa havaitsemaan staattista asentoa tai dynaamista liikettä. Useimmat mobiililaitteissa käytettävät kiihtyvyysanturit havaitsevat sekä gravitaatiokiihtyvyyttä että käyttäjän aiheuttamaa kiihtyvyyttä. Staattisen asennon havaitseminen on helppoa, sillä se voidaan laskea suoraan gravitaatiokiihtyvyydestä. Dynaamisen asennon havaitseminen on hieman hankalampaa, sillä siinä joudutaan yhdistämään epämääräisen pitkän ajanvälin yli tapahtuvia mittaustuloksia. [22; 9; 35; 18; 28]

Kosketusperustaisia järjestelmiä ovat kaikki sellaiset, joissa sormella tai jollain välineellä painetaan pintaa ja painamisen seurauksena tapahtuvat paine- tai sähköominaisuusmuutokset tulkitaan syötteiksi. Tällä määritelmällä ylivoimaisesti laajimmin käytössä oleva kosketussyötelaite on lähestulkoon jokaisesta pc-tietokoneesta löytyvä näppäimistö. Jostain syystä näppäimistöstä kuitenkin harvemmin puhutaan kosketus- tai eleohjauslaitteena. Syynä tähän voi hyvinkin olla laitteen arkipäiväisyys. Tyypillisemmin nykyään kosketuslaitteina tarkoitetaankin tasaisia pintoja, joista kosketuksen havaitaan pinnassa tapahtuvan sähköisen muutoksen myötä. Tämä muutos voidaan havaita joko resistiivisesti tai

kapasitiivisesti. Sinänsä kosketusnäyttöjen idea ei ole erityisen uusi; E.A. Johnson esitti ajatuksen resistiivisestä kosketusnäytöstä jo vuonna 1965 [24]. Resistiivisessä kosketuslaitteessa on tyypillisesti kaksi erillään olevaa pintaa, joille on vedetty sähköä johtavia juovia pysty- ja vaakasuuntaan. Kun pinnat kosketuksesta painautuvat yhteen, voidaan kosketuspisteen sijainti laskea juovien jännitemuutoksista. Kapasitiivisessa kosketuslaitteessa sormi tai jokin muu sähköä johtava materiaali, laitteen pinnalla ollessaan, muuttaa havaintosensorien yhteiskapasitanssia, jolloin kosketuspinta voidaan tulkita.

2.2.3 Tasapainoon perustuvat

Vestibulaariseen, eli tasapainoaistiin perustuvissa eletunnistusjärjestelmissä syötteet vastaanotetaan kallistuvien tai epäsymmetristä painetta havaitsevien alustojen ja laitteiden avulla. Tätä modaliteettia hyödynnetään enimmäkseen peliteollisuudessa ja simulaattoreissa, joissa kallistuvat ohjauslaitteet ovat tietyn tyyppisissä peleissä erittäin luonnollinen ohjaustapa. Esimerkiksi lumilautailuun tai moottoripyöräilyyn perustuvissa peleissä ja simulaattoreissa sovellusalueen omia liikkeitä imitoiva ohjaustapa, joka aikaansaa välittömän palautteen lisää sovelluksen todenmukaisuutta ja elämyksellisyyttä.

Kaupan hyllyltä kuluttajille suunnattuja vestibulaarisuutta hyödyntäviä syötelaitteita ei ole kovin paljoa tarjolla. Lähinnä kyseeseen tulee Nintendo Wiin WiiFit -pelin mukana tuleva elektronista vaakaa muistuttava Balance Board -tasapainolauta, joka mittaa etu- ja takakulmiin kohdistuvaa painetta, ja mittaa näin muun muassa pelaajan painoa ja kallistumista. Viihdepelaamisen lisäksi Balance Boardia on hyödynnetty muun muassa tasapainoongelmaisten kuntoutuksessa [14].

2.2.4 Auditiivisuuteen perustuvat

Auditiivisissa menetelmissä syöte tulkitaan äänen (puhe, sorminapsaukset jne) tulosuunnan ja äänenvoimakkuuden perusteella. Tässä on tarkoituksella tehty ero puheentunnistukseen perustuvaan järjestelmään. Näin auditiivista syötettä voidaan käyttää sekä erillisenä syötteenä että puheentunnistusjärjestelmän osana. Tosin esimerkiksi Inkeroinen [22] käsittelee lyhyesti puheentunnistusta eleohjausjärjestelmän osana. Tärkeimpänä johtopäätöksenään on, että vaikka puheentunnistus selvästi voisi olla luonnollinen tapa ohjata mobiililaitetta, on siinä silti runsaasti ongelmakohtia – kuten tunnistuksen tekniset rajoitteet ja julkisella paikalla laitteelleen puhumisen sosiaalinen hyväksyttävyys – ratkottavana. [22]

Eräs mielenkiintoinen esimerkki auditiivisesta syötteestä Olwalin ja Feinerin [43] kehittämä puheohjaukseen perustuva osoittimen liikutusjärjestelmä. Järjestelmä käyttää paitsi

puheentunnistusta myös äänen voimakkuutta ja karkeaa tulosuuntaa määrittämään, mitä käyttäjä haluaa tehdä. Voimakkaampi ääni tai nopeampi puhetahti saa aikaan voimakkaamman syötteen ja äänen tulosuunta auttaa puheentunnistusjärjestelmää tunnistamaan komentoja paremmin. Äänen suuntaa voi mahdollisesti myös käyttää supistamaan sanastoa, joka puheentunnistuksen tulee oppia. [43]

2.3 Mielenkiintoisia eleohjausta hyödyntäviä järjestelmiä

Pääosa olemassa olevista eleohjaustoteutuksista – etenkin konenäön parissa – on suunniteltu peliteollisuutta varten. Seuraavassa käydään läpi muutamia mielenkiintoisia tuoreita tapauksia muistakin toteutuksista.

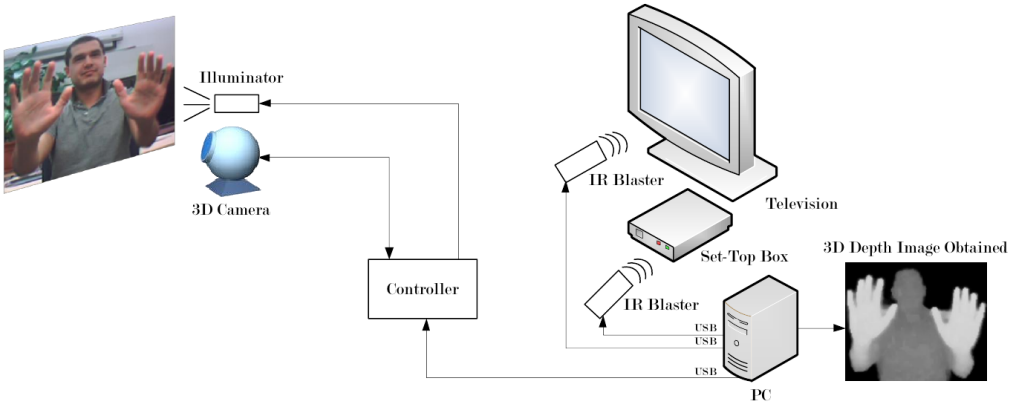
Catwalk on oululaisen MobileSpotin[4] kehittämä ohjelma, jota käytetään vaateliikkeessä ostopäätöksen helpottamiseksi. Yrityksen oman mainonnan mukaan Catwalk on ainutlaatuinen moderni peili, sovitustilan muuttuva mainos ja viihdyttäjä. Käyttäjä voi sen avulla helposti tarkastella sovitustaan useasta kuvakulmasta. Aikaisemmin Catwalk toimi hiiren avulla, mutta siihen on hiljattain lisätty myös eleohjaus [50].

Israelilaisen eyeSightin kehittämä eyeCan on mobiililaitteille tarkoitettu yleiskäyttöinen konenäköpohjainen eleohjausjärjestelmä. EyeSight toimii seuraamalla laitteen kameralla sen edessä tapahtuvia käsieleitä kuten käden heilauttaminen kameran ohi tai kämmenen vienti laitteen lähelle ja kauemmas. Näitä eleitä voi käyttää moniin eri tarkoituksiin, esimerkiksi musiikkisoittimessa toistettavan kappaleen vaihtamiseen tauottamiseen tai äänenvoimakkuuden säätämiseen tai puheluiden tapauksessa vastaamiseen ja vaientamiseen. EyeSight mainostaa eyeCanin käyttävän vain vähän virtaa ja vaativan vain vähän prosessointitehoa toimintaansa.[13]

Eleohjausta on käytetty myös television kontrollointiin. Ionescu et al. [23] toteuttivat laboratorio-oloissa syvyystunnistavaa kameraa hyödyntävän ohjausjärjestelmän. Sinänsä konenäön käyttäminen television ohjaamiseen ei ole erityisen uusi ajatus, mutta Ionescun et al. menetelmä on mielenkiintoinen sen käyttämän laitteiston takia. Kamerana käytettiin tarkoitusta varten rakennettua ja tietokoneeseen liitettyä syvyystunnistavaa kameraa, jonka avulla käyttäjän käsiä oli helppo seurata. Kun eleet oli saatu tunnistettua, tietokone lähetti infrapunasäätimellä normaalin kaukosäätimen tavoin purskeen televisiolle. Periaatekuva tästä on esitetty kuvassa 2.1.

Yksi viime aikoina paljon huomiota saanut erittäin mielenkiintoinen hanke on Mistryn & Maesin [34] kehittämä SixthSense, mukana kannettava (puettava) työkalu päivittäisen elämän helpottamiseksi. SixthSense laajentaa (engl. augment) ympärillämme olevaa maa-

[4] http://www.mobilespot.fi



Kuva 2.1: Periaatekuva Ionescun et al. konenäköperustaisesta TV:n ohjausjärjestelmästä[23].

ilmaa runsaalla digitaalisella tiedolla. Järjestelmään kuuluu kaulassa kannettavat kamera ja miniprojektori, sormimerkkaimet sekä taskussa kuljetettava mobiililaite. Järjestelmää ohjataan luontaisilla käsieleillä, jotka SixthSense tunnistaa käyttäjän sormenpäihin asetettujen merkkainten avulla. Käyttäjälle SixthSense antaa tietoa lähettämällä projektorilla kuvaa mille tahansa pinnalle. Käytännönläheinen demonstraatio SixthSensen käytöstä löytyy Mistryn pitämän TED Talks -puheen tallenteesta[5]. [34]

Viimeisenä esimerkkinä on FaceTracker -kasvonseurantakirjasto[6]. FaceTrackerin:n avulla on helppoa toteuttaa eleohjausta kasvojen liikkeiden avulla. Kirjasto esimerkiksi tunnistaa hyvin pään kiertämisen, kallistamisen eteen ja taakse sekä suun sijainnin ja koon. Näillä eleillä on mahdollista toteuttaa jo hyvin rikkaita, vain kasvojen eleillä ohjattuja sovelluksia. Esimerkiksi McDonald on kirjoittanut FaceTrackerin päälle eleohjaustyökalun jota käyttää äänisyntetisoija OSCulatorin ohjaamiseen [33].

[5] http://www.ted.com/talks/pranav_mistry_the_thrilling_potential_of_sixthsense_technology.html

[6] http://web.mac.com/jsaragih/FaceTracker/FaceTracker.html

3. ELEOHJAUS MOBIILILAITTEISSA

Mobiililaitteen – kuten kännykän tai tablettitietokoneen – käyttäminen eroaa selvästi perinteisempien tietokoneiden – kuten pöytä- ja kannettavien tietokoneiden – käytöstä. Niiden on tarkoitus kulkea käyttäjän mukana ja olla välittömästi käytettävissä. Siinä missä perinteinen tietokone on kiinteästi paikallaan tai vähintään lasketaan pöydälle tai jollekin muulle tasolle käyttämistä varten, mobiililaitetta pääasiallisesti käytetään kantamalla kädessä ja laitteen koosta riippuen operoidaan samalla tai vapaalla kädellä.

Ei siis ole kovin kaukaa haettua väittää, että mobiililaitteista on tullut hyvin luonnollinen ja välitön tapa käyttää tietotekniikkaa. Pitkän aikaa mobiililaitteiden pääasiallinen operointitapa oli käyttää vain laitteessa olevia fyysisiä painikkeita, vaikka myös stylus -kynillä käytettävät resistiiviset kosketusnäytöt olivat jossain määrin käytössä Esimerkkeinä tällaisista laitteista olkoon vaikkapa Palm Pilot 1000 (1996) ja Nokia 7710 (2004). Viimeisen muutaman vuoden aikana kosketusnäytöt ovat vallanneet runsaasti alaa mobiililaitteiden syötemenetelmänä ja laitteissa on mukana komponentteja, jotka mahdollistavat monipuolisten eleiden käyttämisen syötemenetelminä.

3.1 Eleohjausmenetelmät mobiililaitteella

Nykyaikaisessa mobiililaitteessa on useasti liitettynä mukaan runsaasti komponentteja, jotka mahdollistavat hyvin rikkaiden ja monipuolisten eleohjausta hyödyntävien sovellusten toteuttamisen. Laitteista löytyy esimerkiksi yksi tai useampia kameroita konenäköä varten, kosketusnäyttö ja liikesensoreita haptista eleohjausta varten, sekä mikrofoni auditiivisia järjestelmiä varten. Oikeastaan vain tasapainoa käyttävää ohjaus voi olla mahdoton laitteen omilla komponenteilla.

Toisaalta useat nykyaikaiset modernit laitteet tukevat bluetooth -tiedonsiirtomenetelmää, joka mahdollistaa lisälaitteiden liittämisen langattomasti. Tämän avulla voisi myös tasapainoon perustuvan ohjauksen toteuttaminen olla mahdollista mobiililaitteella. Esimerkiksi Nokian N900-puhelimelle on toteutettu kommunikointi Wiin ohjainlaitteiden kanssa[1], joten myös Wiin Balance Boardin käyttö pitäisi periaatteessa olla mahdollista.

[1] esimerkiksi `http://maemo.org/packages/view/wiicontrol/`

Myös mobiililaitteissa peliteollisuus on merkittävässä asemassa eleohjauksen edistäjänä, sillä huomattavan monet pelit käyttävät kosketuksen lisäksi kiihtyvyysanturia tai gyroskooppia hyödykseen pelin ohjauksessa. Tyypillisiä tapauksia ovat esimerkiksi laitteen kallistaminen, mikä kallistaa pelimaailman objekteja tai muuttaa painovoimaa ja laitteen liikuttaminen ilmassa, mikä muuttaa pelaajan näkökulmaa pelimaailmaan. Selvästikään ohjauksen ei tarvitse rajautua näihin perustapauksiin, vaan ainoastaan kehittäjien oma mielikuvitus on rajana.

Mobiililaitteelle eleohjausta suunnitellessa tulee kuitenkin pitää mielessä niiden rajoitteet: laitteella on käytössään rajoitetusti prosessointitehoa ja muistia perinteiseen tietokoneeseen nähden, ja prosessorin sekä muiden komponenttien ylimääräinen käyttö kuluttavat akkua. Viime vuosina mobiililaitteiden prosessorien nopeudet ovat olleet huimassa kasvussa, mutta nopeusero perinteisten koneiden prosessoreihin on vielä merkittävä. Tätä suuremmaksi rajoitteeksi voi tulla akkukesto, sillä koko ajan verkkovirran päässä olevaa laitetta ei voi jatkuvasti kuljettaa mukanaan.

3.2 Multimodaalisuus

Multimodaaliset järjestelmät käsittelevät kahta tai useampaa käyttäjän antamaa syötettä yhdessä multimediajärjestelmän ulostulon kanssa. Syötteet ovat esimerkiksi puheella, kynällä, katseella, kosketus- tai kallistuseleellä tai kehon liikkeellä aikaansaatuja. Multimodaalisten järjestelmien tarkoitus on tunnistaa ihmiskielen ja käyttäytymisen luonnollisia tapoja, joissa on mukana ainakin yksi tunnistukseen perustuva teknologia (esimerkiksi kynä, puhe, näkö). Multimodaalinen syötetapa on luontevaa ihmisille ja sen käyttö on havaittu toimivaksi monissa sovelluksissa. Lisäksi sen on havaittu vaativan vain vähän harjoittelua ja sen on huomattu parantavan käyttötehokkuutta sekä vähentävän virheherkkyyttä. [45]

Klassinen esimerkki multimodaalisesta käyttöliittymästä on “Media Room”, joka on huone, jonka keskellä käyttäjä istuu nojatuolissa ja ohjaa huoneen edessä olevalla näytöllä olevia kohteita puhumalla ja osoittamalla kädellä. Järjestelmä tunnistaa yksikertaisia komentoja, kuten “luo”, “siirrä”, “kasvata” ja “poista” sekä argumentteja kuten “tuo”, “tämä”, “tuosta” ja “tuohon”. Käyttäjä voi esimerkiksi sanoa “*Luo* sininen neliö *tuohon*” osoittaen näytölle kohtaa, johon neliö halutaan luotavaksi. Esimerkki on karkea, mutta on helppo huomata, miten se laajenee vaikkapa tekstinkäsittelyyn komennoilla: “*Valitse tuosta...tuohon* (käyttäjä koskettaa alku- ja loppupisteitä), *leikkaa...liitä tuohon* (käyttäjä koskettaa kohtaa, johon teksti halutaan liitettävän)”. [4]

Mobiililaitteille multimodaalisuus soveltuu mainiosti, koska niissä olevat sensorit tarjoavat runsaasti vaihtoehtoja eri syötetavoille. Käytössä on esimerkiksi mikrofoni, koske-

tusnäyttö, liiketunnistimia ja kompassi. Käyttö ei kuitenkaan ole ainakaan vielä erityisen laajaa. Tässäkin tapauksessa tekniikan käyttöönoton eturintamassa on peliteollisuus. Esimerkiksi Moto X Mayhem -pelissä[2] yhdistetään kallistus- ja kosketuseleitä niin, että laitetta kallistamalla ohjataan pelihahmon nojaamista ja kosketuseleillä kaasua ja jarrua. Hieman toisenlainen esimerkki on Nintendo DS:n Phoenix Wright: Ace Attorney -peli[3], jossa yhdistetään mikrofonin ja kosketusnäytön syötteet niin, että käyttäjä "levittää" kosketusnäytön avulla sormenjälkipuuteria ja "puhaltaa" sen pois. Laitteen mikrofoni sattuu sopivasti sijaitsemaan juuri kosketusnäytön alapuolella, joten puhaltaminen havaitaan selvästi lisääntyneenä kohinana mikrofonista.

Multimodaalisuudesta voisi myös olla apua ainoastaan konenäköohjaukseen läheisesti liittyvään aloitus-lopetus -ongelmaan, eli milloin kameran edessä olevan kohteen liikkeet tulisi tulkita eleiksi eikä vain satunnaiseksi liikehdinnäksi. Tunnistuksen aloituksen ja lopetuksen voisi havaita esimerkiksi käyttäjän antamilla äänikomennoilla, tai syötteen antajan pitäisi asettua vaikkapa paineherkän levyn päälle antaessaan elekomentoja.

3.3 Tyypillisiä eleitä mobiililaitteessa

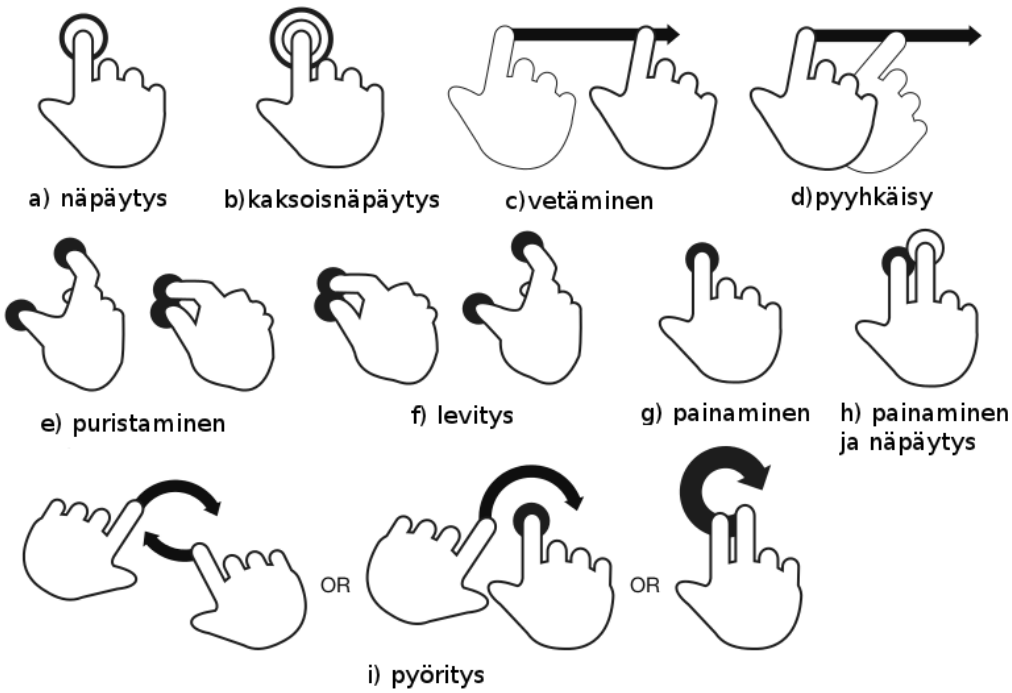
Tällä hetkellä huikean suosion saanut ja jatkuvasti kasvava eleohjausmenetelmä mobiililaitteiden parissa on kosketuksen avulla toimiva ohjaus. Tämän aliluvun eleiden ja niiden käyttökohteiden esittely perustuu pitkälti Wroblewskin [63] viiteohjeeseen. Kannattaa huomata, että vaikka Wroblewskin ohje on kirjoitettu kosketuslaitteita silmällä pitäen, voi monia siinä esitettyjä eleitä – ainakin rajoitetusti – pyrkiä käyttämään myös esimerkiksi konenäköön perustuvassa liikeohjauksessa.

Wroblewski jaottelee kosketuseleet seuraaviin ydineleisiin (eleiden nimet ovat suluissa englanniksi vakiintumattoman suomenkielisen termistön puutteen vuoksi): näpäytys (tap), kaksoisnäpäytys (double tap), vetäminen (drag), pyyhkäisy (flick), puristaminen (pinch), levittäminen (spread/expand), painaminen (press) ja pyöritys (rotate). Näitä eleitä havainnollistaa kuva 3.1. [63]

Näpäytyksessä (kuva 3.1a) käyttäjä koskettaa näyttöä lyhyesti sormenpäällään. Tyypillinen käyttökohde näpäytykselle on jonkin valinnan tekeminen koskettamalla valittavaa objektia näytöllä. Toinen käyttökohde on pysäyttää jonkin aiemman toimenpiteen seurauksena liikkuva näkymä, kuten lista tai verkkosivu. Näpäytyksen voi tehdä myös useammalla sormella yhtä aikaa, jolloin eleen merkitys voi muuttua.

Kaksoisnäpäytys (kuva 3.1b) on nopeasti peräkkäin toteutettu (yksöis)näpäytys. Tälle

[2] http://www.motoxmayhem.com/
[3] http://fi.wikipedia.org/wiki/Phoenix_Wright:_Ace_Attorney



Kuva 3.1: Wroblewskin [63] jaottelemia kosketuseleitä.

eleelle tyypillinen käyttökohde on automaattinen zoomaaminen; kuvaa tai muuta sisältöä lähennetään tietyn verran näpäytetyn kohdan ympäriltä ja loitonnetaan myöhemmillä kerroilla.

Vetämiseleessä (kuva 3.1c) käyttäjä kuljettaa sormeaan kosketuspinnalla irroittamatta kontaktia välillä. Tyypillisin käyttökohde on näkymän vierittäminen käyttäjän valitsemaan suuntaan, jolloin sisältö ikään kuin tarttuu sormeen ja liikkuu siten sormen mukana. Toinen käyttötapaus ovat säätötoimenpiteet kuten listanäkymän järjesteleminen niin että yhtä listan elementtiä siirretään sormella muiden elementtien pysyessä paikallaan, liukusäätimen (engl. scrollbar) liu'uttaminen sormella ja listan elementin poistaminen vetämällä elementti sivuun listasta.

Pyyhkäisyssä (kuva 3.1d) sormea kuljetetaan suoraan pitkin kosketuspintaa ja sitten irrotetaan. Tyypillisin käyttökohde tälle eleelle on pitkän listan tai muun sisällön nopea selaaminen. Kuten vetämisessä, sisältö ikään kuin tarttuu sormeen ja liikkuu sormen liikkeen mukana, ja kun käyttäjä laskee sormen irti, sisällön liike jatkuu.

Puristaminen ja erottaminen (kuva 3.1e,f)liittyvät läheisesti toisiinsa, sillä ne ovat vastakkaisia liikkeitä. Puristamisessa käyttäjä siirtää kahta kosketuspinnalla olevaa sormeaan kohti toisiaan ja erottamisessa kauemmas toisistaan, irroittamatta kosketusta. Käytännössä tärkein kohde tälle on zoomaaminen siten että erotusleellä näkymää lähennetään ja puristuseleellä loitonnetaan. Tämä mahdollistaa kaksoisnäpäytystä hienojakoisemman

zoomauksen.

Painamiseleessä (kuva 3.1g) käyttäjä koskettaa näyttöä sormenpäällään, mutta näpäytyksestä poiketen ei nosta sitä ylös. Tätä käytetään esimerkiksi näyttämään lisätietoa painamisen alla olevasta elementistä, avaamaan kontekstimenu ja näyttämään painamisen alla olevaa elementtiä "suurennuslasilla". Painamiseleeseen voidaan myös liittää näpäytys(kuva 3.1h), joka tehdään toisella sormella kuin itse painaminen ja nostamatta alkuperäistä painavaa sormea. Myös tällä eleellä voidaan luoda esimerkiksi kontekstimenu, sillä se muistuttaa hiiren toisen painikkeen käyttöä, mitä WIMP-mallissa usein käytetään kontekstimenun avaamiseen.

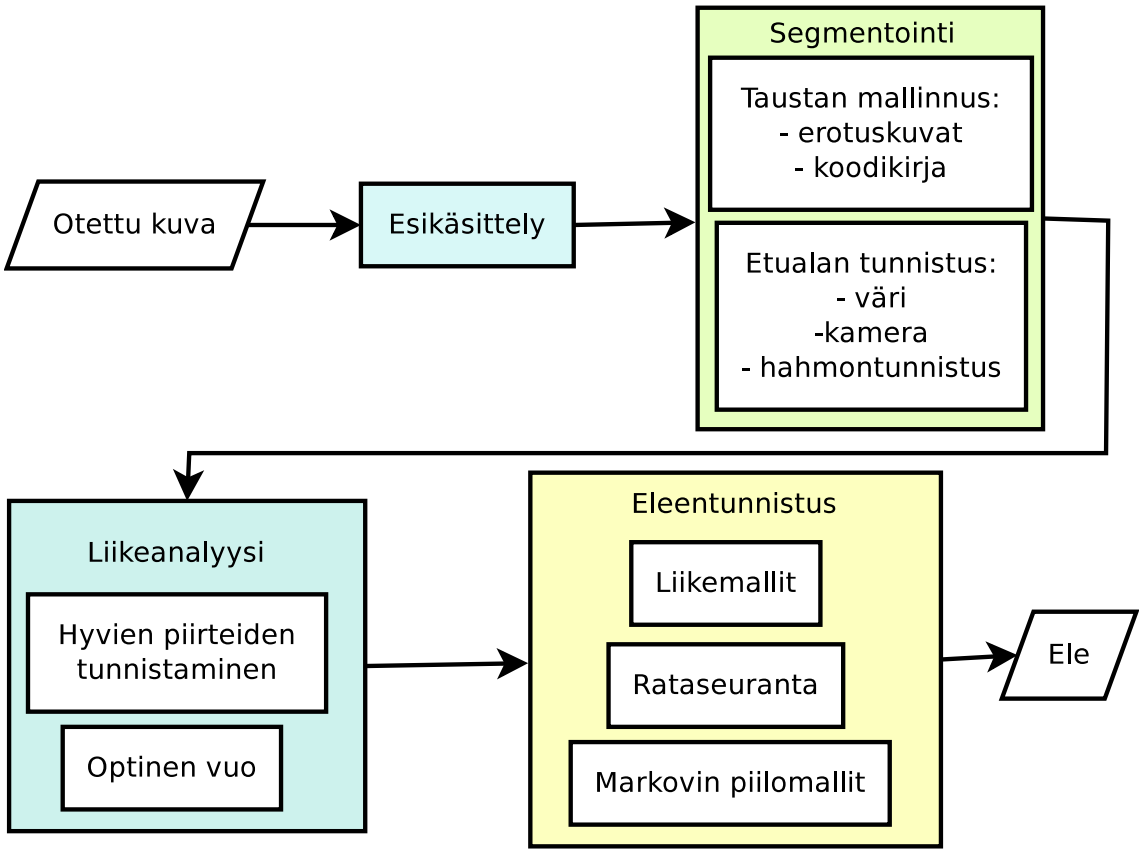
Pyörityseleessä (kuva 3.1i) käyttäjä pitää kahta sormea kosketuspinnalla ja joko kiertää niitä toisiinsa nähden tai pyörittää niitä yhdessä myötä- tai vastapäivään. Tyypillisin käyttökohde pyöritykselle on kääntää kuvia tai muita elementtejä. Kun kaksi sormea asetetaan elementin päälle, sormien alla olevat kohdat taas "liimautuvat"sormiin kiinni ja kun sormia pyöritetään, kiinni otettu elementti pyörii mukana.

Joissakin tapauksissa on mahdollista toteuttaa järjestelmiin käyttäjien omia tallennettuja eleitä. Tässä tapauksessa käyttäjä ensin toistaa haluamansa eleen useaan otteeseen, jotta järjestelmä oppii sen piirteet. Tämän jälkeen tallennettu ele voidaan kohdentaa haluttuun toimintoon. Tyypillistä tällaisille eleille on tietynlainen epätarkkuus, joten ne eivät täysin sovellu käytettäväksi tarkkuutta vaativiin toimintoihin, kuten listan alkioiden järjestämiseen tai poistamiseen. [53; 22]

4. ELEIDEN TUNNISTAMINEN KONENÄÖLLÄ

Tyypillisiä konenäköön perustuvasta eleohjausjärjestelmästä löytyviä komponentteja ovat esikäsittely, segmentointi, seuranta ja eleentunnistus. On mahdollista, että yksittäisissä järjestelmissä jotkut näistä voivat olla yhdistettynä tai komponenttina puuttua, mutta niiden toiminnallisuus on joka tapauksessa yleensä läsnä. [36; 50; 32; 3]

Kaavio tällaisesta tyypillisestä järjestelmästä on esitetty kuvassa 4.1. Kameralta saadusta kuvasta siis ensin pyritään löytämään mielenkiintoiset alueet, joiden liikkumista seurataan useiden kehyksien ajan ja lopputuloksesta päätellään mitä liikettä on tapahtunut.



Kuva 4.1: Tyypillisen konenäköön perustuvan eletunnistusjärjestelmän osat.

4.1 Taustan mallintaminen

Konenäköjärjestelmässä etu- ja taka-alalla tarkoitetaan yksinkertaisesti niitä kuvan alueita, joissa on sovelluksen kannalta mielenkiintoista tietoa (etuala) ja niitä joissa sitä ei ole (taka-ala). Taka-alan poistamisella tarkoitetaan siis sovelluksen kannalta epäoleellisten osien poistamista. Koska kulloinenkin taka-alan määritelmä on sovellusalueesta riippuvainen, joudutaan taka-alan poistamisessa monesti käyttämään eri menetelmiä. [58]

Tyypillisesti taka-alana kuitenkin pidetään sellaisia kuvan alueita, jotka joko pysyvät staattisina tai liikkuvat jaksollisesti kuvassa. Taustan liike voi olla myös aikariippuvaista. Esimerkiksi kuvassa voi olla kasvillisuutta, joka huojuu aamulla ja illalla tuulessa, mutta keskipäivällä pysyy paikallaan. Vastaavasti moottoritiellä voi autojen tiheä jaksollinen kulkeminen kuulua taustakuvaan, mutta yöllä liikennettä on vain harvakseltaan. [5]

Toyama et al. [58] luettelevat lukuisia taka-alan mallinnuksessa ongelmallisia tilanteita: taka-ala voi liikkua, kellonaika voi vaikuttaa valaistukseen, samoin kuin muut nopeat muutokset valaistuksessa (esim. valokytkimen kääntö), etualalla olevan kohteen pikselit saattavat vastata taka-alaa, etualan pysähtynyt kohde voi olla vaikea erottaa ensin liikkuneesta ja sitten pysähtyneestä taustasta. Ja tämä oli vain osa listaa! Kattava taka-alan mallinnus siis vaatii paneutumista ongelmakenttään.

Yleisiksi ohjenuoriksi hyvän taka-alamallin luomisessa ja ylläpitämisessä Toyama et al. [58] antaa viisi periaatetta:

- Taustamallin ei pitäisi tehdä semanttista erottelua kohteille
- Taustamallin pitäisi segmentoida kiinnostavat kohteet kun ne ilmestyvät tai uudelleenilmestyvät näkymään
- Taustamallin tulee määritellä mitä tarkoitetaan paikallaan olemisella. Pisteet, jotka täyttävät tämän määritelmän, ovat taustaa
- Taustamallin tulee kyetä sopeutumaan sekä nopeisiin että rauhallisiin muutoksiin taustassa
- Taustamallin pitää ottaa huomioon muutokset eri spatiaalisilla skaaloilla.

Yksinkertaisin tapa on laskea kahden perättäisen kuvan erotus. Jos taustan oletetaan voidaan olettaa olevan staattinen, kamera ei liiku, eikä valaistuksessa tapahdu muutoksia, erotuskuvasta nollautuu peräkkäisissä kuvissa oleva samanlainen tausta ja muuttuneet osat voidaan käsittää etualaksi. Menetelmä on karkea, mutta nopea. Useimmiten ei kuitenkaan voida tehdä mainittuja oletuksia ja tarvitaan hienostuneempia tapoja. [5]

Käytännössä useimmissa kuluttajille myytävien kameralaitteiden kuvassa läsnä pientä kohinaa, jota ei juuri huomaa varsinaista kuvaa katsoessa, mutta jonka läsnäolon huomaa helposti erotuskuvassa. Erotuskuvasta on siis tarpeen kynnystää (engl. threshold) tiettyä raja-arvoa pienemmät arvot nollaksi ja sitä suuremmat 255:een. Esimerkiksi Manchanda & Bing [32] käyttävät erotuskuvien kynnystyksen raja-arvona 30:a, jonka on todettu riittävän pienen kamerakohinan eliminoimiseksi useimmissa valaistusolosuhteissa.

On tärkeää ymmärtää, että erotuskuvista saadaan vain tieto alueesta, joka on muuttunut. Joissakin tapauksissa tämäkin voi olla riittävä; voidaan esimerkiksi tallentaa kuvassa muuttuneiden alueiden keskipisteen sijainteja. Yleensä ollaan kuitenkin kiinnostuneita tietyistä muuttuneiden alueiden piirteiden liikkeistä.

Vähän kehittyneemmät metodit luovat useamman perättäisen kuvan muuttumattomista tai vain vähän muuttuvista osista mallia tausta-alaksi. Esimerkiksi *keskiarvomenetelmässä* lasketaan kuvan kaikista pisteistä keskiarvo tietyn ajanjakson yli. Jos pikselin arvon ja keskiarvon erotus myöhemmin on suurempi kuin tietty kynnys, kyseinen pikseli katsotaan kuuluvaksi etualaan. *Derivaattamenetelmässä* puolestaan opetusjakson aikana tallennetaan kunkin pikselin maksimi- ja minimiarvot sekä suurin muutos, mitä kyseisen pikselin arvossa kahden peräkkäisen ajanhetken välillä on tapahtunut. Jos jälkikäteen pikselin arvo on kyseistä maksimimuutosta suurempi tai pienempi kuin löydetyt raja-arvot, katsotaan kyseinen pikseli etualaan kuuluvaksi. Toyama et al. [58] sekä Kim et al. [26] esittelevät tarkemmin näitä ja monia muita kehittyneempiä menetelmiä taustan mallintamiseksi.

4.2 Etualan tunnistaminen

Monesti kuvan etualasta voidaan tehdä vahvoja oletuksia, joiden avulla etuala voidaan tunnistaa ilman erillisen taustamallin rakentamista. Etualan tunnistamista voidaan mahdollisesti käyttää hyödyksi myös taustamallin rakentamisessa tai etu- ja taka-alan erottamisessa voidaan käyttää molempia tietoja hyödyksi.

Tunnettuja käyttökelpoisia menetelmiä etualan tunnistamiseksi ovat mm. värin perusteella rajaaminen, erinäiset kamerajärjestelmät ja etualan tunnistaminen hahmontunnistuksen menetelmin käyttäen niin sanottuja Haar-tyyppisien piirteiden tunnistusta. Seuraavissa kohdissa käydään näitä menetelmiä tarkemmin läpi.

Kuriositeettina mainittakoon etualan tunnistaminen korrelaatioita laskien. Tässä menetelmässä kuvasta tai sen alueesta lasketaan “samankaltaisuutta” haluttuun kohteeseen esimerkiksi luomalla molemmista histogrammi ja vertailemalla niitä. Tämä on hieman vanhentunut ja rajoittunut mutta tietyissä tapauksissa käyttökelpoinen menetelmä, joka voi soveltua käytettäväksi, mikäli halutaan havaita tietynlaisen histogrammin sisältävän koh-

teen ilmestyminen rajatulle alueelle. Menetelmän suurin ongelma on histogrammin laskemisessa katoava tilatieto; kuvan pikselien järjestystä muuttamalla saadaan lähes mielivaltaisen suuri määrä kuvia, joilla on sama histogrammi. Vastaavasti kahta alun perin eri kuvaa on mahdollista muokata niin, että niiden histogrammit täsmäävät. [15; 2]

4.2.1 Värirajaaminen

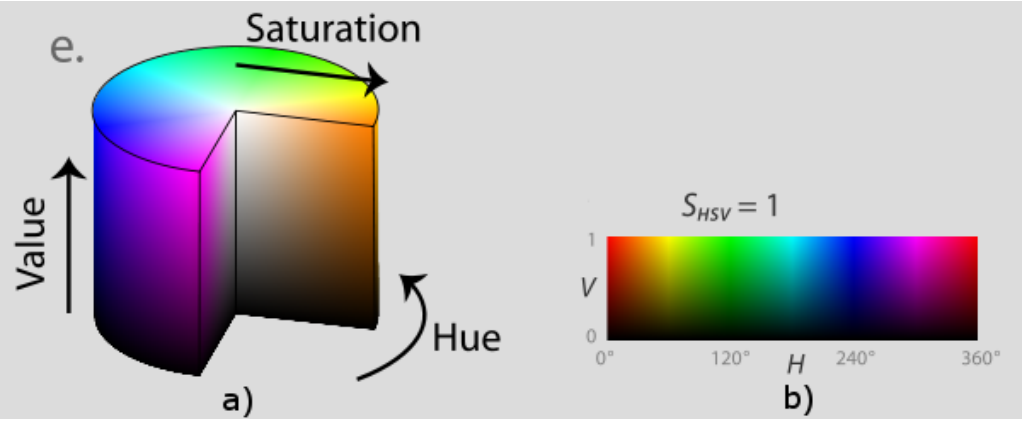
Useimmille vain vähän kuvankäsittelyn kanssa tekemisissä olleille ihmisille tunnetuin värimalli on RGB-malli, jossa kuvan yksittäisen pisteen väri kuvataan kolmen arvon: punaisen (**R**ed), vihreän (**G**reen) ja sinisen (**B**lue) yhdistelmänä. Esimerkiksi television ja tietokoneen näytöllä yhden kuvapisteen väri luodaan kolmella vierekkäisellä RGB-juovalla. Kun pistettä tarkastellaan riittävän läheltä, erilliset juovat on mahdollista erottaa paljain silmin, mutta tyypillisellä katseluetäisyydellä (> 30cm) juovat “sulautuvat” yhdeksi väriksi.

RGB-mallia intuitiivisempi tapa hahmottaa värejä on kuitenkin HSV -värimalli, jossa yksittäisen pisteen väri määritellään kolmen attribuutin: värisävyn (**H**ue), värikylläisyyden (**S**aturation) ja kirkkauden (**V**alue) perusteella. Siinä missä RGB-mallissa yhden värikolmikon arvon muuttaminen vaikuttaa sekä sävyyn, värikylläisyyteen että kirkkauteen, HSV-mallissa voidaan näitä arvoja muuttaa riippumattomasti. Näin juuri tietyn värin valitseminen on intuitiivisempaa. [7]

Eräs visuaalinen tapa esittää HSV-malli on pystyssä oleva sylinteri, jonka pystyakselilla on värikirkkaus. Kun tämän akselin ympäri piirretään ympyrä, on värisävy tälle ympyrälle piirretty kulma suhteessa punaiseen väriin niin, että punainen kulmassa 0 radiaania, vihreä kulmassa $\frac{2\pi}{3}$ radiaania, sininen kulmassa $\frac{4\pi}{3}$ radiaania jatkuen punaiseen taas kulmassa 2π radiaania. Värikylläisyyttä taas mallintaa pisteen etäisyys sylinterin keskiakselilta. Kuva 4.2a havainnollistaa tätä sylinterimallia.

Kuva 4.2b, puolestaan havainnollistaa värisylinterin yhtä kerrosta, jossa värikylläisyys on rajattu maksimiinsa. Tästä kuvasta on helppo havaita, miten esimerkiksi useimmat kirkkaan vihreät sävyt saadaan rajaamalla HSV-arvoihin $S = 1, V \in \{v : v > 0.5 \wedge v \leq 1.0\}$ ja $H \in \{h : h > 100^\circ \wedge h < 140^\circ\}$

Jos tunnetaan etualaksi tunnetun kohteen väriprofiili, voidaan kuvasta rajata näkyväksi vain tietyn väriset kohteet. Esimerkiksi ihonväri on monesti käytetty menetelmä sopivan etualan rajaamiseksi [50; 27]. Rajaus voi myös olla käänteinen; haluttu väri katsotaan kuuluvaksi taustaan, joka halutaan poistaa. Kyseinen menetelmä on hyvin yleinen televisio- ja elokuvatuotannon erikoistehosteiden kanssa. Näyttelijät kuvataan kirkkaan vihreän tai sinisen kankaan edessä, minkä ansiosta kuvaan on myöhemmin helppoa liittää



Kuva 4.2: a)HSV värisylinteri, b)HSV-rajaus kylläisimmän värin mukaan. Muokattu lähteestä Rus [52].

täysin erillinen taustakuva. Tällaista menetelmää kutsutaan kroma-avainnukseksi (engl. chroma keying).

Kuvan rajaaminen väriarvojen perustella on suhteellisen nopeaa, sillä kukin kuvan piste joudutaan tutkimaan vain kertaalleen, jotta voidaan päättää kuuluuko piste etualaan. Toisaalta pisteelle joudutaan vertailun lisäksi tekemään muunnos RGB-mallin mukaisesta pisteestä HSV-mallin mukaiseksi pisteeksi, sillä digitaalikamerat monesti tallentavat otetun kuvan RGB-mallin mukaiseksi. Yleensä tällä ei ole merkitystä, mutta mikäli käytettävissä oleva prosessointiaika on hyvin rajoitettu, muunnos voidaan joutua ottamaan huomioon.

Yleisesti ongelmana värin mukaan rajaamisessa on toisaalta riippuvuus valaistuksesta [50], ja toisaalta sopivien rajausväriarvojen päättäminen [27].

4.2.2 Kameramenetelmiä

Värirajauksen lisäksi etuala voidaan tunnistaa myös käyttämällä erityisiä kameroita. Tyypillisesti käytetään kameraa, jolla voidaan valmiiksi luoda jonkinlainen 3d-malli näkymästä, kuten stereokamera tai syvyystunnistavaa kameraa Vaihtoehtoisesti kamera voi tallentaa jotain muuta kuin näkyvää valoa, esimerkiksi infrapunaa.

Infrapunatuikkumenetelmässä näkymää kuvataan infrapunakameralla ja näkymässä mielenkiintoisiin kohteisiin kiinnitetään pistemäiset infrapunavalonlähteet, esimerkiksi LEDit. Olettaen että näkymää ei muuten valaista infrapunavalolla, voidaan näitä valaistuja pisteitä käyttää etualana. Mikäli pisteiden etäisyys on tunnettu, voidaan myös niiden etäisyys kamerasta ja asento laskea. Esimerkiksi Nintendo Wiin Wiimote-ohjain toimii tismalleen näin. Ohjaimeen upotettu infrapunakamera seuraa näytön päälle tai alle ase-

tetun sensor barin reunoilla olevia infrapunaLEDejä ja muodostaa näin mallin sijainnistaan, suunnastaan ja asennostaan. Ohjain välittää nämä tiedot itse konsolille langattomasti. Myös Reifinger et al. [49] käyttivät tätä menetelmää. Heidän työssään järjestelmän käyttäjän peukaloon ja etusormeen laitetaan joukko kevyitä infrapunaLEDejä. LEDien määrä ja sijainti mahdollistivat monipuolisen käden sijainnin ja asennon määrittämisen, jolloin virtuaaliesineitä oli mahdollista nostaa peukalon ja etusormen väliin. [49; 28]

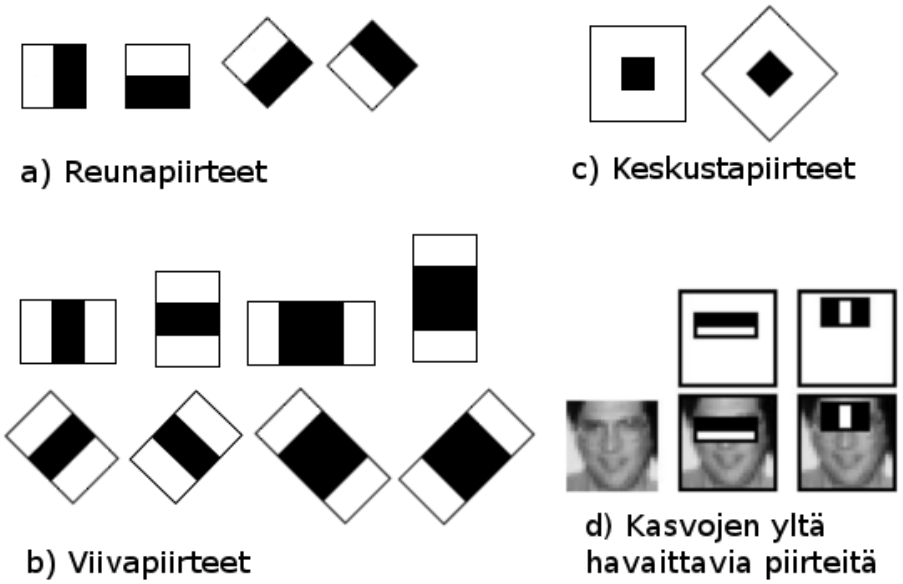
Syvyystunnistavalla kameralla tarkoitetaan niin sanottua Time-of-Flight (TOF) -kameraa, joka toimii valaisemalla ympäristöään infrapunapulsseilla ja hyvin nopealla suljinajalla. Jokaisen pulssin lähettämisestä seurataan aikaa, joka kestää näkymässä olevasta kohteesta heijastuvan valon saapumiseen takaisin kuvasensorille. Mitä lyhyempi tämä vaihe-ero on, sen lähempänä kohdekin on. Tällaiseen laskentaan tarvittava laitteisto on aiemmin ollut liian kallis kuluttajakäyttöön. Kamerat ovat maksaneet jopa 12000 dollaria, mutta tarvittavien laitteiden hinnat ovat laskemassa ja jopa 100 dollarin hintaisia kameroita on ennustettu tulevan piakkoin saataville [12]. Tällaisella kameralla kuvan etuala saadaan muodostettua helpohkosti; jos oletetaan että näkymässä on mukana vain yksi ihminen, on etuala lähin sopivan kokoinen objekti, kuten ihminen tai käsi. Tämä mahdollistaa luotettavan ja hyvin hienovaraisen eleohjauksen hyvinkin kaoottisen ja taustamallinnuksen kannalta hankalan taustan edessä. Kun kameralaitteisto tarjoaa tausta-alan erotuksen ikään kuin puoli-ilmaiseksi, ei itse sovelluksen tarvitse käyttää tähän prosessointiaikaa. Näin syvyystunnistavan kameran avulla mahdollista toteuttaa sovelluksia, jotka vaativat suhteellisen lyhyttä viivettä käyttäjän syötteen ja sovelluksen reagoinnin välillä. Erityisesti nopeatempoisissa peleissä käsittelyviiveen minimointi on tärkeää. Myös “gorillakäsiefekti” minimoituu, kun aika, jonka käyttäjä joutuu pitämään raajojaan kaukana kehostaan pienenee. [23; 17; 30]

Stereokameran käytön ajatus on ottaa samasta näkymästä kaksi samanaikaista kuvaa hieman eri kuvakulmista. Nämä kuvat voidaan ottaa joko kahdella eri linssillä tai yhtenä kuvana erityisten prisma- tai peilijärjestelyiden avulla, ja niiden avulla voidaan päätellä näkymässä olevien kohteiden etäisyys. Käytännössä tämä on melko raskas laskennallinen tehtävä. Ensin jokaiselle vasemman kuvan jokaiselle pisteelle pitää etsiä vastaava piste oikeasta kuvasta, eli löytää kuvista pisteet, jotka kuvaavat alkuperäisen näkymän samaa pistettä. Vastepisteiden välinen etäisyyttä kutsutaan siirtymäksi (engl. disparity). Siirtymän suuruus on kääntäen verrannollinen tarkasteltavan pisteen etäisyyteen kamerasta; mitä suurempi siirtymä on sitä lähempänä piste on kameraa. Stereokameralla on pari merkittävää etua infrapunaa käyttäviin. Ensinnä se on täysin passiivinen menetelmä, eli sen ei toimiakseen tarvitse lähettää itse (infrapuna)valoa. Tällä on merkittävä vaikutus tilanteissa joissa ylimääräinen valo on haitallista tai halutaan säästää virrankulutuksessa. Toisekseen stereokameralla otetusta kuvaparista myös ihminen kykenee havaitsemaan kolmiulotteisen kuvan katsomalla rinnakkaisia kuvia yhtä aikaa. [62; 55]

4.2.3 Haar-tyyppisten piirteiden tunnistaminen

Haar-tyyppiset piirteet saavat nimensä Alfréd Haarin aloittamasta työstä aallokefunktioiden (engl. wavelet) parissa. Haar-piirteet muistuttavat Haar-aalloketta, joka on kulmikas aaltofunktio. Näitä allokkeita käytetään enkoodaamaan signaalin keskimääräisen intensiteetin muutoksia tietyillä alueilla [46]. Haar-piirteet myös lasketaan samoin kuin Haar-aallokemuunnoksen kertoimet [8].

Alkuperäiset Haar-tyyppiset piirteet koostuvat kahdesta tai kolmesta vierekkäisestä mustasta ja valkoisesta neliöstä, mutta monesti käytetään myös niin sanottuja laajennettuja piirteitä, jotka voivat olla kierrettyjä ja joissa viivapiirre voi olla neliön sijasta suorakaide [8]. Haar-tyyppiset piirreprimitiivit on havainnollistettu kuvassa 4.3a-c. Kuvassa 4.3d puolestaan on esitetään miten erilaiset piirretyypit voidaan tunnistaa kasvokuvan alueiden kirkkauseroista: silmien alueella on silmistä ja kulmakarvoista johtuva voimakas vaakasuuntainen reunapiirrealue ja silmien välissä on nenän aiheuttama vaalea viivapiirrealue.



Kuva 4.3: Laajennetut Haar-tyyppiset piirteet.

Haar-tyyppisen piirteen arvo on primitiivin mustien ja valkoisten suorakaiteiden peittämien alueiden pikselisummien erotus. Tämä arvo on nopea laskea mielivaltaisen kokoiselle primitiiville tuottamalla ensin alkuperäisestä kuvasta integraalikuva. Integraalikuvassa pisteen (x, y) arvo on kaikkien sen vasemmalla ja yläpuolella olevien pisteiden summa. Kun integraalikuva on kerran laskettu, saadaan minkä tahansa alkuperäistä kuvaa peittävän suorakulmion alueen pikselisumma laskettua vakioaikaisesti vertaamalla neljää integraalikuvan pistettä. [47; 60]

Luokittimen opettamiseen voi käyttää mitä tahansa koneoppimismenetelmää, mutta kan-

nattaa pitää mielessä, että jo kohtuullisen pienessä havainnointi-ikkunassa erikokoisien primitiivipiirteiden yhdistelmiä runsaasti. Esimerkiksi Viola & Jones [60] käyttävät 24x24 pikselin ikkunaa, jolloin piirteitä on yli 160000. Minkä tahansa luokittimen opettaminen näin usean piirteen kanssa on erittäin työlästä. Tehokas luokitin on kuitenkin mahdollista muodostaa yhdistämällä hyvin pieni joukko näistä piirteistä. [60]

Haar-tyyppisten piirteiden tunnistamisessa käytetään usein tehostettuja luokittimia (engl. boosted classifiers), joissa vahva luokitin luodaan käyttämällä mielivaltaisen suurta määrää heikkoja luokittimia. Heikko luokitin on luokitin, joka kykenee luokittelemaan näytteitä satunnaisvalintaa paremmin. Käytännössä riittää jos yksittäisen heikon luokittimen tarkkuus on vaikkapa vain 51%. Yhdistetyn vahvan luokittimen luokitusvirhe lähenee tällöin eksponentiaalisesti nollaa heikkojen luokittimien määrän kasvaessa. [60]

Haar-tyyppisiin piirteisiin perustuvan luokittimen ehdottoman hyviä puolia ovat sekä sen laskennallinen nopeus, että sopivalla opetusmateriaalilla opetettuna toimintavarmuus. Viola & Jones [60] ovat esimerkiksi saanet tunnistettua kasvoja erittäin tarkasti 384x288 -kokoisista kuvista 700MHz PIII -suorittimella varustetulla PC-tietokoneella 15 kuvan sekuntivauhdilla.

Huonona puolena on kattavan opetusmateriaalin hankkimisen vaikeus, luokittimen opettamisen hitaus ja monimutkaisuus, sekä sopivan etualan riippuminen luokittimen opetusmateriaalista. Esimerkiksi Barczak & Dadgostar [2] ovat saaneet muuten hyviä tuloksia kämmenen havaitsemisessa Haar-tyyppisien piirteiden avulla ja kontrolloiduilla käden asennoilla. Ongelmaksi oli tullut käsien liikkuvuus. Kämmenet voivat olla lukuisissa eri asennoissa ja käden asento vaikuttaa ratkaisevasti siinä esiintyviin piirteisiin. [2; 60]

Haar-tyyppisten piirteiden hyödyntämisessä kannattaa myös pitää mielessä, että vaikka menetelmä on hahmontunnistuksellisessa mielessä nopea, on se silti suhteellisen raskas laskea. Tämän merkitys korostuu, mikäli kuvasta joudutaan etsimään useita Haar-tyyppisillä piirteillä tunnistettavia muotoja. Ince et al. [21] ovat helpottaneet tätä ongelmaa tekemällä kullakin kehyksellä vain yhden haun, ja haettava muoto on ollut kiertänyt haettavien muotojen joukossa. Tätä on tehostettiin edelleen suorittamalla haku vain kerran usean kehyksen intervallilla.

4.3 Optinen seuranta

Etu- ja taka-alan erottelun väliin sijoituu vielä yksi mielenkiintoinen työkalu. Joskus kattavan taustamallin rakentaminen on resurssien kulutuksen kannalta liian raskasta ja toisaalta etualasta ei tiedetä riittävästi värejä tai muotoja, jotta vain sen perusteella voitaisiin erotella kuvan etu- ja taka-ala. Tällöin yksi keino on etsiä videovirrasta jollain tapaa mie-

lenkiintoisia pisteitä ja pyrkiä seuraamaan niiden liikkumista peräkkäisten kuvien välillä.

Toisaalta jos etualan ominaisuuksia tunnetaan riittävästi, jotta erottelu voidaan tehdä vain niiden perusteella, mutta itse erottelu on raskasta, voi olla järkevämpää tehdä etualan tunnistaminen vain harvoin ja pyrkiä seuraamaan tämän tunnistetun alueen liikkeitä. Menetelmiä, joilla näitä tällaisia mielenkiintoisia kohtia tunnistetaan, ja jo mielenkiintoiseksi tunnistettujen pisteiden liikkumista kuvien välillä seurataan, kutsutaan optiseksi seurannaksi.

4.3.1 Hyvien piirteiden tunnistaminen

Jotta kuvan n piste olisi helposti löydettävissä ja tunnistettavissa samaksi pisteeksi kuin kuvassa n-1, olisi tällä seurattavalla pisteellä hyvä olla tiettyjä ominaisuuksia. Bradski & Kaehler [5] ehdottavat seuraavaa: pisteen pitäisi olla uniikki lähellään olevien muiden pisteiden joukossa ja parametrisoitavissa siten että sitä voidaan verrata muihin pisteisiin toisessa kuvassa. Lisäksi piirteen olisi toivottavaa tulla tunnistetuksi samaksi piirteeksi, vaikka kuvassa tulisi (kohtuullisia) valaistus- ja kuvakulmamuutoksia [59].

Hyvässä tunnistettavassa piirteessä voisi siis olla vahva derivaatta-arvo kahteen toisiaan vastakkaiseen suuntaan. Tällaisia kohdat näkyvät kuvassa intuitiivisesti kulmina, joten niitä tavataan sanoa kulmapisteiksi. Tällaisten pisteiden määrittämiseksi käytetään usein Harris-menetelmää, jossa kulmapisteeksi määritellään sellainen kuvapiste, jonka toisen derivaatan autokorrelaatiomatriisilla on kaksi suurta ominaisarvoa. Matemaattista perustaa menetelmälle löytyy esimerkiksi lähteistä [19] ja [54].

Oleellista siis on, että kulmapisteessä löytyy voimakkaasti muuttuva kuvakirkkausgradientti kahteen suuntaan. Tällöin kulmapiste on joko “oikea” kuvapiste tai yksittäinen kirkas piste tummalla (tai tumma piste vaalealla) taustalla. [53]

4.3.2 Optinen vuo

Kun kuvasta on löydetty hyvät seurattavat piirteet, selvitetään seuraavaksi miten ne liikkuvat perättäisten kuvien välillä. Optinen vuo on kuvassa näkyvien piirteiden nopeuksien jakauma, joka aiheutuu havainnoijan, kuten kameran, ja näiden piirteiden välisestä suhteellisesta nopeuserosta [20]. Optisella vuolla voidaan siis määritellä peräkkäisten kuvien välillä tapahtunutta liikettä. Monesti myös juuri tapahtunut liike kertoo, missä osassa kuvaa on jotain mielenkiintoista.

Optiset vuot voidaan jakaa tiheään ja harvaan vuohon. Tiheä optinen vuo on rakennelma, jossa jokaiseen kuvan pisteeseen liitetään tieto millä nopeudella mihin suuntaan kyseinen

piste on siirtynyt verrattuna edelliseen kuvaan. Käytännössä tiheän optisen vuon laskeminen on raskasta ja eletunnistuksen kannalta myös tarpeetonta, sillä olemme harvoin kiinnostuneita kuvan kaikkien pisteiden siirtymisistä.

Harvan optisen vuon menetelmät sen sijaan seuraavat tiettyjen ennalta määrättyjen pisteiden liikkumista kuvien välillä. Mikäli kuvilla on hyvien seurattavien piirteiden ominaisuuksia, niiden seuraaminen on yleensä melko luotettavaa ja (paremman suomenkielisen sanan puutteessa) robustia. Laskennallisessa mielessä harvan optisen vuon menetelmät ovat huomattavasti tiheää kevyempiä.

4.4 Eleiden tunnistaminen

Kun perättäisistä kuvista on kyetty tunnistamaan mielenkiintoiset alueet, olisi tieto näistä alueista jotenkin kyettävä yhdistämään tiedoksi tapahtuneista eleistä. Seuraavassa esitellään muutamia esimerkkejä tavoista, joilla liikkeitä voidaan tunnistaa.

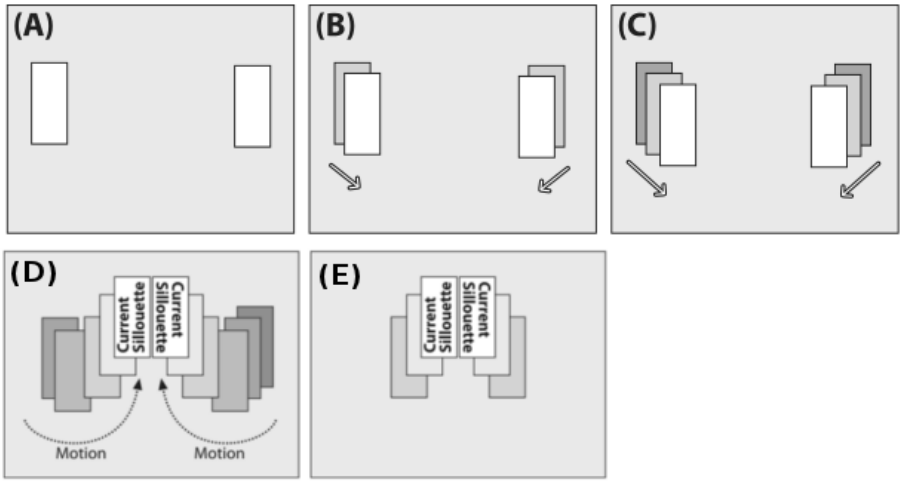
4.4.1 Liikemallit

Tämän alakohdan tiedot ovat peräisin lähteistä [5] ja [6] ellei muuta mainita.

Liikemallit (engl. motion templates) ovat tehokas työväline videokuvassa tapahtuvan liikkeen havaitsemiseen, seurantaan ja analysointiin. Käytännössä jokaisesta videovirran kuvasta muodostetaan siluettikuva, jossa kuvassa liikkuva tai seurattava alue on valkoinen ja muu alue mustaa. Tämä siluetti voidaan hankkia joko rajaamalla suoraan kuvasta tunnistettu etuala tai piirtämällä itse sellainen.

Oleellinen rakennelma liikemalleille on liikehistoriakuva. Aina kun videovirrasta saadaan poimittua yksittäinen siluettikuva, sen valkoinen osa ladotaan liikehistoriakuvaan uudeksi päällimmäiseksi tasoksi ja siihen liitetään tieto tallennuksen ajanhetkestä. Samalla kuvan alempien tasojen kirkkautta tummennetaan hieman. Näin kuvaan jää tieto miten siluettialue on liikkunut ajan funktiona. Kun siluetti on ollut liikehistoriassa kauemmin kuin jonkin ennalta määrätyn ajan t, se poistetaan rakennelmasta. Tätä toimintaa havainnollistaa kuvasarja 4.4, jossa valkoiset suorakaidesiluetit liikkuvat kuvan reunalta keskelle koukkaamalla alhaalta jättäen liikehistoriakuvaan jäljen liikkeistään. Viimeisessä kuvassa vanhentuneet siluetit on poistettu.

Näin syntyneestä kuvasta saadaan monia mielenkiintoisia asioita selville. Kun liikehistoriakuvaan on kertynyt muutaman kuvan mittainen historia, voidaan syntyneitä gradienttiliukumia tarkastelemalla havaita kuvasta sekä globaali liikevektori että lokaalit liikevektorit. Globaali liike on kaikkien liikehistoriakuvan osavektoreiden vektorisumma, eli



Kuva 4.4: Liikehistoriakuvan yksinkertaistettu toimintaperiaate. Muokattu lähteestä [5].

se kertoo mihin suuntaan tarkasteluvälillä kuvan liikemassa on siirtynyt. Lokaalit vektorit puolestaan lasketaan etsimällä kuvasta viimeisimmän uusimman aikaleiman omaavat eristetyt alueet ja tutkimalla millainen gradienttipolku siihen on johtanut.

Tietoa globaalista liikkeestä voidaan käyttää hyödyksi eleiden tunnistamiseen. Esimerkiksi Manchanda & Bing [32] ovat käyttäneet globaalia liikevektoria suorien liikkeiden tunnistamiseen seuraavasti. Liikehistoriakuvan päivityksen jälkeen piirretään ratakuvaan uusi piste globaalin liikevektorin osoittamaan suuntaan, mikäli tämä piste ei poikkea liikaa jo syntyneeltä radalta. Mikäli piste poikkeaa liikaa, koko rata hylätään. Jos kertynyt rata on riittävän pitkä (tai leveä), todetaan vaakasuora liike tapahtuneeksi.

Koska liikehistoriakuvaan on tallentuneena tieto syntyneestä liikkeestä, samanlaiset liikkeet synnyttävät melko samanlaisen, ja usein tunnistettavan liikehistoriakuvan. Tällöin myös tapahtunut liike voidaan tunnistaa koneoppimisen ja hahmontunnistuksen menetelmin. Esimerkiksi Davis [11] on saanut tällä tavoin hyviä tuloksia.

4.4.2 Rataseuranta

Rataseuranta on ehkä yksinkertaisin mahdollinen tapa tunnistaa käyttäjän liikkeitä. Kuvan etualasta tallennetaan halutut pisteet liikkeen sijainniksi kuva kuvalta. Näin kunkin pisteen kohdalla päätetään otetaanko uusi piste mukaan tunnistettavaan ratajoukkoon, nollataanko haluttu rata tai onko jokin tietty ele suoritettu.

Tällaisen tunnistimen toteuttaminen on yleisesti ottaen helppoa, mutta se on mahdollista toteuttaa vain ennalta määrätyille, suhteellisen yksinkertaisille eleille. Käyttäjä ei tällöin voi tallentaa omia, mielivaltaisia eleitä. Tapa ei myöskään ole akateemisesti kovin kiinnostava.

Mikäli nämä rajoitteet eivät ole esteenä, on tällainen rataseuranta erittäin toimiva vaihtoehto eletunnistukseen. Esimerkiksi Manchanda & Bing [32] ja Bernardes et al. [3] ovat käyttäneet tätä onnistuneesti.

4.4.3 Markovin piilomallit

Eräs mahdollinen tapa tunnistaa eleitä on käyttää perinteisiä aika-avaruuteen (engl. spatio temporal) soveltuvia tilastollisia hahmontunnistuksen työkaluja. Tässä käsitellään näistä erästä, Markovin piilomalleja (engl. Hidden Markov Model, HMM). Perinteisesti HMM:a on käytetty puheentunnistuksessa, mutta sen on huomattu soveltuvan myös eletunnistuksen välineeksi. Erityisen hyvin se soveltuu käyttötapauksiin, joissa käyttäjä voi opettaa luokitinta, tallentaa eleitä, tai halutaan, että eleiden toistamisessa voi olla variaatioita mutta ne on silti voitava tunnistaa. [30; 64; 53]

HMM on joukko tiloja, joiden välillä voidaan siirtyä perättäisillä ajanhetkillä, ja kuhunkin tilaan liittyviä havaintoja. Käytettävissä ei ole tietoa, mitkä tilasiirtymät kullakin ajanhetkellä tapahtuvat, vaan ainoastaan tilasiirtymien todennäköisyydet tunnetaan. Tästä tulee nimitys piilomalli. Koska kuitenkin tiedetään, millaisia havaintoja kustakin tilasta syntyy, ja näiden havaintojen esiintymistodennäköisyydet, on mahdollista tehdä arvauksia tapahtuneista tilasiirtymistä.

Käytettäessä HMM:ja, ollaan yleensä kiinnostuneita kolmen eri ongelman ratkaisuista. Ensinnä jos tiedetään jokin malli ja havaintosarja, miten lasketaan todennäköisyys, että kyseinen malli on tuottanut juuri tämän havaintosarjan? Toiseksi jos tiedetään malli ja havaintosarja, miten valitaan tilasiirtymäsarja, joka todennäköisimmin on tuottanut kyseisen havaintosarjan? Kolmanneksi miten muokataan mallin parametreja niin, että tietyn havaintosarjan esiintymistodennäköisyys kyseisen mallin kanssa maksimoituu? Kuhunkin ongelmaan on olemassa ratkaisunsa, joita käsitellään paremmin esimerkiksi lähteessä [48].

Matemaattisesti määriteltynä Markovin piilomalli on N kokoinen kokoelma tiloja $\mathbf{S} = \{s_1, s_2, ..., s_N\}$, joiden välillä voidaan siirtyä. Tilaa ajanhetkellä t merkitään q_t Kullakin tilalla j on kaksi todennäköisyysjoukkoa: siirtymätodennäköisyyden jakauma $\mathbf{A} = \{a_{ij}\}$ ja diskreetti havaintojen todennäköisyysjakauma $\mathbf{B} = \{b_j(k)\}$, joka kertoo ehdollisen todennäköisyyden että kyseisessä tilasta syntyy tietty havainto, joka kuuluu M kokoiseen havaintosymbolistoon $\mathbf{V} = \{v_1, v_2, ..., v_M;\}$. Lisäksi tarvitaan alkutilojen todennäköisyysjakauma π. Siirtymätodennäköisyyden ja havaintojen todennäköisyyksien jakaumissa:

$$\begin{aligned} a_{ij} &= P[q_{t+1} = S_j | q_t = S_i] \qquad 1 \leq i, j \leq N \\ b_j(k) &= P[v_k \text{ hetkellä } t | q_t = S_j] \quad j \in \mathbf{S}, k \in \mathbf{V}. \text{ [48]} \end{aligned}$$

Kannattaa huomata, että juuri tässä tapauksessa rajoitumme yksinkertaisuuden vuoksi käsittelemään vain diskreettejä havaintojen jakaumia. Todellisissa tapauksissa ulostulo on jatkuvaluonteista, jolloin havaintojen todennäköisyysjakauman sijasta käytetään havaintojen todennäköisyyden tiheysfunktiota, jonka arvot kuuluvat jatkuvalle satunnaisvektorille. Vaihtoehtoisesti havainnot voidaan myös kvantisoida diskreeteiksi. [48]

Havainnollistetaan tätä hyvin yksinkertaisella elemallilla, jolla voisi tunnistaa pyyhkäisyn oikealle. Mallin parametrit ovat listattuna taulukossa 4.1. Kun käyttäjä mallin mukaan on piirtämässä elettä, on malli tilassa *piirto*, muuten se on tilassa *ele*. Havaintoina pidetään suuntaa, johon elettä kullakin hetkellä piirretään, tai pysähtymistä. Käytännössä näin yksinkertainen malli ei ole järkevä, mutta se soveltuu esimerkkiin. Parempia malleja löytyy esimerkiksi lähteistä [64; 29].

$$\mathbf{A} = \begin{bmatrix} 0.8 & 0.2 \\ 0.05 & 0.95 \end{bmatrix} \quad \mathbf{V} = \{vasen, oikea, seis\}$$
$$\mathbf{S} = \{piirto, ele\}$$
$$\pi_{piirto} = 0.95 \quad b_{piirto}(vasen) = 0.3,\ b_{piirto}(oikea) = 0.6,\ b_{piirto}(seis) = 0.1$$
$$\pi_{ele} = 0.05 \quad b_{ele}(vasen) = 0.3,\ b_{ele}(oikea) = 0.1,\ b_{ele}(seis) = 0.6$$

Taulukko 4.1: Oikealle pyyhkäisevän eleen tunnistavan mallin parametrit.

Parametreista on melko intuitiivista huomata, että piirto jatkuu todennäköisesti piirtona ja voi myös päättyä eleeksi, mutta ele ei vaihdu piirtotilaan kuin hyvin pienellä todennäköisyydellä. Mallia käytettäessä ei kuitenkaan "nähdä" näitä siirtymiä vaan kustakin tilasta tehtyjä havaintoja, eli havaittaessa siirtymä oikealle, luultavasti edelleen piirretään ja havaittaessa pysähtyminen, ollaan luultavasti siirrytty eletilaan. Taululukossa 4.2 on kirjattuna joitakin Viterbi-algoritmilla [48; 61] saatuja esimerkin mallin "parhaiten" selittämiä havaintojonoja. Kuten huomataan, on parhaimmankin tapauksen todennäköisyys hyvin pieni (n. 7%), mutta tämä johtuu mallin kehnoista (lähes arvotuista) parametriarvoista.

Havaintojono	Tilajono	Todennäköisyys
$(oikea, seis)$	$(piirto, ele)$	0.0684
$(oikea, oikea, seis)$	$(piirto, piirto, ele)$	0.0328
$(oikea, vasen, seis)$	$(piirto, ele, ele)$	0.0195
$(oikea, oikea, vasen, oikea, seis)$	$(piirto, piirto, piirto, piirto, ele)$	0.0038

Taulukko 4.2: Eräitä taulukon 4.1 HMM-mallin selittämiä havaintojonoja.

5. KONENÄKÖPOHJAINEN ELEOHJAUSKIRJASTO

Työn teknisenä kontribuutiona toteutettiin yleiskäyttöinen konenäköön perustuva eleohjauskirjasto. Vaatimuksena kirjastolle oli toimiminen jossakin mobiililaitteessa ja muutaman eleen suurpiirteinen tunnistaminen. Lisäksi työssä toteutettiin pienimuotoinen esityksen ohjaamiseen tarkoitettu sovellus, joka käyttää kyseistä kirjastoa hyödykseen.

5.1 Qt-ohjelmistokehys

Qt on poikkialustallinen sovelluskehityskehys (engl. cross platform application development framework). Tämä poikkialustallisuus toimii Qt:ssa “kirjoita kerran, käännä kaikkialle” -periaatteella, eli samaa ohjelman lähdekoodia voidaan käyttää kaikilla tuetuilla alustoilla kunhan sen vain kääntää uudelleen. Tämän jälkeen sovellusta voidaan ajaa natiivissa ympäristössä ilman esimerkiksi Javan vaatimaa virtuaalikonetta tai C#:n kanssa tyypillisesti käytettyä välikooditulkkia. Tämän kirjoitushetkellä viimeisin julkaistu vakaa versio Qt:sta on 4.8, ja se tukee seuraavia alustoja [42]:

- Windows (työpöytä)
- Linux/X11 (työpöytä)
- Mac OS X (työpöytä)
- Embedded linux (sulautettu/mobiili)
- Symbian (sulautettu/mobiili).

Qt on ennen kaikkea tunnettu graafisten käyttöliittymien sovelluskehyksestään, mutta se tarjoaa runsaasti muitakin hyödyllisiä komponentteja sovelluskehittäjän käyttöön. Tällaisia ovat esimerkiksi tässäkin työssä käytetyt monisäikeisyys ja rinnakkaisuus, liitännäiset (engl. plugins) ja signaalit ja lokerot -menetelmä (engl. signals and slots) olioiden välisen kommunikointiin. [41]

5.2 QtQuick

QtQuick koostuu QML(Qt Meta Language) -kuvauskielestä, QtDeclarative -moduulista ja QtCreator työkalusta. QML on yksinkertainen kieli, joka on suunniteltu dynaamisten, näyttävien ja sujuvien (engl. fluid) käyttöliittymien toteuttamiseen kosketusnäytöllä. Koska kieli on alusta alkaen tehty tätä tarkoitusta varten, on tällaisten käyttöliittymien kirjoittaminen helppoa ja nopeaa. Se perustuu komponenteille ja niihin sidottuihin ominaisuuksiin, jotka kuvaillaan tai esitellään (engl. declare). Tämä on ehkä helpointa ymmärtää kuvan 5.1 erittäin yksinkertaisella esimerkkiohjelmalla. Ohjelmassa on vaaleankeltainen suorakulmio, jonka keskellä on tekstiä. Suorakulmion korkeus on määritelty eksplisiittisesti, mutta sen leveys on sidottu korkeuteen. Samoin tekstin pikselikoko on sidottu korkeuteen. Tällöin, kun korkeutta muutetaan vaikkapa venyttämällä ikkunaa, siihen sidotut arvot muuttuvat automaattisesti. [57; 40]

```
import QtQuick 1.0
Rectangle {
    height: 100
    width: 1.5 * height
    color: "lightyellow"
    Text {
        anchors.centerIn: parent
        font.pixelSize: parent.height * 0.25
        text: "Hello World"
    }
    MouseArea {
        anchors.fill: parent
        onClicked: Qt.quit()
    }
}
```

Kuva 5.1: Vasemmalla: esimerkkiohjelma kirjoitettuna QML:llä. Oikealla sama ohjelma suorituksessa eri kokoisiksi venytetyissä ikkunoissa.

QtDeclarative puolestaan eräs Qt:n C++ -moduuli. Sen avulla voidaan QML:llä kirjoitettuja käyttöliittymiä yhdistää C++:lla kirjoitettuihin muihin ohjelman osiin. Vastaavasti QtDeclarativen avulla voidaan QML-ohjelmia laajentaa C++:n avulla. Esimerkiksi kuvan 5.1 koodissa kutsu `Qt.quit()` on periaatteessa juuri tällainen laajennus. Siinä kutsutaan ensimmäisen rivin `import` -lauseen mukana ohjelmaan tuodun globaalin `Qt` -olion `quit()` -funktiota.

5.3 OpenCV -konenäkökirjasto

Konenäön toteuttamiseksi työssä nojauduttiin vahvasti OpenCV -konenäkökirjaston puoleen. OpenCV on alun perin lähtöisin Intel Research tutkimuslaboratorioista ja se on alusta asti rakennettu täyttämään muutama tehtävä. Ensiksi sen tarkoitus on edistää konenäkötutkimusta tarjoamalla avoin mutta optimoitu lähdekoodi, jonka päälle rakentaa konenäköjärjestelmiä; konenäön parissa työskentelevien ei tulisi joutua “keksimään pyörää uudestaan”. Toiseksi kirjaston tehtävä oli levittää konenäköosaamista ja -tietoa tarjoamalla pohja, jonka päälle rakennettuja järjestelmiä voisi käyttää muissakin konenäköjärjestelmissä. Kolmanneksi sen piti edistää kaupallisten konenäkösovellusten syntymistä tarjoamalla tehokkaaksi optimoitu alusta ilmaiseksi niin että lähdekoodikin on saatavilla, mutta lisensoitu niin ettei kaupallisten sovellusten lähdekoodia tarvitse julkaista. [5]

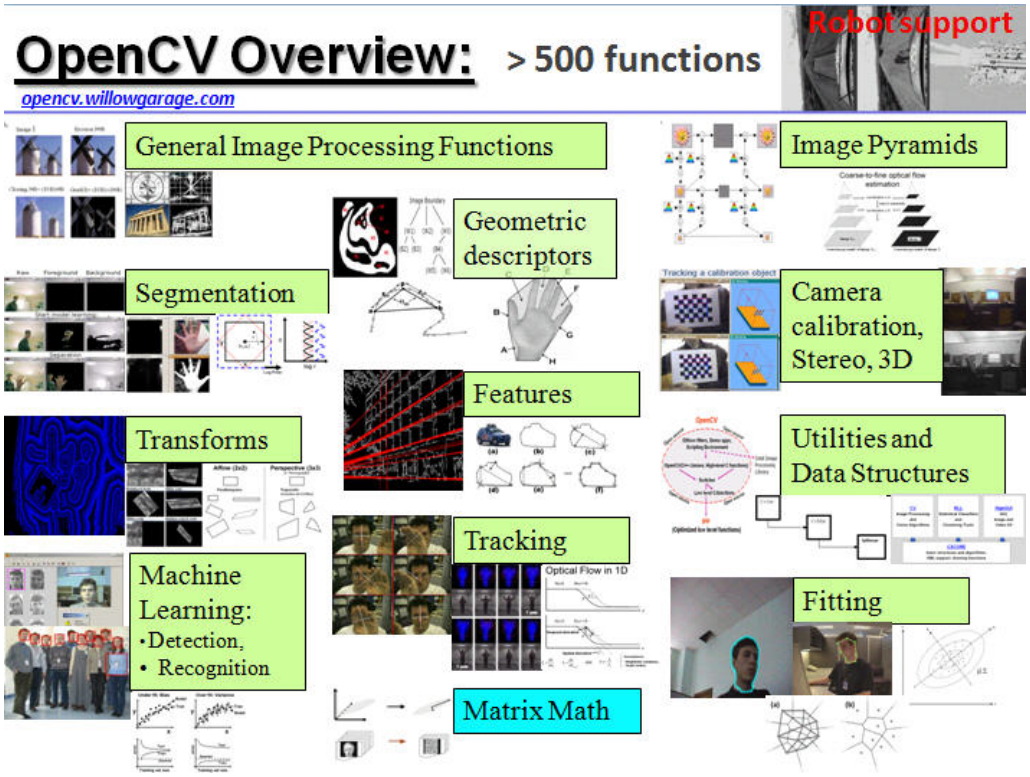
OpenCV on 1999 julkaisunsa jälkeen selvästi saavuttanut tavoitteensa, sillä toisaalta kirjasto on hyvin laajassa käytössä niin tieteellisen tutkimuksen parissa kuin myös kaupallisissa sovelluksissa. Itse asiassa nykyään on huomattavan vaikeaa löytää konenäköprojektia, jossa perustuisi jollekin muulle kuin OpenCV:lle. Kuva 5.2 antaa melko hyvän kuvan käyttökohteiden moninaisuudesta. Kirjaston nauttimasta suosiosta kertoo myös sen laajennettavuus useille ohjelmointikielille. Vaikka se on alun perin kirjoitettu C- ja C++ -kielillä, sille löytyvät rajapinnat myös esimerkiksi pythonille, rubylle ja matlabille, ja tätä kirjoittaessa Java-rajapinta on työn alla. Nykyään OpenCV kattaa yli 2500 optimoituja algoritmia konenäön parista. Sitä on ladattu yli 2.5 miljoonaa kertaa ja sillä on yli 40000 käyttäjää. Esimerkkejä käyttökohteista ovat vaikkapa videovalvonta, interaktiivinen taide, panorama-kuvien “liimaaminen” ja robotiikka. [44]

5.4 Kohdelaitteisto

Työssä käytettäväksi mobiililaitteeksi valittiin Nokian N900 -multimediatietokone. N900 on Nokian Maemo 5 -alustalla toimiva älypuhelimen ja Internet-tabletin yhdistävä laite. N900 valittiin kohdelaitteeksi sen Linux-pohjaisuuden, helpon ohjelmoinnin ja käytössä olevien kameroiden vuoksi.

Laitteen takana oleva "pääkamera"kykenee tallentamaan 848x480 resoluutioista ja edessä oleva "kakkoskamera"640x480 -resoluutioista videota 25 kuvan sekuntivauhdilla. Tämän vastaa hyvin olemassa olevien verkkokameroita hyödyntävien konenäköjärjestelmien vaatimuksia, joten laitteen kameroiden pitäisi olla riittävän hyvät työtä ajatellen.

Nokia on tehnyt runsaasti töitä, jotta ohjelmien ja kirjastojen kirjoittaminen N900:lle olisi mahdollisimman helppoa. Pääasiallisena ohjelmointikehyksenä N900-laitteen kehitystyössä käytetään Qt-ohjelmistokehystä, joka poikkialustallinen kirjasto. Käytännössä tuki



Kuva 5.2: Katsaus OpenCV:n käyttökohteisiin [44].

usealle alustalle mahdollistaa ohjelmien kehittämisen ja testaamisen pöytäkoneella ja vasta lopuksi koostaa ja asentaa itse laitteelle. Tämä on omiaan helpottamaan kehitystyötä.

Linux-pohjaisuus mahdollistaa OpenCV:n helpohkon käytettävyyden laitteella ja on muutenkin työn tekijän henkilökohtaisen mieltymyksen kohde. N900:ssa on käytettävissä V4L (Video for Linux) rajapinta, jota OpenCV osaa käyttää videovirran lukemiseen kameroilta.

5.5 Arkkitehtuurikuvaus

Toteutetun eleohjauskirjaston melko yksinkertainen arkkitehtuuri jakautuu yleisesti kahteen osaan: kehysluokkiin, jotka tarjoavat puitteet järjestelmän toteuttamiselle ja toteutusluokkiin, jotka toteuttavat taustanpoiston ja eleiden tunnistuksen.

Kehysluokat koostuvat `GestureTracker`ista, jolla on yksi `BackgroundSubtractor`, joitakin `GestureDetector` -olioita, yksi `CaptureThread` ja yksi `ContourStorage` -olio. Nämä luokat vielä yleisesti periytetään `GestureRecognitionBase` -luokasta. Seuraavassa käydään tarkemmin läpi luokkien toimintaa. UML-entiteettikaavio kirjaston oleellisista luokista ja tärkeimmistä funktioista on kuvassa 5.3.

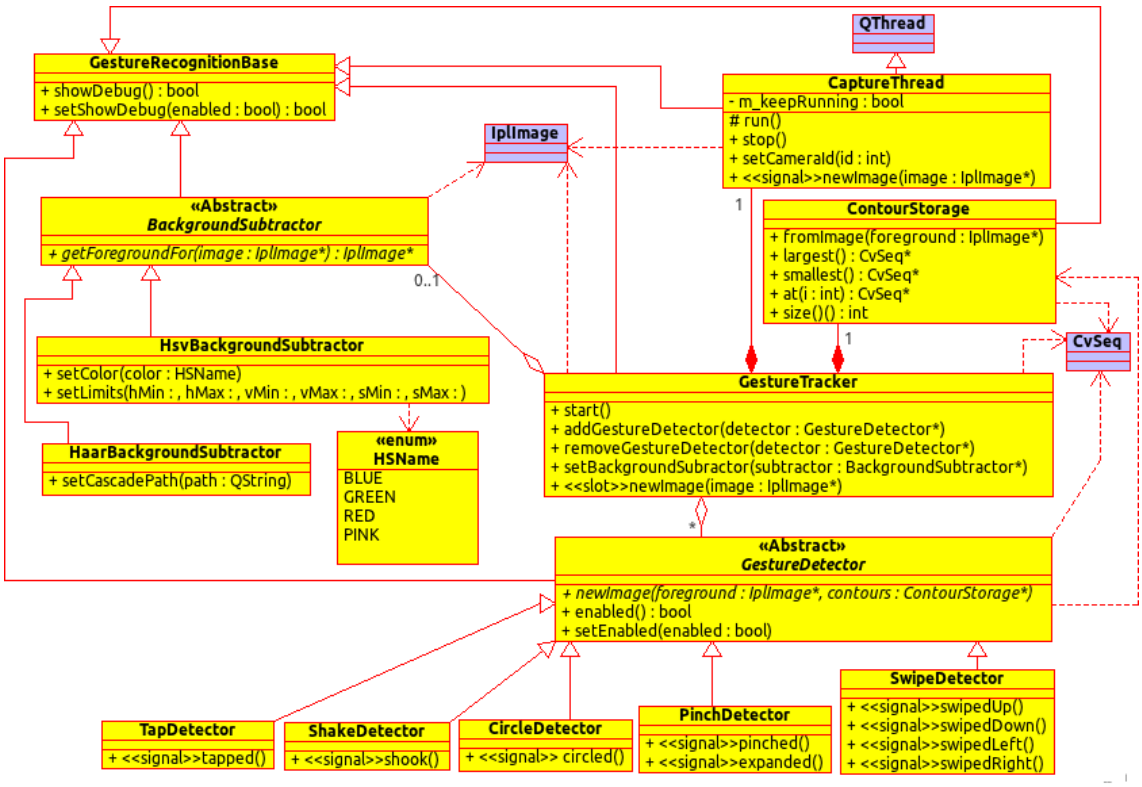
Kehysluokat toimivat yhdessä niin, että `CaptureThread` lähettää otetun kuvan

`GestureTracker`ille, joka kysyy `BackgroundSubtractor`ilta saadun kuvan etualan, antaa `ContourStorage`lle etualakuvan ääriviivojen etsimistä varten ja lopuksi antaa kuvan jokaiselle käytössä olevalle `GestureDetector`illeen. Kuvan 5.4 sekvenssikaavio havainnollistaa tätä kehysluokkien yhteistyötä yhden kameralta saadun kuvan käsittelyssä.

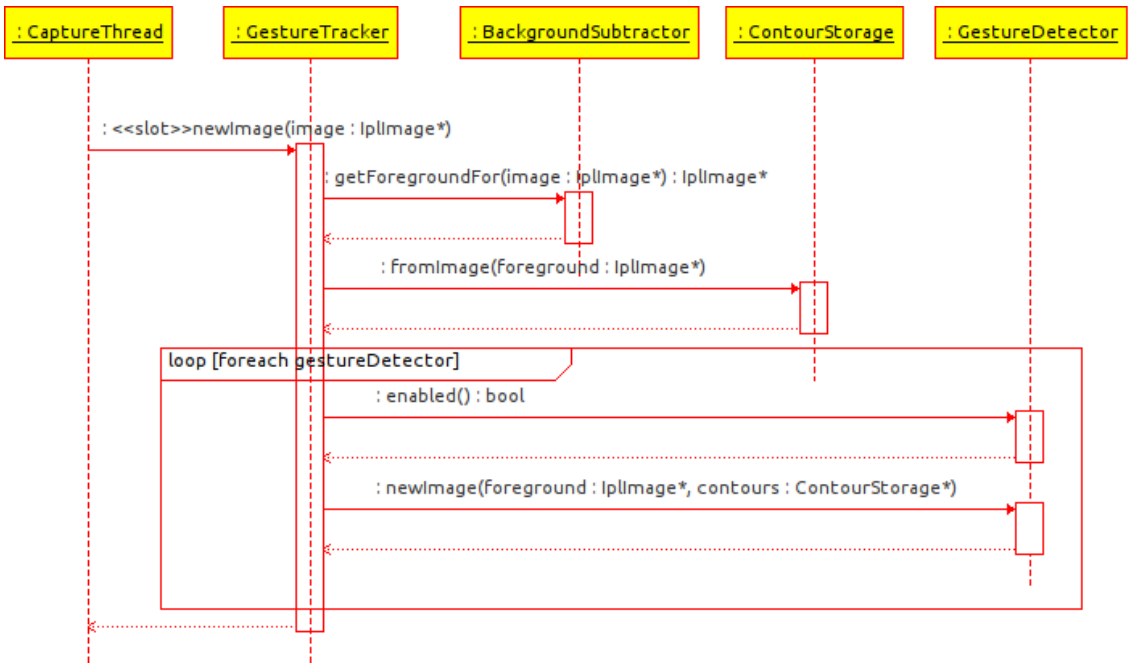
5.5.1 Kehysluokat

`GestureRecognitionBase` on kantaluokka, josta kaikki järjestelmään toteutettavat luokat on periytettävä. Sen tärkein tehtävä tällä hetkellä on tarjota funktiot `setShowDebug(bool enabled)` ja `bool showDebug()`, jotka määräävät, saako kyseinen luokka luoda uusia ikkunoita, joihin piirtää omien vastuualueidensa kannalta oleellisia kuvia virheenetsimistä tai luokan toiminnan havainnollistamista varten.

`CaptureThread` -luokka periytyy `QThread` -luokasta, joten se on käytännössä säie. Sillä on kolme oleellista funktiota: `void run()`, `void stop()` ja `void setCameraId(int id)`. Lisäksi `CaptureThread`illa on signaali `newImage(IplImage*)`, jolla se tarjoaa kaappaamansa kuvan kiinnostuneille tahoille. `SetCameraId(int id)` kertoo `CaptureThread`ille, mistä kameralähteestä kuvia kaapataan. Mikäli käytössä on vain yksi



Kuva 5.3: Kirjaston luokkakaavio. Luettavuuden parantamiseksi joitakin funktioita on jätetty pois kuvasta.



Kuva 5.4: Sekvenssikaavio yhden saadun kuvan käsittelystä kirjaston kehysluokkien kesken.

kamera, on tämä lähde 0, muuten se on nollasta alkava kameran järjestysnumero. Jäsenfunktio `run()` toteuttaa `QThread`in säiesilmukan, eli kun säie käynnistetään, kutsutaan `run()`:a, ja säikeen suoritus loppuu kun `run()` palaa. Jäsenfunktio `stop()` puolestaan lopettaa kaappauksen ja säiesilmukan toiston. Käytännössä `stop()` kääntää `bool` tyyppisen jäsenmuuttujan `m_keepRunning`, jonka arvon `run()` joka kierroksensa alussa tarkastaa ja lopettaa suorituksen kun se muuttuu.

`GestureTracker` -luokan tehtävä on liittää kirjaston osat yhteen. Se alustaa ja ottaa käyttöön `CaptureThread` -olion, jolta saadut kuvat se ohjaa ensin kuvan etualan etsimistä varten `BackgroundSubtractor`eilleen ja sitten `GestureDetector`eilleen etualatietoineen eleiden tunnistamista varten. Jäsenfunktioita `addGestureDetector()`, `removeGestureDetector()` ja `setBackgroundSubtractor()` käytetään vastaavien luokkien käyttöönotoissa. Jäsenfunktio `start()` käynnistää `CaptureThread` -säikeen. Lokero (engl. slot) `newImage(IplImage*)` on yhdistetty `CaptureThread`in vastaavaan signaaliin, joten se on kuvankaappauksen jälkeen seuraava kohta eletunnistusprosessin toteutuksessa.

`ContourStorage` -luokan tehtävä on tunnistaa, lajitella ja tarjoilla etualan maskikuvasta siluettien ääriviivoja (engl. contour). Ääriviivat ovat oleellinen osa konenäköjärjestelmää, sillä ääriviivat – toisin kuin ulkonäköön perustuvat maskikuvan siluetit – sisältävät numeerista ja siten ohjelmallisesti käsiteltävää tietoa kuvasta. Yksittäinen jäsenfunktio on tallennettuna OpenCV:n `CvSeq` -rakenteeseen. Lisää tietoa ääriviivoista, niiden hakemisesta ja käsittelystä OpenCV:n avulla löytyy esimerkiksi lähteestä [5]. Jäsenfunk-

tio `fromImage(IplImage*)` ottaa maskikuvan etualasta, etsii niistä ääriviivat ja tallentaa löytyneet ääriviiva-alueet suuruusjärjestyksessä. Luokalta voi kysellä ääriviivoja jäsenfunktiolla `at(int i)`, missä `i` on nollasta alkava järjestysluku ja parametrilla `i==0` se palauttaa pinta-alaltaan suurimman ääriviivan. Saatavilla olevien ääriviivojen määrän kertoo jäsenfunktio `size()`. Mukavuusfunktiot `largest()` ja `smallest()` antavat suurimman ja pienimmän ääriviivan.

`BackgroundSubtractor` on abstrakti kantaluokka, josta kaikki järjestelmään toteutettavat taustan poistoon tarkoitetut luokat on periytettävä. Rajapinnan oleellisin funktio on `IplImage *getForegroundFor(IplImage* image)`. Se ottaa parametrikseen osoittimen IplImageen (OpenCV:n käyttämä rakenne kuvan esittämiseen), joka esittää kuvaa, josta etuala pitäisi löytää, ja palauttaa maskikuvan, jossa kuvan etuala on valkoinen ja taka-ala mustaa.

`GestureDetector` on abstrakti kantaluokka, josta kaikki järjestelmään toteutettavat eleentunnistimet on periytettävä. Tärkein luokan funktio `newImage(IplImage* foreground, ContourStorage* contours)`, jolla syötetään tieto löydetystä etualasta eleentunnistimelle. Kyseisen funktion parametri `foreground` sisältää etualan maskikuvan ja `contours` samasta maskikuvasta löydetyt ääriviivat, joita hyväksi käyttämällä toteutetun eleentunnistimen on suoritettava tehtävänsä. Maskikuva tarjotaan, koska eleentunnistin saattaa haluta tehdä ääriviivojen tunnistamisen uudelleen. Lisäksi siitä löytyy myös alkuperäisen kuvan resoluutio. Jäsenfunktioilla `enabled()` sekä `setEnabled(bool)` voidaan asettaa tunnistin aktiiviseksi ja kysyä aktiivisuus tilaa. `GestureTracker` kutsuu `newImage(...)`:a vain aktiivisena oleville tunnistimille.

5.5.2 Toteutusluokat

Toteutusluokat ovat luokkia, jotka mahdollistavat järjestelmän laajentamisen lisäämällä uusia menetelmiä taustan poistamiseksi ja eleiden tunnistamiseksi. Tässä työssä toteutettiin värierotteluun ja Haar-tyyppisiin piirteisiin perustuvat taustanpoistoluokat, ja erinäisiä eleitä tunnistavia luokkia.

`HsvBackgroundSubtractor` on pääasiallinen etu- ja taka-alan erotteluun tarkoitettu luokka. Sen rajapinnasta löytyy paitsi kaikkien `BackgroundSubtractor`eiden toteuttama `getForegroundFor(IplImage*)`, myös työkalut haluttujen värirajausarvojen asettamiseen. Jäsenfunktio `setLimits(hMin, hMax, vMin, vMax, sMin, sMax)` asettaa tarkalleen käyttäjän määrittelemät rajausarvot; sen sijaan mukavuusfunktio `setColor(HSName color)` mahdollistaa muutamien `HSName` -enumeraatiossa lueteltujen värien asettamisen.

HaarBackgroundSubtractor käyttää Haar-tyyppisiä piirteitä etualan tunnistamiseksi. Luokan tärkein funktio on `setCascadePath(QString path)`, joka määrittelee polun, josta erotin löytää opetetun luokitinkaskadin [5].

`GestureDetector`ista periytetyillä **SwipeDetector**illa, **PinchDetector**illa, **CircleDetector**illa, **ShakeDetector**illa ja **TapDetector**illa ei ole rajapinnoissaan toteuttamiensa `newImage(IplImage*,ContourStorage*)`n lisäksi muita funktioita. Kullakin kuitenkin on omat omaan eleeseensä liittyvät signaalit, jotka ne lähettävät kun ovat tunnistaneet eleen. Kirjaston käyttäjän vastuulla on yhdistää oma ohjelmakoodinsa näihin signaaleihin.

5.6 Esikäsittely

Toteutetussa järjestelmässä kuvaa ei juurikaan esikäsitellä. Tärkeimmät operaatiot ovat kuvan pienentäminen ja peilaaminen vaakatasossa. Pienentämisen tarkoitus on parantaa suorituskykyä, sillä 640x480 -kokoisen kuvan käsittely N900:n laitteistolla osoittautui liian hitaaksi, myös kehityskoneena käytetyllä pc-tietokoneella havaittiin kuvan pienentämisen parantavan suorituskykyä. Käytännössä eletunnistuksen kannalta riittäväksi resoluutioksi huomattiin pienentäminen 320x240 -kokoon. Pienentäminen toteutettiin OpenCV:n `cvResize` -operaatiolla.

Kuvan peilaaminen on puolestaan täysin kehitystyön helpottamista varten. Kun verkkokameralla itsestä otettua kuvaa katsoo kuvaruudulla, ei näe itseään peilikuvaa, vaan näkee itsensä ikään kuin vastapäätä katsovan ihmisen silmin. Toisin sanoen, kun käyttäjä liikuttaa kättään oikealle, kuvaruudulla näkyvä kuva liikkuukin vasemmalle. Toisaalta pystytasossa tätä ei tapahdu, joten käyttäjä joutuu käsittelemään ristiriitaista tietoa. Vaakatason peilaaminen helpottaa siis kehitystyötä.

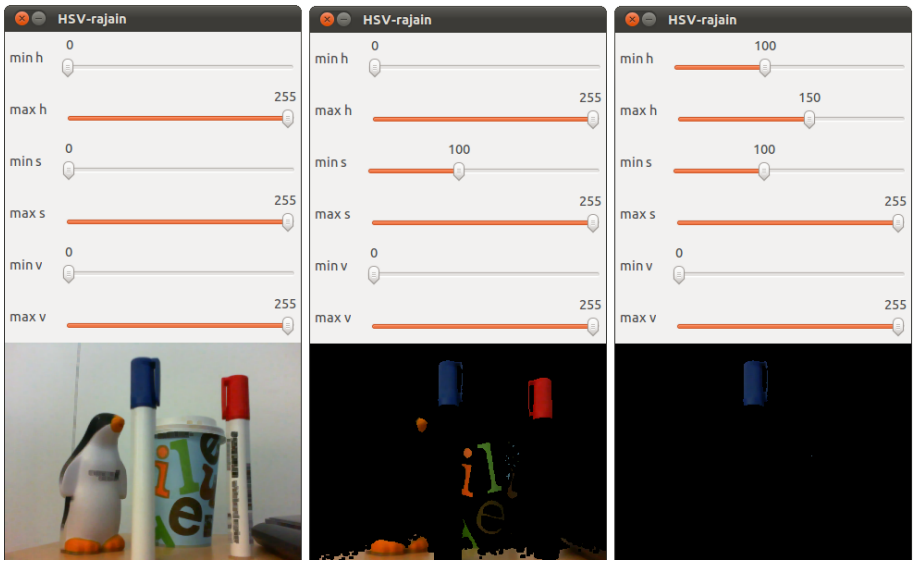
5.7 Taustan poistaminen OpenCV:llä

Työssä käytettiin pääsiallisena työkaluna taustan poistamiseen värierottelua sen nopeuden vuoksi. Kuvan kukin piste vaatii vain yhden vertailun, joten X x Y -kokoisen kuvan värierottelun asymptoottinen nopeus on $\Theta(XY)$. Työn arkkitehtuuri kuitenkin mahdollistaa muidenkin taustan poistajien toteuttamisen ja käyttämisen. Kuriositeettina ja mielenkiinnon vuoksi työssä toteutettiinkin myös toinen, Haar-tyyppisten piirteiden tunnistamiseen perustuva taustan poistaja. Tämä on kätevää sillä OpenCV:n lähdekoodipaketin mukana tulee esimerkiksi kasvojen tunnistukseen soveltuva luokitin, joilla kyseistä taustanpoistajaa on helppo testata. Enemmän tietoa Haar-tyyppisiin piirteisiin perustuvasta kohteen tunnistamisesta OpenCV:llä löytyy esimerkiksi lähteestä [5].

Värirajaus on yksinkertainen toteuttaa, mutta se pitää tehdä itse, sillä OpenCV ei tarjoa kyseistä palvelua. OpenCV:n `IplImage` -luokan rajapinnasta löytyy kaksi oleellista jäsenmuuttujaa: `char* imageData` ja `int widthStep`. `ImageData` on osoitin jatkuvaan muistialueeseen, johon kuvapisteiden "raakadata" on tallennettu. HSV-värimallin tapauksessa perättäisistä muistipaikoista löytyy värisävyn, -kirkkauden ja -kylläisyyden arvot. `widthStep` puolestaan kertoo kuinka monta HSV-kolmikkoa yhdellä rivillä on. Mikäli kyseessä on yksikanavainen harmaasävykuva, ovat kunkin pisteen kirkkausarvot (V) peräkkäisissä muistipaikoissa.

Tällä kirkkausarvojen sijaintien logiikalla on merkitystä niin sanotun maskikuvan luomisessa. Maskikuva on yksikanavainen mustavalko- tai harmaasävykuva, joka on alkuperäisen kuvan (josta etu- ja taka-ala erotellaan) kokoinen. Taustanpoistossa eroteltu tausta on maskikuvassa musta (arvolla 0) ja etuala valkoinen maksimiarvo (tämän työn toteutuksessa 255). Nyt kun verrataan alkuperäisen kuvan pisteiden arvoja rajausarvoihin, löytyvät arvot muistipaikasta `(imageData + widthStep*y)[x*3]`, missä `x` ja `y` ovat tutkittavan kuvapisteen x- ja y-koordinaatit. Maskikuvassa vastaava muistipaikka on `(imageData + widthStep*y)[x]`, koska kyseessä on vain yksikanavainen kuva.

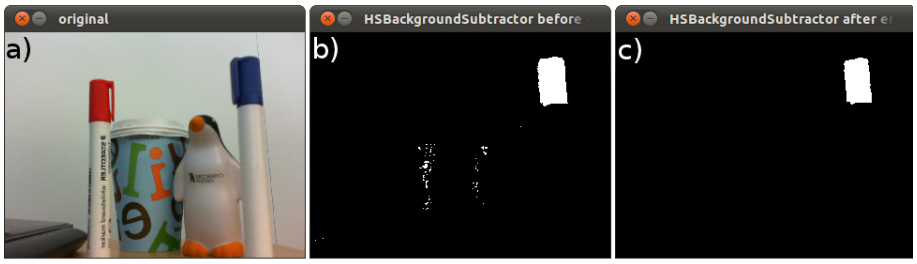
Koska väriin perustuva rajaaminen on melko herkkä ympäristön valaistukselle, voi usein tulla tarpeen säätää rajausparametreja. Tätä varten työssä toteutettiin myös yksinkertainen työkalu, joka näyttää kameran ottamaa kuvaa ja tarjoaa käyttäjälle liukusäätimet, joilla rajausparametreja voi muokata käsin. Kuva 5.5 havainnollistaa työkalun toimintaa.



Kuva 5.5: Kuvasarja HSV-arvoihin perustuvaan rajaamiseen tarkoitetusta työkalusta. Vasemmassa reunassa on alkuperäinen kuva ilman värirajaamista, keskellä on osittain rajattu kuva ja oikeassa reunassa on tilanne, kun väriarvot on saatu rajattua sinisen tussin (vasemmalla) korkkiin.

Koska värirajaus perustuu vain kuvan "ulkonäköön", eikä mihinkään tilastolliseen malliin, rajauksen jälkeen kuvassa voi mahdollisesti ensisijaisen rajauksen jälkeen jäädä vielä

kohinankaltaisia jäänteitä muista läheisesti samanvärisistä kohteista. Nämä ovat erityisen harmillisia, mikäli kuvassa ei edes pitäisi esiintyä varsinaista etualaa. Tällöin tämä kohina saattaa aiheuttaa virheellistä toimintaa. Kuva 5.6b havainnollistaa kuvaan jäänyttä "värikohinaa".



Kuva 5.6: Kuvasarja värirajauksen jälkeen etualaan jäävästä kohinasta: a) alkuperäinen kuva, b) värirajauksen jälkeen, ennen `cvErode`a ja c) `cvErode`n jälkeen.

Näistä ylimääräisistä kohinapisteistä päästään eroon käyttämällä saatuun maskikuvaan OpenCV:n `cvErode` -operaatiota. `cvErode` "pyyhkii" kuvasta mustaksi sellaisia kirkkaita alueita, jotka ovat valmiiksi mustan ympäröimiä tai reunustamia mutta jättää pyyhkimättä suuremmat yhtenäiset kirkkaat alueet. Kuva 5.6c havainnollistaa kuvaan jäänyttä "värikohinaa" ja sen poistamista `cvErode`n avulla.

5.8 Eleiden tunnistaminen

Kirjaston halutaan tunnistavan seuraavat eleet: pyyhkäisy neljään suuntaan, pyöritys, ravistus, painaminen ja nipistys ja levittäminen. Seuraavissa kohdissa määritellään, miten käyttäjä suorittaa tämän työn eleet, ja tavat joilla eleiden tunnistus on toteutettu.

Tosimaailmassa – eli kontrolloitujen "laboratorio-olosuhteiden" ulkopuolella – eräs tärkeimmistä konenäköpohjaisen järjestelmän toteuttamisessa huomioon otettavista asioista on eleiden alku- ja loppumääritys. Pitää siis ottaa huomioon, että käyttäjä voi olla jatkuvasti kameran edessä, mutta kaikkia satunnaisia huitomisia ei kuitenkaan pidä olettaa eleiksi. Tämän vuoksi etenkin yksinkertaisissa eleissä tulisi olla jonkinlainen mekanismi, jolla eletunnistus aloitetaan. Vastaavasti yleisen käytettävyyden kannalta pitäisi olla mekanismi, jolla jo aloitettu eletunnistusseuranta lopetetaan. Toteutetussa kirjastossa tämän suhteen kahdenlaisia eleitä: sellaisia, jotka vaativat seurannan aloittamiseksi jotakin ja sellaisia jotka ovat jatkuvassa seurannassa.

5.8.1 Pyyhkäisy

Pyyhkäisyeleessä käyttäjä pitää seurattavaa kohdetta hetken paikallaan ja sitten siirtää sitä joko ylös, alas, vasemmalle tai oikealle. Alun paikallaan pitämisen tarkoitus on vähentää

vääriä tunnistuksia käyttäjän huomaamattaan liikuttaessa seurattavaa kohdetta kameran edessä. Eleen seuranta lopetetaan, kun ele on tunnistettu, tai kun käyttäjä vie seurattavan kohteen hetkeksi kameran kuva-alueen ulkopuolelle.

Pyyhkäisy tunnistetaan seuraavasti. Tunnistin laskee jokaisella kierroksella aluksi pienimmän suurinta reunaviiva-aluetta peittävän ympyrän keskipisteen p ja ottaa sen sijainnin talteen kiertopuskuriin. Sijainnin määrittämiseen käytetään OpenCV:n `cvMinEnclosingCircle` -operaatiota. Mikäli keskipisteet ovat olleet "suunnilleen paikallaan" N perättäisen kuvan ajan, aloitetaan varsinainen eleen tunnistamiseen tähtäävä seuranta. "Suunnilleen paikallaan" määritellään laskemalla sijaintipisteiden x- ja y-koordinaattien keskihajonnat σ_x ja σ_y. Käytännössä hyvin toimiviksi arvoiksi osoittautuivat $N = 10$, $\sigma_x = \lceil \frac{leveys}{100} \rceil$ ja $\sigma_y = \lceil \frac{korkeus}{100} \rceil$.

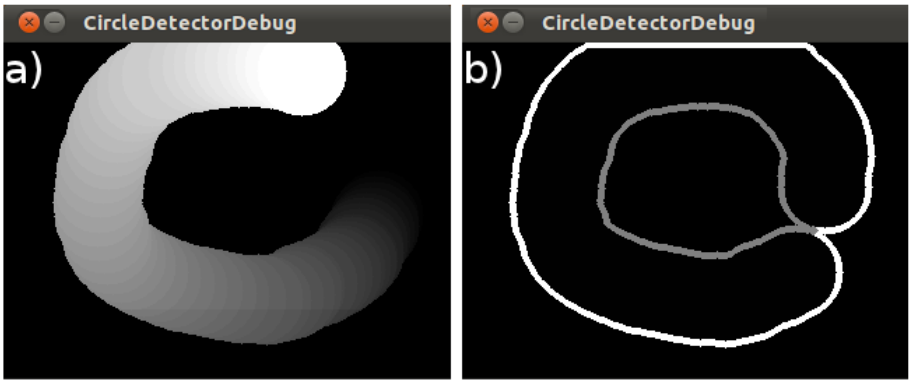
Seurantaprosessissa luodaan jokaisella kierroksella väliaikainen kuva, johon piirretään ympyrä keskipisteenään p ja säteenään $\frac{leveys}{10}$. Tämä kuva päivitetään liikehistoriakuvaan, minkä jälkeen kuvasta otetaan talteen globaalin liikkeen suunta, joka kvantisoidaan neljään mahdolliseen suuntaan: ylös, alas, vasemmalle ja oikealle. Kvantisoitu suunta tallennetaan kiertopuskuriin. Mikäli puskuriin on tallennettu alle M suuntaa, lopetetaan kierros. Muuten lasketaan kunkin neljän suunnan puskurissa olevat frekvenssit ja mikäli jonkin suunnan frekvenssi on yli 80% puskurin koosta, todetaan tapahtuneeksi pyyhkäisy kyseiseen suuntaan. Melko hyvin toimiviksi M arvoiksi osoittautui $M \in [10..15]$.

5.8.2 Pyöritys

Pyörityseleessä käyttäjä liikuttaa seurattavaa kohdetta ympyrän muotoisella radalla kameran edessä. Ele tunnistuu, kun ympyrärata on valmis ja seuraavaa ympyräelettä varten tulee piirtää uusi ympyrä. Ympyräradan säteellä ei ole suurta merkitystä, joten kaari voi olla suuri tai pieni. Radan ei myöskään tarvitse olla välttämättä ympyrän muotoinen; riittää että se muodostaa suljetun piirin. Pyörityksen suunnalla ei ole väliä.

Pyörityseleen tunnistamisessa käytetään myös hyödyksi liikehistoriakuvaa, mutta siitä ei tällä kertaa etsitä liikesuuntia, vaan ollaan ennemmin kiinnostuneita kuvan etualan jälkeensä jättämästä "vanasta". Eletunnistusprosessin jokaisella kierroksella lasketaan ensin ääriviivatiedoista peittävän ympyrä keskipiste p kuten pyyhkäisyeleessäkin. Tämän jälkeen piirretään väliaikaiseen kuvaan $\frac{leveys}{10}$ -säteinen ympyrä, ja tämä väliaikainen kuva päivitetään liikehistoriakuvaan. Liikehistoriakuvassa näkyy nyt siis etualan liikevana viimeiseltä T sekunnilta (kuva 5.7a). Tunnistuksen kannalta hyväksi T:n arvoksi osoittautui $T = 2$.

Seuraavaksi liikehistoriakuvasta etsitään OpenCV:n `cvFindContours` -operaatiolla ää-



Kuva 5.7: a) Liikehistoriakuvassa näkyvä liikevana. b) Ääriviivakuva samasta liikkeestä hetkeä myöhemmin, kun pyöritysliike on valmistunut.

riviivat antaen `mode` parametriksi `CV_RETR_TREE`. Näin saadussa rakenteessa on helppo selata löydettyjä ulompia ääriviivoja ja niiden sisäänsä sulkemia reikiä [5]. Mikäli ääriviivakuvasta löytyy yksikin reikä, on käyttäjä liikuttanut seurattavaa aluetta pyörityksekseksi tunnistettavalla tavalla. Kuvassa 5.7b ulompi ääriviiva on kirkkaan valkoinen ja reikä harmaa.

5.8.3 Nipistys ja levitys

Nipistys- ja levityseleissä käyttäjä pitää kahta seurattavaa kohdetta hetken paikallaan ja sitten joko tuo niitä lähemmäksi toisiaan (nipistys) tai vie ne etäämmälle toisistaan (levitys). Ele tunnistetaan kun nipistystä tai levitystä on jatkunut muutaman viimeisimmän kuvan ajan ja sitten käyttäjä pysäyttää levittämisen tai vaihtaa eleen suuntaa. Väärien tunnistusten välttämiseksi eleen loppuessa seurattavien kohteiden tulee olla lähentyneet tai etääntyneet riittävän paljon. Kuten pyyhkäisyeleessä, toisen seurattavan kohteen vieminen kuva-alueen ulkopuolelle hetkeksi keskeyttää seurannan.

Alussa stabilointi toteutetaan pitkälti samoin kuin pyyhkäisyeleessäkin, mutta tarkkailtavana pisteenä käytetään seurattavien kohteiden p_1 ja p_2 välisen suoran keskipistettä m. Keskipisteen sijainti voidaan laskea helposti: $m.x = min(p_1.x, p_2.x) + \frac{|p_2.x - p_1.x|}{2}$ ja $m.y = min(p_1.y, p_2.y) + \frac{|p_2.y - p_1.y|}{2}$. Keskipiste valittiin seurattavaksi pisteeksi, sillä sen vakaana pitäminen osoittautui subjektiivisesti helpommaksi kuin kahden erillisen pisteen vakaana pitäminen.

Nipistyseleen seurantaprosessissa kullakin kierroksella otetaan talteen seurattavien kohteiden välinen etäisyys. Saatua etäisyyttä verrataan edelliseen mitattuun etäisyyteen ja mikäli etäisyys on pienentynyt tai kyseessä on ensimmäinen mittaus, se otetaan mukaan seurattavien etäisyyksien joukkoon. Mikäli etäisyys tällä kierroksella kasvoi, etäisyyttä

on mitattu vähintään N kuvan ajan ja edellinen mitattu etäisyys on vähintään n prosenttia pienempi kuin ensimmäinen etäisyys, on nipistysliike tapahtunut. Muuten aloitetaan seurantaprosessi alusta. Levitysliikkeen tunnistaminen on mekaniikaltaan päinvastainen, sillä etäisyyksien pitää kasvaa. Työssä toimiviksi parametreiksi osoittautuivat $N = 5$ ja $n = 40$.

5.8.4 Painaminen

Painamiseleessä käyttäjä kuljettaa seurattavaa kohdetta hetken aikaa kohti kameraa ja sitten pysäyttää lähestymisen tai vetää kohdetta kauemmas. Ele tunnistetaan, kun kohdetta on tuotu kohti kameraa riittävän monen kuvan ajan ja loppuetäisyys on riittävän paljon lähempänä.

Toimintaperiaatteeltaan painamiseleen tunnistaminen on lähes identtinen puristus- ja levityseleiden kanssa. Seurattavien kohteiden etäisyyden mittaamisen sijasta mitataan pienimmän suurinta etualan ääriviiva-aluetta kiertävän ympyrän läpimittaa. Selvästikin tämä läpimitta kasvaa kohteen lähestyessä kameraa, mikäli seurattavan alueen fyysinen koko pysyy samana. Kohtuullisen hyvin toimiviksi parametriarvoiksi osoittautuivat samat kuin puristus- ja levityseleissä: $N = 5$ ja $n = 40$.

5.8.5 Ravistus

Ravistuseleessä käyttäjä pyyhkii seurattavalla kohteella seurattavaa aluetta, ikään kuin heiluttaen tervehdysliikettä erittäin innostuneesti. Ele tunnistetaan kun käyttäjä on pyyhkinyt tietyn ajan sisään riittävän suuren osan kuvapinta-alasta.

Eleen tunnistuksessa käytetään hyödyksi liikehistoriakuvaa vähän samalla tavalla kuin pyörityseleessäkin. Kullakin kierroksella siis ensin luodaan ääriviivatiedoista ympyräkuva, joka päivitetään liikehistoriakuvaan. Tämän jälkeen liikehistoriakuvasta lasketaan kaikki nollasta poikkeavat pisteet nz käyttämällä OpenCV:n `cvCountNonZero` -operaatiota. Niiden osuus N kaikista pisteistä on $\frac{|nz|}{korkeus * leveys}$. Käytännössä toimivaksi N arvoksi osoittautui $N \geq 0.7$.

5.9 Sovellus esityksen ohjaamiseen

Tässä työssä toteutettiin myös yksinkertainen QML-sovellus esityksen pitämistä varten. Lähtökohdaksi otettiin valmis pohja QML Presentation System -sovelluksesta[1], ja lisäksi tehtiin laajennosliitännäinen, joka käyttää työssä muuten toteutettua kirjastoa.

[1] `http://labs.qt.nokia.com/2011/05/30/a-qml-presentation-system/`

Laajennosliitännäisen tekeminen QML -sovellusta varten on melko yksinkertaista. Kirjastoon tarvitsee toteuttaa vielä yksi luokka, joka on periytetty `QDeclarativeExtensionPlugin`ista. Sen tulee toteuttaa `QDeclarativeExtensionPlugin::registerTypes(const char*)` -funktio, jossa jokainen kirjaston luokka rekisteröidään käytettäväksi QML-koodissa tietyllä nimellä. Tämä toteutetaan käyttämällä `QtDeclarative`sta löytyvää `qmlRegisterType` aihiofunktiota (engl. template function). Toisaalta QML-liitännäinen tarvitsee tietoa myös abstrakteista luokista, joiden luominen ei kuitenkaan ole mahdollista niistä löytyvien puhtaiden virtuaalifunktioiden takia. Tällaiset luokat rekisteröidään `qmlRegisterUncreatableType` -funktion avulla. Kuvassa 5.8 on liitännäisen rekisteröintifunktion toteutus olennaisilta osin: sekä luokkien rekisteröinti että itse luokan liitännäiseksi ilmoittaminen `Q_EXPORT_PLUGIN` -makron avulla.

```
void OpenCvQmlPlugin::registerTypes(const char *uri)
{
    //@uri GestureTracker
    qmlRegisterType<OpenCvGestureTracker>    (uri, 1, 0, "OpenCvGestureTracker");
    qmlRegisterType<SwipeDetector>           (uri, 1, 0, "SwipeDetector");
    qmlRegisterType<TapDetector>             (uri, 1, 0, "TapDetector");
    qmlRegisterType<HSBackgroundSubtractor>  (uri, 1, 0, "HSBackgroundSubtractor");
    qmlRegisterType<CircleDetector>          (uri, 1, 0, "CircleDetector");
    qmlRegisterType<HaarBackgroundSubtractor>(uri, 1, 0, "HaarBackgroundSubtractor");
    qmlRegisterType<ShakeDetector>           (uri, 1, 0, "ShakeDetector");
    qmlRegisterType<PinchZoomDetector>       (uri, 1, 0, "PinchZoomDetector");

    qmlRegisterUncreatableType<GestureRecognitionBase>(uri, 1, 0,
        "GestureRecognitionBase", "GestureRecognitionBase is abstract");
    qmlRegisterUncreatableType<BackgroundSubtractor>  (uri, 1, 0,
        "BackgroundSubtractor",   "BackgroundSubtractor is abstract");
    qmlRegisterUncreatableType<GestureDetector>       (uri, 1, 0,
        "GestureDetector",        "GestureDetector is abstract");
}
Q_EXPORT_PLUGIN2(opencvqml, OpenCvQmlPlugin)
```

Kuva 5.8: Pätkä QML-laajennosliitännäistä varten tarvittavasta koodista.

Tämän jälkeen kaikkia rekisteröityjä luokkia voidaan käyttää QML -koodissa. Varsinaiseen esityksen ohjaamiseen käytettiin tunnistettavina eleinä ainoastaan pyyhkäisyä siirtymään näytettävien “kalvojen” välillä. Tämä onnistuu ottamalla kiinni `SwipeDetector` -luokalta `swipeLeft` ja `swipeRight` signaalit. Koodilistaus kuvassa 5.9 havainnollistaa tätä. Listauksessa on pyyhkäisysignaalien käsittelyn lisäksi myös tunnistimen seurantatilan käsittely, jotta käyttäjä ei yrittäisi vaihtaa kalvoa ennen kun tunnistin on valmiina seurantaan. Kannattaa huomata, että esimerkissä 5.9 tehty seurantatilaa osoittavan värin asettaminen tehdään “väärin” käyttämällä imperatiivista logiikkaa ja siistimpi vaihtoehto olisi sitoa värin arvo `SwipeDetector`in `status` -muuttujaan. Muita eletunnistusluokkia käytettiin demonstroimaan niiden läsnäoloa ja toimintaa, mutta niitä ei käytetty erityisesti ohjaamiseen.

```
SwipeDetector {
    id: swipeDetector
    enabled: true
    onSwipeLeft: {swipeArrow.text = "←"; goToNextSlide();}
    onSwipeRight: {swipeArrow.text = "→"; goToPreviousSlide();}

    onStatusChanged: {
        if (status == SwipeDetector.IDLE) {
            swipeStatusRect.color = "transparent"
        } else if (status == SwipeDetector.STABILIZING) {
            swipeStatusRect.color = "yellow"
        } else if (status == SwipeDetector.TRACKING) {
            swipeStatusRect.color = "green"
        }
    }
}
```

Kuva 5.9: Koodiesimerkki pyyhkäisyeleen käytöstä esityssovelluksessa.

6. ARVIOINTI

Työn teknistä kontribuutiota arvioitaessa oli melko vaikeaa keksiä hyviä mitattavia ja objektiivisia kriteereitä kirjaston toteutuksen onnistumiselle. Tämän vuoksi arvioinnissa pidättäydytäänkin enimmäkseen subjektiivisissa kriteereissä. Usean käyttäjän kattavan käyttäjätutkimuksen toteuttamista harkittiin, mutta tämä olisi runsaasti resursseja tuomatta kuitenkaan merkittävää lisäarvoa itse kirjaston arviointiin.

Pääasiallisina arviointikriteereinä työssä käytetään Moeslundin ja Nørgaardin [36] esittämiä toivottavia piirteitä eleohjausjärjestelmän toteuttavalle teknologialle. Kyseiset piirteet ovat pääosin tarkoitettu puettavien apuvälineiden avulla toteutettavan järjestelmän arviointiin, mutta sen painopiste on silti konenäköpohjaisen järjestelmän arvioinnissa. Tämän vuoksi listaa käytetään tässäkin työssä eräänä arvioinnin välineenä.

6.1 Arviointiperusteet

Moeslundin ja Nørgaardin [36] esittämät kuljetettavan eleohjauskäyttöliittymän toivottavat piirteet ovat seuraavat:

- **Toimiva alustus ja uudelleenalustus:** Järjestelmän pitää mukautua tilanteeseen, jossa mielenkiinnon kohde, kuten käsi poistuu ja palaa näkymään usein. Seurannan tulee kyetä alustamaan itsensä nopeasti, mutta sen tulee myös kyetä luotettavasti toteamaan, mikäli kohdetta ei ole näkymässä
- **Taustasotkun sieto:** Kuvan taka-alan kuuluvat kohteet eivät saa häiritä seurannan toimintaa. Toisin sanoen etu- ja taka-alan erottelun täytyy toimia luotettavasti
- **Valaistuksesta riippumattomuus:** Mikäli eleohjauskäyttöliittymää on tarkoitus käyttää monipuolisissa käyttöympäristöissä, sen tulee sietää sekalaisia ja vaihtelevia valaistusolosuhteita
- **Laskennallinen tehokkuus:** Koska mobiililaitteiden prosessoreilla on merkittävästi heikompi suorituskyky kuin työpöytälaitteilla, tulee suosia laskennallisesti kevyitä algoritmeja.

Näiden piirteiden toteutumisen lisäksi arvioidaan kirjaston käyttökelpoisuutta seuraavilla kriteereillä:

- **Virrankulutus:** Etenkin mobiililaitteella virrankulutus on tärkeä pitkäaikaiseen käyttöön vaikuttava tekijä. Kaikki mobiililaitteen prosessorin ja kameralaitteiston käyttö luonnollisesti kuluttaa virtaa akusta, mutta virrankulutuksen suuruudesta voi tehdä hyödyllisiä päätelmiä toteutetun kirjaston käyttökelpoisuudesta pitkän ajan kuluessa
- **Suorituskyky:** Toinen lyhytaikaisemman käyttökelpoisuuden mittari on tarkastella kuinka nopeasti järjestelmä selviytyy yhden kuvan käsittelystä.

6.2 Tuloksia

Koska taustan erottaminen toteutettiin värirajauksella, toimivat alustus ja uudelleenalustus nopeasti ja luotettavasti olettaen että näkymässä ei ole muita juuri seurattavan kohteen värisiä objekteja. Koska värirajaus tehdään jokaiselle videovirran kuvalle erikseen, selviää välittömästi, onko seurattava kohde kuvassa. Kaikki työssä toteutetut eleentunnistimet osaavat selvitä tilanteesta, jossa seurattava kohde katoaa ja ilmestyy uudestaan näkyviin.

Värirajauksella toteutettu taustan erotus ei sovellu käytettäväksi kontrolloimattomassa ympäristössä, sillä kameran kuvaan voi helposti päätyä seurattavan kohteen värisiä objekteja esimerkiksi käyttäjän vaatteissa, jotka häiritsevät seurantaa. Värirajaus on myös jossain määrin herkkä valaistuksen muutokselle. Kontrolloiduissa olosuhteissa värirajaus kuitenkin toimii erittäin luotettavasti taustan erottelijana.

Toteutetun eletunnistuskirjaston yksittäiset osat ovat melko hyvin optimoitu ja niiden ajoaikaa ei ole juuri mahdollista parantaa esimerkiksi nopeampiin kertaluokkiin. Yksittäisille pienille parannuksille on toki runsaasti sijaa. Merkittävä ongelma kuitenkin on monissa eleentunnistimissa toteutetut toistuvasti suoritettavat toimenpiteet. Esimerkiksi liikehistoriakuva lasketaan erikseen pyyhkäisy-, ympyrä- ja ravistuseleiden tunnistimissa. Sopivilla koodimuutoksilla tämä tarvitsisi tehdä vain kerran. Käytetyistä OpenCV:n funktioista on pyritty valitsemaan nopeimmat muodot tarkkuuden kustannuksella. Esimerkiksi kuvan koon muuttamisessa annetaan `cvResize` -funktiolle interpolointiparametriksi `CV_RESIZE_NN`. Tällöin interpolointi lasketaan nopeimmalla lähimmän naapurin menetelmällä.

6.2.1 Virrankulutus

Virrankulutus laskettiin käyttämällä apuna BatteryGraph -ohjelmaa[1]. Ensin laitteen akku ladattiin täyteen varaukseen (1222mAh). Tämän jälkeen mitattavaa aktiviteettia ajettiin tunnin verran, kirjattiin akussa oleva varaus ja ladattiin se taas täyteen. Kirjastoa testattiin yksinkertaisella testisovelluksella, joka käytti kaikkia kirjaston eletunnistimia. Näin saatiin mitattua suurimman rasituksen aiheuttama virrankulutus. Vertailuaktiviteeteiksi mitattiin myös virrankulutus laitteen ollessa joutotilassa ja toistettaessa 624x352 -kokoista Xvid MPEG4 -enkoodattua videota. Vain videotoistossa laitteen näyttö oli päällä. Alenema A laskettiin kaavalla $A = \frac{Varaus_{alku} - Varaus_{loppu}}{Varaus_{alku}}$. Mittaustulokset on listattu taulukossa 6.1.

Toiminto	**Varaus** (mAh)	**Alenema** (%)
Eletunnistus	809	33,1
Jouten	1175	3,9
Videotoisto	962	21,3

Taulukko 6.1: Testilaitteen virrankulutus eletunnistuksen ja vertailuaktiviteettien aikana.

Eleohjauskirjaston käyttö aiheutti tunnissa lähes kolmasosan aleneman akun varaukseen; jatkuvalla käytöllä akku siis tyhjentyisi noin kolmessa tunnissa. On selvää, että kirjastoa ei ole mielekästä hyödyntää pitkäkestoisessa päivittäisessä käytössä ilman mahdollisuutta pitää laitetta latauksessa käytön aikana. Toisaalta ero sinänsä melko tyypilliseen älypuhelimen käyttötapaukseen,videon katsomiseen, ei ole valtavan suuri.

6.2.2 Suorituskyky

Suorituskykymittauksessa tarkastellaan kuinka nopeasti kirjasto kykenee käsittelemään yhden kuvan laitteella. Eletunnistuksen tarkkuus on riippuvainen kuvavirran nopeudesta. Mitä pienempi väli kahden käsitellyn kuvan välillä on sitä vähemmän arvauksia seurattavan kohteen liikkeistä kahden otospisteen välillä joudutaan tekemään. N900:n videokamera kykenee tarjoamaan 30 kuvaa sekunnissa, jolloin maksimi yhden kuvan käsittelyaika ilman että tarkkuus laskee on noin 33ms. Todellisissa käyttötapauksissa pitää muistaa, että eleentunnistus ei ole ainoa prosessoriaikaa käyttävä tehtävä, vaan myös ohjattava sovellus vaatii oman osansa.

Suorituskykymittaukset tehtiin lisäämällä ohjelmakoodiin suoritusaikojen tallennus ja laskemalla keskimääräinen suoritusaika minuutin käytön aikana. Mittaukset tehtiin seuraaville käyttötapauksille: ei tunnistimia, yksittäiset tunnistimet käytössä, kaikki tunnistimet

[1] `http://maemo.jeroenwitteman.com/BatteryGraph/`

käytössä. Vertailun vuoksi toteutettiin myös kaksi yksinkertaista vertailusovellusta, joista toinen vain pienentää kuvan eleohjauskirjaston käyttämään kokoon ja toinen tekee pienennyksen ja värierotteluun perustuvan etualan tunnistamisen. Tuloksia on kirjattu taulukkoon 6.2.

Toiminto	**Aika** (ms)	**Kuvanopeus** (fps)
Pienennys	4	250
Pienennys ja värierottelu	20	50
Ei tunnistimia	37	27
Pyyhkäisy	47	21
Pyöritys	58	17
Nipistys	46	21
Painaminen	38	26
Ravistus	43	22
Kaikki tunnistimet	75	13

Taulukko 6.2: Testilaitteen suorituskyky eleohjaukirjaston käytössä.

Testitapaus *ei tunnistimia* osoittaa ilmeisen pullonkaulan kirjastossa, sillä kyseisessä tapauksessa tehdään pienennyksen ja värierottelun lisäksi vain reunaviivojen tunnistaminen ja varastointi. Tämä on selvästi melko raskasta vieden 17 ms (37ms - 20ms). Onneksi kyseinen operaatio tehdään vain kerran kullekin kuvalle, joten yksittäisten eleiden suoritusajoista voi vähentää operaatioon kuluvan ajan pois. Tämän vuoksi myös testitapauksen *kaikki tunnistimet* suoritusaika ei suinkaan ole yksittäisten tunnistimien ajoaikojen summa.

Laitteen suorituskyky ei selvästikään ole riittävä kirjaston käyttämiseen kaikkien eleiden kanssa. Jo pelkkä värierottelu ja reunaviivojen etsintä ja tallettaminen yhdessä enemmän kuin kameran kuvanantovälin. Etenkin kaikkien eleiden kanssa kuluva 75 ms käsittelyaika aiheuttaa käyttäjän havaittavissa olevia ongelmia tunnistustarkkuuden kanssa. Toisaalta yksittäisiä eleitä on vielä – subjektiivisesti arvioituna – suhteellisen helppo käyttää. Etenkin painamisen havaitseminen on erittäin nopeaa.

6.3 Jatkokehitysajatuksia

Työn alussa harkittiin vakavasti Haar-piirteiden käyttämistä käden tunnistamiseen. Käytännössä menetelmää ei kuitenkaan käytetty, sillä riittävän hyvän luokittimen opettaminen ja hyvän opetusmateriaalin hankkiminen olisi vienyt liikaa aikaa. Luokittimen opettaminen voi viedä useita päiviä [38], ja opetuksen joutuu luultavasti tekemään useaan otteeseen parametrien säätämistä varten [47]. Siltikään lopputulos ei ole täysin varma [21]. Tämä on vähän harmillista, sillä teoriassa laadukas luokitin mahdollistaisi erittäin tarkan

kädenseuraamisen. Luonteva jatko työlle olisikin seuraavaksi keskittyä toimivan luokittimen opettamiseen.

Luonnollisesti sekä taustanpoistoon että eleiden tunnistamiseen erikoistuneiden luokkien parametreja pitäisi säätää tarkemmaksi, sillä työtä tehdessä yleisesti siirryttiin nopeasti eteenpäin heti kun saadut parametrit edes välttävästi toimivat. Esimerkiksi väriin perustuvan etualan tunnistamisessa tuli monesti vääriä tunnistamisia koska järjestelmä otti tietyt värit liian herkästi mukaan etualaksi ja toisaalta tietyt värit eivät tunnistuneet, vaikka HSV-mallin mukaan niiden olisi pitänyt tunnistua. Toisaalta tämä johtui työssä käytettyjen kameroiden eroista ja toisaalta eroissa valaistuksissa. Kirjaston ohella siis jonkinlaiselle puoliautomaattiselle värikalibrointityökalulle voisi olla tarvetta.

Yksi selvästi kirjaston käyttöä merkittävästi parantava ajatus on koittaa käyttää sitä paremman prosessointikyvyn omaavassa laitteistossa. Esimerkiksi useissa mobiililaitteissa on jo nyt moniydinprosessoreita. Arkkitehtuurin voisi kuvitella olevan melko helposti muokattavissa paremmin säikeistetyksi, jolloin useammasta ytimestä olisi enemmän hyötyä. OpenCV:n suorituskykyä on saatu parannettua käyttämällä laskentaan yleiskäyttöistä grafiikkaprosessoria (GPGPU) [1; 31]. Työtä tehtäessä ei kuitenkaan ollut käytettävissä mobiililaitetta, jolla voisi hyödyntää GPGPU:n ominaisuuksia, joten tämän lähestymistavan kokeilu ei ollut mahdollista.

7. YHTEENVETO

Konenäköpohjaisuus mobiililaitteen ohjaamisessa mahdollistaa mielenkiintoisia sovelluksia, joissa laitetta ei ole tarvetta pitää kädessä koko aikaa. Tällöin esimerkiksi puhelimeen voi vastata heilauttamalla kättään laitteen kameralle ja keskustelu alkaa kaiuttimien kautta. Käytännössä yleisiä kirjastoja tällaisen toiminnallisuuden toteuttamiseksi ei juuri ole tarjolla. Työssä toteutettiin yleiskäyttöinen konenäöllä toteutettu eleohjauskirjasto, jota voisi käyttää apuvälineenä tällaisten eleohjattujen sovellusten luomisessa. Esimerkiksi kehitysprosessin aikana kirjastoa käytettiin PowerPoint -tyylisen esityksen ohjaamiseen.

Toteutettu kirjasto osaa kohtuullisen luotettavasti tunnistaa yhdeksän erilaista elettä: pyyhkäisy (4 suuntaa), pyöritys, nipistys (2 suuntaa), painaminen ja ravistus. Näiden avulla on mahdollista luoda jo melko monipuolisia sovelluksia. Toistaiseksi kirjasto tunnistaa seurattavan kohteen värin perusteella. Käytännössä käyttäjällä pitää siis olla koko ajan esimerkiksi kirkkaan sininen tai vihreä esine (työssä käytettiin taulutussin korkkia) kädessään.

Käytännössä eletunnistus ei ollut mobiililaitteella kovin käytännöllistä, sillä käytössä olleen Nokia N900 -älypuhelimen prosessointiteho ei ollut riittävä luotettavaan eletunnistukseen. Toinen ilmeinen ongelma on eletunnistuksen aiheuttama virrankulutuksen kasvu. Selvästikään mikään työkalu mobiililaitteella ei ole käyttökelpoinen jos se rajoittaa käyttöajan muutamaan tuntiin. Mikäli laitteen voi käytön ajaksi liittää laturiin ja käytössä on selvästi N900:aa enemmän prosessointitehoa, voisi kirjaston hyödyntäminen olla paljon mielekkäämpää.

Vaikka toteutetun kirjaston tutkimuksen kannalta olennainen käyttöympäristö on mobiililaitteella, voi kirjastoa käyttää myös PC-tietokoneella. Tämä onnistuu helposti, sillä kaikki kirjaston toteuttamisessa käytetyt komponentit ovat niin sanotusti poikkialustaisia; sama ohjelmakoodi kääntyy suoraan sekä pöytäkoneelle että mobiililaitteelle. Tästä on ilmeistä hyötyä ohjelmistokehityksessä kun tuotettua koodia voi “koeajaa” heti kehityskoneella ilman ylimääräistä asennusaskelta mobiililaitteelle. Koska pöytäkoneella on selvästi enemmän prosessointitehoa käytettävissään, eikä virrankulutus ole merkittävä rajoite, on toteutettu kirjasto tällä hetkellä paremmin käytettävissä työpöytäympäristössä.

LÄHTEET

[1] Allusse, Y., Horain, P., Agarwal, A. & Saipriyadarshan, C. 2008. GpuCV: an Open-source GPU-accelerated Framework for Image Processing and Computer Cision. In: Proceeding of the 16th ACM International Conference on Multimedia. pp. 1089–1092.

[2] Barczak, A.L.C. & Dadgostar, F. 2005. Real-time hand tracking using a set of cooperative classifiers based on haar-like features. In: Research Letters in the Information and Mathematical Sciences. pp. 29–42.

[3] Bernardes, J., Nakamura, R. & Tori, R. 2009. Design and Implementation of a Flexible Hand Gesture Command Interface for Games Based on Computer Vision. In: VIII Brazilian Symposium on Games and Digital Entertainment (SBGAMES). pp. 64–73.

[4] Bolt, R.A. 1980. Put-that-there: Voice and gesture at the graphics interface. In: Proceedings of the 7th annual conference on Computer graphics and interactive techniques. pp. 262–270.

[5] Bradski, G. & Kaehler, A. 2008. Learning OpenCV. O'Reilly Media, Sebastopol, CA, USA. ISBN 978-0-596-51613-0, 556 pp.

[6] Bradski, G.R. & Davis, J.W. 2002. Motion segmentation and pose recognition with motion history gradients. Machine Vision and Applications, 13, 174–184.

[7] Burger, W. & Burge, M.J. 2009. Principles of Digital Image Processing. Undergraduate Topics in Computer Science. Springer-Verlagen, 1 ed., 259 pp.

[8] Chen, Q., Georganas, N. & Petriu, E. 2007. Real-time Vision-based Hand Gesture Recognition Using Haar-like Features. In: Instrumentation and Measurement Technology Conference Proceedings, 2007.

[9] Choi, E.S., Bang, W.C., Cho, S.J., Yang, J., Kim, D.Y. & Kim, S.R. 2005. Beatbox music phone: gesture-based interactive mobile phone using a tri-axis accelerometer. In: IEEE International Conference on Industrial Technology. pp. 97–102.

[10] van Dam, A. 1997. Post-WIMP user interfaces. Communications of the ACM, 40, 2.

[11] Davis, J.W. 2001. Representing and recognizing human motion: From motion templates to movement categories. In: Paper presented at the International Conference on Intelligent Robots and Systems, Maui.

[12] Deyle, T. 2010. Low-Cost Depth Cameras (aka Ranging Cameras or RGB-D Cameras) to Emerge in 2010? Online. http://www.hizook.com/blog/2010/03/28/low-cost-depth-cameras-aka-ranging-cameras-or-rgb-d-cameras-emerge-2010. Viitattu 14.10.2011.

[13] eyeSight 2011. Advanced Gesture Control Capabilities for Devices. Online. http://www.eyesight-tech.com/solutions/eyecan/. Viitattu 14.10.2011.

[14] Fitzpatrick, M. 2009. Using the Wii for Vestibular rehabilitation. Vestibular Disorders Association. Publication C-7.

[15] Freeman, W.T. & Weissman, C.D. 1995. Television Control by Hand Gestures. In: IEEE International Workshop on Automatic Face and Gesture Recognition.

[16] Geer, D. 2004. Will Gesture Recognition Technology Point the Way? Computer, 37, 10, 20–23.

[17] Goodwins, R. 208. Windows 7? No arm in it. ZDNet UK. Online. http://www.zdnet.co.uk/blogs/mixed-signals-10000051/windows-7-no-arm-in-it-10008314/. Viitattu 25.9.2011.

[18] Greenwald, W. 2010. Kinect vs. PlayStation Move vs. Wii: Motion-Control Showdown. PcMag. Online. http://www.pcmag.com/article2/0,2817,2372244,00.asp. Viitattu 25.9.2011.

[19] Harris, C. & Stephens, M. 1988. A combined corner and edge detector. In: Proceedings of the 4th Alvey Vision Conference. pp. 147–151.

[20] Horn, B.K. & Schunck, B.G. 1981. Determining optical flow. Artificial Intelligence, 17, 1–3.

[21] Ince, I.F., Socarras-Garzon, M. & Yang, T.C. 2010. Hand Mouse: Real Time Hand Motion Detection System Based on Analysis of Finger Blobs. International Journal of Digital Content Technology and its Applications, 4, 2.

[22] Inkeroinen, T. 2008. Eleohjaus matkapuhelimessa. Master's thesis, Oulun yliopisto, sähkö- ja tietotekniikan osasto.

[23] Ionescu, D., Ionescu, B., Gadea, C. & Islam, S. 2011. An intelligent gesture interface for controlling TV sets and set-top boxes. In: 6th IEEE International Symposium on Applied Computational Intelligence and Informatics (SACI). pp. 159–164.

[24] Johnson, E.A. 1965. Touch display – a novel input/output device for computers. Electronics Letters, 1, 8, 219–220.

[25] Kelley, J.F. 1984. An iterative design methodology for user-friendly natural language office information applications. ACM Transactions on Information Systems, 2, 1, 26–41.

[26] Kim, K., Chalidabhongse, T.H., Harwood, D. & Davis, L. 2005. Real-time foreground-background segmentation using codebook model. Real-Time Imaging, 11, 172–185.

[27] Kulkarni, S., Manoj, H., David, S., Madumbu, V. & Kumar, Y. 2011. Robust hand gesture recognition system using motion templates. In: 2011 11th International Conference on ITS Telecommunications. pp. 431–435.

[28] Lee, J.C. 2008. Hacking the Nintendo Wii Remote. Pervasive Computing, IEEE, 7, 3, 39–45.

[29] Liu, N. & Lovell, B.C. 2003. Gesture Classification Using Hidden Markov Models and Viterbi Path Counting. In: Proceedings of the Seventh Biennial Australian Pattern Recognition Society Conference. pp. 273–282.

[30] Liu, X. & Fujimura, K. 2004. Hand gesture recognition using depth data. In: Proceedings of the Sixth IEEE International Conference on Automatic Face and Gesture Recognition. pp. 529–534.

[31] Lopez, M.B., Nykanen, H., Hannuksela, J., Silven, O. & Vehvilainen, M. 2011. Accelerating image recognition on mobile devices using GPGPU. In: Owens, JD and Lin, IJ and Zhang, YJ and Beretta, GB (ed.), Parallel Processing for Imaging Applications. Proceedings of SPIE, vol. 7872.

[32] Manchanda, K. & Bing, B. 2010. Advanced mouse pointer control using trajectory-based gesture recognition. In: Proceedings of the IEEE SoutheastCon 2010 (SoutheastCon). pp. 412–415.

[33] McDonald, K. 2011. FaceOSC. Vimeo. Online. `http://vimeo.com/26098366`. Viitattu 21.9.2011.

[34] Mistry, P. & Maes, P. 2009. SixthSense: a wearable gestural interface. In: ACM SIGGRAPH ASIA 2009 Sketches.

[35] Mizell, D. 2003. Using Gravity to Estimate Accelerometer Orientation. In: Proceedings of the Seventh IEEE International Symposium on Wearable Computers.

[36] Moeslund, T.B. & Nørgaard, L. 2003. A Brief Overview of Hand Gestures Used in Wearable Human Computer Interfaces. Tech. rep., Laboratory of Computer Vision and Media Technology Aalborg University.

[37] Myers, B.A. 1996. A Brief History of Human Computer Interaction Technology. ACM INTERACTIONS, 5, 44–54.

[38] Naotoshi, S. 2008. Tutorial: OpenCV haartraining (Rapid Object Detection With A Cascade of Boosted Classifiers Based on Haar-like Features). Online. http://note.sonots.com/SciSoftware/haartraining.html. Viitattu 31.8.2011.

[39] Nielsen, M., Störring, M., Moeslund, T.B. & Granum, E. 2003. A Procedure for Developing Intuitive and Ergonomic Gesture Interfaces for HCI. In: Gesture Workshop'03. pp. 409–420.

[40] Nokia 2010. Introduction to QtQuick for C++ Developers. Online. http://qt.nokia.com/files/pdf/qt-quick-for-c-developers. Viitattu 15.2.2012.

[41] Nokia 2012. Modular Class Library. Online. http://qt.nokia.com/products/library. Viitattu 15.2.2012.

[42] Nokia 2012. Supported Platforms. Online. http://developer.qt.nokia.com/doc/qt-4.8/supported-platforms.html. Viitattu 15.2.2012.

[43] Olwal, A. & Feiner, S. 2005. Interaction Techniques Using Prosodic Features of Speech and Audio Localization. In: Proceedings of International Conference on Intelligent User Interfaces.

[44] OpenCV 2012. OpenCV Wiki. Online. http://opencv.willowgarage.com/wiki/. Viitattu 15.2.2012.

[45] Oviatt, S. 2002. Breaking the Robustness Barrier: Recent Progress on the Design of Robust Multimodal Systems. Elsevier, Advances in Computers, vol. 56, pp. 305–341.

[46] Papageorgiou, C., Oren, M. & Poggio, T. 1998. A general framework for object detection. In: Sixth International Conference on Computer Vision, 1998. pp. 555–562.

[47] Pisarevsky, V. 2010. OpenCV Object Detection: Theory and Practice. Presentation. Online. http://fsa.ia.ac.cn/files/OpenCV_FaceDetection_June10.pdf. Viitattu 14.10.2011.

[48] Rabiner, L.R. 1989. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. In: Proceedings of the IEEE. vol. 77, pp. 257–286.

[49] Reifinger, S., Wallhoff, F., Ablassmeier, M., Poitschke, T. & Rigoll, G. 2007. Static and Dynamic Hand-Gesture Recognition for Augmented Reality Applications.

In: Human-Computer Interaction. HCI Intelligent Multimodal Interaction Environments, Lecture Notes in Computer Science, vol. 4552, pp. 728–737.

[50] Rimpiläinen, T. 2011. Eleohjauksen lisääminen catwalkiin. Opinnäytetyö, Oulun seudun ammattikorkeakoulu, Tietotekniikan koulutusohjelma.

[51] Rosenfeld, A. 1988. Computer vision: basic principles. Proceedings of the IEEE, 76, 8, 863–868.

[52] Rus, J. 2010. Hsl-hsv models. Online. http://commons.wikimedia.org/wiki/File:Hsl-hsv_models.svg. Viitattu 25.1.2012.

[53] Schlömer, T., Poppinga, B., Henze, N. & Boll, S. 2008. Gesture recognition with a Wii controller. In: Proceedings of the 2nd international conference on Tangible and embedded interaction. ACM, New York, NY, USA, TEI '08, pp. 11–14.

[54] Shi, J. & Tomasi, C. 1994. Good Features to Track. In: Proceedings in 1994 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. pp. 593–600.

[55] Simanek, D. 2008. Mirror and Prism Methods for 3d Macro Photography. Online. http://www.lhup.edu/~dsimanek/3d/stereo/3dgallery16.htm. Viitattu 25.1.2012.

[56] Ten, S. 2010. How Kinect depth sensor works – stereo triangulation? Mirror2Image. Online. http://mirror2image.wordpress.com/2010/11/30/how-kinect-works-stereo-triangulation/. Viitattu 28.3.2012.

[57] Thelin, J. 2011. Quick user interfaces with Qt. Linux Journal, 2011.

[58] Toyama, K., Krumm, J., Brumitt, B. & Meyers, B. 1999. Wallflower: principles and practice of background maintenance. In: The Proceedings of the Seventh IEEE International Conference on Computer Vision. vol. 1, pp. 255–261.

[59] Vincent, E. & Laganiere, R. 2001. Matching Feature Points in Stereo Pairs: A Comparative Study of Some Matching Strategies. Online: <http://www.site.uottawa.ca/~laganier/publications/matchingMGV01.pdf>

[60] Viola, P. & Jones, M.J. 2004. Robust Real-Time Face Detection. International Journal of Computer Vision, 57, 137–154.

[61] Wikipedia 2012. Viterbi Algorithm. Online. http://en.wikipedia.org/wiki/Viterbi_algorithm. Viitattu 9.2.2012.

[62] Woodfill, J., Gordon, G. & Buck, R. 2004. Tyzx DeepSea High Speed Stereo Vision System. In: CVPRW '04. Conference on Computer Vision and Pattern Recognition, Workshop. p. 41.

[63] Wroblewski, L. 2010. Touch Gesture Reference Guide. Online. http://www.lukew.com/touch/. Viitattu 10.10.2011.

[64] Yang, J. & Xu, Y. 1994. Hidden Markov Model for Gesture Recognition. Tech. rep., Carnegie Mellon University. Robotic Institute.