UNIVERSITY OF OULU

TIETO- JA SÄHKÖTEKNIIKAN TIEDEKUNTA

Seppo Pakonen
Juha Ohtamaa
Tuomas Savunen

Viiveen optimointi lisätyn todellisuuden sovelluksessa

Kandidaatintyö
Tietotekniikan tutkinto-ohjelma
4/2017

Pakonen S., Ohtamaa J., Savunen T. (2017) Viiveen optimointi lisätyn todellisuuden sovelluksessa. Oulun yliopisto, tietotekniikan tutkinto-ohjelma. Kandidaatintyö, 39 s.

TIIVISTELMÄ

Lisätyn todellisuuden sovellukset suorittavat konenäön ja kuvankäsittelyn tehtäviä, joita voidaan toteuttaa eri menetelmillä ja vaihtelevalla tehokkuudella. Tämä dokumentti esittää tehokkuusvertailun lisätyn todellisuuden ilmakiekkopelin kautta. Vertailu keskittyy pääkohtien tunnistimiin ja piirteiden kuvaajiin niiden ollessa merkittävin muokattava tekijä pelissä.

Pääkohtien tunnistimista testattiin ORB, MSER, FAST, BRISK, SIFT ja SURF algoritmeja ja piirteiden kuvaajista testattiin ORB, BRISK, BRIEF, FREAK, SIFT ja SURF algoritmeja. Testi suoritettiin ilmakiekkopelissä nauhoitetulle videolle, jossa ympäristö vastasi lisätyn todellisuuden pelin normaalia peliympäristöä.

ORB algoritmia pidetään parhaana vaihtoehtona tehokkuutta vaativiin sovelluksiin ja SIFT algoritmia pidetään parhaana tarkkuutta vaativiin sovelluksiin. Myös testitulokset osoittavat ORB algoritmin olevan paras vaihtoehto tämän työn ohessa tehdyssä ilmakiekkopelissä.

Avainsanat: lisätty todellisuus, ilmakiekko, viive, orb, mser, fast, brisk, sift, surf, freak

Pakonen S., Ohtamaa J., Savunen T. (2017) Optimization of delay in an augmented reality application. University of Oulu, Degree Programme in Computer Science and Engineering. Bachelor's Thesis, 39 p.

ABSTRACT

Augmented reality applications require computer vision and image processing functions that can be implemented using different methods and with varying degrees of efficiency. This document presents a comparison of the efficiency with the air hockey game. The comparison focuses on keypoint detectors and feature descriptors as they are the most important factor in the game that can be changed.

We tested ORB, MSER, FAST, BRISK, SIFT and SURF algorithms from all keypoint detectors and ORB, BRISK, BRIEF, FREAK, SIFT and SURF algorithms from all feature descriptors. The test was carried out with a recorded video from the air hockey game, that corresponded to the normal game environment of an augmented reality game.

The ORB algorithm is considered to be the best option for demanding applications and the SIFT algorithm is considered to be the best option in precision applications. Also, the test results show that the ORB algorithm is the best option in the air-hockey game that was programmed alongside this work.

Keywords: augmented reality, air hockey, delay, orb, mser, fast, brisk, sift, surf, freak

SISÄLLYSLUETTELO

TIIVISTELMÄ
ABSTRACT
SISÄLLYSLUETTELO
ALKULAUSE
LYHENTEIDEN JA MERKKIEN SELITYKSET
1. JOHDANTO....8
1.1. Taustoitus....10
1.2. Tavoitteet ja aihealue....13
2. LYHYT HISTORIA....14
2.1. 80-luku....14
2.2. 90-luku....15
2.3. 2000-luku....16
2.4. 2010-luku....17
2.5. Tulevaisuus....18
3. KESKEISET REAALIAIKAISET MENETELMÄT....19
3.1. SIFT....19
3.2. SURF....20
3.3. FAST....21
3.4. BRIEF....22
3.5. BRISK....23
3.6. FREAK....24
3.7. MSER....25
3.8. ORB....26
4. ILMAKIEKKOPELIN TOTEUTUS....27
4.1. Kehitysympäristö....27
4.2. Suunnittelu....28
4.3. Kuvankäsittelyn komponentit....29
4.4. Mailan tunnistus....29
4.5. Pelin kulku....30
4.6. Ohjelmointi....31
4.7. Lopputulos....31
5. PELIN HYÖDYNTÄMINEN VIIVEEN MITTAUKSESSA....32
6. TULOKSET....34
6.1. Testiaineisto....34
6.2. Pääkohtien tunnistimien vertaaminen....35
6.3. Kuvaajien vertaaminen....37
7. YHTEENVETO....39
8. LÄHTEET....40

ALKULAUSE

Kiitämme kaikkia, jotka ovat olleet tukemassa kiinnostusta tietotekniikkaan lapsuudesta tähän päivään saakka.

Oulu, Maaliskuussa 2017
Seppo Pakonen, Juha Ohtamaa, Tuomas Savunen

LYHENTEIDEN JA MERKKIEN SELITYKSET

SIFT	Lyhenne sanoista Scale-invariant feature transform. SIFT on patentoitu algoritmi kuvien pääkohtien tunnistamiseen ja piirteiden kuvaamiseen.
SURF	Lyhenne sanoista Speeded up robust features. Kuten SIFT, se on patentoitu algoritmi kuvien pääkohtien tunnistamiseen ja piirteiden kuvaamiseen.
FAST	Lyhenne sanoista Features from Accelerated Segment Test. FAST on nopea menetelmä kulmien tunnistamiseen.
BRIEF	Lyhenne sanoista Binary Robust Independent Elementary Features. BRIEF on nopea menetelmä binäärimuotoisten kuvaajien määrittämiseen.
ORB	Lyhenne sanoista Oriented FAST and rotated BRIEF on nopea patentoimaton algoritmi kuvien pääkohtien tunnistamiseen ja piirteiden kuvaamiseen.
MSER	Lyhenne sanoista Maximally stable extremal regions. MSER on menetelmä, jolla tunnistetaan yhtenäisiä alueita.
BRISK	Lyhenne sanoista Binary Robust Invariant Scalable Keypoints. BRISK on menetelmä pääkohtien tunnistamiseen.
FREAK	Lyhenne sanoista Fast Retina Keypoint. FREAK on menetelmä pääkohtien tunnistamiseen.
Pääkohta, engl. key point	Pääkohta on mielenkiintoinen piste kuvassa. Erikoiseksi pääkohdat tekee se, että ne voidaan löytää helpommin eri kuvista huolimatta kuvan muutoksesta.
Kuvaaja, engl. descriptor	Dataa, joka ilmaisee miten jokin toinen data on tallennettu.
Piirre, engl. feature	Piirre on kuvankäsittelyssä kuvaan liittyvää dataa, joka auttaa suorittamaan jonkin tehtävän tai ratkaisemaan ongelman.
PCA	PCA on lyhenne sanoista Principal component analysis ja tarkoittaa pääkomponenttianalyysiä, joka on

	tilastollinen menetelmä keskeisimpien komponenttien tuomiseen esiin datasta.
DFOB	DFOB on epäsuora lyhenne lauseesta “detecting and describing features by octagon filter bank”. Se on menetelmä pääkohtien tunnistamiseen ja kuvaajien laskemiseen.

1. JOHDANTO

Kaupallisen pelin ensireaktio on tärkeä tekijä sen menestykselle. Keskittyminen pelin sisältöön kuitenkin häiriintyy, jos peli on jo teknisesti puutteellinen, kuvassa on virheitä tai se päivittyy huomattavan hitaasti. Kameran tai pään liikkeeseen nopeasti reagoiva lisätyn todellisuuden sovellus on miellyttävämpi kuin hieman myöhässä reagoiva sovellus, mikä on motivaatio tälle kandidaatintyölle.

Lisätyn todellisuuden pelit ovat vielä odottamassa läpimurtoaan suurelle yleisölle, kun taas perinteiset 3D-pelit ovat olleet jatkuvassa kehityksessä jo 90-luvulta asti. Lisätty todellisuus tuo mukanaan uusia ongelmia, kuten ympäristön 3D-mallinnuksen ja kuvakulman tarkan tunnistamisen tilassa.

Viimeisimmän vuosikymmenen aikana älypuhelimet ovat mahdollistaneet uudenlaisia lisätyn todellisuuden sovelluksia. Saatavilla on pelejä ja ohjelmia, jotka tunnistavat ympäristön kameran, kompassin ja GPS-sijainnin avulla.

Vuonna 2017 lisätyn todellisuuden sovellukset ovat tavallisesti kaupallisia hyötyohjelmia ja pelejä, joita suoritetaan kannettavilla laitteilla ja jotka mahdollistavat yhteistyötä ja informaation yhdistämistä kuvaan. Myös ammatinharjoittajien tarkkuutta parantavat sovellukset ovat hyvä kaupallinen kohde.

Erityisessä suosiossa on ollut vuonna 2016 julkaistu Pokemon Go peli, jossa kameran kuvan päälle piirretään Pokemon-hahmoja ja jossa GPS-sijainti on merkittävässä roolissa pelin kulkuun. Merkittävin varsinaisista stereokuvan mahdollistavista katselulaitteista on vuonna 2016 julkaistu Microsoft HoloLens [Kuva 1.], johon on mahdollista ostaa ja ladata valmiita lisätyn todellisuuden sovelluksia.

Kuva 1. Microsoftin näkemys lisätyn todellisuuden sovelluksen hyödyllisyydestä autojen kokoonpanolinjastolla.

Uudet lisätyn todellisuuden katselulaitteet käyttävät stereokuvaan perustuvaa esitystä. Objektien etäisyyden määrittäminen stereo-kuvista on ollut merkittävä syy kulmapisteiden etsimiselle ja niiden moderneille vastineille, pääkohdille ja kuvaajille. Kiinnostus on lisääntynyt tietotekniikan yleistymisen myötä tehokkaisiin reaaliaikaisiin menetelmiin matemaattisesti tarkkojen mallien lisäksi.

Tämän dokumentin ohessa tehtiin lisätyn todellisuuden sovellus käyttäen yleisiä työkaluja ja dokumentissa tarjotaan arvio viimeisimmistä menetelmistä, kun tehokkuus on tärkein arvosteluperuste.

1.1 Taustoitus

Lisätyn todellisuuden sovellus näyttää kuvan todellisesta ympäristöstä ja lisää siihen kuvasta ja muista sensoreista saadun informaation pohjalta piirrettyä grafiikkaa. Sovellus parantaa käyttäjän käsitystä ympäristöstä tai lisää siihen viihteellisiä elementtejä. Virtuaalitodellisuus sen sijaan näyttää kokonaan simuloidun maailman. Käytännössä lisätyn todellisuuden sovellus reagoi nopeasti ja reaaliaikaisesti katselulaitteen suunnan ja sijainnin muutoksiin, sekä havaittuun kuvaan ympäristöstä.

Sovellusten avulla ympäristöstä tulee interaktiivisempi, kuin mitä se todellisuudessa on. Interaktiivisuus voi perustua tarkkoihin todellisiin mittausarvoihin ympäristöstä tai se voi lisätä kokonaan uusia elementtejä.

Laitteistolta sovellukset vaativat tietokoneen, näytön, sensorit ja ohjaimet. Älypuhelimet ja tablet-tietokoneet sisältävät komponentit, jotka mahdollistavat jotkin lisätyn todellisuuden sovellukset. Kameran lisäksi ne sisältävät usein kiihtyvyysanturin, GPS:n ja kompassin, jotka yhdessä luovat tarkimman kuvakulman tunnistuksen.

Parhaita katselulaitteita ovat puettavat tietokoneet, mutta perinteisesti on käytetty myös älypuhelimia, tavallisia tietokoneen näyttöjä ja projektoreita. Päähän kiinnitettävät näytöt ovat näyttölaitteita, jotka kiinnitetään otsan tasolle silmien eteen. Ne näyttävät todellisen maailman näytön läpi ja virtuaalisen maailman sen edessä. Niissä on sensorit, jotka havaitsevat suunnan ja paikan muuttumisen ympäristöön nähden. Lisäksi joissakin malleissa on käden liikkeet tunnistavia 3D-kameroita.

Arkisempaan käyttöön on suunniteltu silmälasien kokoisia katselulaitteita. Ne sisältävät myös itse tietokoneen. Jotta lisätyn todellisuuden sovellusta voi käyttää koko päivän, tarvitsee tällaisen laitteen. Piilolinsseihin mahtuvat näytöt ovat mahdollisesti miellyttävin vaihtoehto käyttökokemukseltaan, mutta niistä ei ole kaupallista tarjontaa. Niiden toteutus oikeassa mittakaavassa on edelleen suuri käytännön haaste. Eräs erikoisempi puettava näyttölaite on EyeTap [1], joka perustuu katselulinssiin ja sen puoliläpäisevään peiliin, joka peilaa ympäristön kuvan kameraan ja monitorin kuvan silmään.

Avaruudellisessa lisätyn todellisuuden sovelluksessa lisätään grafiikkaa todellisen maailman objektien tai maiseman päälle käyttämällä projektoria. Nämä sovellukset eivät yleensä ole interaktiivisia yleisölle tai se on rajoitettua. Esimerkkejä ovat pienoismallien, pöytien, huoneen seinien tai rakennusten ulkoseinien päälle projisoidut animaatiot, jotka lisäävät simuloidun tekstuurin tai muun efektin. [Kuva 2.]

Kuva 2. Avaruudellinen lisätyn todellisuuden sovellus, jossa heijastetaan simuloitu tekstuuri Empire State pilvenpiirtäjän seinustaan.

Lisätyn todellisuuden sovelluksen oleellinen ominaisuus on mukautua laitteen avulla havaittuun ympäristöön mahdollisimman hyvin. Ohjelmointikirjastot, joiden varaan sovellus rakennetaan, käsittelevät tavallisesti sensoreista saadun datan ja tarjoavat sen abstrahoituna ohjelmoijalle. Ne suorittavat ohjelman taustalla sensoreiden datan käsittelyä, kuten esimerkiksi videon liikkeentunnistusta, nopeaa suunnan muutoksen mittausta kiihtyvyysantureilla ja hidasta sijainnin määritystä GPS-koordinaattien perusteella. Eri antureista saadut liikkeen- ja suunnanmuutokset yhdistetään algoritmilla yhdeksi muutokseksi.

Liikkeentunnistus videosta on tarkka keino mitata suunta ympäristössä. Kiihtyvyysanturit antavat vain suhteellisen muutoksen ja kompassi on altis häiriöille. Katselukulman muutoksen mittaaminen videosta tapahtuu etsimällä kahdesta eri ajanhetkillä otetuista kuvista samat pisteet ja laskemalla niiden perusteella projektiivinen muutos.

On useita erilaisia ohjelmointikirjastoja lisätyn todellisuuden sovellusten ohjelmointiin, mutta ne usein vaativat tiettyjen katselulaitteiden omistamista, eivätkä ne mahdollista eri algoritmien suorituskyvyn vertailua. Yksinkertaisen sovelluksen voi tehdä yleisillä standardeja tukevilla kirjastoilla, mutta monimutkainen sovellus vaatii jo paljon enemmän suunnittelua, jolloin kannattaa harkita tarkoitukseen tehtyä ohjelmointiympäristöä.

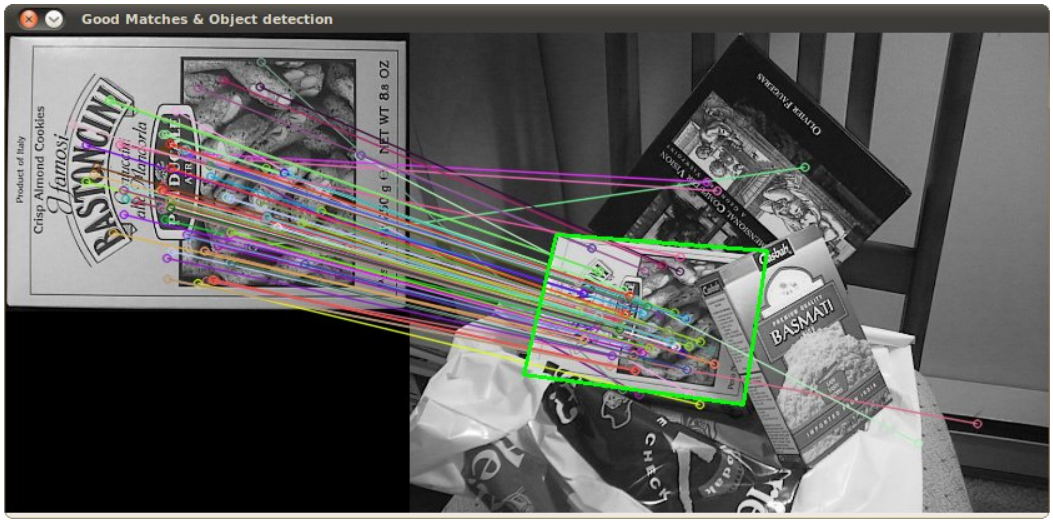
Tämän kandidaatintyön ohessa tehtävän sovelluksen aiheen valintaan vaikuttavat monet seikat. Työn määrän täytyy pysyä kohtuullisissa rajoissa ja toteuttamisen vaatimien uusien taitojen opiskeluun on rajoitetusti aikaa. Myös mielikuvitus ainutlaatuisiin ratkaisuihin on rajallinen ja mallia joudutaan ottamaan kauppapaikkojen tarjonnasta.

Internetissä on ostettavissa ja ladattavissa paljon erilaisia lisätyn todellisuuden sovelluksia. Mediassa on vertailtu lisätyn todellisuuden ohjelmia parhausjärjestyksissä ja niissä on esillä nykyhetken kehityksen kärki. Hyödyllisiä ohjelmia ovat kaikenlaiset kameran kuvan päälle informaatiota lisäävät sovellukset,

jotka hyödyntävät esimerkiksi karttaa, Google-hakua, wikipediaa, säätietoja, kompassia tai GPS-sijaintia. Kandityön aihe, lisätyn todellisuuden ilmakiekkopeli, on helposti toteutettava ja sen alkuperäinen versio kuvassa 3. on tunnettu ja suosittu pelihalleissa.

Kuva 3. Perinteinen ilmakiekkopeli.

OpenCV:n esimerkit esittelevät alkuperäisen objektin löytämisen ympäristön kuvasta ja sen reunojen piirtämisen. [Kuva 4.] Ilmakiekon pelikentän pystyy helposti piirtämään kyseisten ääriviivojen sisään. Pelin toteutuksessa tarvitsee vain 2D-kuvan pelikentästä, jonka näyttämiseen 3D-ympäristössä riittää kuvan perspektiivin muuntaminen ja siirto oikeaan paikkaan.



Kuva 4. OpenCV esimerkin "Features2D + Homography to find a known object" kuvakaappaus. [2]

1.2 Tavoitteet ja aihealue

Lisätyn todellisuuden sovelluksen viiveen minimointi jakaantuu useaan eri osa-alueeseen: nopeaan kuvan renderöintiin, lyhyeen syöttölaitteiden ja sensorien vasteaikaan, nopeaan pelitilanteen päivitykseen ja nopeaan todellisuuden ja virtuaalitodellisuuden täsmäämiseen. Kuvan renderöimisen nopeuteen vaikuttavat piirrettävän grafiikan tarkkuus eli resoluutio, objektien määrä ja jälkikäsittely, mikä on yleistä kaikissa 3D-sovelluksissa. Syöttölaitteiden ja sensorien vasteaika on pieni jopa tavallisissa älypuhelimissa ja niihin pystyvät vaikuttamaan ensisijaisesti vain laitteiden valmistajat.

Erityinen ongelma lisätyn todellisuuden sovelluksissa on vasteajan minimointi todellisuuden muutosten käsittelyssä ja siirtämisessä virtuaalimaailmaan. Suunnan määritys kiihtyvyysanturilla ja kompassilla on nopeaa, mutta ei tarkkaa. Pikselin tarkkuudella määritettävään sijaintiin tarvitaan suunnan tunnistus videokuvasta. Suunnan muutoksen mittaus tapahtuu seuraamalla kiinnostavia pisteitä videossa ja määrittämällä sen avulla projektiivinen muutos.

Objektin löytämistä ympäristön kuvasta on pidetty historiallisesti tarkkuuden ongelmana. Siihen tarvitaan algoritmeja, jotka löytävät samat pisteet eri kuvista riippumatta eri kuvakulmasta tai etäisyydestä. Eri algoritmeja on verrattu ja mittauksen suorittamiseen on esitetty yhtenäinen protokolla. [3]

Tavoitteenamme oli toteuttaa lisätyn todellisuuden ilmakiekkopeli ja tehdä sen avulla algoritmien vertailu. Pelissä tuli olla myös minimaalinen grafiikan piirtämisen viive ja kameran vasteaika. Aihealueeseen kuului ohjelman suunnittelu, ohjelmointityö, käytettävien komponenttien vertailu ja viiveen minimointi.

2. LYHYT HISTORIA

Tärkein tekijä lisätyn todellisuuden sovelluksen laadukkaalle esitykselle on komponentti, joka yhdistää virtuaalimaailman havaittuun maailmaan reaaliaikaisesti. Laadukkaassa sovelluksessa käytetään kiihtyvyysanturin, kameran seuraajan, kompassin ja GPS:n yhdistelmää avaruudellisen sijainnin ja katselukulman määrittämiseksi, joista nopeuden pullonkaulana on kameran kuvan seuraaja. Ilmakiekkopelissä keskitytään kameran seuraajan optimointiin.

Kameran kuvan seuraamisessa käytetään algoritmeja, jotka tunnistavat kuvasta pääkohdat ja laskevat niiden piirteille kuvaajat. Näitä ovat esimerkiksi ORB, MSER, FAST, BRISK, SIFT, BRIEF ja FREAK. Vanhemmat näistä ovat keskittyneet tarkkuuteen, kun taas uudemmat ovat keskittyneet nopeuteen käyttämällä mielenkiintoisia oikopolkuja prosessoinnissa. Tarkkuuteen perustuvat algoritmit ovat väistämättä olleet nopeampien algoritmien esikuvia ja tässä luvussa tuodaan esille aikajana, jossa kehitys on tapahtunut.

Lisätyn todellisuuden ilmakiekossa yksi keskeisimmistä komponenteista on pelikentän tunnistus ympäristön kuvasta, eikä se olisi mahdollista ilman monia merkittäviä tieteellisiä julkaisuja. Ensimmäinen tarkkoja algoritmeja vastaava ja reaaliaikaisen prosessoinnin mahdollistava menetelmä on vain 6 vuotta vanha ja siihen johtaneet välivaiheet on syytä huomioida. Aikajana perustuu esiteltyjen menetelmien kehitykseen vaikuttaneisiin julkaisuihin.

Varhaisimmat tutkimukset aiheeseen liittyen on tehty aikana, jolloin tutkimukseen käytettäviä resursseja on ollut paljon niukemmin. Viimeisen kolmen vuosikymmenen aikainen kehitys tietokoneiden laitteistossa ja yhteydenpidon välineissä on ollut merkittävä. Jotta tutkimuksen haastavuus olisi paremmin esillä, esitämme lyhyet kuvaukset aikakausien kehityksen tasoista.

2.1 1980-luku

1980-luvun alussa kuvankäsittelyn mahdollistavat tietokoneet olivat kalliita. Vuosikymmenen aikana julkaistiin merkittäviä massatuotannon kotitietokoneita, jotka laskivat hintaa myös tutkimuskäyttöön tarkoitetuista tietokoneista. Vuonna 1980 julkaistiin Commodore VIC-20 kotitietokone ja aloitettiin MS-DOS käyttöjärjestelmän kehitys. Ensimmäinen IBM PC julkaistiin 1981, josta kehittyi ajan kuluessa teollisuuden standardi. Muita merkittäviä tietokoneita julkaistiin, kuten Commodore 64 (1982), Apple Lisa (1983) ja Commodore Amiga (1985). Myös Microsoft Windowsin kehitys aloitettiin vuonna 1985.

1981 Moravec kehitti menetelmän stereokuvan pääkohtien täsmäämiseen käyttäen kulmien tunnistusta. Tietokoneohjattu robotti ottaa 9 stereokuvaa liikuttamalla kameraa sivuttain. Kuvista etsitään kulmat ja määritetään niiden kuvaajat, joita verrataan edellisen sijainnin kuvaajiin samojen objektien löytämiseksi. Etäisyys objekteihin

lasketaan yhdeksän kuvan stereokuvasta löytyneen täsmäävän pisteen perusteella. Robotin siirtyminen metrin verran, kuvien ottaminen ja niiden käsittely kestää robotilta 10-15 minuuttia. [4]

1983 Kuvan mittasuhteiden muutoksista riippumattomien sijaintien havaitseminen voitiin suorittaa etsimällä vakaita piirteitä eri mittakaavoista käyttäen jatkuvaa skaalan funktiota, joka tunnetaan myös nimellä *scale space.* [5]

1984 Crowley ja Parker kehittivät esityksen, joka tunnistaa huippuja ja harjanteita skaala-avaruudessa ja yhdistää ne puurakenteeksi, joka voidaan yhdistää kuviin erilaisilla mittakaavan muutoksilla. [6]

1988 Harris ja Stephens paransivat Moravecin kulman tunnistajaa. Sen sijaan, että kulmia yritettiin löytää etsimällä täsmääviä pisteitä stereokuvista, tarkasteltiin autokorrelaation eroa suunnan suhteen korrelaatioikkunan avulla. [7]

2.2 1990-luku

1990-luvun alussa yleinen prosessori henkilökohtaisissa tietokoneissa oli Intel 80286 (12 MHz) ja 80386SX (16 MHz), joilla pystyi suorittamaan Microsoft Windows 3.x käyttöjärjestelmää, mutta joille aloitettiin myös Linux kernelin työstäminen vuonna 1991. Ensimmäinen web-selain, Mosaic, julkaistiin vuonna 1993. 32-bittiset keskusyksiköt yleistyivät ja 32-bittinen Windows 95 käyttöjärjestelmä julkaistiin vuonna 1995. Internetiin yhdistäminen tapahtui yleensä 14.4k - 56k modeemilla. Vuosikymmenen aikana 3D-pelit saavuttivat suuren yleisön ja tehokkaimpien keskusyksikköjen kellotaajuus läheni 1 GHz rajaa.

1992 Harris osoitti, kuinka arvokas kulman tunnistaja on tehokkaassa liikkeen tunnistuksessa ja 3D-mallin rakentamisessa. [8]

1994 Lindeberg tutki perusteellisesti piirteen tunnistuksessa olevaa mittakaavan muuttumattomuuden ongelmaa. [9]

1995 Zhang ja muut osoittivat, että on mahdollista täsmätä Harrisin kulmat kahden kuvan välillä käyttämällä korrelaatioikkunoita kulmien ympärillä vastaavien kulmien löytämiseen. [10]

1997 Schmid ja Mohr osoittivat, että piirteiden vertaamista voidaan käyttää yleisesti kuvantunnistuksen ongelmissa, joissa piirrettä verrataan suureen kuvien tietokantaan. He myös käyttivät Harrisin kulmia pääkohtien tunnistamiseen, mutta sen sijaan että he käyttäisivät korrelaatioikkunaa piirteiden vertaamiseen, he käyttivät tasossa kääntymisestä riippumatonta kuvaajaa paikallisessa kuva-alueessa. [11]

1997 Basri ja Jacobs demonstroivat uuden tavan perspektiivistä riippumattomaan piirteiden vertaamiseen. [12]

1998 Nelson ja Selinger esittivät hyviä tuloksia pääkohtien löytämiseen perustuen kuvien ääriviivojen ryhmittymiin. [13]

1999 David Lowen aiemmat työt kehittivät piirteiden kuvaamista, jotta saavutettaisiin riippumattomuus mittakaavasta. Uusi julkaisu esitti uuden piirteiden kuvaajan, joka on paljon selkeämpi ollen samalla vähemmän altis paikalliselle kuvan vääristymiselle, kuten 3D katselukulman muutokselle. [14]

1999 Shokoufandeh ja muut esittivät graafiin perustuvan objektien piirteiden täsmäämisen. [15]

2.3 2000-luku

2000-luvulla julkaistiin ensimmäiset 1 GHz kellotaajuuden keskusyksiköt: AMD Athlon ja Intel Pentium III julkaistiin maaliskuussa 2000. Uusia käyttöjärjestelmiä julkaistiin: Microsoft Windows XP vuonna 2001 ja Ubuntu linux vuonna 2004. Ensimmäiset 64-bittiset ja moniytimiset prosessorit julkaistiin kuluttajakäyttöön 2005 ja ne yleistyivät vuosikymmenen loppua kohden. Netbook-tyyliset tietokoneet saavuttivat suosion Asus Eee kannettavan tietokoneen myötä vuonna 2007. Älypuhelimet saavuttivat suuren suosion Apple iPhonen (2007) ja Google Androidin (2008) myötä.

2000 Schiele ja Crowley esittivät moniulotteisten histogrammien käytön tehden yhteenvedot mittauksien jakautumisesta kuvan sisäisillä alueilla. Tämän tyyppinen piirre on erityisen hyödyllinen tunnistamaan objektit, joilla on vääristyneitä muotoja. [16]

2002 Pope ja Lowe käyttivät hierarkkiseen ryhmittelyyn perustuvaa kuvaajaa kuvan ääriviivojen piirteistä, jotka ovat erityisen hyödyllisiä objekteille, jonka tekstuureilla ei ole paljon yksityiskohtia. [17]

2002 Carneiro ja Jepson esittivät jaksoihin perustuvan kuvaajan, joka esittää jaksoa spatiaalisten taajuuksien sijaan, mikä todennäköisesti tarjoaa paremman muuttumattomuuden eri valaistuksissa. [18]

2003 Mikolajczyk ja muut kehittivät uuden kuvaajan, joka käyttää paikallisia reunoja ottamatta huomioon lähellä olevia piirteeseen kuulumattomia reunoja, tarjoten kyvyn löytää vakaita piirteitä jopa sotkuisen taustan kanssa päällekkäin olevien kapeiden muotojen lähellä. [19]

2004 David Lowe esitteli menetelmän nimeltään SIFT, jota käytetään yksilöllisten muuttumattomien piirteiden kuvaamiseen. Kuvaajia voidaan käyttää luotettavaan vertaamiseen eri kuvakulmista samaan objektiin tai tapahtumapaikkaan. [20]

2004 Ke ja Sukthankar esittivät hienosäätöä SIFT-algoritmille. Heidän kuvaajansa on erottelevampi vertailussa ja se vie vähemmän muistia. [21]

2006 Rosten ja Drummond esittivät koneoppimisella kehitetyn kulmantunnistimen nimeltään FAST, joka on yllättävän tarkka sen nopeuden lisäksi. [22]

2008 SURF ylitti viimeisintä tekniikkaa edustavat menetelmät nopeudessa ja tarkkuudessa. [23]

2.4 2010-luku

2010-luvulla pienet ja vähän sähköä kuluttavat tietokoneet yleistyivät. Passiiviset keskusyksikön jäähdytysratkaisut tulivat mahdollisiksi ja SSD-kovalevyt veivät viimeisetkin liikkuvat osat kevyistä kotitietokoneista. Vuonna 2012 julkaistiin luottokortin kokoinen Raspberry Pi, joka loi menestyksellään uuden aallon pienille koteloimattomille ARM-prosessorin tietokoneille. Prosessointi yleiskäyttöisillä näytönohjaimilla ja Intel Phi kortin kaltaisilla laskentayksiköillä mahdollisti entistä vaativammat ohjelmat.

2010 Michael Calonder ja muut ehdottivat binäärijonojen käyttämistä tehokkaana piirteiden kuvaajana. Heidän menetelmä on nimeltään BRIEF. He osoittivat, että menetelmä on erittäin erotteleva vaikka käytettäisiin suhteellisen vähän bittejä. Kuvaaja voidaan lisäksi laskea käyttäen yksinkertaisia testejä intensiteettien eroista ja sen samankaltaisuutta voidaan arvioida käyttäen Hamming-etäisyyttä, joka on erittäin tehokas laskea. [24]

2011 Ethan Rublee ja muut esittivät erittäin nopean binäärisen kuvaajan, joka perustuu BRIEF menetelmään, nimeltään ORB. Se on muuttumaton kuvan kääntyessä tasossa ja kohinan kestävä. He havainnollistavat testin avulla, kuinka ORB on kaksi suuruusluokkaa nopeampi kuin SIFT ja suoriutuu yhtä hyvin monissa tilanteissa. [25]

2015 F. A. Khalifa ja muut mittasivat tehokkaimpien pääkohtien tunnistimien ja kuvaajien nopeudet ja tehokkuudet. He käyttivät mittauksissaan eri luokkiin kuuluvia kuvia. Tulokset osoittivat, että eri luokissa parhaimmat tunnistimet ja kuvaajat vaihtelevat. [26]

2016 Zhenyu Xu ja muut esittivät uudenlaisen tavan pääkohtien tunnistamiseen ja kuvaajien laskentaan, joka on nimeltään DFOB. Se on verrattavissa SIFT ja SURF menetelmiin tarkkuudeltaan, mutta se on jopa 50 kertaa nopeampi kuin SURF menetelmä. [27]

2.5 Tulevaisuus

Nykyisen kehityssuunnan perusteella keskitytään tehokkuuteen ja erilaisiin nopeisiin binäärijonoihin perustuviin algoritmeihin. Viimeisimmät menetelmät ovat myös olleet patentoimattomia, mikä on ollut osa suurempaa vapaan lähdekoodin kulttuuria.

Lisätyn todellisuuden katselulaitteista on mahdollista tulla arkipäiväistä työntekoa helpottavia välineitä, jolloin niiden tarjontaan tulisi kilpailua yrityksille isoissa määrissä myytävistä edullisista perusmalleista, mikä mahdollistaisi paremmin myös harrastelijoiden kevyen ohjelmistokehityksen. Jos katselulaitteet jäävät kalleimmaksi kulutuselektroniikaksi, voi kehityksen odottaa jäävän keskittymään ainoastaan varmoihin myyntiartikkeleihin ja Internet-yhteisöjen keskustelun voi odottaa jäävän asiakaskokemusten vaihtamisen tasolle.

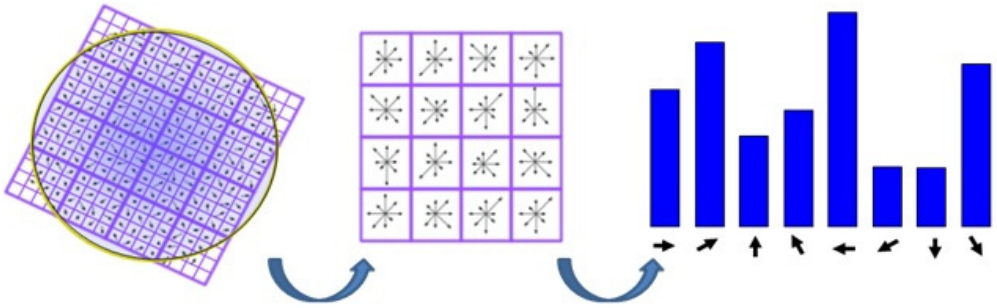
3. KESKEISET REAALIAIKAISET MENETELMÄT

Pääkohdat ovat kiinnostavia ja ympäristöstä erottuvia pisteitä kuvassa. Niitä voidaan havaita useilla eri menetelmillä. Samankaltaiset pääkohdat voidaan löytää eri kuvista ja niitä voidaan käyttää kuvan siirtymän ja kääntymisen mittaamiseen. Pääkohdat voidaan ilmaista kääntymisestä tai mittakaavasta riippumatta, mutta jotkin menetelmät eivät ota niitä huomioon. Konenäön sovellukset vaativat usein menetelmän, joka löytää täsmäävät pisteet kahdesta eri kuvasta mahdollisimman nopeasti.

3.1 SIFT

Yksi konenäön alueella paljon vaikuttaneista tieteellisistä julkaisuista on uraauurtava SIFT artikkeli. [20] Menetelmää on käytetty monissa sovelluksissa, mutta se on laskennallisesti monimutkainen, eikä sovellu sovelluksiin, joilla on rajoitetusti laskenta-aikaa. Menetelmän tuottamat piirteiden kuvaajat ovat muuttumattomia kokoon ja kääntymiseen nähden. Kuvaajien on osoitettu pysyvän lähes samana lievän affiinisen muutoksen, 3D-katselupisteen muutoksen ja häiriön lisäyksen jälkeen.

SIFT tunnistin koostuu neljästä eri vaiheesta, jotka ovat kuvan ääriarvojen tunnistus eri mittakaavoissa, pääkohtien paikannus, suunnan määritys ja pääkohtien kuvaajien laskeminen. Ensimmäisessä vaiheessa kuvasta tunnistetaan ääriarvojen pisteet laskemalla kuvan eri mittakaavojen Gaussian suodatusten erotukset, mikä on muuttumatonta mittakaavassa ja suuntautuneisuudessa. Toisessa vaiheessa merkityksettömät pisteet karsitaan hylkäämällä matalan kontrastin pisteet ja reunoihin sijoittuvat pisteet. Jotta piirteiden kuvaajista saataisiin muuttumattomia suunnan muuttuessa, luodaan histogrammi pääkohdan lähialueiden suunnille käyttäen niiden intensiteettien erotusta. Viimeinen vaihe SIFT menetelmällä on määrittää histogrammista voimakkain suunta. [Kuva 5.]

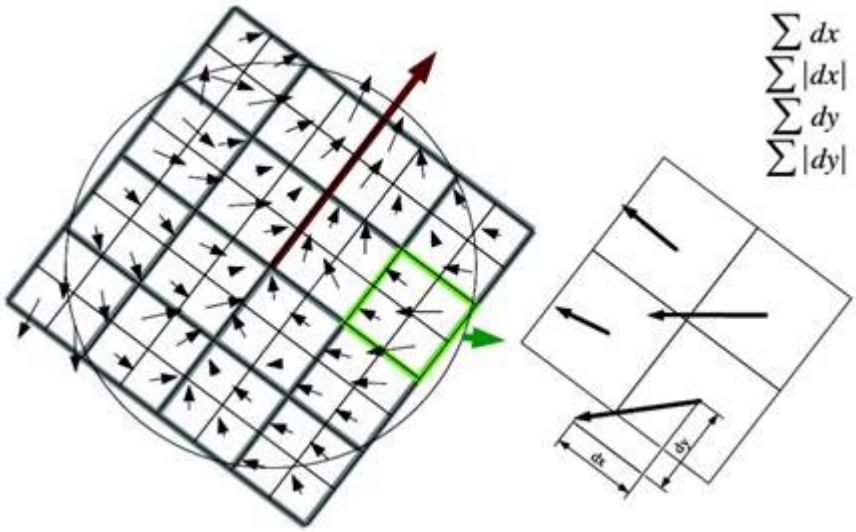


Kuva 5. Pikselien intensiteettimuutokset käännettynä laskettuun suuntaan, siitä muodostettu SIFT kuvaaja ja kuvaajasta määritetty suuntien histogrammi.

3.2 SURF

Monet sovellukset sisältävät paljon dataa, jonka käsittelyyn SIFT prosessi vie liikaa aikaa. Herbert Bay ja muut esittivät SURF menetelmän nopeampana piirteiden kuvaajana. [23] Molemmat menetelmät tuottavat pääkohtia ja kuvaajia, jotka ovat muuttumattomia mittakaavaan ja kääntymiseen nähden.

SURF menetelmä käyttää Haar-aalloke muunnosta ja integraali-kuvaa. Pääkohdat tunnistetaan laskemalla Haar-aalloke vasteiden summat tarkasteltavan pisteen ympäriltä. Kuvaaja lasketaan jakamalla pääkohtien ympäröimä alue 4x4 ruudukkoon ja laskemalla Haar-aalloke vaste sen neliöille. [Kuva 6.] Jokainen ruutu määrittää 4 arvoa kuvaajaan, joten kuvaaja on 64-arvoinen. [28]

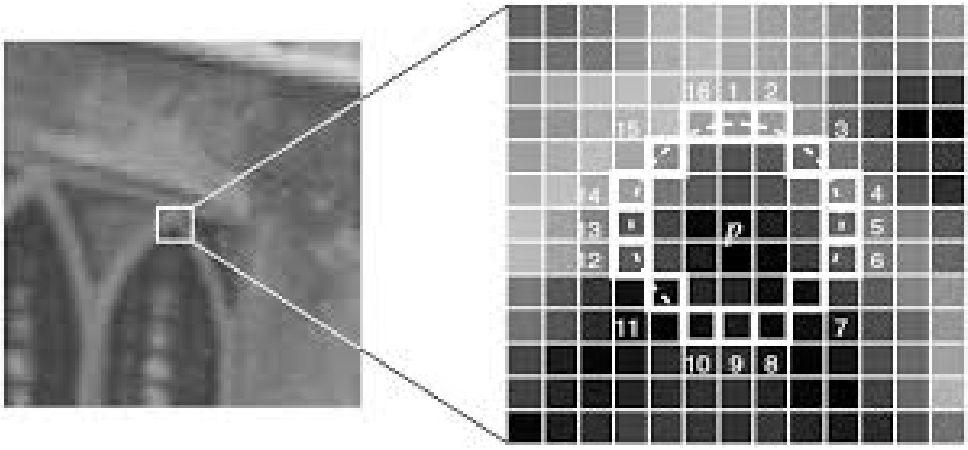


Kuva 6. SURF pääkohtien tunnistus laskemalla Haar-aalloke vasteiden summat tarkasteltavan pisteen ympäriltä. [23]

3.3 FAST

FAST on kehitetty nopeaksi kiinnostavien pisteiden tunnistimeksi. [22] Se toimii tehokkaasti reaaliaikaisessa käytössä, mutta ei sisällä pisteiden suunnan tunnistamista, mikä heikentää pisteiden seuraamista kuvan kääntyessä tasossa.

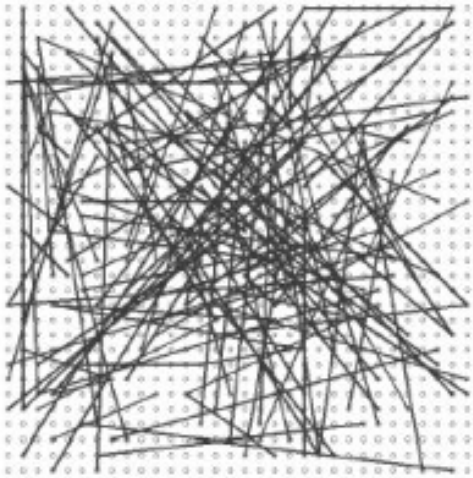
FAST käyttää 16 pikselin ympyrää määrittääkseen kulmapisteet. [Kuva 7.] Jokainen ympyrän piste merkitään myötäpäivään kokonaisluvuilla 1-16. Jotta menetelmä olisi nopeampi, ensiksi verrataan intensiteetit pikseleistä 1, 5, 9 ja 13 ympyrällä. Jos vähintään kolme neljästä pikselistä ylittää kynnysarvon, niin piste on kiinnostava. Toisaalta, jos vähintään kolme neljästä pikselistä jää kynnysarvon alle, niin piste varmasti ohitetaan. Jos piste on kiinnostava, niin ympyrästä vaaditaan lisäksi 12 jatkuvaa kynnysarvon ylittävää pikseliä, jotta se voidaan merkitä kulmaksi. [28]



Kuva 7. FAST kuvaajan ympyräkuvio tarkasteltavan pisteen ympärillä.

3.4 BRIEF

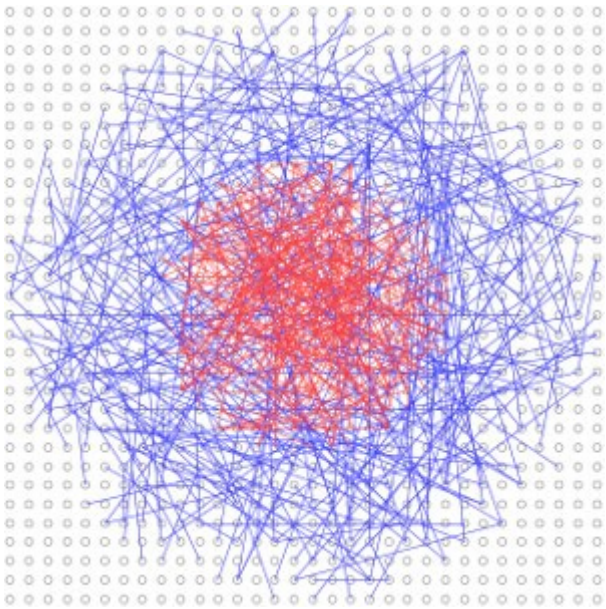
BRIEF on parillinen binäärimuotoinen kuvaaja. Tietty määrä (128, 256 tai 512) pareja valitaan satunnaisesti pysyviin sijainteihin pisteen ympärille. [Kuva 8.] Kuvaaja lasketaan vertaamalla intensiteettejä näiden parien välillä. Vertaus tuottaa arvon 1, jos intensiteetti ensimmäisessä on isompi kuin toisessa, mutta muuten se tuottaa arvon 0. BRIEF kuvaajan vertaaminen toiseen on laskennallisesti nopeampaa kuin SIFT tai SURF kuvaajan, koska se perustuu binäärijonojen vertaamiseen, joka on kahden käskyn operaatio moderneissa keskusyksiköissä (xor ja popcount). BRIEF on muuttumaton valaisuun nähden, mutta ei mittakaavaan tai kääntymiseen. [29]



Kuva 8. BRIEF tunnistimen satunnaisesti valittuja pareja mittauspisteen ympäriltä yhdistettynä viivoilla.

3.5 BRISK

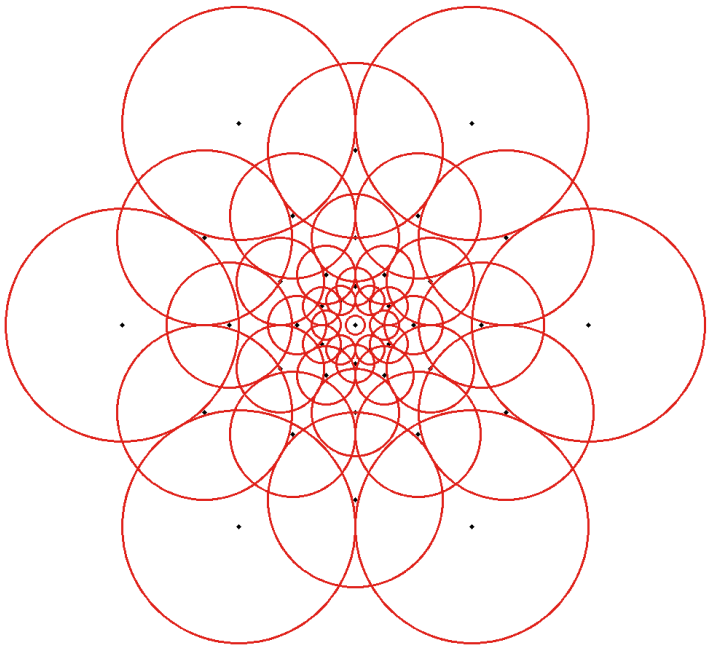
BRISK on binäärimuotoinen kuvaaja ja hyvin samankaltainen kuin BRIEF, mutta kehittyneempi. Sen kuvaajat ovat muuttumattomia mittakaavaan ja suuntaan nähden. Pisteen ympäriltä valitaan satunnaisesti pareja, kuten myös BRIEF menetelmässä, mutta ne jaetaan kahteen osaan: lyhyen etäisyyden ja pitkän etäisyyden pareihin. [Kuva 9.] Pitkän etäisyyden pareille määritetään intensiteettien erotusten suunnat ja niitä käytetään pisteen suunnan määrittämiseen. Lyhyen etäisyyden parit käännetään käyttäen pisteen suuntaa. Kuvaaja kootaan yhdistämällä lyhyen etäisyyden parien intensiteettien vertaamisen tulokset ja se on pituudeltaan 512 bittiä.



Kuva 9. BRISK kuvaajan satunnaiset lyhyen etäisyyden parit punaisella ja pitkän etäisyyden parit sinisellä.

3.6 FREAK

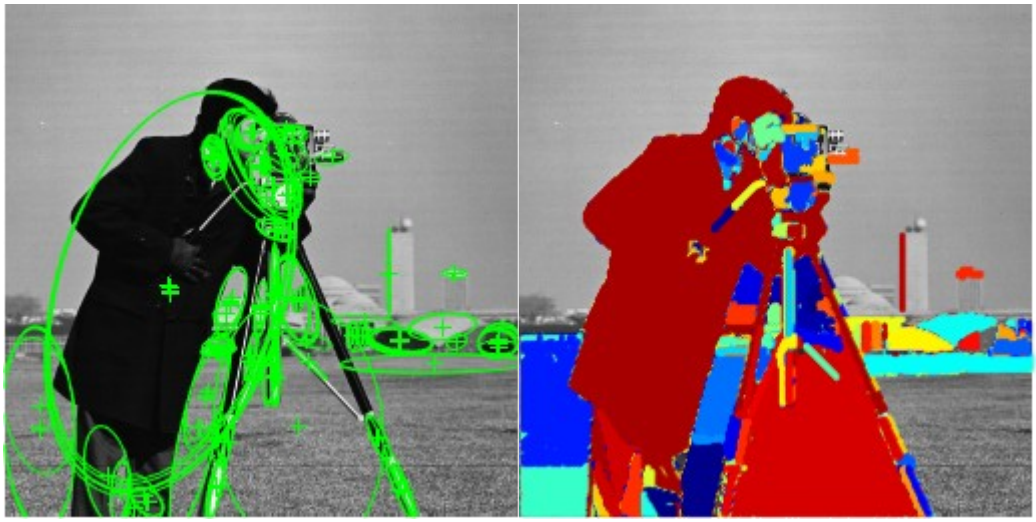
FREAK on binäärimuotoinen kuvaaja, jota on inspiroinut ihmisen näkökyky. Sen näytteenoton kuvio on verkkokalvon mukainen. [Kuva 10.] Kuvaaja määritetään muiden binäärimuotoisten kuvaajien tapaan vertaamalla kuvion pikselien intensiteettejä. Pääkohdan suunta määritetään symmetristen kenttien pareja verraten, joita valitaan 45, mikä on paljon vähemmän kuin useampi sata BRISK:ssä. Mittaavat alueet ovat myös paljon isompia verkkokalvon mukaisen alueen ulkoreunalla, mikä aiheuttaa suuremman virheen laskettuun suuntaan. [30]



Kuva 10. FREAK algoritmin verkkokalvon mukainen kuvio.

3.7 MSER

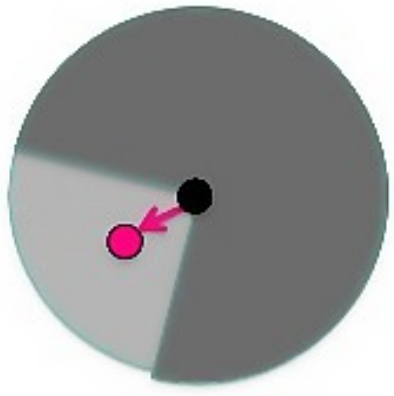
MSER on pääkohtien tunnistin, ilman kuvaajan määritystä. Se tunnistaa yhtenäisiä alueita Watershed algoritmia vastaavalla tavalla. Siinä vaihdetaan kynnysarvoa kirkkaimmasta tummimpaan ja etsitään alueita, joiden pinta-ala muuttuu vain vähän kynnysarvon vaihtuessa. [31, 32]



Kuva 11. MSER:in tunnistamat pääkohdat ja kynnysarvolla rajatut alueet.

3.8 ORB

ORB tunnistin on FAST ja BRIEF tunnistimien yhdistelmä. [25] Se lisää muuttumattomuuden mittakaavassa käyttäen kuvaa eri mittakaavoissa ja pyramidin kaltaisessa rakenteessa. Jokaisessa eri mittakaavassa tunnistetaan kulmat FAST tunnistimella. Tunnistettuihin kulmapisteisiin tehdään lisäksi Harris kulmantunnistus ja vain parhaimmat kulmat kelpuutetaan. Pääkohdan suunta määritetään mittaamalla sen ympäröivän alueen intensiteetillä painotetun keskipisteen suunta pääkohtaan nähden. [Kuva 12.] Kuvaaja lasketaan BRIEF menetelmällä pääkohdan suunnan mukaisesti. [32]



Kuva 12. Pääkohdan suunnan määrittäminen intensiteetillä painotetun keskipisteen suunnan perusteella.

4. ILMAKIEKKOPELIN TOTEUTUS

Sovelluksen viiveen minimoimiseen vaikuttaa käytettävät menetelmät. Ohjelmassa on paljon kuvankäsittelyä, joka kannattaa tehdä näytönohjaimella kiihdytetyllä prosessoinnilla, mutta se jää tämän työn ulkopuolelle aiheen monimutkaisuuden vuoksi. Kuvankäsittelyä lukuunottamatta merkittävin tekijä viiveelle on lisätyn todellisuuden toiminnot, jotka tässä sovelluksessa sisältävät pelikentän pohjan tunnistamisen web-kameran videosta.

Ilmakiekkopelin pääpiirteinen suunnitelma on tunnistaa kentän pohja ja pelimaila kuvasta, viedä sijainti virtuaaliseen ilmakiekkopeliin ja piirtää pelikenttä ympäristön kuvan päälle. OpenCV esimerkki esittää oleellisen toiminnon ohjelmassa. [Kuva 4.]

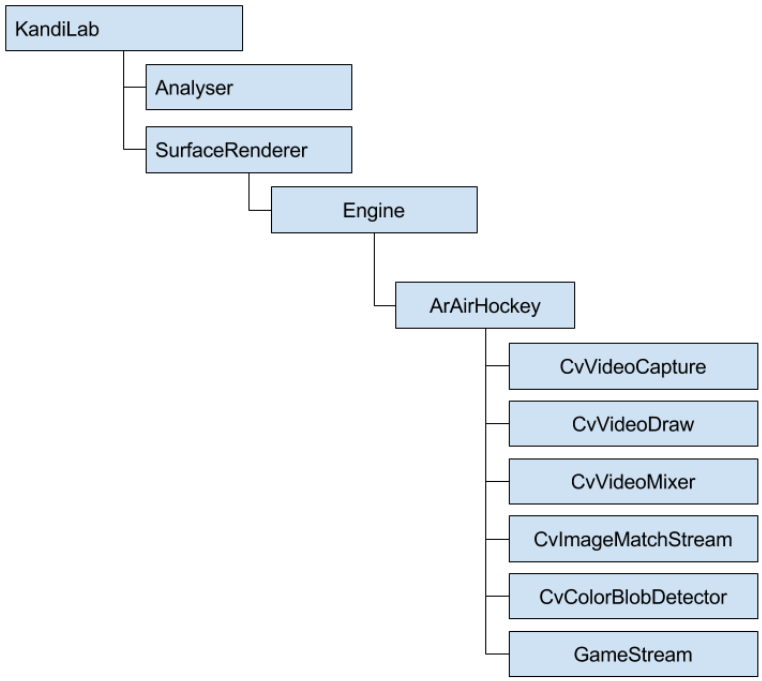
4.1 Kehitysympäristö

Ohjelmointikielenä sovelluksen toteuttamiseen käytetään C++ kieltä sen tehokkuuden ja monipuolisuuden vuoksi. Ohjelmointiin on mahdollista käyttää useita eri ohjelmointiympäristöjä, joista tämän sovelluksen tekoon käytetään Ultimate++ r9251 versiota sen järjestelmäriippumattomuuden ja hyvän ohjeistuksen takia. [33] Kuvan pääkohtien tunnistukseen ja kuvankäsittelyyn käytetään OpenCV kirjaston versiota 2.4.13.

Ilmakiekkopelin virtuaalinen versio vaatii fysiikkamallinnusta. Internetissä on ladattavissa useita C++ kielen fysiikkakirjastoja eri lisensseillä ja vaihtelevalla kehitysyhteisön tarjoamalla sisällöllä. Pelissä käytetään Box2D kirjastoa, jonka esimerkit nopeuttavat kehitystä huomattavasti. Kaikki pelin komponentit ovat yhteensopivia eri käyttöjärjestelmien välillä. Peliä on testattu Windows, Linux ja FreeBSD käyttöjärjestelmillä.

4.2 Suunnittelu

Ohjelman rakenne on yksinkertainen ja sisältää vähän merkittäviä eri osia.



Kuva 13. Ohjelman luokkien hierarkia.

Pääohjelman nimi on KandiLab. Se käyttää Analyser luokkaa suorituskykyvertailun suorittamiseen. SurfaceRenderer piirtää ruudulle kuvan pelin nykyhetkestä, joka on saatu Engine luokkaa käyttäen. Engine luokassa kutsutaan ArAirHockeyn kuvankäsittelyn eri komponentit järjestyksessä. [Kuva 13.]

Kuvaformaattien vaihto OpenCV:n ja Ultimate++ piirturin välillä tehtiin lopulta kopioimalla yksittäisiä pikseleitä, mikä lisäsi ylimääräistä viivettä. Pikseleiden kopioiminen ei ole optimaalista tehokkuuden suhteen, mutta se on helppo ymmärtää ja ohjelmoida.

4.3 Kuvankäsittelyn komponentit

Ilmakiekkopelin kuvankäsittelyn keskeisimmät komponentit ovat kentän pohjan tunnistus kuvasta, mailan tunnistus kentän alueelta, pelikentän renderointi ja piirto alkuperäisen kuvan päälle. Kentän pohja tunnistetaan web-kameralla otetusta kuvasta käyttäen ennalta otettua kuvaa kentän pohjasta. Molemmista kuvista etsitään pääkohdat ja niiden kuvaajia verrataan täsmäävien pisteiden tunnistamiseksi. Tuloksena on yhtenäisiä pisteitä, joiden alueella arvioidaan homografia, eli projektiivinen muutos alkuperäiseen kuvaan nähden, jota käyttäen voidaan määrittää alkuperäisen kuvan kulmapisteet ympäristöstä otetussa kuvassa. Nelikulmion sisältämä kuva muunnetaan suorakulmion muotoiseksi ja siitä tunnistetaan pelimaila. Pelimailana käytetään paperinpalaa, johon on piirretty punainen ympyrä. Suorakulmioksi muutetusta kuvasta suodatetaan punainen kanava ja siitä tunnistetaan suurin yhtenäinen alue, jonka sijainti viedään pelikentälle pelaajan mailan sijainniksi.

4.4 Mailan tunnistus

Mailan tunnistuksessa kuva voi olla esimerkiksi RGB-muodossa tai HSV-muodossa. RGB-kuva koostuu kolmesta kanavasta, jotka ovat punainen, vihreä ja sininen. HSV-kuva koostuu myös kolmesta kanavasta, jotka ovat värisävy, värikylläisyys ja valoisuusarvo. Jälkimmäinen muoto on tunnistuksessa parempi, koska suodatuksessa voidaan rajata erikseen värien lisäksi valoisuutta. HSV arvojen rajat määritetään, jolloin kuvasta saadaan suodatettua haluttu väri. Tuloksena on binäärikuva, jossa 0 on musta ja 1 valkoinen. Näistä valkoinen on mailan suodatettu väri.

Ilmakiekkopelissä käytetään punaista objektia mailana, eli tällöin suodatetaan punaisen värin sävyt kuvasta. OpenCV:n HSV-kuvan värisävyn arvot vaihtelevat 0—179:n välillä. Näistä punainen sijaitsee noin 0—10:n ja 175—179:n välillä. Värikylläisyys ja valoisuusarvo saavat arvoja väliltä 0—255. Kynnysarvoilla suodattamisen jälkeen kuva sisältää vielä yksittäisiä valkoisia pisteitä, mikä johtuu kohinasta. Näistä yksittäisistä pisteistä pääsee eroon käyttämällä morfologisia operaatioita: eroosio ja dilaatio. Ensimmäisenä käytetään eroosio suodatinta, joka määritetään mailan halutun minimikoon mukaan ykkösiä sisältäväksi neliömatriisiksi. Mailan täytyy tällöin olla vähintään eroosio suodattimen kokoinen kuvassa, jotta se jäisi näkyviin suodatettuun kuvaan. Jos kuvassa ei ole vähintään suodattimen kokoista kohinaa, yksittäiset kohinan aiheuttamat pisteet suodattuvat pois. Tämän jälkeen voidaan käyttää vielä dilaatiota suodatettavan objektin alkuperäisen koon varmistamiseksi ja mahdollisten aukkojen täyttämiseksi. Lopuksi kynnysarvoilla suodatetusta kuvasta tunnistetaan suurin yhtenäinen valkoinen alue, jonka keskipiste määritetään mailan sijainniksi.

4.5 Pelin kulku

Ennen pelin alkua ohjelman avulla otetaan kuva kentän päältä. Pelin aikana laitteella voi katsoa kenttää eri kulmista ja ohjelma tunnistaa perspektiivin muuttumisen automaattisesti. Pelaaja pelaa tietokonetta vastaan käyttäen mailana paperinpalaa, johon on piirretty punainen ympyrä, jonka ohjelma tunnistaa mailaksi.

Pelaajat ovat kentän vastakkaisilla puolilla ja heillä on kentällä pyöreät mailat. Pelaajat yrittävät saada pyöreän kiekon vastakkaiseen maaliin ja ensimmäinen 7 maalia saanut pelaaja voittaa.

4.6 Ohjelmointi

Ohjelmointi tapahtuu tarkentamalla suunnitelman yksityiskohtia ja kirjoittamalla niitä koodiksi dokumentaatiota, valmiita esimerkkejä ja ohjelmoijan kokemusta hyödyntäen. Vertailukohteetonta ohjelmaa kirjoittaessa täytyy myös olla valmis kehittämään tavoitetta. Välivaiheet pyritään pitämään käännettävissä toimivaksi ohjelmaksi.

Pelin ohjelmointi vaatii fysiikkakirjaston oikeaa käyttöä, joka selviää parhaiten esimerkeistä. Pelin objektit ovat myös fysiikkakirjaston objekteja. Maalin tekemisen tunnistus vaatii erikseen maalin ja kiekon törmäyksen tunnistamista. Peli vaatii vähintään yhden tietokonepelaajan, jonka yksinkertaisin logiikka on vain töniä kiekkoa, jos se on omalla alueella. Myös kahden tietokonepelaajan ottelun tarvitsee olla toimiva.

Web-kameran kautta tapahtuva pelaaminen on toteutettavissa vasta, kun virtuaalinen peli kahden tietokonepelaajan välillä toimii. Pelaajan koordinaatit siirretään pelikenttään, mikä tapahtuu muuntamalla tunnistettu pelialue nelikulmiosta suorakulmioksi ja selvittämällä siitä suurimman punaisen alueen keskipiste. Kameran kuvan kohina aiheuttaa pientä häiritsevää liikettä pelaajan mailaan ja sitä lievennetään laskemalla sijainnin keskiarvo viidestä viimeisimmästä sijainnista.

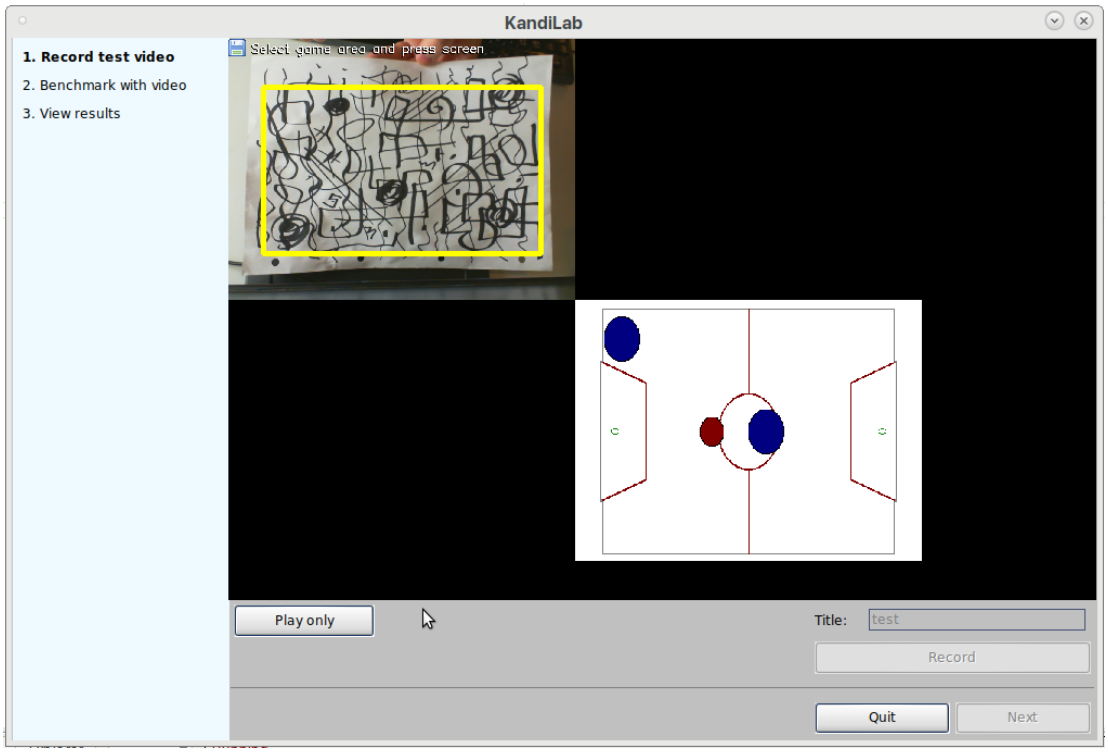
Pelikentän taustalla on vaikutusta sen tunnistettavuuteen. Taustan täytyy sisältää riittävän suuria piirteitä, jotta siitä löytyy pääkohtia. Mustavalkoinen ja paljon ainutlaatuisia muotoja sisältävä paperi kentän taustana pysyy paremmin seurattavissa, kuin kirjan tai dvd-kotelon kansi. Hyvä valaistus on silti tarpeellinen myös parhaimmalla alustalla.

4.7 Lopputulos

Peliä testanneet ovat sanoneet pelin olevan pelattava ja että peli reagoi hyvin mailan liikkeeseen. Testaajat eivät meinanneet uskoa, että kameraa voi liikutella kentän pysyessä silti oikeassa kohdassa. Viive on edelleen huomattava grafiikan piirtämisen hitauden vuoksi, mutta rauhalliset pelimailan liikkeet pysyvät silti riittävän viiveettömän tuntoisina.

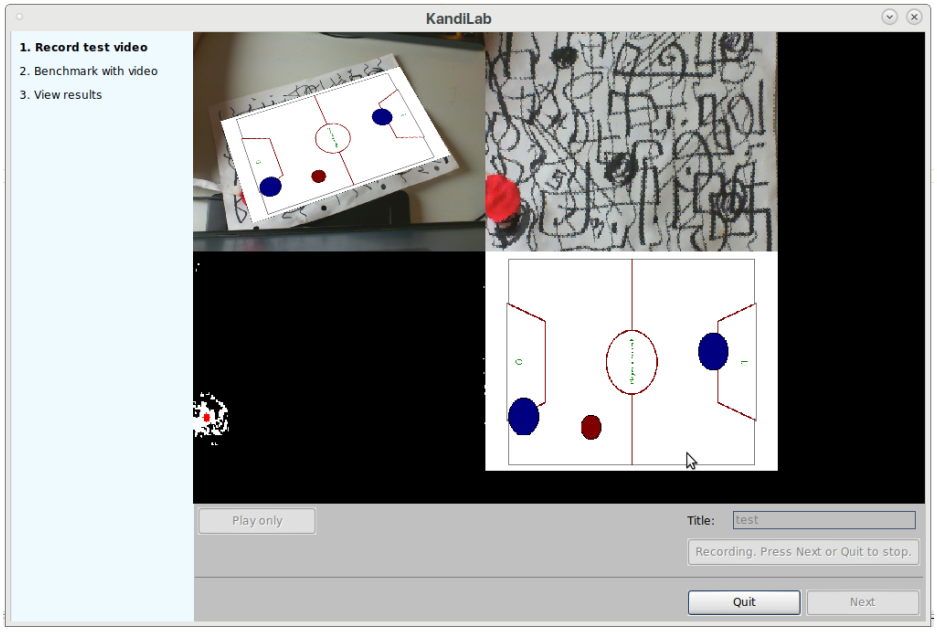
5. PELIN HYÖDYNTÄMINEN VIIVEEN MITTAUKSESSA

Ilmakiekkopeli ei vertaile erilaisia pääkohtien tunnistimia ja piirteiden kuvaajia, joten se tarvitsee toteuttaa erikseen. Pelin ympärille rakennetaan KandiLab ohjelma, joka käyttää pelaamisen aikana nauhoitettua videota ja toistaa pääkohtien tunnistuksen ja piirteiden kuvaamisen eri algoritmeilla. Testivideosta tulee samanlainen kuin oikeassa käyttötilanteessa. Sitä ei kuitenkaan voi käyttää tarkkuuden mittaukseen, koska se ei sisällä tietoa oikeasta homografiasta. Mittausohjelma on modulaarinen ja siihen pystyy lisäämään tarkkuusmittaukset tarvittaessa, mutta oikeiden homografioiden määrittäminen testivideon kuviin täytyy tehdä käsin, mikä on työlästä.



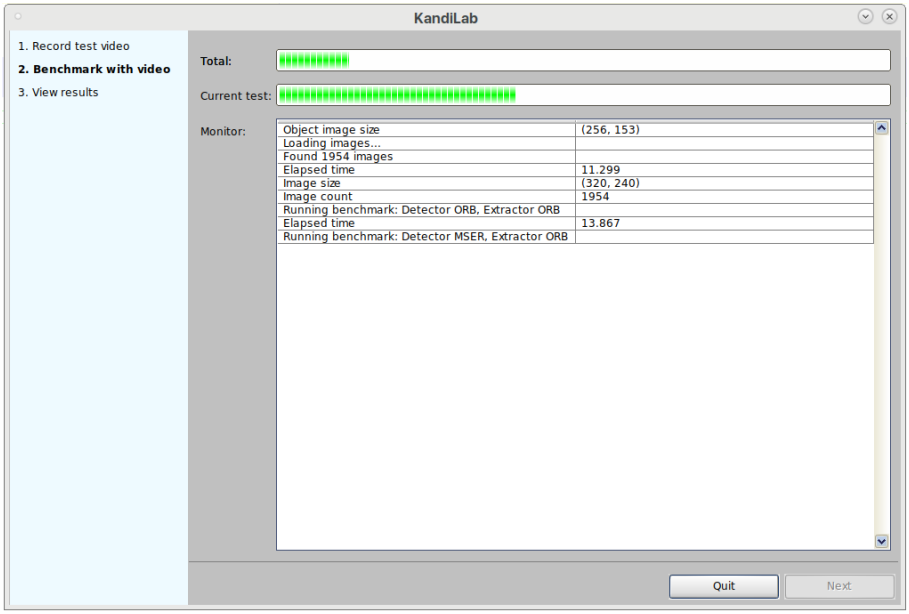
Kuva 14. Pelialustan kuvaaminen.

Peli on sisällytettynä mittaustulokset tekevään ohjelmaan. Ennen pelaamista kuvataan pelialusta. [Kuva 14.] Kun testi halutaan suorittaa, nauhoitetaan video, jota käytetään kaikkien menetelmien testaamiseen.



Kuva 15. Pelaaminen ja testivideon nauhoitus.

Mittausohjelma näyttää pelin lisäksi myös välivaiheet. [Kuva 15.] Pelin keskeiset komponentit ovat pelikentän tunnistus ja sen muunnos nelikulmioksi, punaisen pelimailan tunnistus pelikentän alueelta, mailan koordinaatin vienti virtuaaliseen ilmakiekkopeliin ja virtuaalisen pelin piirtäminen havaitun pelikentän päälle.



Kuva 16. Toisessa vaiheessa vertaillaan pääkohtien tunnistajat ja piirteiden kuvaajat.

Vertailun suoritus kestää minuutteja lyhyelläkin videolla. Ohjelma näyttää etenemispalkin, ajankäytön ja käytetyn videolähteen tiedot. Tulokset tallennetaan tekstitiedostoon ohjelman kansiossa automaattisesti. [Kuva 16.]

6. TULOKSET

On olemassa paljon erilaista aineistoa, jota voi käyttää mittauksissa. Eräs tunnettu ja tarkkuusmittauksissa [32,3] usein käytetty kuvasarja on Oxford Dataset, jossa on 8 kuvien luokkaa, 6 kuvaa luokassa ja kuvien mukana on niiden oikea homografia-matriisi. [34] Tässä mittauksessa keskitytään kuitenkin nopeuteen, eikä kuvasarjat sisällä videoaineistoa, joka vastaa käyttökohdetta.

Pääkohtien tunnistimien ja piirteiden kuvaajien nopeuden mittaamiseksi on esitetty yhtenäinen protokolla. [28] Niiden nopeudet ovat tärkein tekijä viiveen minimoinnin kannalta kuvankäsittelyä lukuun ottamatta. Tässä luvussa esitetään viimeisimpien menetelmien suorituskykymittaukset. Kaikki tunnistimet ja kuvaajat on suoritettu niiden oletusasetuksilla.

6.1 Testiaineisto

Kaikki tulokset ovat laskettu 1954 kuvan testivideosta, joka sisältää nauhoitettua ilmakiekon pelaamista. Testivideo koostuu kolmesta eri osasta, jotka näkyvät kuvaajassa. [Kuva 18.] Ensimmäinen osa, noin ⅙ koko kuvasarjasta, koostuu taustan tunnistamisesta. Toinen osa, myös noin ⅙ , koostuu pelialustan ja kameran asettelusta pelattavaan muotoon, ja viimeinen ⅔ osa sisältää pelaamista ja keskenään hyvin samanlaisia kuvia. [Kuva 17.]

Kuva 17. Testivideon kolmen eri vaiheen tyypilliset kuvat: pelikentän kuvaus, kentän asettelu pelattavaan muotoon ja itse pelaaminen.

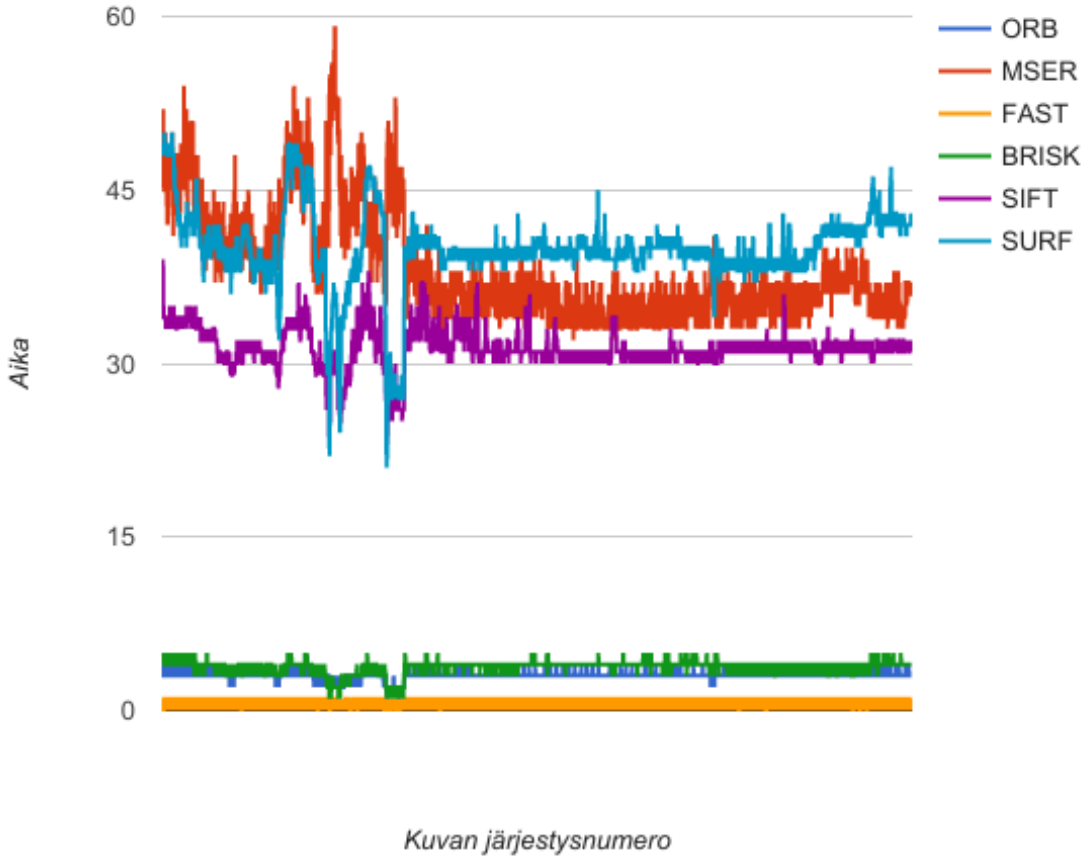
6.2 Pääkohtien tunnistimien vertaaminen

Tässä alaluvussa esitetään tulokset pääkohtien tunnistamisen kestolle ja löydettyjen pääkohtien määrälle. Taulukko 1 esittää tulokset yhden kuvan käsittelyn keskiarvolle.

Nopeutuksen arvo saadaan kaavasta $S_T = T_S / T_E$, jossa T_S on tunnistimen kesto ja T_E on tehokkaimman tunnistimen kesto.

Taulukko 1. Eri pääkohtien tunnistimien prosessoinnin kestojen keskiarvot.

Tunnistin	**Kesto [ms]**	**Nopeutus**	**Pääkohtia**
ORB	3,0	12,85	471
MSER	38,1	1,02	117
FAST	0,6	61,91	1527
BRISK	3,8	10,32	216
SIFT	31,3	1,25	682
SURF	39,9	1,00	733



Kuva 18. Yksittäisten kuvien pääkohtien tunnistamisen kestot millisekunneissa. Voimakas muutos johtuu kameran ja pelikentän taustan asettamisesta pelattavaan asentoon.

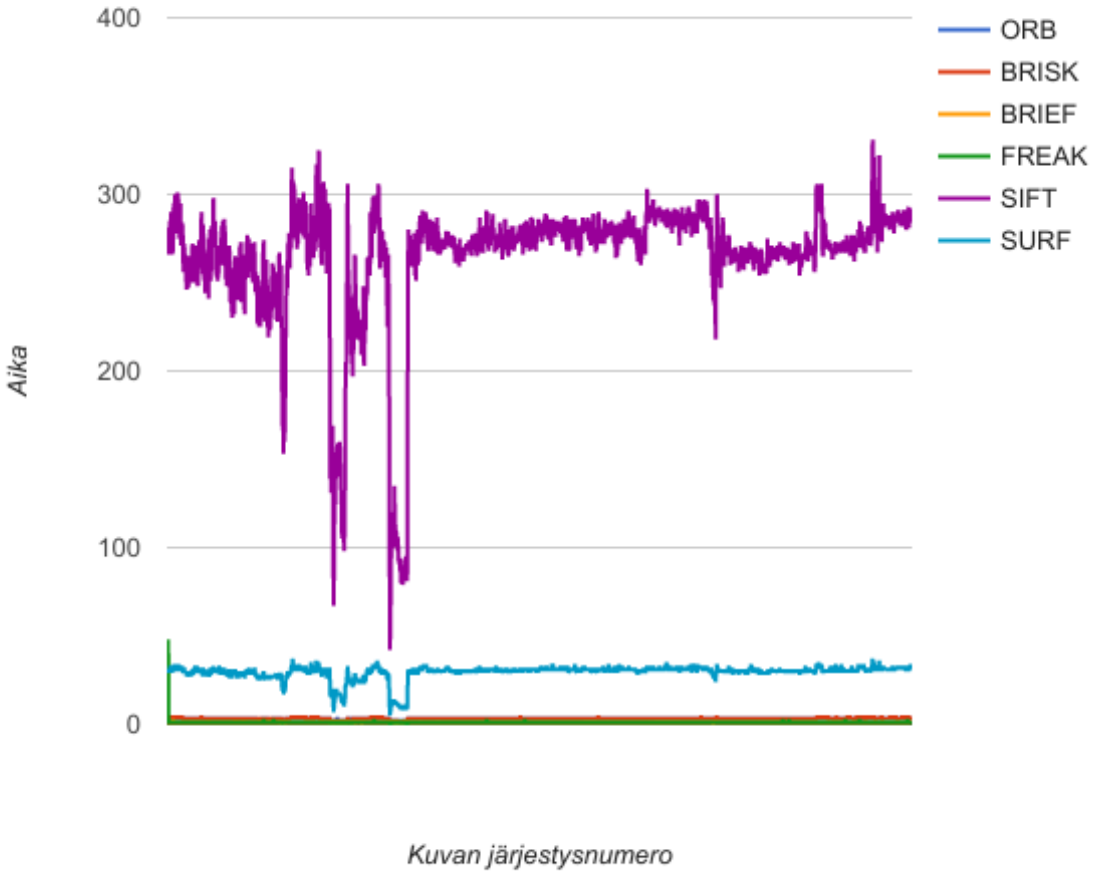
Kuvaajasta [Kuva 18.] näkyy kuinka binäärijonoihin perustuvat algoritmit ORB, FAST ja BRISK ovat paljon nopeampia kuin liukuluku-kuvaajiin perustuvat MSER, SIFT ja SURF. Nopein algoritmi FAST ei sisällä suunnan tai eri mittakaavojen tunnistusta ja on oletetusti nopein. Seuraavaksi nopeimmat pääkohtien tunnistajat, BRISK ja ORB, ovat binäärijonoihin perustuvia, kuten FAST, mutta sisältävät prosessoinnin eri mittakaavoille, mikä tekee niistä monta kertaa hitaampia kuin FAST. Käytettävyydeltään samankaltaisten ORB:in ja SIFT:in erona on se, että SIFT käyttää matematiikkasuoritinta kuvan intensiteettien vertailuun ja ORB on ohittanut sen nopeilla suorittimen operaatioilla. SIFT algoritmia nopeammaksi väitetty SURF on tunnistimen osalta hitaampi, mutta kuvaajan osalta nopeampi.

6.3 Kuvaajien vertaaminen

Tässä alaluvussa verrataan pitkään käytössä olleita kuvaajia, kuten SIFT ja SURF, viimeaikaisiin kuvaajiin, kuten ORB. Kaikki kuvaajat on laskettu SURF tunnistimen pääkohdille.

Taulukko 2. Eri kuvaajien prosessoinnin kestojen keskiarvot SURF pääkohdille.

Kuvaaja	**Kesto [ms]**	**Nopeutus**
ORB	2,98	88,68
BRISK	3,32	79,60
BRIEF	1,22	216,84
FREAK	1,32	200,25
SIFT	264,53	1,00
SURF	29,45	8,97



Kuva 19. Yksittäisten kuvien piirteiden kuvaajien määrittämisen kestot millisekunneissa.

Myös kuvaajien määrittämisen ajoissa näkyy testivideon kolme eri vaihetta. [Kuva 19.] Jälkimmäinen ⅔ on varsinaista pelaamista ilman kameran kuvakulman vaihtumista, mikä johtaa hyvin toistuvaan suoritukseen. Vanhimman SIFT algoritmin kuvaaja on hidas ja sitä uudempi SURF on onnistunut nopeuttamaan kuvaajan käsittelyä huomattavasti. Binäärijonoihin perustuvat ORB, BRISK ja BRIEF ovat nopeuden puolesta täysin eri luokkaa kuin vanhempi esikuva SIFT. Nopein algoritmi BRIEF ei sisällä suuntaan mukautumista. Binäärimuotoisista suuntaan mukautuvista algoritmeista FREAK on nopein, mutta sen laskemassa suunnassa on suurempi virhearvo kuin BRISK:issä.

7. YHTEENVETO

Tämä dokumentti esittää nopeusmittaukset viimeisimmistä pääkohtien tunnistimista ja piirteiden kuvaajista, joiden nopeuden suosiminen on merkittävin tekijä lisätyn todellisuuden sovelluksen viiveen optimoinnissa. Reaaliaikaiseen prosessointiin pidetään parhaimpana vaihtoehtona ORB:ia ja tarkkuutta vaativiin sovelluksiin SIFT:iä. Myös mittaukset osoittavat ORB:in olevan soveltuva reaaliaikaiseen prosessointiin.

Viimeisimmät binäärijonoihin perustuvat kuvaajat BRIEF, BRISK ja ORB ovat nopeampia kuin liukulukuihin perustuvat kuvaajat SIFT ja SURF, mutta niitä pidetään heikompina tarkkuudessa. Ne ovat hyviä algoritmeja lisätyn todellisuuden sovelluksiin, joilla on rajoitetusti aikaa käytössä yhden kuvan käsittelyä varten. Binäärijonoihin perustuvilla kuvaajilla on myös se etu, että ne käyttävät vähemmän muistia, jota voi olla myös rajoitetusti käytössä, kuten esimerkiksi mobiililaitteilla.

Yhteensä 6 pääkohtien tunnistajaa ja 6 piirteiden kuvaajaa verrattiin toisiinsa. ORB mitattiin parhaaksi ilmakiekkopelin kannalta, koska peli vaatii kuvaajan muuttumattomuuden mittakaavan tai perspektiivin muuttuessa, mikä ei toteudu nopeammilla menetelmillä, kuten BRIEF:llä.

Pelkkä valmiiksi kuvatun pelikentän etsiminen ympäristöstä ei riitä kaikkiin peleihin, mutta kehittyneemmätkin lisätyn todellisuuden pelit vaativat kameran liikkeen seuraamista. Esitelty menetelmä ilmakiekon toteuttamiseen on uudelleenkäytettävissä ja helppo ymmärtää. Viive on minimoitu niin pieneksi, että se mahdollistaa monet sovellukset älypuhelinten tasoisilla laitteilla. Tämä ilmakiekkopeli on mahdollista toteuttaa java-kielellä älypuhelimia varten ja sille saattaa olla kaupallista kysyntää. Pelimoottoria voi käyttää hyväksi muihin vastaavanlaisiin peleihin, kuten esimerkiksi virtuaaliseen minigolfiin.

Tämän kandidaatintyön ohessa tehty ilmakiekkopeli on julkaistu githubissa. [35]

8. LÄHTEET

[1] S. Mann and J. Fung. "EyeTap devices for augmented, deliberately diminished, or otherwise altered visual perception of rigid planar patches of real world scenes", *PRESENCE*, 2002, Vol. 11, No. 2, pages 158-175, MIT Press.

[2] OpenCV tutorial “Features2D + Homography to find a known object”, http://docs.opencv.org/2.4.10/doc/tutorials/features2d/feature_homography/feature_homography.html

[3] Mikolajczyk K. and Schmid C. A performance evaluation of local descriptors, Pattern Analysis and Machine Intelligence, 2005, Vol. 27, Number 10, pp. 1615-1630.

[4] Rover visual obstacle avoidance, Hans P. Moravec. IJCAI'81

[5] Scale-space filtering, Andrew P. Witkin. IJCAI'83

[6] A Representation for Shape Based on Peaks and Ridges in the Difference of Low-Pass Transform. James L. Crowley, Alice C. Parker. IEEE Transactions on Pattern Analysis and Machine Intelligence Volume 6 Issue 2, February 1984

[7] A combined corner and edge detector. C. Harris and M. Stephens. Proceedings of the 4th Alvey Vision Conference. pp. 147–151.

[8] A parallel implementation of a structure-from-motion algorithm. H. Wang, C. Bowman, M. Brady, C. Harris. ECCV '92

[9] Scale-Space Theory in Computer Vision. Tony Lindeberg. Kluwer Academic Publishers Norwell, MA, USA 1994. ISBN:0792394186.

[10] A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry. Z. Zhang, R. Deriche, O. Faugeras, and Q.-T. Luong. Artificial Intelligence Journal, 78:87–119, October 1995.

[11] Local Grayvalue Invariants for Image Retrieval. C. Schmid, R. Mohr. IEEE Transactions on Pattern Analysis and Machine Intelligence: Volume 19 Issue 5, May 1997

[12] Recognition Using Region Correspondences. R. Basri, D. Jacobs. International Journal of Computer Vision Volume 25 Issue 2, Nov. 1997

[13] A Cubist Approach to Object Recognition. R. Nelson, A. Selinger. ICCV '98

[14] Towards a computational model for object recognition in IT cortex. D. Lowe. BMVC '00

[15] Shock Graphs and Shape Matching. Ali Shokoufandeh et al. International Journal of Computer Vision archive Volume 35 Issue 1, Nov. 1999

[16] Recognition without Correspondence using Multidimensional Receptive Field Histograms. B. Schiele, J. Crowley. Journal International Journal of Computer Vision archive Volume 36 Issue 1, Jan. 2000, Pages 31 - 50

[17] Probabilistic Models of Appearance for 3-D Object Recognition. A. Pope, D. Lowe. Journal International Journal of Computer Vision Volume 40 Issue 2, Nov. 2000, Pages 149 - 167

[18] Phase-Based Local Features. Carneiro, Jepson. ECCV '02

[19] Face detection and tracking in a video by propagating detection probabilities. K. Mikolajczyk et al. IEEE Transactions on Pattern Analysis and Machine Intelligence archive, Volume 25 Issue 10, October 2003, Page 1215-1228.

[20] Distinctive Image Features from Scale-Invariant Keypoints. DAVID G. LOWE. University of British Columbia. Vancouver, Kanada.

[21] PCA-SIFT: a more distinctive representation for local image descriptors. Sukthankar et al. CVPR'04

[22] Machine learning for high-speed corner detection. Edward Rosten and Tom Drummond. Department of Engineering, Cambridge University, UK.

[23] SURF: Speeded Up Robust Features. Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Katholieke Universiteit Leuven. Zürich, Sveitsi.

[24] BRIEF: Binary Robust Independent Elementary Features. Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. CVLab, EPFL, Lausanne, Sveitsi.

[25] ORB: an efficient alternative to SIFT or SURF. Ethan Rublee, Vincent Rabaud, Kurt Konolige, Gary Bradski. Willow Garage, Menlo Park, California.

[26] A Comparison of Local Detectors and Descriptors for Multi-Object Applications. Faten A. Khalifa , Noura A. Semary Hatem M. El-Sayed, Mohiy M. Hadhoud. Faculty of Computers and Information, Menoufia University, Menoufia, Egypt.

[27] DFOB: Detecting and describing features by octagon filter bank

for fast image matching. Zhenyu Xu, Yiguang Liu n , Shuangli Du, Pengfei Wu, Jie Li. SiChuan University, College of Computer Science, Chendu 610065, China.

[28] Evaluation of Local Detectors and Descriptors for Fast Feature Matching. Ondrej Miksik, Krystian Mikolajczyk. ICPR ‘12

[29] Feature description with SIFT, SURF, BRIEF, BRISK, or FREAK? A general question answered for bone age assessment. M. Kashif, T. Deserno, D. Haak, S. Jonas. Journal Computers in Biology and Medicine. 2016.

[30] FREAK: fast retina keypoint. A. Alahi, R. Ortiz, P. Vandergheynst. IEEE Conference on Computer Vision and Pattern Recognition, 2012

[31] Robust wide-baseline stereo from maximally stable extremal regions. J. Matas, O. Chum, M. Urban, T. Pajdla. Image and vision computing, Vol. 22, Number 10, 2004

[32] A Comparative Evaluation of Well-known Feature Detectors and Descriptors. Şahin Işık, Kemal Özkan. International Journal of Applied Mathematics, Electronics and Computers. 2014

[33] Ultimate++, cross-platform rapid application development framework. http://ultimatepp.org/

[34] Oxford Dataset
http://www.robots.ox.ac.uk/~vgg/data/data-aff.html

[35] Tämän kandidaatintyön ohessa tehty ilmakiekkopeli. https://github.com/sppp/AR-AirHockey