Tampereen yliopisto

Otto Haarahiltunen

VISUAALISTEN 2D-TUNNISTEIDEN VERTAILU

Kandidaatintutkielma
Informaatioteknologian ja viestinnän tiedekunta
Tarkastajat: Prof. Joni Kämäräinen
Huhtikuu 2025

TIIVISTELMÄ

Otto Haarahiltunen: Visuaalisten 2D-tunnisteiden vertailu
Kandidaatintutkielma
Tampereen yliopisto
Tietotekniikka, Tieto- ja sähkötekniikan koulutus
Huhtikuu 2025

Kandidaatintutkielmassa vertaillaan OpenCV-kirjaston tarjoamia ratkaisuja kameran 3D-asennon arviointiin visuaalisten tunnisteiden avulla. Työhön valikoituivat tunnisteet ArUco ja AprilTag. Tunnisteita vertaillaan niiden antaman asentotiedon virheen mukaan. Työssä käydään läpi taustaa visuaalisista tunnisteista ja asennon arvioinnista niiden avulla. Tämän jälkeen käsitellään ArUco- ja AprilTag-tunnistusalgoritmien eroavaisuudet. Eroavaisuuksien jälkeen käsitellään teoria, miten tunnistetun tunnisteen avulla voidaan arvioida kameran 3D-asentoa.

Asennon arvioinnin jälkeen käydään läpi asetelma, testattavat arvot ja tulokset. Johtopäätöksenä todettiin, että AprilTag soveltuu 3D-asennon arviointiin hieman paremmin tarkemman asennon tunnistuksen ja pienemmän väärien positiivisten määrän takia.

Työn lähdekoodit: `https://github.com/ottohh/bachelor-s-thesis`.

Avainsanat: ArUco, AprilTag, Asennon arviointi, PnP, robotiikan paikannus

Tämän julkaisun alkuperäisyys on tarkastettu Turnitin OriginalityCheck -ohjelmalla.

ABSTRACT

Otto Haarahiltunen: Comparison of 2D fiducial markers
Bachelor's Degree
Tampere University
Bachelor's Degree Programme in Information Technology
April 2025

This bachelor's thesis compares the solutions provided by the OpenCV library for estimating camera's 3D pose using 2D fiducial markers. The study focuses on two marker types: ArUco and AprilTag. These marker types are compared based on the error in the estimated pose data they provide.

The thesis first presents background information on fiducial markers and pose estimation using them. Thesis explains the differences between the ArUco and AprilTag detection algorithms. After discussing these differences, the theoretical basis for estimating a camera's 3D pose using a detected marker is covered.

Following the theoretical discussion, the testing phase is described, including the test setup, evaluated values, and results. It was concluded that AprilTag is slightly better suited for 3D pose estimation due to its more accurate pose detection and lower rate of false positives.

The code used in the thesis can be found at: `https://github.com/ottohh/bachelor-s-thesis`.

Keywords: ArUco, AprilTag, Pose estimation, PnP, Localization in Robotics

The originality of this thesis has been checked using the Turnitin OriginalityCheck service.

TEKOÄLYN KÄYTTÖ OPINNÄYTTEESSÄ

Tekoälyn käyttö OPINNÄYTTEESSÄ

Opinnäytteessäni on käytetty tekoälysovelluksia: Kyllä

Ilmoitukseni mukaan olen käyttänyt opinnäytteessäni tutkielmaprosessin aikana seuraavia tekoälysovelluksia: ChatGPT 4o ja o1.

Käyttötarkoitus: Tekoälyä on käytetty tiedonhankintavaiheessa sopivien kirjastojen etsimiseen ja tutkimuskysymyksen parempaan muotoiluun. Tekoälyä on pyydetty selittämään esimerkein teoriaa ja tuottamaan LaTeX-komennot kaavoihin. Tutkimusosuuden aikana on koodia tuotettu ensisijaisesti tekoälyn avulla ja sen epäonnistuessa korjattu itse. Tekstin tuottamisvaiheessa tekoälyltä kysyttiin neuvoja ja parannusehdotuksia, sekä sitä käytettiin tekstin oikolukemiseen. Neuvot ja parannusehdotukset sisälsivät muun muassa sitä, miten asiat ilmaistaisiin kanditutkielmassa ja onko kappale ymmärrettävä. Tekoälyä kysyttiin myös arvioimaan tieteellistä tarkkuutta ja yksiselitteisyyttä. Tämä tehtiin jokaiselle kappaleelle erikseen aina kappaleen kirjoittamisen jälkeen ja lopussa koko tutkielmalle. Kaava (4.1) johdettiin itse teorian perusteella ja tekoälyä pyydettiin tekemään sama. Näin pyrittiin saamaan varmuus siitä, että johtaminen meni oikein. Tekoälyä käytettiin myös englanninkielisen tiivistelmän luontiin suomenkielisen tiivistelmän pohjalta muutamien korjauksien kera. Tämä kappale on myös lisätty LaTeX-pohjaan tekoälyn tuottamilla ohjeilla (tätä kappaletta ei ollut LaTeX-pohjassa valmiina).

Osiot, joissa tekoälyä on käytetty: Kaikissa osioissa.

Olen tietoinen siitä, että olen täysin vastuussa koko opinnäytteeni sisällöstä, mukaan lukien osat, joissa on hyödynnetty tekoälyä, ja hyväksyn vastuun mahdollisista eettisten ohjeiden rikkomuksista.

ALKUSANAT

TÄÄ ON JUST SULLE!!!

Hei mahtavaa et säki oot päätynyt lukemaa tätä. Olit sitten kaveri, työkaveri, pomo, perheenjäsen, muu semmalainen, yms. Sä oot tehny oikeen valinnan. Toivon, että tuut nauttimaan ees vähän enemmän ku mä nautin kirjottamisesta(ajoittain). Tsemppiä mikäli satut olemaan just se tyyppi, joka tulee jatkamaan tän työn pohjalta. Toivottavasti robotin ohjaaminen AprilTagin avulla onnistuu. Pahoittelen, että mun koodi on vähän spagettia, jos päädyt sitä käyttämään.

Kiitokset vielä Jonille ohjaamisesta ja kiinnostavasta aiheesta, sekä Jani Käpylälle mocapin käytöstä.

Nauttikaa!!!

Tampereella, 23. huhtikuuta 2025

Otto Haarahiltunen

SISÄLLYSLUETTELO

1. Johdanto 1
2. Taustakirjallisuus 3
2.1 2D-tunnisteet 3
2.2 Kameramatriisi 3
3. tunnistaminen 6
3.1 Kuvan binarisointi 6
3.2 Reunantunnistus 6
3.3 Nelikulmiotunnistus 7
3.4 Koodin lukeminen 8
4. kameran Asennon ja paikan arviointi 9
4.1 Perspektiivi-N-Piste (PnP) ongelma 9
4.2 Gauss–Newton epälineaarinen minimointi 10
5. Testaaminen 12
5.1 Testausasetelma 12
5.2 Testattavat arvot 13
5.3 Tulokset 15
6. Yhteenveto 18
Lähteet 19

LYHENTEET JA MERKINNÄT

DLT	Suora lineaarimuunnos
MSE	Keskineliövirhe
PCA	Pääkomponenttianalyysi
PnP	Perspektiivi-N-Piste

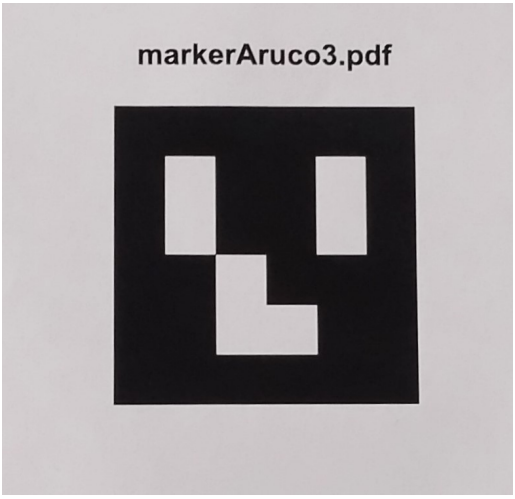
1. JOHDANTO

Kameran 3D-asennon ja sijainnin arviointi on yleinen ongelma monissa eri sovelluksissa, kuten lisätyssä todellisuudessa ja robotiikassa. Kuvadataan perustuvassa arvioinnissa ympäristöstä täytyy tunnistaa ankkuripisteitä. Osa menetelmistä perustuu luonnossa esiintyvien muotojen ja objektien havainnointiin ja toiset taas keinotekoisten visuaalisten 2D-tunnisteiden eli tagien tunnistamiseen. [1]

Visuaalinen 2D-tunniste on yksinkertainen kuvio, joka voidaan tulostaa ja asettaa ympäristöön. Tunnisteen visuaalinen suunnittelu on tarkoituksella yksinkertainen ja erottuva, jotta se olisi helppo tunnistaa kuvankäsittelyn ja koneoppimisen menetelmin.[2] Tunnisteet muistuttavat QR-koodeja, mutta niiden käyttötarkoitus on erilainen. Toisin kuin QR-koodit, jotka on kehitetty tiedon tallentamiseen, 2D-tunnisteet on kehitetty ensisijaisesti tarkkaa ja luotettavaa tunnistamista varten. [2]

Tunnisteita on olemassa useita eri tyyppejä, ja niiden tunnistamiseen on kehitetty monia erilaisia Algoritmeja. Ne on jaoteltu eri sovelluskohteiden mukaan.[3] Tässä työssä keskitytään vertailemaan OpenCV:n tarjoamia ratkaisuja robotiikan paikannustehtävissä. OpenCV on maailman suurin avoimen lähdekoodin konenäkökirjasto.[4] Se tarjoaa paikantamista varten kahta eri tunnisteperhettä AprilTag ja ArUco. Katso kuvat 1.1(a) ja 1.1(b).

(a) AprilTag tunniste



(b) ArUco tunniste

Kuva 1.1. *AprilTag ja ArUco tunnisteet*

2. TAUSTAKIRJALLISUUS

Visuaalisia tunnisteita on useita erilaisia, joissa tagin tunnistaminen ja ankkuripisteiden löytäminen kuvasta toimivat eri tavalla. Tästä huolimatta kaikissa tunnisteissa asennon ja paikan arviointi ankkuripisteiden avulla noudattaa samaa perusperiaatetta. Arviointi perustuu tunnettuihin maailman pisteisiin ja niitä vastaaviin kuvapisteisiin [5].

2.1 2D-tunnisteet

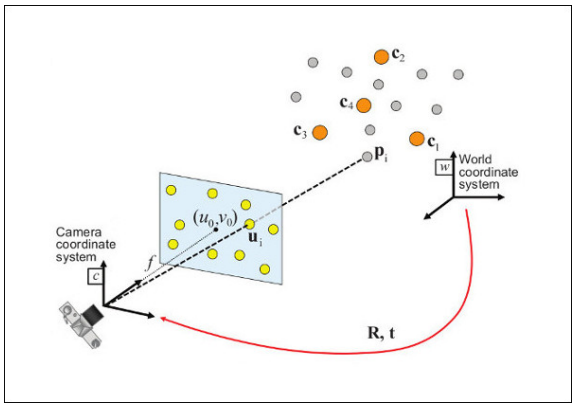
Taulukko 2.1. *Esimerkkejä tunnisteista [6]*

Nimi	**Merkki**
AprilTag	
ARTag	
ChromaTag	
ArUco	
CCC	

2D-tunnisteiden tarkoituksena on olla kuvasta helposti havaittavia ja tarjota ankkuripisteitä. Käyttötarkoituksissa, joissa niiden pitää tarjota vain ankkuripisteitä, ne ovat yleensä yksinkertaisia muotoja, kuten ympyröitä (CCC tunniste). Käyttötarkoituksissa, joissa niiden on tarkoitus tarjota ankkuripisteiden lisäksi jotain informaatiota, niihin on koodattu viesti QR-koodien tapaan. [6]

2.2 Kameramatriisi

Kameramatriisilla kuvataan ne parametrit, joiden avulla maailman 3D-piste voidaan projisoida 2D-kuvatasolle oheisen kuvan 2.1 mukaan. Muunnos maailman pisteistä kuvatasoon tapahtuu perspektiivi projektiolla. [5]



Kuva 2.1. *Pisteiden projisointi kuvatasoon [5]*

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = A\,\Pi^c\,T_w \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{2.1}$$

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{2.2}$$

missä x ja y ovat kuvatason koordinaatit (kuvassa (u_0, v_0)), A kameran sisäiset (intrinsic) parametrit, Π projektio-operaattori, ${}^cT_w$ maailman ja kameran välinen muunnosmatriisi eli ulkoiset (extrinsic) parametrit ja X_w, Y_w, Z_w maailman koordinaatit. [5]

Sisäiset parametrit liittyvät kameran rakenneominaisuuksiin, joten ne eivät muutu kameraa liikuttaessa. Parametrit (f_x, f_y) ilmaisevat kameran polttovälit, ja (c_x, c_y) määrittelevät kuvatasoon projisoitumisen optisen keskipisteen. Nämä parametrit on kalibroitava erikseen, mutta tässä työssä ei perehdytä kalibrointiprosessiin tarkemmin. [7] Ulkoiset parametrit r_{11},r_{12},r_{13},r_{21},r_{22},r_{23},r_{31},r_{32},r_{33} rotaatiot ja t_x,t_y,t_z translaatiot määrittelevät kameran sijainnin ja orientaation suhteessa maailman koordinaatistoon:

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = {}^{c}\mathbf{T_w} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

Tässä X_c, Y_c, Z_c kameran koordinaatit. Ulkoisten parametrien perusteella saadaan siis kameran asento ja sijainti suhteessa maailmaan. [5]

3. TUNNISTAMINEN

Työssä esiintyvät tunnisteet ArUco ja AprilTag ovat hyvin samankaltaisia, ja tunnistamiseen liittyvät vaiheet ovat pääpiirteittäin samat. Vaiheet eroavat kuitenkin toisistaan niissä käytettävin menetelmin. Menetelmät voidaan jakaa pääpiirteittäin seuraaviin askeliin:

- Kuvan binarisointi
- Reunantunnistus
- Nelikulmiotunnistus
- Koodin lukeminen

3.1 Kuvan binarisointi

ArUco-tunnisteiden binarisointi käyttää paikallista adaptiivista suodatusta. Tämä menetelmä toimii luotettavasti myös vaihtelevissa valaistusolosuhteissa. [1] AprilTag käyttää hieman mukautettua adaptiivista suodatusta. Se jakaa kuvan 4x4-alueisiin ja laskee kunkin alueen ääriarvot. Näistä ääriarvoista lasketaan kynnysarvo

$$\frac{\text{max} + \text{min}}{2} \tag{3.1}$$

Välttääksemme vääristymiä 4x4-alueen reunoilla, missä ääriarvot eivät välttämättä vastaa todellisuutta, aluetta laajennetaan 3x3-kokoisella ikkunalla kyseisen pikselin kohdalta. Näin varmistetaan, että vierekkäiset alueet menevät edes yhden pikselin verran päällekkäin. [8]

3.2 Reunantunnistus

Kuvan binarisoinnin jälkeen siitä pitää tunnistaa pikselit, jotka muodostavat tagin ulkoreunan.

ArUco hyödyntää Suzuki ja Aben ääriviivojen tunnistusalgoritmia [1] Algoritmi lukee binaarikuvaa järjestelmällisesti ylhäältä alas ja vasemmalta oikealle. Havaitessaan pikselin, jonka arvo on 1, algoritmi aloittaa reunan seuraamisen. Reunaa seurataan käymällä läpi pikselin naapurusto myötäpäivään aloittaen siitä suunnasta mistä tähän pikseliin ollaan

saavuttu. Löydettyään naapurustosta seuraavan jo läpi käymättömän pikselin, jonka arvo on 1 algoritmi siirtyy siihen. Tätä vaihetta toistetaan kunnes ollaan löydetty se pikseli, josta alunperin lähdettiin. Naapureiden läpikäynti myötäpäivään ja aloittamalla siitä suunnasta, josta tultiin takaa sen, että löydetty reuna on alueen ulkoreuna.[9]

AprilTagin reunantunnistuksessa etsitään ensin kaikki pikselit, joilla on vähintään yksi erivärinen naapuri. Näitä pikseleitä voidaan pitää reunapikseleinä. Näistä pikseleistä muodostetaan ryhmät sen perusteella mihin muihin reunapikseleihin ne ovat yhteydessä. Näin muodostuu erilliset ryhmät valkoisille ja mustille reunapikseleille. Lopussa tallennetaan kutakin reunaa vastaavat valkoiset pixelit ryhmiin sen perusteella minkä mustan ja valkoisen alueen rajoja ne ovat. Pikseli voi kuulua useamman alueen rajaan. Näin vältytään ongelmalta, jos kahta mustaa aluetta erottava reuna on vain yhden pikselin levyinen.[8]

3.3 Nelikulmiotunnistus

Reunojen tunnistamisen jälkeen on saatu lista reunapisteitä, jotka rajaavat alueen(reuna on yhden pikselin levyinen). Tässä vaiheessa halutaan selvittää, onko kyseinen alue nelikulmio, joka täyttäisi tagin vaatimukset.

ArUco käyttää tähän Douglas–Peucker-algoritmia. Algoritmi sovittaa suoran viimeisen ja ensimmäisen pisteen väliin. Tämän jälkeen lasketaan, jokaisen väliin jäävän pisteen etäisyys suorasta. Otetaan se piste, jonka etäisyys tästä suorasta on kaikkein pisin ja tarkastetaan onko se isompi kuin määritetty etäisyyskynnys. Mikäli etäisyyskynnys ylittyy, piste säilytetään, ja viiva pilkotaan kahteen osaan. Tätä toistetaan kunnes viivat yksinkertaistavat muodon riittävän hyvin.[10]

AprilTag-algoritmin nelikulmion sovitusvaihe alkaa järjestämällä reunapisteet niiden keskipisteen ympäri olevan kulman mukaan, mikä varmistaa johdonmukaisen kiertosuunnan ja helpottaa naapuripisteiden määrittelyä. Järjestämisen jälkeen algoritmi alkaa etsiä kulmapisteitä liikuttamalla liukuvaa ikkunaa pistejoukon yli. Jokaisessa ikkunan asennossa sovitetaan suora hyödyntämällä pääkomponenttianalyysiä (PCA), jossa pistejoukolle määritetään keskiarvo ja kovarianssi. Näiden avulla muodostetaan ellipsi, jonka ensimmäinen pääkomponentti kertoo suunnan, johon pistejoukon varianssi on suurin, tämä toimii parhaiten sovitettuna suorana. [8]

Kulmaehdokkaat tunnistetaan niistä kohdista, joissa suoran sovituksen virhe (MSE) saavuttaa huippuarvonsa. Suuret virhepiikit viittaavat kohtiin, joissa pistejoukon muoto muuttuu voimakkaasti, mikä yleensä vastaa nelikulmion kulmia. Kun kulmaehdokkaat on tunnistettu, seuraava vaihe on testata kaikki mahdolliset neljän kulmaehdokkaan yhdistelmät ja sovittaa suorat niiden välille. Tavoitteena on löytää se nelikulmio, jossa suorien sovitusvirhe on pienin. [8]

Nelikulmioiden tunnistamisen jälkeen molemmat algoritmit tarkistavat, että löydettyjen ne-

likulmioiden kulmat ovat lähellä 90 astetta ja täyttävät muut laatuvaatimukset. [1, 8]

3.4 Koodin lukeminen

Nelikulmio ehdokkaiden löytämisen jälkeen pitää niistä vielä lukea koodi ja suorittaa virheenkorjaus.

ArUco aloittaa poistamalla perspektiiviprojektion homografian avulla. Saatu kuva binarisoidaan Otsun [11] menetelmällä, joka valitsee optimaalisen kynnysarvon, kun kuvan intensiteettijakauma on bimodaalinen. Tämän jälkeen kuva jaetaan säännölliseen ruudukkoon ja jokainen ruudukko saa arvon 1 tai 0 sen pikselien enemmistön perusteella. Tässä kohtaa tarkastetaan mustan reunuksen löytyminen tagin ympäriltä ja reunan löytymisen jälkeen edetään virheenkorjaukseen ja koodin lukemiseen sisemmältä ruudukolta. [1]

AprilTag käyttää myös homografiaa hyödyksi laskemalla kunkin bittikentän koordinaatit suhteessa tägiin, muuntamalla ne kuvan koordinaateiksi ja binarisoimalla tuloksena saadut pikselit. Binarisoinnissa käytetään tilallisesti muuttuvaa intensiteettimallia:

$$I(x, y) = Ax + Bxy + Cy + D \tag{3.2}$$

Mallin neljä parametria ratkaistaan reunoilta saatavien tunnettujen pikselien avulla. Mustille ja valkoisille pikseleille luodaan omat mallinsa ja kynnysarvona käytetään näiden mallien keskiarvoa. Koodin purkamisen jälkeen edetään virheenkorjaukseen. [8]

4. KAMERAN ASENNON JA PAIKAN ARVIOINTI

Asennon ja paikan arvioinnissa halutaan löytää kierto- ja siirtymävektorit, jotka minimoivat uudelleenprojisointivirheen niitä vastaaviin kuvatason pisteisiin.[5]

4.1 Perspektiivi-N-Piste (PnP) ongelma

PnP-ongelma tarkoittaa kameran ulkoisten parametrien arviointia. Ulkoisista parametreista voidaan muodostaa kierto- ja siirtymävektorit. Ulkoisia parametreja arvioidaan pistevastaavuuksien avulla. Pistevastaavuuksia tarvitaan 6, koska matriisilla on 11 vapausastetta. Tähän erikoistapauksena, kun kaikki pisteet sijaitsevat samassa tasossa. Tällöin voidaan olettaa pisteiden olevan tasossa $Z = 0$. [12]

Pisteiden ollessa samassa tasossa projektio voidaan esittää muodossa

$$\mathbf{x}_0 = \Pi\, {}^{c}T_w\, {}^{w}\mathbf{X} = \Pi \begin{bmatrix} c_1 & c_2 & c_3 & {}^{0}t_w \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ 0 \\ 1 \end{bmatrix} \tag{4.1}$$

jossa c_i viittaa rotaatiomatriisin i:teen sarakkeeseen ja ${}^{0}t_w$ on siirtymävektori. Tämä voidaan edelleen muotoilla seuraavasti

$$\mathbf{x}_0 = \Pi \begin{bmatrix} c_1 & c_2 & {}^{0}t_w \end{bmatrix} \begin{bmatrix} {}^{w}X, {}^{w}Y, 1 \end{bmatrix}^{\top} = {}^{0}\mathbf{H}_w \begin{bmatrix} {}^{w}X, {}^{w}Y, 1 \end{bmatrix}^{\top} \tag{4.2}$$

Tässä ${}^{0}\mathbf{H}_w$ on homografiamatriisi, joka kuvaa objektin tasoon $Z = 0$. Matriisissa on yhdeksän tuntematonta suuretta, mutta vain kahdeksan vapausastetta (yhdeksästä vähennetään skaalaus). Pistevastaavuuksia tarvitaan siis 6 sijaan 4. Homografiamatriisi voidaan ratkaista suoran lineaarimuunnoksen (DLT) avulla

$$Ah = 0 \tag{4.3}$$

$$h = \begin{bmatrix} r_{11} \\ r_{21} \\ r_{31} \\ t_x \\ r_{12} \\ r_{22} \\ r_{32} \\ t_y \\ t_z \end{bmatrix} \tag{4.4}$$

$$\text{A} = \begin{bmatrix} \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ X_{wi} & Y_{wi} & 1 & 0 & 0 & 0 & -x_i\,X_{wi} & -x_i\,Y_{wi} & -x_i \\ 0 & 0 & 0 & X_{wi} & Y_{wi} & 1 & -y_i\,X_{wi} & -y_i\,Y_{wi} & -y_i \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}. \tag{4.5}$$

Kun homografia ${}^0\mathbf{H}_w$ on ratkaistu, voidaan määrittää kameran asento ${}^cT_w$. Tämä onnistuu, koska tiedetään yhteys $\left(\mathbf{c}_1, \mathbf{c}_2, {}^0\,t_w\right) = \Pi^{-1}\,{}^0\mathbf{H}_w$. Lisäksi voidaan huomata, että rotaatiomatriisin täytyy olla ortogonaalinen. Tämä mahdollistaa kolmannen sarakkeen laskemisen $\mathbf{c}_3 = \mathbf{c}_1 \times \mathbf{c}_2$. Näin saadaan ratkaistua kameran ulkoisten parametrien matriisi ${}^cT_w$. DLT-menetelmällä saatu ratkaisu ei ole tarkka, koska parametreihin liittyy epälineaarisuuksia. Tulosta parantaaksemme voimme käyttää jotakin epälineaarista minimointimenetelmää [12]

4.2 Gauss–Newton epälineaarinen minimointi

Gauss–Newton-menetelmä on epälineaarinen ja perustuu iteratiiviseen projektiovirheen minimointiin, jolloin ratkaisu konvergoituu paikalliseen minimiin. Paikallisen minimin takia vaaditaan hyvä alkuarvaus, joka saadaan esimerkiksi DLT-menetelmällä. Gauss–Newton-menetelmä antaa parhaan arvion, kun mittaustuloksiin oletetaan normaalijakautunutta virhettä. Ongelma voidaan muotoilla seuraavasti[12]

$$\hat{\mathbf{q}} = \arg\min_{\mathbf{q}} \sum_{i=1}^{N} d\left(\mathbf{x}_i, \Pi\, {}^{c}T_w\, {}^{w}\mathbf{x}_i\right)^2 \tag{4.6}$$

Tässä $\mathbf{q}$ on 6-ulotteinen (rotaatio + translaatio) minimiesitys matriisille ${}^{c}T_w$ ja funktio $d(x, x')$ kuvaa euklidista etäisyyttä pisteiden välillä. Gauss-Newton menetelmässä lähdetään etsimään optimaalista ratkaisua edellä esitettyyn kaavaan. Optimaalinen ratkaisu saadaan minimoimalla virhefunktio $E(\mathbf{q}) = \|\mathbf{e}(\mathbf{q})\|$, joka on määritelty [12]

$$E(\mathbf{q}) = \mathbf{e}(\mathbf{q})^\top \mathbf{e}(\mathbf{q}), \quad \text{missä} \quad \mathbf{e}(\mathbf{q}) = \mathbf{x}(\mathbf{q}) - \mathbf{x}, \tag{4.7}$$

jossa $\mathbf{x}(\mathbf{q}) = \left(\ldots, p\left({}^{c}T_w\, {}^{w}\mathbf{X}_i\right), \ldots\right)^\top$ ja $\mathbf{x} = \left(\ldots, \tilde{\mathbf{x}}_i, \ldots\right)^\top$ tässä $p(X)$ on perspektiiviprojektiofunktio 3D-pisteistä 2D-pisteisiin ja $\tilde{\mathbf{x}}_i = \left(x_i, y_i\right)$ 2D-piste. Ratkaisu perustuu virhefunktion $\mathbf{e}(\mathbf{q})$ linearisointiin q-arvon ympärillä ja pienen päivityksen $\delta\mathbf{q}$ ratkaisemiseen pienimmän neliösumman menetelmällä. Virheen ensimmäisen asteen Taylorin sarjakehitelmä on [12]

$$\mathbf{e}(\mathbf{q} + \delta\mathbf{q}) \approx \mathbf{e}(\mathbf{q}) + \mathbf{J}(\mathbf{q})\delta\mathbf{q} \tag{4.8}$$

,jossa $\mathbf{J}(\mathbf{q})$ on $2N \times 6$ kokoinen jacobin matriisi $\mathbf{q}$:n suhteen. Gauss-Newton menetelmällä minimisoidaan [12]

$$E(\mathbf{q} + \delta\mathbf{q}) = \|\mathbf{e}(\mathbf{q} + \delta\mathbf{q})\| \approx \|\mathbf{e}(\mathbf{q}) + \mathbf{J}(\mathbf{q})\delta\mathbf{q}\| \tag{4.9}$$

Tämä minimointiongelma voidaan ratkaista iteratiivisen pienimmän neliösumman(ILS) avulla [12]

$$\delta\mathbf{q} = -\mathbf{J}(\mathbf{q})^{+}\mathbf{e}(\mathbf{q}) \tag{4.10}$$

jossa $\mathbf{J}(\mathbf{q})^{+}$ on Moore–Penrose-pseudoinverssi jabobin matriisille $\mathbf{J}$.

Gauss-newton on iteratiivinen menetelmä, joten asentoa korjataan, jokaisella iteraatiolla [12]

$$\mathbf{q}_{k+1} = \mathbf{q}_k \oplus \delta\mathbf{q} = e^{\delta\mathbf{q}}\mathbf{q} \tag{4.11}$$

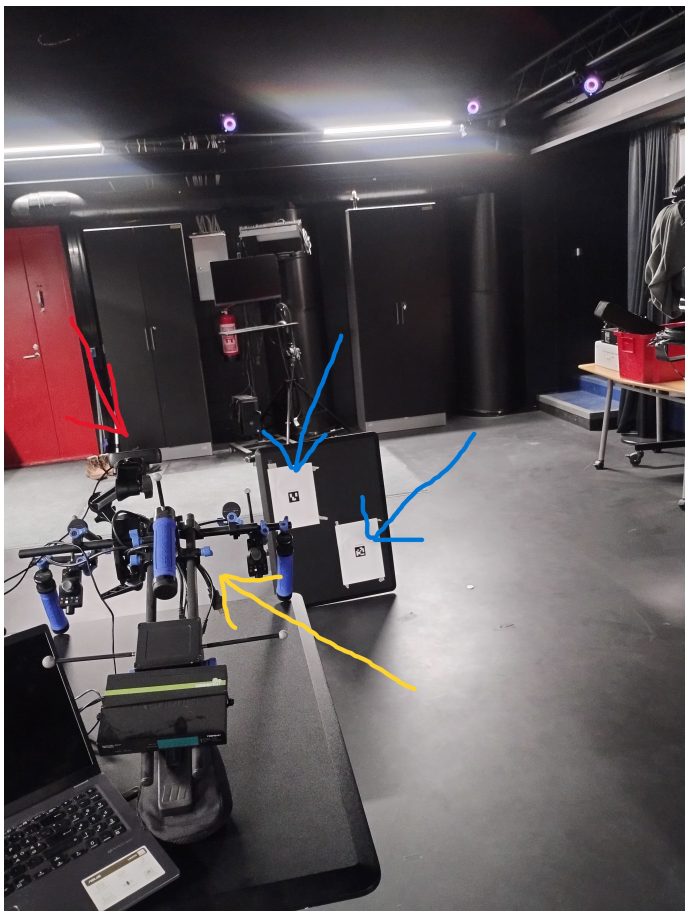
jossa $\oplus$ tarkoittaa SE(3)-ryhmän (ryhmä, joka koostuu 3D-avaruuden pyörähdys- ja siirtymämuunnoksista) elementin päivitystä eksponenttikuvauksen kautta. Vektorimuotoinen päivitysaskel $\delta\mathbf{q}$ muunnetaan SE(3)-ryhmän elementiksi eksponenttifunktion avulla, minkä jälkeen se kerrotaan nykyisellä asento-matriisilla $\mathbf{q}_k$, jolloin saadaan päivitetty asento. Iteraatiota jatketaan, kunnes ratkaisu konvergoituu paikalliseen minimiin tai pysähtymisehto täyttyy. [12]

5. TESTAAMINEN

5.1 Testausasetelma

Testausta varten tarvitaan tarkkaa aineistoa kameran todellisesta sijainnista suhteessa visuaaliseen tunnisteeseen. Tätä aineistoa saatiin Civit Motion Capture -studiosta [1].

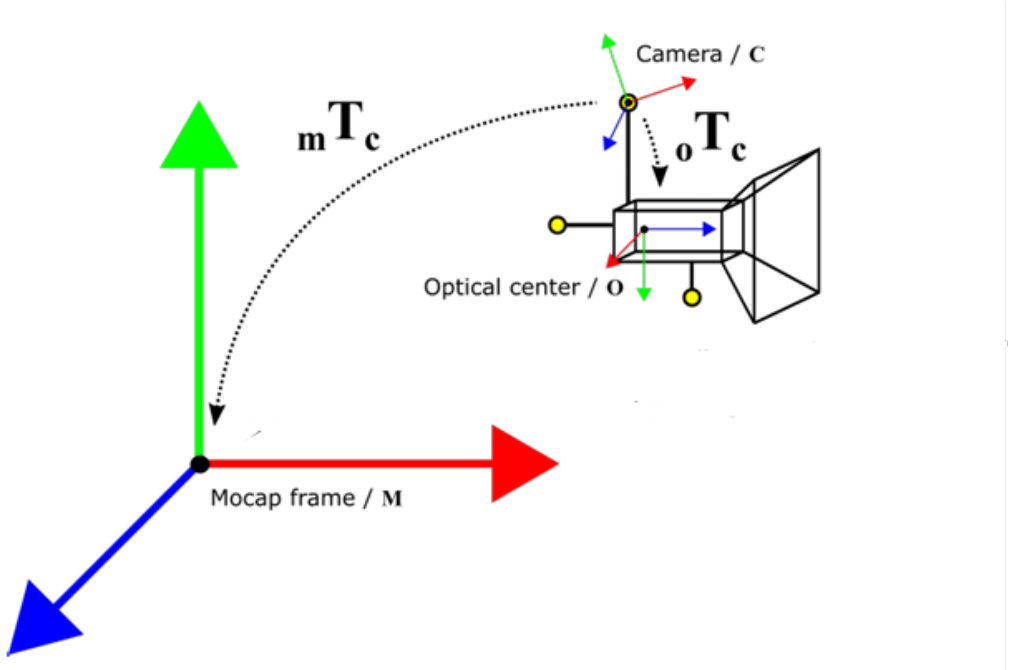
Motion capture -studiossa käytetty kamera oli Sony WebCam, jolla nauhoitettiin 640x480-videota. Vertailua varten kuvattiin 3 minuutin video, jossa kameraa liikuteltiin ympäriinsä pitäen sitä välillä hetken paikoillaan. Asetelma oli kuten kuvassa 5.1: April- ja ArUco-tunnisteet oli kiinnitetty pöytään (siniset nuolet), ja webkamera (punainen nuoli) oli kiinnitetty kameran pidikkeeseen (keltainen nuoli).



Kuva 5.1. *Testausasetelma*

[1] `https://civit.fi/`, viitattu 21.3.2025

Motion capture -studiosta saatu sijainti aineisto muunnettiin vastaamaan OpenCV:stä saatua aineistoa oheisen kuvan mukaisesti 5.2.



Kuva 5.2. *Maailman ja kameran koordinaatistojen suhde [13]*

Kuvassa 5.2 ${}_m\mathbf{T}_c$ on muunnosmatriisi, joka siirtää koordinaatteja pivot-koordinaatistosta maailman koordinaatistoon ja ${}_o\mathbf{T}_c$ kompensaatiomatriisi, joka siirtää pivot koordinaatistosta kameran koordinaatistoon. Matriisi ${}_m\mathbf{T}_c$ lasketaan kameran pidikkeen kääntö- ja siirtovektoreiden avulla, jotka saadaan suoraan Mocapista. Se muuttuu, kun kameraa liikutetaan. Matriisi ${}_o\mathbf{T}_c$ lasketaan kalibroinnin yhteydessä Käpylän tutkielmassa [13] esittämällä tavalla. Se ei muutu kalibroinnin jälkeen, kunhan kameraa ei irrota pidikkeestä.

5.2 Testattavat arvot

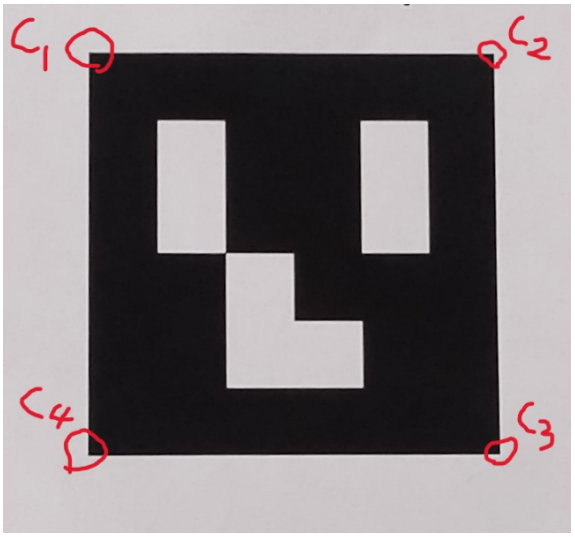
Tässä työssä vertaillaan tunnisteita robotiikan paikannustehtävien näkökulmasta. Virhe määritetään laskemalla euklidinen etäisyys koordinaattien välillä, jotka on muunnettu kameran koordinaatistoon ulkoisten parametrien sekä Mocap-järjestelmän muunnos- ja kompensaatiomatriisin avulla.

$$\mathbf{E} = d\left({}^o\mathbf{T}_c\,({}^m\mathbf{T}_c)^{-1}\mathbf{K}, {}^cT_w\mathbf{K}\right) \tag{5.1}$$

$$\mathbf{K} = \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{5.2}$$

Kaavassa 5.1 d kuvaa euclidista etäisyysfunktiota ja K visuaalisen tunnisteen keskipistettä maailman koordinaateissa. Tätä virhettä verrataan kuvassa esiintyvän tunnisteen lyhyimmän sivun pituuteen pikseleinä.

$$\textbf{Smin} = \min\left(d(\mathbf{C}_1, \mathbf{C}_2), d(\mathbf{C}_2, \mathbf{C}_3), d(\mathbf{C}_3, \mathbf{C}_4), d(\mathbf{C}_4, \mathbf{C}_1)\right) \tag{5.3}$$



Kuva 5.3. *Kuvatasosta löydetyt kulmat*

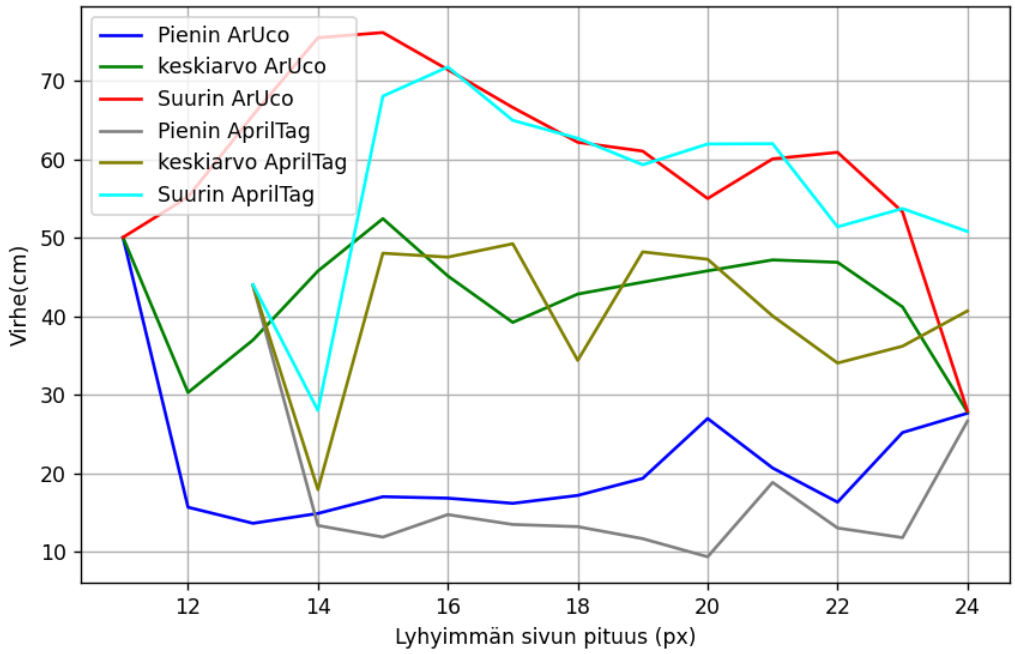
Kaavassa 5.3 $\mathbf{C}_1, \mathbf{C}_2, \mathbf{C}_3, \mathbf{C}_4$ tarkoittavat tunnisteen kuvatasosta 5.3 löydettyjä kulmia $\mathbf{C}_i = (x_i, y_i)$.

Tämän jälkeen ryhmitellään sivun pituudet. Esimerkiksi ryhmä 16 sisältää kaikki pienimmän sivun pituudet väliltä 15.5-16.5. Tämän jälkeen vertaillaan virheen minimi-, maximi- ja keskiarvoa jokaisella eri ryhmällä.

Tämä mittaustapa valittiin sen yksinkertaisuuden ja helpon toteutettavuuden vuoksi. Tarvitaan vain video, jossa molemmat markkerit ovat vierekkäin. Jokaisesta ruudusta arvioidaan molempien tunnisteiden etäisyyden virhe, sekä tunnisteiden lyhyimmän sivun pituus.

Lyhyimmän sivun pituus valittiin virheen vertailuarvoksi, koska se ottaa huomioon sekä tunnisteen etäisyyden, että rotaation. Lyhyimmän sivun pituus pienenee, kun kamera siirretään kauemmas. Se myös pienenee, jos tunnisteen kulma suhteessa kuvatasoon muuttuu jyrkemmäksi.

5.3 Tulokset

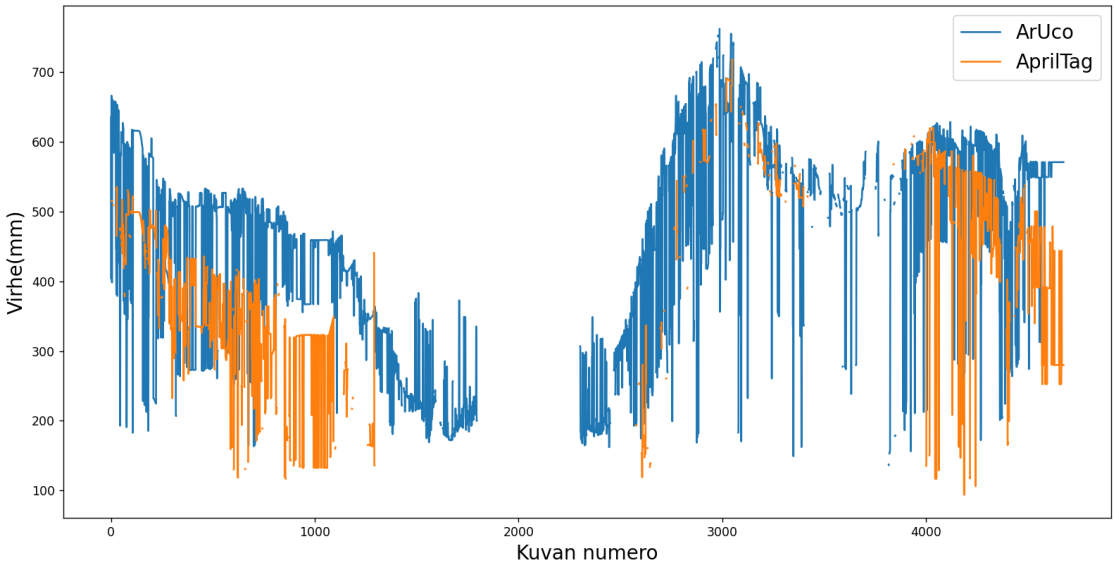


Kuva 5.4. *Kuvaaja ArUco ja AprilTag paikannustarkkuuksista*

Vertailtaessa ArUco- ja AprilTag-merkkien paikannustarkkuutta lyhyimmän sivun pituuteen 5.4, tuloksista ei selviä yksiselitteisesti, kumpi merkintätapa on parempi. Osassa pituuksista havaintojen määrä on pieni (vain muutamia mittauksia), mikä saattaa korostaa satunnaisvaihtelua ja vääristää kuvaajaa.

Perhe	minimi (cm)	maximi (cm)	keskiarvo (cm)	hajonta (cm)	kuvia yhteensä
ArUco	13.65	76.19	44.25	14.75	3639
AprilTag	9.38	71.80	40.36	13.33	1739

Taulukko 5.1. *Virhetilastoja ArUco- ja AprilTag-tunnisteille.*



Kuva 5.5. *Kuvaaja ArUco ja AprilTag paikannustarkkuuksista videon aikana*

Kaikille mittausarvoille lasketuista virhetilastoista 5.1 ja paikannustarkkuuksista videon aikana 5.5 voidaan huomata, että AprilTag suoriutui hieman paremmin kuin ArUco. Niitä tunnistettiin kuitenkin huomattavasti vähemmän. Molempien tunnisteiden virheissä oli suurta hajontaa, todennäköisesti kameran liikkeen takia. Tämä korostaa vakauden merkitystä. Kuvaajasta 5.5 on huomioitavaa, että välillä noin 1500–2000 kameran ja tunnisteiden normaalien välinen kulma on suuri, ja AprilTag on kauempana kuin ArUco 5.6. Välillä noin 3500–3800 AprilTag ei ole kokonaan näkyvissä.

Kuva 5.6. *Suuri kulma kameran ja tunnisteiden normaalien välillä. AprilTag on taka-alalla. Kuvan numero 1800.*

Vääriä positiivisia havaintoja oli erittäin vähän: ArUco-tunnisteilla 2 ja AprilTag-tunnisteilla ei lainkaan. Vääriksi positiivisiksi tunnistetut kohdat olivat hyvin pieniä. Niistä päästiin helposti eroon suurentamalla minMarkerPerimeterRate-parametria, jolloin liian pieniä markkereita ei yritetty tunnistaa.

6. YHTEENVETO

Tutkielmassa vertailtiin OpenCV:n tarjoamia toteutuksia visuaalisille tunnisteille. Haluttiin selvittää niiden toimivuutta robotiikan paikannustehtävissä. Tämä tehtiin mittaamalla tunnisteiden paikannusvirhettä. Molemmilla tunnisteilla oli virheessä suhteellisen paljon hajontaa, mikä todennäköisesti selittyy kameran liikkeellä. Tunnisteiden koko videolla oli myös pieni (suurimmillaan lyhimmän sivun pituus oli 24 pikseliä), mikä osaltaan selittää paikannustarkkuuden heikkoutta. Suurentamalla tunnisteen kokoa tai lisäämällä videon resoluutiota saataisiin todennäköisesti parannettua paikannustarkkuutta, koska muutaman pikselin virhe kulmien paikoissa ei tällöin olisi suhteellisesti niin merkittävä.

Arvioitaessa koko videon perusteella laskettuja arvoja voidaan todeta, että AprilTag suoriutui paikannuksessa hieman paremmin kuin ArUco. Lisäksi se ei havainnut yhtään väärää positiivista. Nämä havainnot ovat linjassa aikaisempien tutkimusten kanssa [6].

LÄHTEET

[1] Garrido-Jurado, S., Muñoz-Salinas, R., Madrid-Cuevas, F. ja Marín-Jiménez, M. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition* 47.6 (2014), s. 2280–2292. ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2014.01.005. URL: https://www.sciencedirect.com/science/article/pii/S0031320314000235.

[2] Olson, E. AprilTag: A robust and flexible visual fiducial system. *2011 IEEE International Conference on Robotics and Automation*. 2011, s. 3400–3407. DOI: 10.1109/ICRA.2011.5979561.

[3] Patru, G.-C., Pırvan, A.-I., Rosner, D. ja Rughinis, R.-V. Fiducial marker systems overview and empirical analysis of ArUco, AprilTag and CCTag (2023).

[4] OpenCV Documentation. *OpenCV - Open Computer Vision Library*. Accessed: 2025-03-05. n.d. URL: https://opencv.org/.

[5] OpenCV Documentation. *SolvePnP - Perspective-n-Point Problem*. Accessed: 2025-01-24. n.d. URL: https://docs.opencv.org/4.x/d5/d1f/calib3d_solvePnP.html.

[6] Kalaitzakis, M., Cain, B., Carroll, S., Ambrosi, A., Whitehead, C. ja Vitzilaios, N. Fiducial markers for pose estimation: Overview, applications and experimental comparison of the artag, apriltag, aruco and stag markers. *Journal of Intelligent & Robotic Systems* 101 (2021), s. 1–26.

[7] OpenCV Documentation. *Camera Calibration with OpenCV*. Accessed: 2025-02-09. n.d. URL: https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html.

[8] Wang, J. ja Olson, E. AprilTag 2: Efficient and robust fiducial detection. *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2016, s. 4193–4198.

[9] Suzuki, S. et al. Topological structural analysis of digitized binary images by border following. *Computer vision, graphics, and image processing* 30.1 (1985), s. 32–46.

[10] Douglas, D. H. ja Peucker, T. K. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: the international journal for geographic information and geovisualization* 10.2 (1973), s. 112–122.

[11] Otsu, N. et al. A threshold selection method from gray-level histograms. *Automatica* 11.285-296 (1975), s. 23–27.

[12] Marchand, E., Uchiyama, H. ja Spindler, F. Pose estimation for augmented reality: a hands-on survey. *IEEE transactions on visualization and computer graphics* 22.12 (2015), s. 2633–2651.

[13] Käpylä, J. *Development of a Toolset for Calibrating Optical Motion Capture with External RGBD Cameras*. 2023. URL: https://urn.fi/URN:NBN:fi:tuni-2023121610919.