



OULUN YLIOPISTO  
UNIVERSITY of OULU

TIETOTEKNIIKAN OSASTO

**Marko Seppälä**

**YMPÄRISTÖN 3D-MALLINTAMINEN  
QUADROKOPTERILLA KUVATUN 2D-VIDEON  
POHJALTA**

Diplomityö  
Tietotekniikan koulutusohjelma  
Tammikuu 2014

Seppälä M. (2013) Ympäristön 3D-mallintaminen quadrokopterilla kuvatun 2D-videon pohjalta. Oulun yliopisto, Tietotekniikan osasto. Diplomityö, 73 s.

## TIIVISTELMÄ

Ympäristöistä muodostetuille 3D-malleille löytyy lisääntyvää kysyntää useilta eri teollisuuden aloilta, esimerkiksi robotiikan, arkkitehtuurin, arkeologian, filmitoiminnan ja virtuaaliturismin parista. Lisääntyneen kysynnän vuoksi 3D-malleja halutaan luoda aikaisempaa nopeammin, edullisemmin ja automaattisemmin. Perinteisten 3D-mittaukseen perustuvien järjestelmien avulla suurten ympäristöjen 3D-mallintaminen ei ole käytännössä järkevää, sillä 3D-informaation mittaaminen on liian hidasta, vaikeaa ja kallista. Tämän vuoksi kehitetään järjestelmiä, joiden avulla 3D-informaatio voidaan laskea suoraan 2D-kuvien pohjalta käyttämällä apuna useita samasta näkymästä otettuja kuvia. Tällöin puhutaan niin sanotusta monen näkymän geometriasta. Tässä diplomityössä tehdään alustava toteutus järjestelmästä, jonka avulla ympäristö voidaan 3D-mallintaa videokuvan pohjalta hyödyntämällä monen näkymän geometriaan perustuvia menetelmiä. Kuvausrobotina käytetään Parrot-yhtiön AR Drone 2.0 quadrokopteria. Ohjelmiston toteuttamisessa hyödynnetään avoimeen lähdekoodiin perustuvia VisualSFM- ja OpenCV-ohjelmistoja. Puuttuvat ohjelmiston osat on toteutettu Python-ohjelmointikielen avulla. Työssä on selvitetty, että mitä ongelmia tällaisen järjestelmän toteuttamisessa kohdataan ja mitä menetelmiä näiden ongelmien ratkaisemiseen on olemassa.

**Avainsanat:** monen näkymän geometria, kuviin pohjautuva mallintaminen, monen näkymän stereo, mallintaminen liikkeen avulla

**Seppälä M. (2013) Environment 3D-modeling from 2D-video captured by a quadricopter.** University of Oulu, Department of Computer Science and Engineering. Master's thesis, 73 p.

## **ABSTRACT**

**Increasing demand for environment based 3D models can be found in various different industries such as robotics, architecture, archeology, the film industry, and virtual tourism. Due to the increasing demand, 3D models now require faster, cheaper and more automatized creation than before. 3D modelling of large environments with the use of traditional 3D measuring systems is not reasonable in practice because measuring of 3D information is too slow, difficult, and expensive. Because of this, systems are being developed that allow 3D information to be calculated directly from 2D images by utilizing several images taken of the same view. This is so-called multiple view geometry. In this Master's thesis a preliminary implementation of a system, by the aid of which a 3D model of an environment can be created from the basis of video capture, was completed. The system utilizes methods based on multiple view geometry. For the video capturing robot, a Parrot AR Drone 2.0 quadricopter was used. In the software implementation, open-source based VisualSFM and OpenCV software were used. Remaining parts of the software were implemented using Python software language. This thesis determines what problems are encountered when implementing this type of system, and what methods exist for solving those problems.**

**Key words: multiple view geometry, image-based modelling, multi-view stereo, structure from motion**

# SISÄLLYSLUETTELO

TIIVISTELMÄ

ABSTRACT

SISÄLLYSLUETTELO

ALKULAUSE

LYHENTEIDEN JA MERKKIEN SELITYKSET

1	JOHDANTO .....	8
1.1	Tutkittava ongelma .....	8
1.2	Ratkaisun yleiskuvaus .....	9
2	TEOREETTISTA TAUSTAA .....	10
2.1	Kolmiulotteinen euklidinen geometria ja jäykän kappaleen liike .....	10
2.2	Projektiivinen geometria .....	12
2.3	Kameramalli ja perspektiivinen projektio .....	14
2.4	Epipolaarinen geometria ja 3D-informaation palauttaminen .....	17
3	YMPÄRISTÖN 3D-MALLINTAMINEN .....	21
3.1	Geometrinen mallintaminen robotin avulla .....	21
3.2	Samanaikainen paikannus ja kartoitus .....	22
3.3	Mallintaminen liikkeen avulla .....	24
3.4	Monen näkymän stereo .....	26
3.5	Yleiskatsaus olemassa oleviin järjestelmiin .....	27
4	JÄRJESTELMÄN KUVAUS JA TOTEUTTAMINEN .....	30
4.1	Järjestelmän kuvaus .....	30
4.1.1	Käyttötarkoitus .....	31
4.1.2	Järjestelmän suorituskyvyille asetetut vaatimukset .....	31
4.1.3	AR Drone 2.0 .....	32
4.2	Järjestelmän toteuttaminen .....	34
4.2.1	Kameran kalibrointi .....	34
4.2.2	Datan kerääminen .....	37
4.2.3	Datan esikäsittely .....	38
4.2.4	Pistepilven muodostaminen .....	40
4.2.5	Mallin luominen .....	42
5	JÄRJESTELMÄN TESTAUS .....	45
5.1	Kuvien määrän vaikutus .....	45
5.2	Kuvien resoluution vaikutus .....	48
5.3	Järjestelmän parametrien asettaminen .....	50

6	ONGELMAKOHDAT JA JATKOKEHITYS .....	54
6.1	Datan keräämiseen liittyvät haasteet .....	54
6.2	Kuvien laadun parantaminen .....	56
6.3	SFM-menetelmän kehittäminen.....	59
6.4	Lopullisen mallin parantaminen .....	61
6.5	Järjestelmän vaiheiden integrointi .....	61
7	POHDINTA.....	62
8	YHTEENVETO .....	64
9	LÄHDELUETTELO .....	65
10	LIITTEET.....	69

## ALKULAUSE

Tämä diplomityö on tehty älykkäiden järjestelmien tutkimusryhmässä Oulun yliopiston tietotekniikan osastolla. Haluan kiittää Professori Juha Röningiä mahdollisuudesta tehdä tämä diplomityö tällä osastolla sekä hänen antamasta palautteesta työn viimeisteleminen vaiheessa. Haluan kiittää myös työn ohjaajaa Antti Tikanmäkeä, jonka kanssa olen käynyt useita mielenkiintoisia keskusteluja työn tekemisen aikana. Useimmat keskustelut olemme käyneet hieman aiheen vierestä, mutta ne ovat olleet joka tapauksessa kehittäviä ja tärkeitä keskusteluja.

Lisäksi haluan kiittää Markus Ylimäkeä, joka on antanut ajatuksia työn tekemiseen ja antanut luvan käyttää joitain hänen kuviaan tässä työssä, sekä Jukka Holappa, joka on avustanut minua kalibroituvaiheen suorittamisessa Oulun yliopistossa kehitetyn kalibroitijärjestelmän avulla.

Työ on kirjoitettu pääasiassa vaihto-opiskelun aikana Guildfordissa, Englannissa syksyllä 2013. Kiitos kämpäkavereille, jotka antoivat minulle työrauhan, eivätkä tulleet hakemaan minua huoneestani, vaikka minua ei muutamaan päivään näkynytäkään huoneen ulkopuolella. Suurin kiitos kuuluu niille ystäville, jotka sanoivat minulle jo opintojeni alkuvaiheessa, että minä en tule ikinä saamaan opintojani valmiiksi. En voinut suoda heille sitä tyydytystä, että he olisivat olleet oikeassa.

Oulussa 14.1.2014

Marko Seppälä

## LYHENTEIDEN JA MERKKIEN SELITYKSET

2D	Kaksiulotteinen
3D	Kolmiulotteinen
DOF	Vapausaste (Degrees of freedom)
GPS	Global Positioning System
OpenCV	Open Source Computer Vision Library
HD	Korkealaatuinen videostandardi (High Definition)
INS	Inertial Navigation System
MP4	Häviöllinen datan pakkausformaatti
MVS	Multi-view stereo
PLY	Tiedostoformaatti (Polygon File Format)
RGB	RGB-värimalli
SFM	Structure from motion
SIFT	Scale Invariant Feature Transform
SLAM	Simultaneous Localization and Mapping
$\times$	Ristitulo
$\in$	Kuuluu joukkoon
$\forall$	Kaikilla
$\rightarrow$	Kuvaus
$\leftrightarrow$	Vastaavuus
$A$	Kameran kalibrointimatriisi
$A^{-1}$	Matriisin $A$ käänteismatriisi
$A^T$	Matriisin $A$ transpoosi
$A^{-T}$	$(A^{-1})^T = (A^T)^{-1}$
$C$	Kameran koordinaatisto
$D$	Kameran vääristymäparametrit, $8 \times 1$ -vektori
$E$	Essentiaalinen matriisi
$\mathbb{E}^n$	$n$ -ulotteinen euklidinen avaruus
$f$	Kameran polttoväli, kameravakio
$F$	Fundamentaalin matriisi
$I$	Yksikkömatriisi
$P$	Projektiomatriisi
$P_i$	Ideaalin projektiomatriisi
$\mathbb{P}^n$	$n$ -ulotteinen projekttiivinen avaruus
$R$	Rotaatiomatriisi
$\mathbb{R}^n$	$n$ -ulotteinen karteesinen avaruus
$t$	Translaatiovektori
$x, y, z$	Karteesisen koordinaatiston koordinaattiakselit
$x$	Koordinaatti tasolla
$x'$	Koordinaattia $x$ vastaava koordinaatti
$X$	Koordinaatti avaruudessa
$X_c$	Karteesinen koordinaatti kameras koordinaatistossa
$X_w$	Karteesinen koordinaatti maailman koordinaatistossa
$[x_1, x_2]^T$	Karteesinen koordinaatti avaruudessa, myös $[x, y]^T$
$[X_1, X_2, X_3]^T$	Karteesinen koordinaatti avaruudessa, myös $[X, Y, Z]^T$
$W$	Maailman koordinaatisto
$\lambda$	Tuntematon skaalakerroin

# 1 JOHDANTO

Ympäristöstä muodostetuille tarkoille 3D-malleille on tarvetta useissa eri käytännön sovelluksissa. Roboteilta vaaditaan nykyään yhä monimutkaisempien toimintojen hoitamista [1], joten ympäristöistä tarvitaan entistä parempia 3D-malleja robottien toimintojen tueksi. 3D-malleja tarvitaan muun muassa robotin navigoimiseen, tavaroiden liikuttamiseen sekä toimintojen suunnittelemiseen etukäteen. 3D-malleille löytyy käytännön sovelluksia myös monelta muultakin teollisuuden alalta, esimerkiksi arkkitehtuurin, arkeologian, filmitieteellisuuden ja virtuaaliturismin parista.

Ympäristön 3D-mallien luominen on mahdollista tehdä joko manuaalisesti tai automaattisesti. Pienempiä ympäristöjä mallinnettaessa 3D-malli muodostetaan useimmiten ainakin osittain ihmisen avustamana, sillä täysin automaattisesti ympäristöä mallintavan järjestelmän toteuttaminen on sovelluskohteena huomattavasti vaikeampi. Suuria ympäristöjä mallinnettaessa työmäärä kasvaa kuitenkin nopeasti niin suureksi, että manuaalinen ympäristön mallintaminen ei ole enää järkevä vaihtoehto. Tämän vuoksi ollaan kiinnostuneita kehittämään entistä parempia ja itsenäisempiä järjestelmiä ympäristön automaattiseen 3D-mallintamiseen.

3D-mallin luominen robotin avulla edellyttää, että robotti saa kerättyä ympäristöstä 3D-informaatiota jollain tarkoitukseen sopivalla menetelmällä. 3D-datan keräämisen on useita eri vaihtoehtoja [2], mutta useimmiten toteutuksissa käytetään jotain optiseen menetelmään perustuvaa tekniikkaa, jolla 3D-informaatio on mahdollista kerätä luotettavasti etäisyyksien päästä. Yleisiä käytettyjä ratkaisuja ovat muun muassa 3D-kamerat, 3D-skannerit tai erilaiset rakenteista valoa hyödyntävät kamerat. Näilläkin teknologioilla on kuitenkin omat rajoituksensa, minkä vuoksi ne eivät sovellu käytettäväksi mallinnettaessa erittäin suuria ympäristöjä, esimerkiksi kokonaisia kaupunkia.

Suuria ympäristöjä mallinnettaessa parempi vaihtoehto on muodostaa 3D-malli pelkästään 2D-kuvia hyödyntämällä. Tällöin tarvittava data saadaan kerättyä esimerkiksi suoraan videosta, jolloin datan kerääminen on huomattavasti halvempaa, käytännöllisempää ja nopeampaa, kuin erilaisia 3D-datan mittaamiseen tarkoitettuja kameroita käyttämällä. 3D-mallin muodostaminen 2D-kuvien pohjalta perustuu niin sanottuun *monen näkymän geometriaan* (multiple view geometry).

## 1.1 Tutkittava ongelma

Tämän diplomityön aiheena on ympäristön 3D-mallintaminen quadrokopterilla kuvatun 2D-videon pohjalta. 3D-mallin muodostaminen perustuu monen näkymän geometriaa hyödyntävään ohjelmistoon. Diplomityössä ei ole toteutettu robotin automaattista navigointia, vaan on keskitytty ainoastaan ympäristön 3D-mallintamiseen liittyvien ongelmien ratkaisemiseen.

Ympäristön 3D-mallintamisessa luodaan geometrinen malli ympäristöstä, josta geometrinen mallia ei ole vielä olemassa. Tällaista jo olemassa olevan kohteen geometrinen mallintamista kutsutaan yleisesti *käänteiseksi insinöörityöksi* (reverse engineering). Käänteisen insinöörityön ongelman ratkaisemisen vaiheet ovat sovelluksesta riippumatta karkeasti samanlaiset. Esimerkiksi Várady ym. [2] jaottelevat vaiheet datan keräämiseen, datan esikäsittelyyn, segmentointiin ja pintojen sovittamiseen sekä mallin luomiseen. Vastaavanlainen ongelman jaottelu löytyy

useasta muustakin lähteestä, esimerkiksi [3], [4] ja [5]. Tällainen jaottelu on tietenkin hyvin suuntaa antava kuvaus ongelmasta. Usein vaiheet ovat päällekkäisiä ja niiden toteuttaminen vaatii useita iteraatioita. Jaottelu auttaa kuitenkin jakamaan ongelmaa helpommin käsiteltävissä oleviin kokonaisuuksiin ja tällainen jaottelu kuvaa hyvin useimpia järjestelmiä. Tässä työssä ongelman jaottelu on tehty pääpiirteissään vastaavalla tavalla ja vaiheet on nimetty seuraavasti:

- Datan kerääminen
- Datan esikäsittely
- Pistepilven muodostaminen
- Mallin muodostaminen

Tutkittavan ongelman ratkaiseminen edellyttää, että jokainen ongelman osa-alueista saadaan ratkaistua. Kyseessä on siis hyvin laaja ja vaikea ongelma ratkaistavaksi. Reaalimaailman kaltaisessa monimutkaisessa ympäristössä itsenäisesti toimivaa järjestelmää ei ole onnistuttu käytännössä toteuttamaan vielä kovinkaan hyvin, sillä jokaiseen ongelman osa-alueeseen liittyy vielä ratkaisemattomia haasteita [6]. Olemassa olevat järjestelmät on suunniteltu toimimaan reaalimaailmasta yksinkertaistetuissa ympäristöissä. Yleisin oletus on, että ympäristö oletetaan staattiseksi. Yleensä järjestelmä myös suunnitellaan toimimaan jossain erityisessä ympäristössä, jonka rakenteesta tiedetään jotain jo etukäteen. Esimerkkinä järjestelmät, jotka on suunniteltu toimimaan joko ulko- tai sisätiloissa. Myös ympäristön koko oletetaan useimmiten rajoitetuksi.

Ympäristön 3D-rakenteen selvittäminen videokuvan avulla on sellainen ongelma, jonka ratkaisemisessa keskeisiä menetelmiä ovat niin kutsutut *mallintaminen liikkeen avulla* ja *monen näkymän stereo*. Näitä menetelmiä tarvitaan, kun ympäristön 3D-informaatiota lasketaan 2D-kuvien pohjalta.

## 1.2 Ratkaisun yleiskuvaus

Tässä diplomityössä esitellään järjestelmä, jonka avulla ympäristöstä on mahdollista luoda 3D-malli pelkästään edullista quadrokopteria ja tavallista pöytätietokonetta käyttämällä. 3D-malli luodaan kuvatun 2D-videon pohjalta. Mallinnettava ympäristö on oletettu ongelman ratkaisun yksinkertaistamisen vuoksi staattiseksi ja kooltaan rajoitetuksi. Järjestelmässä käytettävä quadrokopteri on Parrot-yhtiön AR Drone 2.0 [7]. Tarvittavan ohjelmiston toteuttamisessa on hyödynnetty avoimeen lähdekoodiin pohjautuvaa VisualSFM-ohjelmistoa [8] sekä OpenCV-kirjastoa [9]. Puuttuvat ohjelmiston osat on toteutettu Python-ohjelmointikielen avulla.

Diplomityön tavoitteena ei ole toteuttaa nykyistä parempia järjestelmiä tai parantaa ongelman ratkaisemiseen käytettäviä menetelmiä, sillä aiheen laajuuden vuoksi yhdenkään osa-alueen tutkimiseen ei voida keskittyä kovinkaan tarkasti. Tämän diplomityön tarkoituksena on tehdä aiheesta alustavaa tutkimusta, jotta aiheeseen perehtyvä lukija pääsee nopeasti aiheen perusasioista selville sekä esitellä alustava toteutus järjestelmälle, jota on helppo kehittää myöhemmin paremmaksi. Työn tarkoituksena on myös selvittää sitä, että miten hyvin tällainen edullinen quadrokopteri soveltuu ongelman ratkaisemiseen ja toisaalta sitä, että mitä ongelmia tällaisen järjestelmän kehittämisessä kohdataan. Järjestelmän kehityksessä on huomioitu se, että järjestelmä pyritään toteuttamaan mahdollisimman edullisella laitteistolla ja ohjelmistolla.

## 2 TEOREETTISTA TAUSTAA

Ennen kuin siirrytään varsinaiseen ympäristön 3D-mallintamiseen 2D-kuvien pohjalta, niin lukijan on hyvä ymmärtää ongelman ratkaisemiseen tarvittava teoria ja matemaattinen perusta. 3D-avaruuden kuvaaminen 2D-tasolle on niin sanottu *perspektiivinen projektio*, minkä seurauksena 3D-informaatio menetetään. 3D-informaatio on kuitenkin mahdollista palauttaa, mikäli hyödynnetään useampia samasta näkymästä otettuja 2D-kuvia. Tämän kappaleen lukemisen jälkeen lukija ymmärtää tämän prosessin perusteet. Kappaleessa käsiteltävä teoria pohjautuu pääasiassa lähteisiin [10], [11] ja [12] sekä myös lähteisiin [13] ja [14]. Lähteissä on paljon päällekkäistä tietoa, joten kappaleiden lopussa on viitattu yleensä vain siihen lähteeseen, johon kappaleen sisältö pääasiassa perustuu.

Tämän kappaleen rakenne on seuraava: Kappaleessa 2.1 palautellaan mieliin tuttua euklidista geometriaa ja perehdytään ongelman ratkaisun kannalta tärkeään käsitteeseen *jäykkä kappale*. Kappaleessa 2.2 perustellaan ensin, että miksi ongelman ratkaisemiseen tarvitaan projektiivista geometriaa ja sen jälkeen perehdytään lyhyesti projektiivisen geometrian perusteisiin. Kappaleessa 2.3 jatketaan tarkemmin projektiivisen geometrian tutkimista kameramallin ja perspektiivisen projektion parissa. Lopuksi kappaleessa 2.4 perehdytään epipolaariseen geometriaan, jossa tutkitaan eri näkymien välillä olevaa yhteistä projektiivista geometriaa, sekä siihen, että miten 3D-informaatio saadaan palautettua epipolaarisen geometrian avulla.

### 2.1 Kolmiulotteinen euklidinen geometria ja jäykän kappaleen liike

Kolmiulotteista euklidista avaruutta kuvataan merkinnällä  $\mathbb{E}^3$ . Euklidisessa avaruudessa on voimassa: jokainen piste  $p \in \mathbb{E}^3$  voidaan esittää pisteenä avaruudessa  $\mathbb{R}^3$  kolmen koordinaatin avulla. Koordinaatit voidaan esittää vektorin

$$\mathbf{X} = [X_1, X_2, X_3]^T \quad (1)$$

avulla, usein käytetään myös muotoa  $[X, Y, Z]^T$ . Euklidisen geometrian koordinaatteja kutsutaan *karteesisiksi koordinaateiksi*. [10]

Euklidisessa avaruudessa kahden pisteen välinen etäisyys määritellään vektoreiden avulla. Vektori  $v$  on suunnattu nuoli pisteiden  $p$  ja  $q$  välillä, missä  $p, q \in \mathbb{E}^3$ . Jos  $p$ :llä on koordinaatit  $\mathbf{P}$  ja  $q$ :lla on koordinaatit  $\mathbf{Q}$ , niin silloin vektorilla  $v$  on koordinaatit

$$\mathbf{v} = \mathbf{Q} - \mathbf{P} \in \mathbb{R}^3 \quad (2)$$

Euklidisessa geometriassa *vapaa vektori* tarkoittaa sellaista vektoria, jonka määritelmä ei ole riippuvainen sen alkupisteestä. Toisin sanoen, jos on olemassa vektorit  $v$  ja  $v'$ , joiden koordinaatit täyttävät ehdon  $\mathbf{Q} - \mathbf{P} = \mathbf{Q}' - \mathbf{P}'$ , niin silloin ne määrittelevät saman vapaan vektorin. Vektorin  $v$  normi eli pituus ei siis muutu, kun se siirretään eri pisteeseen  $\mathbb{E}^3$ :ssa. Matemaattisesti tämä esitetään siten, että pisteiden  $p$  ja  $q$  koordinaateille  $\mathbf{P}(t)$  ja  $\mathbf{Q}(t)$  on voimassa [10]:

$$\|\mathbf{v}(t)\| = \|\mathbf{P}(t) - \mathbf{Q}(t)\| = \text{vakio}, \quad \forall t \in \mathbb{R}^3 \quad (3)$$

Fysiikassa puhutaan usein käsitteestä jäykkä kappale, joka tarkoittaa sellaista kappaletta, jonka muoto ei muutu siihen kohdistuneiden voimien seurauksena. Tällaista kappaletta voidaan siis fysikaalisesti siirtää, kääntää, pyörittää tai kallistaa, ilman että sen muoto muuttuu alkuperäisestä. Matemaattisesti tämä tarkoittaa sitä, että kappaleen pisteiden väliset etäisyydet eivät muutu muutoksien seurauksena. Lisäksi vaaditaan, että kappaleen koordinaatiston orientaatio säilyy muutoksessa. Esimerkiksi kuvaus

$$[X_1, X_2, X_3]^T \rightarrow [X_1, X_2, -X_3]^T \quad (4)$$

ei ole sallittu muutos, sille siinä kappaleen koordinaatiston orientaatio ei säily. Tämä muutos kuvaa peilikuvautumista, eikä sellainen ole fysikaalisessa mielessä jäykälle kappaleelle mahdollista.

Jäykän kappaleen liike muodostuu jäykän kappaleen siirtymästä  $t$  sekä rotaatiosta  $\mathbf{R}$ . Siirtymä eli *translaatio* voidaan ilmaista yhtälön (2) vektorin avulla ja rotaatio voidaan ilmaista yleisesti 3x3-matriisin

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \quad (5)$$

avulla, missä matriisin elementit  $r_{nm}$  ovat sini- ja kosinilauseita kulmien  $\Phi$ ,  $\theta$  ja  $\varphi$  suhteen [15]. Rotaatio suoritetaan kappaleen oman koordinaatiston  $C$  koordinaattiakselien suhteen, ensin  $x$ -, sitten  $y$ - ja lopuksi  $z$ -akselille. Jäykän kappaleen liike on siis geometrinen kuvaus  $\mathbb{R}^3 \rightarrow \mathbb{R}^3$ , joka kuvaa jäykän kappaleen jokaisen pisteen koordinaattien muuttumista ajan funktiona. Jäykän kappaleen pisteiden väliset suhteet pysyvät jokaisella ajan hetkellä vakiona, joten muutosta ei tarvitse tarkastella jokaisella ajan hetkellä erikseen, vaan se voidaan ilmaista yhden kuvauksen avulla  $\mathbf{X} \rightarrow g(\mathbf{X})$ , missä  $g = (\mathbf{R}, t)$ . [10]

Jäykän kappaleen siirtymän seuraaminen ei onnistu sen omasta koordinaatistosta käsin, sillä kappaleen siirtyessä sen jokainen piste siirtyy suhteessa muihin pisteisiin saman verran. Tämän vuoksi ei ole olemassa mitään pistettä, jonka suhteen siirtymää voisi tarkastella. Tämä ongelma ratkaistaan siten, että jäykkä kappale sijoitetaan ulkopuoliseen kiinteään koordinaatistoon  $W$ , jonka suhteen siirtymää voidaan tarkastella. Tätä koordinaatistoa kutsutaan usein *maailman koordinaatistoksi*. Maailman koordinaatiston origo voidaan sijoittaa yhtälön (3) perusteella mihin pisteeseen tahansa, sillä kappaleen suhteellinen siirtymä maailman koordinaatistoon verrattuna on joka tapauksessa sama ja se voidaan näin ollen kuvata kappaleen paikasta riippumatta saman vektorin avulla. [10]

Jäykän kappaleen koordinaatit maailman koordinaatistossa  $W$  voidaan ratkaista yhtälöllä

$$\begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix}_W = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}_{wc} \begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix}_C + \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix}_{wc} \quad (6)$$

tai tiiviimmin esitettynä

$$\mathbf{X}_W = \mathbf{R}_{wc} \mathbf{X}_C + t_{wc} \quad (7)$$

missä  $X_C$  on kappaleen koordinaattien sijainti suhteessa sen omaan koordinaatistoon  $C$ , ja  $R_{wc}$  sekä  $T_{wc}$  ovat koordinaatistojen  $W$  ja  $C$  suhteen tapahtunut rotaatio ja translaatio. [10]

## 2.2 Projektiivinen geometria

Euklidista geometriaa käytetään yleisesti kuvaamaan matemaattisia objekteja ja avaruuksia, sillä euklidisen geometrian avulla voidaan tarkasti esittää kappaleiden muoto, koko ja sijainti avaruudessa. Tämän vuoksi se on useimmiten käytetty vaihtoehto kuvaamaan sellaisia 3D-ympäristöjä, joiden voidaan olettaa olevan ainakin ympäristön ratkaisevilta osin muuttumattomia, toisin sanoen jäykkiä kappaleita. Ympäristön 3D-mallintaminen 2D-kuvien pohjalta ei kuitenkaan onnistu pelkästään euklidista geometriaa hyödyntämällä, sillä valokuva on perspektiivinen projektio 3D-avaruudesta 2D-tasolle, toisin sanoen  $\mathbb{R}^3 \rightarrow \mathbb{P}^2$ , missä  $\mathbb{P}^2$  kuvaa kaksiulotteista projektiivista avaruutta. Tämä kuvaus edellyttää geometrian laajentamista projektiiviseen geometriaan. Projektiivinen geometria on geometrian osa-alue, joka tarkastelee sellaisia geometrisia ominaisuuksia, jotka säilyvät projektiivisissä muunnoksissa, toisin sanoen muunnoksissa  $\mathbb{P}^n \rightarrow \mathbb{P}^n$ . [11]

Taulukko 1 [11], [14] vertailee eri geometrian osa-alueita. Taulukosta huomataan, että euklidisen ja projektiivisen geometrian välissä ovat niin sanotut *affiini* ja *metrinen geometria*. Nämä eri geometriat voidaan erotella toisistaan *vapausasteiden* (degrees of freedom, lyh. DOF) avulla [14], jotka ilmaisevat niiden parametrien määrän, jotka voivat muuttua toisistaan riippumatta. Näiden parametrien avulla määritellään geometriset muunnokset, jotka on esitelty taulukon ylemmässä lohossa. Taulukon alemmassa lohossa on esitelty ne geometriset ominaisuudet, jotka säilyvät ylemmässä lohossa mainituissa geometrisissa muunnoksissa. Taulukosta on luettavissa, että perspektiivinen projektio riippuu 15:sta eri parametrasta ja muutoksen seurauksena geometrisista ominaisuuksista säilyvät muuttumattomana muun muassa suoruus ja kaksoissuhde. Tämä tarkoittaa käytännössä sitä, että kun 3D-informaatiota lähdetään palauttamaan 2D-näkymien pohjalta, niin se tehdään näiden geometrinen ominaisuuksien pohjalta. [11]

Taulukko 1. Eri geometrian osa-alueiden vertailua

<b>DOF 3D-avaruudessa</b>	euklidin. 6	metrinen 7	affiini 12	projekt. 15
<b>Geometrinen muunnos</b>				
rotaatio, translaatio	x	x	x	x
isotrooppinen skaalaus		x	x	x
skaalaus akseleita pitkin, leikkaus			x	x
perspektiivinen projektio				x
<b>Muuttumattomat ominaisuudet</b>				
absoluuttiset etäisyydet	x			
kulmat, suhteelliset etäisyydet	x	x		
yhdensuuntaisuus, keskipiste	x	x	x	
suoruus, kaksoissuhde	x	x	x	x

Taulukko 1 informaatiota voidaan tarkastella vielä käytännön esimerkin avulla, mikäli asiaa mietitään ihmisen näkemisen avulla. Ihmisen näköjärjestelmä voidaan ajatella toimivan kameran tavoin, sillä silmä tuottaa perspektiivisen projektion 3D-ympäristöstä. Perspektiivisen projektion vaikutusta ympäristön geometriaan tarkastellaan usein esimerkin avulla, jossa ihminen seisoo junaradalla ja katsoo kaukaisuuteen loittonevia raiteita. Euklidisen geometrian pohjalta tiedämme varmasti, että 3D-avaruudessa raiteet ovat yhdensuuntaisia ja samankokoisia, sekä lähellä että kaukana. Ihmisen silmin raiteet näyttävät kuitenkin lähenevän toisiaan kauempana ja lopulta kohtaavan samassa pisteessä, jota kutsutaan *pakopisteeksi*. Samoin kappaleiden koko muuttuu pienemmäksi, kun ne etääntyvät katsojasta. Tämän pohjalta osamme jo sanoa, että yhdensuuntaisuus ja absoluuttiset etäisyydet eivät ole sellaisia geometrisia ominaisuuksia, jotka säilyvät perspektiivisessä projektiossa. Samoin myös kulmat ja suhteelliset etäisyydet näyttävät vaihtelevan, kun ympäristöä tarkkaillaan eri kuvakulmista. Sen sijaan suora viiva on aina suora viiva, riippumatta siitä, että mistä kuvakulmasta sitä tarkastellaan. Suoruus on siis sellainen geometrinen ominaisuus, joka säilyy perspektiivisessä projektiossa, samoin kuin kaikissa muissakin geometrisissa muunnoksissa. Tätä tietoa voidaan hyödyntää, kun siirrytään eri geometriasta toiseen.

Projektiivisessa geometriassa pisteet kuvataan niin sanottujen *homogeenisten koordinaattien* avulla. Homogeeninen koordinaatti projektiivisessä avaruudessa  $\mathbb{P}^n$  esitetään  $(n + 1)$ -ulotteisena vektorina  $[X_1, \dots, X_n, w]^T$ . Samalla lisäksi rajoitetaan, että kaikki pisteet, joiden arvot ovat verrannollisia, esittävät samaa pistettä. Tällöin  $\mathbb{R}^2$  esitetyn euklidisen koordinaatin ja projektiivisessä tasossa  $\mathbb{P}^2$  esitetyn homogeenisen koordinaatin välillä on yhteys

$$[x_1, x_2]^T \leftrightarrow [x_1, x_2, w]^T \quad (8)$$

Homogeeninen koordinaatti voidaan muuntaa takaisin euklidiseksi koordinaatiksi jakamalla alkuperäiset koordinaatit lisätyllä koordinaatilla  $w$  ja pudottamalla  $w$ -koordinaatti sen jälkeen pois, toisin sanoen

$$[x_1, x_2, w]^T \leftrightarrow [x_1/w, x_2/w]^T \quad (9)$$

kun  $w \neq 0$ . Yksi syy homogeenisten koordinaattien käyttöön on, että niiden avulla voidaan mallintaa helposti niin sanottu ”piste äärettömyydessä” käyttämällä äärellisiä arvoja. Projektiivisessä geometriassa se saavutetaan silloin, kun koordinaatti  $w$  on nolla. Euklidisessä geometriassa tällainen piste ajatellaan ”pisteeksi äärettömyydessä”. Homogeeninen koordinaatti ei määrittele yksikäsitteisesti pistettä, sillä mikäli  $w \neq 0$ , niin silloin kaikki koordinaatit  $[x_1, x_2, w]^T$  kuvautuvat samaksi pisteeksi. Kun homogeenisia koordinaatteja muutetaan euklidisiksi koordinaateiksi, niin usein tehdään oletus, että  $w = 1$ , jolloin yhtälö (9) yksinkertaistuu muotoon

$$[x_1, x_2, 1]^T \leftrightarrow [x_1, x_2]^T \quad (10)$$

Tällainen oletus johtaa siihen, että eri avaruuksissa esitettyjen koordinaattien välistä skaalakerrointa ei saada määritettyä yksikäsitteisesti. [12], [14]

Projektiivinen muunnos 3D-avaruudessa voidaan määritellä yleisesti matriisin

$$\mathbf{T}_P = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \\ p_{41} & p_{42} & p_{43} & p_{44} \end{bmatrix} \quad (11)$$

avulla. Matriisissa  $\mathbf{T}_P$  on 16 parametria, mutta se on määritelty vain, mikäli nollasta poikkeava skaalakerroin on valittu, jolloin vapausasteita siinä on tällöin 15. Yleinen valinta on edellisessä kappaleessa esitetty  $w = 1$ . [14]

Kaksoissuhde määritellään seuraavasti [14]: Kuvitellaan neljä samalla suoralla olevaa pistettä  $\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3, \mathbf{X}_4$ . Ilmaistaan ne sen jälkeen  $\mathbf{X}_i = \mathbf{X} + \sigma_i \mathbf{X}'$  sillä oletuksella, että mikään pisteistä  $\mathbf{X}_i$  ei ole sama pisteen  $\mathbf{X}'$  kanssa. Silloin kaksoissuhde määritellään

$$\{\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3, \mathbf{X}_4\} = \frac{\sigma_1 - \sigma_3}{\sigma_1 - \sigma_4} : \frac{\sigma_2 - \sigma_3}{\sigma_2 - \sigma_4} \quad (12)$$

Kaksoissuhde ei ole riippuvainen pisteiden  $\mathbf{X}$  ja  $\mathbf{X}'$  sijainnista ja se ei muutu projektiivisen muutoksen seurauksena. Kaksoissuhdetta voi tarkastella myös neljän suoran avulla, jotka leikkaavat samassa pisteessä. Olkoon tämä leikkauspiste  $\mathbf{C}$  ja jokainen neljästä suorasta kulkee yhden pisteen  $\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3$  tai  $\mathbf{X}_4$  kautta, siten että jokaisen pisteen kautta kulkee täsmälleen yksi suora. Tämän jälkeen suora  $l$  leikkaa nämä neljä suoraa, jolloin se muodostaa näille suorille pistejoukon  $\mathbf{X}'_1, \mathbf{X}'_2, \mathbf{X}'_3, \mathbf{X}'_4$ , jotka kaikki sijaitsevat suoralla  $l$ . Näiden pisteiden välinen kaksoissuhde säilyy, riippumatta suoran  $l$  leikkauskohdasta tai pisteen  $\mathbf{C}$  sijainnista. [14], [16]

### 2.3 Kameramalli ja perspektiivinen projektio

Perspektiivinen projektio kuvaa sitä, että miten 3D-piste  $\mathbf{X}$  kuvautuu 2D-pisteeksi  $\mathbf{x}$  kuvatasolle. Se voidaan esittää matemaattisesti yksinkertaisessa muodossa

$$\mathbf{x} = \mathbf{P}\mathbf{X} \quad (13)$$

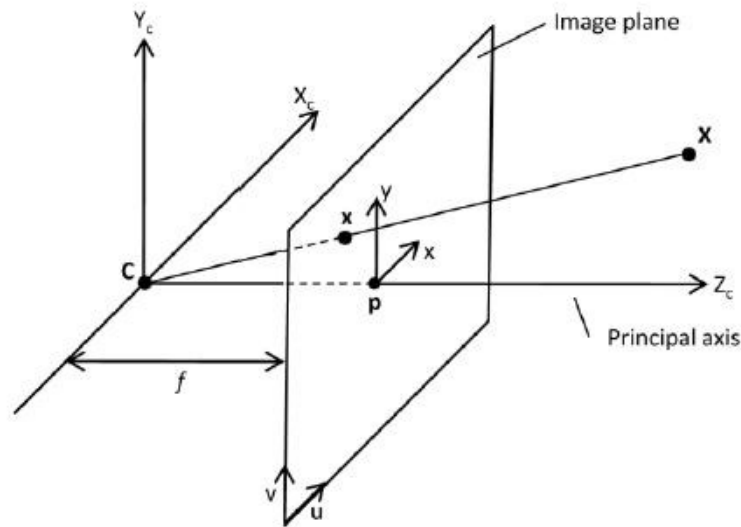
missä  $\mathbf{x}$  ja  $\mathbf{X}$  on esitetty homogeenisessa muodossa ja  $\mathbf{P}$  on 3x4 matriisi, jota kutsutaan *kameran projektiomatriisiksi*. Johdetaan seuraavaksi kameran projektiomatriisi  $\mathbf{P}$ .

Ideaalinen perspektiivinen projektio pohjautuu ideaaliseen kameramalliin, niin sanottuun *neulanreikäkameramalliin*. Tällainen on matemaattinen kameramalli, mutta se approksimoi erittäin hyvin useimpia käytettyjä kameroita. Neulanreikäkamera ei sisällä linssiä, vaan siinä kaikki valo kulkee yhden pisteen kautta. Tätä pistettä kutsutaan *projektiokeskukseksi*. Neulanreikäkameramallin koordinaatiston origo sijaitsee tässä pisteessä ja sen z-akselia sanotaan *optiseksi* akseliksi. Pisteet kuvautuvat kuvatasolle, joka on kohtisuorassa optiseen akseliin nähden. Kuvataason etäisyyttä projektiokeskuksesta sanotaan *kameran polttoväliseksi* tai *kameravakioksi*. Kuvataason ja optisen akselin leikkauskohtaa sanotaan *pääpisteeksi*. [10]

Piste  $\mathbf{X}$  siis kuvautuu projektiokeskuksen kautta sen takana olevalle kuvatasolle. Pisteen  $\mathbf{X}$  ja projektiokeskuksen kautta kulkevaa suoraa kutsutaan *projektiosuoraksi*. Trigonometrian avulla saadaan selville, että jokainen projektiosuoralla oleva piste kuvautuu samoihin koordinaatteihin

$$x = -f \frac{X}{Z}, \quad y = -f \frac{Y}{Z} \quad (14)$$

Usein kuvataso tuodaan kameran takaa kameran eteen, jolloin kameran eteen muodostuu niin sanottu *virtuaalinen kuvataso*. Tällöin  $-f$  voidaan merkitä  $f$ . Kuva 1 [17] havainnollistaa tätä toimenpidettä. [10]



Kuva 1. Neulanreikäkameramalli. Kuvassa  $C$  on projektiokeskus,  $f$  on polttoväli ja  $p$  on pääpiste. 3D-piste  $X$  kuvautuu pisteeksi  $x$  kuvatasolle.

Homogeenisessa muodossa ilmaistuna kuvautuminen voidaan lopulta esittää yhtälöllä

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \mathbf{A}_f \mathbf{P}_i \mathbf{X}_c = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix} \quad (15)$$

missä matriisi  $\mathbf{A}_f \mathbf{P}_i$  on ideaalinen projektiomatriisi. Yhtälössä  $1/Z_w$  on kerrottu yhtälön toiselle puolelle ja korvattu merkinnällä  $\lambda$ .  $Z$ :n arvoa ei yleensä tiedetä, koska sitä ei voida perspektiivisessä projektiossa määrittellä yksikäsitteisesti. Tätä voi tarkastella vielä yllä olevan kuvan avulla. Kuvasta selviää, että jokainen suoralla  $CX$  oleva piste kuvautuu samaksi pisteeksi kuvatasolla, joten pisteen  $Z$ -koordinaattia ei voida määrittellä yksikäsitteisesti. Jatkossa sitä käsitellään tuntemattomana parametrina  $\lambda$ . Se kuvaa käytännössä sitä, että minkä kokoinen kuvautunut objekti on suhteessa sen alkuperäiseen kokoon. [10]

Kuvauksesta on huomioitava myös se, että kuvaus on esitetty kameran koordinaatistossa  $C$ . Lopputulos halutaan kuitenkin esittää kiinteässä koordinaatistossa  $W$ , joten koordinaatit pitää vielä muuttaa tähän koordinaatistoon. Tämä tapahtuu aikaisemmin esitetyn yhtälön (7) avulla, joka on homogeenisessa muodossa esitettynä

$$\mathbf{X}_w = \begin{bmatrix} \mathbf{R}_{wc} & t_{wc} \\ 0 & 1 \end{bmatrix} \mathbf{X}_c = \begin{bmatrix} \sigma r_{11} & \sigma r_{12} & \sigma r_{13} & t_x \\ \sigma r_{21} & \sigma r_{22} & \sigma r_{23} & t_y \\ \sigma r_{31} & \sigma r_{32} & \sigma r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix} \quad (16)$$

Tämän jälkeen päästään yhtälöön

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \mathbf{A}_f \mathbf{P}_i \begin{bmatrix} \mathbf{R}_{wc} & t_{wc} \\ 0 & 1 \end{bmatrix} \mathbf{X}_c \quad (17)$$

mikä on siis perspektiivinen projektio koordinaatistossa  $W$  esitettyä ja ideaalisella kameralla kuvattuna. Tätä kuvausta sanotaan *ideaaliseksi keskusprojektiokuvaksi*. Rotaatio- ja translaatiomatriisin parametreja kutsutaan *kameran ulkoisiksi parametreiksi*. [10]

Tällainen kuvaus on voimassa vain ideaalisessa kameramallissa, missä kaikki pisteet kuvautuvat kuvatasolle saman pisteen kautta. Tällainen ei ole realistinen ajattelumalli, sillä oikeissa kameroissa on linssi ja näin ollen pisteet kuvautuvat kuvatasolle useamman pisteen kautta. Samoin on huomioitava se, että digitaalivalokuvauksessa kuva muodostetaan diskreettien yksiköiden eli pikseleiden avulla, eikä jatkuvia arvoja käyttämällä. Realistisemman yhtälön muodostamiseksi matriisi  $\mathbf{A}_f$  korvataan niin sanotulla *kalibrointimatriisilla*. Yleisin tapa esittää kalibrointimatriisi [18] on muodossa

$$\mathbf{A} = \begin{bmatrix} \alpha_x & s_\theta & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \quad (18)$$

missä  $\alpha_x = f \cdot m_x$ ,  $\alpha_y = (f \cdot m_y) / \sin \theta$  ja  $s_\theta = f \cdot m_x \cdot \cot \theta$ . Parametri  $f$  on jo aikaisemmista yhtälöistä tuttu ja ilmaisee kameras polttovälin. Parametrit  $m_x$  ja  $m_y$  ilmaisevat pikseleiden määrän suhteessa vastaavien akselien etäisyyksiin, parametrit  $x_0$  ja  $y_0$  ovat pääpisteen koordinaatit ja parametri  $\theta$  kuva-akselien  $x$  ja  $y$  välistä kulmaa. Useimmiten kuva-akselit ovat toisiinsa nähden kohtisuoria, jolloin  $s_\theta = 0$  ja  $\sin \theta = 1$ . Matriisin  $\mathbf{A}$  parametreja kutsutaan *kameran sisäisiksi parametreiksi*. [10], [18]

Yhtälössä (18) esitetyn kalibrointimatriisin käyttö edellyttää, että kuvauksessa ei tapahdu virhettä. Virheillä tarkoitetaan tässä tapauksessa poikkeamaa ideaalisesta keskusprojektiokuvasta. Tosiasiassa kamerat eivät ole kuitenkaan täydellisiä, joten kuvaukseen tulee virheitä. Aiheutunut virhe on jokaiselle kameralle yksilöllinen ja vääristymä aiheutuu pääasiassa kahdesta syystä: radiaalisesta ja tangentialisesta vääristymästä [9]. Radiaalinen vääristymä aiheutuu linssin muodosta, sillä siitä on valmistusteknisesti vaikea tehdä täysin oikean muotoista. Tangentialinen vääristymä johtuu taas siitä, että linssiä ei kokoamisprosessin aikana saada välttämättä yhdensuuntaiseksi kuvatason kanssa. Vääristymät kasvavat kuvan reunoja kohti mentäessä. Yhtälön (18) matriisin käyttö ei siis käytännössä anna tarkkaa kuvausta, joten tämän vuoksi matriisi  $\mathbf{A}$  määritellään uudelleen

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \quad (19)$$

siten, että parametrit  $a_{nm}$  huomioivat myös kameran linssin aiheuttamat vääristymät. Kameran kalibrointi vaatii siis yhdeksän eri sisäisen parametrin tuntemisen. Tässä diplomityössä ei perehdytä enää sen tarkemmin näiden parametrien selvittämiseen, mutta kiinnostunut lukija voi perehtyä aiheeseen esimerkiksi lähteiden [15] ja [19] avulla.

Lopulta perspektiivinen projektio voidaan siis esittää yhtälön

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \mathbf{A} \mathbf{P}_i \begin{bmatrix} \mathbf{R}_{wc} & t_{wc} \\ 0 & 1 \end{bmatrix} \mathbf{X}_c \quad (20)$$

avulla, lyhemmin ilmaistuna  $\lambda \mathbf{x} = \mathbf{P} \mathbf{X}$ . Parametria  $\lambda$  ei voida selvittää, mikäli  $Z$ -koordinaatin arvoa ei tiedetä.

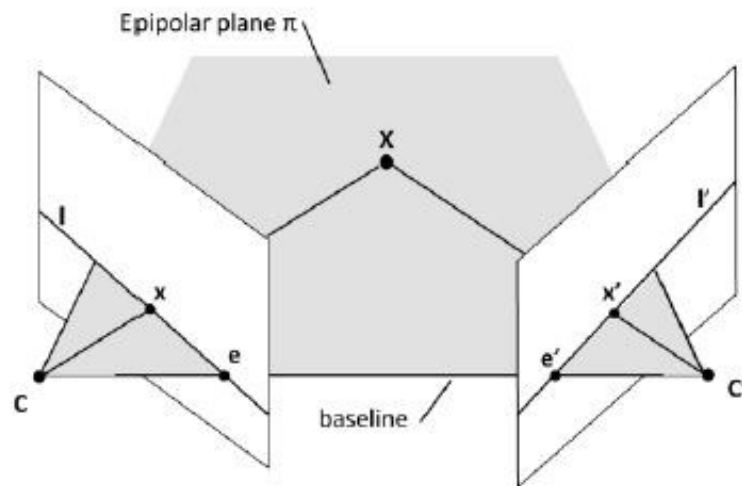
## 2.4 Epipolaarinen geometria ja 3D-informaation palauttaminen

Edellisen kappaleen perusteella tiedetään, että perspektiivisessä projektiossa ei saada selvitettyä koordinaattia  $Z$ , jolloin pisteen etäisyyttä kamerasta ei tiedetä. Tämän vuoksi 3D-informaation palauttaminen yhden näkymän avulla ei ole mahdollista. Sen sijaan kahden tai useamman näkymän avulla 3D-koordinaattien laskeminen on mahdollista, mikäli tiedetään, että ne ovat perspektiivisiä projektioita samasta 3D-näkymästä. Tällöin tiedetään, että eri kuvakulmista otetut kuvat ovat muunnoksia  $\mathbb{P}^2 \rightarrow \mathbb{P}^2$  samasta avaruudesta  $\mathbb{R}^3$  ja sisältävät näin ollen yhteistä projektiiivista geometriaa. Yleisesti kahden eri näkymän yhteistä projektiiivista geometriaa kutsutaan *epipolaariseksi geometriaksi*.

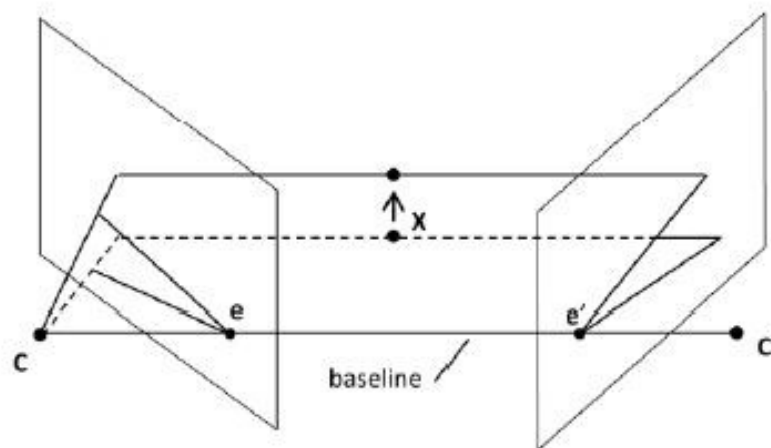
Kuvitellaan piste  $\mathbf{X} = [X_1, X_2, X_3]^T$  avaruudessa  $\mathbb{R}^3$  ja kuvataan se pisteeksi  $\mathbf{x} = [x_1, x_2]^T$  koordinaatiston  $W$  pisteestä  $\mathbf{X}_c$  tarkasteltuna, jolloin yhtälön (13) mukaisesti  $\mathbf{x} = \mathbf{P} \mathbf{X}$ . Vastaavasti  $\mathbf{X}$  voidaan kuvata myös mistä tahansa muustakin koordinaatiston  $W$  pisteestä, merkitään tätä pistettä  $\mathbf{X}'_c$ . Tällöin  $\mathbf{X}$  kuvautuu pisteeksi  $\mathbf{x}' = \mathbf{P}' \mathbf{X}$ . Pisteiden  $\mathbf{x}$  ja  $\mathbf{x}'$  välillä tiedetään nyt olevan geometrinen yhteys, sillä ne ovat perspektiivisiä projektioita samasta pisteestä  $\mathbf{X}$ . Tämän geometrisen yhteyden avulla pisteen  $\mathbf{X}$  koordinaatit on mahdollista selvittää. [12]

Kuva 2 [17] havainnollistaa kahden eri näkymän välillä olevaa geometrista yhteyttä. Kuvassa kameroiden keskuksia on merkitty  $\mathbf{C}$  ja  $\mathbf{C}'$ . Kameroiden etupuolelle on sijoitettu virtuaaliset kuvatason, joita merkitään jäljempänä  $R$  ja  $R'$ . Pisteiden  $\mathbf{C}$ ,  $\mathbf{C}'$  ja  $\mathbf{X}$  muodostamaa tasoa sanotaan *epipolaariseksi tasoksi* ja se on merkitty kuvassa  $\pi$ . Tämä taso leikkaa kuvatason  $R$  ja  $R'$  ja muodostaa näin suorat  $l$  ja  $l'$ , joita sanotaan *epipolaarisiksi suoriksi*. Pisteiden  $\mathbf{X}$  tarkka sijainti ei ole tiedossa, mutta se tiedetään varmasti, että sen täytyy olla pisteiden  $\mathbf{C}$  ja  $\mathbf{x}$  kautta kulkevalla suoralla. Tämä suora näkyy kamerasta  $\mathbf{C}$  tarkasteltuna pisteenä, mutta kamerasta  $\mathbf{C}'$  tarkasteltuna se näyttää suoralta ja se kuvautuu sen kuvatason  $R'$  suoraksi  $l'$ . Tästä saadaan *epipolaarinen rajoite*, että pisteen  $\mathbf{x}'$  täytyy sijaita suoralla  $l'$ . Toisin sanoen, vaikka pisteen  $\mathbf{x}'$  sijaintia ei tiedettäisikään, niin sen sijainti on hyvin rajoitettu ja sen etsiminen voidaan rajoittaa epipolaariselle suoralle. [12], [13]

Suoran  $CC'$  kautta voidaan muodostaa useita epipolaarisia tasoja, joista jokainen muodostaa kuvapinnoille omat epipolaariset suoransa. Kuva 3 [17] havainnollistaa tätä. Kuvasta on havaittavissa, että epipolaariset suorat kulkevat pisteiden  $e$  ja  $e'$  kautta. Näitä pisteitä kutsutaan *epipolaarisiksi pisteiksi* ja ne ovat käytännössä projektio toisen kameran projektiokeskuksesta. Epipolaariset pisteet sijaitsevat siellä, missä suora  $CC'$  leikkaa kuvatasot  $R$  ja  $R'$ . Tällä tavoin tulee selväksi, että jokaiselle millä tahansa epipolaarisella tasolla sijaitsevalle pisteelle  $x$  löytyy vastine  $x'$  tasoa vastaavalta epipolaariselta suoralta  $l'$ . Mikäli pisteiden  $x$  ja  $x'$  koordinaatit ovat tiedossa, niin silloin myös niiden projektiosuorat tiedetään. Molempien pisteiden projektiosuorat leikkaavat pisteessä  $X$ , jolloin sen koordinaatit voidaan laskea trigonometrian avulla. [12], [13]



Kuva 2. Näkymien  $C$  ja  $C'$  välinen geometrinen yhteys rajoittaa pisteen  $x'$  sijainnin siten, että sen täytyy olla suoralla  $l'$ .



Kuva 3. Epipolaarinen geometria. Pisteet  $C$  ja  $C'$  yhdistävä suora muodostaa joukon epipolaarisia tasoja ja jokaiselle näistä tasosta löytyy vastaavat epipolaariset suorat  $l$  ja  $l'$ . Jokainen epipolaarinen suora  $l$  ja  $l'$  kulkee vastaavien epipolaaristen pisteiden  $e$  ja  $e'$  kautta.

Matemaattisesti epipolaarinen geometria esitetään *fundamentaalisin matriisin* (fundamental matrix)  $\mathbf{F}$  avulla. Fundamentaalisiksi matriisiksi kutsutaan sitä matriisia, joka täyttää ehdon

$$\mathbf{x}'^T \mathbf{F} \mathbf{x} = 0 \quad (21)$$

$\mathbf{F}$  on  $3 \times 3$  matriisi ja  $\mathbf{x}$  sekä  $\mathbf{x}'$  on esitetty homogeenisessa muodossa. Tähän esitysmuotoon päästään, kun palataan yhtälöön (15) ja muodostetaan siitä yhtälöpari kahden pisteen avulla seuraavasti:

$$\begin{cases} \lambda \mathbf{x} = \mathbf{A} \mathbf{P}_i \begin{bmatrix} \mathbf{I} & 0 \\ 0 & 1 \end{bmatrix} \mathbf{X}_c \\ \lambda' \mathbf{x}' = \mathbf{A}' \mathbf{P}_i \begin{bmatrix} \mathbf{R} & t \\ 0 & 1 \end{bmatrix} \mathbf{X}_c \end{cases} \quad (22)$$

Yhtälöön (15) verrattuna matriisi  $\mathbf{A}_f$  on korvattu matriisilla  $\mathbf{A}$ , jotta kameran sisäiset parametrit saadaan huomioitua. Kameroiden välinen suhteellinen liike on kuvattu yhtälöparin alemmassa yhtälössä rotaatio- ja translaatio matriisin avulla. Koordinaatti  $\mathbf{X}_c$  on esitetty koordinaatistossa  $C$  ja koska tiedetään, että  $\mathbf{X}_c$  on molemmissa näkymissä sama, niin se voidaan eliminoida yhtälöparista. Näin tehdään myös matriisille  $\mathbf{P}_i$ . Samoin parametrit  $\lambda$  ja  $\lambda'$  voidaan eliminoida, sillä niillä ei ole yhtälöparin ratkaisemisen kannalta merkitystä, mutta samalla hyväksytään se, että skaalakerrointa ei tiedetä. Lopulta yhtälöpari voidaan esittää [20], [21] muodossa

$$\mathbf{x}'^T \mathbf{A}'^{-T} \mathbf{E} \mathbf{A}^{-1} \mathbf{x} = 0. \quad (23)$$

missä  $\mathbf{F} = \mathbf{A}'^{-T} \mathbf{E} \mathbf{A}^{-1}$ . Matriisi  $\mathbf{E}$  on  $3 \times 3$  matriisi ja niin kutsuttu *essentiaalinen matriisi* (essential matrix)  $\mathbf{E} = \mathbf{T} \mathbf{R}$ , missä  $\mathbf{R}$  on tuttu  $3 \times 3$  rotaatiomatriisi ja matriisi  $\mathbf{T}$  määritellään

$$\mathbf{T} = \begin{bmatrix} 0 & -t_3 & t_2 \\ t_3 & 0 & -t_1 \\ -t_2 & t_1 & 0 \end{bmatrix} \quad (24)$$

Parametrit  $t_i$  ovat translaatiovektorin elementit  $t = [t_1, t_2, t_3]^T$ . Matriisi  $\mathbf{T}$  on käytännössä matriisimuotoinen ristitulo matriisin  $\mathbf{R}$  kanssa. Tämä ristitulo pitää sisällään kameroiden välisen suhteelliseen liikkeen. Kalibroinnin merkitys ongelman ratkaisemisen helpottamiseksi hahmottuu fundamentaalisen matriisin avulla. Mikäli kamera on kalibroitu, niin  $\mathbf{A}$  voidaan olettaa vakioksi, jolloin  $\mathbf{F} = \mathbf{E}$ . Tällöin  $\mathbf{F}$  riippuu ainoastaan matriisista  $\mathbf{R}$  ja vektorista  $t$ . [12], [18]

$\mathbf{F}$  voidaan ratkaista usealla eri menetelmällä [22], [23]. Kaikki menetelmät perustuvat siihen, että näkymistä etsitään toisiaan vastaavia pistepareja  $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$ . Matriisin  $\mathbf{R}$  ja vektorin  $t$  määrittäminen ei nimittäin onnistu yhden pisteen avulla, sillä niiden määrittämiseksi näkymissä oleva yhteinen projektiivinen geometria pitää tuntea. Yhteisen projektiivisen geometrian selvittäminen perustuu paljolti yhtälössä (12) esitettyyn kaksoissuhteen säilymiseen. Teoriassa  $\mathbf{F}$  voidaan ratkaista vähimmillään seitsemän molemmista näkymistä löytyvän pisteen avulla [23], mutta lineaarisen ratkaisun löytyminen vaatii kahdeksan pisteen käyttämistä [24]. Tätä menetelmää kutsutaan *kahdeksan pisteen lineaariseksi algoritmiksi* (eight-point linear algorithm) [20]. Menetelmän käyttö perustuu oletukseen, että pisteparit  $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$

saadaan määritettyä tarkasti, joten käytännössä se ei ole kuitenkaan kovin toimiva. Kuvissa olevan kohinan vuoksi on kehitetty useampia näkymiä käyttäviä menetelmiä, jolloin lopputulosta voidaan tarkentaa ja näin ollen kohinan vaikutusta saadaan vähennettyä. Tällaiset menetelmät etsivät pienimmän neliösumman ratkaisua [23].

Kun  $\mathbf{F}$  saadaan ratkaistua, niin sen jälkeen myös  $\mathbf{R}$  ja  $t$  voidaan ratkaista [10], jolloin koordinaatin  $\mathbf{X}_c$  arvot voidaan laskea trigonometrian avulla. Viimeisenä  $\mathbf{X}_c$  siirretään koordinaatistoon  $\mathcal{W}$  yhtälöä (16) käyttämällä. Tämä edellyttää, että ainakin toisen kameran sijainti suhteessa koordinaatistoon  $\mathcal{W}$  on tiedossa.

### 3 YMPÄRISTÖN 3D-MALLINTAMINEN

Tässä kappaleessa perehdytään robotin avulla tapahtuvaan ympäristön 3D-mallintamiseen. 3D-mallintamisella tarkoitetaan yleisesti prosessia, jossa kolmiulotteisesta pinnasta muodostetaan matemaattinen esitysmuoto 3D-avaruuteen tarkoitukseen suunnitellun ohjelmiston avulla. Tämän diplomityön kannalta oleellisinta asiaa on 2D-videon avulla tapahtuva ympäristön 3D-mallintaminen, joten pääasiassa keskitytään tämän ongelman käsittelyyn. Ongelman ratkaisemisessa ovat keskeisiä menetelmät nimeltään *mallintaminen liikkeen avulla* ja *monen näkymän stereo*, jotka mainitaan usein samassa yhteydessä. Usein käytetään myös nimitystä *monen näkymän geometria*, jolla saatetaan viitata kumpaan tahansa aiemmin mainittuun menetelmään tai sitten näiden menetelmien yhdessä muodostamaan kokonaisuuteen, jolla ympäristö voidaan mallintaa useiden 2D-kuvien pohjalta. Menetelmät ovatkin hyvin samankaltaisia siinä mielessä, että ne molemmat perustuvat samaan teoriaan, joka käsiteltiin kappaleessa 2. Ratkaiseva eroavaisuus on siinä, että monen näkymän stereo-menetelmät edellyttävät, että kameroiden paikat ovat tiedossa. Mallintaminen liikkeen avulla-menetelmä pyrkii mallintamaan ympäristön rakenteen karkeasti kameran liikkeen avulla, sekä selvittämään kameroiden paikat. Tämän jälkeen monen näkymän stereo-menetelmällä rakennetaan koko ympäristöstä yksityiskohtaisempi 3D-malli.

Kappaleen rakenne on seuraava: Kappaleessa 3.1 perehdytään ympäristön mallintamiseen robotin avulla yleisellä tasolla. Kappaleessa 3.2 käsitellään ongelmaa nimeltä *samanaikainen paikannus ja kartoitus*, joka on yksi keskeisimpiä ongelmia ympäristön automaattisessa geometrisessa mallintamisessa. Tässä diplomityössä tämän ongelman ratkaisemiseen ei ole käytetty aikaa, mutta lukijan on kuitenkin hyvä tietää asiasta perusteet. Kappaleissa 3.4 ja 3.3 käsitellään aikaisemmin mainitut mallintaminen liikkeen avulla ja monen näkymän stereo. Lopuksi kappaleessa 3.5 perehdytään olemassa oleviin järjestelmiin.

#### 3.1 Geometrinen mallintaminen robotin avulla

Robotin avulla tapahtuvaa ympäristön mallintamista on tutkittu aktiivisesti jo 1980- ja 1990-luvuilta lähtien [25]. Käytännössä ongelma tarkoittaa sitä, että ympäristöstä luodaan spatiaalinen malli liikkuvan robotin avulla. Ympäristön mallintaminen voidaan jakaa karkeasti joko topologiseen tai geometriseen mallintamiseen [25]. Topologisen mallin avulla kuvataan sitä, että miten ympäristössä olevat asiat, esineet ja paikat ovat liittyneet suhteessa toisiinsa. Topologisia malleja voidaan käyttää esimerkiksi yksinkertaisissa navigointitehtävissä. Monimutkaisempia tehtäviä toteutettaessa topologinen malli ei kuitenkaan ole riittävä, joten tämän vuoksi tarvitaan tarkempia geometrisia malleja. Geometrisessa mallissa ympäristö kuvataan matemaattisesti, esimerkiksi kuvaamalla ympäristö karteesisen koordinaatistoon.

Itsenäisesti ympäristössään toimiva robotti on sovelluskohteena hyvin vaativa, joten se asettaa vaatimuksia myös rakennettavalle geometriselle mallille. Esimerkiksi Wurm ym. asettivat omassa työssään [26] mallille seuraavat vaatimukset:

- *Täydellinen 3D-malli*: Mikä tahansa satunnainen ympäristö pitää pystyä esittämään mallin avulla ilman mitään aikaisempia oletuksia ympäristöstä. Mallin avulla pitää pystyä erottamaan vapaat alueet varatuista alueista sekä tutkimattomat alueet jo tunnetuista alueista. Vapaiden alueiden tunteminen on

tärkeää robotin sujuvan ja turvallisen liikkumisen kannalta. Tutkimattomien alueiden tunteminen auttaa puolestaan robottia tekemään päätöksiä siitä, että mitä alueita sen kannattaa seuraavaksi tutkia.

- *Päivitettävyyys*: Mallin pitää olla sellainen, että siihen voi helposti lisätä uutta informaatiota tai päivittää jo olemassa olevaa mallia uuden informaation pohjalta.
- *Joustavuus*: Reaalimaailmassa mallin kokoa ei voida tietää etukäteen, joten mallia pitää pystyä kasvattamaan dynaamisesti tarpeen vaatiessa. Samoin mallin tarkkuuden pitää pystyä mukautumaan tarpeen mukaan. Esimerkiksi suunnistukseen riittää usein hyvin karkea malli ympäristöstä. Esineen etsiminen ja liikuttaminen puolestaan vaatii tarkempaa mallia.
- *Tiivisyys*: Mallin pitää olla tallennettavissa mahdollisimman pieneen tilaan, sekä muistissa että kovalevyllä.

Geometriseen malliin tarvittavan 3D-informaation kerääminen on mahdollista toteuttaa karkeasti kahdella eri menetelmällä: aktiivisen tai passiivisen näkemisen avulla. Aktiivisessa näkemisessä keskitytään tarkkailemaan jotain tiettyä mielenkiintoista kohdetta kerrallaan. Ympäristöä havainnoivat sensorit ovat liikuteltavia, joten tarkkailtavaa kohdetta voidaan aktiivisesti vaihdella. Aktiivinen näkeminen perustuu siihen, että ympäristöön lähetetään energiaa, jonka vastetta voidaan mitata. Esimerkiksi 3D-kamerat, 3D-skannerit ja rakenteiseen valoon perustuvat tekniikat ovat tällaisia aktiivisen näkemisen menetelmiä. Sen sijaan passiivisessa näkemisessä pyritään tarkkailemaan koko ympäristöä kerrallaan. Tällöin ympäristöön ei lähetetä mitään energiaa, vaan mitataan ainoastaan ympäristön lähettämiä signaaleja. Käytännössä tämä tarkoittaa sitä, että ympäristöstä saadaan käyttöön vain 2D-kuvia, jolloin 3D-informaation selvittämiseen pitää käyttää monen näkyvän geometriaan perustuvia tekniikoita.

### 3.2 Samanaikainen paikannus ja kartoitus

Kun kirjallisuudessa puhutaan liikkuvan robotin avulla tapahtuvasta ympäristön mallintamisesta, usein viitataan sekä tarvittavan mallin muodostamiseen, että robotin itsenäiseen navigoimiseen ja paikantamiseen ympäristössä. Kyseessä on kaksi eri asiaa, mutta usein ongelmat ovat kuitenkin liitännäisiä. Automaattisesti ympäristöä mallintavan järjestelmän toteuttaminen edellyttää, että robotti kykenee liikkumaan ympäristössä itsenäisesti ja mallintamaan ympäristöä samanaikaisesti. Tällöin ollaan tekemisissä ongelman kanssa, josta käytetään nimitystä *samanaikainen paikannus ja kartoitus* (simultaneous localization and mapping, lyh. SLAM) kanssa.

SLAM-ongelmassa kysytään, että voiko tuntemattomaan ympäristöön sijoitettu robotti mallintaa ympäristöä ja samaan aikaan määrittellä oman sijaintinsa ympäristössä tätä mallia hyödyntämällä [27]. Automaattisesti ympäristöä 3D-mallintava robotti joutuu tekemisiin SLAM-ongelman kanssa, sillä tarkan geometrisen mallin muodostaminen edellyttää, että robotti on tietoinen omasta sijainnistaan ympäristössä. Toisaalta oman sijainnin tarkka määrittäminen edellyttää, että käytössä on tarkka malli ympäristöstä. Kyseessä on siis tavallaan klassinen *muna ja kana* -ongelma. Ilman tarkkaa mallia ympäristöstä robotti ei voi paikallistaa omaa sijaintiaan tarkasti, mutta tarkkaa mallia ei voida muodostaa ilman, että oma sijainti on tarkasti tiedossa. Ongelma kuulostaa siis sellaiselta, että sitä on mahdotonta ratkaista.

SLAM-ongelma on kuitenkin onnistuttu todistamaan teoriassa ratkeavaksi [28], mitä pidetäänkin yhtenä aiheen tutkimuksen merkittävimmistä saavutuksista. Siitä huolimatta SLAM-menetelmien käytännöntoteuttamiseen liittyy vielä useita ratkaisemattomia haasteita [25]:

- *Mittauskohina:* Tarkkojen mallien muodostaminen edellyttää, että käytössä on tarkkaa dataa ympäristöstä. Mittauskohinasta johtuen datassa on kuitenkin aina virheitä. Virheet pitää pystyä hallitsemaan, sillä inkrementaalissa ympäristön mallintamisessa virheet kertaantuvat ja pienikin virhe voi suuria ympäristöjä mallinnettaessa kasvaa todella isoksi. Mittauskohinan vuoksi mallinnuksessa käytettävien algoritmien toteuttaminen on paljon monimutkaisempaa, kuin mitä voisi aluksi kuvitella.
- *Reaalimaailman monimutkaisuus:* Reaalimaailmassa on todella paljon robotin toimintaan vaikuttavia muuttujia ja niiden kaikkien huomioiminen on jo pelkästään laskennallisessa mielessä erittäin vaikeaa. Muuttujien lisääminen kasvattaa aina toteutuksen kompleksisuutta, jolloin datan käsittelyyn tarvittava aika kasvaa. Reaalimaailmassa tapahtuu paljon myös täysin ennustamattomissa olevia asioita, joten tänäkään vuoksi kaikkien muuttujien huomioiminen ei ole käytännössä mahdollista.
- *Havaintojen vastaavuuden löytäminen:* Tätä ongelmaa pidetään mahdollisesti kaikkein vaikeimpana ongelmana ratkaista. Havaintojen vastaavuudella tarkoitetaan sitä osaa kahdesta eri näkymästä, joka on molemmissa näkymissä sama. Käytännössä robotin pitää siis tunnistaa mallin perusteella, mikäli se saapuu sellaiseen ympäristöön, joka on sille jo aikaisemmin tuttu. Havaintojen vastaavuuden löytäminen on erityisen vaikeaa mallinnettaessa syklisiä ympäristöjä, toisin sanoen sellaisia ympäristöjä, joissa robotti joutuu vierailemaan toistuvasti samassa paikassa ja paikkaan saavutaan eri reittiä pitkin kuin mistä sieltä lähdettiin.
- *Dynaaminen ympäristö:* Reaalimaailmassa tapahtuu paljon erilaisia muutoksia, joihin robotin on osattava reagoida oikealla tavalla. Esimerkkeinä ympäristössä tapahtuvista lukuisista muutoksista mainittakoon liikkuvat ihmiset ja eläimet, rakennusprojektit, vuodenaikojen vaihtelut sekä satunnaisesti paikkaa vaihtavat huonekalut, esineet ja autot. Esimerkiksi ihmisten ja eläinten liikkeet ovat sellaisia muutoksia, että niitä ei tarvitse huomioida ympäristön 3D-mallintamisessa, mutta robotin liikkumisessa ne pitää kuitenkin huomioida. Rakennusprojektit muuttavat ympäristöä pysyvästi, joten tällaiset muutokset pitäisi osata päivittää 3D-malliin. Vuodenaikojen vaihtelut eivät taas tavallaan muuta ympäristön rakennetta, mutta saavat sen näyttämään hyvin erilaiselta, mikä vaikeuttaa ympäristön tunnistamista. Dynaamisessa ympäristössä toimiminen edellyttää, että robotti osaa huomioida kaikki ympäristössä esiintyvät dynaamiset piirteet oikealla tavalla, joten käytännössä se on erittäin vaikeaa.
- *Robotin pitää tehdä päätöksiä puutteellisen tiedon varassa:* Robotin on pystyttävä tekemään päätöksiä liikkeistään ja toiminnoistaan samalla kun se mallintaa ympäristöä. Tunteamattomassa ympäristössä toimiessa liikkeiden ja toimintojen suunnittelu ei ole kuitenkaan niin helppoa, sillä päätökset joudutaan tekemään puutteellisen tiedon varassa. Robotti ei voi tietää, että mitä seuraavan kulman tai oven takana odottaa, mutta siitä huolimatta sen on jollain järkevällä tavalla osattava arvioida, että mitä sen kannattaa seuraavaksi tehdä.

Edellä mainitut haasteet vaikeuttavat niin robotin itsenäistä liikkumista ja navigointia kuin myös ympäristön geometrista mallintamista. SLAM-ongelman haasteita miettiessä onkin helppo ymmärtää, että ympäristön geometrinen mallintaminen ja robotin itsenäinen liikkuminen ja navigointi eivät ole helppoja ongelmia ratkaistavaksi reaali maailman kaltaisissa monimutkaisissa ympäristöissä. Näiden haasteiden lisäksi on huomioitava se, että reaali maailmassa toimiminen edellyttää asioiden käsittelyä reaaliajassa, sillä muuten järjestelmä ei kykene vastaamaan ympäristössä tapahtuviin muutoksiin riittävän nopeasti ja on näin ollen vaaraksi itselle ja muille ympäristössä liikkuville objekteille. Reaaliaikavaatimus tekee suurimmasta osasta mahdollisista menetelmistä käyttökelvottomia käytännötoteutuksissa [25].

SLAM-ongelman ratkaisemista käytännössä pidetään yhtenä tärkeimmistä haasteista, ennen kuin voidaan rakentaa oikeasti itsenäisesti ympäristössään liikkuvia ja toimivia robotteja [6]. SLAM-algoritmeja hyödyntäviä sovelluksia on toki kehitetty jo monenlaisiin eri ympäristöihin [27], mutta kaikille olemassa oleville sovelluksille yhteistä on se, että ne on suunniteltu toimimaan reaali maailmasta yksinkertaistetussa ympäristössä.

### 3.3 Mallintaminen liikkeen avulla

3D-informaation laskeminen 2D-videon pohjalta pohjautuu *mallintaminen liikkeen avulla* -menetelmään (structure from motion, lyh. SFM). 3D-informaation palauttaminen edellyttää, että ympäristöstä on olemassa useita eri kuvakulmista otettuja näytteitä, toisin sanoen kamera on liikkunut näytteiden välillä. Tämän vuoksi puhutaan SFM-ongelmasta. SFM-ongelma on sovelluskohteena hyvin haastava, joten useimmiten käytännön sovelluksissa 3D-informaatio kerätään aktiivisia menetelmiä hyödyntämällä [2]. Suuria ympäristöjä mallinnettaessa 3D-informaation kerääminen on kuitenkin useimmiten järkevämpi toteuttaa 2D-kuvausta hyödyntämällä. 2D-informaation kerääminen ympäristöstä on monessakin mielessä paljon helpompaa, sillä 3D-informaation mittaaminen vaatii aina erityisen mittalaitteen. Tällaiset mittalaitteet ovat 2D-kameraan verrattuna kalliita ja isokokoisia sekä lisäksi 3D-mittaaminen on 2D-kuvaukseen verrattuna erittäin hidasta. Vaikka 3D-laitteilla mittaaminen voidaan suorittaa tarkemmin, niin silti käytännön syistä suuria ympäristöjä mallinnettaessa päädytään kuitenkin useimmiten käyttämään 2D-kameran liikkeeseen perustuvia toteutuksia.

Mallintaminen liikkeen avulla perustuu kahden keskeisen geometrisen muunnoksen väliseen vuorovaikutukseen [10]: jäykän kappaleen liikkeeseen, mitä käsiteltiin kappaleessa 2.1, sekä perspektiiviseen projektioon, mitä käsiteltiin kappaleessa 2.3. Kahden näkymän tapauksessa yhteisen geometrian selvittäminen tapahtuu fundamentaalisen matriisin avulla, jota käsiteltiin kappaleessa 2.4. Fundamentaalisin matriisin ratkaiseminen pohjautuu kappaleessa 2.2 esitettyyn teoriaan.

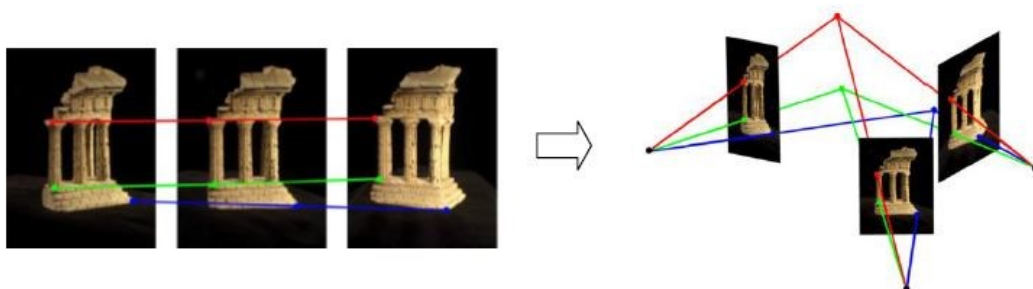
Ympäristön mallintaminen perinteisiä SFM-menetelmiä käyttämällä edellyttää, että kameran sisäiset parametrit ovat tiedossa, toisin sanoen kamera pitää kalibroida. Tämä perusteltiin teoriassa kappaleessa 2.4. Mikäli kameraa ei kalibroida, niin ympäristöstä saadaan rakennettua korkeintaan projektiiivisessä avaruudessa kuvattu malli [13]. Kamera voidaan kalibroida karkeasti kahdella eri menetelmällä: kalibrointi suoritetaan niin kutsutun *kalibrointikappaleen* avulla, jonka geometrinen muoto tunnetaan

tarkasti, tai sitten kamera kalibroidaan pelkästään ympäristöstä otettujen useiden kuvien avulla, jolloin puhutaan *itsekalibroinnista* (self-calibration) [10]. Kalibroitikappaleen käyttäminen edellyttää, että sen muoto saadaan selvitettyä tarkasti etukäteen, joten käytännössä se rajoittaa jossain sovelluksissa tämän menetelmän käyttöä. Toisaalta itsekalibrointi vaatii aina sen, että ympäristöstä tiedetään jotain etukäteen [12].

Kameran kalibroinnin jälkeen SFM-menetelmillä ratkaistaan kamerasen ulkoiset parametrit, jolloin kamerasen koko projektiomatriisi  $\mathbf{P}$  saadaan ratkaistua. SFM-menetelmien perusajatuksena on löytää useammista kuvista vastaavia havaintopisteitä ja laskea 3D-koordinaatit löydetyille vastaavuuksille, sekä ratkaista kameroiden sijainnit näihin pisteisiin suhteutettuna [29]. SFM-algoritmien toiminta voidaan yleensä jaotella karkeasti neljään eri vaiheeseen:

- *Etsitään eri näkyvissä olevat yhteiset 2D-piirteet:* Tässä vaiheessa näkymiä verrataan keskenään ja niistä pyritään löytämään yhteisiä piirrepisteitä, toisin sanoen sellaisia pisteitä, jotka ovat helposti tunnistettavissa jokaisessa näkyvässä. Hyviä piirrepisteitä ovat esimerkiksi kulmat, kappaleiden keskipisteet tai janojen päätepisteet. Piirrepisteiden etsiminen voidaan tehdä esimerkiksi SIFT-algoritmeilla [30].
- *Estimoidaan 3D-sijainnit:* SFM-menetelmällä ratkaistaan alustavat estimaatit kameroiden projektiomatriiseille  $\mathbf{P}_n$  ja piirrepisteille  $\mathbf{X}_m$ , missä  $n$  on kameroiden lukumäärä ja  $m$  on piirrepisteiden lukumäärä.
- *Optimoidaan 3D-estimaatit:* Useimmiten alustavasti laskettuja 3D-estimaatteja joudutaan tarkentamaan, jotta päästään hyväksyttävään lopputulokseen. Tarkentaminen voidaan suorittaa niin kutsutun *blokkitasoituksen* (bundle adjustment) avulla [31]. Blokkitasoitus on epälineaarinen optimoinnin menetelmä, joka minimoi pienimmän neliösumman virhettä. Käytännössä se toteutetaan siten, että näytteiden ensimmäinen kuva kopioidaan myös viimeiseksi kuvaksi, jolloin saadaan ehto  $y(n) = y(1)$ . Tällä tavalla saadaan arvioitua 3D-estimaattien virhettä, jolloin sitä on mahdollista korjata. Virhettä voidaan korjata myös vähitellen laskennan aikana.
- *Pintojen sovittaminen:* Viimeisessä vaiheessa lasketaan 3D-sijainnit myös muillekin ympäristön pisteille, jolloin ympäristöstä saadaan muodostettua yksityiskohtainen 3D-malli. Tämän vaiheen toteuttamiseen käytetään monen näkymän stereo-algoritmeja, joita käsitellään kappaleessa 3.4.

Kuva 4 [17] havainnollistaa SFM-menetelmän toimintaa käytännössä. Kuvassa vasemmalla on kolme näkymää samasta ympäristöstä eri kuvakulmista kuvattuna. Näkymistä on etsitty kolme yhteneväistä piirrepistettä, jotka on merkitty kuvaan. Kuten kappaleessa 2.4 jo todettiin, oikeasti kolme pistettä ei ole riittävä määrä ongelman ratkaisemiseen, mutta riittää kuitenkin SFM-menetelmän visuaaliseen havainnollistamiseen. Piirrepisteiden etsimisen jälkeen niiden 3D-sijainnit estimoidaan, samoin kuin myös kameroiden sijainnit suhteessa näihin pisteisiin. Tätä on havainnollistettu kuvassa oikealla, missä piirrepisteet ja kamerat on sijoitettu 3D-avaruuteen. Tämän jälkeen ympäristön yksityiskohtaisempi 3D-rakenne voidaan tarvittaessa muodostaa monen näkymän stereo-algoritmeilla.



Kuva 4. Vasemmalla puolella on kolme eri kuvakulmista kuvattua näkymää samasta ympäristöstä, joista on löydetty kolme yhteneväistä piirrepiistettä. Tämän jälkeen niiden 3D-sijainnit suhteessa toisiinsa selvitetään SFM-algoritmeilla. Oikeanpuoleisessa kuvassa kamerat ja piirrepiisteet on sijoitettu 3D-avaruuteen.

Muodostetusta 3D-mallista on syytä muistaa se, että se on niin sanottu *metrinen* rekonstruktio ympäristöstä, toisin sanoen 3D-mallin ja fyysisen ympäristön välinen skaalakerroin  $\lambda$  on tuntematon. Parametrin  $\lambda$  selvittäminen edellyttäisi, että ympäristöstä tiedettäisiin jokin absoluuttinen mitta. Ilman tätä absoluuttista mittaa ympäristöstä ei voida rakentaa metristä rekonstruktioita tarkempaa mallia. Matemaattisesti tämä perusteltiin yhtälön (20) avulla.

### 3.4 Monen näkymän stereo

Monen näkymän stereo-menetelmien (multi-view stereo, lyh. MVS) tarkoituksena on muodostaa ympäristöstä yksityiskohtainen 3D-malli käyttämällä useita 2D-näkymiä, joiden sijainnit suhteessa toisiinsa ovat tiedossa. MVS-menetelmät eivät siis sellaisenaan ole käyttökelpoisia liikkuvan robotin avulla toteutettavassa ympäristön 3D-mallintamisessa, sillä liikkuvan robotin tapauksessa kameroiden sijaintia on useimmiten mahdotonta määrittää tarkasti jokaisella ajan hetkellä. Tästä johtuen MVS-menetelmiä käytetään tällaisissa sovelluksissa yhdessä SFM-menetelmien kanssa.

MVS-menetelmiä voidaan erotella toisistaan muun muassa seuraavien ominaisuuksien perusteella [32], [33], [34]:

- *3D-mallin esitystapa*: 3D-malli on mahdollista esittää usealla eri tavalla. Sopiva esitystapa riippuu mallin käyttötarkoituksesta ja mallinnettavan ympäristön rakenteesta. 3D-rakenne voidaan esittää esimerkiksi tasojen, *vokseleiden* (pikselin 3D-vastine, kuutio) tai pistepilven avulla.
- *Näkymien yhteneväisyyksien selvittäminen*: Näkymissä olevat yhteneväiset pisteet voidaan selvittää karkeasti kahdella tavalla: verrataan pisteitä 2D-tasolla tai 3D-avaruudessa. Tasolla tapahtuva vertaaminen tapahtuu siten, että näkymän  $A$  piste ennustetaan tasolle  $B$  ympäristön karkean 3D-rakenteen avulla. Tämän jälkeen tasolla  $B$  voidaan verrata pisteen ennustettua ja mitattua arvoa, jolloin saadaan käsitys mallinnuksen virheestä. Toinen vaihtoehto on palauttaa molemmat pisteet 3D-avaruuteen ja verrata niiden välillä olevaa eroavaisuutta siellä.
- *Näkymien vertaaminen*: MVS-menetelmän on päätettävä, että mitä näkymiä se vertailee keskenään 3D-informaation laskemisessa. Jotkut algoritmit vertailevat kaikki näkymät keskenään, mutta käytännössä tällainen

toimintatapa ei ole toimiva suuria ympäristöjä mallinnettaessa. Tämän vuoksi näkymät jaotellaan pienempiin joukkoihin, joita vertaillaan keskenään. Yleisin tapa jaotella näkymät on käyttää hyväksi SFM-algoritmin tuottaman karkean 3D-rakenteen tarjoamaa informaatiota ympäristöstä.

- *Tunnettujen muotojen hyödyntäminen*: 3D-mallia on mahdollista tarkentaa tunnettujen muotojen avulla. 3D-informaation perusteella voidaan päätellä, että mitä tunnettua muotoa kyseinen kohta mallista esittää, jolloin 3D-informaatio voidaan korjata siten, että se vastaa haluttua muotoa. Tyypillinen esimerkki tunnettujen muotojen hyödyntämisestä on sisätilojen mallintaminen, sillä silloin ympäristön voidaan olettaa sisältävän paljon tasopintoja.
- *3D-mallintamisen algoritmi*: MSV-algoritmit voidaan jakaa karkeasti neljään eri kategoriaan. Ensimmäisen kategorian algoritmit toimivat siten, että ympäristön koko rajoitetaan etukäteen ja tämän jälkeen malli muodostetaan yhdellä kertaa. Toisessa kategoriassa algoritmit muodostavat mallin iteratiivisesti, lisäämällä uusi näkymä aina olemassa olevan mallin jatkeeksi. Kolmannen kategorian algoritmit muodostavat ympäristöstä useita erillisiä malleja, jotka yhdistetään lopuksi yhdeksi malliksi. Neljännessä kategoriassa algoritmit etsivät ympäristöstä sopivat piirrepisteet, joita käytetään mallin geometrisena runkona. Tämän jälkeen näiden piirrepisteiden välille sovitetaan pinnat.
- *Tarvittava alustus*: Kaikki MVS-algoritmit edellyttävät, että niillä on käytössään kalibroituja kuvia, ja että kameroiden sijainnit suhteessa toisiinsa ovat tiedossa. Jotkut algoritmit vaativat myös, että niitä alustetaan etukäteen. Esimerkiksi ympäristön koko pitää joissain menetelmissä alustaa, samoin jotkut menetelmät vaativat, että näkymien etu- ja taka-ala eritellään etukäteen.

Kaikissa MVS-menetelmissä perusajatus on sovelluskohteesta riippumatta kuitenkin sama ja toiminta perustuu samaan teoriaan kuin SFM-menetelmien tapauksessa. Myös SFM-menetelmiä voidaan erotella näiden ominaisuuksien perusteella, joten loppujen lopuksi kyseessä onkin melkein kaksi rinnastettavissa olevaa ongelmaa. Lopullinen erottelu tehdäänkin sen perusteella, että mikä on 3D-mallintamisen tavoite. Mikäli tavoitteena on karkea 3D-malli, niin silloin puhutaan SFM-ongelmasta. Tarkempaa mallia muodostettaessa tarvitaan hieman erilaisia menetelmiä, jolloin puhutaan yleensä MVS-ongelmasta ja mikäli kameroiden sijaintia ei tiedetä, niin silloin SFM-menetelmiä käytetään yhdessä MVS-menetelmien kanssa. Tässä diplomityössä ei perehdytä enää sen tarkemmin erilaisiin MVS-menetelmiin, mutta kiinnostunut lukija voi perehtyä niihin annettujen lähteiden avulla. Esimerkiksi lähteeseen [32] perehtymällä saa hyvän yleiskuvan yleisimmistä MVS-menetelmistä.

### 3.5 Yleiskatsaus olemassa oleviin järjestelmiin

Tässä vaiheessa diplomityössä on hahmoteltu yleiskuva liikkuvan robotin avulla toteutettavasta ympäristön 3D-mallintamisesta, sekä ongelman ratkaisemiseen tarvittava teoria on käsitelty. Järjestelmän toteuttamiseen tarvittavat perusteet ovat siis kunnossa. Ennen kuin siirrytään käsittelemään varsinaista järjestelmän toteuttamista, niin on vielä hyvä ottaa yleiskatsaus olemassa oleviin vastaavanlaisiin järjestelmiin.

Järjestelmän toteutuksen kannalta tärkein huomioitava asia on tarkoitukseen sopivan kuvausrobotin valinta. Tässä diplomityössä kuvausrobotiksi valittiin Parrot-

yhtiön AR Drone 2.0 quadrokopteri, joten perehdytään ensimmäiseksi vastaavalla robotilla toteutettuihin järjestelmiin. Kopteri on edullisen hintansa vuoksi ollut varsin suosittu valinta kuvausrobotiksi, joten toteutettuja järjestelmiä löytyy useita. Yksi varhaisemmista toteutuksista on Krajinikin ym. [35] järjestelmä, joka on toteutettu kopterin 2.0 versiota edeltävällä 1.0 versiolla. Tämä kopteri on ominaisuuksiltaan vaatimattomampi kuin uudempi 2.0 kopteri, mutta järjestelmä on siitä huolimatta suhteellisen toimiva. Järjestelmässä on toteutettu muun muassa objektien seuraaminen, ympäristön mallintaminen ja itsenäinen lentäminen muodostettua mallia hyödyntämällä. Kopterin uudemmalla 2.0 versiolla on toteutettu useita vielä kehittyneempiä järjestelmiä, esimerkkeinä [36], [37] sekä [38]. Kopterin avulla on siis onnistuttu toteuttamaan järjestelmiä, joissa robotin itsenäinen navigoiminen, objektien seuraaminen sekä ympäristön mallintaminen on saavutettu. Nämä varsin onnistuneesti toteutetut järjestelmät osoittavat siis, että kopteri on soveltuva robotti ympäristön automaattiseen mallintamiseen. Aikaisemmin toteutetuissa järjestelmissä ympäristöstä muodostettavalle mallille ei ole kuitenkaan asetettu kovinkaan tiukkoja vaatimuksia, sillä järjestelmissä mallia on käytetty ainoastaan robotin itsenäisen navigoinnin tukena. Tämän vuoksi esimerkiksi Maravall ym. [36] toteuttama järjestelmä muodostaa ympäristöstä ainoastaan topologisen mallin. Tässä diplomityössä tarkoituksena on muodostaa ympäristöstä yksityiskohtaisempi 3D-malli, joten tällöin joudutaan ratkaisemaan uudenlaisia haasteita.

Tarkempia 3D-malleja muodostavia järjestelmiäkin on toteutettu useita. Yksi mielenkiintoinen järjestelmä on Pollefeys ym. [39] toteutus, joka mallintaa erittäin suuria ympäristöjä videokuvan pohjalta. Järjestelmässä ei ole toteutettu automaattista datan keräämistä, vaan video kuvataan ihmisen avustamana. Järjestelmä on toteutettu mallintamalla ympäristö inkrementaalisesti, jolloin on päästy reaaliaikaiseen suoriutumiseen. Yksi järjestelmän mielenkiintoisimmista ominaisuuksista on se, että se käyttää mallinnuksessa apuna satelliittipaikannusta (Global Positioning System, lyh. GPS) ja inertiasuunnistusta (Inertial Navigation System, lyh. INS). Tällä tavalla SFM-vaiheen toteuttamista on mahdollista nopeuttaa ja MVS-vaiheen lopputulosta tarkentaa. Toinen suurten ympäristöjen mallintamiseen keskittynyt järjestelmä on Furukawa ym. [40] toteutus, jossa ympäristö mallinnetaan internetistä ladattujen valokuvien avulla. Tässä järjestelmässä ei ole pyritty reaaliaikaiseen suoriutumiseen. Tällainen järjestelmä on sovelluskohteena videokuvaukseen verrattuna vielä haastavampi, sillä järjestelmään syötettävät kuvat ovat täysin satunnaisia. Videokuvaa käyttämällä sen sijaan tiedetään, että peräkkäiset näytteet ovat hyvin lähellä toisiaan.

Ympäristön 3D-mallintamista 2D-kuvien perusteella on tutkittu myös Oulun yliopistossa [41] [42] [17], mutta Oulun yliopistossa tutkimus on keskittynyt pääasiassa MVS-menetelmien parantamiseen. Tässä diplomityössä tarkoituksena onkin tehdä alustavaa tutkimusta koko järjestelmän toteuttamisesta, datan keräämisestä lähtien 3D-mallin muodostamiseen asti.

Lopuksi on syytä vielä mainita, että kaikista toteutetuista järjestelmistä kenties parhaimpia ovat ongelman ratkaisemiseen keskittyneiden yritysten tuottamat kaupalliset järjestelmät. Eräitä parhaimpia kaupallisia järjestelmiä tarjoavia yrityksiä ovat esimerkiksi Acute3D [43], Ascending Technologies [44] ja 2d3 Sensing [45]. Näiden yritysten tarjoamat järjestelmät tuottavat lähes valokuvantarkkoja 3D-malleja ja vielä siten, että ihmisen ei tarvitse juurikaan puuttua järjestelmän toimintaan. Järjestelmät kykenevät myös reaaliaikaiseen suoriutumiseen. 3D-mallintamista 2D-kuvien pohjalta voidaankin pitää staattisten ympäristöjen tapauksessa käytännössä ratkaistuna ongelmana. Tämän vuoksi tälle diplomityölle onkin turha asettaa sellaisia

tavoitteita, että pyrittäisiin kehittämään nykyisiä järjestelmiä paremmaksi. Sen sijaan tässä diplomityössä on keskitytty etsimään sellaisia ratkaisuja, joilla tarkoitukseen soveltuva järjestelmä voitaisiin toteuttaa riittävän edullisesti. Kaikki edellä mainitut kaupallisia järjestelmiä tuottavat yritykset käyttävät järjestelmissään niin kalliita laitteita ja ohjelmistoja, että tavallisella kuluttajalla ei ole mahdollisuutta hankkia järjestelmiä omaan käyttöönsä. Yritysten tärkeimmät asiakkaat ovatkin näin ollen valtiot ja suuret kansainväliset yritykset.

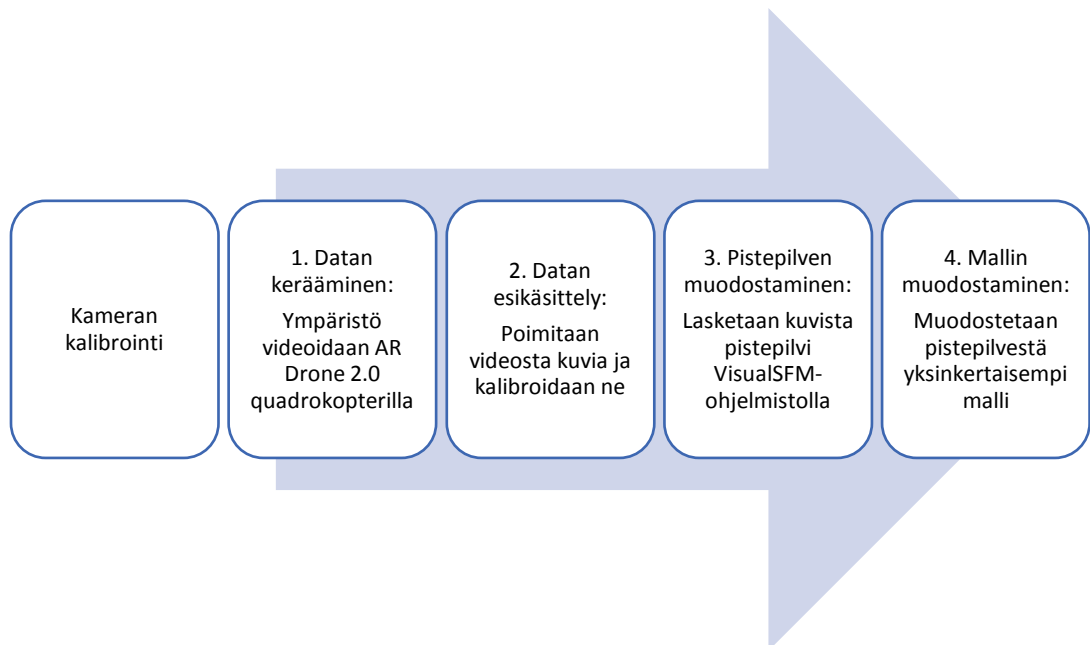
## 4 JÄRJESTELMÄN KUVAUS JA TOTEUTTAMINEN

Ympäristön 3D-mallintamiseen videon avulla on kehitetty jo useita erilaisia järjestelmiä. Hyvien järjestelmien käyttöä rajoittaa kuitenkin se, että ohjelmistot ovat usein kaupallisia ja tarvittava laitteisto erittäin kallista. Tällaiset järjestelmät eivät ole tavallisen kuluttajan saatavilla. Tässä kappaleessa esitellään alustava toteutus sellaiselle järjestelmälle, joka vaatii toimiakseen vain noin 300 euron hintaisen AR Drone 2.0 quadrokopterin sekä tavallisen pöytätietokoneen. Järjestelmä tuottaa ympäristöstä yksinkertaisen 3D-mallin pelkästään kopterilla kuvatun 2D-videon pohjalta. Tällainen järjestelmä on hintansa puolesta myös tavallisen kuluttajan saatavilla. Järjestelmän toteuttaminen perustuu teoriaan, joka käsiteltiin kappaleessa 2 sekä menetelmiin, joita käsiteltiin kappaleessa 3.

Kappaleen rakenne on seuraava: Kappaleessa 4.1 perehdytään järjestelmän toimintaa yleisellä tasolla sekä määritellään järjestelmän suorituskyvyllä asetetut vaatimukset. Kappaleessa perehdytään tarkemmin myös käytettävään AR Drone 2.0 quadrokopteriin. Kappaleessa 3.4.2 käydään läpi järjestelmän toteuttamisen vaiheet.

### 4.1 Järjestelmän kuvaus

Kuva 5 hahmottelee järjestelmän rakennetta. Jaottelu noudattaa karkeasti käänteisen insinööriyön ongelman ratkaisemisen vaiheita.



Kuva 5. Järjestelmän toteutuksen yleiskuvaus.

Ongelman ratkaiseminen edellyttää, että käytössä oleva kamera on kalibroitu. Kameran kalibrointi suoritetaan OpenCV-kirjastoon [9] perustuvalla ohjelmistolla. Kalibroinnin avulla videokuvasta saadaan korjattua kameran aiheuttamat vääristymät, mikä mahdollistaa tarkan 3D-mallin muodostamisen ympäristöstä. Kalibrointivaihe ei varsinaisesti ole osa järjestelmän toimintaa siinä mielessä, että se suoritetaan vain kerran. Se on kuitenkin välttämätöntä suorittaa, että järjestelmän käyttöön saadaan tarvittavat sisäiset kameraparametrit.

Varsinainen järjestelmän toiminta alkaa datan keräämisestä. Datan kerääminen suoritetaan AR Drone 2.0 quadrokopterilla [7], jolla kerätään ympäristöstä lennon aikana HD-tasoista videokuvaa. Toinen vaihe on datan esikäsittely. Ensin videosta poimitaan kuvia Python-ohjelmointikielellä toteutetun ohjelman avulla ja sen jälkeen kuvat kalibroidaan laskettujen kameraparametrien avulla OpenCV:n pohjautuvaa ohjelmistoa käyttämällä. Kolmannessa vaiheessa 2D-videosta muodostetaan 3D-pistepilvi VisualSFM-ohjelmistolla. Ohjelmisto on avoimeen lähdekoodiin pohjautuva useamman toimijan projekti, joka on saatavilla vapaasti internetistä [8]. Neljännessä vaiheessa pistepilvestä muodostetaan yksinkertaisempi 3D-malli Python-ohjelmointikielellä toteutetun ohjelman avulla.

#### **4.1.1 Käyttötarkoitus**

Tätä diplomityötä lähdettiin tekemään alustavasti osana Oulun yliopiston laajempaa tutkimusprojektia, jossa kehitetään monikäyttöistä maassa liikkuvaa robottia. Tätä robottia kutsutaan Mörriksi [46]. Mörrin avulla voidaan suorittaa monenlaisia tehtäviä ympäristössä, sillä robotilla on käsivarsi, jonka avulla se poimia ympäristöstä esineitä, sekä monipuoliset sensorit, joiden avulla se saa ympäristöstä 3D-informaatiota. Mörrille tuntemattomassa ympäristössä toimiessa näiden tehtävien hoitamisesta olisi mahdollista helpottaa, mikäli Mörrillä olisi käytössään 3D-malli ympäristöstä.

Järjestelmää lähdettiin kehittämään sillä ajatuksella, että muodostettu 3D-malli hyödyttäisi Mörrin-robottia sille suunniteltujen tehtävien toteuttamisessa. 3D-mallin avulla Mörrin voisi suunnitella reittejä etukäteen eri paikkojen välillä, sekä välttää sellaisia reittivalintoja, jotka johtavat umpikujaan tai joiden varrella on Mörrille ylitsepääsemättömiä esteitä. 3D-mallin tärkein tehtävä on tuottaa Mörrille suuntaa antavaa informaatiota sellaisesta osasta ympäristöstä, jota Mörrin ei kykene sillä hetkellä itse havaitsemaan.

Ympäristön mallintaminen haluttiin suorittaa lentävällä robotilla, jotta järjestelmällä voidaan mallintaa sujuvasti sellaisia ympäristöjä, joihin Mörrillä ei ole helppoa pääsyä. Ajatus on, että lentävä robotti käy kartoittamassa maaston etukäteen, minkä jälkeen Mörrin voi käyttää mallia omien toimintojensa tukena.

#### **4.1.2 Järjestelmän suorituskyvylle asetetut vaatimukset**

Kappaleessa 3.1 käsiteltiin vaatimuksia, joita tämän kaltaisessa sovelluskohteessa käytettävältä ympäristön 3D-mallilta yleensä edellytetään. Näitä vaatimuksia olivat täydellinen 3D-malli, päivitettävyyden, joustavuus ja tiiviys. Myös tässä käyttötarkoituksessa näiden vaatimusten täytyminen on suotavaa. Ainoastaan mallin tiiviys on käytännössä sellainen asia, minkä suhteen voidaan joustaa. Tällaisessa sovelluskohteessa tiedetään, että ympäristön kokoa rajoittaa robotin rajallinen toimintasäde, joten 3D-malli ei voi kasvaa rajattoman suureksi. Mallin ei tarvitse olla myöskään kovinkaan tarkka, sillä lähiympäristössä toimiessa Mörrin voi käyttää 3D-informaation keräämiseen sen omia sensoreita, joilla se saa mitattua ympäristön 3D-rakenteen paljon tarkemmin.

3D-mallille asetettujen vaatimusten perusteella, sekä käyttötarkoitus huomioiden, järjestelmän toteuttamiselle asetettiin seuraavat reunaehdot:

- *Muodostetun mallin käsittely pitää onnistua reaaliaikaisesti:* Tämän vaatimuksen vuoksi mallin esitystavasta pitää saada sellainen, että sen laskennallinen käsittely on mahdollisimman helppoa.
- *Mallia ei tarvitse luoda reaaliaikaisesti:* Malli voidaan luoda etukäteen, jolloin reaaliaikaisuusvaatimus ei tarvitse toteutua mallia luotaessa. Tämä mahdollistaa monipuolisempien menetelmien käyttämisen 3D-mallia muodostettaessa.
- *Mallin avulla ei tarvitse välttämättä pystyä paikallistamaan robotin sijaintia:* Myöhemmässä vaiheessa paikantamisessa on tarkoitus hyödyntää GPS-koordinaatteja, joten mallin ei tarvitse sisältää kovinkaan yksityiskohtaisia piirteitä. Tärkeintä on, että siitä erottaa vapaan tilan varatusta, jotta Mörrin voi hyödyntää sitä reitin suunnittelemisessa. Käytännössä GPS-koordinaattien hyödyntäminen rajoittaa järjestelmän toimivaksi ainoastaan ulkotiloissa. GPS-paikanninta ei ollut vielä tätä järjestelmää toteutettaessa käytössä, mutta sellainen on kuitenkin saatavilla ja tarkoitus hankkia myöhemmin.
- *Mallin resoluution pitää olla helposti muutettavissa:* Käyttötarkoitukseen parhaiten soveltuva resoluutio ei ole vielä tiedossa, joten sen pitää olla helposti muutettavissa paremmaksi. Lopullinen parhaaseen lopputulokseen johtava resoluutio voidaan selvittää vasta sitten, kun eri resoluutioilla toteutettuja 3D-malleja päästään testaamaan Mörrin avulla.
- *Järjestelmän pitää olla mahdollisimman automaattinen:* Tavoitteena on, että jossain vaiheessa järjestelmä saataisiin toimimaan täysin automaattisesti, siten että lentävä robotti käy itsenäisesti tutkimassa ympäristön ja lähettää 3D-mallin Mörrin käytettäväksi. Ympäristön itsenäinen mallintaminen ei ollut tämän diplomityön aiheena, mutta pidemmällä tähtäimellä tämä vaatimus pitää kuitenkin huomioida. Sen takia käytettävät menetelmät pitää valita siten, että ne mahdollistavat itsenäisesti toimivan järjestelmän toteuttamisen. Sellaiset menetelmät, missä ihminen joutuu manuaalisesti valitsemaan sopivia näytteitä, merkkamaan piirrepiirteitä, tai korjaamaan mallinnuksessa tapahtuneita virheitä, eivät ole hyväksyttäviä.

#### 4.1.3 AR Drone 2.0

Järjestelmän kuvausrobotina käytetään Parrot-yhtiön AR Drone 2.0 quadrokopteria [7]. Kopteri on tähän tarkoitukseen todella edullinen, ainoastaan 300 euron hintainen [47] ja se ostettavissa useista eri verkkokaupoista. Edullisesta hinnasta huolimatta kopteri sisältää HD-tasoisien videokameran ja useita erilaisia sensoreita, jotka mahdollistavat robotin käyttämisen tämän kaltaisissa sovelluksissa. Robotin julkaisemisen jälkeen se onkin ollut varsin suosittu vaihtoehto tämän kaltaisten järjestelmien toteuttamisessa [35].

Kopterin tukirunko on muovia ja siinä on lisäksi hiilikuituisia rakenteita lisätukena. Lentämistä varten kopterissa on neljä moottoria, joista jokaisella on oma virtapiiri moottorin ohjaamista varten, jolloin moottoreita voidaan ohjata toisistaan riippumatta. Lisäksi kopterista löytyy useita erilaisia sensoreita sen asennon ja paikan määrittämiseen, kaksi erilaista kameraa sekä ostettaessa mukana tulee kaksi erilaista runkoa, joista toinen on tarkoitettu käytettäväksi ulko- ja toinen sisätiloissa. Kuva 6 [47] havainnollistaa, että miltä AR Drone 2.0 näyttää ulkokäyttöön tarkoitettun rungon kanssa. Kopteri on suunniteltu ohjattavaksi mobiililaitteella, esimerkiksi

kosketusnäytöllä varustetulla matkapuhelimella tai tabletilla. Ohjaaminen onnistuu kuitenkin tarvittaessa millä tahansa laitteella, jossa on käytössä langaton yhteys. Ohjaamista varten on olemassa viralliset ohjelmistot iOS- ja Android-käyttöjärjestelmille. [47]

AR Drone 2.0 quadrokopterin tärkeimpiä ominaisuuksia ovat [47]:

- Ulkoiset mitat ovat ulkokäyttöön tarkoitetun rungon kanssa 451 millimetriä, sisäkäyttöön tarkoitetun rungon kanssa 517 millimetriä.
- Paino on ulkokäyttöön tarkoitetun rungon kanssa 380 grammaa, sisäkäyttöön tarkoitetun rungon kanssa 420 grammaa.
- Lentoaika 12 minuuttia.
- Eteenpäin suunnattu HD-kamera, 1280x720 resoluutio, 30 kuvaa sekunnissa.
- Alaspäin suunnattu kamera, 320x240 resoluutio, 60 kuvaa sekunnissa.
- Sisältää useita erilaisia sensoreita, kuten ultraäänimittari, ilmanpainemittari, kiihtyvyyssanturi, gyroskooppi ja magnetometri.
- Langaton yhteys.



Kuva 6. AR Drone 2.0 quadrokopteri ulkokäyttöön tarkoitetun rungon kanssa.

Kopterin pieni koko mahdollistaa sen, että sitä voidaan käyttää monenlaisissa ympäristöissä, tarvittaessa jopa sisätiloissa. Tässä työssä keskityttiin kuitenkin mallintamaan ainoastaan ulkona olevia ympäristöjä. Kopterin ohjaamiseen käytettiin Android-laitetta, johon on saatavilla AR.FreeFlight-ohjelmisto [47]. Ohjelmisto tarjoaa rajapinnan kopterin ohjaamiseen ja videon tallentamiseen. Video tallennetaan lennon aikana Android-laitteen muistiin, josta se on helppo siirtää tietokoneelle jatkokäsittelyä varten.

Kopterin ohjaaminen on kohtalaisen helppoa, sillä siinä on käytetty monia ohjaamista helpottavia ratkaisuja. Kopterissa on muun muassa ultraääneen ja ilmanpaineeseen perustuvat korkeusmittarit, joiden avulla kopteri säätää korkeuden automaattisesti. Kopterista löytyy myös kiihtyvyyssanturi, gyroskooppi ja magnetometri, joiden avulla kopteri tunnistaa oman asentonsa ja vakauttaa itsensä automaattisesti. Kopteri onnistuu pysyttelemään itsenäisesti paikallaan jopa kohtalaisen kovassa tuulessakin. Kopterista löytyy eteenpäin osoittavan kameran lisäksi toinen kamera, joka sijaitsee kopterin pohjassa ja kuvaa suoraan alaspäin. Kameran avulla robotti mittaa liikkeen nopeutta, jonka avulla robotti voi tarkentaa omaa liikkumistaan ympäristössä. [7]

## 4.2 Järjestelmän toteuttaminen

Tässä kappaleessa käydään läpi järjestelmän toteuttaminen vaihe vaiheelta. Kappaleen rakenne on seuraava: Kappaleessa 4.2.1 käsitellään kameran kalibrointivaiheen toteutus. Varsinainen järjestelmän toteuttaminen aloitetaan datan keräämisellä, joka käsitellään kappaleessa 4.2.2. Datan esikäsittely käsitellään kappaleessa 4.2.3. Kappaleessa 4.2.4 käsitellään tarvittavan 3D-pistepilven muodostaminen 2D-kuvien pohjalta. Tämän vaiheen toteuttamisessa on käytetty apuna VisualSFM-ohjelmistoa. Viimeisessä vaiheessa pistepilvestä muodostetaan yksinkertaisempi vokseleista rakentuva 3D-malli. Tämän vaiheen toteutus käsitellään kappaleessa 4.2.5

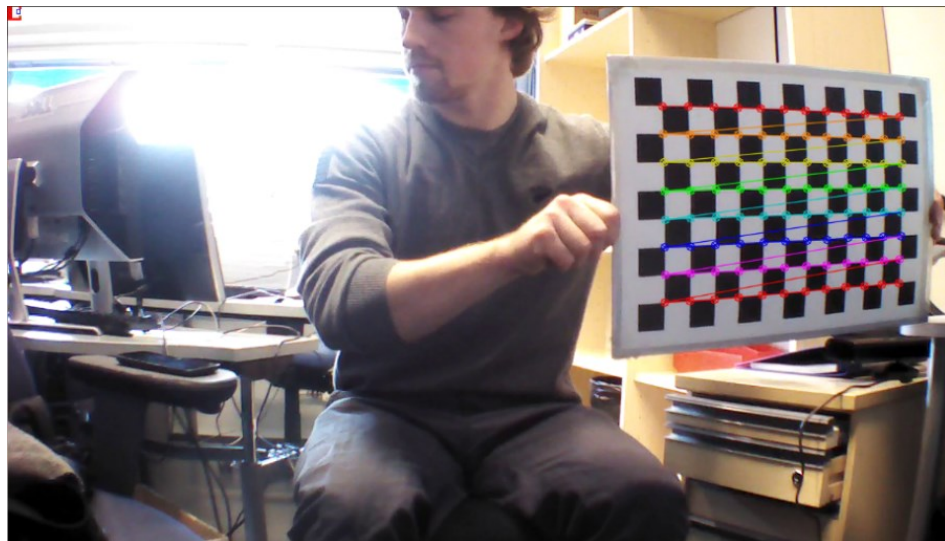
### 4.2.1 Kameran kalibrointi

Kameran kalibrointi toteutetaan OpenCV:seen [9] perustuvalla ohjelmistolla. OpenCV käyttää kameran kalibrointiin hieman poikkeavaa menetelmää verrattuna kappaleessa 2.3 esitettyyn teoriaan. Kameran sisäiset parametrit esitetään yhtälön (18) mukaisella matriisilla  $\mathbf{A}$ , siten että parametri  $\theta = \pi / 2$ . Tämä teoria pohjautuu OpenCV:ssä lähteeseen [19] ja noudattaa kappaleessa 2.3 esitettyä teoriaa. Kamerasta aiheutuneet vääristymät korjataan sen sijaan hieman poikkeavalla tavalla. Nämä parametrit esitetään  $8 \times 1$  -vektorin avulla  $\mathbf{D} = [k_1, k_2, p_1, p_2, k_3, k_4, k_5, k_6]^T$ , missä parametreilla  $k_i$  korjataan radiaalista vääristymää ja parametreilla  $p_j$  korjataan tangentialista vääristymää. Tämä on niin sanottu *rationaalinen kalibrointimalli* ja siihen voi aloittaa perehtymisen tarvittaessa esimerkiksi lähteen [48] avulla. Kameran kalibrointi suoritetaan kiinteän kalibrointikappaleen avulla. OpenCV käyttää kalibrointikappaleena shakkilautaa, jonka ruutujen väliset suhteet ovat järjestelmän tiedossa.

Kalibrointivaiheen toteuttamisessa käytetään apuna Oulun yliopistossa kehitettyä järjestelmää, joka poimii videosta automaattisesti sopivia näytteitä kalibrointiprosessia varten. Järjestelmä on toteutettu C-ohjelmointikielellä ja OpenCV-kirjaston funktioita käyttämällä. Myös omaa Pythonin avulla ohjelmoitua OpenCV:seen pohjautuvaa toteutusta kokeiltiin, mutta lähinnä oppimistarkoituksessa. Oma toteutus toimii muuten vastaavalla tavalla, mutta siinä ei ole automaattista näytteiden ottamista, vaan sopivat näytteet pitää poimia videosta manuaalisesti. Huolellisesti valituilla manuaalisilla näytteillä olisi kenties mahdollista saavuttaa vielä parempi lopputulos, mutta järjestelmästä haluttiin mahdollisimman automaattinen, joten tämän vuoksi päädyttiin käyttämään automatisoitua toteutusta. Automaattisesti poimittujen näytteiden avulla kamera on saatu kalibroituja silmämääräisesti arvioiden erittäin hyvin.

Kalibrointiprosessin tuloksena ratkaistaan sisäinen kameramatriisi  $\mathbf{A}$ , kamerassa olevien vääristymien kertoimet  $\mathbf{D}$ , rotaatio- sekä translaatiovektorit. Kaksi ensimmäiseksi mainittua ovat kameran sisäisiä parametreja ja kaksi viimeisintä ovat kameran ulkoisia parametreja, jotka kuvaavat shakkilaudan asentoa ja paikkaa suhteessa kameraan. Näistä meitä kiinnostavat järjestelmän myöhemmässä vaiheessa vain kameran sisäiset parametrit. Kameran kalibroinnissa ratkaistut  $\mathbf{A}$  ja  $\mathbf{D}$  ovat esitetty liitteessä 1.

Kuva 7 havainnollistaa kalibrointiprosessin suorittamista Oulun yliopistossa kehitetyn OpenCV:seen pohjautuvan järjestelmän avulla. Järjestelmä tunnistaa automaattisesti shakkilaudan ruudut ja kykenee niiden avulla arvioimaan kuvassa olevat vääristymät, koska järjestelmä tietää, että millainen kalibrointikappaleen muoto oikeasti pitäisi olla. Kalibroinnin kannalta on oleellista, että kameran näkymän jokaisesta kohdasta saadaan otettua riittävä määrä näytteitä kalibrointikappaleen kanssa. Kuvan vasemmassa ylälaudassa näkyy pienellä punainen neliö, jonka sisällä näkyy harmaata ja valkoista aluetta. Punainen alue kuvaa koko kameran näkymää, harmaa alue on kalibrointikappale ja valkoinen alue kuvaa niitä alueita, joilta on saatu poimittua kalibrointinäytteitä. Tällainen kalibrointiprosessin visualisointi helpottaa interaktiivista kameran kalibrointia ja varmistaa sen, että jokaisesta kohdasta kameran näkymää saadaan poimittua kunnollinen kalibrointinäyte. Järjestelmä ottaa uuden kalibrointinäytteen aina silloin, kun muutos edelliseen näytteeseen on riittävän suuri, kalibrointikappale on selvästi näkyvässä ja alueelta ei ole olemassa vielä riittävästi käyttökelpoisia kalibrointinäytteitä. Tällä tavalla vältetään turhilta näytteiltä ja näin ollen näytteiden määrä saadaan pidettyä pienenä, mikä nopeuttaa kalibrointi-parametrien laskemista.

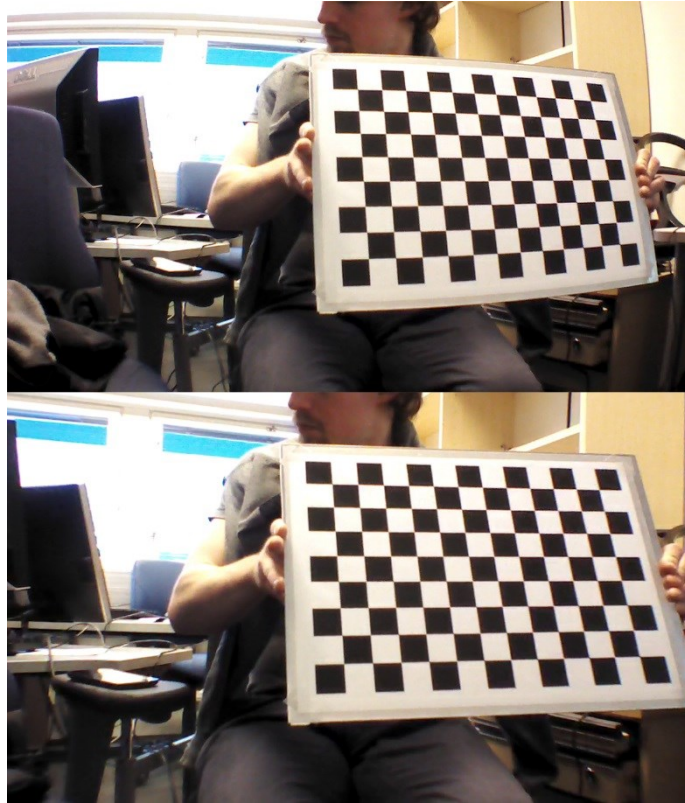


Kuva 7. Kuva kameran kalibrointiprosessin kulusta.

Kuva 8 havainnollistaa kalibroinnin vaikutusta kuvan laatuun. Ylempänä kuvassa on alkuperäinen kalibroimaton kuva. Kuvassa olevat vääristymät ovat erityisen selvästi näkyvässä kuvan reuna-alueilla. Esimerkiksi shakkilaudan alareunassa sekä oikealla olevan kirjahyllyn oikeassa reunassa on nähtävillä selvästi vääristymien aiheuttamaa suorien pintojen kaareutumista. Alempi kuva on kalibroitu laskettujen sisäisten parametrien avulla. Kalibroinnin avulla vääristymät on saatu korjattua, jolloin suorien pintojen kaareutumista ei ole enää havaittavissa. Kalibroinnin seurauksena osa kuvan reunoilla olevasta informaatiosta on menetetty, sillä kalibrointiprosessin jälkeen kuvasta ei tule enää suorakaiteen mallinen, vaan osa pikseleistä kuvautuu suorakaiteen ulkopuolelle. Lopullinen kuva on kuitenkin rajattu suorakaiteen muotoiseksi, jolloin tämän alueen ulkopuolelle kuvautuneet pikselit menetetään.

Kuva 9 havainnollistaa vielä epäonnistunutta kameran kalibrointia. Kuvan tapauksessa kalibrointi on epäonnistunut huonojen kalibrointinäytteiden takia, minkä vuoksi kuvan kalibrointi on johtanut vain alkuperäistäkin kuvaa huonompaan

lopputulokseen. Tällaisen kuvan perusteella ympäristön 3D-informaatiota on mahdotonta määrittää tarkasti, jolloin kelvollista mallia ei saada muodostettua. Kalibroitiprosessin onnistumisella on siis ratkaiseva merkitys tarkan 3D-mallin muodostamisen kannalta.



Kuva 8. Esimerkki kalibroinnin vaikutuksesta kuvan laatuun.



Kuva 9. Esimerkki epäonnistuneesta kalibroinnista.

#### 4.2.2 *Datan kerääminen*

Datan kerääminen toteutetaan lennättämällä AR Drone 2.0 quadrokopteria ympäristössä ja videoimalla ympäristö lennon aikana. Lentäminen suoritetaan tässä vaiheessa vielä manuaalisesti, sillä tämän diplomityön tarkoituksena on keskittyä pääasiassa ympäristön 3D-mallintamiseen eikä robotin itsenäiseen navigointiin. Kopterin ohjaaminen tapahtuu Android-puhelimella, jolle on saatavissa AR.FreeFlight-ohjelmisto kopterin ohjaamista varten. Kuvattu video tallennetaan puhelimen muistiin, josta se siirretään tietokoneelle jatkokäsittelyä varten.

Ympäristön kuvaamista varten käytössä on kaksi AR Drone 2.0 quadrokopteria. Toinen kopteri on täysin alkuperäinen malli, jossa HD-kamera osoittaa suoraan eteenpäin. Kopterin kamera on kiinteästi asennettu, joten sen ohjaaminen lennon aikana ei ole mahdollista. Myöhemmässä vaiheessa kopteria on kuitenkin tarkoitus kehittää niin, että kameraa voidaan ohjata maasta käsin. Suoraan eteenpäin osoittavan kameran avulla ei saada kerättyä tarpeeksi informaatiota maassa olevista kohteista, joten sellaisenaan se ei ole kovin käyttökelpoinen järjestelmän käyttötarkoitusta varten. Tämän vuoksi ympäristö kuvataan myös toisella kopterilla, jossa kamera on säädetty osoittamaan noin 45-asteen kulmassa alaspäin. Kun molemmista videoista poimitaan näytteitä 3D-mallin muodostamista varten, niin informaatiota saadaan kerättyä tarpeeksi sekä seinä- että maapinnoista. Maa-alueen kuvaaminen voitaisiin suorittaa myös kopterin maahan osoittavalla kameralla, mutta se on laadultaan selkeästi huonompi ja toisaalta suoraan maahan kohdistettuja näytteitä on vaikea sijoittaa ympäristöön, sillä niissä ei ole yleensä näkyvillä mitään helposti tunnistettavia rakenteita, kuten seinäpintoja, joiden avulla ne voitaisiin sijoittaa oikeaan paikkaan ympäristössä.

Kuva 10 sisältää neljä peräkkäistä näytettä ympäristöstä, joka on kuvattu alkuperäisellä kopterilla, jossa kamera osoittaa suoraan eteenpäin. Kuvasarja on Oulun yliopiston sisäpihalta, missä on kuvattu kaikki järjestelmän testauksessa käytettävä videomateriaali. Kuvasarjasta auttaa hahmottamaan hieman ympäristön kokoa ja rakennetta.



Kuva 10. Kuvasarja videosta poimituista näytteistä. Ensimmäinen näyte on vasemmalla ylhäällä, viimeinen näyte oikealla alhaalla.

### 4.2.3 Datan esikäsittely

AR Drone 2.0 quadrokopterin kamera ottaa sekunnissa 30 kappaletta HD-tasoisia valokuvia, mikä tarkoittaa pikseleinä ilmoitettuna yli 27 miljoonaa pikseliä sekunnissa. Tällaisen datamäärän käsitteleminen vaatii erittäin paljon laskentatehoa ja data on myös sovelluksen kannalta liian huonolaatuista. Tämän vuoksi kaikkea sensorilta saatavaa dataa ei käytännössä voida käyttää sellaisenaan, joten sitä pitää esikäsitellä ennen varsinaisen raskaamman laskennan aloittamista. Datan esikäsittelyllä on siis kaksi tärkeää tarkoitusta: vähentää käsiteltävän datan määrää ja parantaa sen laatua. Näin päästään nopeampiin suoritusaikoihin ja saadaan myös laadultaan parempi lopputulos.

Tässä toteutuksessa datan esikäsittelyinä tehdään *alinäytteistäminen*, jotta datan määrää saadaan vähemmäksi, sekä kuvan kalibrointi aikaisemmin laskettujen sisäisten kameraparametrien avulla, jotta kuvista saadaan korjattua niissä olevat kamerasta aiheutuneet vääristymät. Alinäytteistämistä on mahdollista tehdä kuvalle kahdella eri tavalla [49]:

- *Spatiaalinen alinäytteistäminen*: Pienennetään kuvan resoluutiota, jolloin yhdessä kuvassa olevan informaation määrä vähenee. Näin voidaan vähentää datan määrää, mutta samalla menetetään datan tarkkuutta, jolloin lopullisen mallin tarkkuus kärsii.
- *Temporaalinen alinäytteistäminen*: Lasketaan näytteenottotaajuutta, jolloin kuvien kokonaismäärä saadaan pienemmäksi. Yksittäisen kuvan tarkkuus pysyy tällöin samana, mutta jokaiselta mahdolliselta ajanhetkeltä ei saada informaatiota, jolloin jotain olennaista informaatiota voi jäädä huomaamatta.

Kuva 11 havainnollistaa spatiaalista alinäytteistämistä. Vasemmanpuoleisimpana on alkuperäinen videosta poimittu kuva ja sen oikealla puolella on sama kuva esitettynä siten, että kuvan resoluutiota on pudotettu kertoimilla kaksi, neljä ja kahdeksan. Kuvasarjasta on selvästi nähtävissä se, että millä tavalla spatiaalinen alinäytteistäminen heikentää kuvan laatua. Oikeanpuoleisimmasta kuvasta alkaa olla jo vaikea erottaa, että mitä kuva esittää, mikäli ympäristöstä ei tiedetä mitään etukäteen. Tämän vuoksi spatiaalista alinäytteistämistä ei voida suorittaa määrättömän paljoa. Rajan asettaa käytännössä se, että kuinka tarkka malli ympäristöstä halutaan saada muodostettua. Mitä yksityiskohtaisempi malli ympäristöstä halutaan rakentaa, sitä vähemmän spatiaalista alinäytteistämistä voidaan suorittaa.



Kuva 11. Esimerkki spatiaalisesta alinäytteistämisestä.

Kuva 12 havainnollistaa puolestaan temporaalista alinäytteistämistä. Ylempänä kuvassa on videosta poimitut kymmenen peräkkäistä kuvaa ympäristöstä. Näille kuville on tehty temporaalinen alinäytteistäminen, joten vain joka neljäs kuva on

hyväksytytty järjestelmän käyttöön. Nämä kuvat ovat nähtävillä kuvassa isommassa koossa alkuperäisten kymmenen kuvan alapuolella. Kuvien koolla ei tässä tapauksessa ole mitään merkitystä, sillä oikeasti kaikki kuvat ovat samankokoisia. Alempana olevat kuvat on esitetty isompana vain, jotta kuvat olisivat selkeämmin tunnistettavissa. Kuvasarjasta näkee, että temporaalisella alinäytteistämällä on menetetty jotain hyödyllistäkin informaatiota. Ensimmäisen kuvan ottamisen jälkeen robotti on heilahtanut tuulen vuoksi, jolloin seuraavat kolme kuvaa on kuvattu rakennuksen kattoa kohti. Tämä on havaittavissa siten, että kuvien yläreunassa näkyy valkoisella taivasta. Robotin heilahdus on kuitenkin kestänyt niin lyhyen ajan, että viidettä kuvaa otettaessa robotti on jo palautunut alkuperäiseen asentoon, eikä kuvannut näin ollen enää rakennuksen kattoa. Temporaalisen alinäytteistämisen vuoksi näytteiden joukosta on poistettu kaikki sellaiset kuvat, joissa rakennuksen kattoa on ollut näkyvissä. Samalla tavalla liian suuren temporaalisen alinäytteistämisen vuoksi voidaan menettää tietoa myös ympäristössä liikkuvista objekteista. Temporaalisen alinäytteistämisen suuruudelle rajan asettaakin käytännössä robotin lentonopeus sekä ympäristön objektien liikkumisnopeus. Staattista ympäristöä kuvattaessa vain robotin lentonopeudella on merkitystä.



Kuva 12. Esimerkki temporaalisesta alinäytteistämisestä.

Alinäytteistämällä on siis se huono puoli, että siinä voidaan menettää myös jotain tarpeellista informaatiota ympäristöstä. 3D-mallin tarkkuuden kannalta olisi tietenkin parasta, mikäli alinäytteistämistä ei suoritettaisi ollenkaan, jolloin kaikkea kerättyä dataa voidaan käyttää 3D-mallin rakentamiseen. Tehokkaalla alinäytteistämällä voidaan kuitenkin merkittävästi nopeuttaa järjestelmän toimintaa, joten se on usein välttämätöntä, jotta järjestelmällä päästään järjellisiin suoritusaikoihin. Mitä enemmän dataa alinäytteistetään, sitä nopeammin data saadaan käsiteltyä ja toisaalta mitä vähemmän alinäytteistämistä tehdään, sitä tarkempi malli ympäristöstä on mahdollista luoda. Käytännössä joudutaan siis käymään vaihtokauppaa tarkkuuden ja nopeuden välillä. Alinäytteistäminen pitääkin suunnitella siten, että sen seurauksena menetetään mahdollisimman vähän järjestelmän suoriutumisen kannalta oleellista informaatiota, mutta päästään kuitenkin järjellisiin suoritusaikoihin.

Datan esikäsittely toteutetaan Python-ohjelmointikielillä ohjelmoidun ohjelman avulla. Ohjelma poimii videosta joka  $n$ :nen näytteen, siten että kun  $n = 1$ , niin jokainen videon kuva käsitellään, kun  $n = 2$ , niin vain joka toinen kuva käsitellään ja niin edelleen.  $n$ :n arvo asetetaan ohjelman parametriksi, joten tällä tavalla voidaan säätää temporaalisen alinäytteistämisen taajuutta. Järkevä näytteistystaajuus riippuu robotin lento- ja pyörähtämisnopeudesta. AR Drone 2.0:ssa lento- ja pyörähtämisnopeutta on mahdollista rajoittaa, jolloin näytteistystaajuus voidaan asettaa suurimpien mahdollisten lento- ja pyörähtämisnopeuksien mukaan. Temporaalisen alinäytteistämisen jälkeen jäljelle jääneet kuvat kalibroidaan

aikaisemmin laskettujen kameran sisäisten parametrien avulla, jolloin kuvassa olevat kameran aiheuttamat vääristymät saadaan korjattua. Lopuksi kuvalle voidaan suorittaa spatiaalinen alinäytteistäminen. Ohjelmalle voidaan antaa parametrina spatiaalisen alinäytteistämisen kerroin  $m$ , siten että kun  $m = 1$  niin kuva säilyy alkuperäisenä, kun  $m = 2$ , niin kuvan resoluutiota pudotetaan kertoimella kaksi ja niin edelleen. Tämän jälkeen kuvat on esikäsitelty ja valmiita käytettäväksi pistepilven muodostamista varten.

#### 4.2.4 Pistepilven muodostaminen

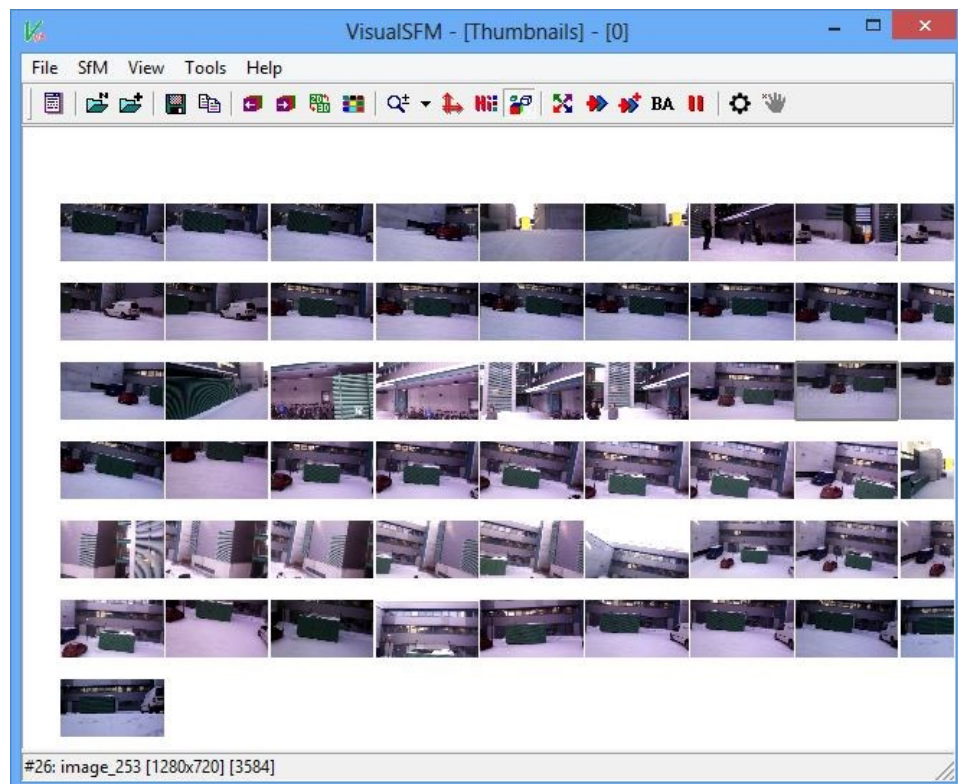
Kuvien esikäsitteilyn jälkeen niiden pohjalta muodostetaan 3D-malli ympäristöstä. Ympäristön 3D-mallintaminen pelkästään 2D-kuvissa olevaa informaatiota hyödyntämällä on erittäin haastava sovelluskohde ohjelmoitavaksi, joten tämän vuoksi tätä järjestelmän osa-aluetta ei ole pyritty ohjelmoimaan itse, vaan sen sijaan on etsitty ongelman ratkaisemiseen soveltuva ohjelmisto jo olemassa olevien ohjelmistojen joukosta. Sopivaa ohjelmistoa etsittäessä kokeiltiin useita erilaisia internetistä ladattavissa olevia ohjelmistoja, esimerkiksi VisualSFM [8], Autodesk 123D [50], 3DF Samantha [51] ja Bundler [52]. Kaikki edellä mainitut ohjelmistot ovat sellaisia, että ne ottavat sisääntulona joukon samaa ympäristöä esittäviä kuvia ja pyrkivät muodostamaan niistä jonkinlaisen geometrisen mallin.

Tämän vaiheen toteuttamisessa päädyttiin käyttämään lopulta VisualSFM-ohjelmistoa. VisualSFM:n käyttäminen on selkeän käyttöliittymän ansiosta erittäin helppoa ja se muodostaa ympäristöstä pistepilven, jonka muokkaaminen meidän käyttötarkoitukseen sopivaksi 3D-malliksi onnistuu helpommin, kuin esimerkiksi pintoina esitettyjen 3D-mallien muokkaaminen. Internetistä katsottujen esimerkkien perusteella VisualSFM:n avulla on mahdollista muodostaa erittäin tarkkoja pistepilviä ympäristöstä ja myös omissa testauksissa sen avulla saavutettiin parempi lopputulos, kuin muilla testatuilla ohjelmistoilla.

VisualSFM on avoimen lähdekoodin projekti, joka koostuu useammasta osasta ja jota on ollut toteuttamassa useampi eri tekijä. VisualSFM:n toiminnan pohjalla oleva teoria on selitetty tarkemmin lähteessä [53] ja lisäksi aiheeseen liittyvää tutkimusta löytyy lähteistä [54], [55] ja [56]. Itse ohjelma on vapaasti ladattavissa internetistä [8]. VisualSFM ei kykene sellaisenaan muodostamaan kovinkaan yksityiskohtaista pistepilveä ympäristöstä. Käytännössä ohjelmisto laskee 3D-koordinaatit ainoastaan löydetyille piirrepiisteille, jolloin se tuottaa hyvin karkean pistepilven ympäristöstä. Ohjelma tarjoaa kuitenkin rajapinnan käyttäjä Yasutaka Furukawan kehittämää PMVS/CMVS-ohjelmistoa [57], joka kykenee laskemaan 3D-koordinaatit myös muille ympäristön pisteille, jolloin ympäristöstä saadaan muodostettua tiheämpi pistepilvi. PMVS/CMVS-ohjelmiston toiminnan pohjalla oleva teoria on selitetty lähteissä [40] ja [58]. Lähteessä [59] on vertailtu Furukawan algoritmin suoriutumista muihin MVS-algoritmeihin nähden ja se pärjää vertailussa erittäin hyvin, joten tämä osaltaan vahvistaa sitä käsitystä, että VisualSFM yhdessä Furukawan PMVS/CMVS-algoritmin kanssa on hyvä ohjelmisto SFM- ja MVS-ongelman ratkaisemiseen.

Kuva 13 havainnollistaa VisualSFM-ohjelmiston graafista käyttöliittymää. Ohjelmiston tärkeimpien perustoimintojen käyttäminen onnistuu ylhäällä olevan tekstivalikon alapuolelta löytyvillä painikkeilla. Ohjelmiston käyttö jakaantuu yksinkertaisimmillaan kappaleessa 3.3 esitettyihin neljään SFM-algoritmin vaiheisiin. Ensin ohjelmistolla avataan kuvat, joiden pohjalta malli halutaan muodostaa. Kuvan

esimerkissä ohjelmistolla on avattu 63 kuvaa, joista osa näkyy pieninä kuvakkeina ohjelmiston työpöydällä. Kuvien avaamisen jälkeen ensimmäinen vaihe on etsiä kuvista yhteneväiset piirrepiisteet. Tämä vaihe toteutetaan painamalla painiketta, jossa on neljä eriväristä nuolta osoittamassa kaikkiin eri väli-ilmansuuntiin. Ohjelmisto etsii jokaisesta kuvasta piirrepiisteet ja vertaa sen jälkeen kuvia keskenään, aina kahta kuvaa kerrallaan, jolloin sille muodostuu käsitys siitä, että missä kuvissa on yhteisiä piirrepiisteitä ja tällöin ne esittävät samaa ympäristöä. Tämän jälkeen piirrepiisteille estimoidaan 3D-sijainnit ja kameroille estimoidaan ulkoiset parametrit. Tämä tapahtuu painamalla kuvaketta, jossa kaksi eriväristä nuolta osoittaa oikealle. Karkea pistepilvi ympäristön 3D-rakenteesta tulee nähtäville ohjelman työpöydälle, kuten myös kameroiden sijainnit suhteessa ympäristöön. 3D-estimoinnin jälkeen 3D-pisteille suoritetaan blokkitasoitus (bundle adjustment) painamalla BA-painiketta, jolloin 3D-estimaatteja saadaan tarkennettua. Viimeisessä vaiheessa muodostetaan tiheä pistepilvi käyttämällä hyväksi Furukawan PMVS/CMVS-ohjelmistoa. Tämä voidaan tehdä painamalla BA-painikkeen vieressä olevaa CMVS-painiketta. CMVS-painike ei ole nähtävillä kuvassa, mutta se tulee näkyviin heti 3D-estimaattien laskemisen jälkeen. Tiheä pistepilvi tallennetaan PLY-tiedostoon, mistä se voidaan avata PLY-formaattia tukevan ohjelmiston avulla. Tiedoston formaatti on esitetty liitteessä 2.

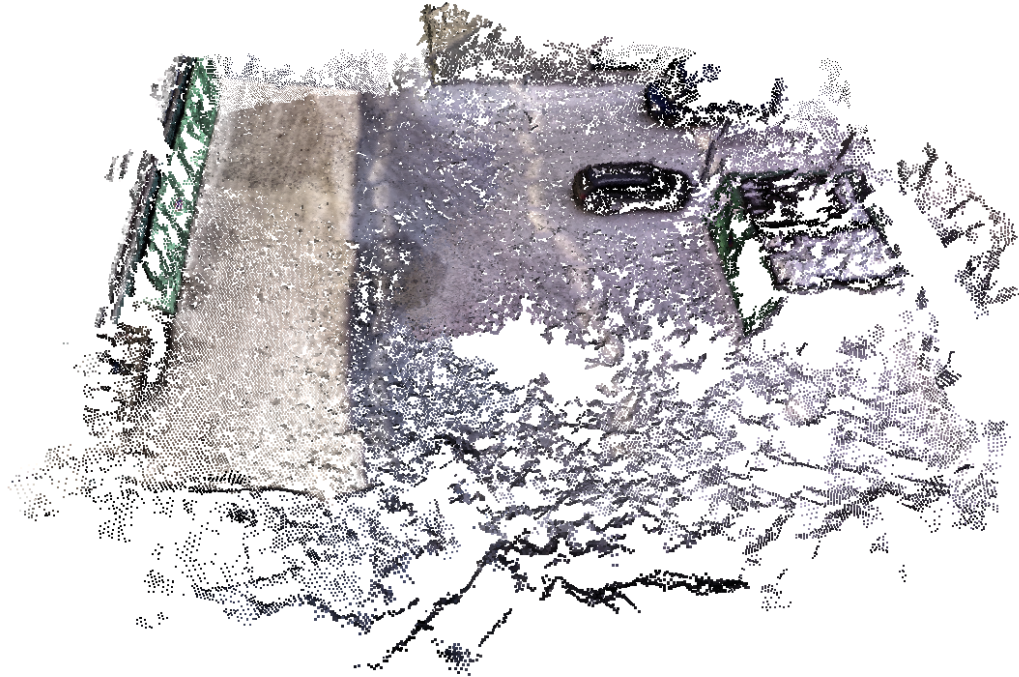


Kuva 13. VisualSFM-ohjelmiston graafinen käyttöliittymä. Ohjelman työpöydällä näkyy videosta poimittuja kuvia.

Ohjelmiston peruskäyttäminen on siis erittäin helppoa ja tapahtuu yksinkertaisimmillaan neljää painiketta painamalla. Ohjelmistosta löytyy kuitenkin paljon monipuolisempia ja monimutkaisempia toimintoja vaativampaan käyttöön. Ohjelmiston toimintaa ja käyttöä on selitetty tarkemmin lähteessä [8].

Kuva 14 esittää FisualSFM-ohjelmistolla muodostettua pistepilveä ympäristöstä. Tiheän pistepilven muodostamiseksi on käytetty PMVS/CMVS-ohjelmistoa

VisualSFM:n tarjoaman rajapinnan kautta. Pistepilven muodostamiseksi videosta on poimittu 500 näytettä, jotka on annettu VisualSFM:n käsiteltäväksi. Muodostetussa pistepilvessä on yhteensä 131 407 erillistä 3D-pistettä.



Kuva 14. Ympäristöstä muodostettu pistepilvi 2D-videosta poimittujen kuvien pohjalta.

#### 4.2.5 Mallin luominen

Järjestelmän viimeisessä vaiheessa pistepilvestä muodostetaan yksinkertaisempi, laskennallisesti kevyempi 3D-malli. Pistepilvi ei sellaisenaan ole sopiva esitysmuoto 3D-mallille, sillä jo suhteellisen pienestäkin ympäristöstä muodostettu pistepilvi sisältää valtavan määrän dataa. Esimerkiksi edellisen kappaleen pistepilvessä (Kuva 14) on yli 131 000 pistettä ja se on muodostettu ainoastaan noin 20x30 metrin kokoiselta sisäpihalta. Tällaisen mallin tallentaminen vaatii noin 4 megatavua tilaa ja sen laskennallinen käsittely on erittäin hidasta. Pistepilvi ei myöskään sisällä informaatiota ympäristön jokaisesta kohdasta, sillä pisteiden välissä on aina tyhjää tilaa. Tämän vuoksi ympäristöstä pitää muodostaa paremmin suunniteltuun käyttötarkoitukseen sopiva malli.

Kappaleissa 3.1 ja 4.1.2 käsiteltiin muodostettavalle 3D-mallille asetettuja vaatimuksia ja reunaehtoja. Näiden pohjalta päädyttiin muodostamaan malli, jossa ympäristö esitetään samankokoisten vokseleiden avulla. Tällainen menetelmä on yleinen 3D-ympäristöjä mallinnettaessa ja sen pääasiallinen heikkous on se, että mallin tallentaminen vie paljon tilaa muistissa [26]. Muuten tällainen malli on hyvin käyttökelpoinen ja ympäristön jakaminen samankokoisiin diskreetteihin osiin tekee mallin käsittelemisestä helppoa verrattuna jatkuvista pinnoista muodostettuihin malleihin. Tarkastellaan seuraavaksi, että miten malli onnistuu vastaamaan kappaleessa 3.1 esitettyihin vaatimuksiin ja miten se soveltuu suunniteltuun käyttötarkoitukseen:

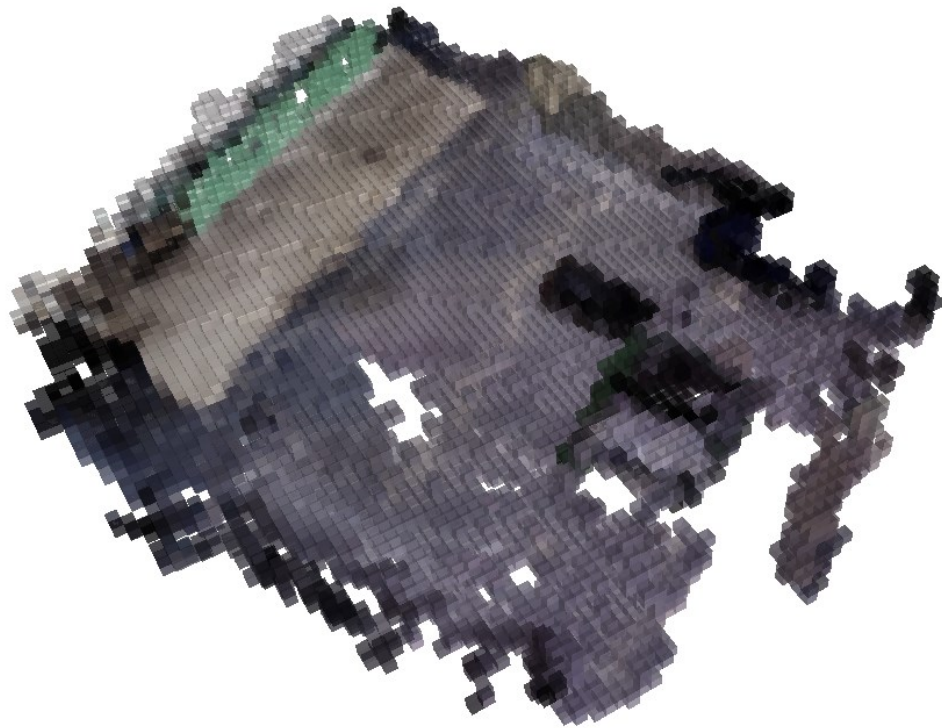
- *Täydellinen 3D-malli:* Mallin avulla voidaan esittää mikä tahansa ympäristö eikä ympäristön rakenteesta tarvitse tehdä mitään oletuksia etukäteen. Ympäristö voidaan esittää riittävällä tarkkuudella, mikäli resoluutio säädetään riittävän isoksi. Vapaiden ja varattujen alueiden erottaminen on helppoa: mallissa on käytössä erillinen parametri, joka ilmoittaa, että onko kyseinen kohta ympäristöstä vapaa vai varattu. Tätä parametria voi käyttää myös ilmaisemaan, että mikä osa ympäristöstä on vielä tutkimatta, mutta sitä ei ole ehditty tässä järjestelmässä vielä toteuttamaan.
- *Päivitettävyyys:* Mallin päivittäminen on erittäin helppoa. Vokselin sijaintia on helppo muuttaa, mikäli sijaintia saadaan tarkennettua, vokseli on helppo poistaa mallista tarvittaessa ja malliin on helppo lisätä uusia vokseleita, mikäli ympäristöstä saadaan kerättyä uutta informaatiota. Vokselit ovat toisistaan riippumattomia, joten mallia päivitettäessä voidaan käsitellä yhtä vokselia kerrallaan.
- *Joustavuus:* Malli skaalautuu helposti erikokoisiin ympäristöihin, samoin resoluutiota on helppo muuttaa. Tällainen malli mahdollistaa myös ympäristön eri osien esittämisen eri tarkkuuksilla. Vokselin tilavuus on helppo jakaa esimerkiksi kahdeksaan yhtä suureen osaan ja edelleen jokainen pienempi osa kahdeksaan yhtä suureen osaan, jolloin osa ympäristöstä voidaan esittää hyvin karkealla tarkkuudella ja tärkeämmät, tarkkuutta vaativat, osat ympäristöstä saadaan esitettyä tarvittaessa erittäin tarkasti.
- *Tiivis:* Malli ei ole matemaattisiin esitysmuotoihin verrattuna erityisen tiivis. Mikäli ympäristö esitettäisiin esimerkiksi matemaattisina kappaleina ja pintoina, niin malli saataisiin tallennettua huomattavasti pienempään tilaan. Tällaisen mallin käsittely on kuitenkin monimutkaisempaa, joten tämän vuoksi päädyttiin käyttämään yksinkertaisempaa mallia. Tähän käyttötarkoitukseen tällainen malli on kuitenkin riittävän tiivis, sillä robotin toimintasäde rajoittaa mallinnettavan ympäristön kokoa ja tällöin tilankäyttö ei muodostu todelliseksi ongelmaksi.

Tällainen vokseleista muodostuva malli vaikuttaisi siis olevan erittäin käyttökelpoinen vaihtoehto tämänkaltaisessa sovelluksessa. Mallin käyttämisestä aiheutuu kuitenkin yksi todellinen ongelma, joka pitää ratkaista jollain tavalla: ympäristön koosta on tiedettävä ainakin jotain etukäteen, sillä muuten vokselin koko, toisin sanoen resoluutio, on mahdotonta valita järkevällä tavalla. Tässä vaiheessa ongelma ratkaistiin siten, että ympäristön koko tiedettiin etukäteen, jolloin resoluutio voitiin asettaa sen tiedon perusteella järkeväksi. Ongelma on mahdollista ratkaista myös silloin, jos ympäristöstä tiedetään jotain mittoja, minkä suhteen vokseleiden kokoa voidaan suhteuttaa. Tällä tavalla saataisiin myös parannettua malli metrisestä rekonstruktioista euklidiseen rekonstruktioon. Jos järjestelmä kykenisi tunnistamaan esimerkiksi autoja tai liikennemerkkejä ja tietäisi niiden absoluuttiset mitat, niin silloin ympäristön koko olisi mahdollista selvittää. Koko voitaisiin selvittää myös silloin, mikäli kameroiden paikoista saataisiin jotain tarkkaa tietoa, esimerkiksi GPS-paikannuksen avulla. Nyt kameroiden paikat on selvitetty ainoastaan suhteessa ympäristöön ja muihin kameroihin

Yksinkertaistetun mallin luominen on toteutettu Python-ohjelmointikielellä ohjelmoidulla ohjelmalla. Mallin luominen on yksinkertaista ja suoraviivaista. Ympäristö jaotellaan samankokoisiin kuutioihin, joiden kokoa voidaan helposti muuttaa tarvittavan resoluution saavuttamiseksi. Kuution sisältämien pisteiden

perusteella tehdään päätös siitä, että sisältääkö kuutio ympäristön kannalta niin oleellista informaatiota, että se kannattaa lisätä lopulliseen malliin. Ohjelmaan syötetään erillinen kynnyisarvo, joka määrittelee sen, että kuinka monta pistettä lopulliseen malliin lisättävän kuution tulee sisältää. Mikäli kuution sisällä on vähemmän pisteitä kuin asetettu kynnyisarvo, kuution sisältämien pisteiden oletetaan olevan vain satunnaista kohinaa ja näin ollen sitä ei lisätä lopulliseen malliin. Mikäli kynnyisarvo ylittyy, kuution sisällä olevista pisteistä lasketaan niiden keskimääräinen väriarvo, joka asetetaan lopulliseen malliin lisättävän kuution arvoksi. Kynnyisarvoa käytetään sen vuoksi, että sen avulla voidaan hallita kohinan aiheuttamia häiriöitä. Ohjelma tallentaa lopullisen 3D-mallin tekstitiedostoon, jonka formaatti on esitetty liitteessä 3. Ohjelman toiminta on selitetty yksityiskohtaisemmin liitteessä 4.

Mallin visualisointiin tarvitaan ohjelma, joka ymmärtää liitteessä 3 esitettyä tiedostoformaattia. Visualisointia varten on tehty yksinkertainen ohjelma Python-ohjelmointikielellä käyttämällä apuna VPython-kirjastoa. Kuva 15 esittää resoluutiolla 20 ja kynnyisarvolla 3 muodostettua 3D-mallia ympäristöstä. Malli sisältää yhteensä 6430 näkyvää kuutiota. Suurin osa ympäristön kuutioista on tyhjiä, joten niitä ei ole tässä tapauksessa tallennettu ympäristöstä muodostettuun malliin. Nekin on mahdollista lisätä tarvittaessa malliin mukaan, mutta tällöin mallin tallentaminen vie enemmän tilaa.



Kuva 15. Ympäristöstä muodostettu pistepilveä yksinkertaisempi malli, joka koostuu samankokoisista kuutioista eli vokseleista. Malli on muodostettu edellisessä kappaleessa esitetyn pistepilven (Kuva 14) pohjalta. Lopullisessa mallissa on yhteensä 6430 kuutiota alkuperäisen 131 407 pisteen sijaan.

## 5 JÄRJESTELMÄN TESTAUS

Edellisessä kappaleessa esiteltiin järjestelmä, jonka avulla ympäristöstä on mahdollista muodostaa 3D-malli pelkästään edullisella quadrokopterilla kuvatun videon pohjalta. Järjestelmän käyttäminen edellyttää sen, että temporaaliselle ja spatiaaliselle alinäytteistämiseksi asetetaan sovelluskohteen kannalta järkevät arvot, samoin kuin myös resoluutiolle ja kynnysarvolle. Tässä kappaleessa tutkitaan sitä, että millä tavalla alinäytteistäminen kannattaa käytännössä toteuttaa. Alinäytteistämällä voidaan nopeuttaa järjestelmän toimintaa, mutta sillä on aina se haittapuoli, että siinä menetetään informaatiota. Yksi testattava asia on se, että miten kuvien määrän pudottaminen vaikuttaa mallin tarkkuuteen ja sen muodostamiseen tarvittavaan aikaan. Toinen kiinnostava asia testattavaksi on se, että onko HD-tasoisien kuvien käyttäminen ylipäänsä edes järkevää vai pärjättäisiinkö mahdollisesti epätarkemmalla kuvalla. Muodostettavan mallin ei kuitenkaan tarvitse olla kovinkaan tarkka, joten ajatuksena on selvittää, että sisältääkö HD-tasoinen kuva liian paljon tarpeetonta informaatiota. Kappaleessa tutkitaan myös järjestelmään asetettavien resoluutio- ja kynnysarvo-parametrien vaikutusta lopullisen 3D-mallin tarkkuuteen.

Järjestelmän testaamista varten Oulun yliopiston sisäpihalta videoitiin alue, joka on suuruudeltaan noin 20x30 metriä. Tällainen ympäristö soveltuu hyvin järjestelmän alustavaan testaukseen, sillä ympäristö on kooltaan rajoitettu, se voidaan olettaa kuvaushetkellä staattiseksi ja se sisältää paljon helposti tunnistettavia rakenteita, kuten ikkunoita, ovia ja seinäpintoja. Ympäristö kuvattiin molemmilla käytössä olleilla koptereilla. Järjestelmää testattiin tavallisella pöytäkoneella, jossa on Intel Core ULV 13-2365M 1,4 gigahertsin suoritin.

Kappaleen rakenne on seuraava: Kuvien määrän vaikutus pistepilven tarkkuuteen käsitellään kappaleessa 5.1 ja kuvien resoluution vaikutus pistepilven tarkkuuteen käsitellään kappaleessa 5.2. Kappaleessa 5.3 tutkitaan kynnysarvo- ja resoluutioparametrien vaikutusta, kun pistepilvestä muodostetaan vokseleista rakentuvaa 3D-mallia.

### 5.1 Kuvien määrän vaikutus

Intuitiivisesti saattaisi ajatella, että pistepilven tarkkuutta voidaan parantaa merkittävästi kuvien määrää nostamalla. Tosiasiassa kuvien määrällä on kuitenkin hyvin vähän vaikutusta pistepilven tarkkuuteen. Monen näkymän geometrian käyttö ei nimittäin tarvitse teoriassa kuin kaksi eri kuvakulmista kuvattua näkymää samasta osasta ympäristöä, joiden perusteella on jo mahdollista laskea ympäristön 3D-informaatio. Kohinan vuoksi käytännössä joudutaan käyttämään useampia kuvia, sillä useamman kuvan avulla on mahdollista parantaa tarkkuutta ja näin ollen vähentää kohinan vaikutusta. Kameroiden paikat ja 3D-pisteiden sijainnit vakiintuvat normaalisti kuitenkin nopeasti [53], joten kovinkaan montaa näkymää samasta kohtaa ympäristöä ei tarvita.

Kuvien määrän vaikutusta pistepilven tarkkuuteen testattiin sillä tavalla, että samasta ympäristöstä muodostettiin useampia pistepilviä eri määrää kuvia käyttämällä, jonka jälkeen muodostettuja pistepilviä vertailtiin silmämääräisesti toisiinsa. Useimmissa tapauksissa saavutettiin silmämääräisesti arvioituna parempi tulos noin 50:llä manuaalisesti valitulla kuvalla verrattuna siihen, että videosta poimittiin kuvia automaattisesti jopa yli kymmenkertainen määrä. Kuvien määrän

lisäämisen positiivinen vaikutus tuli esille vasta silloin, mikäli vähäisetkin kuvat poimittiin videosta automaattisesti. Tällöin kasvoi todennäköisyys sille, että kaikista osista ympäristöä ei saatu poimittua kunnollisia kuvia, jolloin pistepilven tarkkuus kärsi. Kuva 16 havainnollistaa tällaista tilannetta. Kuvassa on kaksi samasta ympäristöstä muodostettua pistepilveä, joista molemmat on muodostettu automaattisesti videosta poimittujen kuvien avulla. Mallinnettava ympäristö on ollut Oulun yliopiston sisäpiha, joka on pistepilvissä kuvattu ylhäältä päin, jotta erot saadaan selkeämmin esille. Vasemmanpuoleinen pistepilvi on muodostettu 61 kuvan avulla ja oikeanpuoleinen pistepilvi 629 kuvan avulla. Kuvasta voidaan havaita, että useamman kuvan avulla on saatu tässä tapauksessa muodostettua malli isommalta alalta ympäristöä, sillä vähäisellä kuvien määrällä ei ole saatu kerättyä tarpeeksi informaatiota ympäristön alaosan mallintamista varten. Sen sijaan se osa ympäristöstä, mikä saatiin mallinnettua myös pienemmällä kuvien määrällä, ei ole tarkentunut merkittävästi kuvien määrää lisäämällä. Paikoitellen kuvien määrän lisääminen on jopa huonontanut lopputulosta..



Kuva 16. Kuvassa on kaksi samasta ympäristöstä muodostettua pistepilveä ylhäältä päin kuvattuna. Vasemmanpuoleisen pistepilven muodostamisessa on käytetty 61 kuvaa ja oikeanpuoleisen pistepilven muodostamisessa 629 kuvaa.

Kuvien määrää nostamalla ei voida siis vaikuttaa merkittävästi pistepilven tarkkuuteen. Sen sijaan mallin muodostamiseen tarvittavaan aikaan sillä on erittäin suuri merkitys. VisualSFM:n laskenta-aika on usein käytännössä lähes lineaarisesti verrannollinen kuvien määrään eli sen aikakompleksisuus on parhaimmillaan  $O(n)$ . Teoriassa VisualSFM:n aikakompleksisuus on kuitenkin  $O(n^2)$  ja ohjelmisto voi pahimmillaan toimia tällä tavalla [53]. Tämä selittää pistepilvien muodostamiseen käytettyä aikaa. Vasemman puoleisen pistepilven (Kuva 16) muodostamiseen meni tavallisella 1,4 gigahertsin suorittimella varustetulla pöytäkoneella aikaa yhteensä hieman yli 20 minuuttia. Eniten aikaa meni piirrepisteiden etsimiseen ja vertaamiseen, noin 14 minuuttia. 3D-informaation estimointi, blokkitasoitus ja tiheän pistepilven muodostaminen veivät yhteensä hieman yli 6 minuuttia. Oikeanpuoleista pistepilveä muodostettaessa pelkästään piirrepisteiden etsiminen ja vertaaminen vei 854 minuuttia! 3D-informaation estimointi vei aikaa noin 40 minuuttia, blokkitasoitus 13 minuuttia ja tiheän pistepilven muodostaminen 47 minuuttia. Kokonaisaika oli siis lähes 16 tuntia, jolloin noin 10-kertaisella kuvien määrällä jouduttiin käyttämään lähes 50-kertainen määrä aikaa.

Kuvien määrän vaikutusta mallin tarkkuuteen eri MVS-menettelmillä on testattu lähteessä [59]. Tähän lähteeseen perehtymällä saadaan vahvistusta sille havainnolle,

että kuvien määrällä on hyvin vähän merkitystä lopullisen mallin tarkkuuteen. Sivustolla on testattu myös tässä diplomityössä käytetty Furukawan PMVS/CMVS-algoritmi. MVS-algoritmien testaaminen perustuu lähteessä [32] esitettyyn menetelmään. Taulukko 2 [59] sisältää Furukawan eri MVS-algoritmien testaustuloksia. Tässä järjestelmässä käytettävä MVS-algoritmi esiintyy taulukossa nimellä Furukawa3. Algoritmeilla on mallinnettu kaksi eri 3D-objektia, temppeleli ja dino, joiden malleja on sen jälkeen verrattu referenssimalliin, jolloin saadaan käsitys mallinnuksessa tapahtuneesta virheestä. Taulukon ensimmäinen numero  $0,xx$  mittaa mallin tarkkuutta siten, että 90 % mallinnetuista pisteistä on etäisyyden  $d$  sisällä referenssimallista, missä  $d$  mitataan millimetreinä. Taulukon toinen numero  $x,xx$  % mittaa mallin täydellisyyttä siten, että tämä prosenttiosuus mallin pisteistä on 1,25 millimetrin sisällä referenssimallista. Taulukosta on luettavissa, että jo 16 kuvan avulla voidaan muodostaa kohtuullisen tarkka malli objektista. Kun kuvien määrää nostetaan 47 tai 48 kuvaan, niin mallin tarkkuus paranee jonkin verran. Kun siirrytään käyttämään yli 300 kuvaa, niin malli ei enää tarkennu merkittävästi ja jossain tapauksissa tarkkuus jopa heikentyy. Taulukossa esitetty informaatio tukee omia havaintoja kuvien määrän vaikutuksesta. Lähteessä [59] voi perehtyä halutessaan eri kuvamäärillä muodostettujen mallien eroavaisuuksiin, jolloin tuloksen voi hahmottaa myös visuaalisesti.

Taulukko 2. Kuvien määrän vaikutus pistepilven tarkkuuteen

	mm	%	mm	%	mm	%
<b>Temppeleli</b>	<b>312 kuvaa</b>		<b>47 kuvaa</b>		<b>16 kuvaa</b>	
Furukawa	0,65	98,7	0,58	98,5	0,82	94,3
Furukawa2	0,54	99,3	0,55	99,1	0,62	99,2
Furukawa3	0,49	99,6	0,47	99,6	0,63	99,3
<b>Dino</b>	<b>363 kuvaa</b>		<b>48 kuvaa</b>		<b>16 kuvaa</b>	
Furukawa	0,52	99,2	0,42	98,8	0,58	96,9
Furukawa2	0,32	99,9	0,33	99,6	0,42	99,2
Furukawa3	0,33	99,8	0,28	99,8	0,37	99,2

Omien havaintojen ja yllä esitetyn taulukon perusteella voidaankin tehdä johtopäätös, että kuvien määrä kannattaa pyrkiä rajoittamaan mahdollisimman pieneksi, sillä kuvien määrää nostamalla ei voida juurikaan parantaa pistepilven tarkkuutta. Oleellisempaa on se, että jokaisesta kohdasta ympäristöä saadaan kerättyä ainakin muutamia hyviä kuvia.

Kuvien määrää voidaan vähentää helposti temporaalisen alinäytteistämisen avulla. AR Drone 2.0 quadrokopterin käytännöllinen lentonopeus on maksimissaan noin 5 metriä sekunnissa ja sen kamera kuvaa 30 kuvaa sekunnissa, joten peräkkäisten kuvien välillä on erittäin vähän eroavaisuuksia. Tämän vuoksi jokaista videossa olevaa kuvaa on tarpeetonta käsitellä. Optimaalinen näytteistystaajuus riippuu muun muassa sovelluskohteesta, ympäristön rakenteesta ja kuvaustavasta, joten sitä ei voida määritellä yksiselitteisesti. Testauksen perusteella näytteistystaajuudet välillä 4-10 tuottivat tässä työssä halutun lopputuloksen kannalta hyviä tuloksia, hieman ympäristön kuvaustavasta riippuen. Huolellisesti kuvaamalla näytteistystaajuudella 10 saavutettiin erittäin hyviä tuloksia, mutta nopeasti ympäristö kuvaamalla peräkkäisten

kuvien eroavaisuudet kasvoivat jo osittain liian suuriksi. Näytteistystaajuudella 4 kuvaustavalla ei ollut enää merkitystä. Pelkästään temporaalisen alinäytteistämisen avulla kuvien määrä on siis mahdollista pudottaa ainakin neljäsosaan alkuperäisestä, ilman että menetetään juurikaan järjestelmän kannalta oleellista informaatiota.

## 5.2 Kuvien resoluution vaikutus

Toinen tärkeä testattava asia on se, että miten kuvien resoluutio vaikuttaa pistepilven ja näin ollen myös lopullisen 3D-mallin tarkkuuteen. Tässä kohtaa intuitio on helposti sellainen, että HD-kuva on mahdollisesti liian tarkka tällaisen varsin pelkistetyn 3D-mallin muodostamista varten. Esimerkiksi kappaleessa 4.2.3 esitetyn kuvasarjan (Kuva 11) pohjalta voisi ajatella, että resoluutiota voidaan huoletta pudottaa, sillä ainakin kolmessa ensimmäisessä kuvassa on selvästi tunnistettavissa sama ympäristö. Testaus kuitenkin osoittaa, että kuvien resoluution pudottaminen heikentää merkittävästi pistepilven laatua ja näin ollen myös lopullisen 3D-mallin laatua.

Taulukko 3 vertailee eri resoluutioilla muodostettujen pistepilvien sisältämien 3D-pisteiden määrää sekä resoluution vaikutusta kuvien kokoon levyllä. Taulukosta on luettavissa se, että resoluution pudottaminen vähentää hyvin nopeasti pistepilvessä olevien pisteiden määrää. Resoluution pudottaminen kertoimella 2 aiheuttaa jo sen, että pisteiden määrä putoaa alle viidesosaan alkuperäisestä. Kun resoluutiota pudotettiin kertoimella kahdeksan, niin pistepilveä ei saatu muodostettua enää lainkaan. Kuvien resoluutiota pudottamalla voidaan toki vähentää kuvien vaatimaa tilantarvetta, mutta tällaisessa järjestelmässä on kuitenkin paljon olennaisempaa, että ympäristöstä saadaan muodostettua laadukas pistepilvi.

Taulukko 3. Kuvien resoluution vaikutus

<b>Resoluutio / n</b>	<b>n = 1</b>	<b>n = 2</b>	<b>n = 4</b>	<b>n = 8</b>
Kuvien määrä	500	500	500	500
Kuvien koko levyllä (MB)	82,4	28,5	10,7	4,3
Pisteiden määrä pistepilvessä	131407	25576	1501	0

Taulukko 3 sisältämä informaatio on selkeämpi hahmottaa kuvien avulla. Kuva 17 sisältää kolme samaa ympäristöä esittävää pistepilveä, jotka on muodostettu VisualSFM:n avulla käyttämällä eri resoluutioisia kuvia. Kuvissa on havaittavissa silmämääräisesti arvioituna myös se, että miten kuvien resoluution pudottaminen vaikuttaa pistepilven tarkkuuteen. Ylimmäinen pistepilvi on muodostettu alkuperäisellä resoluutiolla esitettyjen kuvien pohjalta. Keskimmäisen pistepilven muodostamisessa kuvien resoluutiota on pudotettu kertoimella kaksi, jolloin resoluution pudottamisen vaikutus pistepilven laatuun on jo selkeästi nähtävillä. Pääasiassa tärkeimmät kohdat ympäristöstä ovat kuitenkin edelleen mallinnettuna, joten joissain sovelluksissa tällaisen ratkaisun käyttäminen saattaisi olla mahdollista. Alimmaisen pistepilven muodostamisessa kuvien resoluutiota on pudotettu kertoimella neljä. Tällaisen pistepilven käyttäminen käytännönsovelluksissa on jo mahdotonta, sillä siitä puuttuu useita tärkeitä kohteita ympäristöstä ja pisteiden sijainnit ovat erittäin epätarkkoja. Kun alkuperäisen kuvan resoluutiota pudotettiin kertoimella kahdeksan, VisualSFM ei saanut muodostettua pistepilveä enää lainkaan.



Kuva 17. Esimerkki kuvien resoluution vaikutuksesta pistepilven laatuun. Ylimmän pistepilven muodostamisessa on käytetty resoluutioltaan alkuperäisiä kuvia, keskimmäisessä resoluutiota on pudotettu kertoimella kaksi ja alimassa kertoimella neljä. Kun resoluutiota pudotettiin kertoimella kahdeksan, niin mallia ei saatu muodostettua enää lainkaan.

Kuvien resoluutiolla on yllättävänkin suuri merkitys pistepilven laatuun, jos sitä suhteuttaa siihen, että ihmiselle resoluution pudottamisella ei ole kuitenkaan kovinkaan suurta merkitystä. Ihminen nimittäin tunnistaa vaivatta saman ympäristön myös pienemmällä resoluutiolla esitetystä kuvasta. SFM-algoritmeille kuvien resoluution pudottaminen on kuitenkin merkittävä haitta. SFM-algoritmien toiminta perustuu paljolti siihen, että kuvista löydetään yhteneväisiä piirrepiisteitä, joiden sijainti saadaan määritettyä tarkasti. Kuvan resoluution pudottaminen aiheuttaa sen, että kuvassa olevia useampia pikseleitä joudutaan interpoloimaan yhdeksi pikseliksi, jolloin menetetään alkuperäiseen kuvaan verrattuna yksityiskohtia ja tarkkuutta. Tämä puolestaan johtaa siihen, että hyviä piirrepiisteitä ei löydetä enää niin useita tai niiden

sijaintia ei saada määritettyä niin tarkasti, jolloin pistepilven tarkkuus kärsii hyvin nopeasti. Tarkkuutta menetetään jo pienelläkin resoluution pudottamisella, sillä pikseleitä joudutaan interpoloimaan riippumatta siitä, että kuinka paljon resoluutiota pudotetaan.

Testauksen perusteella voidaankin tehdä johtopäätös, että kuvien resoluution pudottaminen ei ole useimmiten järkevää. Joissain sovelluksissa kohtuullinen resoluution pudottaminen on kenties mahdollista, mutta testauksen perusteella on todennäköistä, että se ei siitä huolimatta ole paras mahdollinen tapa toimia. Kuvien laadun pudottaminen nopeuttaa toki jonkin verran kuvien käsittelyä, mutta tällöin ympäristön mallintamisessa pitää käyttää useampia kuvia, jotta mallin tarkkuus saadaan pidettyä vastaavana. Paras mahdollinen lopputulos on todennäköisesti saavutettavissa siten, että käytetään mahdollisimman korkealaatuisia kuvia ja pyritään kehittämään menetelmä, jolla kuvien määrä saadaan minimoitua järkevällä tavalla. Tällöin päästään sekä parhaaseen mahdolliseen tarkkuuteen, että pienimpään mahdolliseen suoritusajkaan.

### 5.3 Järjestelmän parametrien asettaminen

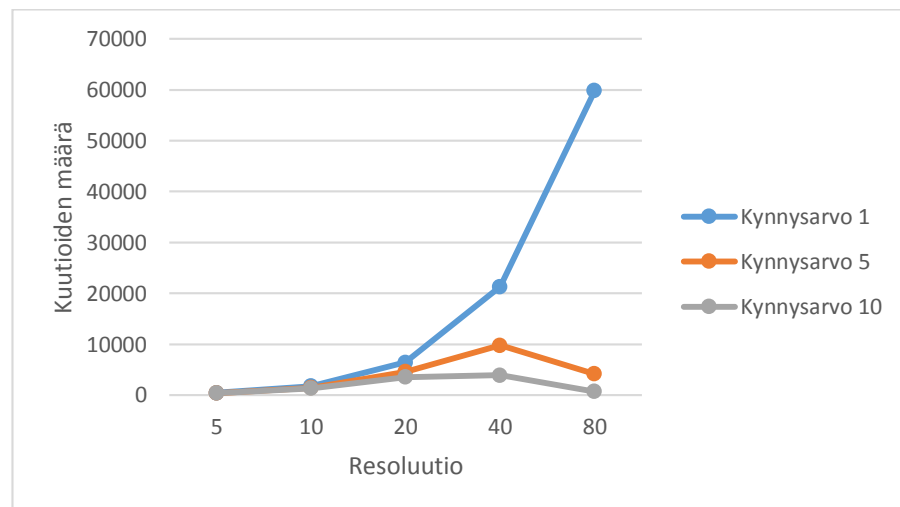
Kun pistepilveä lähdetään muokkaamaan yksinkertaisempaan vokseleista koostuvaan esitysmuotoon, niin järjestelmän tiedossa pitää olla kaksi parametria: kynnysarvo ja resoluutio. Kynnysarvolla siis kontrolloidaan sitä, että kuinka paljon pisteitä yhden vokselin täytyy sisältää, ennen kuin se käsitellään varattuna alueena. Pistepilven tulee helposti kohtia, joissa saattaa olla vain muutamia kohinan aiheuttamia pisteitä. Tällaista kohtaa ympäristössä ei tule käsitellä varattuna alueena, joten tämän vuoksi ohjelma on suunniteltu siten, että tällaiset kohinan aiheuttamat virheet voidaan tarvittaessa korjata kynnysarvoa säätämällä. Resoluutiota säätämällä voidaan taas muuttaa mallin tarkkuutta.

Resoluutiota ja kynnysarvoa ei voida käytännössä kuitenkaan asettaa toisistaan riippumattomasti. Jos joku kynnysarvo osoittautuu käytännössä hyväksi jollain tietyllä resoluutiolla, niin se ei ole sitä välttämättä jollain toisella resoluutiolla. Tämä johtuu siitä, että kun resoluutio-parametrin arvoa kasvatetaan, niin mallin muodostavista kuutioista tulee tilavuudeltaan pienempiä ja näin ollen ne sisältävät myös vähemmän pisteitä. Mikäli kynnysarvo pidetään samana, niin silloin osa tarpeellisistakin kuutioista tulkitaan kohinaksi ja tällöin mallista menetetään käyttökelpoista informaatiota. Tämän vuoksi resoluutio-parametrin arvoa kasvatettaessa kynnysarvoa tulee pienentää.

Taulukko 4 esittää numeerisesti eri kynnysarvojen ja resoluutioiden arvojen vaikutusta 3D-malliin tulevien varattujen kuutioiden lukumäärään. Kynnysarvolla 1 mallista ei siis ole suodatettu yhtään kuutiota pois, joten siellä on mukana kohinasta aiheutuvia varattuja kuutioita. Kynnysarvolla 1 kuutioiden määrä kasvaa aina kun resoluutiota suurennetaan, kuten pitääkin. Kun kynnysarvoa kasvatetaan, niin jossain vaiheessa resoluution kasvattaminen aiheuttaa kuitenkin sen, että resoluution kasvattaminen johtaa kuutioiden määrän vähenemiseen. Kuva 18 havainnollistaa tätä vielä kuvaajan avulla.

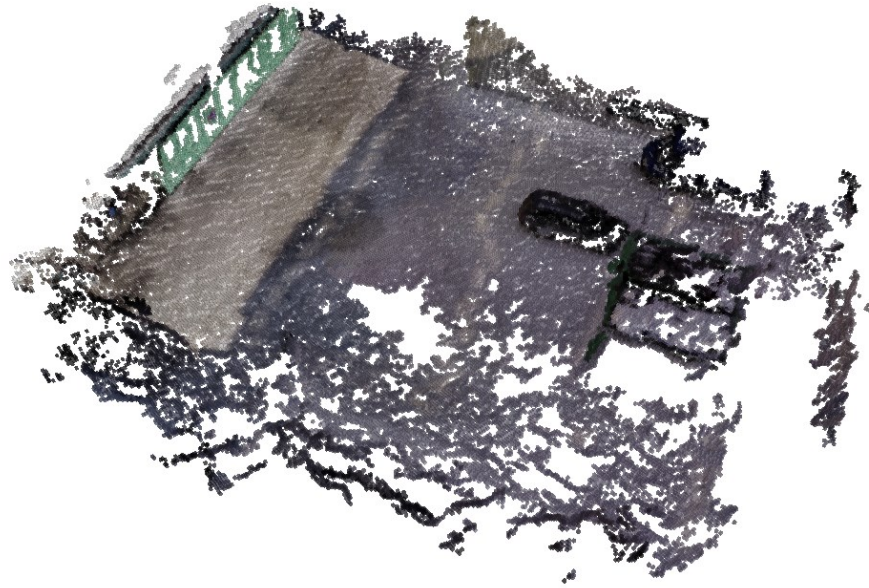
Taulukko 4. Järjestelmän parametrien vaikutus lopulliseen 3D-malliin

Kynnysarvo	Resoluutio	Kuutioiden määrä
1	5	457
	10	1768
	20	6430
	40	21291
	80	59836
5	5	415
	10	1473
	20	4589
	40	9781
	80	4168
10	5	397
	10	1337
	20	3550
	40	3911
	80	735

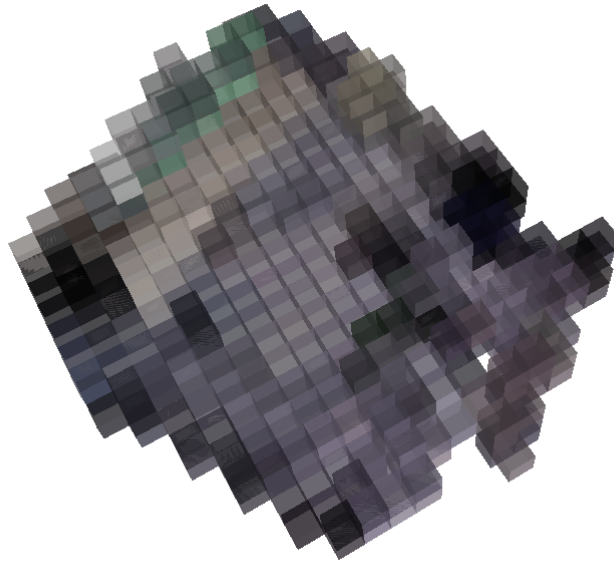


Kuva 18. Resoluution ja kynnysarvon vaikutus lopullisessa mallissa olevien kuutioiden määrään.

Resoluution ja kynnysarvon suuruuksien suhdetta on hankalaa määrittää tarkasti, sillä suhde riippuu hyvin pitkälti pistepilvessä olevan kohinan määrästä. Tässä diplomityössä muodostetuissa malleissa saavutettiin yleensä ottaen hyviä tuloksia silloin, kun kynnysarvo oli resoluutiosta riippuen 2-4. Käyttökelpoisia resoluution arvoja testatussa ympäristössä oli käytännössä 10-20, jotka tuottivat jo selkeästi tunnistettavia malleja, mutta eivät vielä liian tarkkoja mallin nopeaa käsittelyä ajatellen. Absoluuttisina mittoina ajateltuna tämä tarkoittaa käytännössä kuutioita, joiden sivujen pituudet ovat fyysisessä ympäristössä noin 25-50 senttimetriä. Resoluutiolla 20 muodostettu malli nähtiin jo kappaleessa 4.2.5 (Kuva 15), joten sen avulla on helppo hahmottaa tällaisella resoluutiolla muodostetun mallin tarkkuutta. Kuvan malli on muodostettu kynnysarvolla 3. Kuva 19 ja Kuva 20 havainnollistavat vielä eri kynnysarvoilla ja resoluutioilla muodostettuja 3D-malleja. Molemmat mallit on muodostettu täsmälleen saman pistepilven pohjalta.



Kuva 19. Resoluutiolla 80 ja kynnyksarvolla 1 muodostettu malli ympäristöstä. Tällainen malli on jo tarpeettoman tarkka tässä diplomityössä suunniteltuun käyttötarkoitukseen, sillä lähialueella toimiessa robotti voi käyttää sen omia sensoreita 3D-informaation mittaamiseen.



Kuva 20. Resoluutiolla 5 ja kynnyksarvolla 5 muodostettu malli ympäristöstä. Näin pienellä resoluutiolla muodostettu malli yhdistettynä suureen kohinaan ei tuota enää käytännössä käyttökelpoista mallia. Edes tasaista maa-aluetta ei ole saatu kuvattua enää tasaisena, vaan se on jakautunut selkeästi kahteen eri tasoon. Robotin toiminnan kannalta tällainen virhe voi olla hyvinkin haitallinen, sillä eri tasossa olevat maapinnat saatetaan tulkita sellaiseksi esteeksi, jota robotti ei voi ylittää.

Hyväksi todetut resoluutio-parametrin arvot ovat sopivia vain ympäristössä, joka on samankokoinen kuin testauksessa käytetty ympäristö. Mikäli ympäristön koko muuttuu, niin silloin myös resoluutio-parametrin arvoa on muutettava. Tämän

diplomityön käyttötarkoituksessa hyviä tuloksia saatiin aikaan sellaisella resoluutio-parametrin arvolla, että se johtaa fyysisessä ympäristössä sivun pituuksiltaan noin 25-50 senttimetrin kokoiisiin kuutioihin, mutta erilaisessa käyttötarkoituksessa tilanne voi olla toinen. Kuution sivun pituuden määrittäminen vaatii sen, että ympäristöstä saadaan mitattua jotain absoluuttisia mittoja, sillä ympäristön kokoa ei käytännössä voida aina tietää etukäteen. Toinen huomioitava asia on se, että kynnyсарvon käyttäminen ei ole välttämättä lainkaan tarpeellista, mikäli järjestelmään kehitetään parempi menetelmä kohinan hallitsemiseen.

## 6 ONGELMAKOHDAT JA JATKOKEHITYS

Järjestelmän toteutuksen ja testauksen aikana huomattiin useita ongelmakohtia, joita korjaamalla ja kehittämällä järjestelmästä olisi mahdollista kehittää huomattavasti nykyistä parempi. Ihanteellisessa tapauksessa järjestelmä kykenisi mallintamaan ympäristön riittävällä tarkkuudella luotettavasti ja täysin itsenäisesti. Järjestelmän alustavan toteutuksen ja testauksen perusteella on perusteltua olettaa, että tällainen järjestelmä on mahdollista toteuttaa käytetyllä laitteistolla, ainakin silloin, mikäli järjestelmälle ei aseteta minkäänlaisia reaaliaikavaatimuksia. Tällaisen ohjelmiston kehittäminen vaatii kuitenkin huomattavan työpanoksen, eikä sellaista voitu tämän diplomityön puitteissa ajanpuutteen vuoksi toteuttaa. Tässä kappaleessa perehdytään kuitenkin siihen, että miten järjestelmää olisi mahdollista kehittää tarkemmaksi, luotettavammaksi ja automaattisemmaksi. Hyvän järjestelmän toteuttaminen edellyttää käytännössä jokaisen järjestelmän vaiheen jatkokehittelyä.

Kappaleen rakenne on seuraava: Kappaleessa 6.1 käsitellään datan keräämiseen liittyviä haasteita. Kappaleessa 6.2 käsitellään eri vaihtoehtoja kuvien laadun parantamiseen. Kappaleessa 6.3 käsitellään sitä, että miten SFM-vaiheen toteuttamista voitaisiin kehittää paremmin tähän käyttötarkoitukseen sopivaksi. Kappaleessa 6.4 pohditaan vaihtoehtoja lopullisen kuutioista muodostetun 3D-mallin jatkokehitykseen. Lopuksi kappaleessa 6.5 käydään läpi vielä järjestelmän eri vaiheiden integrointia.

### 6.1 Datan keräämiseen liittyvät haasteet

Itsenäisen järjestelmän kehittäminen vaatii ensimmäiseksi tietenkin sen, että datan kerääminen saadaan toteutettua automaattisesti robotin avulla. Tässä diplomityössä tämän ongelman ratkaisemiseen ei perehdytty sen tarkemmin, mutta esimerkkejä jo varsin hyvin toimivista järjestelmistä esiteltiin kappaleessa 3.5, kuten [36], [37] ja [38]. Tämän ongelman ratkaiseminen on siis mahdollista nykyisiä SLAM-algoritmeja käyttämällä. Esimerkit toteutetuista järjestelmistä ovat kuitenkin keskittyneet lähinnä robotin itsenäiseen navigointiin, joten niissä ei ole juurikaan huomioitu ympäristön mallintamista eikä mallille ole asetettu kovinkaan kummoisia vaatimuksia.

Mikäli ympäristöstä halutaan muodostaa tarkka malli SFM- ja MVS-menetelmien avulla, niin silloin ympäristön kuvaustapaan on kiinnitettävä erityistä huomiota, riippumatta siitä, että kuvataanko ympäristö automaattisesti vai ihmisen avustamana. Koko ympäristön kuvaaminen on helpointa toteuttaa seuraavalla menetelmällä: Kopterilla lennetään suoraan pisteestä  $A$  pisteeseen  $B$  ja pyörähdetään siellä 360 astetta ympäri. Pisteestä  $B$  suunnassa on tässä tapauksessa mielenkiintoinen kohde, josta halutaan lisä informaatiota, joten tämän vuoksi siirrytään lähemmäksi kohdetta. Tämän jälkeen lennetään pisteestä  $B$  pisteeseen  $C$ , jonka suunnassa on seuraava mielenkiintoinen kohde, ja pyörähdetään siellä 360 astetta ympäri. Näin jatketaan kunnes jokainen mielenkiintoinen kohta ympäristöstä on saatu kuvattua useammasta eri kuvakulmasta. Tällainen kuvaustapa on ihmiselle hyvin intuitiivinen, sillä se ei vaadi juurikaan suunnittelua ja myös robotin ohjaaminen tällä tavalla kuvattuna on helpointa. Tällainen kuvaustapa on helppo ohjelmoida myös tietokoneelle, joten nopeasti ajateltuna tuntuisi siltä, että ympäristö kannattaa kuvata juurikin tällä tavalla.

Kuvaustapa on kuitenkin erittäin ongelmallinen SFM- ja MVS-menetelmien kannalta. Ongelma on nimittäin siinä, että paikallaan pyörähtäminen ei tuota mitään uutta informaatiota ympäristön 3D-rakenteesta, joten peräkkäisiä videon kuvia ei

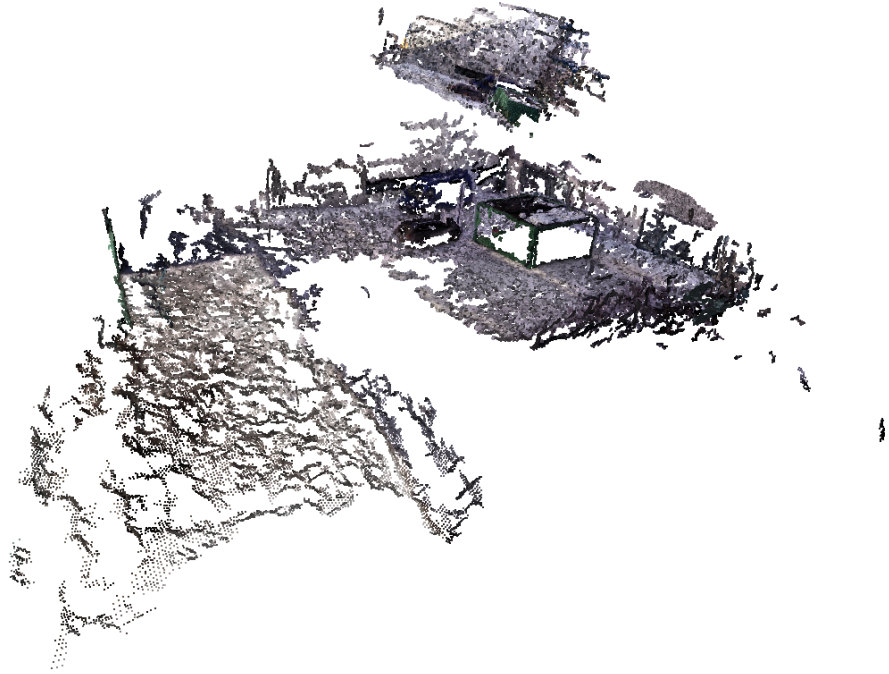
voida tällöin käyttää 3D-informaation selvittämiseen. Myös suoraan pisteestä  $A$  pisteeseen  $B$  siirtyminen tuottaa käytännössä hyvin vähän uutta informaatiota ympäristön 3D-rakenteesta.

Asiaa voidaan tarkastella matemaattisesti yhtälön (21) avulla, kun muistetaan, että kameran ollessa kalibroitu  $F = E = TR$ . Mikäli  $T$  tai  $R$  on kuvien välillä muuttumaton, niin silloin yhtälö ratkeaa muotoon  $0 = 0$ . Paikallaan pyörähtäessä  $T = 0$ , joten tällöin yhtälö (21) on aina tosi, riippumatta  $x$  ja  $x'$  sijainneista. Tätä voi tarkastella myös siten, että 3D-informaation laskeminen perustuu lopulta trigonometriaan, sillä koordinaatin  $X$  paikka koordinaatistossa  $W$  määritellään kahden suoran leikkauspisteeseen. Paikallaan pyörähtäessä koordinaatti  $X$  määritellään koordinaatistossa  $W$  saman vektorin avulla riippumatta siitä, että tarkastellaanko sitä pisteen  $x$  vai  $x'$  kautta. Tällöin ei ole olemassa mitään leikkauspistettä, jota voitaisiin trigonometrian avulla ratkaista. Vastaava ongelma tulee vastaan myös silloin, mikäli liikutaan suoraan pisteestä  $A$  pisteeseen  $B$ . Tällöin suoralla  $AB$  olevien pisteiden 3D-sijainneista ei saada mitään uutta informaatiota, sillä tällöin  $R = 0$ . Kaikista muista ympäristön pisteistä informaatiota saadaan toki teoriassa kerättyä, mutta suoran  $AB$  läheisyydessä muutos on niin pieni, että käytännössä sitä ei välttämättä havaita kohinan vuoksi ollenkaan.

Tällainen kuvaustapa johtaa siis siihen, että 3D-informaation laskemiseen on olemassa hyvin vähän oikeasti käyttökelpoisia kuvia. Suurin osa otetuista kuvista ovat vain saman informaation toistoa, joten ne ainoastaan hidastavat laskennan suorittamista ja saattavat pahimmillaan myös lisätä kohinan vaikutusta, jolloin mallista tulee epätarkempi. Käyttökelpoiset kuvat ovat taas kuvattu todennäköisesti hyvin eri kuvakulmista, sillä pisteet  $A$ ,  $B$ ,  $C$  jne. eivät ole yleensä kovin lähellä toisiaan. Tällöin 3D-informaation selvittämiseen parhaiten soveltuvien kuvien skaala voi olla hyvin erilainen tai sitten ne ovat kuvattu hyvin eri suunnista ympäristöä. Tämä vaikeuttaa vastaavien piirrepisteiden löytämistä, jolloin SFM-algoritmit eivät välttämättä tunnista kuvien esittävän samaa ympäristöä.

Kuva 21 havainnollistaa tällaisen kuvaustavan vaikutusta pistepilven muodostamiseen vielä visuaalisesti. Kuvan esimerkissä VisualSFM ei ole tunnistanut eri skaalalla esitettyjen kuvien kuvaavan samaa kohtaa ympäristöstä, joten kuvista on tämän vuoksi muodostettu kaksi erillistä pistepilveä. Tämän lisäksi isosta osasta ympäristöä ei ole saatu kerättyä lainkaan 3D-informaatiota. Tällainen ongelma voidaan välttää paremmin toteutetulla ympäristön kuvauksella.

SFM- ja MVS-algoritmien kannalta ihanteellisimmin vertailtava kuvapari on sellainen, jossa kuvat on kuvattu saman etäisyyden päästä kohteesta ja hieman eri kuvakulmista. Tällöin kuvista on helppo löytää vastaavat piirrepisteet ja näin ollen ne tunnustetaan esittävän samaa ympäristöä. Teoriassa piirrepisteiden etsimiseen käytetyt SIFT-algoritmit ovat invariantteja skaalalle [30], joten kuvausetäisyydellä ei näin ollen pitäisi olla merkitystä, mutta käytännössä kauemmaksi mennessä menetetään useita piirrepisteitä, jotka ovat lähempänä kuvattuna erotettavissa. Hyvä menetelmä ympäristön kuvaamiseen onkin sellainen, että ympäristöstä otetaan jokin kiintopiste, jota kuvataan saman etäisyyden päästä eri kuvakulmista. Kiintopistettä vaihtelemalla jokaisesta osasta ympäristöä saadaan riittävästi 3D-informaation laskemiseen soveltuvia kuvia. Tällaisella kuvaustavalla robotilla liikutaan pääasiassa sivuttain ja sitä pyöritetään samalla kun siirrytään paikasta toiseen. Robotin ohjaaminen on tällöin vaikeampaa ja ympäristön kuvaaminen on hitaampaa, mutta tällä tavalla saavutettiin huomattavasti parempi lopputulos paljon pienemmällä määrällä videota.



Kuva 21. Kuvassa on esimerkki osaltaan huonosti toteutetun kuvauksen aiheuttamasta ongelmasta pistepilvää muodostettaessa. VisualSFM ei ole löytänyt läheltä ja kaukaa kuvatuista näkymistä riittävästi yhteneväisiä piirrepiisteitä, joten se on olettanut niiden kuvaavan eri ympäristöä ja näin ollen muodostanut niistä kaksi erillistä pistepilvää. Ylempänä on kaukaa kuvattujen näytteiden pohjalta muodostettu pistepilvi ja vastaavasti alempana oleva pistepilvi on läheltä kuvattujen näytteiden pohjalta muodostettu. Ihminen kuitenkin tunnistaa pistepilvien esittävän samaa ympäristöä vain eri skaalalla esitettynä.

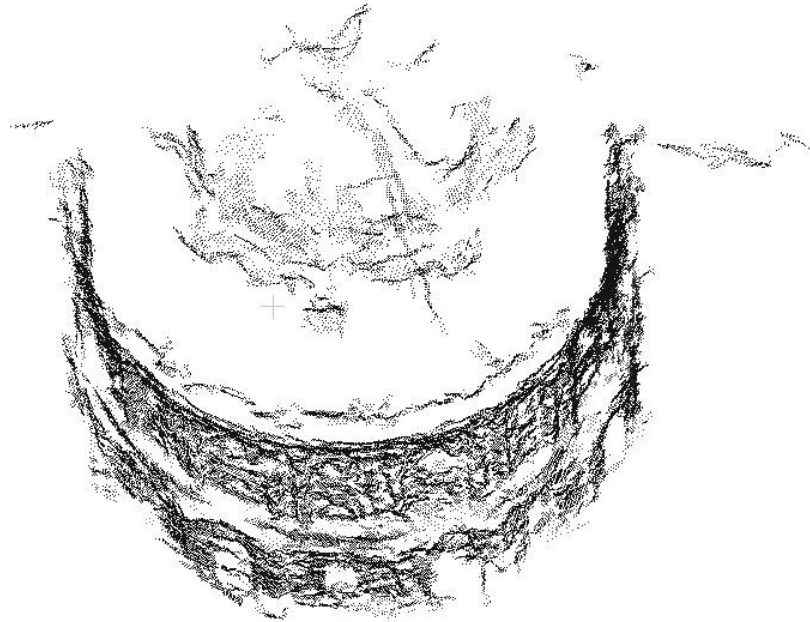
## 6.2 Kuvien laadun parantaminen

Järjestelmän toteuttamisen ylivoimaisesti ongelmallisoin vaihe oli hyvälaatuisen pistepilven muodostaminen videosta poimittujen kuvien pohjalta. Kuva 14 oleva pistepilvi on kenties kaikkein paras, mitä VisualSFM:llä saatiin tällä järjestelmällä muodostettua. Tässäkin tapauksessa pistepilvi piti valita manuaalisesti useiden muiden pistepilvien joukosta, sillä VisualSFM muodosti yleensä kuvien pohjalta useampia pistepilviä. Tähän ongelmaan viitattiin jo edellisessä kappaleessa ja se on osaltaan huonon kuvaustavan aiheuttamaa. Kuva 14 pistepilvi on kuitenkin muodostettu aineiston pohjalta, joka on kuvattu erittäin huolellisesti sekä kuvat on kalibroitu erittäin tarkasti. Internetistä katsottujen esimerkkien perusteella VisualSFM:ltä olisi voinut tällaisen kuvamateriaalin pohjalta odottaa huomattavasti parempaa suoriutumista, kuin mitä tässä työssä saavutettiin.

VisualSFM:n heikon suoriutumisen pääasiallinen syy tässä järjestelmässä on todennäköisesti AR Drone 2.0 kopterilla otettujen kuvien sisältämä runsas kohina. Kohinalla on nimittäin ihan vastaava vaikutus SFM-algoritmien toiminnalle, kuin kappaleessa 5.2 käsitellyllä kuvien resoluution pudottamisella. Runsas kohina vaikeuttaa hyvien piirrepiisteiden löytämistä ja niiden sijainnin tarkkaa määrittämistä, jolloin sillä on erittäin merkittävä vaikutus SFM-algoritmin onnistumiseen.

Kuva 22 on esimerkki pistepilvestä, joka on muodostettu VisualSFM:llä internetistä kerättyjen valokuvien pohjalta Oxfordissa sijaitsevasta Radcliffe Camera-

rakennuksesta. Mallin luomiseen käytettiin vain 56 kuvaa, joiden resoluutio oli hieman pienempi kuin AR Drone 2.0 kamerassa, yleensä 1024x683 pikseliä. Pistepilven muodostaminen vei kaikkine vaiheineen hieman alle puoli tuntia ja siinä on yhteensä 56598 erillistä 3D-pistettä. Tämä pistepilvi on silmämääräisesti arvioituna paljon tarkempi ja yksityiskohtaisempi, kuin mitä AR Drone 2.0 kopterilla otettujen kuvien pohjalta saatiin vastaavalla kuvamäärällä muodostettua. Kuva 23 on esimerkki AR Dronella muodostetusta pistepilvestä. Tällainen lopputulos tuntuu odottamattomalta, sillä AR Dronen ottamisessa kuvissa on parempi resoluutio. Silmämääräisesti arvioituna ne ovat kuitenkin selvästi kohinaisempia kuin internetistä kerätyt kuvat, joten tämä tukee sitä olettamusta, että kohinalla on ollut suuri merkitys VisualSFM:n heikkoon suoriutumiseen tässä järjestelmässä käytettynä.

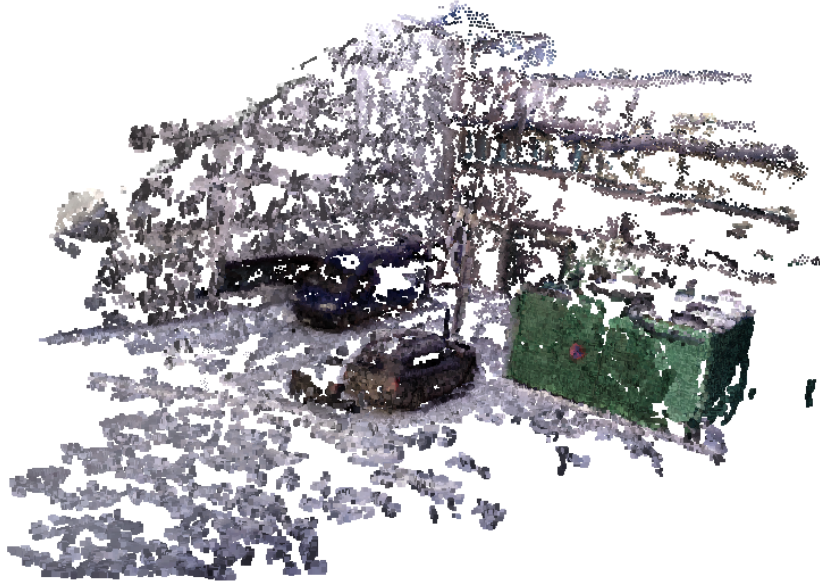


Kuva 22. Esimerkki internetistä kerättyjen valokuvien pohjalta muodostetusta pistepilvestä. Rakennuksesta muodostettu pistepilvi on kattopintaa lukuun ottamatta erittäin tarkka ja yksityiskohtainen.

Kuva 24 tarkastelemalla saa käsitystä AR Dronella otetuissa videokuvissa esiintyvistä runsaasta kohinasta. Kuva on rajattu alkuperäisestä kalibroidusta kuvasta 632x422 pikselin kokoiseksi. Kuvasta on selvästi havaittavissa runsaan kohinan vaikutus kuvan laatuun. Kuvasta erottaa kyllä suuret rakenteet ympäristössä, mutta esimerkiksi tarkkoja kasvonpiirteitä ja auton rekisterinumeroa on mahdotonta tunnistaa kuvan avulla, vaikka kuva onkin otettu varsin läheltä näitä kohteita. Normaalisti päivänvalossa ja näin läheltä kohdetta otetusta kuvasta tällaisten piirteiden erottaminen ei olisi ongelma.

Kuvissa oleva runsas kohina voi aiheutua useasta eri syystä. Todennäköisesti pääasiallinen syy on AR Dronen heikkolaatuinen kamera. Kamerassa on CMOS-kenno [7], joten se asettaa omat rajoituksensa kameran suorituskyvylle erityisesti videosovelluksissa [60]. Myös kameran edullinen linssi asettaa rajoituksia kameran tarkkuudelle. Oma merkityksensä voi olla myös sillä, että AR Drone pakkaa videon MP4-formaatilla, ennen kuin se lähettää sen langattomasti isäntälaitteeseen. Tällöin

videon dataan aiheutuu häviötä, jolloin se saattaa vaikuttaa kohinan lisääntymiseen. Tässä työssä kohinan aiheutumisen syihin ei perehdytä sen tarkemmin, mutta jatkokehityksen kannalta niitä on kuitenkin tarpeen tutkia, sillä pääasiallisen kohinan lähteen poistamalla järjestelmän suorituskykyä on mahdollista parantaa.



Kuva 23. Esimerkki AR Drone 2.0 kopterilla otettujen kuvien pohjalta muodostetusta pistepilvestä. Pistepilvessä on erittäin paljon kohinaa ja useita kohtia ympäristöstä ei ole saatu mallinnettua ollenkaan.



Kuva 24. Esimerkki kuvissa esiintyvstä runsaasta kohinasta. Kuva on rajattu alkuperäisestä kuvasta ja sen resoluutio on 632x422 pikseliä, eli se on kooltaan noin neljäsosa alkuperäisestä kuvasta.

Kohinan vähentämiseen on olemassa useita erilaisia ratkaisuja. Helpoin vaihtoehto ongelman ratkaisemiseen olisi vaihtaa AR Droneen laadultaan parempi kamera, jolloin päästään mahdollisesti jo paljon parempaan lopputulokseen. Tämä kuitenkin nostaa käytettävän laitteiston hintaa, mikä ei ole toivottava asia. Toinen vaihtoehto olisi

kokeilla sellaista ratkaisua, että videokuva tallennetaan suoraan AR Droneen kiinnitettyyn USB-muistiin. Tällöin videota ei tarvitse lähettää langattomasti, joten sen pakkaaminen ei näin ollen ole välttämätöntä. Tällainen vaihtoehto ei tosin ole toimiva siinä tapauksessa, mikäli järjestelmästä halutaan täysin automaattinen, koska datan siirtäminen tietokoneelle pitää tällöin toteuttaa ihmisen avustamana. Kaikissa järjestelmissä käyttökelpoinen ratkaisu kohinan vähentämiseen on tehdä se ohjelmallisesti. Tähän löytyy useita eri vaihtoehtoja, esimerkiksi erilaiset superresoluutio-tekniikat [61]. Tällaiset tekniikat perustuvat siihen, että niissä hyödynnetään videokuvassa olevaa datan toistoa, jolloin usean huonolaatuisen kuvan avulla saadaan muodostettua parempilaatuinen kuva esimerkiksi datan keskiarvoistamisen avulla. Tällaista tekniikkaa on käytetty esimerkiksi 2d3 sensing järjestelmässä [45].

### 6.3 SFM-menetelmän kehittäminen

Kuten edellisessä kappaleessa mainittiin, järjestelmän suurin haaste oli saada muodostettua kuvien avulla laadukas pistepilvi ympäristöstä. Merkittävin syy VisualSFM:n heikkoon suorituskykyyn tässä diplomityössä toteutetun järjestelmän tapauksessa on edellisen kappaleen havaintojen perusteella mitä todennäköisimmin kuvissa oleva runsas kohina. Kohinaa vähentämällä VisualSFM:n muodostaman pistepilven tarkkuutta on mahdollista parantaa, mutta siitä huolimatta VisualSFM ei ole paras mahdollinen vaihtoehto toteuttamaan videon pohjalta ympäristöä mallintavaa järjestelmää. VisualSFM on nimittäin suunniteltu täysin erilaiseen käyttötarkoitukseen, joten se ei ole lähelläkään optimaalinen tapa SFM-ongelman ratkaisemiseen tämänkaltaisessa järjestelmässä. SFM-vaihetta kehittämällä järjestelmän toimintaa on mahdollista nopeuttaa merkittävästi sekä myös pistepilven laatua on mahdollista parantaa, kenties jopa enemmän kuin kohinan määrää vähentämällä.

VisualSFM on suunniteltu toimimaan erittäin suurien kuvamäärien kanssa, joten siinä mielessä se tuntuu luontevalta valinnalta tämänkaltaiseen sovellukseen, jossa ympäristö mallinnetaan videon pohjalta. VisualSFM on kuitenkin tarkoitettu käytettäväksi sellaisten kuvien kanssa, joiden oletetaan olevan kuvattu täysin sattumanvaraisista paikoista, esimerkiksi sellaisissa tapauksissa, joissa ympäristö mallinnetaan internetistä ladattujen kuvien pohjalta. Tällöin peräkkäisten kuvien välisistä suhteista ja kameroiden paikoista ei voida tehdä mitään oletuksia etukäteen. Tällainen lähestymistapa johtaa siihen, että kaikki käytettävät kuvat pitää olla heti alusta asti järjestelmän käytössä, jolloin inkrementaalinen mallintaminen ei ole mahdollista. VisualSFM kyllä estimoii 3D-sijainnit inkrementaalisesti, mutta piirrepisteiden etsimisessä tällainen ei ole mahdollista, sillä peräkkäiset kuvat eivät välttämättä sisällä mitään yhteisiä piirrepisteitä. Tämän vuoksi yhteisiä piirrepisteitä etsiessä kaikkia kuvia pitää verrata kaikkien kuvien kanssa. Ympäristön koon ja näin ollen myös kuvien määrän kasvaessa tämän vaiheen suorittaminen on erittäin hidasta, kuten kappaleen 5.1 esimerkissä todettiin. VisualSFM ei myöskään oletta, että kaikki kuvat esittävät samaa ympäristöä, joten mikäli kuvien väliltä ei löydy yhteneväisyyksiä, niin se muodostaa niiden pohjalta useampia pistepilviä ympäristöstä. Tästä nähtiin esimerkki Kuva 21. Tämä johtaa siihen, että paras mahdollinen pistepilvi pitää valita manuaalisesti kaikkien pistepilvien joukosta.

Videokuvaa käytettäessä tilanne on siinä mielessä merkittävästi erilainen, että videon peräkkäiset kuvat eivät ole sattumanvaraisia, vaan ne on kuvattu hyvin läheltä toisiaan. Tällöin tiedetään, että kameroiden paikat eivät poikkea toisistaan kovinkaan paljoa, jolloin peräkkäisissä kuvissa on paljon yhteneväisyyksiä. Tätä tietoa voidaan hyödyntää SFM-algoritmin toteuttamisessa. Videokuvaa käytettäessä piirrepisteiden vertaaminen voidaan suorittaa kuvien välillä pienemmällä määrällä kuvia, jolloin järjestelmän toimintaa voidaan nopeuttaa. Tällainen lähestymistapa mahdollistaa myös inkrementaalisen ympäristön mallintamisen, jolloin järjestelmä on mahdollista saada toimimaan kenties jopa reaaliajassa, kuten esimerkiksi kappaleessa 3.5 esitelty järjestelmä [39] osoittaa.

Järjestelmän tarkkuutta ja nopeutta on mahdollista parantaa myös AR Dronen sensoreilta saatavaa informaatiota käyttämällä. AR Dronessa on tarvittavat sensorit INS-paikannukseen ja lisäksi siihen on saatavilla lisälaite GPS-paikannukseen [47]. Mikäli sensoreilta saatava informaatio on edes kohtuullisen tarkkaa, niin sensoreiden tietoa hyödyntämällä voitaisiin tarkentaa kameroiden sijaintia, jolloin välttyttäisiin ainakin karkeimmilta virheiltä. Yksi karkeimmista virheistä on tapahtunut aikaisemmin nähdyssä Kuva 22, vaikka sitä ei kuvan pistepilvestä huomaakaan. Joissain tapauksissa VisualSFM nimittäin sijoitti kameran paikan väärälle puolelle kohteen pintaa, sillä 2D-kuvan perusteella ei voi olla varma, että mistä suunnasta se on kuvattu, koska toiselta puolelta pintaa se näyttää vain peilikuvulta. Oikeasti kuvan rakennus on pyöreä ja kuvanäytteitä on ollut jokaisesta suunnasta rakennusta, mutta se näyttää jokaisesta suunnasta niin samanlaiselta, että VisualSFM on tulkinut rakennuksen puolikaaren muotoiseksi. Asento- ja paikkatietoja hyödyntämällä tällaisilta karkeilta virheiltä voitaisiin välttyä.

Mikäli asento-, liike- ja paikkatiedot osoittautuvat riittävän tarkoiksi ja ne mukautuvat riittävän nopeasti robotin liikkeisiin, niin silloin järjestelmää on mahdollista kehittää todella paljon paremmaksi niitä hyödyntämällä. Niiden avulla voidaan selvittää kameroiden paikat, jolloin SFM-menetelmän ensimmäisen vaiheen suorittaminen helpottuu huomattavasti. Niiden avulla voidaan myös päätellä, että mitkä kuvat on suunnattu samaan kohtaan ympäristössä, jolloin piirrepisteiden etsiminen helpottuu. Yksi tärkeimmistä hyödyistä on se, että sensorien informaatiota hyödyntämällä tarvittavien kuvien määrää voidaan vähentää ja myös varmistaa, että jokaisesta kohdasta ympäristöä saadaan kerättyä sopivasti kuvia. Uusi kuva voidaan ottaa esimerkiksi aina silloin, kun muutos edellisen kuvan paikkaan on riittävän suuri, jolloin välttyään ottamasta turhia kuvia. Uuden kuvan ottaminenkin on mahdollista ajoittaa siten, että se otetaan vain silloin, kun robotti on suhteellisen paikallaan, jolloin liikkeestä ei aiheudu kuvaan ylimääräistä epätarkkuutta. Viimeisimpänä hyötynä mainittakoon se, että GPS-koordinaattien avulla saadaan selvitettyä 3D-mallin koko suhteessa fyysiseen ympäristöön.

Tämän kappaleen tarkastelun perusteella on selvää, että VisualSFM ei ole todellakaan paras mahdollinen ohjelmisto tällaisen järjestelmän toteuttamiseen. Järjestelmän jatkokehittämisessä kannattaakin tutkia sitä, että miten luotettavaa ja tarkkaa informaatiota robotin sensoreilta on saatavissa, jolloin voidaan arvioida tarkemmin niiden käyttömahdollisuuksia järjestelmän toteuttamisessa. SFM-vaihe kannattaa myös toteuttaa inkrementaalisesti, jolloin menetelmä soveltuu paremmin käytettäväksi erittäin suurilla ympäristöillä mallinnettaessa. Tässä vaiheessa kannattaa myös harkita sitä, että muodostetaanko 3D-informaation pohjalta suoraan haluttu 3D-malli ympäristöstä. Pistepilvi ei ole kuitenkaan haluttu esitysmuoto 3D-mallille, joten sen muodostaminen on tarpeeton välivaihe järjestelmän toteuttamisessa.

## 6.4 Lopullisen mallin parantaminen

Järjestelmän toteuttamisessa lopulta kaikkein merkitsevin asia on se, että millaisen 3D-mallin se saa luotua ympäristöstä. Se, että millä tavalla tähän lopputulokseen päästään, ei ole niin oleellista. Samoin suunnitellussa käyttötarkoituksessa mallin muodostamiseen käytetyllä ajalla ei ole niin suurta merkitystä, sillä oletus on, että mallintaminen voidaan suorittaa hyvissä ajoin ennen maanrobotin tehtävän aloittamista. Tämän vuoksi järjestelmän jatkokehittämisessä pitäisikin ihan ensimmäiseksi parantaa niitä asioita, joilla voidaan vaikuttaa lopullisen 3D-mallin laatuun. Tässä mielessä ajateltuna nykyinen järjestelmä on jo sellaisenaan toimiva, joten VisualSFM:n kehittäminen tai korvaaminen paremmin tarkoitukseen soveltuvalla ohjelmistolla ei ole välttämätöntä.

3D-mallin laatu riippuu suoraan muodostetun pistepilven laadusta. Pistepilven laatua on mahdollista parantaa kuvien paremmalla esikäsittelyllä, jota käsiteltiin kappaleessa 6.2, sekä pistepilven jatkokäsittelyllä [4], esimerkiksi ulkopuolisten pisteiden suodattamisella ja tasoittavan suodatuksen avulla. Kun pistepilvestä saadaan poistettua ylimääräinen kohina, niin silloin kynnysarvo-parametrin käytöstä voidaan luopua, jolloin sen säätämisestä kohdalleen ei aiheutuisi enää ylimääräistä vaivaa. Tällä hetkellä parametri on vielä tarpeellinen, sillä se on ainoa keino hallita pistepilvessä olevan kohinan aiheuttamia häiriöitä.

Toinen 3D-malliin helposti tehtävä parannus olisi se, että selvittäisiin sen oikea koko fyysisessä ympäristössä. Tällöin se olisi käyttökelpoisempi ympäristössä toimiessa, koska sen avulla pystyttäisiin arvioimaan etäisyyksiä eri paikkojen välillä. Tämä parannus olisi helppo tehdä pelkästään GPS-informaatiota hyödyntämällä, kuten edellisessä kappaleessa todettiin. Mikäli GPS-koordinaatit myös tallennetaan jokaiselle vokselille mallia muodostettaessa, niin silloin maanrobotin olisi todella helppo paikallistaa itsensä ympäristössä tämän mallin avulla.

Kolmas selkeä parannus olisi tutkimattoman ympäristön selvittäminen ja sen merkkäminen 3D-malliin. Tähänkin järjestelmässä on olemassa jo edellytykset, sillä VisualSFM antaa ulostulona myös kameroiden suhteelliset sijainnit ympäristössä. Näiden sijaintien perusteella on mahdollisuus selvittää sellaiset pisteet, joita ei mistään kuvakulmasta ole voinut nähdä. Esimerkiksi Wurm ym. järjestelmässä [26] on selvitetty myös tutkimaton ympäristö ja se on tallennettu 3D-malliin mukaan.

## 6.5 Järjestelmän vaiheiden integrointi

Viimeinen merkittävä parannus järjestelmän suoriutumiseen on se, että järjestelmän eri vaiheet integroimaan toimimaan yhdessä. Tällä hetkellä jokainen vaihe on toteutettu toimimaan itsenäisenä ohjelmana, jolloin ihmisen pitää jokaisen vaiheen suorittamisen jälkeen käynnistää seuraavan vaiheen suorittaminen manuaalisesti. Tällainen ratkaisu jouduttiin jättämään järjestelmään sen vuoksi, että pistepilven muodostamista ei saatu suoritettua riittävän luotettavasti VisualSFM-ohjelmiston avulla. Myös VisualSFM:n käyttäminen on jaettu neljään eri vaiheeseen, joten automaattisen järjestelmän toteuttaminen vaatisi myös VisualSFM:n muokkaamista. Kun SFM-vaihe saadaan toteutettua sillä tavalla, että se kykenee luotettavasti ja automaattisesti muodostamaan kuvien pohjalta yhden ainoan pistepilven, niin tämän jälkeen järjestelmän eri osa-alueet on helppo integroida toimimaan yhdessä.

## 7 POHDINTA

Järjestelmän toteuttamisessa kohdattuja ongelmia ja niiden mahdollisia jatkokehitysmahdollisuuksia pohdittiin perusteellisesti jo kappaleessa 6. Tässä kappaleessa pohditaan diplomityön tekemisen vaiheita vielä hieman yleisemmällä tasolla.

Diplomityön aihetta suunniteltaessa ja rajattaessa mietittiin aluksi, että aiheen teorian tutkiminen ja käsittely jätetään vähemmälle huomiolle, jotta järjestelmän toteuttamiseen jäisi mahdollisimman paljon aikaa. Esimerkiksi VisualSFM:n tapauksessa ajatus oli aluksi sellainen, että ohjelmisto integroidaan järjestelmään sellaisenaan, jolloin ei ole välttämätöntä ymmärtää niin tarkasti ohjelmiston toimintaa. Ongelmien ilmetessä kävi kuitenkin selväksi, että ohjelmisto ei sellaisenaan sovellu kovinkaan hyvin käytettäväksi tämänkaltaisessa sovelluksessa, jolloin oli välttämätöntä perehtyä ohjelmiston toteutukseen ja sen taustalla olevaan teoriaan perinpohjaisesti, jotta voisi ymmärtää mistä nämä ongelmat mahdollisesti johtuvat. Lopulta diplomityöhön tuli paljon enemmän teorian ja tutkimuksen osuutta kuin mitä alun perin suunniteltiin ja tämän takia järjestelmän toteuttamiseen ei jäänyt niin paljon aikaa.

Toteutettu järjestelmä ei ole vielä käytännössä kovinkaan käyttökelpoinen. Muodostetut 3D-mallit ovat kyllä sellaisia, että mallinnettu ympäristö on niistä tunnistettavissa, mutta tarkkuus ei ole kuitenkaan vielä lähelläkään halutulla tasolla. Joissain sovelluksissa malleja on varmaankin mahdollista hyödyntää antamaan suuntaa antavaa informaatiota ympäristöstä, mutta suunnitellussa käyttötarkoituksessa tarkkuus ei ole vielä riittävä. Myöskään järjestelmän toiminta ei ole vielä riittävän luotettavaa eikä automaattista.

Järjestelmän toteuttamisen osuus jakaantui karkeasti neljään eri vaiheeseen: datan kerääminen, datan esikäsittely, pistepilven muodostaminen ja 3D-mallin muodostaminen sekä näiden lisäksi kameran kalibrointivaiheen toteuttaminen. Kameran kalibrointi toteutettiin Oulun yliopistossa kehitetyn järjestelmän avulla, joten kalibroinnin toteuttamiseen ei käytetty merkittävästi aikaa. Myös datan kerääminen jätettiin tässä vaiheessa vähemmälle huomiolle, joten se toteutettiin ihmisen avustamana. Alkuperäinen ajatus oli, että käytettävissä oleva aika jaetaan pääasiassa tasaisesti datan esikäsittelyyn, pistepilven muodostamiseen ja 3D-mallin muodostamisen välille. Pistepilven muodostaminen tiedettiin vaikeimmaksi vaiheeksi toteuttaa käytännössä, mutta ajatuksena oli, että se toteutetaan jollakin avoimen lähdekoodin ohjelmistolla, jolloin vaihe saadaan toteutettua kohtuullisessa ajassa. Pistepilven muodostaminen VisualSFM:n avulla osoittautui kuitenkin paljon oletettua vaikeammaksi, joten vaiheen tutkimiseen ja toteuttamiseen jouduttiin käyttämään huomattava määrä aikaa. Tämän vuoksi muiden vaiheiden toteuttamiseen ei jäänyt niin paljon aikaa kuin aluksi suunniteltiin, jolloin esimerkiksi datan esikäsittelyyn ei ehditty toteuttaa kohinan suodattamista.

SFM-algoritmien käyttö suuria ympäristöjä mallinnettaessa on teoriassa erittäin hyvin hallittu alue, joten VisualSFM:n suorituskyvyn ongelmat liittyivät lähinnä kappaleessa 3.2 esitettyihin käytännöntoteuttamisen haasteisiin ja toisaalta myös siihen, että VisualSFM on suunniteltu täysin erilaiseen käyttötarkoitukseen, jolloin sen SFM-algoritmi ei ole optimaalinen videokuvaa käytettäessä. VisualSFM:n käyttämä SFM-menetelmä toimii hyvin satunnaisesti poimittujen valokuvien kanssa, mutta videokuvaa käytettäessä se aiheuttaa sen, että mallintaminen vie kohtuuttoman kauan aikaa. Muodostettujen pistepilvien epätarkkuus johtuu puolestaan todennäköisesti

videokuvissa esiintyvistä runsaasta kohinasta. Yksi tärkeä huomio on myös se, että kuvaustavalla on yllättävän paljon vaikutusta pistepilven tarkkuuteen, sillä monen näkymän stereo toimii kohinan vuoksi käytännössä parhaiten tilanteissa, joissa kuvat on otettu suunnilleen samalta etäisyydeltä eri kuvakulmista kuvattuna. VisualSFM:n toimintaan ja aiheen teoriaan tarkemmin perehtymällä saatiin ymmärrys siitä, että millä tavalla SFM-menetelmää olisi mahdollista optimoida videokuvaukseen paremmin soveltuvaksi ja toisaalta myös siitä, että miten AR Dronen sensoreilta saatavaa informaatiota voitaisiin hyödyntää järjestelmän kehittämisessä. Näitä ajatuksia käytiin tarkemmin läpi kappaleessa 6.

Tutkimuksen ja alustavan toteutuksen perusteella on selvää, että järjestelmä on mahdollista rakentaa käyttökelpoiseksi, mikäli ohjelmiston kehittämiseen käytetään tarpeeksi aikaa. AR Drone 2.0 quadrokopteri osoittautui käyttötarkoitukseen soveltuvaksi kuvausrobotiksi, ainakin siinä tapauksessa mikäli siihen vaihdetaan parempi kamera. Kameraan on myös suotavaa lisätä moottori, jonka avulla kameraa voidaan ohjata lennon aikana. Näillä muutoksilla kopterista saadaan erittäin toimiva laite tämänkaltaiseen järjestelmään. Kopterin ominaisuudet mahdollistavat myös paljon nykyistä SFM-menetelmää soveltuvamman ohjelmiston kehittämisen, esimerkiksi hyödyntämällä kopterin sensoreilta saatavaa informaatiota.

Järjestelmän jatkokehittäminen onkin pääasiassa paremmin käyttötarkoitukseen soveltuvan ohjelmiston kehittämistä. Ohjelmiston puutteista ja ongelmista on olemassa hyvä ymmärrys ja tieteestä löytyy menetelmät, joiden avulla ne voidaan ratkaista. Täysin uuden SFM-ohjelmiston toteuttaminen on kuitenkin valtava työ, joten sellainen vaihtoehto tuskin tulee kysymykseen. Yksi vaihtoehto toimivan järjestelmän toteuttamiseen olisi ostaa sellainen kaupallinen ohjelmisto, joka on suunniteltu muodostamaan pistepilvi videokuvan pohjalta. Toisaalta myös VisualSFM on hyvä ohjelmisto pistepilven muodostamiseen, mikäli järjestelmältä ei vaadita nopeaa suoriutumista eikä järjestelmän tarvitse olla täysin automaattinen. Tällöin käyttökelpoisen järjestelmän toteuttamiseen riittää, että VisualSFM:n käyttöön saadaan parempilaatuisia kuvia. Tämän vuoksi toimivan järjestelmän jatkokehittämisessä tärkeintä on datan paremman esi- ja jatkokäsittelyn toteuttaminen. Kohinan vähentämisen tärkeys korostuu paremman laadun saavuttamiseksi ja SFM-vaihetta on mahdollista nopeuttaa siten, että kuvien määrää rajoitetaan hyödyntämällä GPS- ja INS-informaatiota. Tällä tavalla toimimalla hyväksyttävä lopputulos olisi saavutettavissa kohtuullisen pienellä työmäärällä.

Vaikka järjestelmä jäikin vielä käytännössä käyttökelvottomaksi, niin siinä mielessä tässä työssä päästiin kuitenkin tavoitteisiin, että tällainen vaihtoehto järjestelmän toteuttamiseen osoittautui mahdolliseksi. Aiheen laajuuden vuoksi oli alusta asti selvää, että täysin valmista ohjelmistoa ei saada kehitettyä aikamääreessä, joten tärkeintä olikin löytää käyttökelpoinen vaihtoehto tällaisen järjestelmän toteuttamiseen. Tämän diplomityön tuloksena ei saavutettu mitään uutta tieteellistä sisältöä, mutta työ sisältää paljon käytännöllistä näkemystä tämänkaltaisen järjestelmän kehittämisestä. Diplomityössä on perehdytty perusteellisesti teoriaan, jota tällaisen järjestelmän kehittämisessä täytyy ymmärtää, ja myös annettu ajatuksia toteuttamisessa kohdattavien ongelmien ratkaisemiseen. Tämä diplomityö toimii hyvänä pohjana silloin, kun aiheeseen perehtymätön lukija päättää lähteä tällaista järjestelmää toteuttamaan. Työn avulla pääsee nopeasti perille aiheesta, jolloin lukijan ei tarvitse käydä samaa perusteellista tutkimusprosessia lävitse, mitä tämän työn tekemisessä jouduttiin tekemään.

## 8 YHTEENVETO

Tässä diplomityössä esiteltiin alustava toteutus järjestelmälle, jonka avulla ympäristö voidaan 3D-mallintaa quadrokopterilla kuvatun 2D-videon pohjalta. Tutkimuksen ja alustavan kehitystyön tavoitteena oli selvittää, että onko järjestelmän toteutuksessa käytettävä laitteisto soveltuva tällaisen ongelman ratkaisemiseen. Toisaalta tutkittiin myös sitä, että saadaanko järjestelmässä tarvittava ohjelmisto toteutettua kohtuullisella työmäärällä hyödyntämällä avoimeen lähdekoodiin pohjautuvia ohjelmistoja. Yhtenä tavoitteena oli myös selvittää, että mitä ongelmia tällaisen järjestelmän kehittämisessä kohdataan ja antaa suuntaviivoja näiden ongelmien ratkaisemiseksi.

Toteutetussa järjestelmässä käytetään kuvausrobotina AR Drone 2.0 quadrokopterina. Ohjelmiston toteuttamisessa hyödynnetään avoimeen lähdekoodiin pohjautuvia VisualSFM-ohjelmistoa ja OpenCV-kirjastoa, sekä tulosten visualisoinnissa VPython-kirjastoa. Puuttuvat ohjelmiston osat toteutettiin Python-ohjelmointikielen avulla. Järjestelmän toteuttaminen noudattelee käänteisen insinööriyön mukaisia ongelman ratkaisemisen vaiheita. Ensimmäinen vaihe on datan kerääminen, jossa ympäristö videoidaan kopterissa olevan kameran avulla. Seuraava vaihe on datan esikäsittely, jonka tarkoituksena on parantaa käytettävän datan laatua ja myös vähentää datan määrää. Kolmas vaihe on pistepilven muodostaminen, joka toteutetaan VisualSFM-ohjelmiston avulla. Viimeinen vaihe on mallin muodostaminen, jossa pistepilven pohjalta muodostetaan yksinkertaisempi, vokseleista koostuva, malli ympäristöstä. Järjestelmän käyttäminen edellyttää, että käytettävä kamera on kalibroitu, joten kameran kalibrointi suoritetaan ennen muiden vaiheiden suorittamista.

Tutkimuksen ja alustavan toteutuksen perusteella on päädytty johtopäätökseen, että AR Drone 2.0 quadrokopteri on käyttökelpoinen laite tämänkaltaisen järjestelmän toteuttamisessa, mutta täysin itsenäisesti ympäristöä mallintavan järjestelmän toteuttaminen vaatii kuitenkin huomattavan määrän ohjelmiston jatkokehittämistä. Järjestelmässä käytetty VisualSFM-ohjelmisto ei ole suunniteltu käytettäväksi tämänkaltaisessa käyttötarkoituksessa, joten sen avulla ei saada toteutettua täysin itsenäistä ja reaaliaikaista järjestelmää. VisualSFM on kuitenkin käyttökelpoinen silloin, mikäli järjestelmän ei tarvitse suoriutua ongelman ratkaisemisesta täysin itsenäisesti eikä järjestelmältä vaadita reaaliaikaista suoriutumista. Tällaisessa tapauksessa toimivan ohjelmiston toteuttaminen on mahdollista hyvinkin kohtuullisella määrällä jatkokehittämistä. Tärkein yksittäinen asia järjestelmän jatkokehittämisessä on datassa olevan kohinan vähentäminen.

## 9 LÄHDELUETTELO

- [1] S. Schaal, "The new robotics - towards human-centered machines," *HFSP Journal*, osa/vuosik. 1, nro 2, pp. 115-126, 2007.
- [2] T. Várady, R. R. Martin ja J. Cox, "Reverse engineering of geometric models—an introduction," *Computer-Aided Design*, osa/vuosik. 29, nro 4, pp. 255-268, 1997.
- [3] Y. Chen ja G. Medioni, "Object Modeling by Registration of Multiple Range Images," *Image and Vision Computing*, osa/vuosik. 10, nro 3, pp. 145-155, 1992.
- [4] R. B. Rusu, "Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments," 2009.
- [5] H. Woo, E. Kang, S. Wang ja K. H. Lee, "A new segmentation method for point cloud data," *International Journal of Machine Tools and Manufacture*, osa/vuosik. 42, nro 2, pp. 167-178, 1 2002.
- [6] S. Thrun ja J. J. Leonard, "Simultaneous Localization and Mapping," tekijä: *Springer Handbook of Robotics*, Springer, 2008, pp. 871-889.
- [7] S. Piskorski, N. Brulez, P. Eline ja F. D'Haeyer, *AR.Drone Developer Guide SDK 2.0*, Parrot, 2012.
- [8] "VisualSFM : A Visual Structure from Motion System," [Online]. Available: <http://ccwu.me/vsfm/>. [Haettu 11 11 2013].
- [9] G. Bradski ja A. Kaehler, *Learning OpenCV: Computer Vision with the OpenCV Library*, O'Reilly Media, 2008.
- [10] Y. Ma, J. Koseckà, S. Soatto ja S. Sastry, *An Invitation to 3-D Vision: From Images to Geometric Models*, Springer, 2001.
- [11] O. Faugeras ja Q.-T. Luong, *The Geometry of Multiple Images: The Laws that Govern the Formation of Multiple Images of a Scene and Some of Their Applications*, The Massachusetts Institute of Technology, 2001.
- [12] R. Hartley ja A. Zisserman, *Multiple View Geometry in Computer Vision*, Cambridge University Press, 2004.
- [13] M. Pollefeys, "Self-calibration and metric 3D reconstruction from uncalibrated image sequences," K.U.Leuven, 1999.
- [14] M. Pollefeys, "Visual 3D Modeling from Images," University of North Carolina, Chapel Hill, USA, 2004.
- [15] J. Heikkilä ja O. Silvén, "A four-step camera calibration procedure with implicit image correction," tekijä: *Computer Vision and Pattern Recognition*, San Juan, 1997.
- [16] F. Labourie, "What is a Cross Ratio," *Notices of the AMS*, osa/vuosik. 55, nro 10, pp. 1234-1235, 2008.
- [17] M. Ylimäki, "Image-based object modelling from multiple views by efficient and accurate match expansion," University of Oulu, Department of Electrical and Information Engineering, 2011.
- [18] Z. Zhang, R. Deriche, O. Faugeras ja Q.-T. Luong, "A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry," *Artificial Intelligence*, osa/vuosik. 78, nro 1-2, pp. 87-119, 1995.

- [19] Z. Zhang, "A flexible new technique for camera calibration," *Pattern Analysis and Machine Intelligence*, osa/vuosik. 22, nro 11, pp. 1330-1334, 2000.
- [20] H. C. Longuet-Higgins, "A computer algorithm for reconstructing a scene from two projections," tekijä: *Readings in Computer Vision. Issues, Problems, Principles, and Paradigms*, Los Altos, Morgan Kaufmann Publishers, 1987, pp. 61-62.
- [21] T. Huang ja O. Faugeras, "Some Properties of the E Matrix in Two-View Motion Estimation," *Pattern Analysis and Machine Intelligence*, osa/vuosik. 11, nro 12, pp. 1310-1312, 1989.
- [22] P. Torr ja D. Murray, "The Development and Comparison of Robust Methods for Estimating the Fundamental Matrix," *International Journal of Computer Vision*, osa/vuosik. 24, nro 3, pp. 271-300, 1997.
- [23] Z. Zhang, "Determining the Epipolar Geometry and its Uncertainty: A Review," *International Journal of Computer Vision*, osa/vuosik. 27, nro 2, pp. 161-195, 1998.
- [24] R. I. Hartley, "In defense of the eight-point algorithm," *Pattern Analysis and Machine Intelligence*, osa/vuosik. 19, nro 6, pp. 580-593, 1997.
- [25] S. Thrun, "Robotic mapping: A survey," tekijä: *Exploring artificial intelligence in the new millennium*, Morgan Kaufmann, 2002, pp. 1-36.
- [26] K. M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss ja W. Burgard, "OctoMap: A Probabilistic, Flexible, and Compact 3D Map Representation for Robotic Systems," tekijä: *Proceedings of the ICRA Workshop on Best Practice in 3D Perception and Modeling for Mobile Manipulation*, 2010.
- [27] H. Durrant-Whyte, Fellow, IEEE ja T. Bailey, "Simultaneous Localisation and Mapping (SLAM): Part I The Essential Algorithms," tekijä: *IEEE*, 2006.
- [28] M. W. M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte ja M. Csorba, "A Solution to the Simultaneous Localization and map building (SLAM) problem," *Robotics and Automation*, osa/vuosik. 17, nro 3, pp. 229-241, 2001.
- [29] T. S. Huang, "Motion and structure from feature correspondences: a review," *Proceedings of the IEEE*, osa/vuosik. 82, nro 2, pp. 252-268, 1994.
- [30] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision*, osa/vuosik. 60, nro 2, pp. 90-110, 2004.
- [31] B. Triggs, P. F. McLauchlan, R. I. Hartley ja A. W. Fitzgibbon, "Bundle Adjustment — A Modern Synthesis," tekijä: *Vision Algorithms: Theory and Practice*, Springer, 2000, pp. 298-372.
- [32] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein ja R. Szeliski, "A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithms," tekijä: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, New York, 2006.
- [33] C. R. Dyer, "Volumetric Scene Reconstruction from Multiple Views," tekijä: *Foundations of Image Understanding*, Springer US, 2001, pp. 469-489.
- [34] G. Slabaugh, B. Culbertson, T. Malzbender ja R. Schafer, "A survey of methods for volumetric scene reconstruction from photographs," tekijä: *Proceedings of the 2001 Eurographics conference on Volume Graphics*, Aire-la-Ville, 2001.

- [35] T. Krajník, V. Vonásek, D. Fišer ja J. Faigl, "AR-Drone as a Platform for Robotic Research," tekijä: *Research and Education in Robotics - EUROBOT 2011*, Springer, 2011, pp. 172-186.
- [36] D. Maravall, J. d. Lope ja J. P. F. Brea, "A Vision-Based Dual Anticipatory/Reactive Control Architecture for Indoor Navigation of an Unmanned Aerial Vehicle Using Visual Topological Maps," tekijä: *Natural and Artificial Computation in Engineering and Medical Applications*, Springer, 2013, pp. 66-72.
- [37] J. J. Lugo ja A. Zell, "Framework for Autonomous On-board Navigation with the AR.Drone," Springer, 2013.
- [38] J. Engel, J. Sturm ja D. Cremers, "Camera-based navigation of a low-cost quadcopter," tekijä: *Intelligent Robots and Systems (IROS)*, Vilamoura, 2012.
- [39] M. Pollefeys, D. Nistér, J.-M. Frahm, A. Akbarzadeh, P. Mordohai, B. Clipp, C. Engels, D. Gallup, S.-J. Kim, P. Merrell, C. Salmi, S. Sinha, B. Talton, L. Wang, Q. Yang, H. Stewénius, R. Yang, G. Welch ja H. Towles, "Detailed Real-Time Urban 3D Reconstruction From Video," *International Journal of Computer Vision*, osa/vuosik. 78, nro 2-3, pp. 143-167, 2008.
- [40] Y. Furukawa, B. Curless, S. Seitz ja R. Szeliski, "Towards Internet-scale multi-view stereo," tekijä: *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference*, 2010.
- [41] J. Kannala ja S. S. Brandt, "Quasi-Dense Wide Baseline Matching Using Match Propagation," tekijä: *Computer Vision and Pattern Recognition, 2007. CVPR '07*, Minneapolis, 2007.
- [42] P. Koskenkorva, J. Kannala ja S. S. Brandt, "Quasi-dense Wide Baseline Matching for Three Views," tekijä: *Pattern Recognition (ICPR), 2010 20th International Conference*, Istanbul, 2010.
- [43] Acute3D. [Online]. Available: <http://www.acute3d.com/>. [Haettu 11 11 2013].
- [44] Ascending technologies, [Online]. Available: <http://www.asctec.de/>. [Haettu 11 11 2013].
- [45] "2d3 sensing," [Online]. Available: <http://www.2d3.com/>. [Haettu 11 11 2013].
- [46] A. Tikanmäki ja J. Röning, "Development of Mörri, a high performance and modular outdoor robot," tekijä: *Robotics and Automation, 2009. ICRA '09*, Kobe, 2009.
- [47] Parrot, "AR Drone 2.0," [Online]. Available: <http://ardrone2.parrot.com>. [Haettu 24 10 2013].
- [48] D. Claus ja A. W. Fitzgibbon, "A Rational Function Lens Distortion Model for General Cameras," *Computer Vision and Pattern Recognition*, osa/vuosik. 1, pp. 231-219, 2005.
- [49] S. J. Wee, J. G. Apostolopoulos ja N. Feamster, "Field-to-frame transcoding with spatial and temporal downsampling," tekijä: *IEEE*, 1999.
- [50] Autodesk. [Online]. Available: <http://www.123dapp.com/>. [Haettu 11 11 2013].
- [51] 3Dflow. [Online]. Available: <http://www.3dflow.net/technology/samantha-structure-and-motion/>. [Haettu 11 11 2013].
- [52] N. Snavely. [Online]. Available: <http://www.cs.cornell.edu/~snavely/bundler/>. [Haettu 11 11 2013].

- [53] C. Wu, "Towards Linear-time Incremental Structure from Motion," University of Washington, 2013.
- [54] C. Wu, B. Clipp, X. Li, J.-M. Frahm ja M. Pollefeys, "3D Model Matching with Viewpoint-Invariant Patches (VIP)," tekijä: *Computer Vision and Pattern Recognition*, Anchorage, 2008.
- [55] C. Wu, J.-M. Frahm ja M. Pollefeys, "Repetition-based dense single-view reconstruction," tekijä: *Computer Vision and Pattern Recognition*, Providence, 2011.
- [56] C. Wu, S. Agarwal, B. Curless ja S. M. Seitz, "Schematic surface reconstruction," tekijä: *Computer Vision and Pattern Recognition*, Providence, 2012.
- [57] Y. Furukawa ja J. Ponce, "PMVS/CMVS," [Online]. Available: <http://www.di.ens.fr/cmvs/>. [Haettu 31 10 2013].
- [58] Y. Furukawa ja J. Ponce, "Accurate, Dense, and Robust Multiview Stereopsis," *Pattern Analysis and Machine Intelligence*, osa/vuosik. 32, nro 8, pp. 1362-1376, 2010.
- [59] S. Seitz, B. Curless, J. Diebel, D. Scharstein ja R. Szeliski, "Multi-View Stereo," [Online]. Available: <http://vision.middlebury.edu/mview/>. [Haettu 11 11 2013].
- [60] H. Tian, B. Fowler ja A. E. Gamal, "Analysis of Temporal Noise in CMOS Photodiode Active Pixel Sensor," *IEEE Journal of Solid-State Circuits*, osa/vuosik. 36, nro 1, pp. 92-101, 2001.
- [61] S. C. Park, M. K. Park ja M. G. Kang, "Super-resolution image reconstruction: a technical overview," *Signal Processing Magazine*, osa/vuosik. 20, nro 3, pp. 21-36, 2003.

## 10 LIITTEET

- Liite 1. Kalibrointimatriisi ja vääristymäparametrit
- Liite 2. Pistepilven tiedostoformaatti.
- Liite 3. Ympäristön 3D-mallin tiedostoformaatti.
- Liite 4. Yksinkertaistetun 3D-mallin muodostavan ohjelman tarkempi kuvaus.

Liite 1. Kalibrointimatriisi ja vääristymäparametrit.

$$\mathbf{A} = \begin{bmatrix} 1.1097041598e + 03 & 0. & 6.6397237221e + 02 \\ 0. & 1.1076051620e + 03 & 3.2168576207e + 02 \\ \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix}$$

$$\mathbf{D} = \begin{bmatrix} -1.2618887381950910e + 00 \\ 3.9792596214107168e - 01 \\ -9.1332489089328717e - 04 \\ 7.7376648889377079e - 04 \\ -1.0499517943805952e + 00 \\ -7.3205808199795219e - 01 \\ -3.1244045106170870e - 01 \\ -1.1004404549225388e + 00 \end{bmatrix}$$

Liite 2. Pistepilven tiedostoformaatti.

```
ply
format ascii 1.0
element vertex 4
property float x
property float y
property float z
property float nx
property float ny
property float nz
property uchar diffuse_red
property uchar diffuse_green
property uchar diffuse_blue
end_header
-0.146741 0.0369927 0.0629309 0.834882 -0.446203 -0.322296 98 85 115
-0.145773 0.0386061 0.0633188 0.876192 -0.439836 -0.19706 122 104 133
-0.146034 0.0405177 0.0654013 -0.225478 0.147184 -0.963066 123 110 142
-0.141524 0.0456514 0.0655515 -0.254654 0.205618 -0.19706 124 109 139
```

Liite 3. Ympäristön 3D-mallin tiedostoformaatti.

```
txt
format ascii 1.0
element vertex 3
property int occupied
property int x
property int y
property int z
property float red
property float green
property float blue
end_header
1 8 1 0 0.70703125 0.69140625 0.703125
1 11 1 0 0.39453125 0.3671875 0.36328125
1 12 1 0 0.08984375 0.09765625 0.12109375
```

Liite 4. Yksinkertaistetun 3D-mallin muodostavan ohjelman tarkempi kuvaus.

Parametrit:

- Resoluutio: parametrilla säädetään vähiten kuutioita sisältävän (x,y,z)-koordinaattiakselin kuutioiden määrää. Yleensä tämä asettaa y-akselin kuutioiden määrän, eli esimerkiksi arvolla 5 y-akseli jaetaan viiteen yhtä suureen diskreettiin osaan.
- Kynnysarvo: määrittää sen rajan, että kuinka monta pistettä kuution pitää vähintään sisältää, ennen kuin se hyväksytään lopulliseen malliin mukaan.
- PLY-tiedoston nimi: tiedosto sisältää pistepilven ympäristöstä.

Ulostulo: Lopullinen vokseleista koostuva 3D-malli tekstitiedostoon tallennettuna.

Ohjelman toiminta:

1. Asetetaan arvot resoluutiolle ja kynnysarvolle.
2. Avataan PLY-tiedosto ja luetaan pisteiden sisältämä informaatio points[-taulukkoon]. Informaatio talletetaan Point-oliona, joka sisältää attribuutit (x,y,z)-koordinaatit, koordinaattien normaalit sekä RGB-väriarvon. Tämän jälkeen tiedosto suljetaan.
3. Selvitetään minimi- ja maksimiarvot jokaiselle koordinaatille.
4. Selvitetään se koordinaattiakseli, jonka arvoalue on suppein ja lasketaan arvo, jolla koordinaattiakseli tulee jakaa, jotta päästään haluttuun resoluutioon. Tämä tapahtuu seuraavasti:
  - o  $\text{delta\_x} = \text{max\_x} - \text{min\_x}$
  - o  $\text{delta\_y} = \text{max\_y} - \text{min\_y}$
  - o  $\text{delta\_z} = \text{max\_z} - \text{min\_z}$
  - o  $\text{delta\_min} = \min(\text{delta\_x}, \text{delta\_y}, \text{delta\_z})$
  - o  $\text{delta} = \text{delta\_min} / \text{resolution}$
5. Jaetaan ympäristön alue samankokoisiin vokseleihin seuraavalla tavalla:
  - o  $\text{x\_count} = \text{int}(\text{delta\_x} / \text{delta}) + 1$
  - o  $\text{y\_count} = \text{int}(\text{delta\_y} / \text{delta}) + 1$
  - o  $\text{z\_count} = \text{int}(\text{delta\_z} / \text{delta}) + 1$
6. Muodostetaan taulukko buckets[x\_count, y\_count, z\_count] sekä taulukko blocks[[]].
7. Käydään läpi points[-taulukko ja tehdään jokaiselle pisteelle seuraava toiminto:
  - o  $\text{x} = \text{int}((\text{point.x} - \text{min\_x}) / \text{delta})$
  - o  $\text{y} = \text{int}((\text{point.y} - \text{min\_y}) / \text{delta})$
  - o  $\text{z} = \text{int}((\text{point.z} - \text{min\_z}) / \text{delta})$
  - o buckets[z][y][x].append(point)
8. Käydään läpi jokainen buckets[-taulukon solu ja tehdään seuraava:
  - o jos buckets[k][j][i] pisteiden määrä on suurempi kuin kynnysarvo:
    - Lasketaan RGB-arvojen keskiarvo ja skaalataan ne välille 0-1 jakamalla arvot kertoimella 256.
    - Lisätään solu blocks-taulukkoon käyttämällä formaattia, joka on esitetty liitteessä 3.
9. Kirjoitetaan blocks-taulukon sisältämä informaatio tiedostoon käyttämällä formaattia, joka on esitelty liitteessä 3.