Markus Hautala

MATLAB SLAMIEN VERTAILU KÄYTTÄEN LIDAR-ANTURIA ROSIN KAUTTA

Kandidaatintutkielma
Tekniikan ja luonnontieteiden tiedekunta
Tarkastaja: Jouni Mattila
Ohjaaja: Eetu Airaksinen
2/2022

TIIVISTELMÄ

Markus Hautala: Matlab SLAMien vertailu käyttäen lidar-anturia ROSin kautta
Kandidaatintutkielma
Tampereen yliopisto
Automaatiotekniikan tutkinto-ohjelma
Helmikuu 2022

Tämän työn tutkimuskohteena oli SLAMien vertailu valotutka-anturilla. Anturilla luodaan syvyyskuva ympäristöstä, jossa lähetetään ja vastaanotetaan valopulssi. Käytössä oli kolmiulotteinen anturi, joka luo kolmiulotteisen pistepilven ympäristöstä. SLAMilla tarkoitetaan reaaliaikaista ympäristön selvittämistä anturilla. Eri ajanhetkien datapisteiden perusteella voidaan selvittää etäisyyden- ja suunnanmuutos kahden pisteen välillä. Suuri sovelluskohde SLAMissa on ajoneuvo, jossa pisteitä otetaan tietyin aikavälein auton tai työkoneen liikkuessa eteenpäin.

Mittausjärjestelmän rakentaminen sisälsi fyysisen anturoinnin lisäksi testiskriptin ohjelmoinnin. Työssä välitettiin dataa ”Velodyne Puck” -anturista Matlabiin, missä SLAMeja testattiin. Välitys toteutettiin ROSin avulla, joka mahdollistaa paremman testausympäristön uudelleenkäytettävyyden. On mahdollista esimerkiksi välittää dataa jostain toisesta anturista, koska ROS tarjoaa suuren määrän eri ajureita. Työssä vertailtiin eri Matlabin SLAM-toteutuksia. Työhön valittiin kolme eri Mathworksin toteuttamaa esimerkkiskriptiä, joita muokattiin käyttämään omaa dataa. Tavoite oli vertailla eri Matlabin työkalukirjastojen SLAM-toteutuksia siten, että kukin esimerkkiskripti edustaisi yhtä Matlabin työkalukirjastoa. Työn edetessä lähdekoodia tutkiessa huomattiin, että eri kirjastot käyttävät osin samoja SLAM-funktioita, ja sen takia havaittiin yhteneväisyyttä testituloksissa.

SLAM-toteutuksessa on kolme merkittävää osaa. Ensin on esikäsittely, joka on merkittävässä osassa sen takia, koska SLAM on suorituskyvyltään raskas suhteessa nykypäivän laskentatehoon. Tämän takia on kannattavaa karsia mahdollisimman paljon epäinformatiivista dataa. Toinen osa on perättäisien pistepilvien kohdistus eli etäisyyden- ja suunnanmuutos. Kolmas osa on silmukkatarkastelu, joka korjaa virheitä edellisissä kohdistuksissa havaitessaan paikan, jossa on vierailtu aiemmin.

Esikäsittelymenetelmissä testattiin eri asioita, joista yksi oli näytteenottotaajuuden ja pistepilven pisteiden koon suhde. Kun kummassakin oli yhteensä yhtä paljon pisteitä, havaittiin, että se ei tuottanut moninkertaista eroa kartan tarkkuudessa ja suoritusajassa. Toinen testi oli alinäytteistyksen toteuttamisen erot laatikointimenetelmällä ja satunnaismenetelmällä. Tämä testi osoitti pistetiheyden säilyttämisen tärkeyden, johon NDT-kohdistamismenetelmä perustuu.

Tuloksissa todettiin, että kolmesta esimerkkiskriptistä ”Navigation”- ja ”Lidar” -työkalukirjastot suoriutuivat kartan piirron oikeellisuuden perusteella parhaiten. Vastaavasti ”Computer Vision” -työkalukirjasto suoriutui nopeimmin, mutta tämän kartan oikeellisuus oli heikompi. Tässä havaittiin yhtäläisyys suorituskyvyn tarpeellisuudessa kartan oikeellisuuteen nähden.

Avainsanat: SLAM, Lidar, Matlab, ROS

ABSTRACT

Markus Hautala: Comparision of Matlab SLAMs using lidar sensor through ROS
Bachelor's thesis
Tampere university
Degree Programme in Automation Engineering
February 2022

The subject of this work was to compare SLAMs with lidar sensors. With the sensor, it can create a depth picture from the environment, where the light impulse is emitted and transmitted by the lidar sensor. In that work was used 3d-sensor that create a 3d point cloud from the environment. SLAM means real-time mapping between two points. The most useful application with SLAM is the traffic, where the points are token between time differences while the vehicle is moving.

The construct of the measurement setup in that work consists of physical sensor and test script programming. In that work, we forwarded the data from the “Velodyne Puck” sensor to the Matlab where SLAMs were tested. The data forward implemented with the ROS allows better reuse of the test environment. It is possible for example to forward the data from some other sensor because ROS offers a really big number of different drivers. In that work was compared different Matlab SLAM-methods. In that work was elected three different example scripts were produced by Mathworks. The aim of that was to compare different SLAM implementations of the Matlab toolboxes so that each of the example script would represent one Matlab toolbox. As the work progressed while researching the source codes was noticed that different toolboxes use partly the same functions. That caused the similarity in the test results.

The SLAM method is three remarkable parts. The first is a preliminary process that is a remarkable role because SLAM is heavy performance compared with nowadays computing performance. That is why it is worth reducing as much irrelevant data as possible. The second part is the registration (difference with distance and rotation) of the consecutive point clouds. The third part is loop detection which fixes errors in previous registration when detecting the place that has been visited before also.

In the preliminary process, it was tested different things were the one was the relation between sample rate and amount of point cloud points. When both tests had an equal number of points overall, it was noticed that it didn’t cause multiple differences with map accuracy and compute times. The other test was downsampling between the grid and the random method. That test pointed out the importance of the point cloud density information because the NDT method is based on that.

The research pointed out that within three example scripts the Navigation and Lidar toolbox examples carried out the best based on map accuracy. Unlike Computer Vision Toolbox example script was carried out the most quickly but the map accuracy was weaker.

Keywords: SLAM, Lidar, Matlab, ROS

ALKUSANAT

Haluan kiittää erityisesti työni tarkastajaa, professori Jouni Mattilaa, joka ehdotti aiheen valintaprosessissa todella monipuolisia ja mielenkiintoisia aiheita. Tämä auttoi löytämään tämän mielenkiintoisen aiheen antureista.

Toinen kiitos menee työni ohjaajalleni Eetu Airaksiselle, joka on tukenut minua tässä työssä ohjaten. Olen saanut häneltä nopeasti apua, kun olen sitä tarvinnut. Ohjaussessiot ovat tuoneet motivaatiota työn tekemiseen.

Tampereella, 27.2.2022

Markus Hautala

SISÄLLYSLUETTELO

LYHENTEET JA MERKINNÄT

SLAM	engl. Simultaneous localization and mapping
Lidar	Valotutka-anturi
ROS	Robot Operating system
Message	ROSin välillä kulkevat viestit
Node	prosessi, ROSin laskentayksikkö
Topic	väylä, ROSin viestien tietotyyppi ja nimi
Subscriber	tilaaja, ROSin laskentayksiköiden lähettämä viestipyyntö muilta laskentayksiköiltä
Publisher	julkaisija, ROSin laskentayksiköiden lähettämä viesti muille laskentayksiköille
Built-in	Sisäänrakennettu
Esikäsittely-menetelmä	engl. preliminary processing
Alinäytteistys	engl. downsampling; datan vähentäminen
Kohdistamis-algoritmi	engl. registration algorithm; etäisyyden- ja suunnanmuutoksen selvittäminen
Silmukkatar-kastelu	engl. Loop detection
ICP	engl. iterative closest point; kohdistamisalgoritmi
NDT	engl. normal distribution transform; kohdistamisalgoritmi
IMU	engl. inertial measurement unit
Mathworks	Yritys, joka on tehnyt Matlabin
Matlab Toolbox	Matlabin työkalukirjasto (esimerkiksi Navigation Toolbox)
Matlab	Matrix Laboratory; tietokoneohjelma

1. JOHDANTO

Työkoneita käytetään muun muassa rakennustöissä ja maa- ja metsätaloudessa. Automaation avulla saadaan parannettua työn tehokkuutta joko ihmistä avustavin keinoin tai täysin automatisoidusti. [1, s. 1-4]

Automaation lisääntyessä tarvitaan anturointia. Tällä kerätään tietoa ympäristöstä, jonka avulla saadaan toiminnallisuutta aputoimintoihin. Anturit mittaavat eri asioita ja perustuvat eri suureiden mittaamiseen jollain teknisellä menetelmällä. Tässä työssä käsitellään valotutkaa (engl. lidar), joka mittaa laserin avulla etäisyyttä. Kun etäisyyttä mitataan laserin avulla useasta pisteestä, saadaan luotua kuva ympäristöstä. Mittausdataa kutsutaan pistepilveksi. Pistepilvi itsessään ei vielä tuota suoraan hyödynnettävää tietoa ympäristöstä. Algoritmien avulla voidaan päätellä sijainti ympäristön perusteella. Tätä kutsutaan SLAMiksi. Tätä voidaan käyttää hyödyksi myös työkoneen automaatiossa [2].

Tässä työssä toteutettiin SLAM-algoritmien vertailua Matlabilla Velodyne Puck -anturin datan avulla. Anturista luettiin dataa ROSin avulla, joka on robotiikkaan tarkoitettu väliohjelmistopaketti. Tässä työssä sitä käytettiin datan lukemiseen Puckista, jonka jälkeen se välitettiin Matlabiin. ROSia hyödynnettiin tässä työssä vähän suhteessa siihen, mihin sitä olisi voinut hyödyntää. Se otettiin sen takia käyttöön, jotta siitä olisi ollut hyötyä jossain muussa yhteydessä esimerkiksi jatkokehityksen kannalta.

Työssä kokeiltiin erilaisia SLAM-algoritmeja ja vertailtiin näitä keskenään. Algoritmien valintaperusteena oli niiden yhteensopivuus Matlabiin. Lisäksi valittavien algoritmien oli oltava riittävän kehittyneitä ja vertailukelpoisia toisiinsa nähden. Algoritmien välillä vertailtiin erityisesti suoritus- ja paikannuskykyä.

2. TEORIA

Tässä luvussa käsitellään valotutkan, ROSin ja SLAMin teoriaa. Nämä kolme asiaa ovat työssä merkittävässä kytköksessä toisiinsa, mutta jokainen aihealue muodostaa silti oman osuutensa.

2.1 Lidar

Lidar tulee sanoista ”Light Detection And Ranging” [3]. Valotutka-anturi lähettää laservalopulssin, jonka se vastaanottaa heijastuksesta. Lähetyksen ja vastaanottamisen välisestä aikaerosta saadaan selville etäisyys kohteeseen, joka voidaan laskea kaavalla

$$d = c * \frac{t}{2} \quad (1)$$

missä c on valon nopeus ja t kulutettu aika laserimpulssin lähettämisen ja vastaanottamisen välissä. [4]

Tällä tavoin toimii 1D-anturi. Anturi lähettää yhteen pisteeseen valopulssin, joka heijastuu takaisin. Kun halutaan mitata 2D ympäristö, eli esimerkiksi rakennuksen seinien pinnanmuodot vaakatasossa, voidaan lisätä mikroskannerin (MEMS mirror) avulla 2D-anturi. 3D-ympäristöä luodessa voidaan heijastaa valopulssi moottoroidun pyörivän peilin avulla eri pisteisiin. [5, s. 1-4]

Kun 1D-valotutka saa selville etäisyyden d tiettyyn kohteeseen, muodostuu tästä yksi pistepilven piste. 3D-valotutka saa selville siten useita pisteitä, ja tästä rakentuu usean pisteen pistepilvi. Pistepilvi tarkoittaa x-y-z-avaruudessa kuvattuja pisteitä, missä jokaisella pisteellä on omat paikkakoordinaatit suhteessa anturiin.

Pistepilvi muodostuu yhden aikahetken mittauksesta. Kun on monta aikahetkeä, muodostuu useita pistepilviä. Kun anturi on suhteellisessa liikkeessä ympäristöön aikaan nähden, pistepilvet ovat eri ajanhetkellä erilaiset.

2.2 ROS

Robotista puhuttaessa tarkoitetaan yleensä kokonaisuutta, missä on useita antureita, toimilaitteita ja prosessointia. Komponentit voivat olla useilta eri valmistajilta. Mittaus- ja ohjausdatan on kuitenkin liikuttava joustavasti eri osien välillä. Laskentaa on kyettävä tekemään eri paikoissa, kuten eri tietokoneilla. Lisäksi modulaarisen kokonaisuuden ehtona on uudelleenkäytettävä ohjelmakoodi, joka voi olla esimerkiksi Kalman-suodatin tai PID-säädin.

Näistä tarpeista on kehittynyt ROS (Robot Operating System), joka on avoimeen lähdekoodiin perustuva työkalu järjestelmän hallitsemiseen kokonaisuudessaan. Se tarjoaa suuren määrän eri raudoille soveltuvia ajureita. Sen avulla voidaan välittää joustavasti dataa komponenttien välillä. Se toimii ikään kuin siltana kahden solmun, esimerkiksi etäisyysanturin ja moottorin välillä. [6] [7]

2.2.1 ROS-laskentayksikkö

ROS koostuu laskentayksiköistä (engl. node), jotka suorittavat prosessointia. Prosessointi voi olla esimerkiksi datan välittäminen anturin ajurin kautta toimilaitteelle. Toimilaite voi olla esimerkiksi moottori. Laskentayksikkö voi olla myös datan prosessointia jossain raudassa. [8]

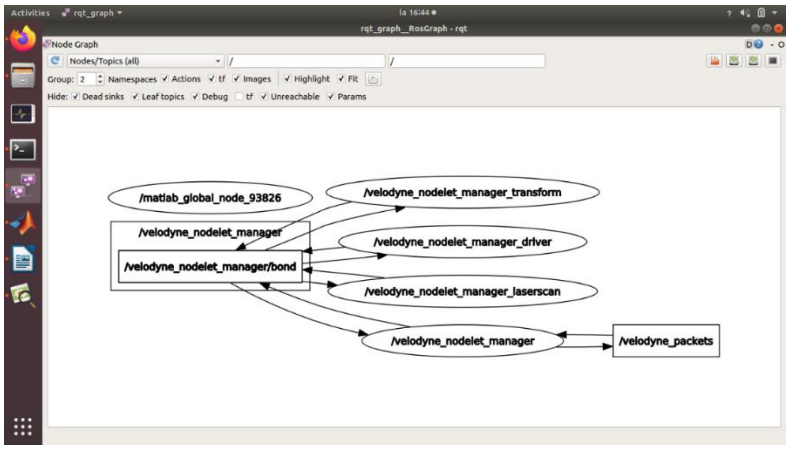
ROSin hyvä puoli on siinä, että se voidaan hajauttaa yksittäisiin laskentayksikköihin eli nodeihin. Kun laskenta on hajautettu, ei koko järjestelmän toiminta ole yhden laskentayksikön varassa, jonka kaatuessa koko järjestelmän toiminta pysähtyisi. [9]

2.2.2 ROS-väylä

Node-laskentayksiköiden välillä kulkee dataa, joita kutsutaan viesteiksi (engl. message). Jotta viesti voi liikkua laskentayksiköiden välillä, tarvitaan kullekin viestille oma väylä (engl. topic), jolla on oma tietotyyppi ja nimi. Väyliä ja viestejä voi olla monia laskentayksiköiden välillä.

Laskentayksiköt lähettävät ja vastaanottavat vain niiden tarvitsemiansa väyliä. Tämä tapahtuu siten, että laskentayksikön toimesta lähetetään tilauspyyntö (engl. subscriber). Vastaavasti laskentayksiköiden toimesta lähetetään vain niiden tuottamia väyliä, joka tapahtuu julkaisemalla (engl. publish) dataa ROS-verkkoon. Tilaus- ja julkaisemismenetelmän avulla laskentayksiköt ovat anonyymejä toisille, ja julkaistessaan dataa laskentayksikkö ei tiedä vastaanottavan laskentayksikön identiteettiä. [10]

ROSissa on oma työkalu ”rqt_graph” laskentayksiköiden ja väylien havainnollistamiseen [11]. Tässä työssä käytetyssä ROS-konfiguraatiossa, jossa data tulee Velodynen anturista ja päätyy Matlabiin, työkalu näyttää kaavion, joka on esitetty näyttökuvassa Kuva 1.



Kuva 1. *rqt_graph työkalun esittämä kaavio käytettävästä koejärjestelystä*

Kullakin väylällä on uniikki nimi ja tietotyyppi. Väylät ovat yleensä sovelluskohtaisia. Viesti voi sisältää useita sisäänrakennettuja (engl. built-in) tietotyyppejä. Lista ROSin sisäänrakennetuista tietotyypeistä löytyy lähteestä [12].

Velodyne-anturin laskentayksikön dataa julkaisevalla väylällä ”/velodyne_points” on oma viestin tyyppi ”sensor_msgs/PointCloud2”, joka sisältää lähteen [13] listassa esitettyjä tietotyyppejä. Listan tietotyypeistä saa komennolla ”rosmsg info sensor_msgs/PointCloud2”.

Data laskentayksiköiden välillä kulkee TCP- tai UDP-protokollien kautta. Tämä ei ole käyttäjälle välttämättömän huomionarvoista, koska ROS hoitaa itse tämän automaattisesti. Kullakin laskentayksiköllä on oma IP-osoite, joka näkyy esimerkiksi siinä, että alustusta tehdessä tarvitsee määritellä IP-osoite. [14] Käytännössä tässä työssä tämä osoite täytyy määrittää Matlabissa.

2.3 SLAM

SLAM tulee sanoista ”Simultaneous Localisation And Mapping”. Se tarkoittaa ympäristön havainnointia eri antureilla, jonka perusteella voidaan selvittää anturin sijainti. [15] SLAMin tavoitteena on kerätä anturidataa kahdesta eri pisteestä, jonka jälkeen algoritmien avulla saataisiin selville kahden eri pisteen sijainnin muutos. Eniten toteutettuja tutkimuksia löytyy kameraan tai valotutka-anturiin perustuen [16], mutta lähteenä voi toimia myös esimerkiksi ääni [17] tai langaton lähiverkko [18]. Päätavoitteena on sama – saada paikannustieto anturin avulla.

Sana ”simultaneous” tulee siitä, että algoritmin on tavoite tehdä paikannusta (localisation) ja ympäristön havainnointia (mapping) yhtä aikaa. Haaste on siinä, että saman aikaisesti pitäisi selvittää sijainti ympäristöstä, mutta sijainnin selvittämiseksi tarvitaan kartta ympäristöstä. [19, s. 3]

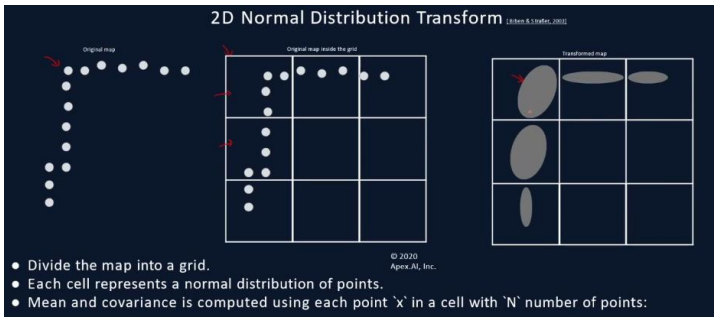
Kuvaan perustuvaa SLAMia kutsutaan visual SLAMiksi. Se perustuu siihen, että yhden (monocular) tai kahden kameran (stereocular) avulla algoritmi erottaa ympäristön elementit. Kameraan verrattuna valotutka-anturi on erilainen, koska se perustuu etäisyyden tunnistukseen. Kamera perustuu vastaavasti elementtien tunnistamiseen värien ja rajojen erottamiseen perustuen. [15]

SLAM-algoritmin perustana on tunnistaa etäisyyden- ja asennonmuutos kahden eri pistepilven välillä. Ilman tätä reitin piirto kartalla ei onnistuisi. Eri SLAM-toteutukset käyttävät eri algoritmeja. Tässä luvussa käydään läpi yleisimmät kohdistamisalgoritmit (engl. registration algorithms). Kohdistamisalgoritmit selvittävät sekä etäisyyden- että suunnanmuutoksen. Suomenkielinen termi sanalle ”kohdistamisalgoritmit” on otettu lähteestä [20].

2.3.1 Kohdistamisalgoritmit

ICP (engl. iterative closest point) on yksi kohdistamisalgoritmi. Se perustuu siihen, että otetaan valikoima pisteitä pistepilvestä, ja yhdistetään pisteet yhteen ryhmään. Tämän jälkeen verrataan kahden eri pistepilven välistä etäisyydenmuutosta. Etäisyyttä iteroidaan eli toistetaan laskentaa niin kauan kunnes löydetään mahdollisimman hyvä tulos, joka täsmää kaikkiin pistepilven ryhmiin. [21] [22] Matlabin Computer vision toolbox käyttää ICP-menetelmää selvittääkseen paikka- ja suuntamuutosta kahden eri pistepilven välillä. Tätä menetelmää käyttävän funktion nimi on ”pcregistericp”.

NDT (engl. normal distribution transform) on toinen hyvin yleisesti käytetty menetelmä, joka perustuu todennäköisyysjakaumaan. Pistepilvi jaetaan osiin, ja lasketaan todennäköisyysjakauma pisteiden jakautumisesta x-y-z-avaruudessa. Tavoitteena on selvittää kunkin jaetun osan sisällä olevien pisteiden jakautumisen muoto. Alla olevassa kuvassa Kuva 2 on havainnollistus tästä todennäköisyyden laskemisesta. Tällä tavoin pistepilvestä muodostuvaa todennäköisyyskarttaa voidaan verrata toiseen todennäköisyyskarttaan. Tämä ei toteuta vielä yksinään etäisyyden- ja suunnanmuutosta pistepilvien välillä, mutta tällä tavoin saadaan ryhmiteltyä pistepilvien pisteet osiin. Etäisyyden- ja suunnanmuutos laskentaan käytetään matemaattisia menetelmiä kuten Newtonin algoritmia. [23]



Kuva 2. *NDT-menetelmän todennäköisyysjakauma* ***[24]***

2.3.2 Silmukkatarkastelu

SLAM-algoritmin merkittävä osa on silmukkatarkastelu (engl. loop detection). Tämän toiminnon tehtävänä on selvittää uudesta vastaan tulevasta pistepilvestä ennalta havaittuja elementtejä. Jos anturin kanssa pyöritään ympyrän muotoista reittiä, algoritmin tulisi tunnistaa samat elementit edelliseltä kierrokselta. Kun havaitaan pistepilvi, joka muistuttaa jotakin edellistä pistepilveä, voidaan laskea näiden pistepilvien suhteellinen kohdistus uudestaan. Tätä verrataan siihen kohdistustietoon, joka on saatu edellisten pistepilvien suhteellisien kohdistuksien summana. Tällä laskettujen kohdistuksien erotuksella voidaan selvittää ajelehtiminen. Tämän jälkeen voidaan korjata ajelehtiminen siten, että muodostunut silmukka alkaa ja loppuu samassa pisteessä.

Silmukkatarkastelumenetelmiä on monia tapoja. Mikäli laskentatehoa olisi rajattomasti, toteutettaisiin skripti, joka tarkistaa jokaisen aikaisemman pistepilven nykyiseen kyseisellä hetkellä tarkasteltavaan pistepilveen. Tässä resurssitarve kasvaisi eksponentiaalisesti. Tätä varten on menetelmä, joka laskee kohdistuksen uudestaan vain niistä pisteistä, jotka ovat tietyn säteen sisällä. [25] [26, s. 120-121]

2.4 Tietotyypit

Mathworksin tekemät omat Matlabin kirjastot tarjoavat SLAM-algoritmeja, jotka perustuvat sekä valotutkaan (lidar SLAM) että valokuvaan (visual SLAM). Valotutka voi olla joko 2D tai 3D [27], ja käytössä ollut Velodynen anturi on 3D. Yksi vertailtava asia on se, mikä algoritmi käyttää 2D- ja mikä 3D-dataa. Tämä edellyttää valotutka-anturin tuottaman pistepilven muuntamista 2D-dataa käsittelevälle algoritmille soveltuvaan muotoon.

Pistepilvi sisältää syvyyskartan ympäristöstä, ja tätä aihetta yleisellä tasolla on käsitelty tässä työssä aiemmin. Matlab käsittelee oletusarvoisesti tietotyyppiä ”pointCloud”, mutta ROS välittää tietotyyppiä ”pointCloud2”. Tätä varten on olemassa tyyppimuunnosalgoritmi lähteessä [28]. Muutos tapahtuu syntaksilla

```
pointCloud = pointCloud(readXYZ(pointcloud2))
```

missä ”pointCloud2” on ROSista tuleva pistepilvi nimensä mukaisella tietotyypillä ja ”pointCloud” on Matlabin yleisesti käyttämän tietotyypin mukainen pistepilvi.

SLAM-algoritmin toiminnan idea on se, että sisääntulona on pistepilvien joukko ja ulostulona on sijaintitietojen ja asennon muutos pistepilvien välillä x-, y- ja z-akselien suhteen. Tähän tarvitaan tietorakenteita, ja esimerkiksi Matlabin Computer Vision Toolboxissa on toteutettu tietorakenne ”pcviewset”, joka sisältää nämä vaadittavat tietorakenteet yhdessä objektissa. Tähän tietorakenteeseen voidaan asettaa komennolla ”addView” pistepilvi. Pistepilvien välinen etäisyyden- ja suunnanmuutos saadaan eri algoritmeilla, joita on lueteltu alaluvussa 2.3. Näillä algoritmeilla saadaan suhteellinen muutos, joka on tietotyyppiä ”rigid3d”. [29] Perättäiset pistepilvet voidaan kytkeä yhteen komennolla ”addConnection”, ja tällöin tallennetaan suhteellinen suunnanmuutos tietorakenteeseen ”pcviewset”.

3. TESTAUSYMPÄRISTÖ

Työssä vertailtiin eri SLAM-algoritmeja Matlabissa siten, että Matlab vastaanottaa anturin dataa Velodyne Puck -anturista. Väliohjelmistona käytettiin ROSia. ROS vastaanottaa UDP-protokollan avulla anturidataa ja lähettää datan Matlabin ROS-liitännäiseen.

Tällä kokoonpanolla on mahdollisuus tehdä SLAM-skriptistä reaaliaikaisesti toimiva. Se tarkoittaa sitä, että skripti voi piirtää karttaa samaan aikaan, kun anturi kerää dataa. Reaaliaikaisuus ei tapahdu silloin, jos ensin kerätään dataa ja tämän jälkeen syötetään SLAM-algoritmille tallennettu data.

Käytännössä tässä työssä toteutettiin skripti, joka ei kerää samaan aikaan dataa anturista. Tämän avulla saatiin toteutettua tasapuolinen testitulos eri SLAM-algoritmien välillä. Mikäli skriptiä ajettaisiin reaaliajassa, perustuisi algoritmin suorituskykytestaus käytettävän tietokoneen suorituskykyyn. Kun skripti ajetaan tallennetulle datalle, voidaan suorituskykyeroja vertailla suhteellisiin aikoihin. Anturidatan tallennus tehdään Matlabin muuttujaan. Tällöin myöskään ROSin suorituskyky dataa lukiessa ei vaikuta.

Käytännössä skriptiä on mahdollisuus muokata sellaiseksi, että se soveltuu reaaliaikaiseen laskentaan. Toteutuksessa, jossa SLAM-skripti käsittelee tallennettua dataa, on käytössä while- tai for-operaattorilla toteutettu silmukka, joka käy järjestyksessä tallennetut pistepilvet läpi. Käytännössä muokkaus tapahtuu siirtämällä SLAM-skripti tallennetun datan silmukasta ROSista dataa keräävään silmukkaan.

3.1 Käyttöjärjestelmät, ohjelmistot ja ajurit

ROS toimii parhaiten Linuxilla, koska se on kehitetty tähän käyttöjärjestelmään [30]. Sen takia valittiin käyttöön Ubuntu, koska se on kandidaatintyön tekijälle tuttu sekä se on suositeltu myös ROSin alustaksi. ROS-versioksi valittiin ”Melodic”. Tähän versioon suositellaan Ubuntun versiota 18.04. Tämä käyttöjärjestelmä ladattiin lähteestä [31].

ROSin asennus tapahtuu ROS wikin asennusohjeita seuraamalla [32]. ROSin asennuksen jälkeen asennetaan Matlab 2021b, johon asennetaan lisäosaksi ROS Toolbox, Computer Vision toolbox, Navigation toolbox ja Lidar toolbox. Työssä datan keräys toteutettiin Linuxilla, mutta SLAMien vertailuvaiheessa käytettiin Windowsiin asennettua Matlabia.

Datan vastaanottaminen Velodyne Puck -anturista ROSiin tapahtuu Velodynelle tehdyn anturin avulla [33]. Asennus tapahtuu ohjeita seuraamalla. Ensin ladataan ajuri Githubista [34], jonka jälkeen rakennetaan ”build”-hakemisto komennolla ”catkin_make”.

3.2 Velodynen ja ROSin välinen dataliikenne

Velodyne Puck -anturi lähettää UDP-väylää pitkin paketteja, jotka sisältävät mittausdataa [35, s. 18]. Tämä data voidaan vastaanottaa toisella verkkolaitteella. Ero TCP:n ja UDP:n välillä on siinä, että UDP ei vahvista lähettäjälle pakettien saapumisesta vastaanottajalle. UDP lähettää siten jatkuvaa virtaa eikä siihen vaikuta se, vastaanottaako paketteja mikään verkkolaite, kuten tässä tapauksessa ROSin Velodynen ajuri. [36, s. 34]

ROSissa käynnistetään Velodynen laskentayksikkö. Tämä vastaanottaa dataa UDP:lla IP-verkossa ja lähettää viestejä ROSin verkkoon. Sen jälkeen voidaan vastaanottaa viestejä toisessa laskentayksikössä, joka on tämän työn tapauksessa Matlab.

3.3 MATLAB ja sen ROS-liitännäinen

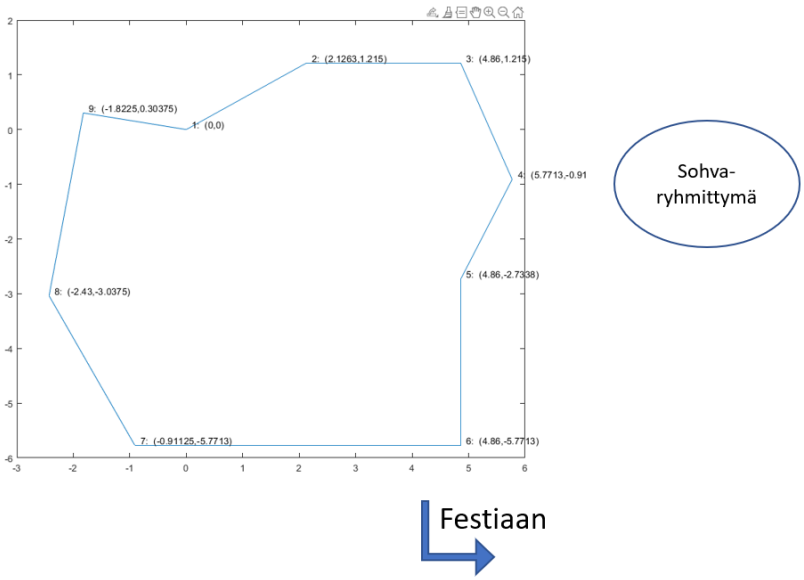
Matlabissa ROS käynnistetään komennolla ”rosinit(<ip_address>)”, missä ip_address on Matlabin käyttämän tietokoneen verkkokortin IP-osoite. Käytännössä tässä työssä Matlab ja ROS ovat samassa koneessa, minkä takia IP-osoite oli ”127.0.0.1”, joka viittaa kyseiseen laitteeseen.

Matlabissa ROS-viestejä luetaan käyttämällä komentoa ”rossubscriber(topic)”. Termi ”topic” on ROSin väylä. Kun halutaan hakea Velodyne Puck -anturin dataa, valitaan väyläksi ”velodyne_points”. Komennon paluuarvoksi saadaan Matlabin tietotyyppiä vastaava ”Subscriber”, jonka sisällä on ”LatestMessage”-niminen muuttuja, joka vastaa tietotyyppiä ”topic”. Velodynen tapauksessa se on tietotyyppiä ”PointCloud2”.

3.4 Näytteenoton testausympäristö

Testaus tapahtui Tampereen yliopiston Konetalon aulassa. Valokeila-anturi asetettiin kärryn päälle (Kuva 3). Anturille tuotiin sähköjohto ja Ethernet-kaapeli. Lattiaan merkattiin teipillä pisteet, joiden kautta kärryä kuljetettiin. Alla olevassa kuvassa Kuva 4 on lattiamerkintöjen sijainti. Sijaintitiedoissa käytettiin hyväksi lattiakaakeleiden ruutuja. Kaakelit olivat neliön muotoisia. Kahdeksan kaakelin mitta oli 243 cm, jonka avulla saatiin suhteutettua kaakelin mitta metreihin. Mittaustarkkuus oli 1 cm.

Kuva 3. *Testausympäristö*



Kuva 4. *Lattiamerkintöjen sijainti (m)*

Matka aloitettiin kohdasta 1 (0;0) m. Kärryn kanssa kierrettiin myötä päivää ensin yksi kerros ympäri, ja sen jälkeen jatkettiin kierrosta pisteeseen 6 (4,86; -5,7713) m. Tällöin saatiin toteutettua silmukkatarkastelu.

Anturin tarkkuutta tutkittiin siten, että anturista mitattiin etäisyys seinään mittanauhalla ja katsottiin pistepilvikuvasta etäisyys samaan pisteeseen. Matlabissa yksikkönä käytetään metrejä, ja tarkkuus oli muutama senttimetri. Muodostuvien karttojen yksikkö on siten metri.

Käyttämä pöytä oli tukeva ja tasainen. Kun yksittäistä pistepilveä tarkasteltaessa etsitään lattiaheijastuksen korkeuskoordinaatteja, huomataan niiden olevan väliltä -88 cm ja -92 cm välillä pystyakselin suuntaisesti.

4. VERTAILU

Vertailu ei keskity SLAM-algoritmien soveltuvuusvertailuun siinä, kuinka paljon paremmin jokin toinen algoritmi suoriutuu esimerkiksi jalankulkijan tunnistamisesta liikenneolosuhteissa. Työ keskittyy sitä vastoin algoritmien ominaisuuksien vertailuun sisätilassa, joka sisältää muun muassa sitä, mitä tietorakenteita jokin algoritmi ottaa parametrikseen ja mitä toiminnallisuuksia algoritmeissa on. Lopputuloksessa vertaillaan pääsääntöisesti kahta asiaa: suorituskyky ja paikannuskyky eli kartan oikeellisuus.

Testin kulku toimii seuraavasti. Ensin testataan erilaisia esikäsittelymenetelmiä (engl. preliminary processing). Niiden tavoitteena on parantaa pistepilven oikeellisuutta ja vähentää suorituskykytarvetta [37]. Testaus toteutetaan tarkastelemalla pistepilviä, eli tässä vaiheessa ei ajeta vielä SLAM-skriptiä. Paras esikäsittelymenetelmäkombinaatio SLAMin testaukselle valitaan arvioimalla silmämääräisesti. Valitulla esikäsittelymenetelmäkombinaatiolla testataan kaikki SLAM-skriptit, jonka jälkeen parhaiten suoriutuneella SLAM-skriptillä vertaillaan eri esikäsittelymenetelmien kombinaatiot. Kun paras esikäsittelymenetelmäkombinaatio on valittu, testataan SLAM-skriptit vielä kertaalleen.

SLAM-algoritmien testauksessa arvioidaan suoritusajan lisäksi kartan oikeellisuutta. Tätä voidaan arvioida hyvin monella eri tavalla. Tähän työhön valittiin mittaustavaksi kaksi menetelmää.

Ensimmäinen menetelmä arvioi, kuinka hyvin SLAM-algoritmi osaa paikantaa koordinaatit. Tämä testi toteutetaan siten, että haetaan esikatselemalla pistepilvien joukosta se pistepilvi, jossa ollaan lattiamerkin kohdalla. Tämän avulla haetaan kyseisen pistepilven kohdalta SLAMin laskema absoluuttinen paikkakoordinaatti. Tästä saadaan poikkeama etäisyytenä. Tässä menetelmässä tiedostetaan, että kaikkien pisteiden etäisyys riippuu siitä, missä suunnassa anturi on z-akselin suhteen alkupisteessä. Mikäli anturi osoittaa hiemankin vinoon, on ennustettu, että Konetalon tolpan toisessa päässä ollessa etäisyysero todelliseen lattiapisteeseen on suurin, koska kiertymä vaikuttaa kaikkein eniten kauimmaisessa pisteessä. Tämä ei kuitenkaan vaikuta suoranaisesti vertailuprosessiin, koska käytössä ollut testidata on sama kaikkien eri testattavien esimerkkiskriptien välillä.

Toinen menetelmä on vertailla, kuinka paljon syntyy ajelehtimista reitissä. Anturia on liikuteltu suoraan lattiamerkkien välillä. Mikäli pisteiden koordinaatit ovat vääristyneet, muodostuu pisteiden välille yhteenlaskettuna suurempi etäisyysero kuin jos matka kahden lattiamerkin välillä olisi suora. Käytännössä lasketaan Pythagoraan lauseella eri pistepilvien väliset suhteelliset etäisyydet yhteen.

Jokainen testattava esimerkkiskripti käyttää erilaisia menetelmiä silmukkatarkasteluun. Tämä vaikuttaa testituloksiin, ja näitä arvoja vertaillaan.

Tämä raportti sisältää testitulokset testauksesta. Testidata, algoritmit ja testin kulku ovat saatavissa Github-repositoriossa, joka löytyy osoitteesta https://github.com/markus-hautala/Matlab-SLAMs. Github repositorion olemassaoloa ei kuitenkaan voida varmistaa tulevaisuuteen, jonka takia tämän työn loppuun on lisätty skriptit, jossa on pistepilvien käsittelyyn liittyvät oleellisimmat koodit. Erilaisia apufunktioita ei ole lisätty sinne. Tiedosto ”start_here.mlx” sisältää testien kokonaiskulun.

4.4 Häiriösuodatus

Anturin tuottamassa pistepilvessä voi olla virheellisiä pisteitä, jotka johtuvat häiriötekijöistä. Se näkyy valotutka-anturin tapauksessa yksittäisinä satunnaisina pisteinä pistepilvessä. Tätä voidaan vähentää suodattamalla satunnaiset pisteet pois. Matlabissa on funktio

```
häiriösuodatettu_pistepilvi = pcdenoise(pistepilvi)
```

jonka lähdekoodin mukaan algoritmi ”poistaa satunnaiset pisteet, joiden etäisyys lähimmäiseen pistejoukkoon x on yli kynnysarvon y”. Nämä arvot ovat parametrejä, jotka voivat valita tilanteesta riippuen. Pistejoukon y koko on oletusarvoisesti arvo 4 ja etäisyyden kynnysarvo arvo 1,0 metriä.

Matlabin häiriösuodatusta kokeiltiin datalle, joka taltioitiin aulassa (Kuva 5), missä oli mahdollisimman suuri etäisyys eri kohteisiin. 24942 pisteen joukosta hävisi 35 pistettä pois. Havaitaan siis, että tämä algoritmin suodattama häiriödata oli suhteellisen pieni.

Kuva 5. *Yksittäisen pistepilven tarkasteluun valittiin Konetalon aula, missä olisi mahdollisimman suuri etäisyys eri suuntiin.*

Mikäli haluttaisiin selvittää tarkemmin soveltuvat parametrit algoritmille, kannattaisi suorittaa mittaus tilassa, missä tiedetään eri elementtien kuten seinän ja katon sijainti. Tämän testin avulla kyseiselle anturille voisi saada selvitettyä tarkemmin sen, kuinka suuri on keskimäärin anturin ominaishäiriökohina. Sen perusteella tiedettäisiin paremmin, kuinka paljon anturin ominaishäiriökohinaa tulisi poistaa eli mitä parametreja olisi hyvä käyttää funktiossa.

4.5 Alinäytteistys

Pistepilvi sisältää ison määrän pisteitä, ja se ei ole monessa tapauksessa kuten pienessä huoneessa tarpeellista. SLAMissa on tärkeä havaita yksittäiset elementit kattavasti ympäristöstä. Myös elementtien muoto on tärkeä. Sen takia on tärkeä saada riittävästi pisteitä pistepilveen. On kuitenkin muistettava, että pistepilven koko vaikuttaa merkittävästi SLAMin laskentatarpeeseen, jolloin on kannattavaa pitää pisteiden määrä mahdollisimman pienenä. On kannattavaa poistaa pisteitä, jotka ovat toistoa toisiinsa nähden. Se tarkoittaa esimerkiksi sitä, että ei ole tarpeen taltioida tiheää pisteiden joukkoa tasaisella seinän pinnalla.

Yksi esikäsittelymenetelmä on alinäytteistys (engl. downsampling). Sen tavoitteena on vähentää pisteiden määrää eli pienentää resoluutiota. Tätä voidaan tehdä monella eri menetelmällä. Ensimmäinen menetelmä poistaa satunnaisesti pisteitä. Matlabissa tämä funktio on

```
alinäytteistetty_pistepilvi = pcdownsample(pistepilvi, 'random', <prosentuaalinen_osuus_pisteistä>)
```

missä <prosentuaalinen_osuus_pisteistä> tarkoittaa lukua välillä 0…1, joka kertoo sen, kuinka suuri osuus pisteistä jätetään.

Toinen menetelmä on jakaa pistepilvi kuutioihin x-y-z-avaruudessa. Näiden kuutioiden sisälle jätetään ainoastaan yksi piste, mikäli siellä on vähintään yksi piste. Tällä menetelmällä ei menetetä yksittäisiä pisteitä pistepilvestä, jonka satunnaisen poistamisen menetelmä tekisi. Laatikointimenetelmällä saadaan säilytettyä jokaiselta alueelta piste, mikäli siellä sijaitsee sellainen. Sen perusteella, että tämä algoritmi joutuu vertailemaan pisteiden sijaintia muihin pisteisiin, on ensioletus se, että tämä on suorituskyvyltään raskaampi laskettava. Tämän menetelmän komento Matlabissa on

```
alinäytteistetty_pistepilvi = pcdownsample(pistepilvi, 'gridAverage', <laatikon_koko>)
```

missä <laatikon_koko> tarkoittaa kuution sivun pituutta.

Satunnais- ja laatikkomenetelmää vertailtaessa käytettiin seuraavaa kulkua testissä. Valittiin ensin etäisyys, kuinka tarkkaan on pystyttävä erottelemaan elementit ympäristöstä.

Sen jälkeen suoritettiin alinäytteistys laatikkomenetelmällä, jossa laatikon kooksi valittiin tämä etäisyys. Tämän jälkeen tarkastettiin, kuinka monta prosenttia pistepilvi pieneni. Tämän prosenttimäärän verran alinäytteistettiin pistepilveä satunnaismenetelmää käyttäen. Kummankin menetelmän kohdalla otettiin talteen keskiarvo laskennan kestosta, ja näitä arvoja vertailtiin keskenään.

Koska testiympäristö oli aula, ajateltiin, että on pystyttävä erottelemaan elementit 10 cm tarkkuudella. Tämä valittiin sen perusteella, koska esimerkiksi pylvään leveys on joitakin kymmeniä senttimetrejä. Vastaavasti aulassa oleva käsidesiannostelija ei välttämättä tulisi taltioiduksi. Tulokset listattiin taulukossa Taulukko 1.

Taulukko 1. *Alinäytteistyksen tulokset*

Menetelmä	Prosessointi-aika (ms)	Pistepilven koko ennen	Pistepilven koko jälkeen	Prosentuaalinen muutos
Laatikointi	4,5	24942	9588	0.3844
Satunnais	2,1			

Muodostuneita pistepilviä katsoen havaitaan, että laatikointimenetelmällä saatu pistepilvi näyttää säännönmukaiselta, ja pisteitä on tasaisesti verrattuna satunnaismenetelmään. Laatikointimenetelmässä joudutaan laskemaan pisteiden koordinaatteja, eli tämä vaatii enemmän suoritustehoa.

4.6 Näytteenottotaajuus

Edellisessä kappaleessa käsiteltiin, mikä on kannattava määrä pisteitä pistepilvessä. Toinen vaikuttava tekijä näytteiden määrään on pistepilvien määrä suhteessa aikaan. Tätä kutsutaan näytteenottotaajuudeksi. Tämän arvon lisäksi vaikuttaa nopeus, jolla anturia kuljetetaan ympäristössä. Nämä kaksi tekijää vaikuttavat siihen, millä tiheydellä etäisyyden suhteen otetaan uusia näytteitä.

Näytteenottotaajuutta on hankala määrittää tarkkaan. Se johtuu siitä, että ROS syöttää dataa, ja Matlab ottaa suorituskyvyn mukaan dataa vastaan. Matlab ei ota vastaan dataa määrätyllä näytteenottotaajuudella, vaan se pyörii while-operaattorin avulla silmukassa oman suorituskykynsä mukaan.

Nopeus, jolla anturia kuljetetaan, ei ole myöskään vakio. Kuljettaminen on tavoite tehdä mahdollisimman tasaisesti lattiamerkkien välillä sillä tarkkuudella, mitä käsivoimin pystyy työntämään.

4.7 Muunnos 3D-pistepilvestä 2D-kuvaan

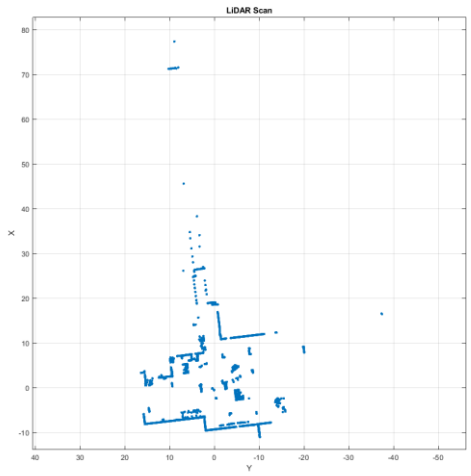
Osa SLAM-toteutuksista ottaa sisääntulona anturin datan 2-uloitteisena ja osa 3-uloitteisena pistepilvenä. Koska käytössä ollut valotutka-anturi kerää dataa 3-uloitteisena pistepilvenä, on tehtävä muunnos. Muunnos voidaan tehdä monella eri tavalla. Voidaan poistaa 3D-pistepilvestä Z-akseli, jolloin kaikki pisteet projisoituvat yhteen tasoon. Vastaavasti voidaan valita tietty korkeusjana, jonka välillä olevat pisteet projisoidaan, ja loput pisteet jätetään huomiotta.

Käytettävää muunnosmenetelmää valittaessa tulee miettiä, mikä osa pistepilvestä on olennaista. Mikäli paikannusta tehdään paikassa, jossa on suoria seiniä ympärillä, on järkevä ottaa vain kaistale korkeussuunnassa. Mikäli taas kyseessä on ympäristö, jossa on esimerkiksi kiviä maassa, on kannattavaa käyttää koko pistepilven data. On huomioitava, että 3D-pistepilven muuttaminen 2D-karttaan kadottaa informaatiota, koska erottelu korkeussuunnassa häviää. Siksi on kannattavaa hylätä mahdollisimman paljon sellaista dataa, joka ei ole informatiivista. Heijastus lattiasta on yksi esimerkki epäinformatiivisesta datasta. Tämä lattiaheijastus voitaisiin poistaa myös toisella menetelmällä, jota käsitellään myöhemmin tässä työssä.

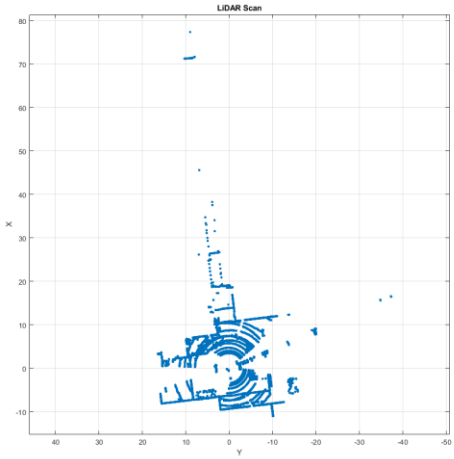
Testausta tehdessä kokeiltiin ensin menetelmää, jossa valittiin korkeusjana. Tältä väliltä litistettiin pisteet yhteen tasoon. Tähän käytettiin itse toteuttamaa funktiota, jonka tiedoston nimi on ”pc2laser.m”. Tämän jälkeen verrattiin muodostuvaa karttaa tilanteeseen, jossa käytössä oli kaikki pisteet kaikilta korkeustasoilta. Toinen vaihtoehto olisi ollut toteuttaa muunnos ROSissa [38]. Se olisi ollut suorituskyvyltään todennäköisesti parempi, koska silloin ei olisi tullut ylimääräistä muunnosta ROSin tietotyyppistä ”pointCloud2” tietotyyppiin ”pointCloud”, joka tehtiin nyt Matlabissa. Testauskoodin yhtenäistämistä varten oli kuitenkin järkevämpi toteuttaa algoritmi itse Matlabissa.

Testausympäristönä toimi edellisissä esimerkeissä käytetty Konetalon aula. Tämä ympäristö sisälsi muun muassa pylväitä, seiniä, tuoleja ja käsidesiautomaatin. Tuloksessa vertailtiin sitä, että pystyykö muunnetusta pistepilvestä tunnistamaan nämä elementit, ja tuleeko muusta pistepilven datasta vääristymiä.

Valittaessa korkeussuunnan janaa oli otettava huomioon ympäristön elementtien suhteellinen korkeus. Valotutka-anturi oli 88 cm korkeudella lattiasta, ja huoneen korkeus oli 250 cm. Valittiin korkeusjana anturin suhteen alkamaan arvosta -0,5 metriä päättyen 1,5 metriin. Tällä välillä ei tullut vastaan katto eikä lattia, mutta vastaavasti lattianrajassa olevat kivet eivät tulisi näkyviin. On huomattava, että tämä vaatii todella tasaisen alustan anturille. Tuloksena saatiin kuvat muodostuneesta kartasta sekä rajatulla korkeusjanalla (Kuva 6) että kaikki pisteet huomioon ottaen (Kuva 7).



Kuva 6. *Muunnos 2D kuvaan käyttäen rajattua korkeutta*



Kuva 7. *Muunnos 2D kuvaan käyttäen koko korkeutta*

Huomataan, että lattiaheijastukset näkyvät merkittävästi kuvassa, jossa on otettu kaikki pisteet huomioon. Tämä aiheuttaa selvää vääristymää. Rajatulla korkeusjanalla saadaan aikaan kuva, jossa on näkyvissä oleellisimmat elementit ympäristöstä SLAMin toteutuksen kannalta.

4.8 Anturia kuljettavan henkilön poistaminen eli kaistalointi

Testi tehtiin laboratoriosta löytyneellä kärryllä, joka näkyy kuvassa Kuva 5. Tämä ei ole kauko-ohjattava, joten kärryn kuljettamiseen tarvittiin ihminen. Ihmisen keho oli anturin takana, mikä vääristi tulosta. SLAM-algoritmille tämä näkyy elementtinä, joka liikkuu jatkuvasti anturin vieressä.

Tämä ongelma ratkaistiin ottamalla tietyn kulman suuruinen kaistale pistepilvestä pois. Arvoksi valittiin 60 astetta. Tähän soveltuva funktio toteutettiin itse, joka on tiedostossa

”pc_fov.m”. Anturin asetuksissa IP-sivustossa tähän olisi ollut valmis toteutus. Ongelma oli tässä, että kaistaleen pienentäminen aiheutti välkkymistä pistepilvessä.

4.9 Muita esikäsittelyprosesseja

Työssä käytettävän ”Computer Vision” -työkalukirjaston SLAM-esimerkkiskripti käyttää esikäsittelymenetelmässään funktiota ”helperProcessPointCloud”. Tämä funktio poistaa pistepilvestä ajoneuvon katon ja lattian heijastukset. Esimerkistä voidaan siis havaita, että pistepilvelle toteutettavia esikäsittelymenetelmiä on useita, ja kehittyneemmillä algoritmeilla saadaan poistettua pistepilvestä epäinformatiivista dataa.

Kyseistä funktiota ei käytetä tässä työssä, koska funktio on ollut käytössä esimerkissä, joka sisältää SLAM-toteutuksen liikenteessä. Funktiota testattaessa Konetalon käytävän pistepilveen havaittiin, että se poisti liikaa dataa. Kaistalointi suoritettiin anturia kuljettavan henkilön poistamiseksi. Lattiaheijastuksien poisto olisi kuitenkin tärkeä esikäsittelymenetelmä. On muistettava, että luvussa 4.7 toteutettiin heijastuksenpoisto siten, että rajattiin pistepilvi korkeussuunnassa. Tässä käsiteltiin muuntamista 3d-pistepilvestä 2d-kuvaan. Olisi mahdollista rajata myös 3d-pistepilvi korkeussuunnassa.

4.10 Valittava data SLAM-algoritmeille

Edeltävissä luvuissa käsiteltiin pistepilvelle toteutettavia esikäsittelymenetelmiä. Näistä tullaan valitsemaan paras esikäsittelymenetelmäkombinaatio parhaimmaksi testatuilla parametreilla. Näillä tullaan testaamaan kolmea SLAM-esimerkkiskriptiä.

Häiriösuodatus otettiin käyttöön oletusparametreilla. Alinäytteistys toteutettiin laatikointimenetelmällä kuution koolla 0,1 metriä. Kaistalointi asetettiin päälle. Muunnos 3-uloitteisesta pistepilvestä 2-uloitteiseksi karttakuvaan toteutettiin ottamalla korkeusjanaksi väli [-0.5 1.5] m anturin korkeuden suhteen. Esikäsittelyalgoritmit suoritettiin tässä järjestyksessä. Testattavien esimerkkiskriptien sisältämät esikäsittelymenetelmät poistettiin, jotta testaus olisi tasapuolinen SLAM-algoritmien suhteen.

Datanäytettä valittaessa tavoiteltiin ympäristöä, missä olisi selkeät seinät ja elementit. Tämän takia ympäristöksi valittiin Konetalon aula (Kuva 3). Alun perin Robolabraa kokeiltaessa todettiin, että tämä ympäristö sisälsi liikaa ikkunoita ja muita epämääräisen muotoisia elementtejä, jotka tekivät algoritmien testauksesta liian monimutkaisen. Tarkempi kuvaus fyysisestä testauksesta on aliluvussa 3.4.

Näytteenottotaajuus pyrittiin valitsemaan sellaiseksi, että etäisyys eri pistepilvien välillä on joitakin kymmeniä senttimetrejä. Tätä tavoitetta vastasi se, että poistettiin joka toinen pistepilvi.

4.11 Testattavat SLAM-toteutukset

Työhön valittiin kolme Mathworksin toteuttamaa SLAM-esimerkkiskriptiä. Nämä skriptit löytyvät Matlabin dokumentaatiosivuilta. Kunkin esimerkkiskriptin on tarkoitus edustaa kunkin Matlabin työkalukirjaston algoritmeja. Nämä kirjastot ovat Navigation Toolbox, Computer Vision Toolbox ja Lidar Toolbox. Työn tarkoitus on vertailla Matlabin SLAM-toteutuksia, ja tasapuolisin tulos saadaan sillä, että käytetään valmiita Mathworksin toteuttamia esimerkkejä.

Kullakin työkalukirjastolla on oma kohdentamisalgoritmin toteutus perustuen yleisiin menetelmiin perustuviin kohdentamisalgoritmeihin. Näitä menetelmiä on lueteltu aliluvussa 2.3.1. Esimerkkiskripteistä havaitaan, että niiden käyttämät funktiot ja tietorakenteet ovat peräisin useasta työkalukirjastosta. Esimerkiksi Lidar toolboxin esimerkkiskripti käyttää funktiota ”optimizePoseGraph”, joka kuuluu Navigation toolboxin funktiokirjastoon. Tämä tieto siitä, mitä työkalukirjastoja mikäkin esimerkkiskripti käyttää, löytyy esimerkin nettisivun dokumentaatiossa kohdassa ”This example uses:”. Tämän vuoksi vertailu tässä työssä perustuukin ennemmin esimerkkien testaamiseen ja siihen, mikä esimerkkien tietorakenteiden ja algoritmien kombinaatio suoriutuu parhaiten.

SLAM perustuu siihen, että selvitetään eri anturin datan välinen etäisyyden- ja suunnanmuutos. Tämä perättäisten pistepilvien selvitys on ensimmäinen vaihe SLAMissa. Toinen vaihe on silmukkatarkastelu, josta on puhuttu yleisellä tasolla luvussa 2.3.2. Tämä ei ole riippuvainen ensimmäisestä vaiheesta siksi, koska tämän tehtävä on korjata ensimmäisessä vaiheessa tehtyjä arvauksia etäisyyden- ja suunnanmuutoksista. Nämä kaksi vaihetta ovat siten erillään toisistaan, ja testausvaiheessa tulee tutustua näihin kumpaankin vaiheeseen erikseen.

Matlabin algoritmien taustalla oleva teoria havaitaan olevan useassa tapauksessa samasta kirjallisesta lähteestä. Lähde sisältää matemaattisen kaavan, jonka perusteella algoritmi on tehty. Lähteitä tutkiessa havaitaan, että esimerkiksi Computer Visionin algoritmin ”pcregisterndt” ja Lidar Toolboxin algoritmin ”MatchScans” lähteet ovat samat [39] [26]. Tästä voidaan päätellä, että algoritmien tulisi olla matemaattiselta NDT-malliltaan samat, ja siten eroavaisuus tulee jostain muusta tekijästä.

4.11.1 Navigation toolbox

Navigation-työkalukirjasto sisältää SLAM-algoritmeja, joita voi käyttää joko ohjelmoimalla skriptin tai käyttämällä valmista graafista työkalua nimeltä ”SLAM Map Builder” [40]. Dokumentaatiosta ei löydy mainintaa siitä, mitä algoritmeja tämä graafinen työkalu käyttää. Sen perusteella, että työkalu on osa Navigation Toolboxia, voidaan olettaa, että

algoritmit ovat samoja, ja käyttäjä voi tehdä SLAM-toteutuksen joko toteuttaen oman skriptin saatavilla olevilla algoritmeilla [41] tai käyttäen valmista graafista työkalua.

Algoritmia testattaessa on muunnettava anturidata 3D-pistepilvikuvasta 2D-karttakuvaan. Tavoite tällä testillä on vertailla erityisesti sitä, miten tämä muutos tulee vaikuttamaan lopputulokseen.

Testauksessa kokeiltiin algoritmia toteuttaen oma skripti graafisen ”SLAM Map Builder” -työkalun sijaan. Tällä tavoin saatiin yhtenäistettyä testikoodi ja päästään käsiksi lähdekoodiin. Käytössä ollut esimerkkiskripti [42] käyttää ainoastaan funktiota ”addScan” selvittääkseen suunnan- ja etäisyydenmuutoksen. Tämä funktio toteuttaa sekä perättäisien pistepilvien tarkastelun että silmukkatarkastelun.

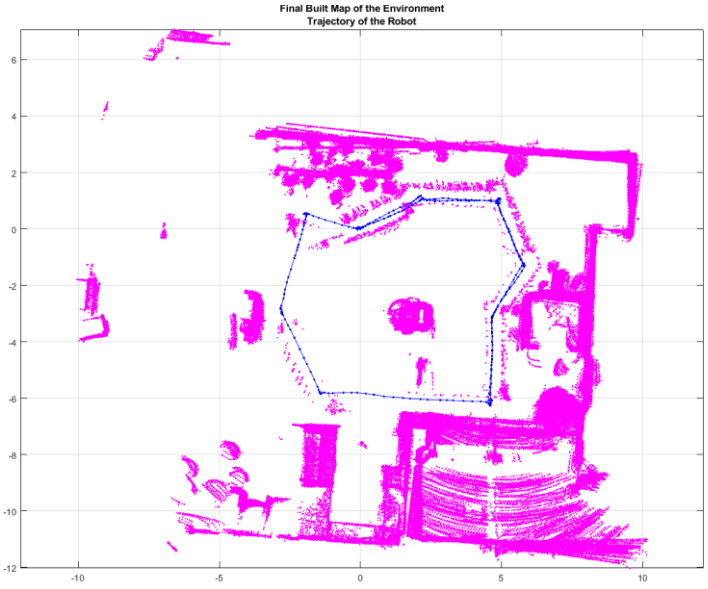
Funktion ”addScan” lähdekoodia tarkastellessa havaitaan, että tämä käyttää paikannukseen funktioita ”matchScans” ja ”matchScansGrid”. Funktio ”matchScans” käyttää NDT-menetelmää lähdekoodin perusteella. Tämän matemaattinen malli perustuu lähteeseen [39]. Lisäksi funktio käyttää lähdettä [26], joka käsittelee kolmannen ulottuvuuden paikannusta. Funktio ”matchScansGrid” perustuu menetelmään, missä kartta jaetaan osiin, jonka perusteella selvitetään etäisyyden- ja suunnanmuutos [43]. Tästä menetelmästä oli vaikea löytää tarkempaa tietoa lähdekoodista. Tutkielmassa [44, s. 14] on puhuttu yleisellä tasolla laatikoinnista (Grid). Kun kartta on jaettu osiin, merkataan kukin osa todennäköisyysarvolla, mikäli osan sisällä esiintyy yksikin piste. Tämän funktion ”matchScansGrid” epäillään perustuvan siksi tähän menetelmään, jotta tällä saadaan ensiarvaus. Funktiossa ”addScan” käytetään näitä kahta funktiota perättäin siten, että funktiolla ”matchScansGrid” haetaan ensimmäinen arvio, ja tämä arvio annetaan parametrina funktiolle ”matchScans”. Tämä funktio taas NDT-menetelmällä laskee tarkemman arvion.

Esimerkkiskriptissä on käytetty tietorakenteena ainoastaan ”lidarSLAM”-objektia. Tähän objektiin tallennetaan suunnanmuutokset, jotka voidaan lukea komennolla

```
lidarSLAM_objekti.PoseGraph.nodeEstimates
```

joka antaa matriisina etäisyydet ja suunnanmuutokset 2D-tasossa.

Alla olevassa kuvassa Kuva 8 on kartta esimerkkiskriptin laskeneesta reitistä. Silmin havaitaan, että reitti on realistisen muotoinen. Se sisältää hyvin teräväreunaisesti lattiaan merkityt kääntöpisteet. Myöhemmässä luvussa on laskentatulokset ja erot muihin esimerkkiskripteihin.



Kuva 8. *”Navigation”-toolbox esimerkkiskriptin kartan piirto*

Silmukkatarkastelua on haastavaa analysoida tässä esimerkkiskriptissä, koska sen toteutus sisältyy saman funktion ”addScan” sisälle. Siten ei päästy vertaamaan ilman lähdekoodin muokkausta sitä, miten silmukkatarkastelu vaikuttaa. Lähdekoodia tutkimalla havaitaan, että toteutuksessa funktio ”acceptRelPoseAndSearchForLoopClosures” tiedostossa ”lidarSLAM.m” etsii silmukoita tietyn säteen sisältä. Säde on oletusarvoisesti kahdeksan (8) metriä, ja tämä muutettiin arvoon yksi metri. Koska testi tehtiin Konetalon aulassa tolppaa ympäri pyörien, olisi kahdeksan metriä ollut suhteellisen ylimitoitettu arvo. Mikäli kyseessä olisi liikenne, olisi risteysalue sellaista suuruusluokkaa, että uudelleen vierailtu paikka olisi kahdeksan metrin sisällä. Konetalon testitapauksessa melkein kaikki pisteet olisivat olleet kahdeksan metrin etäisyydellä toisistaan.

4.11.2 Computer vision toolbox

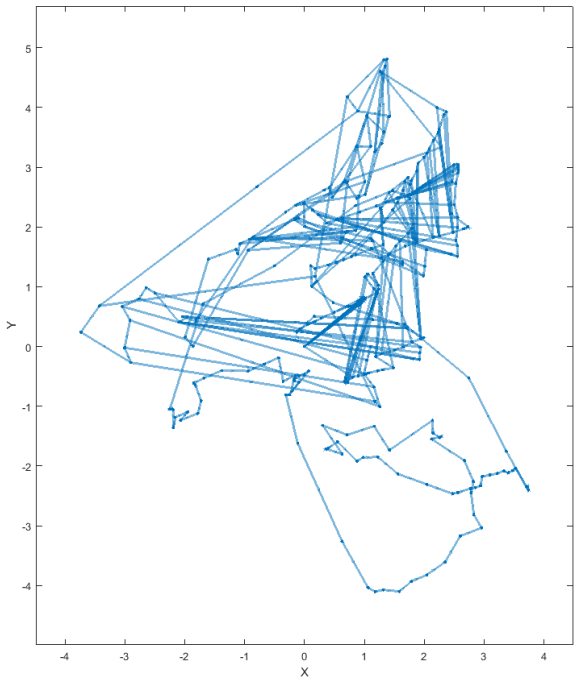
Edellisessä luvussa havaittiin, että ”Navigation”-työkalukirjastolla toteutettu skripti oli suhteellisen lyhyt, ja funktioita oli suhteellisen pieni määrä. Tähän työkalukirjastoon verrattuna ”Computer vision” -työkalukirjaston funktiot ovat monimutkaisempia, joka näkyy esimerkiksi siinä, että välimuuttujia ja ehtolauseita tarvitaan suhteessa enemmän. Testauksessa käytetään esimerkkiskriptiä [45].

Skriptissä käytetään ”Computer Vision” -työkalukirjaston kohdistamisalgoritmeja. Algoritmi antaa ulostuloksi tietorakenteen ”rigid3d”, joka sisältää erikseen sekä etäisyyden- että suunnanmuutoksen 3D-avaruudessa. Algoritmin syntaksi on

```
rigid3d_type = pcregister<x>(pc, previous_pc)
```

missä X kuvaa algoritmityyppiä, joka on ”ndt”, ”icp”, ”corr” tai ”cpd”. Esimerkkiskriptissä käytetään NDT-menetelmää, eli tähän perustuva funktio on ”pcregisterndt”. Tämä funktio perustuu samoihin kirjallisiin lähteisiin kuin funktio ”matchScans” [39] [26].

Suorittamalla esimerkkiskripti saadaan kartta, joka on esitetty kuvassa Kuva 9. Silmin havaitaan, että kartta vastaa hyvin heikosti todellista muotoa. Tämä algoritmi kykenee kuitenkin 3D-pistepilven käsittelyyn, johon muut esimerkkiskriptien algoritmit eivät kykene. Voidaan kuitenkin havaita 3D-tarkastelusta, että ajelehtimista korkeussuunnassa on useita metrejä, vaikka todellisuudessa siirtymä on nolla.



Kuva 9. *”Computer Vision” -toolbox esimerkkiskriptin 1 kartan piirto*

Syy näin heikolle tulokselle tällä esimerkkiskriptillä voi olla algoritmin ”pcregisterndt” suoriutuminen tilanteesta, jossa ei ole olemassa esiarviodataa. Tässä esimerkin skriptissä käytettiin alun perin arviota suunnan- ja etäisyydenmuutoksesta, joka oli peräisin IMU-anturista. IMU (engl. inertial measurement unit) anturi on inertiamittausyksikkö, joka mittaa kiihtyvyyksiä kolmessa ulottuvuudessa sekä pyörimisessä että suoraviivaisessa liikkeessä. Yhteensä siis mitataan kuutta vapausastetta. Kiihtyvyyttä integroimalla saadaan selville paikkakoordinaatti. [46] [47] Voidaan havaita, että tämä SLAM-toteutus tarvitsee luotettavaan suunnan- ja etäisyydenmuutokseen arviodataa. Tämä on myös hyvä asia siksi, että tämän avulla voidaan saada aikaan kahden anturin fuusion [48]. Alkuperäinen esimerkkiskriptin data oli kerätty liikenneympäristössä. Tässä havaittiin realistinen tulos, koska mukana oli IMU-anturi. Havaittiin myös tässä alkuperäisessä esimerkissä, että kun skriptistä poisti IMU-anturin datan käytöstä, vääristyi tulos merkittävästi. Kartasta hävisi selkeä muoto.

Toinen tekijä tälle heikommalle tulokselle saattaa olla algoritmin parametrit. Funktio ”pcregisterndt” käyttää parametriä ”gridStep”. Algoritmi kuutioi pistepilven osiin, ja tämä parametri määrittää tämän kuution reunan pituuden. Algoritmin osoittamasta kirjallisesta lähteestä [39] päätellään, että tämä parametri määrittää sen, kuinka suuriksi osiksi NDT-laskentamenetelmässä pistepilvi kuutioidaan. Kullekin osille lasketaan erillinen todennäköisyysjakauma pistetiheydestä, joka selitettiin aliluvussa 2.3.1. Funktion lähdekoodissa on seuraava teksti:

> Prior to registration, PCREGISTERNDT voxelizes the fixed point cloud using the specified gridStep. A smaller grid step will capture distinct structure within the point cloud at the cost of being dominated by local noise produced by the sensor. Use pcdownsample to determine the effect of voxelization:
> pcdownsample(fixed, 'gridAverage', gridStep).

Kolmannen osapuolen SLAM-toteutuksessa [49] tätä parametrin arvoa kutsutaan resoluutioksi. Lähdekoodin tekstin perusteella kuitenkin tulee välttää liian pienen ”gridStep”-arvon valintaa, koska muuten anturin kohina vaikuttaa. Vastaavasti liian pieni resoluutio ei anna riittävästi informaatiota. Parametrin arvo asetettiin arvoon 0,3, mutta jälkitutkittavaksi jää eri parametrien valinta. Pienempi arvo tuotti virheen skriptin suorituksessa. Arvon yksiköstä löytyy vähän dokumentaatiota Matlabin lähdekoodista. Oletetaan, että arvo on metreinä. Silloin 30 cm elementit ovat aulaympäristössä suhteessa järkevän kokoisia. Vastaavasti liikenteessä etäisyydet olisivat suuremmat. Lisätietona on se, että kuutio otetaan huomioon vain, mikäli siellä on minimimäärä pisteitä. Matlabin lähdekoodissa tämä arvo ”minVoxelPoints” on 6 pistettä.

Silmukkatarkastelun tulosta on vaikea arvioida, koska testitulos oli niin epärealistisen muotoinen. Esimerkkiskriptissä on toteutettu silmukkatarkastelu erikseen käyttäen hyödyksi ”Computer Vision” -työkalukirjaston silmukkatarkastelufunktioita, jotka sijaitsevat tiedostossa ”C:\Program Files\MATLAB\R2021b\toolbox\vision\vision\scanContextLoopDetector.m”. Skriptissä käytetään funktiota ”detectLoop”, jolla etsitään tarkastettavat silmukat. Kriteerinä on aiemmin tässä työssä tarkasteltujen silmukkatarkastelujen tapaan säde, jonka sisältä pisteitä haetaan. Tässä arvo on 0,3 metriä. Etsimisen jälkeen kohdistetaan uudet silmukat, ja lisätään ne tietotyyppiin ”pcviewSet” komennolla ”addConnection”.

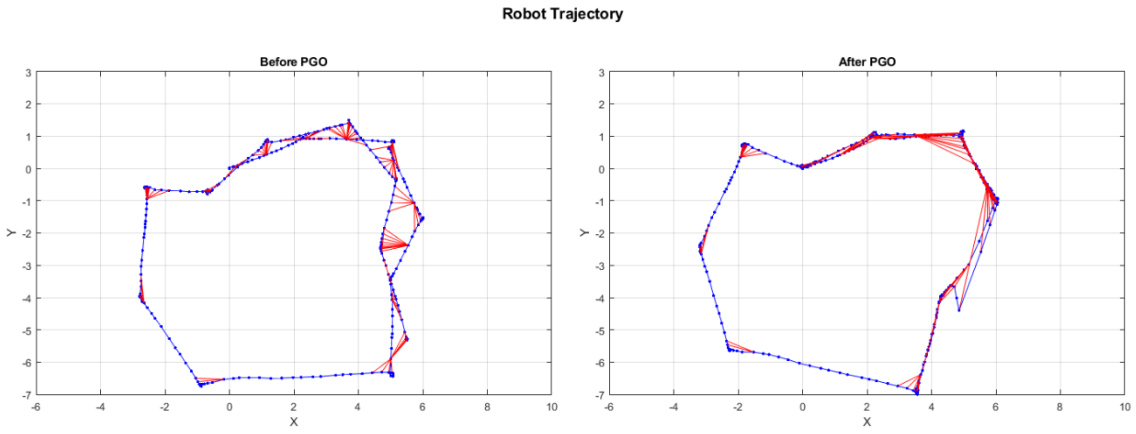
4.11.3 Lidar toolbox

Valotutka-anturiin suunniteltu työkalukirjasto tarjoaa algoritmeja suunnattuna juuri Velodynen anturille [50]. Algoritmi ottaa vastaan 2-ulotteista ”lidarScan”-tietotyyppiä, joten tähän tarvitaan muunnos 3-ulotteisesta pistepilvityypistä. ”Lidar”-työkalukirjaston tietorakenteiden periaate jäljittelee muita esimerkkejä. Ensin luodaan ”poseGraph”-tietotyyppi, johon tallennetaan etäisyyden- ja suunnanmuutoksia.

Testauksessa käytetään esimerkkiskriptiä [51]. Aiemmin tässä työssä kokeiltiin esimerkkiskriptiä, joka painottui ”Navigation”-työkalukirjaston funktioihin. Funktio ”addScan” selvitti sekä suunnan- ja etäisyydenmuutoksen edelliseen pistepilveen nähden. Lisäksi se toteutti silmukkatarkastelun. Tässä luvussa käytettävä esimerkkiskripti käyttää samoja suunnan- ja etäisyydenmuutosalgoritmeja ”matchScans” ja ”matchScansGrid”, mutta niitä on käytetty tässä esimerkissä suoraan. Vastaavasti ”Navigation”-työkalukirjaston esimerkissä funktioita käytettiin erillisen funktion ”addScan” kautta. Tietorakenteeseen ”poseGraph” tallennus ja silmukkatarkastelu on toteutettu itse. Tämä antaa paremman mahdollisuuden tutkia silmukkatarkastelun tuloksia, kun eri muuttujien välivaiheita voidaan tarkastella.

Silmukkatarkastelu on toteutettu erillään tässä skriptissä pistepilvien perättäisestä kohdistamisesta. Skriptissä on ensin käyty perättäin kaikki pistepilvet, ja tästä on muodostettu kartta. Sen jälkeen on toteutettu silmukkatarkastelu erikseen kaikille pistepilvelle. Tietyn välein pistepilvien joukkoa on haettu ympärillä olevat pisteet, jotka ovat lähempänä kuin metri. Tämä säteen arvo vaihdettiin tähän työhön testiympäristön mittakaavaan sopivaksi. Haetut pisteet kohdistetaan kyseiselle pisteelle.

Testauksessa saadaan kuvassa Kuva 10 esitetty kartta, jossa ensimmäisessä on kartan muodostus ennen silmukkatarkastelua (PGO = Pose Graph Optimize) ja toisessa silmukkatarkastelun jälkeen. Kartoissa näkyvät punaiset viivat ovat silmukkatarkastelussa tehty kohdistus kullekin rajatun säteen sisällä oleville pistepilvelle.



Kuva 10. *”Lidar Toolbox” -esimerkkiskriptin kartan piirto*

Tässä esimerkkiskriptissä oli toteutettu silmukkatarkastelu erikseen pistepilvien kohdistamisen jälkeen, jonka ansiosta tämä antoi hyvän todisteen siitä, miten tärkeä silmukkatarkastelu on SLAMissa. Kuvasta voidaan myös havaita, että silmukkatarkastelu aiheuttaa itseasiassa vääristymää suunnanmuutoksessa. Kun tarkastellaan oikeita lattiamerkintöjen sijaintia kohdassa Kuva 4. Lattiamerkintöjen sijainti (m**)**, havaitaan, että muoto

on oikea, mutta on syntynyt kiertymää. Tämä on yksi silmukkatarkastelun haaste, koska algoritmilla ei ole referenssitietoa silmukan toisen puolen pisteistä.

4.11.4 Keskinäinen vertailu

Aiemmissa luvuissa oli sanallisia silmin nähtäviä havaintoja SLAM-esimerkkiskriptien suorituksesta. Tämä luku sisältää vertailua laskennallisista eroista, joiden tavoite on selvittää algoritmien keskinäinen ero. Laskennalliset menetelmät selitettiin luvussa 4, ja alla on esitetty tulokset. Taulukossa Taulukko 2 on esitetty etäisyyserot todellisen lattiaan merkityn koordinaatin ja SLAM-algoritmin laskeman absoluuttisen koordinaatin välillä. Erot ovat laskettu Pythagoraan lauseella x- ja y-pisteiden perusteella. ”Computer Vision” -työkalukirjaston esimerkkiskriptin kohdalla on olemassa z-koordinaatti, mutta sitä ei otettu huomioon tässä tarkastelussa.

Taulukko 2. *Etäisyysero todellisen lattiapisteen ja SLAMin arvion välillä*

Lattiapiste	**Etäisyysero (m)**		
	Navigation Toolbox	**Computer Vision Toolbox**	**Lidar Toolbox**
2	0,15	1,01	0,21
3	0,14	3,33	0,11
4	0,36	9,33	0,26
5	0,43	7,32	1,36
6	0,52	8,86	1,78
7	0,50	4,23	1,36
8	0,36	4,92	0,88
9	0,29	4,64	0,42
1	0,10	1,20	0,09
2	0,04	1,53	0,17
3	0,16	3,84	0,14
4	0,20	7,14	0,14
5	0,34	4,15	1,22
6	0,36	8,90	1,68
Keskiarvo	**0,28**	**5,03**	**0,70**

Alla olevassa taulukossa Taulukko 3 on esitetty ajelehtiminen etäisyytenä kunkin lattiapisteen välillä. SLAM-algoritmi on laskenut suhteellisen etäisyyseron kunkin eri pistepilven välille. Tässä on laskettu kahden lattiamerkin välillä olevien pistepilvien etäisyysero yhteen. Tästä summasta on vähennetty lattiapisteiden todellinen etäisyys.

Taulukko 3. *Ajelehtima arvioitujen paikkakoordinaattien välillä*

Lattiamerkkien väli	Etäisyys (m)		
	Navigation Toolbox	Computer Vision Toolbox	Lidar Toolbox
1-2	0,00	1,35	-0,01
2-3	0,12	3,73	0,19
3-4	0,54	15,70	0,38
4-5	0,15	3,96	1,52
5-6	0,06	0,40	0,15
6-7	0,54	6,25	0,47
7-8	0,08	3,95	0,11
8-9	0,36	6,35	0,34
9-1	0,25	2,99	0,29
1-2	0,24	1,53	0,22
2-3	0,10	0,24	0,06
3-4	-0,01	5,84	0,10
4-5	0,29	6,79	2,92
5-6	0,13	3,00	0,24
Keskiarvo	**0,21**	**4,43**	**0,50**

Suoritusajat ovat listattuna taulukossa Taulukko 4. Suoritusaika on mitattu skriptin alusta loppuun. Esikäsittelyaikaa ei otettu huomioon, koska se oli kaikille sama.

Taulukko 4. *Suoritusajat*

	Navigation Toolbox	Computer Vision Toolbox	Lidar Toolbox
Aika (s)	644,4	43,6	376,2

Alun perin oli tavoite vertailla suorituskykyä ja paikannuksen tarkkuutta. Tuloksia tarkastellessa havaitaan, että nämä tulokset korreloivat keskenään nimenomaan suoritusajan ja paikannuksen tarkkuuden suhteen. Sekä ”Navigation”- että ”Lidar” -työkalukirjastojen esimerkkien tulokset ovat hyvin samankaltaisia. Ne ovatkin toteutettu kummatkin ”matchScans” ja ”matchScansGrid” funktioilla. Vastaavasti ”Computer Vision” -työkalukirjaston esimerkki, joka perustuu ”pcregisterndt” funktioon, suoriutuu kartan paikannuksen tarkkuuden suhteen heikommin, mutta vastaavasti sn suoritusaika on paljon lyhyempi.

Kun tarkastellaan lattiamerkkauspisteiden paikannuksen tarkkuutta, voidaan havaita, että näillä on kymmenkertainen ero. Vastaavasti suoritusajassa on myös lähes kymmenkertainen ero. Algoritmit perustuvat samaan NDT-paikannusmenetelmään. Tämän testin perusteella voidaan todeta, että SLAM-algoritmi tarvitsee laskentatehoa, ja se on osaltaan riippuvainen algoritmin paikannuksen tarkkuuteen.

Näiden tuloksien perusteella valittiin paras algoritmi, jolla jatkettiin testituloksia pistepilvien esikäsittelytestaukseen. Parhaan algoritmin valintakriteereihin oli monia vaihtoehtoja. Tässä valinta tehtiin painottaen kaikkia mitattuja tuloksia tasaisesti. Paras algoritmi valittiin sen perusteella, minkä ajelehtimisen keskiarvon, etäisyyserojen keskiarvon ja suoritusajan tulo oli pienin. Nämä taulukoidaan taulukossa Taulukko 5.

Taulukko 5. *Lopputulos esimerkkiskripteille*

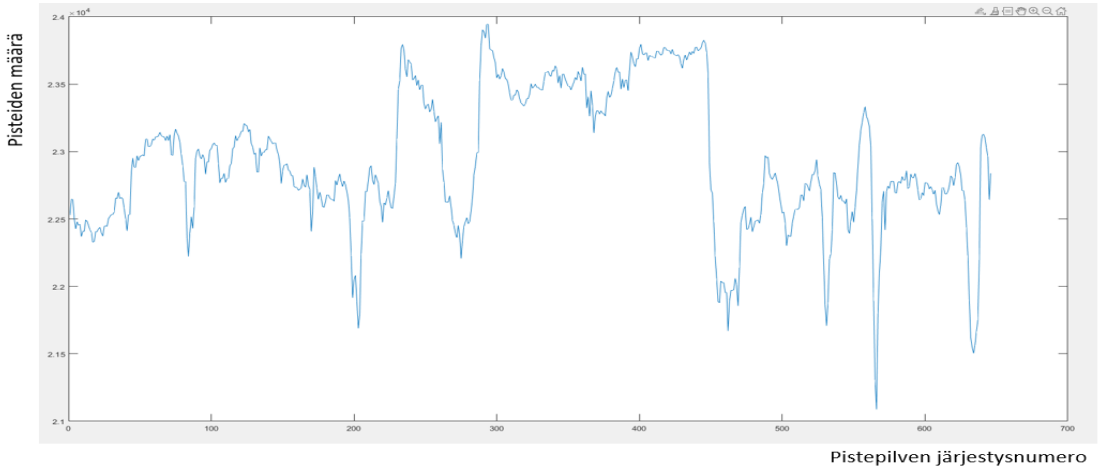
	Navigation Toolbox	**Computer Vision Toolbox**	**Lidar Toolbox**
Tulos	37,5	969,0	131,6

Tämän perusteella valitaan jatkoon esimerkkiskripti, joka käyttää työkalukirjastoja ”Navigation Toolbox”. Sen tulos oli merkittävästi pienin.

4.12 Pistepilvien jatkokäsittely

Aiemmin tässä työssä aliluvusta 4.4 lähtien vertailtiin esikäsittelymenetelmiä silmämääräisesti. Parhaaksi koetulla esikäsittelymenetelmien kombinaatiolla vertailtiin SLAM-esimerkkiskriptejä. Nyt käytetään edellisessä vaiheessa parhaaksi testattua SLAM-esimerkkiskriptiä, jolla testataan erilaisia esikäsittelymenetelmien kombinaatioita. Testeissä tullaan vertaamaan erilaisia esikäsittelymenetelmien pareja. Pareja muodostavat esimerkiksi esikäsittelymenetelmien funktioiden parametreista muodostuvat erot. Arviointikriteereinä käytetään samoja menetelmiä kuin luvussa 4.11.4.

Alinäytteistäessä satunnaismenetelmällä tulee huomioida, minkä perusteella pisteitä poistetaan. Parametrina annettava luku välillä 0…1 määrittää sen, kuinka suuri osa pisteitä jätetään pistepilveen. Se ei siten huomioi pistepilven kokoa. Mikäli tavoitteena on tietty pistepilven pisteiden määrä, tulee suhteuttaa tämä prosentti pistepilven kokoon. Suorituskyvyn kannalta tämän ei pitäisi aiheuttaa asymptoottisen tehokkuuden näkökulmasta ongelmaa, koska pistepilven pisteiden määrä on tallennettu muuttujaan ”Count” objektissa ”pointCloud”. Tässä ei jouduta esimerkiksi mittaamaan vektorin kokoa, jossa olisi riski, että asymptoottinen tehokkuus kasvaisi lineaariseksi. Testitapauksissa toteutetaan muunnos siten, että käytetään tavoitepistemäärää. Siten saadaan yhtenäistettyä pistepilvien koko, jolloin testi olisi mahdollisimman tasapuolinen silloinkin, kun on suoritettu muita esikäsittelymenetelmiä. Alkuperäisien pistepilvien koko käytössä olleessa esimerkkidatassa on esitetty alla olevassa kuvassa. Koko on mitattu Matlabissa ROSista latauksen jälkeen ennen esikäsittelyä.



Kuva 11. *Pistepilvien pisteiden määrä*

Havaitaan, että pistepilven pisteiden määrä vaihtelee suhteessa sijaintiin. Sen takia on merkitystä, että vähennetäänkö pisteiden määrä tiettyyn pisteiden määrään vai käytetäänkö prosentuaalista vähennystä.

4.12.1 Alinäytteistyksen ja näytteenottotaajuuden suhde

Ensimmäisessä esikäsittelymenetelmässä vertaillaan sitä, onko alinäytteistyksellä vai näytteenottotaajuudella merkittävämpi rooli lopputuloksen kannalta. Tällä testataan sitä, että kannattaako poistaa enemmän pisteitä pistepilvestä, jolloin pistepilviä on määrällisesti enemmän eli näytteenottotaajuus on suurempi. Vastaavasti pistepilvessä voi olla enemmän pisteitä, mutta näytteenottotaajuus on pienempi eli otetaan esimerkiksi vain joka neljäs pistepilvi huomioon. Testaukseen otetaan seuraavat arvot: valitaan alkuperäisestä pistepilvestä jokainen pistepilvi ja alinäytteistetään tätä siten, että pisteitä on 2000. Vastaavasti valitaan joka neljäs pistepilvi, jolloin pistepilvessä on pisteitä 8000. Pisteiden karsiminen tehdään satunnaispoistomenetelmällä, ja pistepilvien karsiminen toteutetaan ottamalla joka neljäs pistepilvi järjestyksessä. Testissä käytetään kaistaleen poistoa mutta ei häiriösuodatusta. Alla olevassa taulukossa Taulukko 6 on taulukoituna tulokset kartan tarkkuudesta.

Taulukko 6. *Alinäytteistyksen ja näytteenottotaajuuden suhdevertailun kartanpiirtotulokset*

Testitapaus →	**pisteitä pistepilvessä = 2000 näytteenottotaajuus=1/1**		**pisteitä pistepilvessä = 8000 näytteenottotaajuus=1/4**	
Lattiamerkkien väli / lattiamerkki ↓	**Lattia-merkkien etäisyys (m)**	**Ajelehtima (m)**	**Lattiamerk-kien etäisyys (m)**	**Ajelehtima (m)**
(1)-2	0,21	-0,08	0,19	0,03
(2)-3	0,30	0,06	0,24	0,23
(3)-4	0,44	0,23	0,45	0,45
(4)-5	0,43	0,12	0,44	0,14
(5)-6	0,37	-0,10	0,53	0,08
(6)-7	0,41	0,46	0,50	0,47
(7)-8	0,25	0,10	0,40	0,16
(8)-9	0,15	0,18	0,29	0,32
(9)-1	0,05	0,22	0,10	0,24
(1)-2	0,16	0,00	0,14	0,31
(2)-3	0,27	0,21	0,24	0,06
(3)-4	0,15	-0,12	0,38	0,14
(4)-5	0,13	0,15	0,44	0,18
(5)-6	0,40	0,34	0,44	0,17
Keskiarvo	**0,27**	**0,13**	**0,34**	**0,21**

Taulukko 7. *Alinäytteistyksen ja näytteenottotaajuuksien suhdevertailun suoritusajat*

Testitapaus →	**pisteitä pistepilvessä = 8000 näytteenottotaajuus=1/4**	**pisteitä pistepilvessä = 2000 näytteenottotaajuus=1/1**
Vaihe ↓	**Ajan kuluminen (s)**	
Esikäsittely	37,7	142,2
SLAM	810,5	984,5

Suoritusajoista huomataan, että SLAMia toteuttaessa merkittävä suorituskykyyn vaikuttava tekijä on pisteiden suhteellinen määrä alinäytteistyksen ja näytteenottotaajuuden välillä. Suoritusaikaan vaikuttaa pääasiassa pistepisteiden yhteinen määrä. Havaitaan kuitenkin, että suurempi näytteenottotaajuus lisää hieman suoritusaikaa SLAMissa. Tässä epäillään, että useamman pistepilven käsittely vie enemmän prosessointia, koska kaikki laskennan välivaiheet on jouduttava käymään neljä kertaa useammin.

Esikäsittelyssä havaitaan, että suuremman näytteenottotaajuuden tapaus vie moninkertaisesti enemmän aikaa. Tässä epäiltiin ensin asymptoottisen tehokkuuden vaikutusta. Funktiossa, joka esikäsittelee jokaisen datajoukon pistepilven, kasvatettiin käsiteltyjen pistepilvien tietorakenteen säiliön kokoa aina yhdellä lisää. Matlabin editori-ikkuna antoi tässä varoitusta suorituskyvyn kasvusta. Tätä tietorakennetta muokattiin siten, että tietorakenteen säiliötä kasvatettiin alussa tavoitekokoon pistepilvijoukon ja näytteenottotaajuuden perusteella.

Tämän muutoksen tekeminen ei tehnyt vaikutusta suoritusaikaan. Koska Matlabin suoritusaikamittaustyökalut ”tic” ja ”toc” ovat helposti asetettavissa for-silmukan sisään, tutkittiin jokaisen pistepilven esikäsittelyn aikaa. Näitä tuloksia vertailtiin silmämääräisesti

taulukossa. Havaittiin, että suoritusajat ovat tasaisia. Tämä selitti syyn selvälle erolle pistepilvien käsittelyajoissa. Yksittäisen pistepilven käsittelyyn menee yhtä paljon aikaa riippumatta siitä, kuinka suuri määrä pisteitä pistepilveen jätetään. Olisi myös järkevää suorittaa kaistalointi vasta alinäytteistyksen jälkeen, koska silloin olisi vähemmän pisteitä laskettavana kaistaloinnissa.

4.12.2 Häiriösuodatus

Tässä testissä tarkoitus oli tutkia häiriösuodatuksen vaikutusta käytettävälle anturille vallitsevassa testausympäristössä. Ensimmäisessä testissä poistettiin häiriösuodattamalla pistepilvet ja alinäytteistettiin pistepilven koko arvoon 10000. Toisessa menetelmässä ei käytetty häiriösuodatusta, mutta alinäytteistettiin samaan kokoon 10000. Testiin otettiin joka toinen pistepilvi, ja kaistaleenpoistoa käytettiin.

Taulukko 8. *Häiriösuodatuksen vertailun kartanpiirtotulokset*

Testitapaus →	**Häiriösuodatus käytössä**		**Häiriösuodatus pois käytöstä**	
Lattiamerkkien väli / lattiamerkki ↓	**Lattiamerkkien etäisyys (m)**	**Ajelehtima (m)**	**Lattiamerkkien etäisyys (m)**	**Ajelehtima (m)**
(1)-2	0,18	-0,10	0,19	-0,01
(2)-3	0,21	0,14	0,20	0,13
(3)-4	0,47	0,60	0,48	0,52
(4)-5	0,54	0,15	0,51	0,15
(5)-6	0,58	0,04	0,56	0,04
(6)-7	0,58	0,42	0,57	0,53
(7)-8	0,41	0,12	0,42	0,14
(8)-9	0,26	0,23	0,26	0,23
(9)-1	0,09	0,21	0,09	0,22
(1)-2	0,10	0,18	0,07	0,24
(2)-3	0,27	0,06	0,26	0,05
(3)-4	0,34	0,04	0,31	0,04
(4)-5	0,45	0,26	0,43	0,26
(5)-6	0,51	0,14	0,48	0,18
Keskiarvo	**0,36**	**0,18**	**0,34**	**0,19**

Taulukko 9. *Häiriösuodatuksen vertailun suoritusajat*

Testitapaus →	**Häiriösuodatus käytössä**	**Häiriösuodatus pois käytöstä**
Vaihe ↓	**Ajan kuluminen (s)**	
Esikäsittely	114,1	84,0
SLAM	1926,8	1909,0

Tuloksista huomataan, että häiriösuodatus vie paljon suoritusaikaa, mutta SLAMin tulokset eroavat yksittäisien senttimetrien tasolla pisteiden etäisyyserossa ja ajelehtimisessa. Tässä huomataan, että häiriösuodatuksen käyttöä kannattaa harkita riippuen ympäristöstä ja anturista. Tässä tilanteessa se oli tuloksien tarkkuuden ja suoritusajan nähden hyödytön.

4.12.3 Satunnaus- ja laatikointimenetelmän ero

Kolmas testi vertailee alinäytteistystä satunnaismenetelmän ja laatikointimenetelmän välillä. Tämä on merkittävä tutkittava sen takia, koska pisteiden määrä on oleellinen osa SLAMin suoritusajassa. Testin kulku oli sama, jota käytettiin luvussa 4.5. Ensin valittiin laatikon kooksi 10 cm. Tätä menetelmää verrataan siihen, että alinäytteistetään satunnaismenetelmällä siten, että pisteitä on sama määrä. Testissä valittiin näytteenottotaajuudeksi joka toinen pistepilvi. Häiriösuodatus ja kaistaleenpoisto olivat käytössä.

Taulukko 10. *Alinäytteistysmenetelmien vertailun kartanpiirtotulokset*

Testitapaus →	**Alinäytteistys laatikointimenetelmällä**		**Alinäytteistys satunnaismenetelmällä**	
Lattiamerkkien väli / lattiamerkki ↓	**Lattiamerkkien etäisyys (m)**	**Ajelehtima (m)**	**Lattiamerkkien etäisyys (m)**	**Ajelehtima (m)**
(1)-2	0,26	0,03	0,19	-0,11
(2)-3	0,28	0,16	0,23	0,11
(3)-4	0,50	0,59	0,45	0,55
(4)-5	0,51	0,12	0,46	0,08
(5)-6	0,62	0,04	0,50	0,04
(6)-7	0,55	0,54	0,54	0,48
(7)-8	0,39	0,10	0,41	0,13
(8)-9	0,30	0,38	0,26	0,24
(9)-1	0,08	0,25	0,09	0,22
(1)-2	0,14	0,27	0,11	0,18
(2)-3	0,30	0,09	0,26	0,06
(3)-4	0,34	0,00	0,33	0,03
(4)-5	0,43	0,26	0,39	0,21
(5)-6	0,46	0,20	0,39	0,14
Keskiarvo	**0,37**	**0,22**	**0,33**	**0,17**

Taulukko 11. *Alinäytteistysmenetelmien vertailun suoritusajat*

Testitapaus →	**laatikointimenetelmä**	**satunnaismenetelmä**
Vaihe ↓		**Ajan kuluminen (s)**
Esikäsittely	89,8	88,5
SLAM	686,7	1256,3

Aikaisemmin tässä työssä aliluvussa 4.5 arvioitiin, että laatikointimenetelmä on satunnaismenetelmää parempi, koska se ei hukkaa yksittäisiä pisteitä kuten esimerkiksi käsidesipulloa, joka saattaisi olla tärkeä kiintokohde. Tässä havaittiin, että parempi tulos paikannuskyvyn osalta oli satunnaismenetelmä. Tätä tulkintaa tukee lähteessä [52, s. 3] esitelty tutkimus, jossa laatikointimenetelmän on todettu aiheuttavan nimenomaan pisteiden hukkaamista. Satunnaismenetelmä saattaa huonolla tuurilla hukata pisteitä jostakin. On teoriassa mahdollista, että satunnaisesti valittuna kaikki pisteet jostain elementistä jäävät pois pistepilvikuvasta. Tämä satunnaismenetelmä kuitenkin lisää arvokasta tietoa ympäristöstä pistepilvien tiheydestä. Huomataan myös, että erot ovat suhteellisen

pieniä. Arvio esikäsittelyn kestossa ei vastannut aiempaa oletusta siitä, että laatikointimenetelmä vaatisi suurempaa suorituskykyä.

Suurin havainto on melkein kaksinkertainen ero SLAM-skriptin suoritusajassa satunnaismenetelmää käyttämällä. NDT-menetelmä perustuu pistejakaumaan sen perusteella, miten pisteet ovat jakautuneet pistepilvessä. Mikäli toteutetaan alinäytteistys laatikointimenetelmällä, menetetään arvokasta dataa, koska datasta häviää pistejakauma. Pistepilvi säilyttää edelleen muotonsa, mutta pistetiheytymät eivät tule näkyviin, mikäli käytetään alinäytteistysmenetelmänä laatikointimenetelmää. Epäily tässä kaksinkertaisessa erossa on siinä, että algoritmi hyötyy tehokkuuden näkökulmasta pistepilveä laatikoidessaan siitä, että pistepilvet ovat jo aiemmin järjestetty etäisyyden mukaan, jonka takia se ei tarvitse niin suurta suoritusaikaa. Jatkotutkittavaksi jää se, mitä tapahtuu kartan oikeellisuudelle, jos esikäsittelyssä poistetaan entisestään lisää pisteitä.

4.13 SLAMien vertailun toinen kierros

Edellisessä luvussa 4.12 vertailtiin laskennallisiin tuloksiin perustuvilla menetelmillä esikäsittelykombinaatioita. Aiemmassa vaiheessa luvussa 4.11 SLAMeja vertailtaessa valittiin esikäsittelykombinaatio silmämääräisesti. Tässä luvussa vertaillaan laskennallisella menetelmällä valittu paras esikäsittelykombinaatio, jolla verrataan uudestaan SLAM-esimerkkiskriptejä.

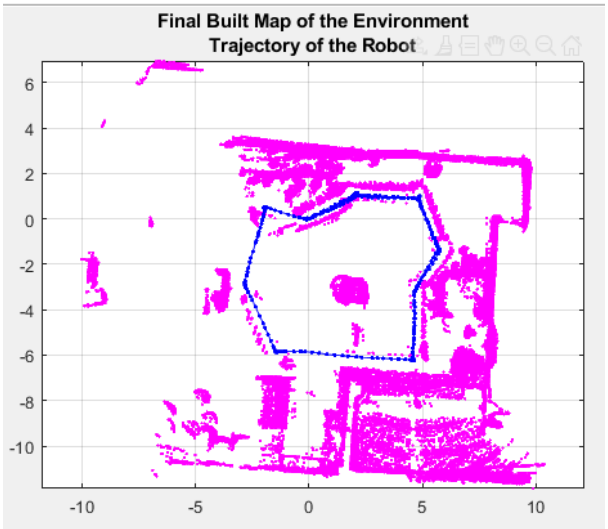
Aliluvussa 4.12.1 vertailtiin eroja sen välillä, onko kannattavaa ottaa tiheämpi näytteenottotaajuus vai suurempi määrä pisteitä yhteen pistepilveen. Tässä todettiin pieni ero SLAMin suoritusajassa, mutta esikäsittely kesti kauemmin, koska pistepilviä oli useampia. Tämän takia valitaan kompromissiksi se, että otetaan joka toinen pistepilvi.

Aliluvussa 4.12.2 vertailtiin häiriösuodatusta. Todettiin, että tätä ei kannata ottaa käyttöön, koska suoritusaika on suhteettoman suuri suhteessa hyötyyn.

Aliluvussa 4.12.3 vertailtiin alinäytteistysmenetelmiä. Tässä todettiin eroavaisuus SLAMin kaksinkertaisessa suoritusajassa. Vastaavasti kartan oikeellisuudessa satunnaismenetelmä oli parempi. Valitaan menetelmäksi satunnaismenetelmä, jotta ei hävitetä pistepilven tiheyden informaatiota. Pisteitä valitaan 10 % alkuperäisen pistepilven koosta, jotta saataisiin selville pistepilven pisteiden määrän entisestään pienentämisen vaikutus.

4.13.1 Navigation toolbox

Kartan oikeellisuus kuvassa Kuva 12 ei eroa silmämääräisesti edellisessä vaiheessa havaittuun tulokseen. Silmukkatarkastelun näkökulmasta uudelleen kuljettu reitti vastaa todellisuutta.



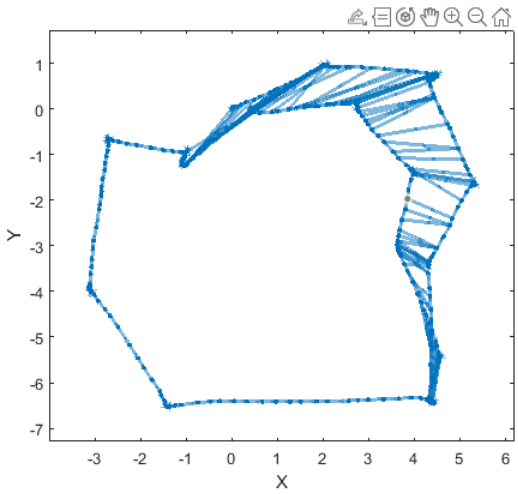
Kuva 12. *"Navigation Toolbox" -esimerkkiskriptin kartan piirto*

Testitulokset ovat esitetty myöhemmässä luvussa. Yksinkertaistuksen vuoksi on esitetty ainoastaan keskiarvot paikannustarkkuuksista.

4.13.2 Computer Vision toolbox

Esimerkkiskriptiä suoritettaessa tapahtui virheen takia keskeytys. Virhe johtui parametristä ”gridStep”. Tätä parametriä nostettiin arvoon 2, jonka jälkeen skriptin suoritus onnistui, ja saatiin alla oleva kartta kuvassa Kuva 13.

Aiemmassa vaiheessa on puhuttu ensiarvausdatasta. Työn ohjaajalta saadun vinkin mukaan kokeillaan vielä tässä vaiheessa lisätä parametrilla ”InitialTransform” alkuarvaus, joka sisältää sekä suunnan- että etäisyydenmuutoksessa nollamuutoksen.

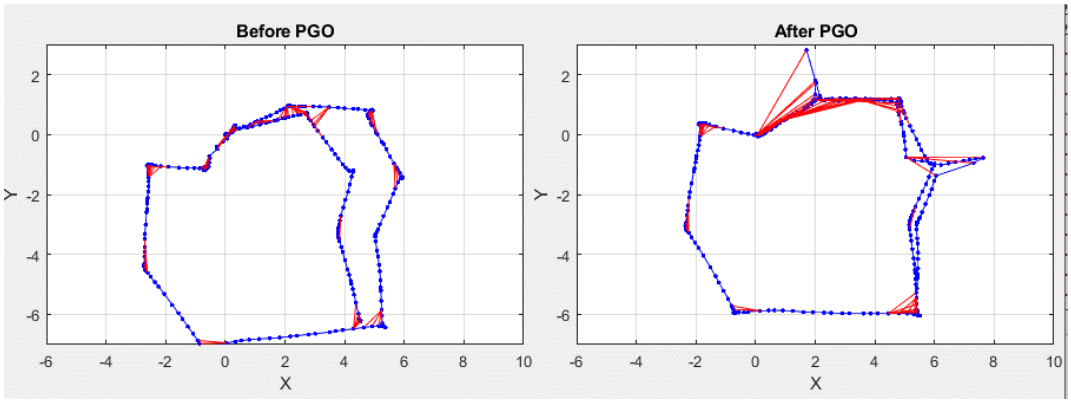


Kuva 13. *”Computer Vision Toolbox” -esimerkkiskriptin kartan piirto*

Aiemmassa vaiheessa karttaan piirretystä reitistä ei tullut realistisen muotoinen. Vastaavasti nyt käytettiin alinäytteistysmenetelmänä satunnaismenetelmää, lisättiin nolla-alkuarvausarvo, ja nostettiin ”gridStep” -parametrin arvoa. Nyt karttaan syntyy karttaan selvää johdonmukaisuutta. Osa kulmapisteistä ovat kuitenkin hieman epätarkkoja. Tämän perusteella havaitaan, että NDT-menetelmää käytettäessä alinäytteistyksen laatikointimenetelmä ei ole kannattava, joka todettiin jo aliluvussa 4.12.3.

4.13.3 Lidar toolbox

Tuloksessa kartan osalta kiinnostaa erityisesti satunnaisen pisteen heitto, joka nähdään kuvassa Kuva 14. Silmukkatarkastelun kulku koostuu tässä esimerkissä siten, että ensin etsitään jokaiselle pisteelle tietyn säteen sisällä olevat pisteet. Sen jälkeen lasketaan etäisyyden- ja suunnanmuutos näille pistepareille. Tästä muodostuu ikään kuin lisäinformaationa oleva tietorakenne. Tässä laskettu informaatio on esitetty punaisina viivoina kuvassa Kuva 14.



Kuva 14. *"Lidar Toolbox" -esimerkkiskriptin kartan piirto*

Tämän jälkeen on laskettu funktiolla ”optimizePoseGraph” [53] [54] lisäinformaatio ja alkuperäinen pisteiden kohdistus vastaamaan parhaiten todellisuutta iteraation avulla. Tämä funktio ei käytä pistepilviä laskennassa, vaan se hyödyntää vain kohdistustietoja eri pisteiden välillä. Funktio käyttää kahta vaihtoehtoista matemaattista menetelmää, jotka ovat ”Trust-region method” tai ”g2o-levenberg-marquardt”. [55] Ensiarvoisesti käytetään ensimmäiseksi mainittua menetelmää, joka suoriutuu tehokkuudeltaan paremmin suurista etäisyyksistä [56]. Syyhyn, miksi tällainen etäisyydenheitto kuvassa syntyi, ei päästy käsiksi. Yksinkertainen menetelmä olisi määrittää maksimietäisyys kahden eri pistepilven välillä. Auton tai työkoneen tapauksessa nopeat kiihtyvyydet eivät ole realistisia.

4.13.4 Tulokset

Alla olevassa taulukossa Taulukko 12 on esitetty lasketut tulokset ajelehtimisesta, lattiamerkkien etäisyyksien keskiarvosta ja suoritusajasta. Tulokset ovat laskettu samalla tavoin kuin aiemmassa vaiheessa, mutta yksinkertaisuuden vuoksi on esitetty ainoastaan keskiarvot.

Taulukko 12. *Toisen iteraatiokierroksen tulokset*

	Navigation Toolbox	**Computer Vision Toolbox**	**Lidar Toolbox**
Ajelehtiman keskiarvo (m)	0,20	-0,04	0,59
Lattiamerkkien etäisyyksien keskiarvo (m)	0,36	1,17	0,45
Suoritusaika (s)	393,3	27,9	228,5

Tuloksista havaitaan, että ”Navigation”- ja ”Lidar” -työkalukirjaston eroavaisuudet lisääntyvät. Suurempi ajelehtima johtuu satunnaisen pisteen siirtymisestä, ja kiertymä aiheuttaa lattiamerkkien etäisyyksissä lisäystä. Lisäksi havaitaan, että ”Computer Vision” -työkalukirjaston esimerkin skriptissä tehdyt muutokset toivat parempia tuloksia.

Tässä vaiheessa ei valita yhtä yksittäistä parasta menetelmää. Voidaan havaita, että ”Navigation”-työkalukirjaston esimerkkiskripti osaa tarkkaa ja vakaata paikannusta. Mikäli halutaan summittainen kartan piirto, käy siihen hyvin ”Computer Vision” -työkalukirjaston esimerkkiskripti, joka on yli kymmenkertaisesti nopeampi.

5. YHTEENVETO

Alkuperäinen tarkoitus oli vertailla eri SLAM-toteutuksia Matlabilla. Toteutukset olisivat voineet olla peräisin monelta eri osapuolelta, jolloin olisi saatu vertailupohjaa Matlabin SLAM-toteutuksiin. Työtä tehdessä Matlabiin soveltuvaa kolmannen osapuolen SLAM-toteutusta oli haastava löytää, jonka takia päädyttiin vertailemaan Matlabin omien työkalukirjastojen SLAM-toteutuksia. Kirjastojen käyttö vaatii perehtyneisyyttä skriptin luomisessa. Koska työn tavoite oli vertailu eikä oman toteutuksen tekeminen, päädyttiin vertaamaan Mathworksin tekemiä esimerkkiskriptejä.

Kukin esimerkkiskripti oli toteutettu omalla tavallaan ja edusti omaa työkalukirjastoa. Kun esimerkkiskriptejä tutkittiin tarkemmin, havaittiin, että nämä skriptit sisältävät päällekkäisyyttä toisiinsa nähden. Lähdekoodia tutkittaessa todettiin, että ”Navigation”- ja ”Lidar” -työkalukirjastot käyttivät pohjimmiltaan samaa kohdistamisfunktiota. Tämä aiheutti samankaltaisuuksia tulosten välillä. Vastaavasti kuitenkin silmukkatarkastelu oli toteutettu eri tavoin, joka antoi mahdollisuuden tutkia näiden eroja. Havaittiin esimerkiksi, että ”optimizePoseGraph” -funktio aiheutti vääristymää.

Työkalukirjastojen eroja tarkastellessa havaittiin, että ”Computer Vision” -työkalukirjaston esimerkki muodosti kaikkein heikoiten kartan oikeellisuuden. Tämä antoi mahdollisuuden tutkia suoritusajan ja kartan oikeellisuuden suhdetta, koska tämän suoritusaika oli kaikkein pienin. Toisessa iteraatiokierroksessa todettiin, että alinäytteistyksessä satunnaismenetelmä on laatikointimenetelmää parempi johtuen NDT-algoritmiin perustuvasta pistetiheysjakaumasta. Tämä oli myös yksi tekijä paremman tuloksen saavuttamiseksi ”Computer Vision” -työkalukirjaston esimerkille. Lisäksi tässä vaiheessa työtä havaittiin, että NDT-algoritmin funktion ”pcregisterndt” parametrin ”gridStep” arvoa kasvattamalla saatiin parempia tuloksia. Lisäksi kokeiltiin lisätä ensiarvausdataksi nollasiirtymä. Nämä keinot paransivat entisestään ”Computer Vision” -työkalukirjaston esimerkin lopputulosta, joka osoitti parametrien valinnan tärkeydestä. Suoritusaika pysyi kuitenkin suunnilleen samana. Tämä antaa jatkotutkimuskohteen vaihtamalla ympäristöä esimerkiksi liikenneympäristöön, missä etäisyydet ovat suurempia. Se antaisi myös mahdollisuuden korkeussuunnan tutkimiseen, koska tämä menetelmä otti ainoana sisääntulona 3d-pistepilven.

”Computer Vision” -työkalukirjaston alkuperäisessä esimerkkiskriptissä oli käytetty IMU-anturia raaka-arviona. Käytössä olleessa testissä poistettiin tämä ensiarviodata. Tässä

jatkotutkimuskohde on anturifuusio sen suhteen, että mitä muita antureita voisi yhdistää, mikä on kartan paikannuksen lopputulos ja mikä olisi suoritusaika tässä tapauksessa.

Esikäsittelyvaihe todettiin tärkeäksi osaksi suorituskyvyn kannalta. Tämän suhteen vertailtiin sitä, mitä esikäsittelymenetelmiä on kannattava tehdä pistepilvelle, jotta sen saatu hyöty olisi järkevä suhteessa menetelmän käyttämään suoritusaikaan ja SLAMin kartan oikeellisuuteen.

Kaikki testikohteet käyttivät paikannusmenetelmänä NDT-menetelmää, joka perustuu pistepilven tiheyden todennäköisyysjakaumaan. Toinen käytetty menetelmä on ICP-menetelmä, jota kannattaisi testata jatkotutkimuksissa.

Tässä työssä testattiin SLAMeja, jotka suoritetaan Matlabissa. Tämä sisältää vain yhden valikoima SLAMeja. Esimerkiksi lähteessä [57] on tutkimus SLAMista, jossa on tehty toteutus ROSille. Tämän lähdekoodi on saatavilla Github-kirjastosta [58].

Testin perusteella todettiin, että SLAM osaa paikantaa yksityiskohtaista reitin piirtoa karttaan, kuten havaittiin ”Navigation” -työkalukirjaston esimerkistä. Testeistä kuitenkin nähdään, että ajelehtiminen on ongelma. Absoluuttinen etäisyydenmuutos perustuu edellisiin relatiivisien muutoksien summaan, jonka takia virhe kasvaa etäisyyden kasvaessa. Silmukkatarkastelulla saadaan korjattua virhettä, mutta se vaatii vierailun uudelleen samassa pisteessä. Tämä puhuu fuusion tärkeydestä esimerkiksi GPS:n kanssa, joka ei perustu edellisien suhteellisien suunnanmuutoksien summaan.

LÄHTEET

[1] M. Frank, ”A Step Towards the Design of Collaborative Autonomous Machines: A Study on Construction and Mining Equipment,” Blekinge Institute of Technology, 2019. [Online]. Available: http://bth.diva-portal.org/smash/record.jsf?pid=diva2%3A1371956&dswid=2916. [Haettu 29.1.2022].

[2] K. I. HYOUNG [KR], C. P. HO, K. D. KEUN, L. Y. SUNG, HUH JI MIN + (강일형, 최평호, 김도근, 이윤성 ja 허지민), ”Safety Management System Using a Lidar for Heavy Machine,” Espacenet, 14.1.2020. [Online]. Available: https://worldwide.espacenet.com/publicationDetails/biblio?FT=D&date=20200114&DB=EPODOC&CC=KR&NR=102065975B1#. [Haettu 29.1.2022].

[3] Jenoptik, ”Innovative LiDAR Technologies,” [Online]. Available: https://www.jenoptik.com/products/lidar-sensors-technologies. [Haettu 30.1.2022].

[4] felix.rohrba.ch, ”An Introduction to LiDAR,” 4.2.2015. [Online]. Available: https://felix.rohrba.ch/en/2015/an-introduction-to-lidar/. [Haettu 30.01.2022].

[5] D. Wang, ”MEMS Mirrors for LiDAR: A review,” ResearchGate, [Online]. Available: https://www.researchgate.net/publication/340971588_MEMS_Mirrors_for_LiDAR_A_review. [Haettu 30.1.2022].

[6] J. Doe, ”ROS Introduction,” ROS Documentation Wiki, 8.8.2018. [Online]. Available: http://wiki.ros.org/ROS/Introduction. [Haettu 28.1.2022].

[7] Niryo, ”8 reasons why you should use ROS for robotics projects,” Niryo, [Online]. Available: https://niryo.com/2018/01/8-reasons-use-ros-robotics-projects/. [Haettu 28.1.2022].

[8] T. R. Back-End, ”What is a ROS Node?,” [Online]. Available: https://roboticsbackend.com/what-is-a-ros-node/. [Haettu 28.1.2022].

[9] R. Wiki, ”ROS Nodes,” ROS Documentation Wiki, 4.12.2018. [Online]. Available: http://wiki.ros.org/Nodes. [Haettu 28.1.2022].

[10] R. D. Wiki, ”ROS Topics,” ROS Documentation Wiki, 20.2.2019. [Online]. Available: http://wiki.ros.org/Topics. [Haettu 15.2.2022].

[11] R. D. Wiki, ”Understanding ROS Topics,” ROS Documentation Wiki, 18.7.2019. [Online]. Available: http://wiki.ros.org/ROS/Tutorials/UnderstandingTopics. [Haettu 28.1.2022].

[12] R. D. Wiki, ”ROS messages,” ROS Documentation Wiki, 13.1.2019. [Online]. Available: http://wiki.ros.org/msg. [Haettu 28.1.2022].

[13] R. Documentation, ”PointCloud2 Message,” ROS Documentation, [Online]. Available: http://docs.ros.org/en/noetic/api/sensor_msgs/html/msg/PointCloud2.html. [Haettu 28.1.2022].

[14] R. D. Wiki, ”ROS/NetworkSetup,” ROS Documentation Wiki, 8.8.2018. [Online]. Available: http://wiki.ros.org/ROS/NetworkSetup. [Haettu 25.1.2022].

[15] Mathworks, ”SLAM,” MathWorks, [Online]. Available: https://se.mathworks.com/discovery/slam.html. [Haettu 28.1.2022].

[16] G. SLAM, ”What is SLAM (Simultaneous Localization and Mapping)?,” Geo SLAM, [Online]. Available: https://geoslam.com/what-is-slam/. [Haettu 15.2.2022].

[17] A. Chau, K. Sekiguchi, A. A. Nugraha, K. Yoshii ja K. Funakoshi, ”Audio-Visual SLAM towards Human Tracking and Human-Robot Interaction in Indoor Environments,” ResearchGate, [Online]. Available: https://www.researchgate.net/publication/338592307_Audio-Visual_SLAM_towards_Human_Tracking_and_Human-Robot_Interaction_in_Indoor_Environments. [Haettu 28.1.2022].

[18] F. Herranz, A. Llamazares, E. Molinos, M. Ocana ja M.-A. Sotelo, ”WiFi SLAM algorithms: an experimental comparison,” ResearchGate, 2014. [Online]. Available: https://www.researchgate.net/publication/265205286_WiFi_SLAM_algorithms_an_experimental_comparison. [Haettu 28.1.2022].

[19] C. S. K. A. M. B. Wolfram Burgard, ”Introduction to Mobile Robotics,” Uni Freiburg, [Online]. Available: http://ais.informatik.uni-freiburg.de/teaching/ss12/robotics/slides/12-slam.pdf. [Haettu 15.2.2022].

[20] L. f. j. t. y. (LFTY), ”Posterinäyttelyn lyhennelmät,” Lääketieteellisen fysiikan ja tekniikan yhdistys (LFTY), 9.1.2003. [Online]. Available: http://www.lfty.fi/lft_paiva/abstraktit03.html. [Haettu 15.2.2022].

[21] H. Z. &. N. A. M. Djehaich, ”ICP-SLAM Methods Implementation on a Bi-steerable Mobile Robot,” ResearchGate, 2013. [Online]. Available: https://www.researchgate.net/publication/261243159_ICP-SLAM_methods_implementation_on_a_bi-steerable_mobile_robot. [Haettu 26.01.2022].

[22] P. J. N. D. M. Besl, ”A Method for Registration of 3-D Shapes.,” IEEE Xplore, 1992. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=121791. [Haettu 28.01.2022].

[23] T. G. University of Tübingen, ”The normal distributions transform: a new approach to laser scan matching,” IEEE, [Online]. Available: https://ieeexplore.ieee.org/document/1249285. [Haettu 26.1.2022].

[24] M. Behl, ”F1tenth (F1/10) Lecture 8]: Scan Matching with LIDAR data,” University of Virginia, 7.4.2021. [Online]. Available: https://www.youtube.com/watch?v=LETPf6eoyYg. [Haettu 21.2.2022].

[25] M. V. A. V. Daniele Cattaneo, ”LCDNet: Deep Loop Closure Detection and Point Cloud Registration for LiDAR SLAM,” IEEE, [Online]. Available: https://arxiv.org/pdf/2103.05056.pdf. [Haettu 9.2.2022].

[26] M. Magnusson, ”The Three-Dimensional Normal-Distributions Transform --- an Efficient Representation for Registration, Surface Analysis, and Loop Detection,” ResearchGate, [Online]. Available: https://www.researchgate.net/publication/229213868_The_Three-Dimensional_Normal-Distributions_Transform_---_an_Efficient_Representation_for_Registration_Surface_Analysis_and_Loop_Detection. [Haettu 9.2.2022].

[27] J. Griffiths, ”Sensor Technology: 2D vs. 3D,” RedZone Robotics, 1.12.2020. [Online]. Available: https://redzone.com/nr/sensor-technology-2d-vs-3d/. [Haettu 8.12.2021].

[28] Mathworks, ”PointCloud2,” Matlab documentation, [Online]. Available: https://se.mathworks.com/help/ros/ref/pointcloud2.html. [Haettu 14.12.2021].

[29] Mathworks, ”pcviewset,” Matlab documentation, [Online]. Available: https://se.mathworks.com/help/vision/ref/pcviewset.html. [Haettu 13.12.2021].

[30] R. d. wiki, ”ROS Installation Options,” ROS documentation wiki, [Online]. Available: http://wiki.ros.org/ROS/Installation. [Haettu 28.1.2022].

[31] Ubuntu, ”Ubuntu 18.04.6 LTS (Bionic Beaver),” Canonical Ltd. Ubuntu, [Online]. Available: https://releases.ubuntu.com/18.04.6/?_ga=2.140828983.462565659.1636045811-116373294.1636045811. [Haettu 28.1.2022].

[32] R. d. wiki, ”Ubuntu install of ROS Melodic,” ROS documentation wiki, [Online]. Available: http://wiki.ros.org/melodic/Installation/Ubuntu. [Haettu 28.1.2022].

[33] R. d. wiki, ”Getting Started with the Velodyne VLP16,” ROS documentation wiki, [Online]. Available: http://wiki.ros.org/velodyne/Tutorials/Getting%20Started%20with%20the%20Velodyne%20VLP16. [Haettu 15.2.2022].

[34] AmetistDrake, ”Velodyne ROS driver,” Github, [Online]. Available: https://github.com/ros-drivers/velodyne. [Haettu 13.12.2021].

[35] I. Velodyne LiDAR, ”VLP16 User manual,” Velodyne LiDAR, Inc., [Online]. Available: https://velodynelidar.com/wp-content/uploads/2019/12/63-9243-Rev-E-VLP-16-User-Manual.pdf. [Haettu 13.12.2021].

[36] A. K. Libor Dostálek, Understanding TCP/IP a clear and comprehensive guide to TCP/IP protocols, Packt Publishing, 2006.

[37] Mathworks, ”Preprocessing,” Matlab Documentation, [Online]. Available: https://se.mathworks.com/help/lidar/pointcloudprocessing.html?s_tid=CRUX_lftnav. [Haettu 17.1.2022].

[38] R. d. wiki, ”Pointcloud to laserscan,” ROS documentation wiki, [Online]. Available: http://wiki.ros.org/pointcloud_to_laserscan. [Haettu 26.1.2022].

[39] P. Biber ja W. Straßer, ”The Normal Distributions Transform: A New Approach to Laser Scan Matching,” ResearchGate, [Online]. Available: https://www.researchgate.net/publication/4045903_The_Normal_Distributions_Transform_A_New_Approach_to_Laser_Scan_Matching. [Haettu 3.2.2022].

[40] Mathworks, ”SLAM (Navigation Toolbox),” Matlab documentation, [Online]. Available: https://se.mathworks.com/help/nav/slam.html?s_tid=CRUX_topnav. [Haettu 18.1.2022].

[41] Mathworks, ”lidarSLAM,” Matlab documentation, [Online]. Available: https://se.mathworks.com/help/nav/ref/lidarslam.html. [Haettu 18.1.2022].

[42] Mathworks, ”Implement Simultaneous Localization And Mapping (SLAM) with Lidar Scans,” [Online]. Available: https://se.mathworks.com/help/nav/ug/implement-simultaneous-localization-and-mapping-with-lidar-scans.html. [Haettu 2.2.2022].

[43] D. K. H. R. D. A. Wolfgang Hess, ”Real-Time Loop Closure in 2D LIDAR SLAM,” Google user content, [Online]. Available: https://static.googleusercontent.com/media/research.google.com/fi//pubs/archive/45466.pdf. [Haettu 2.2.2022].

[44] H. Varjotie, Aalto University, 26.6.2019. [Online]. Available: https://aaltodoc.aalto.fi/bitstream/handle/123456789/39815/master_Varjotie_Henri_2019.pdf?sequence=1&isAllowed=y. [Haettu 21.2.2022].

[45] Mathworks, ”Build a Map from Lidar Data Using SLAM,” Matlab documentation, [Online]. Available: https://se.mathworks.com/help/vision/ug/build-a-map-from-lidar-data-using-slam.html. [Haettu 25.1.2022].

[46] F. Höflinger, J. Müller, R. Zhang, L. M. Reindl ja W. Burgard, ”A Wireless Micro Inertial Measurement Unit (IMU),” IEEE Xplore, [Online]. Available: https://ieeexplore-ieee-org.libproxy.tuni.fi/document/6544690. [Haettu 9.2.2022].

[47] A. Perttula, J. Parviainen ja J. Collin, ”Pedestrian Detection with High Resolution Inertial Measurement Unit,” Trepo Tuni, [Online]. Available: https://andor.tuni.fi/permalink/358FIN_TAMPO/1j3mh4m/alma9911269875205973. [Haettu 9.2.2022].

[48] U. K. Diego Galar, ”Sensor Fusion,” ScienceDirect, 2017. [Online]. Available: https://www.sciencedirect.com/topics/engineering/sensor-fusion. [Haettu 27.2.2022].

[49] PointClouds.org, ”How to use Normal Distributions Transform,” PointClouds.org, [Online]. Available: https://pointclouds.org/documentation/tutorials/normal_distributions_transform.html. [Haettu 8.2.2022].

[50] Mathworks, ”Lidar Toolbox,” Mathworks, [Online]. Available: https://se.mathworks.com/products/lidar.html. [Haettu 30.1.2022].

[51] Mathworks, ”Build Map from 2-D Lidar Scans Using SLAM,” Matlab documentation, [Online]. Available: https://se.mathworks.com/help/lidar/ug/build-map-from-2d-lidar-scans-using-slam.html. [Haettu 27.1.2022].

[52] L. Li, X. Kong, X. Zhao ja W. Li, ”SA-LOAM: Semantic-aided LiDAR SLAM with Loop Closure,” ResearchGate, 6.2021. [Online]. Available: https://www.researchgate.net/figure/Visualization-of-proposed-semantic-based-downsampling-Most-methods-use-voxel-grid_fig3_353071267. [Haettu 20.2.2022].

[53] G. Grisetti, R. Kümmerle, C. Stachniss ja W. Burgard, ”A Tutorial on Graph-Based SLAM,” IEEE, 2010. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/5681215. [Haettu 20.2.2022].

[54] L. Carlone, R. Tron, K. Daniilidis ja F. Dellaert, ”Initialization techniques for 3D SLAM: A survey on rotation estimation and its use in pose graph optimization,” IEEE, 30.5.2015. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7139836. [Haettu 20.2.2022].

[55] Mathworks, ”poseGraphSolverOptions,” Matlab documentation, [Online]. Available: https://se.mathworks.com/help/nav/ref/posegraphsolveroptions.html. [Haettu 20.2.2022].

[56] F. V. Berghen, ”Levenberg-Marquardt algorithms vs Trust Region algorithms,” Universit´e Libre de Bruxelles, 12.11.2004. [Online]. Available: http://www.applied-mathematics.net/LMvsTR/LMvsTR.pdf. [Haettu 20.2.2022].

[57] T. Shan ja B. Englot, ”LeGO-LOAM: Lightweight and Ground-Optimized Lidar Odometry and Mapping on Variable Terrain,” IEEE, 1.10.2018. [Online]. Available: https://ieeexplore.ieee.org/document/8594299. [Haettu 25.2.2022].

[58] T. Shan, J. Oršulić ja W. Wentao, ”LeGO-LOAM,” Github, [Online]. Available: https://github.com/RobustFieldAutonomyLab/LeGO-LOAM. [Haettu 25.2.2022].

LIITE A: MATLAB-KOODI

Löytyy github-kirjastosta:

https://github.com/markus-hautala/Matlab-SLAMs

LIITE B: OLEELLISIMMAT MATLAB-KOODIT

start_here.mlx

Kandidaatintyö 2022
Markus Hautala
Matlab SLAMien vertailu käyttäen lidar-anturia ROSin kautta

Tämä skripti sisältää kandidaatintyöhön sisältyvän testauksen. Osiot etenevät siinä järjestyksessä, miten testit ovat edenneet.
Tiedoston voi ajaa kerralla, mutta se vaatii suuren määrän keskusmuistia.

Työtilan valmistelu

```
% Poista kaikki entiset muuttujat, ikkunat ja komentohistoria
clc
clear all
close all


% Lisää alihakemistot
mainfolder = getPath;
addpath(genpath(strcat(mainfolder,'\Algorithms')));
addpath(genpath(strcat(mainfolder,'\Tools')));
addpath(genpath(strcat(mainfolder,'\ignore')));
```

Tallenna data käyttäen joko ROSista lukua tai tallennettua dataa matlab-muodossa. Tallennetun datan paketin latausosoite on alla. Tämä sisältää tiedoston "savedData_final.mat", joka sisältää testaukseen perustuvan matlab-tietotyypin datan.
Hakemistossa on myös tiedosto "originalRosData.bag", joka sisältää alkuperäisen rosbag-tallenteen testauksesta.
https://tuni-my.sharepoint.com/:f:/g/personal/markus_hautala_tuni_fi/EqMrz-bfX0W1BkObC2WGypbwBZJshim-_KwTy8prTLD4_nw?e=M3OFuB

```
choose = "saved"


if choose == "sensor"
    % tallenna anturista
    time = 60;
    topic = "velodyne_points";
    IP_address = "127.0.0.1";
    pcSet = saveRosData(time, topic, IP_address);


elseif choose == "saved"
    % Vaihtoehtoisesti käytetään tallennettua dataa.
    load("savedData_final.mat")
end
```

Esikäsittelytestaus yksittäiselle pistepilvelle. (Luku 4)
Datapaketti sisältää yksittäisen pistepilven "pc", jolla testataan eri esikäsittelymenetelmiä pistepilvikuvien avulla.

```
show1 = showPc('Alkuperäinen esikäsiteltävä', pc);


%% Häiriösuodatus (luku 4.4)
pc_denoised = pcdenoise(pc);
show2 = showPc('Häiriösuodatettu', pc_denoised);


% Kuinka paljon pisteitä hävisi:
losed_points = size_of_pc(pc) - size_of_pc(pc_denoised)




%% Alinäytteistys (luku 4.5)
laatikon_koko = 0.1; % (m)
tic
pc_downsampled_grid = pcdownsample(pc, 'gridAverage', laatikon_koko);
time_of_grid = toc
show3 = showPc('Alinäytteistetty laatikointimenetelmällä', pc_downsamp-
led_grid);


% Prosentuaalinen pistepilven koko verrattuna ennen alinäytteistystä:
pros_koko = size_of_pc(pc_downsampled_grid) / size_of_pc(pc)


% Verrataan satunnaismenetelmään:
tic
pc_downsampled_rand = pcdownsample(pc, "random", pros_koko);
time_of_random = toc
show4 = showPc('Alinäytteistetty satunnaismenetelmällä', pc_downsamp-
led_rand);




% 3D to 2D (luku 4.7)
lidarScan_limited = pc2laser({pc}, -0.5, 1.5);
figure
plot(lidarScan_limited{1});


lidarScan_unlimited = pc2laser({pc}, -50, 50);
figure
plot(lidarScan_unlimited{1});
```

Esikäsittely SLAM-algoritmeille. (Luku 4.10 Valittava data SLAM-algoritmille)

```
eachFrame = 2;
```

```
preprocessedpcSet = preprocess(pcSet, eachFrame);
lidarSet = pc2laser(preprocessedpcSet, -0.5, 1.5);
size_of_psSet(preprocessedpcSet)
```

Reaalikartta lattian teippimerkinnöistä

```
figure
x = groundTruth(:,1);
y = groundTruth(:,2);
plot(x, y)
for t = 1:9
  text(x(t)+0.1,y(t)+0.1,[num2str(t), ':  (', num2str(x(t)), ',',
num2str(y(t)), ')'])
end
```

SLAM-esimerkkiskriptien suoritus

```
disp('Navigation Toolbox')
tic
optimizedPoses = navigationTB_example(lidarSet)
navigationTB_time = toc;
[routelengths_navi, distances_navi] = getResults(optimizedPoses, groundT-
ruth, eachFrame);
```

```
disp('Computer vision toolbox')
tic
computerVisionTB_example1
computerVisionTB1_time = toc;
[routelengths_vision, distances_vision] = getRe-
sults(pcViewSet2vect(vSet), groundTruth, eachFrame);
```

```
disp('Lidar toolbox')
tic
lidarTB_example
lidarTB_time = toc;
[routelengths_lidar, distances_lidar] = getResults(updatedPGraph.nodeEs-
timates, groundTruth, eachFrame);
```

Lasketaan tulokset

```
calculateResults
```

4.16 Pistepilvien jatkokäsittely

Esikäsittelytestausten testaus parhaimmalla SLAM-skriptillä = navigationTB_example

Pisteiden määrä pistepilvissä (kuvaaja)

```
[points_in_pc_mean, lengths_original_pcSet] = mean_points_in_pc(pcSet);
plot(lengths_original_pcSet)
```

Alinäytteistys vs näytteenottotaajuus (Luku 4.16.1)

```
downsampleVSsamplerate1 = SLAMprocess(pcSet, groundTruth);
downsampleVSsamplerate1.eachFrame = 4*1;
downsampleVSsamplerate1.performPcDenoise = false;
downsampleVSsamplerate1.downsamplemethod = 'random';
downsampleVSsamplerate1.downsampleToPointAmount = 4*2000;
downsampleVSsamplerate1 = downsampleVSsamplerate1.runAll


downsampleVSsamplerate2 = SLAMprocess(pcSet, groundTruth);
downsampleVSsamplerate2.eachFrame = 1*1;
downsampleVSsamplerate2.performPcDenoise = false;
downsampleVSsamplerate2.downsamplemethod = 'random';
downsampleVSsamplerate2.downsampleToPointAmount = 1*2000;
downsampleVSsamplerate2 = downsampleVSsamplerate2.runAll


downsampleVSsamplerate_results_map = [downsampleVSsamplerate1.results_map
downsampleVSsamplerate2.results_map];
downsampleVSsamplerate_results_mtime = [downsampleVSsamplerate1.re-
sults_time downsampleVSsamplerate2.results_time];
```

Häiriösuodatuksen vaikutus (Luku 4.16.2)

```
pcdenoise_results1 = SLAMprocess(pcSet, groundTruth);
pcdenoise_results1.eachFrame = 2;
pcdenoise_results1.performPcDenoise = true;
pcdenoise_results1.downsamplemethod = 'random';
pcdenoise_results1.downsampleToPointAmount = 10000;
pcdenoise_results1 = pcdenoise_results1.runAll


pcdenoise_results2 = SLAMprocess(pcSet, groundTruth);
pcdenoise_results2.eachFrame = 2;
pcdenoise_results2.performPcDenoise = false;
pcdenoise_results2.downsamplemethod = 'random';
pcdenoise_results2.downsampleToPointAmount = 10000;
pcdenoise_results2 = pcdenoise_results2.runAll


pcdenoise_results_map = [pcdenoise_results1.results_map pcdenoise_re-
sults2.results_map];
pcdenoise_results_mtime = [pcdenoise_results1.results_time pcdenoise_re-
sults2.results_time];
```

Alinäytteistysmenetelmät (Luku 4.16.3)

```
gridvsrandom_results1 = SLAMprocess(pcSet, groundTruth);
gridvsrandom_results1.eachFrame = 2;
gridvsrandom_results1.downsamplemethod = 'gridAverage';
gridvsrandom_results1.downsampleToPointAmount = 0.1;
gridvsrandom_results1 = gridvsrandom_results1.runAll


gridvsrandom_results2 = SLAMprocess(pcSet, groundTruth);
```

```
gridvsrandom_results2.eachFrame = 2;
gridvsrandom_results2.downsamplemethod = 'random';
gridvsrandom_results2.downsampleToPointAmount = mean_points_in_pc(gridvs-
random_results1.pcSet_preprocessed);
gridvsrandom_results2 = gridvsrandom_results2.runAll


gridvsrandom_results_map = [gridvsrandom_results1.results_map gridvsran-
dom_results2.results_map];
gridvsrandom_results_mtime = [gridvsrandom_results1.results_time gridvs-
random_results2.results_time];
```

SLAMien vertailu laskennallisesti parhaalla esikäsittelymenetelmäkombinaatiolla (luku 4.17)

```
eachFrame_2nd_iteration = 2;
preprocessedpcSet = preprocess_tested(pcSet, eachFrame_2nd_iteration);
lidarSet = pc2laser(preprocessedpcSet, -0.5, 1.5);
```

```
disp('Navigation Toolbox')
tic
optimizedPoses = navigationTB_example(lidarSet)
navigationTB_time2 = toc;
[routelengths_navi2, distances_navi2] = getResults(optimizedPoses,
groundTruth, eachFrame_2th);
```

```
disp('Computer vision toolbox')
tic
computerVisionTB_example_2nd_iteration
computerVisionTB1_time2 = toc;
[routelengths_vision2, distances_vision2] = getRe-
sults(pcViewSet2vect(vSet), groundTruth, eachFrame_2th);
```

```
disp('Lidar toolbox')
tic
lidarTB_example
lidarTB_time2 = toc;
[routelengths_lidar2, distances_lidar2] = getResults(updated-
PGraph.nodeEstimates, groundTruth, eachFrame_2th);
```

Results:

```
calculateResults_2nd
```

pc_fov.m

```
function pc_fov = pc_fov(pc, a, b)
% palauttaa pistepilven, josta on poistettu sektori

% a määrittää kulman negatiivisen y-akselin suhteen myötäpäivään
```

```
% (negatiiviset x:n arvot)

% b määrittää kulman negatiivisen y-akselin suhteen vastapäivään
% (positiiviset x:n arvot)

a_rad = deg2rad(a); % kulma radiaaneiksi
b_rad = deg2rad(b);

points = [];

for p=1 : length(pc.Location) % Tarkastetaan jokainen piste
    take_account = true;

    p_x = pc.Location(p,1);
    p_y = pc.Location(p,2);

    limit_a_y = p_x * tan(a_rad);
    limit_a_x = p_y / tan(b_rad);

    limit_b_y = -p_x * tan(a_rad);
    limit_b_x = -p_y / tan(b_rad);

    % mikäli negatiivisella x:n arvolla kaistaleen sisäpuolella
    if (limit_a_y < p_y && p_y < 0 && limit_a_x > p_x)
        take_account = false;
    end

    % mikäli positiivisella x:n arvolla kaistaleen sisäpuolella
    if (0 < p_y && p_y < limit_b_y && limit_b_x > p_x)
        take_account = false;
    end

    if take_account % Mikäli on kaistaleen ulkopuolella
        points = [points; pc.Location(p,:)];
    end

end

pc_fov = pointCloud(points);

end
```

pc2laser.m

```
function lidarSet = pc2laser(pcSet, min, max)
% Muunnetaan 3D-pistepilvi 2D-kuvaksi
% min ja max määrittävät korkeuden, jonka välillä olevat pisteet
% pistepilvessä z-akselin suhteen otetaan mukaan

lidarSet = cell(1, size_of_psSet(pcSet)); % 2D kuvien setti
set_container = 1;
lidarScans = {}; % Apumuuttuja, johon tallennetaan pisteet

for n=1 : size_of_psSet(pcSet) % Käydään jokainen pistepilvisetin yksilö
```

```
        pc = pcSet{1,n};


        for p=1 : size_of_pc(pc) % Jokainen pistepilven piste
            zAxel = pc.Location(p,3);

            % Verrataan z-akseliin, onko rajojen sisäpuolella
            if (and ( (zAxel > min), (zAxel < max) ) )
                lidarScans{end+1,1} = pc.Location(p,1); % x-koordinaatti
                lidarScans{end,2} = pc.Location(p,2); % y-koordinaatti
            end

        end

        % Muutetaan x- ja y- koordinaattivektori lidarscan-tietotyyppiin
        lidarSet{set_container} = lidarScan(double(cell2mat(lidarScans)));

        set_container = set_container + 1;
        lidarScans = {}; % Tyhjätään apumuuttuja seuraavaa kierrosta varten

    end

end
```