AUTOMAATTINEN TIENMITTAUSAJONEUVO

Khoa Dang
Silvia Lupea
Samuli Multaniemi
Giang Nguyen
Minh Tran

Sisällysluettelo

Kuvat ja taulukot

Omistuskirjoitus

Me, Suomi Geniuses – tiimi, haluamme kiittää SICK Suomea projektin mahdollistamisesta ja siinä käytetyn laserskannerin tarjoamisesta tutkimuskäyttöön. Lisäksi haluamme kiittää Hämeen ammattikorkeakoulua työskentelytiloista ja –välineistä, sekä opastuksesta projektissa. Erityisesti seuraavien henkilöiden avunantoa arvostimme ja hyödynsimme:

- Katariina Penttilä
- Jan-Peter Nowak
- Juhani Henttonen
- Juha Sarkula

Lopuksi haluamme kiittää perheitämme rakkaudesta ja tuesta.

1 Johdanto

Suomen tieliikennelaitoksen mukaan Suomen 140 000 kilometriä laajalla tieverkostolla kulkee vuosittain yli 55 000 autoa. Tienpinnan kunnon mittaamiselle ja sen perusteella tapahtuville korjauksille on suuri tarve ja järkeviä syitä, jopa hallinnollisella tasolla, sillä tienpinnan kunto vaikuttaa merkittävästi ajoneuvojen ja matkustajien turvallisuuteen, taloudellisuuteen ja liikennevirran sulavaan toimintaan.

Infra-alan yhtiöillä, maanlaajuisesti pääasiassa Destialla, on käytössään laitteistoa tieprofiilin mittaukseen ja mm. pinnan pituus- ja poikkisuuntaisten epätasaisuuksien (IRI), ajourien syvyyden ja karkeuden määritykseen. Käytössä olevia laitteistoja rajoittaa ympäristötietojen puuttuminen, sillä profilointilaitteet voivat mitata vain samansuuntaista pintaa, kun taas laserskanneri voi avautumiskulmasta riippuen skannata jopa 270° - 360° erittäin suurella tarkkuudella. Lisäksi laserskannerien työalue on paljon suurempi, jopa kymmeniä metrejä. Laserskannerien ainoa rajoitus on, että niiden viuhkamallisesta toteutuksesta johtuen tienpinnalla skannerista kauempana sivussa olevat mittauspisteet ovat kauempana toisistaan kuin suoraan skannerin alla olevat. Tämä voidaan lähes eliminoida lisäämällä toinen skanneri vaakatasoon tietä vasten, jolloin skannerien mittaustulokset voidaan yhdistää saumattomasti. Vaikka tienprofilointilaitteiston tarkkuus on teoriassa korkeampi, on huomattava, että niin tarkasta mittauksesta ei ole hyötyä, sillä laserskannerin mittaus on myös enemmän kuin riittävä, ja ne ovat paljon edullisempia kuin tien profilointilaitteet.

Tässä raportissa dokumentoidaan projektin suunnittelutyö, sen toiminnallisuus, ja kuvataan myös laajennus- ja parannusideoita ja myös muita mahdollisia ”Automaattisen tienmittausajoneuvon” käyttökohteita. Raportti kuvaa idean toteuttamiskelpoisuutta prototyypillä varustettuna SICK TiM561 – 2D laserskannerilla. Tiimin pääasiallinen kommunikointikieli oli englanti, jonka vuoksi dokumentaatio on myös ensisijaisesti toteutettu englanniksi. Suomenkielinen dokumentaatio on käännetty.

Mitatun ympäristön virtuaalista uudelleenrakentamista varten on tiedettävä skannerin suuntakulmat ja vakionopeus, jolla se liikkuu. Ympäristödatan keräämiseksi skanneri asennettiin ajoneuvoon skannaava osa tietä kohden. Tarkemmin systeemiin kuuluu kolme osaa:

- Automatisoitu ajoneuvo, jota “ajaa” Arduino Mega 2560 – mikrokontrolleri. Ajoneuvon toimintoihin kuuluu nopeuden säätö ja ohjaus käyttäen differentiaalikaksipyöräohjausta. Inertiamittausyksikön syötteestä algoritmi antaa suuntakulmat ajoneuvolle 200 Hertsin taajuudella.
- Raspberry Pi 3 toimii prosessoivana yksikkönä ja rajapintana SICK TiM561 – laserskannerille. Arduinon ja Raspberry Pi:n välillä on 2Mbps sarjaliitäntä.

- Visualisointiohjelmisto, jolla lisätään jälkiprosessointia ja rekonstruoidaan skannattu data.

2 Toteutuksen haaste - ratkaisumalli

Projektin toteutus jaettiin seuraaviin haasteisiin ja niiden ratkaisuun:

1. Olemassaolevan, HAMK:n opiskelijoiden projektityönä toteutetun automaattisen robottiauton uudelleensuunnittelu ja saneeraus.
2. Asettaa ajoneuvo ajamaan vakionopeudella ja mahdollisesti myös sen ohjaus.
3. Ajoneuvon suuntakulmien määritys jälkikorjausta varten.
4. Rakentaa rajapinta SICK TiM 561 – laserskannerille ja tallentaa skannattu data
5. Lisätä jälkikorjaus skannattuun dataan ja uudelleenrakentaa se visualisointiohjelmistolla.

Ratkaisut:

- Ajoneuvon nopeuden säätö sumean logiikan PI-säätimellä.
- Kiihtyvyysanturin, gyroskoopin ja magnetometrisensorien datan lukeminen, näiden yhdistäminen käyttäen Mahoneyn algoritmia ajoneuvon suuntakulmien arvioimiseksi.
- SGPiApp, C++ - ohjelma, Raspberry Pi 3:lla, jonka vastuulla ovat seuraavat:
    - Auton etäohjaus SSH:n tai Team Viewer –ohjelmiston välityksellä.
    - Antaa vakionopeuden asetusarvo Arduinolle sarjaliitännän kautta
    - Kommunikoida SICK TiM 561 – laserskannerin kanssa ja tallentaa skannattu data.
- SGApp, C++ -ohjelmisto, Windows- tai Linux-tietokoneilla, jonka vastuulla ovat seuraavat:
    - Lisätä Savitzky-Golay –filtterialgoritmi skannattuun dataan kohinan/häiriön vähentämiseksi
    - Renderöidä/uudelleenrakentaa skannattu data.

Alla video, joka esittelee ajoneuvon toimintaa:
https://www.youtube.com/watch?v=_Yoan-qcLMU

2.1. Olemassaolevan ajoneuvon mekaaninen uudelleenrakennus

Koska projektin aikataulu oli melko rajallinen täydelle suunnitteluprojektille ja projektissa haluttiin keskittyä uuden innovaation toteutukseen, päätimme hyöyntää jo olemassa olevan robottiauton komponentteja, jotta projektin pääpaino pysyisi SICK TiM561 – laserskannerin hyödyntämisessä. Lisäksi halusimme hyödyntää automaatiotekniikan osaamistamme ajoneuvossa

Tarkemman tarkastelun jälkeen oli selvää, että lähes kaikki tuki- ja systeemirakenteet oli poistettava, sillä ajoneuvon edellinen käyttötarkoitus oli hyvin erilainen. Päätös tehtiin käyttää ainoastaan vankkaa alumiinikehikkoa, käsintehtyjä enkoodereita ja moottoreita ja muut osat vaihtaa uusiin. Alumiinikehikkoon kiinnitettiin uudet alumiinikiskot, joihin kytkentärasiat kiinnitettäisiin ja talvinastat poistettiin renkaista, sillä niiden tuoma lisäpito ei ollut tarpeellista, ja ne myös lisäsivät tärinää. Komponenttien järjestely kytkentärasioihin ja niiden johdotus oli melko suoraviivaista, ja kytkentärasioihin jätettiin tarkoituksella tilaa laajennuksia ja muutoksia varten.

Toiminnallisuus valittiin tärkeimmäksi kriteeriksi ulkonäön sijasta fyysisessä toteutuksessa, sillä projektissa haluttiin tutkia ja todistaa laserskannerin toimivuus tien/ympäristön kunnon mittaustarkoituksessa.

2.2. Vakionopeuden säätö ja inertiamittausyksikön sulautusalgoritmi

Tässä projektissa ajoneuvon vakionopeuden säätämistä ja suuntakulmien määrittämistä inertiamittausyksikön sulautusalgoritmin avulla ohjaa Arduino Mega 2560 (pohjautuen Atmel ATMEGA 2560:aan) – mikrokontrolleri. Tämän ohjelmointi tapahtui Arduino Wiring – alustan ja Atmel:n C++ -alustan avulla:

- Arduino Wiring -alusta:
    - Abstrakti ja yksinkertainen rajapinta mikrokontrollerin yleisien toimintojen ohjelmointia varten.
    - Mikrokontrollereiden oheisväylien, kuten I^2C –tiedonsiirtoväylän ja sarjaliitännän alustus ja käyttö.
    - Heikko optimointi ja kustomistointi verrattuna perinteisiin mikrokontrollereiden ohjelmointikieliin
    - Atmel:n C++ -koodi voidaan liittää osaksi Arduino Wiring – alustan ohjelmia.
- Atmel:n C++ -ohjelmointi
    - Täysin kustomisoitavat toiminnot (ajastimet, laskurit, keskeytykset, rotajne.)

- Vaatii vahvaa osaamista sulautettujen järjestelmien ohjelmoinnista. Esimerkiksi toimintojen alustaminen voi olla aloittelijalle monimutkaista.
- Kääntäjän optimointi voidaan valita ohjelman tiedostokoon ja nopeuden väliltä, tai jopa niiden yhdistelmänä.

Lopullinen sulautettu ohjelma hyödynsi molempia alustoja, ohjelmoinnin yksinkertaistamisen, mutta samalla toiminnallisuuden säilyttämisen vuoksi. Ohjelman ominaisuudet ovat seuraavat:

- Keskeytyksiin perustuva enkooderien näytteenotto, pyörimisnopeuden tarkistus 30 millisekunnin välein
- Vakionopeuden asetus käyttäen ennakoivaan sumean logiikan PI – säädintä.
- Mahoneyn algoritmi inertiamittausyksikön sensoreiden datan yhdistämiseksi, taajuus 200 Hertsiä.
- Sarjaliitäntä 2 miljoonaa bittiä sekunnissa (2 Mbps).

Ohjelmoinnissa käytettiin vapaan lisenssin, avoimen lähdekoodin kirjastoja Arduino – ohjelmoijien yhteisön toteuttamina. Seuraavia kirjastoja tai niiden osia hyödynnettiin:

- Encoder library by Paul Stoffregen: https://github.com/PaulStoffregen/Encoder - Käytettiin ilman muokkauksia.
- PinChangeInterrupt library by GreyGnome: https://github.com/GreyGnome/PinChangeInt - Käytettiin ilman muokkauksia.
- TimerThree library by Paul Stoffregen: https://github.com/PaulStoffregen/TimerThree - Käytettiin ilman muokkauksia. TimerFour- ja TimerFive-kirjastot kirjoitettiin itse projektia varten TimerThree –kirjastoon ja ATMEGA2560 – mikrokontrollerin tekniseen selvitykseen perustuen.
- MPU-9250 example sketch by kriswiner: https://github.com/kriswiner/MPU-9250 - Muokattiin käyttämään keskeytyksiä, pienempää ohjelmakokoa, luonnonvakioita ja optimoituja muuttujia.

2.2.1. Vakionopeuden säätö

Ajoneuvon vakionopeuden säätö on toteutettu epäsuorasti ohjaamalla renkaiden moottorien pyörimisnopeutta. Arduino vastaanottaa Raspberry Pi:ltä asetusarvon, yksikkönä millimetrejä sekunnissa ja lähettää hetkellisen nopeuden mitattuna renkaan pyörimisnopeudesta, renkaan säteen ja suuntakulman takaisin.

Yksinkertainen sumean logiikan PI – säädin suunniteltiin mittaamaan renkaan pyörimisnopeutta yksikkönä radiaania sekunnissa ja muuttamaan pulssia pulssinleveysmodulaation (PWM) avulla, tarkoituksena pitää pyörimisnopeus mahdollisimman lähellä asetusarvoa. Pulssinleveysmodulaation taajuus asetettiin 20 kilohertsiin, ei-toivotun häiriön välttämiseksi käyttäjälle. Vaikka teknisissä selvityksissä mainittiin käytettyjen komponenttien tuki 25 kilohertsiin asti, kokeiluissa hyötysuhde laski, sillä komponentit alkoivat lämmetä liikaa, joten päädyttiin käyttämään 20 kilohertsiä.

Ajoneuvon enkooderimoduulit valmistettiin manuaalisesti, jonka vuoksi niillä ei ole kovin korkeaa pulssilukua (178 PPM) ja A-B ulostulon vaihe-ero ei ole täysin 90 astetta. Ne kuitenkin toimivat tarpeeksi korkealla tarkkuudella mitatakseen ajon suuntaa ja nopeutta 33 Hertsin taajuudella.

Kolme korkean resoluution ajastinta ja neljä ulkoista keskeytystä ATMEGA2560 mikrokontrollista käytettiin tätä varten. Ulkoisia keskeytyksiä käytettiin enkooderien ulostulosignaalia varten. ”Timer 3” – ajastinta käytettiin nopeuden laskennan aikataulutukseen ja nopeuden säädön alarutiiniin. ”Timer 4”- ja ”Timer 5” – ajastimia käytettiin generoimaan 20 kilohertsin PWM – signaali ajamaan moottoreita.

2.2.2. Inertiamittausyksikön yhdistysalgoritmi

Projektissa käytetty inertiamittausyksikkö oli Invensense MPU-9250, sisältäen Invensensen gyroskoopin, kiihtyvyysanturin ja magnetometrin. Yksikkö konfiguroitiin ottamaan näytteitä kiihtyvyydestä ja pyörimissuuntaisesta nopeudesta 1000 Hertsin taajuudella, josta laskettiin keskiarvo alas 200 Hertsiin vastaamaan magnetometrin näytteenottotaajuutta. Yksikkö kommunikoi Arduinon kanssa 400 kilohertsin I2C – väylän välityksellä ja käyttää keskeytyspinniä viestittääkseen Arduinolle aloittaa yhdistysrutiini. Figure 1 esittelee kiinteän kappaleen suuntakulmia 3D – avaruudessa.

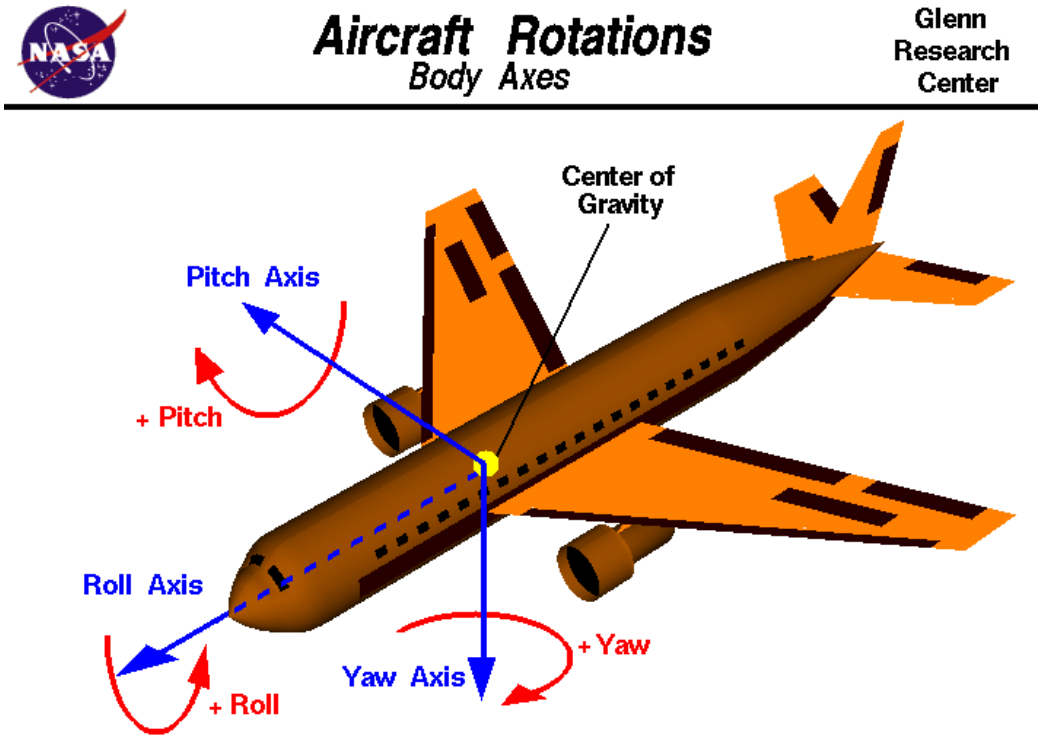

Figure 1 Suuntakulmat (NASA, 2015)

Abyarjoon (Implementing a Sensor Fusion Algorithm for 3D Orientation Detection with Inertial/Magnetic Sensors, 2012) mukaan mittausperusteet yhdistysalgoritmille voidaan kiteyttää seuraavasti:

- Integroidaan gyroskoopin mittaustulokset (ω_x, ω_y, ω_z) ajan kuluessa saaden kulman muutokset
    - Tämä metodi kärsii gyroskoopin mittauksen häiriökohinasta, joka johtaa mitatun kulman ajelehtimiseen.
- Käytetään mitattua kiihtyvyyttä (a_x, a_y, a_z) ja lasketaan kulmat niiden välillä kukin omilla tasoillaan saaden suuntautumiskulmat.
    - Mitattu häiriökohina tässä mittauksessa johtaa häiriöisiin kulmamittauksiin, vaikka mitatut arvot eivät ajelehdi.
- Käytetään magnetometrin mittauksia (m_x, m_y, m_z) to suuntakulmien määrittämiseksi.
    - Magnetometrin näytteenottotaajuus on yleensä alhaisempi kuin kahden edellämainitun, ja mittaa ainoastaan yaw-kulmaa.

Mahonyn yhdistysalgoritmi arvioi Maan painovoimavektorin, g_{earth} ja painovoimavektorin sensorissa, g_{body}. Määrittelemällä pyörimismatriisin joka muuttaa g_{earth}:n g_{body}:ksi, esineen suuntakulmat voidaan johtaa. Mahoney käytti kvaterinoita esittämään pyörimismatriisia, sillä tämä metodi on laskennallisesti kevyempää verrattuna euklidiseen esitystapaan.

Sensorin suuntautuminen 3D – avaruudessa aikana n esitetään tilan kvaterniona $q[n]$. Tämän kvaternion muutoksen nopeus $\dot{q}[n]$ on edellisen tilan kvaternion $q[n-1]$ ja korjatun pyörimisnopeuden $\Omega[n]$ kvaterniotulo:

$$q[n-1] = [q_1 \quad q_2 \quad q_3 \quad q_4]$$
$$\dot{q}[n] = \frac{1}{2}\, q[n-1] \otimes \Omega[n] = \frac{1}{2}\, q[n-1] \otimes [0 \quad \bar{\omega}_x \quad \bar{\omega}_y \quad \bar{\omega}_z]$$
$$q[n] = q[n-1] + \dot{q}[n] * T_S$$

T_S on systeemin näytteenottoaika, joka on 5 millisekuntia.

Tilakvaterniota käytettiin g_{body}:n määrittämiseen; kun taas kiihtyvyysanturin ja magnetometrin mittauksia käytettiin g_{earth}:n arvioimiseen. g_{body}:n ja g_{earth}:n välinen erotus, jota voidaan pitää ”virheenä” käytettiin takaisinkytkentäterminä gyroskoopin mittaustuloksen korjauksessa ennen integrointia, käyttäen PI-säädintä, johtaen termin $\Omega[n]$.

Tätä projektia varten Madgwickin toteutusta Mahoneyn yhdistysalgoritmista käytettiin ajoneuvon suunnan arvioimiseen MPU-9250 – inertiamittausyksikön mitattujen arvojen perusteella.

2.3. SGPiApp – Raspberry Pi 3 -ohjelma

Raspberry Pi:lle luotiin ohjelma, joka skannatun datan kerää SICK TiM561 – skannerista ja tietoa ajoneuvon liikkeestä ja tallentaa ne.

Ohjelma suunniteltiin käyttämään montaa säiettä paremman suorituskyvyn vuoksi, sillä Raspberry Pi:n neliytiminen prosessoriarkkitehtuuri mahdollistaa sen. Monisäikeinen ohjelmointi olisi mahdollistanut myös konenäkösysteemin toteuttamisen, joka kuitenkin jätettiin tässä projektissa toteuttamatta aikamääreiden vuoksi.

2.3.1. Alustat ja ohjelmointirajapinnat

Ohjelma kohdistettiin ajettavaksi Raspbian Jessiellä, Debian-pohjaisella Linux käyttöjärjestelmällä. Lähdekoodi käännettiin ja rakennettiin kyseisellä järjestelmällä, vaikka ohjelmointi ja testaus toteutettiin toisella tietokoneella, myös Linux-pohjaisella Ubuntu – käyttöjärjestelmällä. Järjestelmät ovat melko samankaltaiset, joten ongelmia ei juurikaan ilmennyt.

Lähdekoodin rakentamista varten Raspberry Pi:hin täytyy yhdistää etätyöpöydän avulla Internetin kautta käyttämällä joko SSH- tai TeamViewer – ohjelmia.

Ohjelma kehitettiin käyttäen C++ -ohjelmointikieltä, noudattaen GNU c++ -kääntäjää. Käytetyt ohjelmointirajapinnat olivat seuraavat:

- libusb rajapinta SICK TiM561- laserskannerin ja Raspberry Pi:n välillä
- pthread monisäieohjelmointia varten
- RS232 Arduinon sarjaliitäntää varten

2.3.2. Ajoneuvon liikkeen tiedon keräys

Ajoneuvon liikkeeseen liittyvät tiedot ovat käytännössä ajoneuvon nopeus ja yaw- pitch- ja roll – suuntakulmat. Kuten aiemmin mainittiin, suuntakulmat lasketaan inertiamittausyksikön datan perusteella. Kaikki nämä neljä tietoa ovat reaalityyppisiä lukuja (floating point, 4 byte). Arduino on yhdistetty Raspberry Pi:hin USB – rajapinnan kautta, ja saa sitä kautta myös virtaa sarjaliitäntäyhteyden lisäksi. SGPiApp käyttää RS232 – rajapintaa sarjaliitännän kirjoitus- ja lukutoimintoihin Arduinon kanssa. Joka 33 millisekunti ohjelma lähettää pyyntökomennon sisältäen merkin ”R” ja halutun nopeuden asetusarvon. Arduino vastaa lähettämällä suuntakulmat yhdessä todellisen nopeuden kanssa ja lisää mukaan varmistuskoodin ”0xFA”.

Jokaisen mittausistunnon alussa SGPiApp käynnistetään käyttäjän toimesta. Ohjelma lukee komentoriviltä käyttäjän antaman halutun nopeuden asetusarvon. Alustamisvaiheessa sarjaliitäntä Arduinon kanssa avataan, ohjelma odottaa alustamistoimenpiteiden ajan, jonka jälkeen Arduino lähettää varmistuskoodin 0xFA. Tämän jälkeen ohjelma pyytää ensimmäisen datapaketin tarkistamista varten. Jos mitään suuntakulmista näyttää olevan kalibroimaton, eli näyttää jotain muuta kuin nollaa, ohjelma sulkee sarjaliitännän ja avaa sen uudelleen, kunnes arvot ovat oikein. Tämä johtuu Arduinon rakenteellisesta viata, eikä ole muuten helposti korjattavissa.

2.3.3. Laserskannerin datan keräys

SICK TiM561 – laserskannerin datan keräys tapahtuu USB – liitännän kautta Raspberry Pi:llä. Vastaanottaakseen dataa skannerilta, ohjelma ensin alkuvalmistelee konfiguraatioparametrit sensorille ja aloittaa datapakettien syötteen. Jokainen datapaketti skannerilta sisältää tietoa erillisestä skannauksesta (sensori skannaa 15 Hertsin taajuudella). Ohjelma jäsentää paketin datan ja tuottaa uuden rakennetun paketin, jossa skannattu data on tiivistetty ja joka sisältää myös tietoa ajoneuvon liikkeestä.

Table 1 kuvaa ohjelman jäsentämää pakettirakennetta ja tietomuotoja.

Field name		Description	Data type	Size (bytes)
Num_points		Number of points in one scan	uint32	4
Time_stamp	Hour	The time at which the scan was taken	uint32	4
	Minute		uint8	1
	Second		uint8	1
	Millisecond		uint8	1
Velocity		Velocity of the car when the scan was taken	float	4
Rotation angles	Yaw	Yaw, pitch and roll angles taken from the inertial measurement device when the scan was taken	float	4
	Pitch		float	4
	Roll		float	4
Range list		A vector containing the ranges or the distance between each data point in the scan and the sensor	float array	4*Num_points

Table 1 Jäsennetty pakettirakenne ja tietomuodot

Huom: uint32 – unsigned integer 32 bit, uint8 – unsigned integer 8 bit

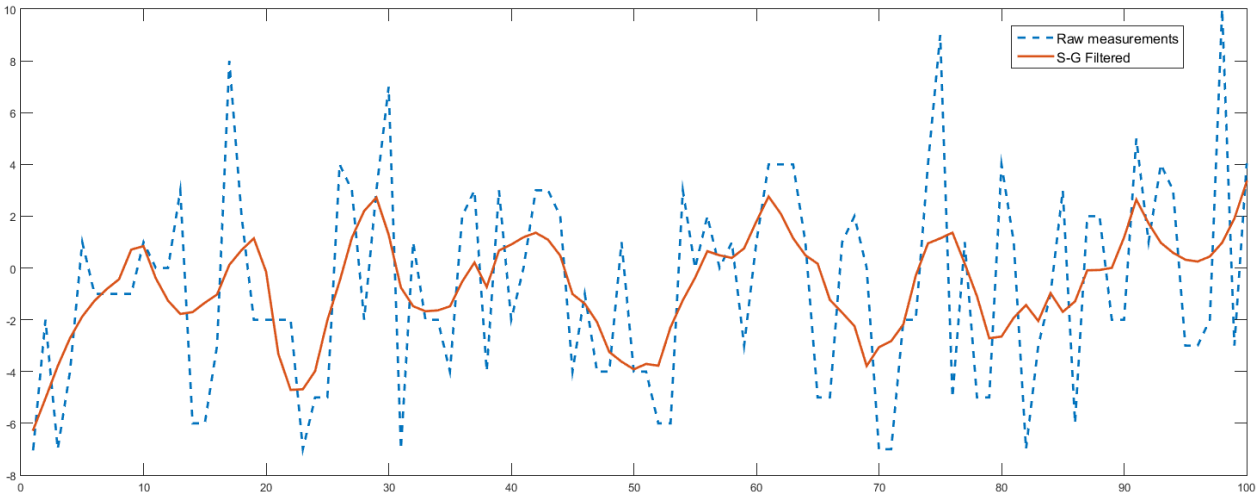

Figure 2 Yhden pisteen skannausalue sadan näytteen ajalta

Figure 2 esittelee TiM561 – skannerin mittaustulosta tasaisesta pinnasta, 251 millimetriä sen yläpuolella, 90 asteen kulmassa alaspäin. Raaka arvo erotetaan sitten 251:sta josta saadaan tason etäisyys täydellisestä arvosta. Varianssin suuruus on -7:stä +10 millimetriin, joten ± 15mm epätarkkuus voidaan johtaa skannerin

tarkkuudeksi. Kuvassa näkyy myös Savitzky-Golay filtteri lisätty, jonka jälkeen laskettu keskipoikkeama laski 3,84 millimetristä 1.93 millimetriin ja voidaan jopa silmämääräisesti todeta, että mittaustulos parani selkeästi.

2.3.4. Dokumentoitu skannaus

Jokaiselle skannausistunnolle luodaan automaattisesti oma ”Road Record” -tietallennedokumentti, joka sisältää kaiken skannatun datan ja konfiguroinnin. SGPiApp- ohjelma tallentaa kaikki datapaketit vektoriin laatien samalla Road Record – dokumentin sisällön. Ohjelma tallentaa tämän automaattisesti SD-kortille, josta se voidaan jakaa myöhempää käyttöä varten. Muistin rajallisuuden vuoksi ohjelma jakaa jäsennetyt paketit tietyin aikavälein ns. varmuuskopiotiedostoihin ja kun skannausistunto päättyy, nämä varmuuskopiotiedostot yhdistetään Road Record -dokumentiksi

Field name		**Description**	**Data type**	**Size (bytes)**
Num_packs		Number of packs in a record	uint32	4
Start time	Year	The time at which the program starts recording the session	uint32	4
	Month		uint8	1
	Day		uint8	1
	Hour		uint8	1
	Minute		uint8	1
	Second		uint8	1
Start angle		Starting angle for each 2D scan	float	4
End angle		Ending angle for each 2D scan	float	4
Angular step		The angle resolution of each scan	float	4
Data		A list of data scan packs	Scan data structure (see Table 1)	(Size of scan data structure) * Num_packs

Table 2 kuvaa Road Record – dokumentin sisältöä.

Road Record – dokumentin tiedostomuoto on .dat, esimerkiksi “record_file.dat”.

2.4. SGApp – Datan prosessointi ja visualisointiohjelma

SGApp on työpöytäsovellus joka kehitettiin skannatun datan prosessointia ja sen uudelleenkonstruktointia 3D – avaruudessa varten

2.4.1. Alustat ja ohjelmointirajapinnat

Ohjelma kohdistettiin ajettavaksi Windows – ja Linux – käyttöjärjestelmillä. WxWidgets – kirjastoa käytettiin rakentamaan graafinen ikkuna, ja OpenGL ohjelmointirajapintoja käytetään skannatun datan 3D visualisointiin.

2.4.2. Datan jälkikäsittely

Savitzky-Golayn filtterialgoritmiä käytetään signaalin kohinan vähentämiseen samalla säilyttäen alkuperäisen datan muoto- ja aikapiirteet

Filtteri lisätään seuraaviin tietoihin:

- Roll, pitch ja yaw - suuntakulmat
- Mitattu nopeus
- Samansuuntaiset skannatut datat

2.4.3. Skannatun datan korjaus

Ohjelma käyttää vektoria tallentaakseen jokaisen skannatun datapisteen koordinaatit. Jokaisella kerralla kun uusi tietallennedokumentti ladataan rekonstruktointia ja visualisointia varten, tämä vektori alustetaan uudelleen.

Ennen jokaisen datapisteen piirtämistä 3D – avaruuteen, niiden koordinaatit täytyy laskea ottaen huomioon suuntakulmien arvot, sillä ne vaikuttavat sensorin paikkaan suhteessa skannattuun pintaan. Näin visualisoinninssa sensorin liike voidaan korjata käyttäen geometrian matematiikkaa.

Muuntomatriisit:

Ennen kuin selitetään kuinka pisteet liittyvät 3D – avaruuden koordinaatteihin, muuntomatriisien matemaattiikka esitellään täten.

Pyörimismatriisit x-, y- ja z-akselien ympäri kiertokulmilla alpha, beta ja gamma 3D – avaruudessa esitellään Figure 2:ssa.

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\ \alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_y(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_z(\gamma) = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 & 0 \\ \sin\gamma & \cos\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Figure 3 Pyörimismatriisit

Siirtomuunto vektorilla v 3D – avaruudessa tapahtuu käyttämällä muuntomatriisia, joka on esitetty Figure 3:ssa.

$$T_V = \begin{bmatrix} 1 & 0 & 0 & v_x \\ 0 & 1 & 0 & v_y \\ 0 & 0 & 1 & v_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Figure 4 Siirtomatriisi

Monien muuntojen ketjun lisäämiseksi, kaikki erilliset muuntomatriisit kerrotaan käänteisessä järjestyksessä.

Ensin se matriisi joka edustaa alkuperäisiä koordinaatteja skannerin jokaisessa datapaketissa x-y – tasossa, sensori ollen tason luontipiste. Sitten siirto- ja kiertomatriisit lisätään jotta löydetään sopiva muuntomatriisi skannatun datan ja 3D – avaruuden koordinaattien muuntoon. Lopuksi nämä kaksi matriisia kerrotaan jotta voidaan lisätä muunto alkuperäisiin koordinaatteihin. Tapa lisätä tämä muunto skannattuun dataan esitellään tarkemmin seuraavissa osioissa.

Pistekoordinaatit skanneri luontipisteessä

Jokaisen x-y – tason pisteen koordinaatit lasketaan seuraavilla kaavoilla:

$$x_i = range_i * \cos(alpha_i)$$
$$y_i = range_i * \sin(alpha_i)$$

$$Missä: alpha_i = alkukulma + kulmaresoluutio * i$$

$$0 \leq i < pisteiden\ määrä$$

Figure 4 esittelee yhden skannatun datapaketin, jonka keskellä on suorakulmainen laatikko tasaisella pinnalla, ja kun skanneri on hieman kallistunut sivulle.

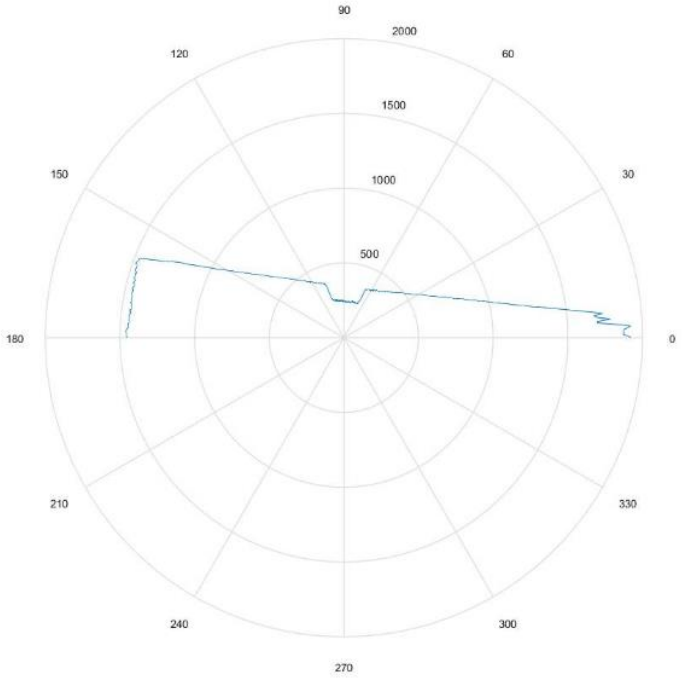

Figure 5 Skannattu datapaketti x – y - tasossa

Jokaisen pisteen koordinaatit asetetaan matriisiin esittämään siirtomuuntoa skannerista nähden.

SICK skannerin muunto 3D – avaruuden koordinaateissa:

Kun ajoneuvo, ja täten myös sensori, on vaakasuorassa, skanneri sijaitsee avaruudessa Figure 5:n esittämällä tavalla.

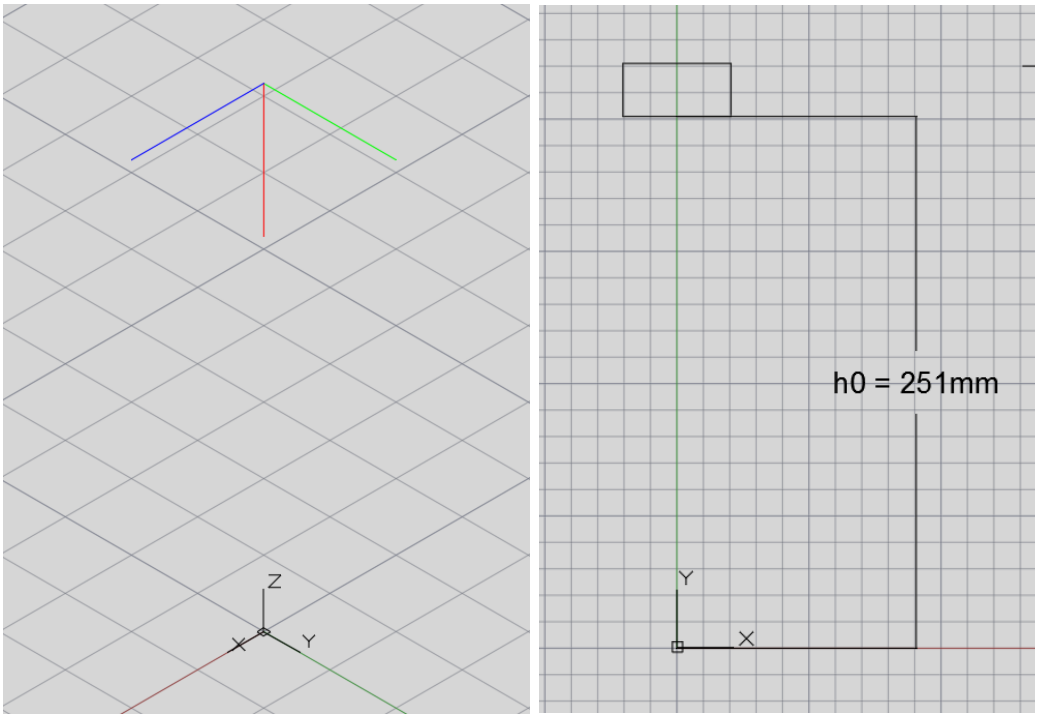

Figure 6 Skannerin sijainti vaakasuorassa

Kun auton kaltevuus roll – suuntakulmaan nähden muuttuu, se johtaa roll – kulman muuttumiseen joko positiiviseksi tai negatiiviseksi, riippuen kallistumisen suunnasta. Kun auton kaltevuus pitch – suuntakulmaan nähden muuttuu, kiertokulma on aina takapyörien akseli, jos auto kulkee suunniteltuun menosuuntaan. Jos tapahtuu roll ja pitch - kaltevuutta samaan aikaan, kiertopiste määräytyy tilanteen mukaan. Roll-, ja pitch – suuntakulmien kaltevuutta esitellään Figure 6:ssa ja Figure 7:ssä.

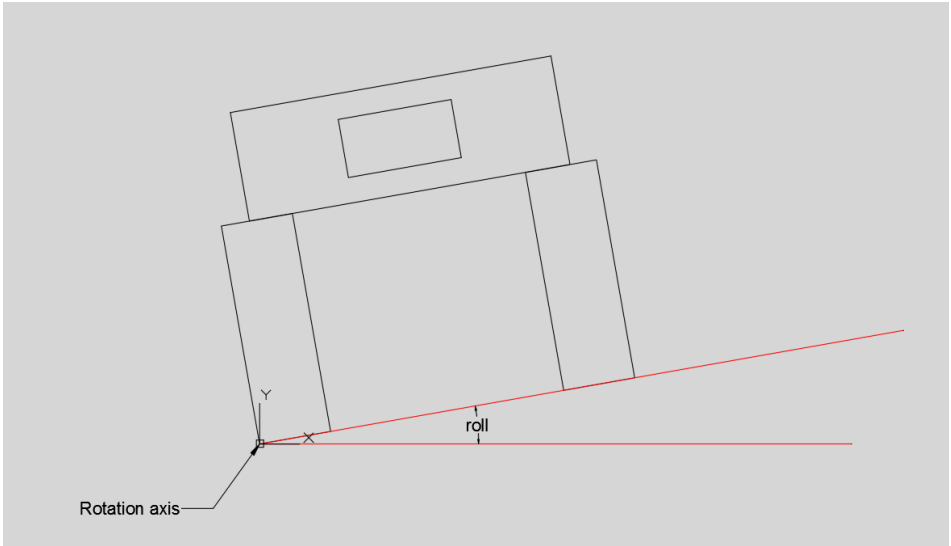

Figure 7 Roll – suuntakulman kierto

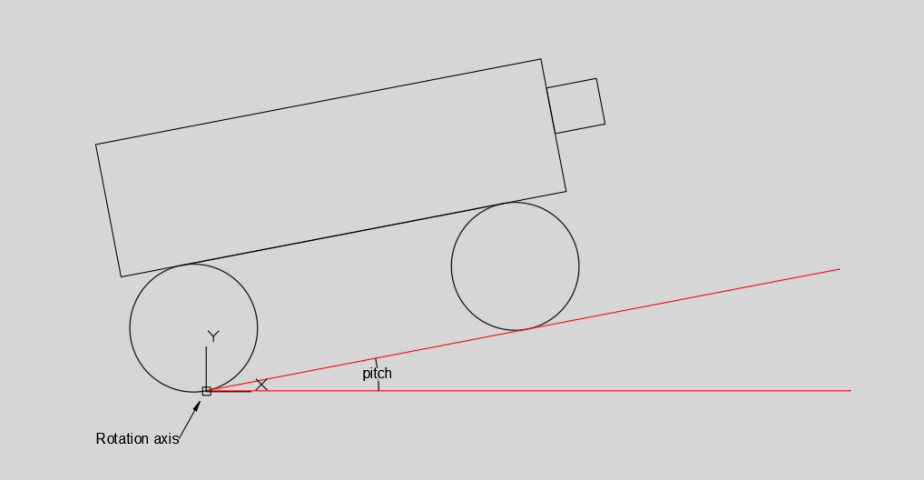

Figure 8 Pitch – suuntakulman kierto

Otetaan esimerkiksi tilanne, jossa molemmat roll-, ja – pitch – kulmat ovat positiivisia, muunto-operaatiot ovat seuraavat:

- Ensin oletetaan, että skanneri on 3D – avaruuden luontipisteessä
- Pyöritetään skanneria x – akselin ympäri 90 astetta, jotta saadaan skannerit suunta kohti maata
- Koska roll – kulma on positiivinen, oikea takapyörä valitaan kiertopisteeksi. Siten, skanneri siirretään vektorin $\boldsymbol{v}$ mukaan joka osoittaa vektorin valitusta renkaasta skanneriin
- Matriisi joka esittää kiertoa y – akselin ympäri, käyttäen roll – arvoa kiertokulmana, lisätään skanneriin. Tämän jälkeen kiertomatriisi x – akselin ympäri käyttäen pitch – arvoa kiertokulmana lisätään.
- Kiertopiste (kohta jossa rengas osuu maahan) siirretään takaisin sen alkuperäiseen paikkaan käyttäen vektoria $-\boldsymbol{v}$
- Lopuksi skanneri siirretään vakiokorkeudelle h_0 siirto-operaatiolla, joka esittää skannerin oikeaa asennuspaikkaa ajoneuvossa.
- Jokaisessa 2D – skannissa ajoneuvo sijaitsee eri pisteessä skannattua matkaa. Tämän vuoksi skannerin sijainti 3D – avaruudessa täytyy muuttaa jokaiseen 2D – skanniin tietyn määrän verran riippuen ajoneuvon nopeudesta. Tämä muutos on ajoneuvon nopeus kerrottuna 1/15 sillä tämä on SICK TiM561 - sensorin näytteenottotaajuus.

Näiden operaatioiden jälkeen skannerin muunto 3D – avaruuden koordinaatteihin samalla lisäten roll -, ja pitch – kulmien korjaukset alkuperäisistä on valmis. Matriisi joka esittää näitä kaikkia muutoksia voidaan esittää kertomalla kaikki erilliset muuntomatriisit käänteisessä järjestyksessä.

Pisteiden muunto 3D – avaruuden koordinaateissa:

Kertomalla kaksi edellälaskettua matriisia, jokaisen datapisteen koordinaatit 3D – avaruudessa saadaan selville. Näitä käytetään skannauksen uudelleenrakentamisessa eli visualisoinnissa.

2.4.4. Visualisointi

Koordinaattivektorin lisäksi SGApp käyttää vektoria esittämään jokaisen pisteen väriä suhteessa pisteen korkeusarvoon skanneriin nähden. Värikaavio vaihtelee sinisestä vihreän kautta punaiseen, jossa normaaliin sensorin etäisyyteen nähden 60 millimetriä kauempana (kuoppa) oleva pinta näkyy täysin sinisenä, normaalilla etäisyydessä täysin vihreänä ja normaalia pistettä 60 millimetriä lähempänä (töyssy) oleva pinta näkyy täysin punaisena.

Visualisoinnissa käytettävää vakionopeutta voi säätää valikosta Settings → Velocity. Datapisteet ja niiden koordinaatit lasketaan uudestaan tälle arvolle ja piirretään uudelleen saman tien. Nopeus vaikuttaa jokaisen 2D – skannin etäisyyteen toisistaan, sillä auto liikkuu skannien välissä aina tietyn matkan.

Figure 7 esittelee esimerkin tallennetun istunnon visualisaatiosta

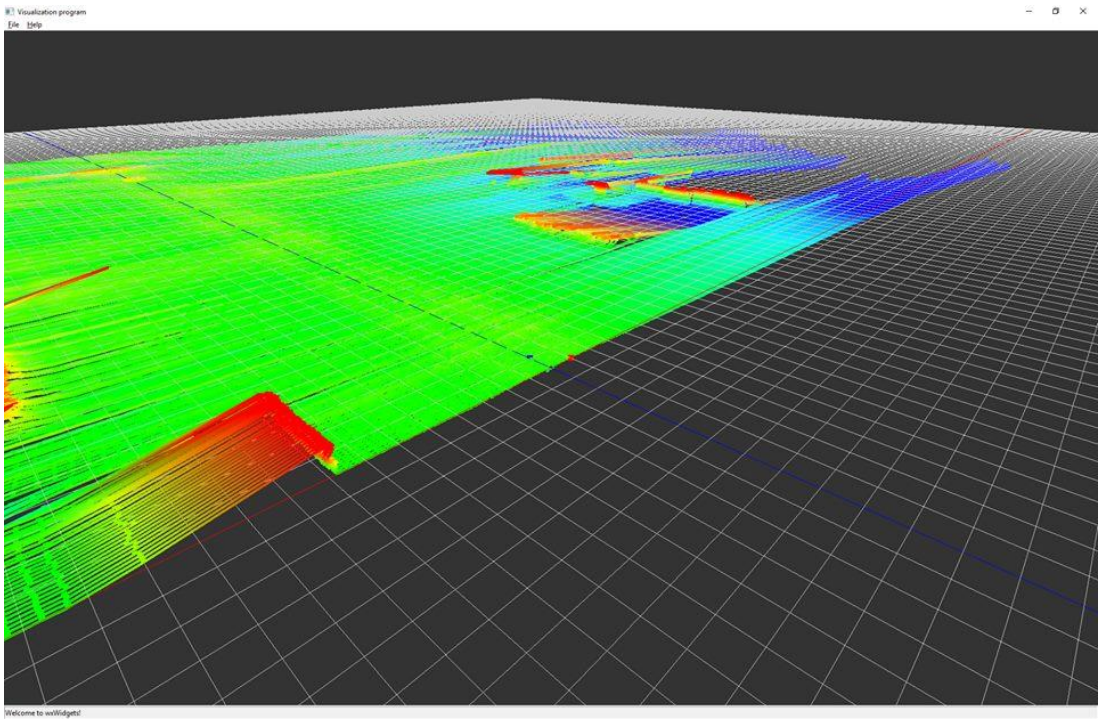

Figure 9 Visualisointiohjelman ruutu

3 Nykyiset rajoitteet ja parannusmahdollisuudet

Ajoneuvon fyysiset ja mekaaniset rajoitteet tulivat vastaan melko nopeasti, tosin lähes kaikkiin niistä löytyy ratkaisu ja hankkimalla parempia/uusia komponentteja näistä rajoitteista voitaisiin päästä eroon. Tähän projektiin aikarajoitteiden vuoksi ei ollut mahdollista hankkia parhaita osia, mutta kaikki käytetyt osat kuitenkin toimivat ja mahdollistivat idean toteuttamisen hyvin.

Mekaanisia tekijöitä, joita voitaisiin parannella:

- Käyttää renkaille tarkoitettuja moottoreita korkeammilla pyörimisnopeuksilla
- Käyttää neljää kaupallista korkean resoluution enkooderia kahden käsintehdyn sijaan
- Väljempi, kehittyneempi jousitusjärjestelmä ja ilmatäytetyt renkaat tärinän vähentämiseksi
- Käyttää kevyempiä Litium-Polymeeri (LiPo) akkuja

Näiden uudistusten toteutus olisi valmiin tuotteen hinnan kannalta lähes merkityksetön verrattuna niiden tuomiin etuihin. Parannus tehokkuuteen jo itsellään olisi hyvä tuotteen käyttöä ajatellen, mutta uudistamalla näitä komponentteja voitaisiin lisätä reaaliaikainen ohjausjärjestelmä, monitoroida ajoneuvoa tarkemmin, tasata ajokokemusta ja pidentää ajoneuvon käyttösädettä.

Tällä hetkellä enkoodereiden laadusta johtuen visualisoinnissa ei voida käyttää mitattua nopeutta, vaikka silmämääräisesti se onkin lähes vakio. Sen sijaan käytettäessä ”synteettistä” nopeusarvoa, visualisointiohjelma antaa oikeita tuloksia, joka tarkoittaa sitä, että vika ei ole nopeuden epätasaisuudessa, vaan enkooderien mitatuissa arvoissa.

Ohjelmointipuolella olisi mahdollista tehdä seuraavia uudistuksia:

- Konenäköjärjestelmä, tarkempaa ympäristöllistä skannausta ja esinetunnistusta varten
- Käyttöliittymä etäohjaukselle ja suora syöte ajoneuvon sensoreilta
- GPS – systeemi tarkkaa paikannusta varten

Näiden uudistusten avulla ajoneuvo voitaisiin asettaa toimimaan autonomisesti tai ainakin autonomisesti. On kuitenkin suositeltavaa tarkastaa skannattava ympäristö vaaratekijöiden varalta ennen automaattisen käytön aloittamista.

Konenäköjärjestelmän avulla olisi mahdollista toteuttaa oppiva järjestelmä, johon valmiiksi syötetyt mallidatat auttaisivat tienpinnan kunnon määrityksessä.

4 Innovaatio - Loppupäätelmä

Automaattinen tienmittausajoneuvo toimii hämmästyttävän hyvin jopa yhdellä SICK TiM561 – 2D laserskannerilla, mutta jos enemmän skanneriresursseja, esimerkiksi toinen TiM561 – skanneri, olisi saatavilla, voitaisiin ajoneuvosta tehdä huomattavasti tarkempi.

Tienmittausapplikaation lisäksi kyseisellä systeemillä on mahdollista skannata ja rekonstruoida esimerkiksi sisätiloja. Tiimi päätyi testaamaan tätä yllättäen, kun ulkona olosuhteet muuttuivat yllättäen liian talvisiksi järkevää mittausta varten. Ajoneuvo suoriutui sisätiloissa mittauksista erittäin hyvin ja jos ajoneuvoon lisättäisiin konenäköjärjestelmä tai edes ohjausjärjestelmä, olisi mahdollista kartoittaa skannaamalla hyvin monimutkaisiakin rakennuksia. Tälle applikaatiolle käyttöä olisi esimerkiksi rakennusten kunnon tarkastuksessa ja automaattisessa 3D – mallinnuksessa. Tulevaisuudessa näitä palveluita tullaan arvostamaan, kun IoT ja datatieteet yleistyvät globaalissa maailmassa.

Loppupäätelmänä tiimi voi ylpeänä kertoa, että Automaattinen Tienmittausajoneuvo suoriutui kaikista alussa sille asetetuista tavoitteista, ja sen kehitystyötä voidaan jatkaa vielä pidemmälle paremmilla resursseilla. SICK TiM561 – laserskanneri osoittautui erittäin kykeneväksi tien- ja sisätilojen mittaajaksi, päästen ± 15 mm tarkkuuteen testeissä, joka on tarpeeksi kyseisiin käyttötarkoituksiin.

Viittaukset

Abyarjoo F., Barreto A., Cofino J. & Ortega F. (2012). Implementing a Sensor Fusion Algorithm for 3D Orientation Detection with Inertial/Magenetic Sensors. CISSE 2012.

Finnish Transport Agency (2015). Finnish Road Statistics. Lappeenranta, Finland

Finnish Transport Agency. Finnish Road network. Retrieved November 29, 2016 from http://www.liikennevirasto.fi/web/en/road-network#.WD3rA_l95DA

Gautam, S (2014). Machine Design and Vision Based Navigation. HAMK University of Applied Sciences, Finland.

IMU Data Fusing: Complementary, Kalman, and Mahony Filter. Retrieved November 29, 2016, from http://www.olliw.eu/2013/imu-data-fusing/

Madgwick, S. O., Harrison, A. J., & Vaidyanathan, R. (2011). Estimation of IMU and MARG orientation using a gradient descent algorithm. 2011 IEEE International Conference on Rehabilitation Robotics. doi:10.1109/icorr.2011.5975346

Mahony, R., Hamel, T., & Pflimlin, J. (2005). Complementary filter design on the special orthogonal group SO(3). Proceedings of the 44th IEEE Conference on Decision and Control. doi:10.1109/cdc.2005.1582367

NASA (2015). Aircraft Rotations Body Axes. Retrieved November 29, 2016, from https://www.grc.nasa.gov/www/k-12/airplane/rotations.html

Liitteet

Kaikki projektiin liittyvät ohjelmat ja lähdekoodit ja muu dokumentaatio ovat samassa kansiossa tämän raportin kanssa, tärkeimpinä:

- Ajoneuvon ohjaus – Arduino – ohjelma
- SGPiApp – Raspberry Pi – ohjelma
- SGApp – Visualisointi – ohjelma
- Ajoneuvon toimintaa ja ominaisuuksia esittelevät esitteet