TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

Krister Looga

Vektorkaardi genereerimine isejuhtivale autole trajektooride põhjal

Bakalaureusetöö (9 EAP)

Juhendajad
Tambet Matiisen, MSc
Edgar Sepp, MSc

Tartu 2021

Vektorkaardi genereerimine isejuhtivale autole trajektooride põhjal

Lühikokkuvõte:

Isejuhtivad sõidukid kasutavad autonoomse juhtimise võimaldamiseks detailseid kaarte, mida kutsutakse täppiskaartideks. Täppiskaardid sisaldavad palju informatsiooni ja nende joonestamine nõuab palju aega, ning ei ole ühte kindlat levinud meetodit nende loomiseks.

Üks tunnus Tartu Ülikooli isejuhtivate sõidukite labori täppiskaardil on sõidurada (ingl. k. *lane*). Sõiduraja oluliseks osaks on sõidujoon, mida mööda isejuhtiv sõiduk liigub. Sõidujoon lihtsustab teekonna ja kiiruse planeerimist isejuhtiva auto jaoks.

Bakalaureusetöös loodi lahendus Tartu Ülikooli isejuhtivate sõidukite laborile, millega saab varasemate sõitude abil genereerida andmeid sõidujoone loomiseks. Loodud andmed vähendavad käsitsi tehtava töö mahtu ja kirjeldavad paremini tegelikku liikluskorraldust.

Bakalaureusetöös tutvustatakse täppiskaartide, trajektooride ja sõidujoone olemust. Lisaks antakse ülevaade loodud lahendusest. Viimaks tutvustatakse saadud tulemust ja mõõdetakse selle täpsust varasemalt kasutuses olnud sõidujoonega. Seejärel tuuakse välja lahenduse puudused ning võimalused edasiseks arendustööks.

Võtmesõnad: Vektorkaart, täppiskaart, trajektoorid, sõidujoon

CERCS: P170 (Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine)

Generating Vector Map Data for Autonomous Driving

Self-driving cars use detail-rich maps called High Definition maps (HD-map) to enable self-driving. These maps contain much information, but there is no standardized way of making them. Furthermore, creating such maps by hand takes much time.

One of the properties on the HD-map used by the Autonomous Driving Lab at the University of Tartu describes a trajectory that a self-driving vehicle uses as virtual rails.

For the bachelor's thesis, a program was created to aggregate data collected during manual drives in Tartu. That data is used to create a new reference trajectory for the HD-map. The reason for generating this data is to decrease the amount of manual labor required to create the HD-map. Furthermore, since data is generated based on actual driving data, it should better reflect vehicles' speed and position on the road.

The first two chapters talk about HD-maps and trajectories and how they can be created. After that, an overview is given of the solution created during the thesis. The results are

introduced, and its accuracy is compared to the manually created trajectory. Finally, the limitations and possibilities for future development are proposed.

Keywords: Vectormap, HD-maps, trajectories, lane

CERCS: P170 (Computer science, numerical analysis, systems, control)

Sisukord

Sissejuhatus

Isejuhtivad sõidukid kasutavad autonoomse juhtimise võimaldamiseks detailseid kaarte ehk täppiskaarte. Täppiskaardid sisaldavad erinevat informatsiooni keskkonna ja liikluskorralduse kohta. Üks oluline osa täppiskaardist on sõidujoon, mida mööda isejuhtiv sõiduk liigub.

Sõidujoone käsitsi joonestamine on aeganõudev ja täpsus sõltub tihti aluskaardi infost. Lisaks puudub standardiseeritud lähenemine täppiskaardi osade loomiseks. Bakalaureusetöö eesmärk on uurida võimalusi ja luua skript, millega Tartu isejuhtivate sõidukite labori täppiskaardile genereerida automaatselt sõidujoone andmed.

Loodav lahendus kasutab sisendina inimjuhi sõitude ajal kogutud andmeid. Kasutades mitme sõidu kiiruse ja trajektooride infot, genereeritakse keskmistamise abil automaatselt sõidujoone andmed. Töö eesmärk on vähendada manuaalsele joonestamisele kuluvat aega ja aidata luua täpsemat sõidujoont.

Esimene peatükk tutvustab taustainfot. Lühidalt kirjeldatakse isejuhtivaid sõidukeid, täppiskaarte ja nende loomist ning Tartu Ülikooli isejuhtivate sõidukite labori täppiskaarti. Teises peatükis tutvustatakse trajektoori ja sõidujoone definitsiooni, ning Ida-Soome Ülikooli segmentide keskmistamise võistlust. Lisaks räägib autor, mis andmeid kasutatakse ja defineeritakse töö eesmärgid. Kolmandas peatükis tutvustatakse loodud lahendust ja tehnoloogilisi valikuid. Kirjeldatakse, kuidas andmeid kasutatakse ja mis protsessid keskmistamise võimaldamiseks toimuvad, et saavutada optimaalne tulemus. Neljandas peatükis tutvustatakse tulemust ja võrreldakse, kui sarnane on automaatselt genereeritud sõidujoon käsitsi tehtud sõidujoonele. Lisaks võrreldakse kui sarnane on inimjuhi sõidu trajektoor ja autonoomselt sõidetud trajektoor käsitsi joonestatud sõidujoonele. Viimaks tuuakse välja lahenduse puudused ja võimalused edasiseks arendustööks.

1. Taustainfo

Tänapäevaste isejuhtivate sõidukite juures mängivad olulist rolli täppiskaardid. Sellise detailse kaardi loomine ei ole standardiseeritud protsess [1, 2]. Järgnevates peatükkides tutvustab autor mitme ettevõtte teadustööd ja paari levinumat lähenemist täppiskaartide loomiseks.

1.1 Isejuhtivad sõidukid ja täppiskaardid

Isejuhtivad sõidukid on pikalt eksisteerinud ulmefantaasia maailmas ja on olnud kättesaadamatud. Viimase paari aasta jooksul on turule jõudnud sõidukid, mis suudavad teatud tasemini end ise juhtida [3].

SAE (ingl. k. *Society of Automotive Engineers*) defineerib isejuhtivate sõidukite viis taset (vt Joonis 1) [4]. Isejuhtivaid sõidukeid tootev Tesla on saavutanud teise autonoomsuse taseme, pakkudes juhiabi süsteeme [5]. Google'i isejuhtivate sõidukite uurimiskeskus ja taksoteenust pakkuv Waymo on praeguseks saavutanud neljanda taseme, võimaldades juhist täielikult loobuda [6, 7]. Erinevus neljanda ja viienda taseme vahel on see, et viienda taseme süsteem saab hakkama kõigis liiklusoludes, aga neljas tase ainult mingis kindlas alas ja ilmastikuoludes [4, 8]. Viiendat taset teadaolevalt keegi veel saavutanud ei ole [9].

SAE tase	Nimetus	Kirjeldus	Roolimise, kiirendamise-pidurdamise teostus	Juhtimise keskkonna moni-tooring	Dünaami-lise juhtimis-ülesande tagavara-plaan
Inimjuht monitoorib juhtimiskeskkonda					
0	Automat-sioon puudub	*Inimjuht* teostab igal ajal ja aspektis *dünaamilisi juhtimisülesandeid*, isegi kui tõhustatud hoiatus või sekkumis süsteemiga	Inimjuht	Inimjuht	Inimjuht
1	Juhiabi	Juhiabi süsteem teostab *juhtimisrežiimi spetsiifikale vastavaid* roolimise või kiirendamise-pidurdamise ülesandeid kasutades selleks juhtimise keskkonna informatsiooni eeldusega, et *inimjuht* teostab *dünaamilised juhtimisülesanded* kõigis teistes aspektides	Inimjuht ja süsteem	Inimjuht	Inimjuht
2	Osaline automat-sioon	Üks või mitu juhiabi süsteemi teostab *juhtimisrežiimi spetsiifikale vastavaid* roolimise, ja/või kiirendamise.pidurdamise ülesandeid kasutades selleks juhtimise keskkonna informatsiooni ja eeldusega, et *inimjuht* teostab *dünaamilised juhtimisülesanded* kõigis teistes aspektides	Süsteem	Inimjuht	Inimjuht
Automaatne juhtimissüsteem („süsteem“) monitoorib juhtimiskeskkonda					
3	Tingimuslik automat-sioon	*Automaatne juhtimissüsteem* teostab igas aspektis *dünaamilisi juhtimisülesandeid* vastavalt juhtimisrežiimi spetsiifikale, eeldusega, et *inimjuht* reageerib kohaselt *taotlusele sekkuda*	Süsteem	Süsteem	Inimjuht
4	Kõrge automat-sioon	*Automaatne juhtimissüsteem* teostab igas aspektis *dünaamilisi juhtimisülesandeid* vastavalt juhtimisrežiimi spetsiifikale, ka olukorras, kus *inimjuht* ei reageeri adekvaatselt *taotlusele sekkuda*	Süsteem	Süsteem	Süsteem
5	Täis-automat-sioon	*Automaatne juhtimissüsteem* teostab igal ajal ja igas aspektis *dünaamilisi juhtimisülesandeid* kõigis tee ja keskkonna tingimustes, milles saaks neid teostada *inimjuht*	Süsteem	Süsteem	Süsteem

Joonis 1. Autonoomsuse viis taset ja nende tingimused [10]

Isejuhtivad sõidukid on realiseeritud keerulises riistvara ja tarkvara koostöös. Kasutusel on palju võimekaid sensoreid ümbruse tuvastamiseks ja sõiduki keskkonnas positsioneerimiseks. Autonoomselt sõidukilt võib leida seadmeid, nagu radar, GPS süsteemid, LIDAR (ingl k. *Light Detection And Ranging)* ning tarkvara, mis seda infot suudab töödelda ja sõidukit juhtida. Lisaks toetab tarkvara otsustusprotsesse täppiskaart, mille abil on võimalik andmelünki täita ja informatsiooni tõlgendada [11, 12].

Täppiskaartidel on mitu rolli autonoomse liikumise võimaldamisel ja kitsaskohtade lahendamisel. Teades sõiduki täpset asukohta on võimalik kaardi abil tuvastada ümbritsevat informatsiooni ja teha kiiremini otsuseid [13]. Lisaks on võimalik sõidukil arvestada liikluskorralduse ja oludega kaugemal, kui seda sensorid võimaldavad. Kasutades täppiskaarti kui virtuaalset sensorit, saab isejuhtiv sõiduk end paremini keskkonnas positsioneerida [14] ja liigelda turvalisemalt [11, 12].

Võrreldes tavaliste kaartidega, mille täpsus on meetrites, on täppiskaartide täpsus sentimeetrites ja nad sisaldavad endas vajalikku infot autonoomse juhtimise võimaldamiseks. Näiteks Lyfti täppiskaart koosneb viiest erinevast kihist, kus esimene kiht sarnaneb tavalisele kaardile ja ülejäänud kihid sisaldavad andmeid 3D ümbrusest, liiklusmärkidest, teekatte märkidest ja reaalaja informatsioonist [1, 2, 15].

1.2 Täppiskaartide loomine

Täppiskaartide loomise jaoks puudub ühtne strateegia, [2] ning igal selliste kaartide loomisega tegeleval asutusel on enda lähenemine [16]. Erinevused saavad alguse juba sellest, et iga isejuhtimist võimaldava süsteemi eesmärgid on erinevad [1].

Üks võimalus täppiskaardi loomiseks on võtta aluseks ortofoto ja selle abil joonestada täppiskaart. See eeldab, et kaardilt on võimalik tuvastada kogu vajalik info, nagu teekattemärgistused ja liiklusmärgid. Tavalise kaardi põhjal täppiskaardi loomisel sõltub palju aluskaardi kvaliteedist ja sisust. Varjud ortofotol võivad varjata valgusfoore, teekattemärgistus võib olla kulunud või kõrghoone varjata teed [17].

Valesti märgistamise korral võib sõiduk liikluses valesti käituda. Ortofoto kasutamisel tuleb arvestada, et neid kaarte uuendatakse tavaliselt perioodiliselt ja pikkade vahemike tagant [18]. See võib tähendada, et kaardil olev info on juba aegunud, kuna liikluskorraldus on vahepeal muutunud.

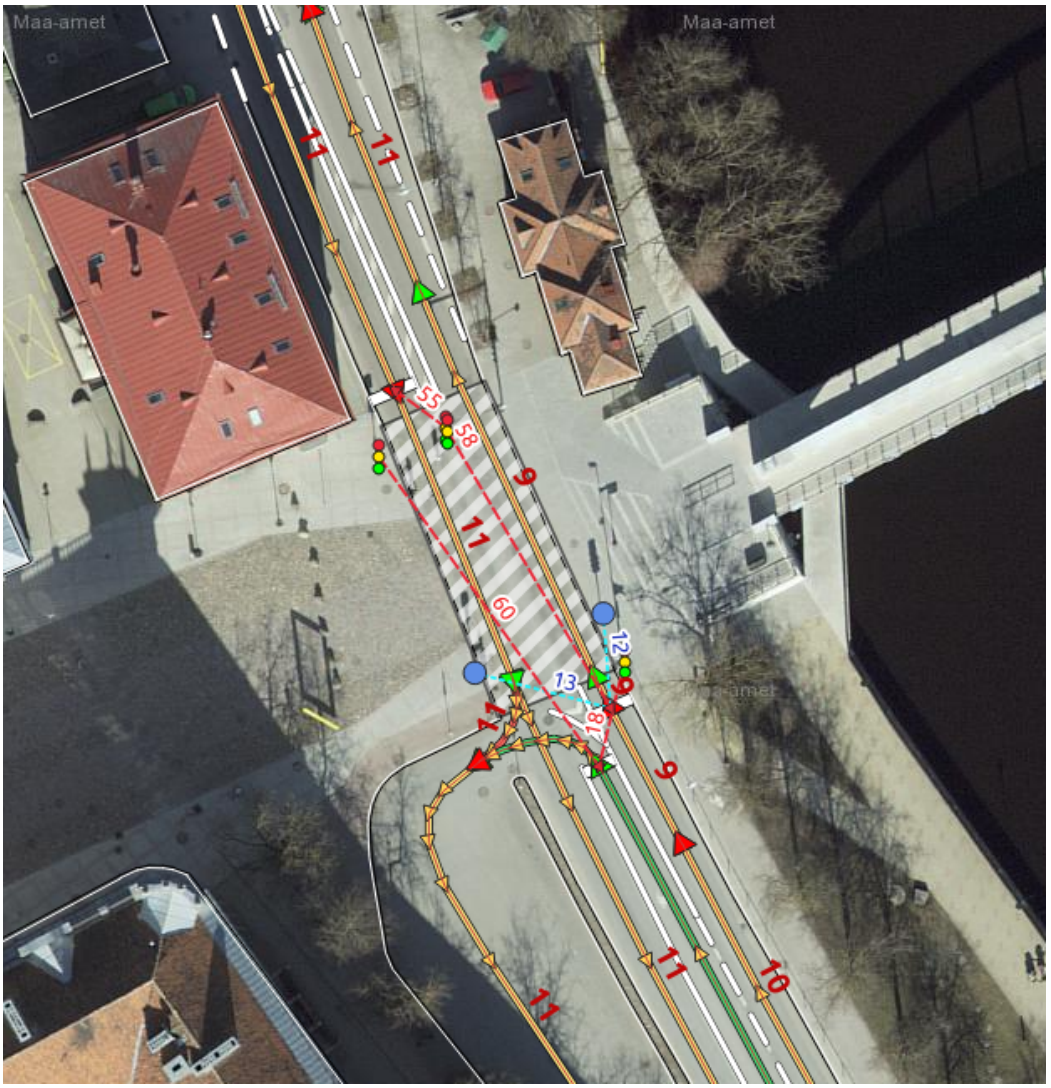
Teine võimalus täppiskaardi loomiseks on teha seda automaatselt. Selle eelduseks on suur hulk andmeid, mille abil saab kas algoritmiliselt või masinõppega täppiskaardi andmeid genereerida [19]. Taksoteenust pakkuv Lyft on andmete kogumise eesmärgil osadele oma taksodele paigutanud erinevad sensorid. Näiteks GPS-i abil on võimalik ära määrata sõidutee keskkoht ja auto kiirus. Kaamerate, radari ja LIDAR-i abil on võimalik kaardistada teekattemärgistused, äärekivid, teised liiklejad ja liiklusmärgid. Igapäevaselt sõitvad taksod tagavad, et olemas on suur kogus informatsiooni ja selle põhjal tehtud täppiskaart on täpne ja ajakohane [4,6].

1.3 Isejuhtivate sõidukite labor

Tartu isejuhtivate sõidukite labori kasutuses on Lexus RX450h, millel on kaks LIDAR-it ümbruse tuvastamiseks, radar, GNSS süsteem GPS andmete kogumiseks ja mitu kaamerat. Kuna GPS andmete täpsus on varieeruv ja võib ulatuda mitme meetrini, siis täpsuse parandamiseks kasutatakse Maanteeameti RTK (ingl. k. *Real Time Kinematic*) teenust, et GPS andmed viia sentimeetri täpsuseni. Isejuhtimise võimaldamiseks kasutatakse mitut vabavaralist tarkvara. Operatsioonisüsteemina on autos kasutusel Ubuntu ja ROS (ingl. k. *Robotic Operating System*), isejuhtimist võimaldav tarkvara Autoware.ai ja auto juhtimist võimaldav moodul Pacmod [17, 20] .

Isejuhtivate sõidukite labori täppiskaardi (vt Joonis 2) on koostanud Edgar Sepp oma magistritöö "Creating High-Definition Vector Maps for Autonomous Driving" raames [17]. Täppiskaart on joonestatud QGIS tarkvara abil ja skripti abil muudetud Autoware vektorkaardi kujule, et isejuhtiv sõiduk saaks seda kasutada [17].

Oma magistritöös kirjeldab Edgar Sepp täppiskaardi QGIS-is loomise viit sammu. Kõigepealt kogutakse manuaalsete sõitudega andmed sõiduraja joonestamiseks. Kogutud trajektooride ja ortofoto abil lisatakse tunnused kaardile ja QGIS-i abil trajektoorid silutakse ja järeltöödeldakse. Peale seda tulemused valideeritakse ja valmistatakse ette ekspordiks. Peale seda saab täppiskaardi konverteerida vektorkaardiks, mida Autoware'is saab kasutada [17].

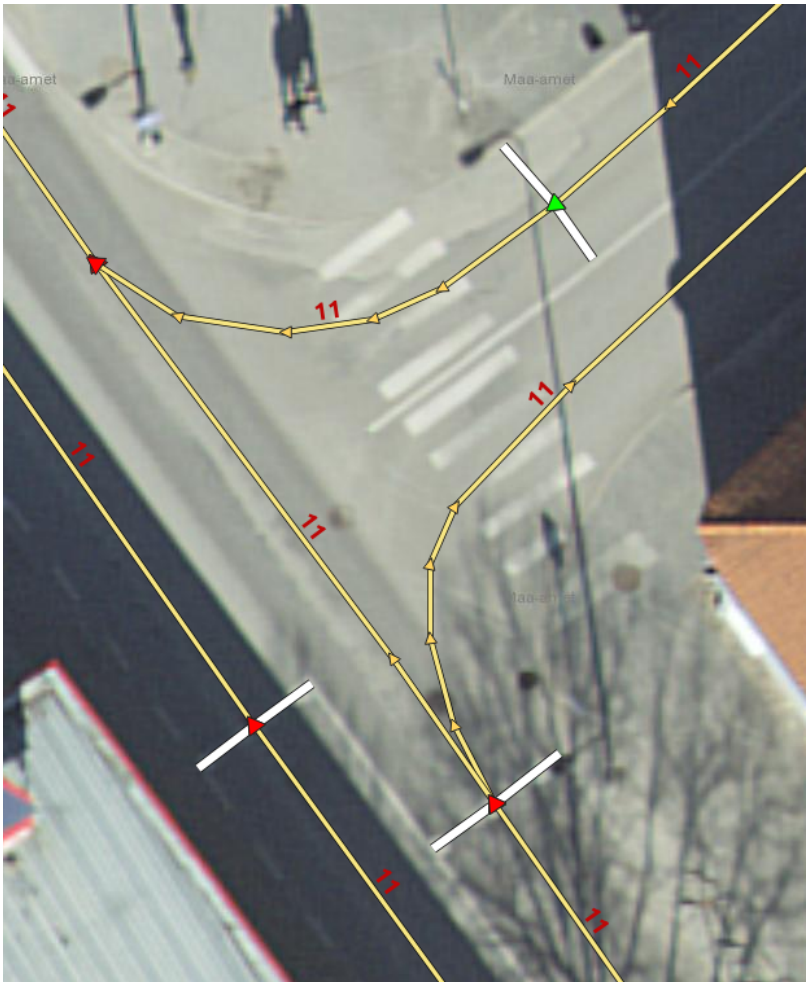


Joonis 2. Tartu Ülikooli isejuhtivate sõidukite labori täppiskaardi näide Tartu raekoja platsi ees

Isejuhtivate sõidukite labori täppiskaardilt leiab järgneva info: valgusfooride asukohad, ülekäigurajad, stoppjooned, teekattemärgistuse, äärekivid, ristmikute alad, liiklusmärgid ja sõidurajad (vt Joonis 2) [17].

2. Trajektoorid

Sõidujoon on joon vektorkaardil, mida isejuhtiv sõiduk saab kasutada autonoomseks juhtimiseks. Sõidujoon kirjeldab keskjoont, ning sõidujoone väiksem osa on sõidulõik, mis on järjest ühendatud teineteisega (vt Joonis 3). Ühe sõidulõigu tunnused on: tee laius vasakule ja paremale, piirkiirus ja soovituslik kiirus, ning lõigu tüüp, mis määrab, kas tegu on sirge või näiteks parem-või vasakpöördega. Selleks, et vältida isejuhtiva sõiduki valet või äkilist sõitu, on oluline sõidujoone täpsus ja kurvide sujuvus.



Joonis 3. Sõidulõikude algust ja lõppu märgivad valged stoppjooned

Lõputöös kasutatud terminitest on oluline eristada sõidujoont, sõidulõiku ja trajektoori. Sõidujoon on keskjoon, mida isejuhtiv sõiduk üritab järgida. Lõputöö käigus genereeritakse automaatselt andmeid sõidujoone, ehk keskjoone tegemiseks. Sõidulõik on väiksem tükk sõidujoonest, mis sisaldab rohkem infot. Trajektoor on auto sõidu ajal kogutud sõitu kirjeldavad andmed. Need võivad olla kogutud inimjuhi sõidu ajal või autonoomse sõidu käigus.

2.1 Varasemad tulemused

Bakalaureusetöö autor kasutas sisendina Ida-Soome Ülikooli professori Pasi Fränti ja järeldoktor-teaduri Radu Mariescu-Istodori korraldatud GPS segmentide keskmistamise võistlust [21]. Võistlus korraldati 2019. aastal ja eesmärgiks oli GPS andmetest segmentide tegemine ja ristmike tuvastamine. Kokku esitati seitsme autori poolt üheksa lahendust [22].

Lahenduste struktuur oli läbivalt sarnane - kõigepealt järjestati punktid ümber, siis segment keskmistati, eemaldati erindid, ehk normist erinevad väärtused [23] ja tehti järeltöötlust. Segmendi keskmistamiseks kasutati kolme erinevat lähenemist [24]:

- regressioonimudel
- kirjeldavate punktide loomine ja mediaani kasutamine
- trajektooride keskmistamine

Kuna sisendandmed on trajektoori lõikes järjestatud ja kogutud GPS andmed piisavalt täpsed, ei esine selle lõputöö raames valminud lahenduses probleeme punktide järjestuse ja ebatäpsete trajektooride eemaldamisega.

Eelmainitud võistluselt kasutas töö autor inspiratsioonina segmentide keskmistamise võistluse osaleja Jiawei Yangi lähenemist kirjeldavate punktide genereerimisele. Yangi algoritmi võtab sisse segmendi, leiab sealt esimese, keskmise ja lõpp-punkti, ning nende kolme punkti abil loob kaks vektorit. Nende kahe vektori vahelist nurka võrreldakse, et saada teada, kas tegu on sirge või kurviga [24, 25].

Sarnaselt Yangi lahendusele loob lõputöö autor kirjeldavad punktid, et tuvastada, kas tegu on kurviga. Küll aga erineb lõputöö raames loodud lahendus ülejäänud lahenduse detailides, kasutades sisendandmete spetsiifilist andmetöötlust ja lähenedes keskmistamisele hoopis teistmoodi.

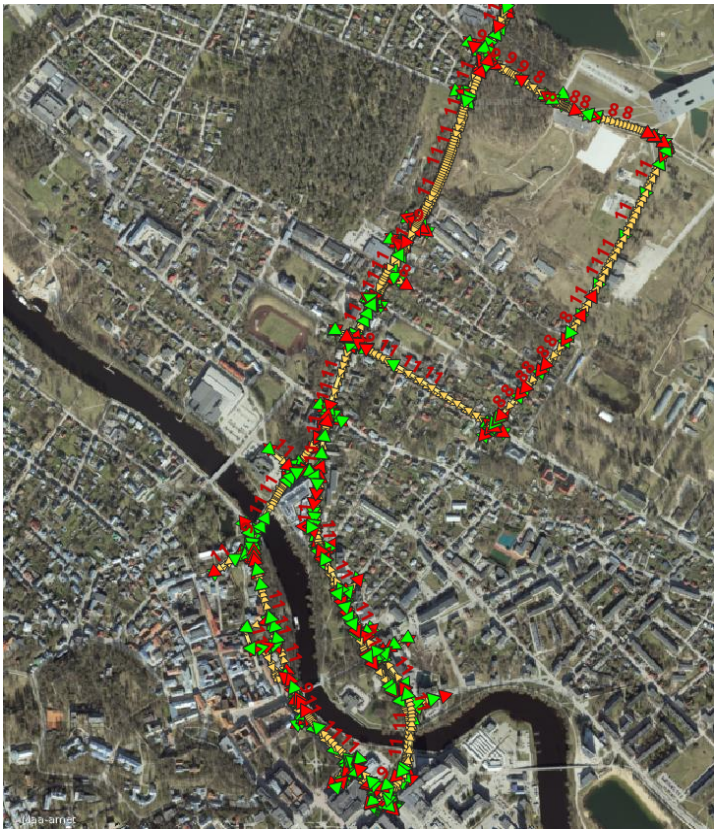
2.2 Sisendandmed

Automaatselt genereeritud sõidujoonte loomiseks on kasutatud varasemalt inimjuhi sõidetud trajektoore. Andmed on CSV (ingl. k *comma separated value*) formaadis ja need on kogutud Tartu Ülikooli isejuhtiva sõiduki sõitude ajal. Automaatselt genereeritud sõidujoone loomiseks on autor kasutanud igas punktis olevat sõiduki hetkekiirust ja positsiooni koordinaate (vt Tabel 1). Koordinaadid on Eesti koordinaatsüsteemis L-EST97.

Tabel 1. Näidis automaatse genereerimise rakenduse sisendandmetest

index	x	y	speed
0	659424.1092252033	6474896.358615189	0.0018595447565814675
1	659424.109254362	6474896.358625153	0.0018595447565814675
2	659424.1093069231	6474896.358633446	0.0018595447565814675
3	659424.1093423603	6474896.358641777	0.0018595447565814675

Lõputöö loomisel käsitletud andmed ja tulemus kirjeldavad Tartu ülikooli Delta õppehoonest alustades ringi ümber Emajõe mööda Narva maanteed ja Vabaduse puiesteed. Lisaks ringi Narva maanteest üles Eesti Rahva Muuseumi (edaspidi ERM) juurde ja mööda Roosi ning Jaama tänavat tagasi Delta juurde (vt Joonis 4).



Joonis 4. Näide täppiskaardil kaardistatud alast

Lõputöö raames kasutatakse automaatse sõidujoone loomiseks kaheksa sõidu andmeid, mis on kogutud Tartu Ülikooli isejuhtivate sõidukite labori autoga. Viie trajektoori andmed on Autoware'i abil puhastatud, ning punktid on ühe meetriste intervallidega. Kolme trajektoori andmed on korrastamata ja seal on punktid kogutud väikese intervalliga Andmestikus on üle 86000 punkti ja kuus sõitu kirjeldavad Emajõe ringi ning kaks sõitu ERM-i ringi.

2.3 Sõidujoone automaatne genereerimine

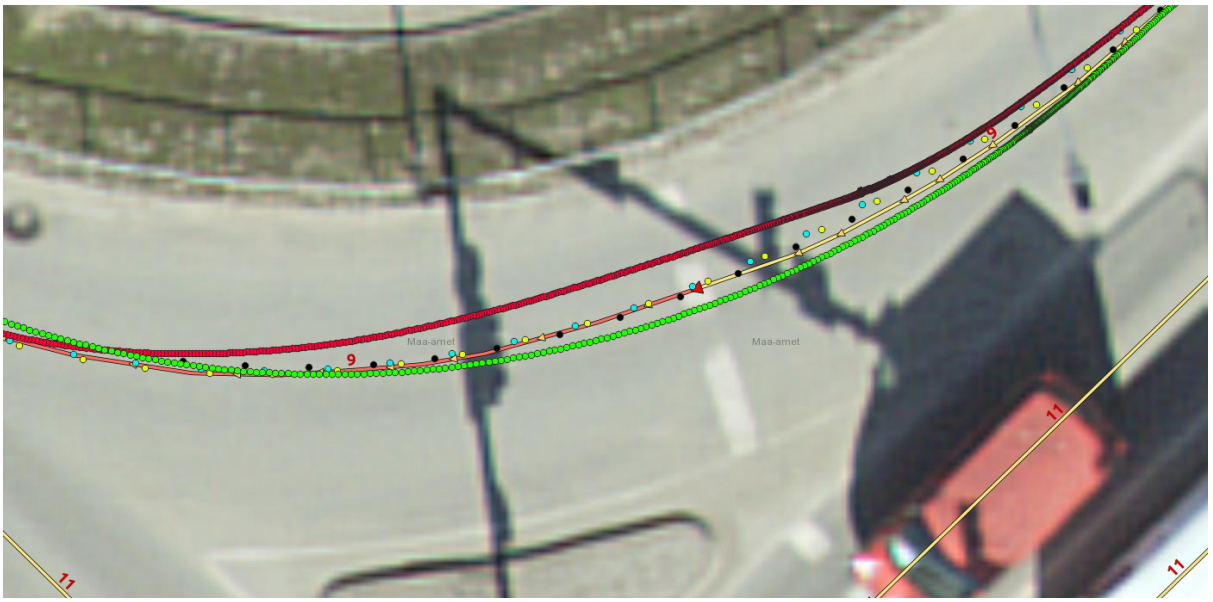
Võrreldes käsitsi joonestamisega, on automaatselt sõidujoone genereerimisel mitu eelist. Esiteks sõidujoone loomiseks kasutatav andmete arv on palju suurem, kui see mida inimene suudaks visuaalselt käsitsi joonestades kasutada. Kui muidu tuleks sõidujoone joonestamisel kasutada ortofotot ja inimjuhi sõidetud trajektoori kirjeldavaid punkte (vt Joonis 5), siis rakenduse genereeritud keskjoon vajab töötamiseks ainult andmeid.

Joonis 5. Ortofoto ja roheliste punktide abil loodud kollane sõidurada

Võib eeldada, et automaatselt genereeritud sõidujoon on täpsem, kuna näiteks iga meetri tagant on võimalik leida varasema saja sõidu põhjal keskmine kiirus ja keskmine asukoht

lõigul. Keskmise kiiruse arvutamine igal lõigul ja täpne positsioneerimine võtaks käsitsi tehes kaua aega.

Varasematele sõitudele (vt Joonis 6) põhinedes leitakse lõigu igas punktis tegelikule liiklusele vastav keskmine kiirus ja keskjoon, mida autojuhid kasutavad. Selle informatsiooni saamine on oluline, kuna käsitsi joonestades eeldatakse, et autod sõidavad tee keskel ja piirkiirusega. Tegelik olukord erineb tihti eeldatavast ja sõidukiirus ning trajektoor kohandatakse liiklusoludele. Näiteks liigeldakse keskjoonest kaugemal täispargitud teeääre ja halva nähtavuse korral, ning alla piirkiiruse sõidetakse koolide ja lasteaedade ümbruses.



Joonis 6. Kollane sõidujoon ja värviliselt kujutatud inimjuhi sõitude trajektoorid

Teine eelis sõidujoone automaatsel genereerimisel on see, et täppiskaarti saab kiiresti muuta. Liikluskorraldus võib muutuda ja sellega kohanemiseks on vaja täppiskaarti uuendada. Kui näiteks päeva jooksul Tartu Ülikooli isejuhtivate sõidukite labori partner Bolt saadaks taksosõitude GPS andmed, oleks võimalik skripti üleöö jooksutada ja genereerida uus automaatne sõidujoon, mida saaks juba järgmisel päeval kasutada.

Edgar Sepa magistritöös [17] selgub, et ühe kilomeetri käsitsi joonestamiseks läheb 5-6 tundi. Tasub arvestada, et selle aja sees on arvestatud lisaks sõidujoonele ka täppiskaardi teiste tunnuste lisamisega.

3. Lahenduse kirjeldus

Automaatse sõidujoone genereerimise lahenduse saab jagada kaheks osaks. Esiteks andmete töötlemine, mille alla käib trajektooride sisselugemine ja vajalike andmestruktuuride loomine. Peale seda keskmistatakse trajektoorid ja tagastatakse uued andmed sõidujoone loomiseks.

Andmete töötlemiseks ja sõidujoone genereerimiseks on autor kasutanud programmeerimiskeelt Python ja selle teeke Numpy ning Pandas [26–28]. Python osutus valituks, kuna autoril oli varasem kogemus selle kasutamisega. Lisaks on sellel hea dokumentatsioon ja teegid andmete töötlemiseks.

Andmetöötluseks programmeerimiskeeles Python on hea kasutada arvutuskeskkonda Jupyter Notebook, millega on võimalik jooksutada koodi erinevaid osi üksteisest eraldi ja säilitada vahetulemused. Rakenduse tükiti jooksutamine võimaldab kokku hoida aega andmete sisselugemise pealt ja saada kiiret tagasisidet arendusele. Lisaks toetab Jupyter Notebook Markdowni süntaksi, võimaldades ühes failis kergesti dokumentatsiooni ja kommentaare lisada [29].

Tulemuse visualiseerimiseks ja valideerimiseks on autor kasutanud vabavaralist geoinfosüsteemi QGIS, mis võimaldab kaarte redigeerida, lisades sinna erinevat informatsiooni ja kihte [30]. Manuaalselt täppiskaardi joonestamiseks on QGIS-i seni kasutanud Tartu Ülikooli isejuhtivate sõidukite labor [17].

3.1 Andmete sisselugemine

Koodi hea tööaeg ja kerge kasutatavus on üks lõputöö eesmärkidest. Lahenduse kasutamiseks piisab, kui anda rakendusele andmekausta asukoht arvutikettal. Lisaks arvestab rakendus andmefailide eripäradega - toetatud on erinevad veerunimed ja väärtuste standardiseerimine. Selle tarbeks on loodud konfiguratsiooni objekt, kuhu saab lisada toetatud veerunimed ja nende oodatavad tulemused. Väärtuse puudumise korral tekitab rakendus erindi.

Andmete sisselugemiseks kasutatakse etteantud andmekausta väärtust. Kasutaja antud kausta asukohast otsitakse üles kõik CSV-failid ja kontrollitakse, kas automaatse sõidujoone genereerimiseks on olemas vajalike andmete veerud. Üldjuhul otsib rakendus veerge

nimedega *E_lest97*, *N_lest97* ja *speed*, kuid konfiguratsioonifailis on võimalik defineerida pseudonüüme näiteks *x*, *y*, *velocity*.

Seejärel kontrollitakse, kas kiirus on meetrit sekundis ja kas koordinaadid on L-EST97 formaadis. Koordinaatide kontrollimiseks vaadatakse, kas sisendandmete koordinaadid jäävad L-EST97 vahemikku. Kiiruse kontrollimiseks on vaja eelnevalt koodis märkida, et selle veerunimega andmeid on vaja teisendada.

Vajadusel tehakse teisendused, et need õigesse vahemikku viia. Kiiruse teisendamiseks viiakse näiteks kilomeetrit tunnis kiirus meetritesse sekundis ja koordinaatide korrastamisel lisatakse L-EST97 offset, ehk x koordinaadile 650000 ja y koordinaadile 6465000 juurde. Korrastatud andmed lisatakse andmete järjendisse, mida kasutatakse sõidujoone genereerimisel ja puustruktuuriga andmestiku loomiseks.

Korrastatud andmetest tehakse Scikit-learn-i lähima naabrite meetodi abil puustruktuuriga andmestik. Kuna järgmistes sammudes on tihti vaja leida ette antud punktile lähimaid punkte, kiirendab sellise struktuuri kasutamine neid samme oluliselt. Selle asemel, et itereerida üle saja tuhande punkti, ja arvutada välja, millised neist on praegu töötlemisel oleva punkti lähedal, on kiirem seda otsingut teha puustruktuuriga andmestikus.

Puustruktuuriga andmestikul on võimalik kasutada otsingu kiirendamiseks kolme erinevat indekseerimise meetodit - jõumeetod (ingl k. *brute force*), Ball Tree ja K-D Tree. Jõumeetod ei sobi suuremate andmemahtude korral ja aeglasem ning ruumimahukam Ball Tree on mõeldud suurema dimensionaalsusega andmete jaoks. Sellest tulenevalt kasutatakse käesoleva lõputöö raames K-D puu meetodit [31].

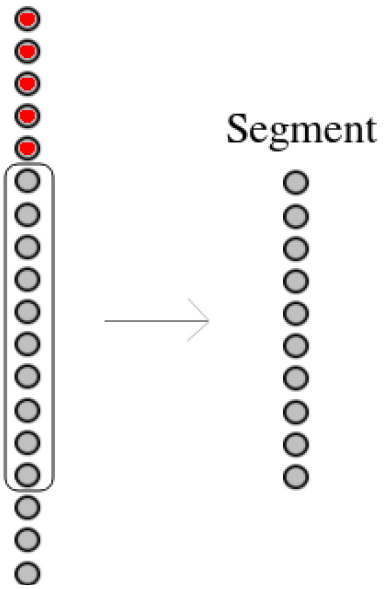
3.2 Segmendi loomine

Peamine eesmärk segmentide loomisel on vähendada punktide arvu ja säilitada seda tehes loodava segmendi täpsus. Järgnevates peatükkides tutvustab autor, kuidas hea tööaja ja täpsuse saavutamiseks on tehtud mitmeid optimeerimissamme.

3.2.1 Segmendi loomine

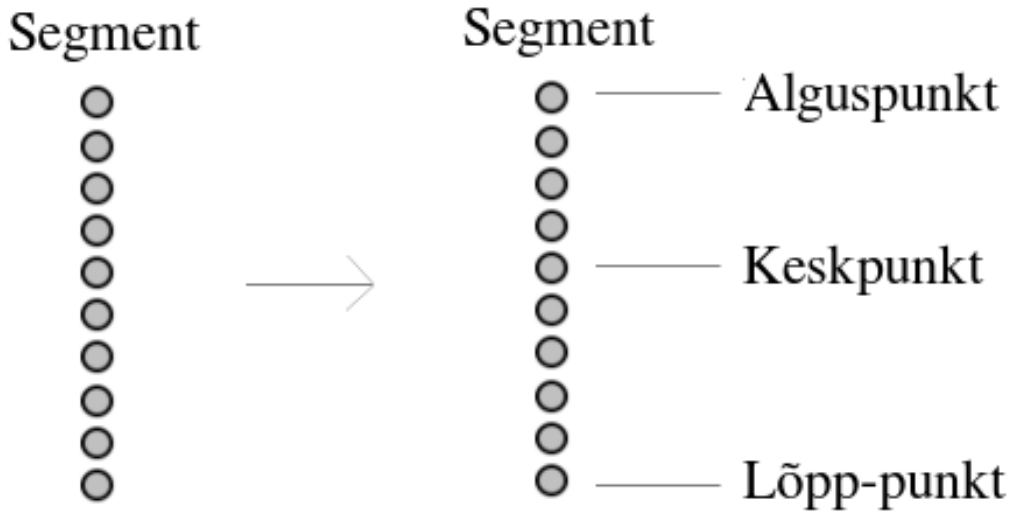
Esimene osa automaatse sõidujoone genereerimise lahendusest võtab iteratiivselt andmetest punkte välja *N* järjestikuse punkti kaupa (vt Joonis 7). See järjend antakse edasi järgmisele

funktsioonile, mis tagastab seda segmenti kirjeldavad punktid. Sirge segmendi kirjeldamiseks on vaja vähem punkte, kuid kurvi kirjeldamiseks jällegi rohkem, et tagada kurvi sujuvus.



Joonis 7. Näide kuidas punkti trajektoorist võetakse kümme punkti segmendina välja

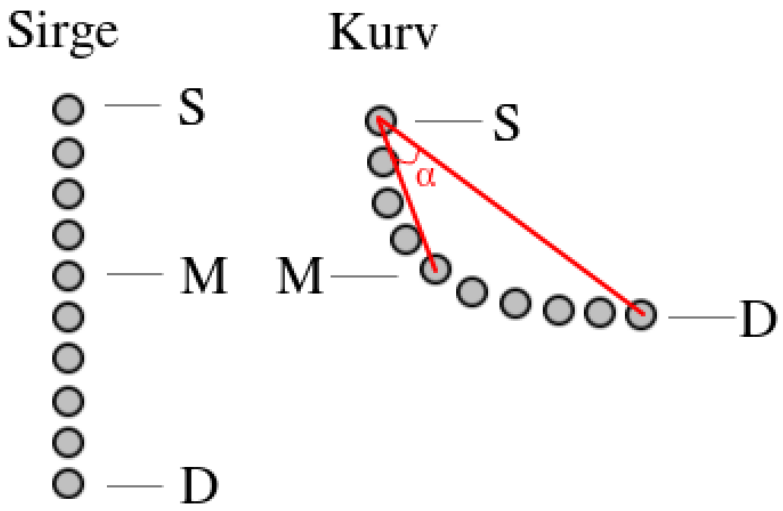
Koostatud segmendid võtab sisse järgmine funktsioon, mille eesmärk on leida vähim arv punkte, millega on võimalik ühte segmenti kirjeldada. Samas tagades, et informatsioon lõigu kohta ei läheks kaduma. Selle segmendi kirjeldamiseks proovib rakendus leida kolm punkti - alguspunkt, keskmise punkt ja lõpp-punkt (vt Joonis 8). Kui segmendis on üks või kaks punkti, siis järelikult need on segmenti kirjeldavad punktid ja kood tagastab need.



Joonis 8. Segmendi algus-, kesk -ja lõpp-punkti valimine

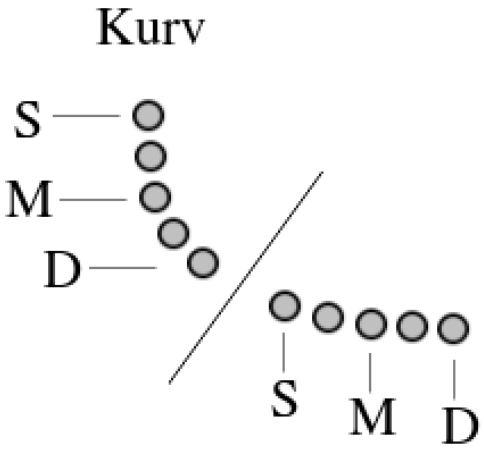
3.2.2 Kirjeldavate punktide võrdlemine

Leides eelnevalt mainitud kolm kirjeldavat punkti, tehakse nende abil vektorid ja võrreldakse nende vahelist nurka. Selle abil tuvastatakse, kas segment kirjeldab sirget või kurvi (vt Joonis 9). Segmenti loetakse kurviks, kui vektorite nurgad erinevad rohkem kui kolm kraadi.



Joonis 9. Segmenti kirjeldavatest punktidest loodud vektorite võrdlemine

Kui tegu on sirgega, siis piisab eelnevalt mainitud kolmest kirjeldavast punktist, et segmenti kirjeldada. Kurvi korral kasutab rakendus jaga ja valitse lähenemist, ehk ta teeb segmendi pooleks ja otsib uuesti kirjeldavaid punkte (vt Joonis 10).



Joonis 10. Segmentide poolitamine ja kirjeldavate punktide loomine kurvi korral

Rakendus otsib rekursiivselt kirjeldavaid punkte nii kaua, kuni on jõutud sirgeni või töötlemisel olevasse segmenti jääb vähem kui kolm punkti. See tähendab, et kui tegu on eriti järsu kurviga, siis selle kirjeldamiseks kasutatakse kõiki segmendis olevaid punkte ja ei

üritata nende hulka vähendada. Selline lähenemine tagab selle, et sõidutee lõigud, mille kirjeldamiseks on vaja vähem punkte, nagu näiteks sirged lõigud, kasutavad vähem punkte ja kurvid jällegi rohkem.

3.3 Kirjeldavate punktide keskmistamine

Leides kirjeldavad punktid, hakatakse tulemust keskmistama. Keskmistamiseks otsitakse varem genereeritud lähimate naabrite puu andmestikust (vt ptk 3.1) kirjeldavatele punktidele lähedal olevad punktid, mis asuvad teatud fikseeritud raadiuses. Käesoleva lõputöö raames tehakse lähimate naabrite otsingut üle 86000 punktiga andmestikust, mis kirjeldavad kokku kaheksat erinevat trajektoori.

Kui lähimad punktid on leitud, teostatakse leitud punktidele valideerimine. Keskmistamiseks kasutatakse nende trajektooride punkte, millel on sama suund praeguse segmendiga. Lisaks kontrollitakse, et punkti ei oleks varem keskmistamiseks kasutatud. Kahe juhi sõidetud trajektoori võivad ühel lõigul kattuda, kuid hiljem erinevatesse suundadesse pöörata. See aga tähendab, et raadiuse abil keskmistades tekivad teravad kurvid (vt Joonis 11).

Joonis 11. Kahe trajektoori hargnemisel tekkiv keskmistamise viga kujutatud joonena

Selleks, et skript teaks, et neid punkte mitte arvestada, on vaja teha keskmistamise hetkel kontroll, selgitamaks, kas punktid liiguvad ühes suunas. Selle probleemi lahendamiseks

arvutatakse sarnaselt kurvi tuvastamise loogikale vektorite vaheline nurk ja võrreldakse neid. Nii saab kontrollida, segmentide suunda. Segmendid loeme mitteparalleelseks, kui nende nurkade vahe on üle kolme kraadi.

Leides sobivad lähimad punktid, selgitatakse nende kiiruse ja positsiooni aritmeetiline keskmine.

3.3.1 Korduste vältimine

Vältimaks sama segmendi uuesti kirjeldamist, kasutatakse andmestikuga sama pikka järjendit. Selle järjendi eesmärk on hoida binaarset väärtust, mis kirjeldab, kas punkti on töödeldud. Väärtus 0 tähendab, et punkti ei ole varem kasutatud keskmistamiseks ja väärtus 1 tähendab, et on. Näiteks kui kontrollida, kas algandmetest indeksil 100 olevat punkti on varem kasutatud, saab indeksi abil seda kiiresti kontrollida binaarsete väärtustega järjendist. Numpy järjend valiti sellepärast, et võrreldes tavalise Pythoni järjendiga on ta kiirem ja võtab vähem ruumi [32].

Kuna segmendis on N punkti ja üldiselt $N > 3$, siis lisaks sellele et kirjeldavate punktide loomisel märgitakse keskmistamisel kasutatud punktid loetuks, märgitakse lisaks ülejäänud segmendi punktid ja nende lähedal olevad sama suunaga punktid kasutatuks. Seda tehakse sellepärast et sõidutee lõik, mida segment kirjeldab on tehtud valmis ja rohkem ei ole neid andmeid vaja kasutada.

3.3.2 Järjestamine

Peale kirjeldavate punktide leidmist ja kõigi andmete töötlemist laetakse saadud andmed geoinfosüsteemi QGIS tarkvarasse. Seal on võimalik tulemusi visuaalset kontrollida ja kasutada olemasolevaid skripte trajektoori sujuvamaks tegemiseks. Lisaks on seal olemas ka tööriist, mis oskab loodud punktidest teha joone, mis sarnaneb sõidujoonele. Selle joone tegemiseks on oluline, et andmed oleksid järjestatud ja et loodud sõidujoontel oleks grupeerimise indeks.

Kui punktid ei ole järjestatud ega grupeeritud, võib juhtuda olukord, kus punkte järjest sisse lugedes jääb kahe punkti vahele suur distants. Et nende kahe punkti vahele joont ei tekiks antakse neile erinev grupi indeks. Kui kahel järjestikusel punktil on erinev grupi tunnus, siis nende vahele joont ei tehta (vt Joonis 12).

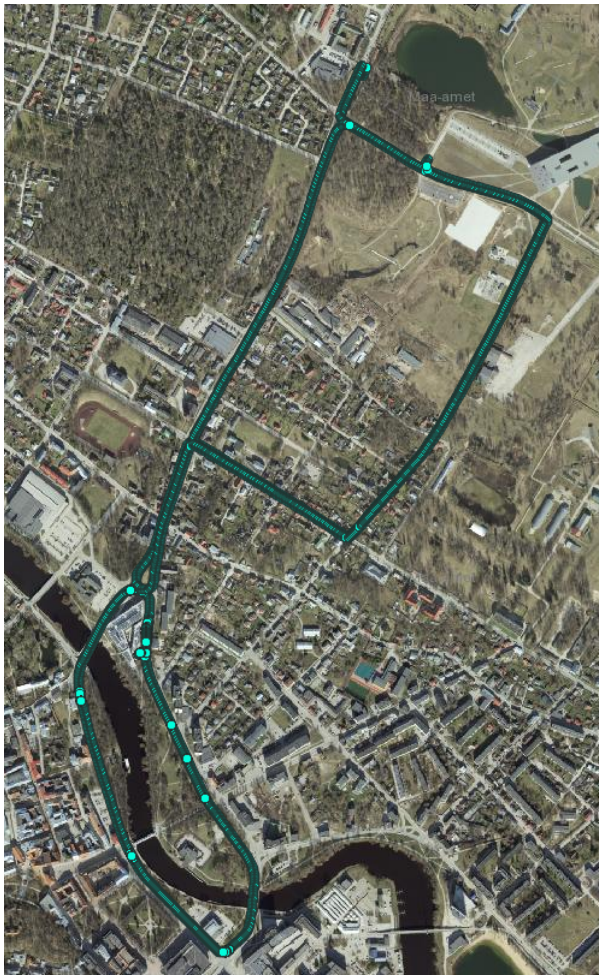
Joonis 12. Näide rakenduse väljundist, kui grupeerimine ja järjestamine ei tööta

Punktide järjestuse tagamiseks lisatakse punkti salvestamisel ka tema järjekorra indeks. See tagab, et üksteise järel olevad punktid ühendatakse hiljem õiges järjekorras üksteisega. Lisaks sellele, et tagatakse ühel trajektooril punktide järjestus, lisatakse igale loodavale andmepunktile grupi indeks. Grupi indeks defineerib ära, mis lõiguga hetkel tegu on, ning grupi indeks muutub sellel hetkel, kui kahe punkti vaheline kaugus on suurem kui 5 meetrit. See tagab selle, et kui viimane punkt on näiteks Delta ees ja järgmine punkt hakkab ERM-i ees trajektoori kirjeldama, siis QGIS ei tõmba nende kahe vahele sirget.

4. Tulemus

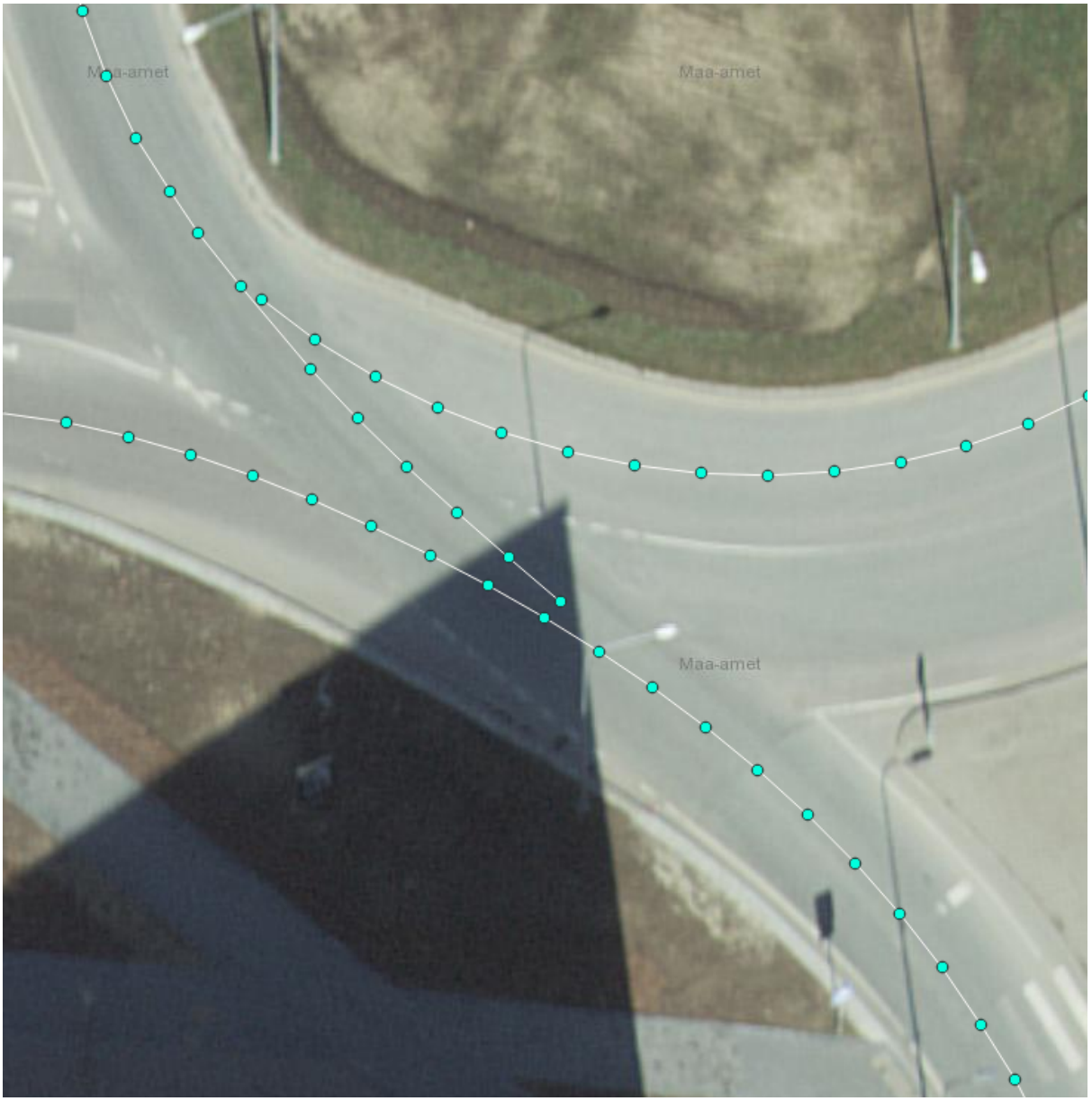
4.1 Genereeritud trajektoor

Lõputöö raames kasutatud andmetes oli kokku üle 86000 andmepunkti. Nende abil genereeriti 5,5 km pikkune sõidujoon, mis koosneb umbes 4000-st punktist (vt Joonis 13). Automaatselt sõidujoone genereerimine võttis kokku 9,9 sekundit.



Joonis 13. Trajektoori loomiseks automaatselt genereeritud punktid. Suuremad punktid märgivad kohti, kus sõiduk peatus

Genereeritud punktide abil saab tarkvaras QGIS teha sõidujoone. Loodud trajektoor vajab siiski lõplikuks kasutamiseks järeltöötlust. QGIS-i silumisalgoritm aitab teha joont sujuvamaks. Lisaks on vaja käsitsi ühendada kohad, kus lõik hargneb (vt Joonis 14). See vajadus tuleneb QGIS-i punktide jooneks (ingl. k *points to path*) tegemise skripti piirangutest, mille tõttu saab joon olla ühe suunaga ja ei saa hargneda. Isejuhtimise võimaldamiseks peab trajektoor aga olema ühtne suletud joon.



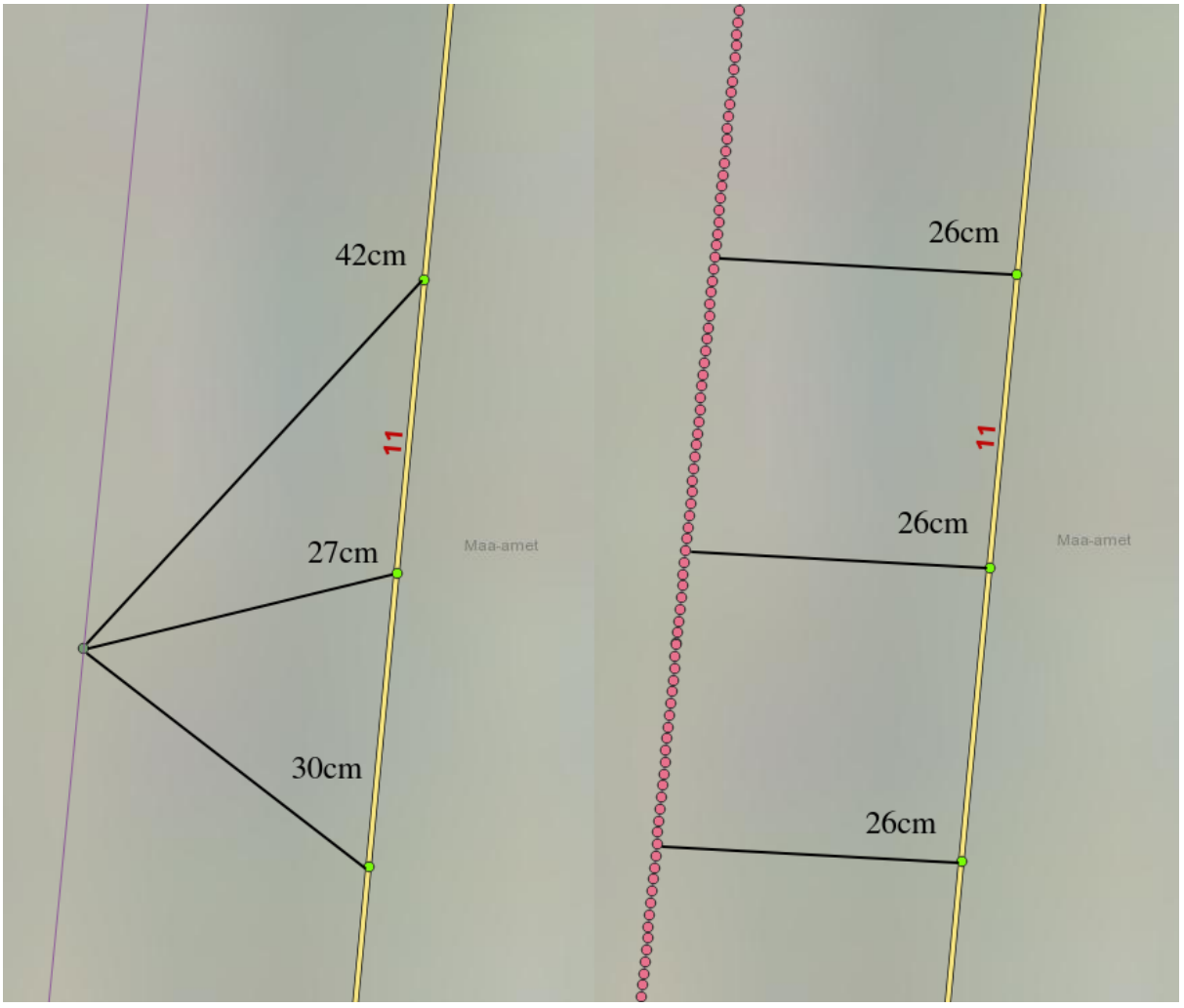
Joonis 14. Automaatselt genereeritud sõidujoon ja selle punktid. Ühendamata lõigud illustreerivad punktidest joone tegemise skripti puudust

Lisaks, et loodav sõidujoon oleks täielikult kasutatav isejuhtiva sõiduki jaoks, on vaja genereeritud sõidujoon lõigata juppideks loogilistes kohtades nagu ristmiku algus, ülekäiguraja algus ja stoppjoon (vt Joonis 3). Sõidulõigu automaatne loomine ja selleks varasema info mujalt sissetoomine ja töötlemine ning seejärel genereeritud sõidujoonega ühildamine ei olnud triviaalne ülesanne ning ei mahtunud ajapiirangutesse.

4.2 Tulemuse täpsuse mõõtmine

Lõputöös valminud rakenduse tulemuse võrdlemiseks ja täpsuse mõõtmiseks loodi eraldi skript ja andmed. Täpsuse mõõtmiseks on käsitsi joonestatud sõidujoon viidud tagasi punktide jadaks. Seejärel otsiti igale punktile käsitsi joonestatud sõidujoones lähim punkt võrdlustrajektooris ja mõõdeti kaugus.

Selleks, et trajektoorid oleksid võrreldavad, suurendati võrdlemise all olevate trajektooride punktide jadade punktitihedust. See tagab selle, et lähim punkt on alati võimalikult lähedal ja tulemused võrreldavad. Joonisel 15 on näha, kuidas käsitsi joonestatud sõidujoone trajektoori rohelistele punktidele otsiti teiselt trajektoorilt lähimat punkti. Välja on toodud, kuidas punktide tihedus mõjutab leitavat kaugust.



Joonis 15. Puntki tiheduste mõju lähima naabri kauguse arvutusel

Käsitsi joonestatud sõidujoone suhtes on kokku tehtud kolm võrdlust. Esiteks on mõõdetud, kui täpselt autonoomse sõidu ajal isejuhtiv sõiduk käsitsi joonestatud sõidujoont järgib.

Teiseks on mõõdetud, kui sarnane on käsitsi joonestatud sõidujoon ühele inimjuhi tehtud sõidule. Viimaks on testitud, kui sarnased on automaatselt genereeritud sõidujoone punktid käsitsi joonestatud sõidujoone trajektoorile.

Tulemused olid järgnevad:

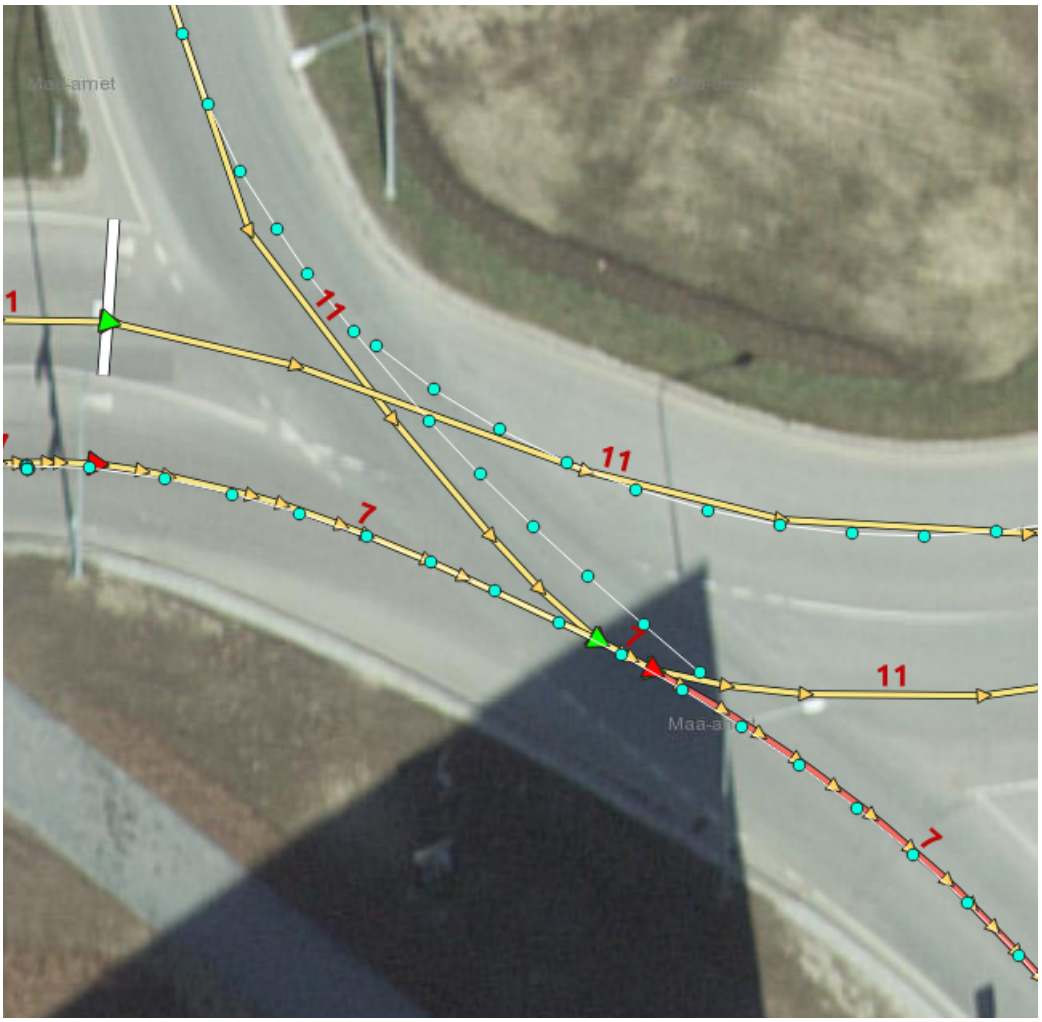
- Ühe inimjuhi sõidu punktide keskmine kaugus sõidujoonest on 22 cm;
- Ühe automaatse sõidu punktide keskmine kaugus sõidujoonest on 12 cm;
- Automaatselt genereeritud sõidujoone punktide keskmine kaugus sõidujoonest on 14 cm.

Automaatse sõidu trajektoori täpsuse võrdlemine annab aimu, kui täpselt isejuhtiv sõiduk tegelikult käsitsi joonestatud sõidujoont jälgib (vt Joonis 16). Üldiselt järgib isejuhtiv sõiduk sõidujoont hästi, kuid on näha, et kurvides kaldub sõiduk sellest eemale. Seetõttu tuleb keskmine kaugus sõidujoonest 12 cm.

Joonis 16. Autonoomse (roheline) sõidu erinevus sõidujoonest (kollane)

Automaatselt genereeritud sõidujoone loomiseks kasutati inimjuhi sõite. Nende tulemuste keskmistamisel saavutab genereeritud sõidujoon sarnasema tulemuse käsitsi joonestatud sõidujoonele, kui üks individuaalne inimjuhi sõit. Seda seetõttu, et automaatselt sõidujoone tegemisel kasutatakse mitme erineva sõidu trajektoore ja osadel sõitudel on sõiduk sõidujoonest vasakul pool ja osadel sõitudel paremal pool (vt Joonis 6).

Tulemuste võrdlemisel tuleb arvestada, et otseselt ei saa öelda, kas 14 cm kaugus käsitsi joonestatud sõidujoonest on hea või halb. Tulemuste visuaalsel võrdlusel selgub, et automaatselt genereeritud sõidujoon on sujuvam (vt Joonis 17).



Joonis 17. Kollaselt on kujutatud sõidujoon ja valge joon on genereeritud sõidujoon

Käsitsi joonestatud sõidujoon on oletuslik sõidutee keskjoon, mida mööda sõidukid liiklevad. Kuna lõputöö raames valminud rakendus genereerib sõitude põhjal keskjoone, siis piisavate trajektooride keskmistamise korral saaks öelda, et hoopis praegune käsitsi joonestatud sõidujoon on keskmiselt näiteks 14 cm kaugusel tegelikust keskjoonest. Võrdluseks tasub arvestada, et Delta õppehoone eest mööduva ühe sõiduraja laius on umbes kolm meetrit.

4.3 Piirangud

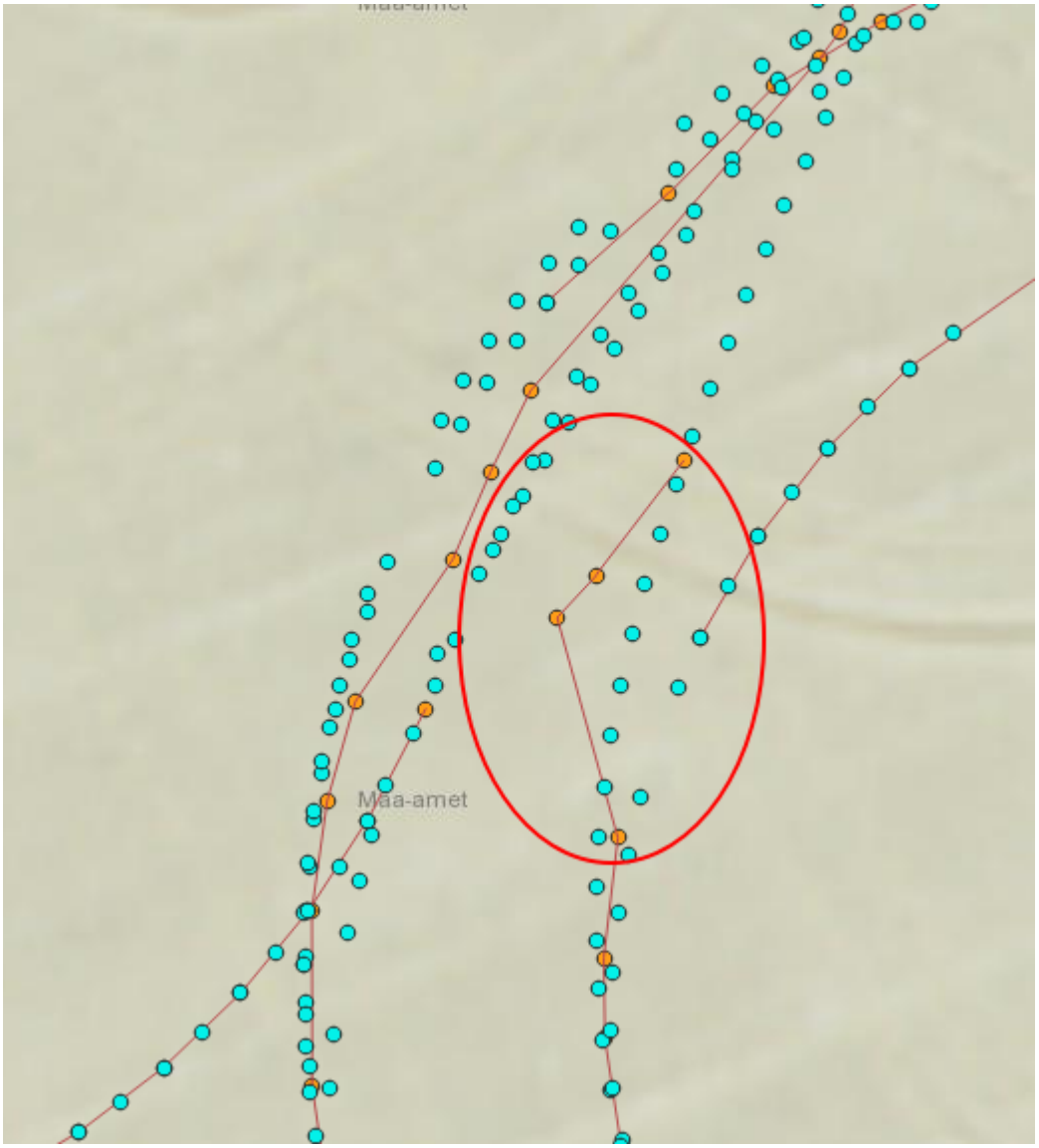
Tartu andmete põhjal genereeritud tulemus on visuaalselt hea, kuid tuleb arvestada, et tegu on peamiselt ühe sõidurajaga, kus on vähe ristmike. Lahenduse piiride testimiseks kasutati Viljandi testrada, kus on üle kaheksa tuhande punkti ja väikese ala peal palju ringe ja ristmikke (vt Joonis 18).

Joonis 18. Viljandi testringi kujutavad punktid

Raadiuse abil keskmistades, võib juhtuda, et keskmistatakse ka punktid mida tegelikult ei tohiks. Näiteks ei tohiks ristmikul punktide keskmistamisel arvestada risti mineva trajektoori punktidega. Selle vastu aitab vektorite võrdlemine, et näha mis suunas punktid liiguvad. Keskmistamisel on kasutatud ainult neid punkte, millel on sama suund.

Joonisel 19 on näha, kuidas helesinised punktid kirjeldavad kahte erinevat trajektoori. Need helesinised punktid keskmistatakse, et saada oranžid punktid. Jooniselt on näha, et parempoolse trajektoori keskmistatud punktid lähevad järsult vasakule. Keskmistamisel jäävad raadiuse sisse ka teise vasakpoolse trajektoori helesinised punktid, mis on paralleelsed parempoolse trajektoori punktidega.

Selle probleemi tõttu on mõlemad rajad vähem sujuvad, kuna punktid, mida üks rada peaks kasutama on kasutusel juba teise raja poolt.

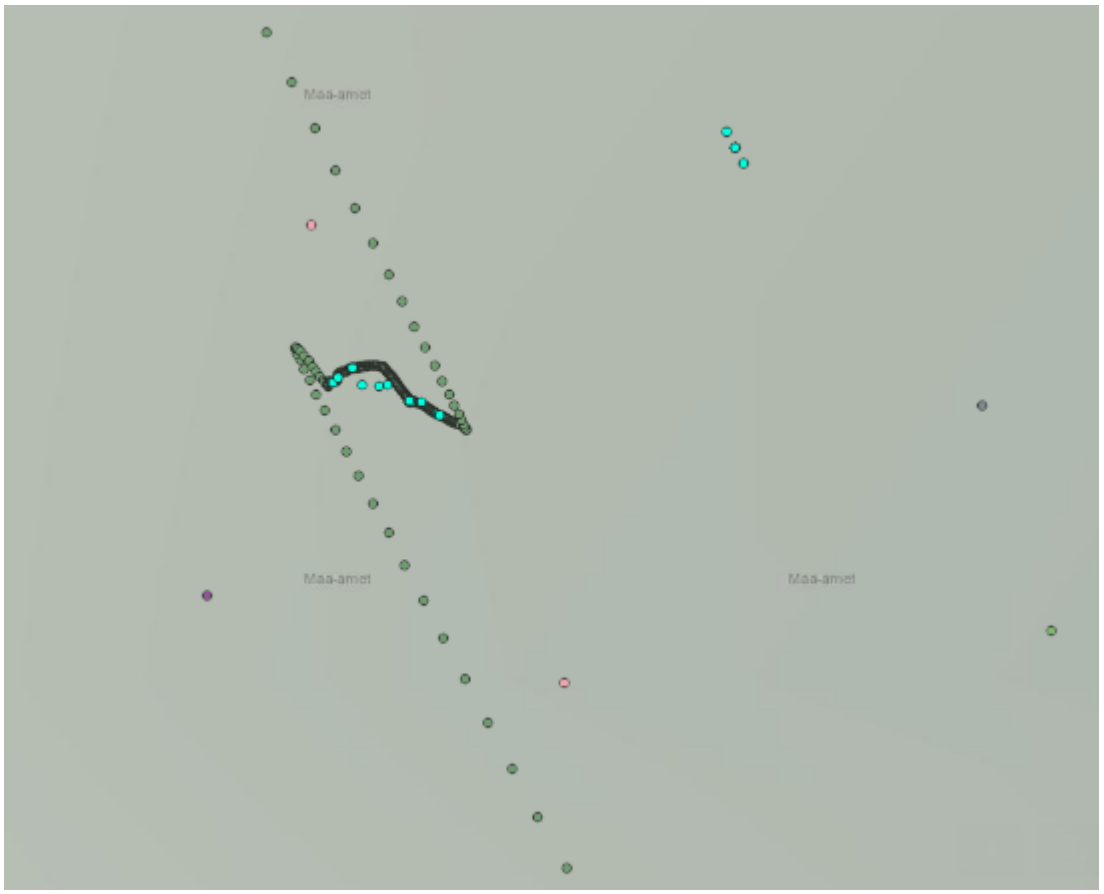


Joonis 19. Keskmistamisel on kasutatud teise raja punkte

Osade puuduste kõrvaldamiseks saab kindlasti muuta rakenduse parameetreid. Rakenduses on võimalik defineerida, mis nurga all peavad vektorid olema, et neid loetakse paralleelseks, ning näiteks mis raadiuse piires lähimaid punkte otsida. Nende parameetrite muutmine võib aga mõjutada tulemust ja tekitada teistsuguseid erindeid.

Joonisel 20 on näha roheliselt ühe sõidu trajektoori punkte. Kui sõiduk jääb seisma, siis GNSS + RTK positsioneerimise tõttu auto trajektoor nihkub. Jooniselt on näha, et nihke kohta on genereeritud palju punkte. Vektorite võrdlemise tõttu neid punkte keskmistamisel ei kasutata ja need esitatakse eraldi. Selle tõttu on ka Joonisel 13. näha lisaks sõidujoont

kirjeldavatele punktidele ka eraldi suuremaid punkte, mis märgivad auto peatumise kohad. Korrastatud andmetes selliseid nihkeid ei ole, ning see viitab sellele, et osi algandmeid jäid töötlemata. Võimaliku edasiarendusena saaks luua ka erindite eemaldamise loogika.



Joonis 20. Sõiduki peatumisel toimub trajektoori nihkumine

4.4 Edasised arendused

Lõputöö raames valminud lahendus loob keskmistatud sõidujoone andmed. Järgmine samm nende andmetega võiks olla täieliku sõiduraja loomine. Praegusel hetkel tuleb genereeritud andmed käsitsi ühendada, kuna QGIS-is punktide jooneks (ingl. k *points to path*) tegemise skript ei oska arvestada sõidujoonte hargnemisega (vt Joonis 14). Lisaks tuleks sõidulõikude tegemiseks stoppjoonte ja ülekäiguradade juures sõidujoonel ära märkida lõpp- ja alguspunkt. Selle võimekuse lisamine jäi lõputöö raames tegemata, kuna sõidujoonest sõidulõikude tegemine ei ole triviaalne ülesanne, ning ei mahtunud lõputöö ajaraamidesse.

Lõputöö autor on kindel, et koodi oleks võimalik optimeerida ja lihtsustada. Loodud lahendust loodi järk järgult, et probleemiga tutvuda. Sellest tulenevalt ei olnud kohe lahenduse arhitektuur teada. Näiteks saaks tagantjärele vaadates lihtsustada segmentide

loomisel kasutatud punktide märkimist. Hetkel tehakse punktide loetuks märkimist kahes kohas ja arvutatakse punktide suuna võrdlemiseks vektorid pidevalt uuesti. Tegelikult ei oleks vaja näiteks sirgete korral vektoreid uuesti arvutada ja punktide märkimist saaks teha ainult ühes kohas, peale segmendi loomist. Koodi profileerimise käigus selgus, et kolmandik ajast kulub punktide loetuks märkimisele.

Kokkuvõte

Lõputöö eesmärk oli uurida võimalusi sõidujoone andmete genereerimiseks ja luua selle jaoks lahendus. Loodav lahendus kasutab inimjuhi sõitude andmeid ja keskmistab need. Rakenduse väljund on CSV-fail, milles olevate andmete abil on võimalik genereerida näiteks geoinfosüsteemi QGIS abil täppiskaardi sõidujoon.

Loodud lahendus võimaldab kasutada palju andmeid sõidujoone genereerimiseks, et loodav tulemus kirjeldaks tegelikku kiirust ja keskjoont sõiduteel. Loodud lahendus on ülesande ja sisendandmete spetsiifiline, et tagada rakenduse sujuvus ja hea tööaeg. Näiteks eeltöötluse käigus korrastatakse sisendandmed ja kasutatakse erinevaid andmestruktuure, et tagada hea tööaeg.

Töö käigus kasutati kaheksa sõidu andmeid, kus oli üle 86000 punkti, millega loodi umbes 5,5 km pikkune trajektoor, mis genereeriti 9,9 sekundiga. Automaatselt genereeritud sõidujoont võrreldi varasemalt käsitsi joonestatud sõidujoonega. Keskmiselt erines automaatselt genereeritud sõidujoon 14 cm praegu kasutusel olevast sõidujoonest, võrdluseks inimjuhi sõidu ja autonoomse sõidu keskmine kaugus sõidujoonest oli vastavalt 22 ja 12 cm. Visuaalse kontrolli käigus selgus, et automaatselt genereeritud sõidujoon on sujuvam ja peegeldab paremini tegelikku liikluskorraldust.

Loodud tulemust ei saa otse kasutusele võtta Tartu Ülikooli isejuhtivate sõidukite labori täppiskaardil. Geoinfosüsteemi QGIS punktide jooneks (ingl. k *points to path*) tegemise skripti piirangute tõttu vajab automaatselt genereeritud sõidujoon käsitsi lõikude ühendamist. Lisaks jäi käesoleva lõputöö eesmärkidest välja täppiskaardi teiste tunnustega arvestamine. Sellest tulenevalt on vaja veel teha manuaalselt sõidujoone tükeldamine sõidulõikudeks, kus arvestatakse ka teiste tunnustega.

Lisaks tõi autor puudusena välja raadiuse abil keskmistamisest tulevad vead. Kui kaks erinevat trajektoori, mis kirjeldavad erinevaid radu, satuvad liiga lähedale, siis mingites

punktides, kus nende suund on sama keskmistatakse need kokku. Seetõttu võib osa vajalikust infost kaduma minna ja kahest erinevast sõidurajast saab üks.

Autor toob välja ka võimalusi edasiarenduseks. Lisaks muu infoga arvestamisele on edaspidi võimalik rakenduse arhitektuuri lihtsustada ja optimeerida. Praeguses lahenduses tehakse ka sirgete korral keerulisemat vektorite vahelist võrdlust, kui tegelikult on vaja. Arvestades, et üle kolmandiku ajast kulub sellele, siis rohkemate andmete korral on ajaline võit arvestatav.

Lähtekoodi nägemiseks kirjutada autorile: looga.krister@gmail.com

Tänan oma juhendajaid Tambet Matiiseni ja Edgar Seppa võimaluse eest teha lõputöö huvitavas ja innovaatilises valdkonnas. Olen tänulik juhendajate pidevale toetusele ja põhjalikule tagasisidele. Tahaksin tänada ka Lõputöö Seminari juhendajat Margus Niitsood põhjaliku tagasiside eest.

Lõputöö valmimist toetas IT-akadeemia.

Viidatud kirjandus

1. Chellapilla K. Rethinking Maps for Self-Driving. Medium, 2018. https://medium.com/lyftself-driving/https-medium-com-lyftlevel5-rethinking-maps-for-self-driving-a147c24758d6 (10. 04.2021)
2. Poggenhans F, Pauls J-H, Janosovits J, Orf S, Naumann M, Kuhnt F, jt. Lanelet2: A high-definition map framework for the future of automated driving. *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. USA: IEEE, 2018. lk 1672–9. doi: 10.1109/ITSC.2018.8569929.
3. Anderson JM, Kalra N, Stanley KD, Sorensen P, Samaras C, Oluwatola OA. Brief History and Current State of Autonomous Vehicles. *Autonomous Vehicle Technology.* RAND Corporation, 2014, lk 55–74. https://www.jstor.org/stable/10.7249/j.ctt5hhwgz.11 (24.04.2021)
4. Society of Automotive Engineers. SAE International Releases Updated Visual Chart for Its "Levels of Driving Automation" Standard for Self-Driving Vehicles. 2018. https://www.sae.org/site/news/press-room/2018/12/sae-international-releases-updated-visual-chart-for-its-%E2%80%9Clevels-of-driving-automation%E2%80%9D-standard-for-self-driving-vehicles (25.04.2021)
5. Baldwin R. Tesla Tells California DMV that FSD Is Not Capable of Autonomous Driving. Car and Driver, 2021. https://www.caranddriver.com/news/a35785277/tesla-fsd-california-self-driving/ (25.04.2021)
6. The Waymo Team. Waypoint - The official Waymo blog: Building maps for a self-driving car. Waymo Blog, 2016. https://blog.waymo.com/2019/09/building-maps-for-self-driving-car.html (11.04.2021)
7. Krafcik J. Waypoint - The official Waymo blog: Waymo is opening its fully driverless service to the general public in Phoenix. Waymo Blog, 2020 https://blog.waymo.com/2020/10/waymo-is-opening-its-fully-driverless.html (25.04.2021)
8. National Highway Traffic Safety Administration. Automated Vehicles for Safety. NHTSA, 2017. https://www.nhtsa.gov/technology-innovation/automated-vehicles-safety (25.04.2021)
9. Eliot L. Explaining Level 4 And Level 5 Of Self-Driving Cars In Plain English. Forbes, 2019.

https://www.forbes.com/sites/lanceeliot/2019/12/20/explaining-level-4-and-level-5-of-self-driving-cars-in-plain-english/ (1.05.2021)

10. Turk K, Pild M, Blumfeldt E. ANALÜÜS SAE TASE 4 JA 5 SÕIDUKITE KASUTUSELE VÕTMISEKS KOOS SEADUSEELNÕU VÄLJATÖÖTAMISKAVATSUSE KIRJELDUSTEGA. 2017. https://triniti.ee/wp-content/uploads/sites/2/2017/09/Anal%C3%BC%C3%BCs-SAE-tase-4-ja-5-s%C3%B5idukite-kasutusele-v%C3%B5tmiseks_Riigikantselei_20....pdf (5.05.2021)
11. Seif H, Hu X. Autonomous Driving in the iCity—HD Maps as a Key Challenge of the Automotive Industry. *Engineering*, 2016, nr 2, lk 159–162. https://doi.org/10.1016/J.ENG.2016.02.010
12. Vardhan H. What are HD Maps, how are HD Maps made, HD Maps for self-driving cars. Geospatial World, 2017. https://www.geospatialworld.net/article/hd-maps-autonomous-vehicles/ (25.04.2021)
13. Dahlström T. How Accurate Are HD Maps for Autonomous Driving and ADAS Simulation?. Atlatec, 2020. https://news.atlatec.de/how-accurate-are-hd-maps-for-autonomous-driving-and-adas-simulation/ (29.04.2021)
14. Lu D. 8 degrees of difficulty for autonomous navigation & robots. The Robot Report, 2020 https://www.therobotreport.com/8-degrees-difficulty-autonomous-navigation-robots/ (25.04.2021)
15. Efland K, Rapp H. Semantic Maps for Autonomous Vehicles. Medium, 2019. https://medium.com/lyftself-driving/semantic-maps-for-autonomous-vehicles-470830ee28b6 (25.04.2021)
16. Surmenok P. HD Maps for Self-Driving Cars. I used to think that the maps are… | by Pavel Surmenok | Medium. Medium, 2018. https://medium.com/@surmenok/hd-maps-for-self-driving-cars-c41bc01e0d40 (25.04.2021)
17. Sepp E. Creating High-Definition Vector Maps for Autonomous Driving. TÜ arvutiteaduse instituudi magistritöö. 2021. https://comserv.cs.ut.ee/ati_thesis/datasheet.php?id=71505&year=2021
18. Ortofotod. Maa-amet, 2021. https://geoportaal.maaamet.ee/est/Ruumiandmed/Ortofotod-p99.html (4.05.2021)

19. How we make our HD Maps | TomTom Blog. TomTom, 2020. https://www.tomtom.com/blog/autonomous-driving/how-we-make-our-hd-maps/ (11.04.2021)
20. Matiisen T. Tartu Ülikooli isejuhtiv auto on jõudnud linnaliiklusesse. Universitas Tartuensis, 2020 https://www.ajakiri.ut.ee/artikkel/3881 (25.04.2021)
21. Fränti P, Mariescu-Istodor R. Applied Sciences: Averaging GPS segments. 2019. http://cs.uef.fi/sipu/segments/ (25.04.2021)
22. Fränti P, Mariescu-Istodor R. Averaging GPS segments competition 2019. *Pattern Recognition*, 2021, nr 122. https://doi.org/10.1016/j.patcog.2020.107730
23. AKIT - Andmekaitse ja infoturbe leksikon. https://akit.cyber.ee/term/3029-erand-erind
24. Marteau P-F. Estimating Road Segments Using Kernelized Averaging of GPS Trajectories. Appl Sci, 2019. https://hal.archives-ouvertes.fr/hal-02176080 (11.04.2021)
25. Yang J, Mariescu-Istodor R, Fränti P. Three Rapid Methods for Averaging GPS Segments. *Appl Sci*, 2019, nr 22. https://doi.org/10.3390/app9224899
26. Python Core Team. Python: A dynamic, open source programming language. Python Software Foundation, 2015. https://www.python.org/about/ (2.05.2021)
27. Harris CR, Millman KJ, van der Walt SJ, Gommers R, Virtanen P, Cournapeau D, jt. Array programming with NumPy. *Nature*, 2020, nr 585, lk 357–362. https://doi.org/10.1038/s41586-020-2649-2
28. The pandas development team. pandas - Python Data Analysis Library. Zenodo, 2020. https://doi.org/10.5281/zenodo.3509134
29. Kluyver T, Ragan-Kelley B, Pérez F, Granger B, Bussonnier M, Frederic J, jt. Jupyter Notebooks – a publishing format for reproducible computational workflows. *IOS Press*, 2016, lk 87–90. https://eprints.soton.ac.uk/403913/ (3.05.2021)
30. QGIS Development Team. QGIS Geographic Information System. 2021. https://qgis.org/en/site/about/index.html (1.05.2021)
31. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, jt. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 2011, nr 12, lk 2825–2830. http://jmlr.org/papers/v12/pedregosa11a.html (30.04.2021)
32. Dr. Kider, Dr. Wiegand. Python Lists vs. Numpy Arrays - What is the difference?. Webcourses UCF, 2017. https://webcourses.ucf.edu/courses/1249560/pages/python-lists-vs-numpy-arrays-what-is-the-difference (25.04.2021)

Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, Krister Looga,

1. Annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose "Vektorkaardi genereerimine isejuhtivale autole trajektooride põhjal", mille juhendajad on MSc Tambet Matiisen ja MSc Edgar Sepp, reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.
2. Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace kaudu Creative Commonsi litsentsiga CC BY NC ND 3.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni.
3. Olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.
4. Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Krister Looga

01.05.2021