Tartu Ülikool

Loodus- ja täppisteaduste valdkond

Tehnoloogiainstituut

Kristo Allaje

DRAIVERIPAKETT KÄTE JÄLGIMISE SEADME LEAP MOTION™ KONTROLLER KASUTAMISEKS ROBOOTIKA ARENDUSPLATVORMIL ROS

Bakalaureusetöö (12 EAP)

Arvutitehnika eriala

Juhendaja:

Robootika dotsent Karl Kruusamäe

Tartu 2018

Resümee / Abstract

Draiveripakett käte jälgimise seadme Leap Motion™ kontroller kasutamiseks robootika arendusplatvormil ROS

Järjest laienev robotite kasutusvaldkond nõuab inimeste ja robotite vahel üha tihedamat koostööd, mida kirjeldatakse koondterminiga inimese ja roboti suhtlus (ingl *„Human-Robot Interaction“*). Seetõttu on vajalik arendada töökindlaid meetodeid inimeste tuvastamiseks ning luua intuitiivseid kasutajaliideseid robotite kergeks juhtimiseks.

Käesoleva bakalaureusetöö eesmärgiks on luua draiver, mis võimaldaks kasutada käte jälgimise seadet Leap Motion kontroller (LM-kontroller) robootika arendusplatvormil ROS (*Robot Operating System*). Kuigi arendusplatvormile ROS eksisteerib juba LM-kontrollerile draiver, siis selle funktsionaalsus ja dokumentatsioon on puudulikud ning alates 2014. aastast pole seda arendatud, sest koodihoidlal puudub aktiivne haldaja. Bakalaureusetöö raames valminud LM-kontrolleri draiveripaketi eesmärgiks on laiendada olemasoleva paketi võimalusi, tõsta seadme kasutajamugavust nii kasutajatele kui ka arendajatele ning koostada terviklik dokumentatsioon (paigaldusjuhendid ja lähtekoodi kommentaarid).

CERCS: T120 Süsteemitehnoloogia, arvutitehnoloogia; T125 Automatiseerimine, robootika, control engineering;

Märksõnad: robootika, avatud lähtekood, Leap Motion, ROS, HRI

Driver package for using Leap Motion™ controller on the robotics development platform ROS

The ever-expanding field of robotics requires more and more interaction between humans and robots, described by the term human-robot interaction. Therefore, it is necessary to develop robust methods for identifying people and create intuitive user interfaces for easy control of robots.

The aim of this bachelor’s thesis is to create a driver that would allow the use of Leap Motion (LM) controller hand tracker on the robotics development platform ROS (Robot Operating System). Although ROS repositories already contain a driver for the LM controller, its functionality and documentation are incomplete and the software package has not been updated since 2014 due to the inactivity of its maintainers. The goal of this thesis is to develop an improved LM controller driver package that expands the capabilities of the existing package and, enhances user experience. Additionally, the developed driver package is well-documented by providing explicit installation instructions and extensively adding comments to its source code.

CERCS: T120 Systems engineering, computer technology; T125 Automation, robotics, control engineering

Keywords: robotics, open-source, Leap Motion, ROS, HRI

Sisukord

Jooniste loetelu

Lühendid, konstandid, mõisted

API (*Application Programming Interface*) ehk rakendusliides võimaldab suhtlust erinevate tarkvarakomponentide vahel kasutades eeldefineeritud eeskirju.

CI (*Continuous Integration*) on tarkvaraarenduse tava, kus meeskonna liikmed integreerivad oma koodi teiste liikmete koodiga nii tihti kui võimalik.

Draiver (*Device driver)* toimib vahetarkvarana seadme ja seda kasutava programmi vahel.

HRI (*Human-Robot Interaction*) on koondtermin kirjeldamaks igasugust inimeste ja robotite vahelist suhtlust.

IMS Lab (*Intelligent Materials and Systems Lab*) ehk Tartu Ülikooli Tehnoloogiainstituudi arukate materjalide ja seadmete labor.

LRF (*Laser Range Finder*) ehk laserkaugusmõõtja on seade, mis kasutab objekti kauguse mõõtmiseks laserkiirt.

LMC, LM-kontroller (*Leap Motion™ Controller*) on käte asendit ja žeste tuvastav seade, mis ühendub arvutiga üle USB liidese.

LTS (*Long Term Support*) versioon mistahes tarkvarast on toetatud pikema ajaliseks kasutamiseks kui tavapärane versioon.

OS (*Operating System*) on kogum programme, mis juhivad, läbi riistavara ja tarkvara ressursside haldamise, arvutisüsteemi tööd.

ROS (*Robot Operating System*) on tarkvararaamistik robotite kasutamiseks, integreerimiseks ja arendamiseks.

RViz (*ROS Visualization*) on programm, mis võimaldab ROSis graafiliselt kuvada andurite näite ja robotite olekuteavet.

SDK (*Software Development Kit*) ehk arendustarkvara on kogum erinevatest tööriistadest, mille abil luuakse tarkvara kindla platvormi jaoks.

UT (*University of Tartu*) ehk Tartu Ülikool.

VR (*Virtual Reality*) ehk virtuaalreaalsus on arvuti poolt genereeritud keskkond, kus on võimalik tekitada inimesel illusiooni, et ta asub tehislikus 3D maailmas.

YOLO (*You Only Look Once*) on avatud lähtekoodiga reaalajaline, närvivõrkudele toetuv, objektituvastus süsteem.

1. Sissejuhatus

Kui küsida inimestelt, mida tähendab neile sõna robot, siis sageli kujutatakse ette pikka konveierliini ümbritsetud robotmanipulaatoritega, mis viivad täide ettenähtud liigutusi tuhandeid kordi päevas, seitse päeva nädalas ja nii 365 päeva aastas. Masinad, mis teostavad programmeeritud korduvaid liigutusi väga kiiresti, suure täpsuse ja töökindlusega.

Tegelikult on praeguseks paljud robotid leidnud oma töökoha inimeste kõrval, mitte olles neist teraspuuriga eraldatud. Näideteks võib tuua koostöörobotid nagu Baxter [1], inimeste kõrval otsingu- ja päästetöid teostavad robotid, isesõitvad autod ning kõiksugused teenindusrobotid alustades klienditeenindus robotist Pepper [2] ning lõpetades majapidamistesse soetatavate robottolmuimejate ja robotmuruniidukitega.

Järjest laienev robotite kasutusvaldkond nõuab inimeste ja robotite vahel tihedamat koostööd, mida kirjeldatakse koondterminiga inimese ja roboti suhtlus (ingl *„Human-Robot Interaction“*, lühidalt HRI). Seetõttu on vajalik arendada toimivaid meetodeid inimeste tuvastamiseks ning luua intuitiivseid kasutajaliideseid robotite kergeks juhtimiseks. Praegu kasutusel olevad võimalused inimeste tuvastamiseks on laserkaugusmõõtjad [3], kaamerapildi töötlus erinevate inimtuvastuse algoritmide abil [4], pöördemomendi mõõtmine [5], inimese poolt kantavad elektrilised majakad [6] ning masstoodangus olevad sügavuskaamerad nagu Microsoft Kinect [7], Playstation Move [8], Intel RealSense [9], ASUS Xtion [10] ja SwissRanger [11].

Leap Motion (LM) kontroller on USB-seade, mis on mõeldud kasutamiseks asetatuna laua peale või kinnitades selle virtuaalreaalsuse prillide külge. Kasutades kolme infrapuna valgusdioodi ja kahte kaamerat suudab see kontaktivabalt tuvastada kasutaja käsi ja sõrmi, töövahemikus 25 kuni 600 mm seadmest. LM-kontrollerist tuleneva videovoo põhjal on muuhulgas võimalik tuvastada käte asukohad, liikumiskiirused ja tüüpilisemaid žeste.

Käesoleva bakalaureusetöö eesmärgiks on luua draiver, mis võimaldaks LM-kontrollerit kasutada robootika arendusplatvormil ROS (*Robot Operating System*). Töös antakse ülevaade kaasaegsetest inimtuvastuse meetoditest robootikas (peatükk 2) ning analüüsitakse seniseid lahendusi LM-kontrolleri kasutamiseks ROS arendusplatvormil (peatükk 5). Peatükkides kuus ja seitse tutvustatakse töö raames valminud draiverpaketi nõudeid ja arhitektuuri.

2. Kaasaegsed inimese tuvastuse meetodid robootikas

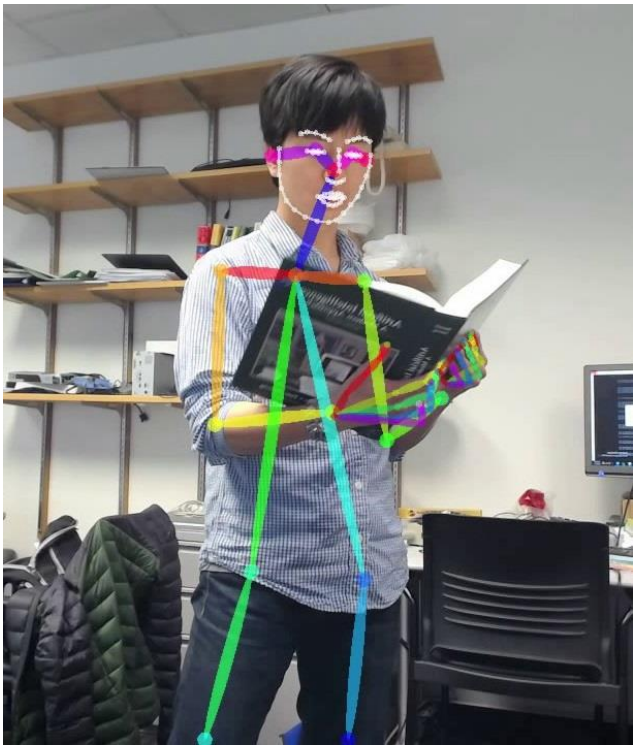
2.1. Sissejuhatus

Täpse informatsiooni hankimine robotit ümbritsevate inimeste tegevuste ja asukoha kohta on üks tähtsamatest ülesannetest robootikas. Osadel juhtudel sobib kui robot on lihtsalt teadlik inimese kohalolekust, kuid tõeliselt inimest abistavate robotit puhul on vajalik inimese või inimeste pidev jälgimine koos nende käitumise tõlgendamisega [12].

Hea inimese tuvastuse meetod on kontaktivaba, universaalne ja laialt kasutatav, kuid igal kasutataval meetodil on omad plussid ja miinused. Alapeatükkides 2.2-2.4 tutvustatakse kolme enamlevinud inimese tuvastamise meetodit robootikas: videopilditöötluse, laserkaugusmõõtjate ning sügavuskaamerate põhjal inimese asukoha ja asendi määramine. Viimasena tutvustatakse alapeatükis 2.5. ainult inimese käte tuvastamist võimaldavat LM-kontrollerit.

2.2. Pilditöötlus

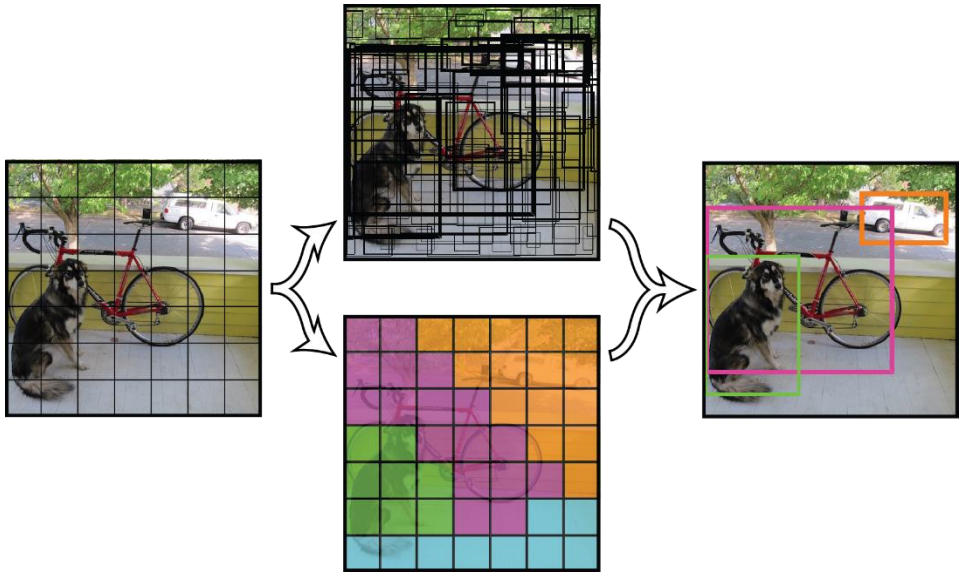
Kaasaskantavate videosalvestusseadmete suure hulga tõttu on huvi pilditöötlusalgoritmide vastu järjest kasvanud. Pilditöötlus, kõige üldisemas mõistes, kujutab endast pildi töötlemist ja analüüsi, et hankida sellest kasulikku informatsiooni.



Joonis 1. *Inimtuvastus kasutades veebikaamerat ja avatud lähtekoodiga OpenPose teeki [13].*

Üldjuhul on inimesel pea, kaks kätt, rindkere ja kaks jalga. Kasutades seda eeldust on võimalik treenida masinõppe algoritme, mis suudavad videokaadritest inimesi tuvastada ning reaalajas jälgida. Üheks selliseks projektiks on OpenPose (joonis 1), mis otsib piltidelt inimeste erinevaid kehaosi ning proovib neid omavahel ühendada üheks terviklikuks mudelskeletiks kasutades tõenäosusfunktsioone [14].

Teine laialdast kasutust leidev vabavaraline lahendus on YOLO (*You Only Look Once*) objektituvastus (joonis 2), mis kasutab närvivõrke, et jaotada töödeldav pilt mitmeteks regioonideks ning annab igale regioonile tõenäosushinnangu. Regioonid, mis on suure tõenäosusväärtusega loetakse tabamusteks [15].

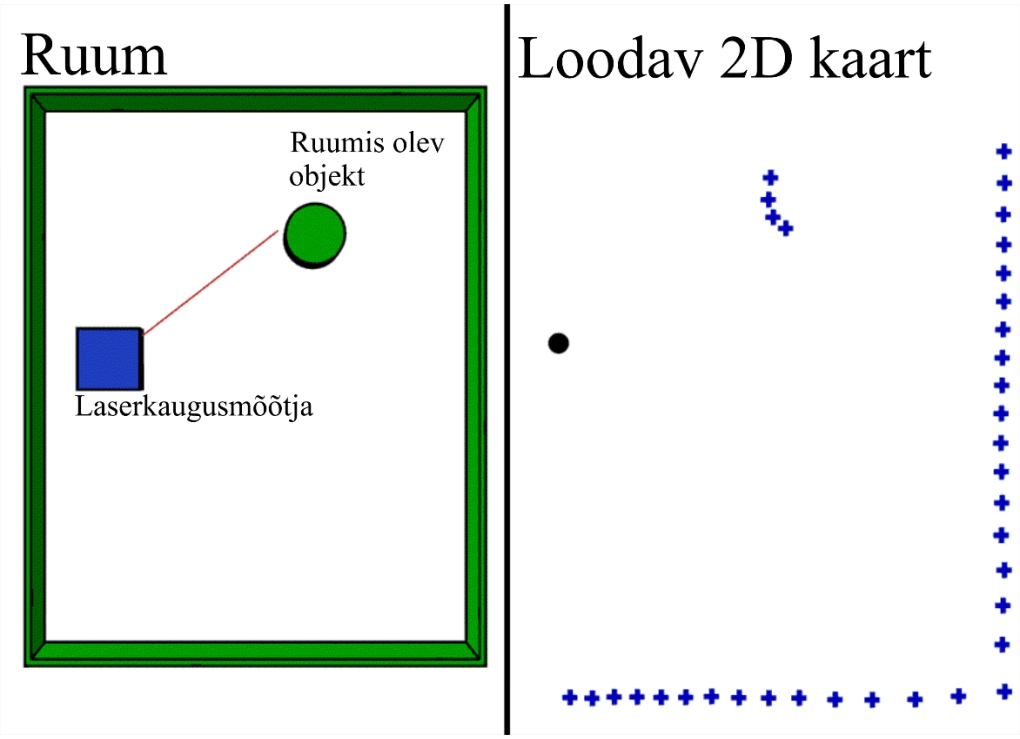


Joonis 2. *YOLO objekti tuvastus algoritmi töö lihtsustatud ülevaade [16]*.

Kuigi pilditöötlus on väga mitmekülgne lahendus ja laialdaselt kasutatav on see ka väga andme- ja töötlusmahukas. Samuti on kaamerapildi kvaliteet väga mõjutatud ümbritsevast valgustasemest, langedes pimeda keskkonna puhul. Viimaks, kasutades masinõppe algoritme, tuleb efektiivse algoritmi loomiseks seda esmalt treenida võimeka riistavara peal kasutades tuhandeid kui mitte kümneid tuhandeid erinevaid näiteid.

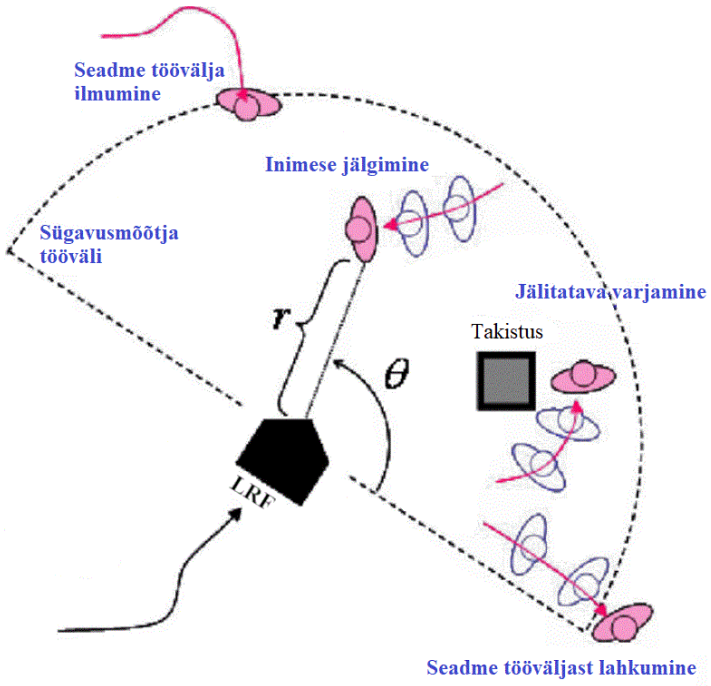
2.3. Laserkaugusmõõtjad

Laserkaugusmõõtja on seade, mis kasutab objekti kauguse mõõtmiseks laserkiirt. Seade kiirgab mõõdetava objekti poole välja laserimpulsi ja mõõdab selle tagasipeegeldumist. Kuna valguskiirus on Maa atmosfääris piisavalt konstante, mõõdetakse aega impulsi algusest kuni peegeldunud impulsi tagasi peegeldumiseni [17]. Selle põhjal arvutatakse laserkiire allika ja mõõdetava keha vaheline kaugus. Kinnitades laserkaugusmõõtja liikuva aluse külge (joonis 3) saab arvutada vahemaid erinevate laseri ette jäänud objektidest ning luua andmetest virtuaalse 2D kaardi [18].



Joonis 3. *Laserkaugusmõõtja konstrueerib ruumist ja selles olevatest objektidest 2D kaardi [18].*

Laserkaugusmõõtja abil saab inimesi reaalajas jälgida, kuid seadmel on mõningasi puudusi (nt vaateväli ja liikuvad osad) ning seda kasutatakse pigem lisaandurina näiteks kaamerale. Siiski on selle abil võimalik reaalajas tuvastada ja jälgida isikuid, otsides inimestele iseloomulikku jalgade edasi-tagasi liikumismustrit [19]. Joonisel 4 on kujutatud stsenaarium, kus inimene siseneb robotseadme tööväljа ja robot jälgib teda järjestikuste sügavusmõõtmiste abil. Kohati on jälitatav erinevate objektide poolt varjatud ning viimaks lahkub inimene seadme tööväljast.



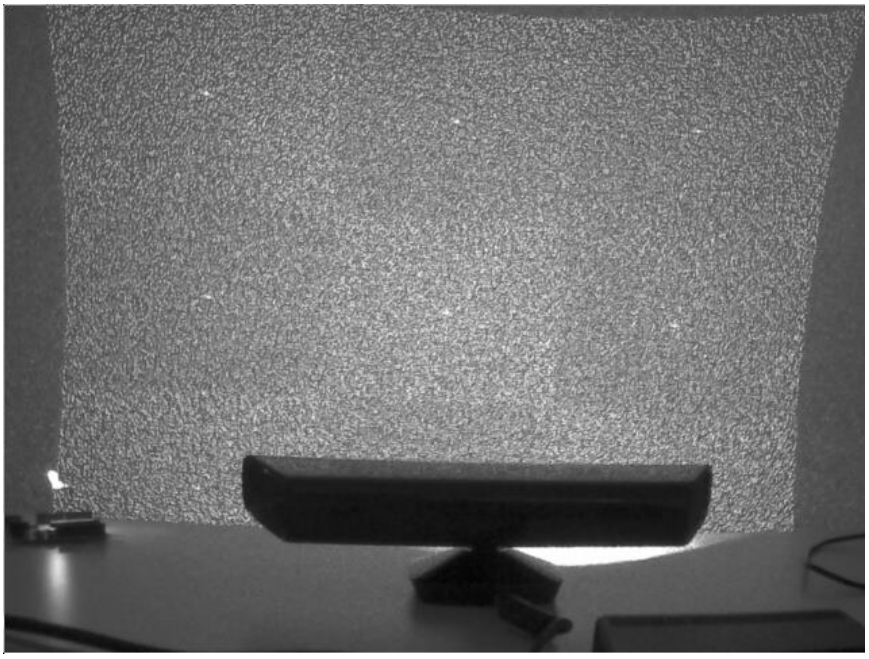
Joonis 4. *Laserkaugusmõõtja (LRF) inimtuvastuse neli etappi [19].*

Anduri suurimaks tugevuseks on selle töökindlus erinevate valgustasemete korral. Seade töötab sama hästi nii öösel kui ka päeval, pilvise või selge taeva korral. Miinusteks on seadme mõõtmistulemuste halvenemine tugeva vihma-, lumesaju või udu korral [20]. Lisaks valguse suure kiiruse tõttu pole seadmetega praktiline mõõta väikeseid vahemaid, sest nõutav diskreetimissagedus muutub väga kõrgeks [21].

2.4. Sügavuskaamerad

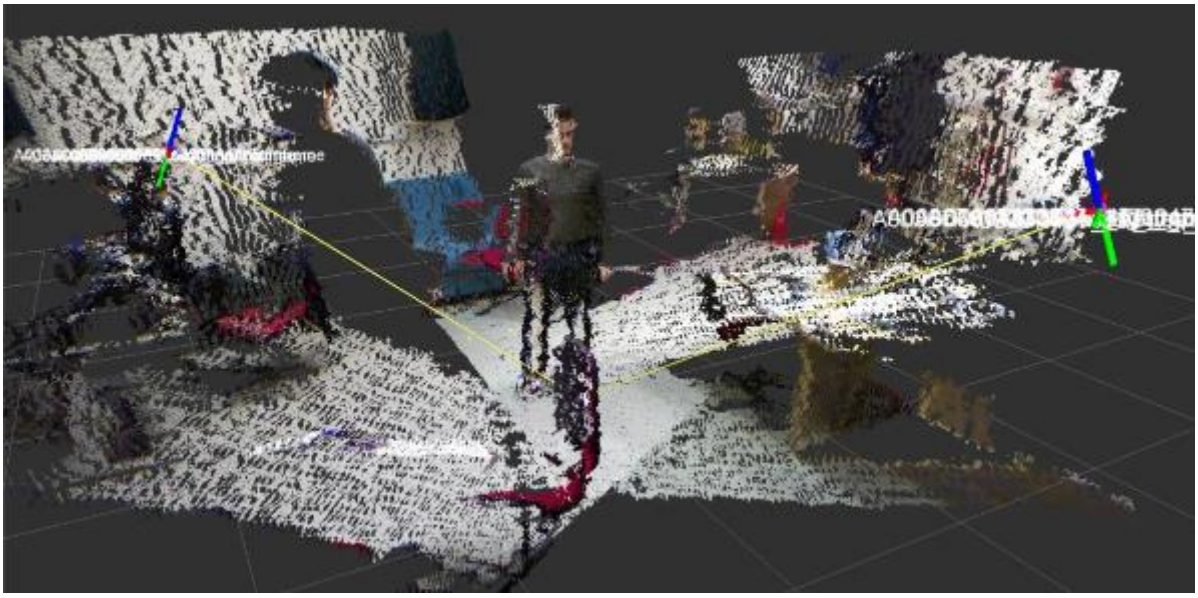
Tavaline kaamera tõlgendab 3D maailma kahemõõtmeliseks kujutiseks, kuid sageli pole 2D pildis töötluse jaoks piisavalt informatsiooni. Taoliselt pildilt on näiteks keeruline aru saada, kumb objekt on lähemal või pikem. Eriti tähtis on objektide kauguse hindamine autonoomsete sõidukite puhul, kus esemete ja inimeste kaugus sõidukist on kriitilise tähtsusega informatsioon.

Sügavuskaamera on seade, mis üldjuhul koosneb kahest kaamerast ja infrapuna kiirgavast komponendist. Infrapuna sensor kiirgab kaamerast kindlaks määratud mustrit (joonis 5), üks RGB kaamera ja üks sügavuskaamera, mis analüüsib mustri moonutust. Seeläbi on võimalik arvutada välja sügavuskaart kasutades erinevaid geomeetria valemeid. Puudujäägiks on meetodi suur vastuvõtlikkus keskkonna heledusele lubades seda kasutada ainult tumedas või siseruumides [22].



Joonis 5. *Kinect sügavuskaamera poolt kiiratav muster [23].*

Teiseks meetodiks on kahe erineva kaamerapildi omavaheliste punktide seostamine. Seadme parem ja vasak kaamera salvestavad kaadreid ning saadavad need pilditöötlusprotsessorile, mis arvutab iga piksli sügavusväärtuse, seostades omavahel punkte, mis esinevad nii paremas kui ka vasakus kaamerapildis. Meetodi suurimaks probleemiks on leida mõlema kaamera piltidelt piisavalt palju selgelt ühised punkte [24].



Joonis 6. *Kahe Kinect kaamera ja OpenPTrack teegi poolt loodud sügavuspilt [25].*

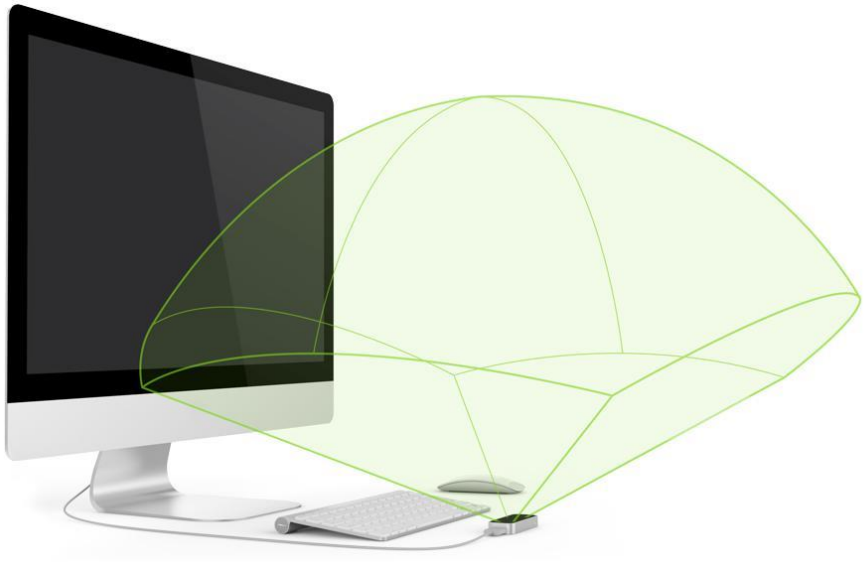
Sügavuskaamerate poolt saadava informatsiooni töötlemiseks on populaarne valik 2013 aastal loodud avatud lähtekoodiga OpenPTrack teek (joonis 6). Projekti eesmärgiks on luua skaleeruv, mitme kaameraga lahendus inimeste jälgimiseks suurtel aladel reaalajas. Kõik individuaalsed seadmed saadavad oma andmed ühte ülemprotsessi, kus need sulandatakse üheks terviklikuks sügavuspildiks [25]. Teek toetab mitmeid masstoodangus olevaid sügavuskaameraid nagu Microsoft Kinect ja Intel RealSense.

2.5. Leap Motion kontroller

Joonis 7. *LM-kontroller igapäevakasutaja töölaual [26] ning OSVR peakomplekti küljes [27].*

LM-kontroller (joonis 7) on seade, mis lubab kasutajal mõjutada liit- või virtuaalreaalsuses olevaid objekte edastades arvutisse videovoogu kasutaja kätest, sõrmedest ja žestidest. Andur on eelkõige mõeldud kasutamiseks virtuaalreaalsuse (VR) prillidega nagu Oculus Rift või HTC Vive, et pakkuda VR programmidesse uut ja intuitiivset sisestusmeetodit. Samas võib seadet kasutada mistahes rakenduses, kus soovitakse tuvastada inimese käte asukohta või käeviipeid.

LM-kontroller kasutab oma tööks kahte kaamerat ja kolme infrapuna valgusdioodi. Kui seade on asetatud lauale, on kõik sensorid suunatud ülespoole 150 kraadise nägemisulatusega (joonis 8), koos efektiivse käte tuvastus vahemikuga 25 kuni 600 mm seadme kohal [28].



Joonis 8. *Leap Motion kontrolleri nägemisulatus [29]*.

LM-kontrolleri arendustarkvara LeapSDK väljastab informatsiooni tuvastatud objektide kohta andmekogumina, mida kutsutakse kaadriks („*frame*“). Iga kaadri objekt sisaldab järjendit jälgitavatest objektidest: käed, sõrmed, tuvastatud žestid ja tegureid, mis kirjeldavad nähtava stseeni liikumist [30].

3. Arendusplatvorm ROS

3.1. ROSi üldkirjeldus

Eelnevas peatükis kirjeldatud meetodid on veelgi kasulikumad kui need muuta laialt kasutatavaks ehk anda igaühele võimalus loodud algoritme oma projektides kasutada. Muutes tarkvara implementatsiooni üldisemaks saavad kasutajad leida viise, kuidas teiste poolt loodud lahendusi teistsugustes olukordades kasutada ilma, et iga uue süsteemi puhul oleks vajadus neid uuesti üles ehitada. Siin tulebki kasuks laialt levinud robootika arendusplatvorm ROS [31].

ROS on avatud lähtekoodiga kogum tööriistu, tarkvarateeke ja kokkuleppeid robotite arendamiseks. ROS raamistiku peamine eesmärk on toetada koodi hõlpsat jagamist ja taaskasutamist, sest loodud algoritme saab rühmitada eraldiseisvateks ROS pakettideks ja seeläbi saavutada keerukate robotsüsteemide modulaarsus [32].

ROSi saab kirjeldada kui meta-operatsioonisüsteemi, mille ülesannete hulka kuulub riistvara abstraktsioonikihi tagamine, madalatasemeline seadmete juhtimine, protsesside vaheline suhtlus ja paketihaldus [32]. Nimest hoolimata, ei asenda ROS arvutis olemasolevat operatsioonisüsteemi, vaid vajab toimimiseks Linux-tüüpi OSi, nt Ubuntu Linux. ROSi tuleb vaadelda kui vahetarkvara, mis peidab riistvara keerukuse arendaja ja seadme vahel. Tänu ROSile saab arendaja keskenduda rohkem kõrgema taseme ülesannete lahendamisele ning ei pea väga palju muretsema riistvara spetsiifiliste sätete seadistamise pärast.

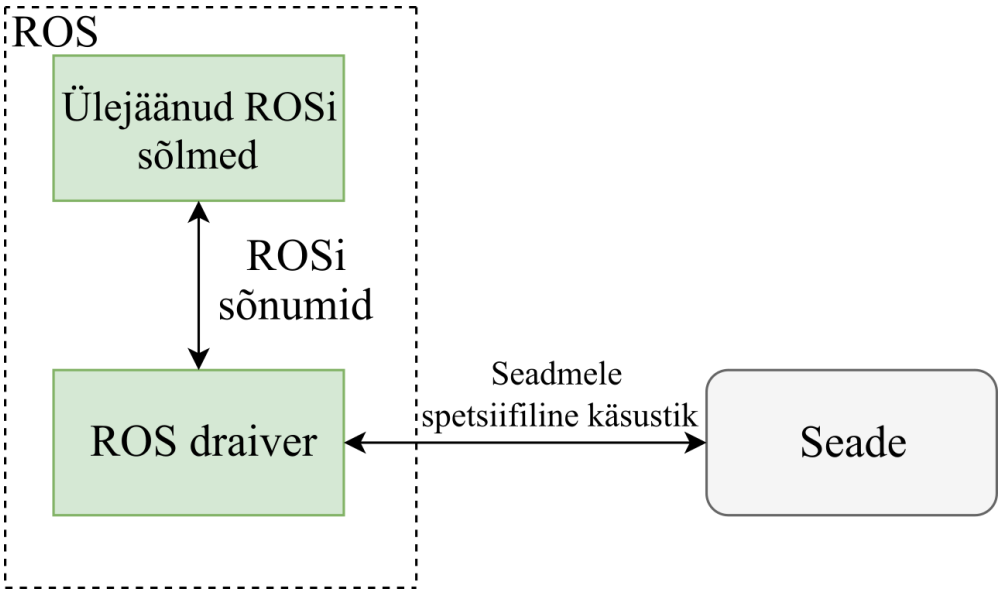
ROSi tööpõhimõte seisneb erinevate iseseisvaid ülesandeid täitvate sõlmede („*node*“) koostöös. Töötavad sõlmed vahetavad omavahel andmeid ehk sõnumeid („*message*“), mis genereeritakse kasutades lihtsaid andmestruktuure .msg tekstifailides, kus on kirjeldatud iga sõnumis oleva välja tüüp [33]. ROS toetab standardseid andmetüüpe nagu täisarv, ujukomaarv, sõne, tõeväärtus ning nendest moodustatud järjendeid. Sõnumifailide poolt defineeritud andmestruktuure kasutatakse lähtekoodi automaatseks genereerimiseks erinevates keeltes.

Sõnumeid edastatakse sõlmede vahel väljaandja ja tellija põhimõttel ning üks sõlm võib samal ajal väljastada ja tellida mitmeid erinevaid sõnumeid erinevate teemade („*topic*“) all. Selleks peab sõlm lihtsalt globaalselt kuulutama, et see annab välja kindlat sõnumit teatud teema all. Peale globaalset teadannet saab iga ROSi sõlm, mis on huvitatud sellest kindlast teemast, teema raames edastatavaid sõnumeid tellida.

Kuna ROSi näol on tegu kogukonna-põhise tarkvaraarendusega, siis on ülimalt tähtis, et loodud lahendused oleksid dokumenteeritud ning lihtsasti kasutatavad. Igal hästi dokumenteeritud ROSi paketil on vikileht (nt https://wiki.ros.org/tf), kus on tarkvara lühikirjeldus, paigaldamis- ja kasutusjuhend, haldajate kontaktandmed ning link lähtekoodi sisaldavale koodihoidlale.

3.2. Draiveripaketi roll ja ülesehitus ROSis

Mistahes seadme ROSi draiveri peamiseks eesmärgiks on muuta see riistavara ROSi kasutajate jaoks ligipääsetavaks. Draiversõlm suhtleb juhitava seadme riistvaraga ja muundab sellele saadetud käsklused seadmele spetsiifilisse käsustikku ning sellelt tulnud vastused omakorda ROS sõnumiteks (joonis 9). Lisaks võib üks draiver toetada mitut sarnast seadet pakkudes nende kontrollimiseks identset liidest. See lubab ROSi sõlmedel kasutada sama tööloogikat mitme eri seadme puhul.



Joonis 9. *Ülevaade ROS draiverisõlme loogikast.*

Hea näide ROS draiverist on Intel RealSense sügavuskaamerate kasutamiseks mõeldud `realsense_camera` pakett [34], mille ROS vikileht:

- kirjeldab tarkvara paigaldamise protseduuri,
- loetleb toetatud sügavuskaamerad,
- loetleb kasutusel olevad ROSi teemad,
- toob näiteid, kuidas muuta seadme sätteid,
- demonstreerib draiveri kasutamiseks vajalikke käske.

4. Töö motivatsioon ja eesmärk

Inimeste tuvastamine on keeruline ülesanne eriti kui on vaja suurt täpsust. ROSi ökosüsteemis on mitmeid pakette inimese käte tuvastamiseks eelkõige Kinect ja muude seadmete baasil, kuid sageli on need koodihoidlad praeguseks hüljatud [35] või loodud lahendused ei paku piisavalt täpsust nagu erinevate käte käelisuse või sõrmede eristamine [36]. LM-kontroller pakuks head lahendust ROSi ökosüsteemis inimeste käte tuvastamiseks kuna seade ongi sellele spetsialiseerunud.

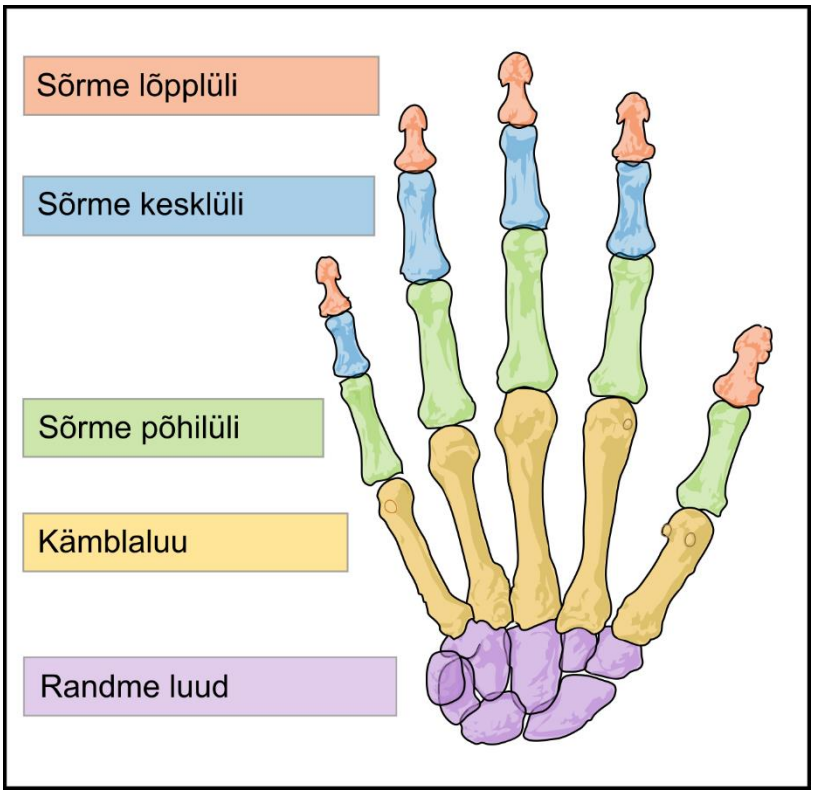
Lõputöö eesmärgiks on luua draiverpakett, mis muudab LM-kontrolleri täieliku võimekuse arendaja jaoks saadavaks, lisada draiver ametlike ROS draiverite koodihoidlasse, luua korrektselt vormistatud vikilehekülg draiveri kasutamiseks ning muuta loodud lahendus paigaldatavaks `sudo apt install ros-<väljalaske nimi>-leap-motion` käsuga.

Aastate jooksul on programmeerijad loonud mitmeid lahendusi LM-kontrolleri ja ROSi integreerimiseks, kuid neil kõikidel esineb olulisi puudusi. Tihti kasutati vaid väikest osa seadme võimekusest, just see väike tükk, mida päevakorral olevas projektis parajasti vaja oli. Seetõttu uuritakse käesoleva lõputöö raames esmalt, mis funktsionaalsus on juba saadaval, selleks analüüsitakse olemasolevaid lahendusi, et võimalusel kombineerida nende funktsionaalsus üheks kergesti hallatavaks terviklikuks ROSi paketiks. Analüüsi käigus dokumenteeritakse iga paketi puudused ja tugevused. Seejärel koostatakse nimekiri nõuetest, mida loodav draiver peab täitma. Loodav pakett peab samaaegselt olema nii lihtsasti kasutatav algajale robootikule kui ka pakkuma maksimaalset LeapSDK tehnilist võimekust keerukate süsteemide inseneridele. Töö lõpptulemusena valmib terviklik draiveripakett LM-kontrolleri integreerimiseks ROSi põhistesse süsteemidesse.

5. Olemasolevate lahenduste analüüs

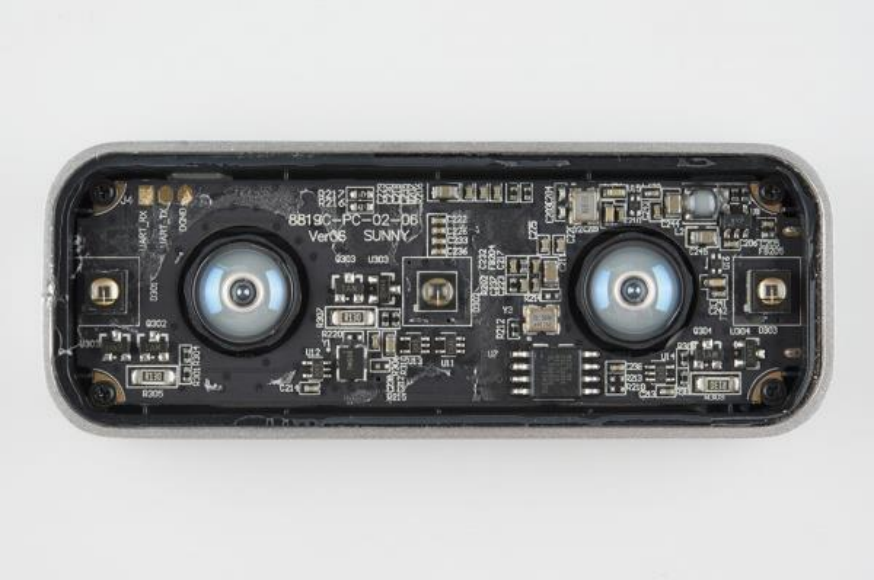
5.1. Käte tuvastamise ja kuvamise põhitõed

Inimkäsi koosneb õlavarrest, küünarvarrest ja labakäest. Labakäsi koosneb omakorda 27 erinevast luust (joonis 10), mis on jaotunud randme, kämbla ja viie sõrme peale. Randme kaheksa luud ühenduvad labakäe viie luuga ning need omakorda iga sõrme kolme erineva sõrmelüliga [37].



Joonis 10. *Labakäe luustik [38].*

Laiemas kasutuses on kaks erinevat lähenemisviisi käte masinjälgimiseks: kaamerapildilt käte tuvastamine ja spetsiaalsed andurkindad [39]. Kinnaste miinuseks on kõrge hind ja nende külge kinnituv riistvara, mis võib piirata käte liikumisvabadust, ent pilditöötlus võib anda suurepäraseid tulemusi juba ühe veebikaamera abil [14]. Kuid kaamerapildi tulemusi mõjutab tausta muutumine, valgustus, jälgitava objekti kaugus kaamerast ja kaamera orientatsioon [39].



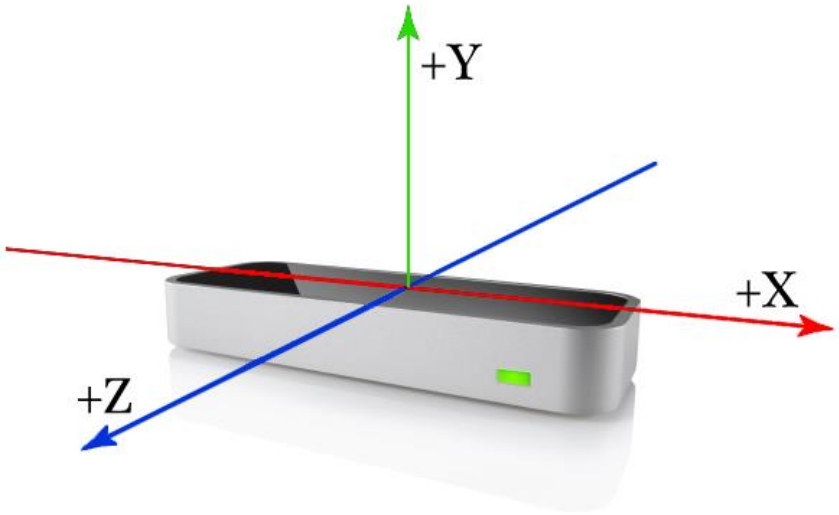
Joonis 11. *LM-kontrolleri kaks optilist sensorit [40]*.

LM-kontroller kasutab kahte optilist sensorit (joonis 11), et võtta vastu kolme infrapuna valgusdioodi poolt kiiratud ja seejärel kätelt tagasi peegeldunud valgust ning tekitab sellest kaamerapildi. Järgnevalt edastatakse andmed USB-liidese kaudu arvutisse töötlemiseks. Seal toimub taustapildis olevate objektide nagu inimese pea ning taustavalguse vastu kompenseerimine [28]. Järgnevalt analüüsitakse iga saabunud kaadrit, et moodustada 3D representatsioon seadme nägemisväljas asuvatest kätest.

Analüüsi käigus kasutatakse mitmeid algoritme, mis tõlgendavad 3D andmeid ning järeldavad nendest peidetud käeosade kõige tõenäolisema asukoha. Iga kaadri töötlemise käigus proovitakse kindlaks määrata käte asukoht koos kõikide erinevate sõrmede tüüpide ja asukohaga kolmemõõtmelises ruumis [28]. Kogu tuvastatud infole pääseb kasutaja ligi kasutades LM-kontrolleri rakendusliidest (API).

5.2. LeapSDK võimekus Linuxi platvormidel

Leap Motion Inc pakub enda seadmele tarkvara arendamiseks kahte erinevat SDK versiooni. Esimene neist on aktiivselt arendatav Leap Motion Orion (SDK versioon 3.2.1), mis toimib ainult Windows operatsioonisüsteemil ja on mõeldud kasutamiseks Unity 3D mängumootoriga [41]. Kuna hetkel saab ROSi kasutada ainult Linux-tüüpi operatsioonisüsteemidel, siis tuli käesolevas töös kasutada arendustarkvara LeapSDK (versioon 2.3.1), mille ametlik arendus ja tugi on lõppenud. LeapSDK v2.3.1 aga toetab nii Mac ja Linux operatsioonisüsteeme ning suudab muuhulgas tuvastada ka käes hoitud pikka silindrikujulist tööriista nagu pliiatsid. Tööriistade tuvastuse funktsionaalsus on uuemas Orion SDK-s eemaldatud.

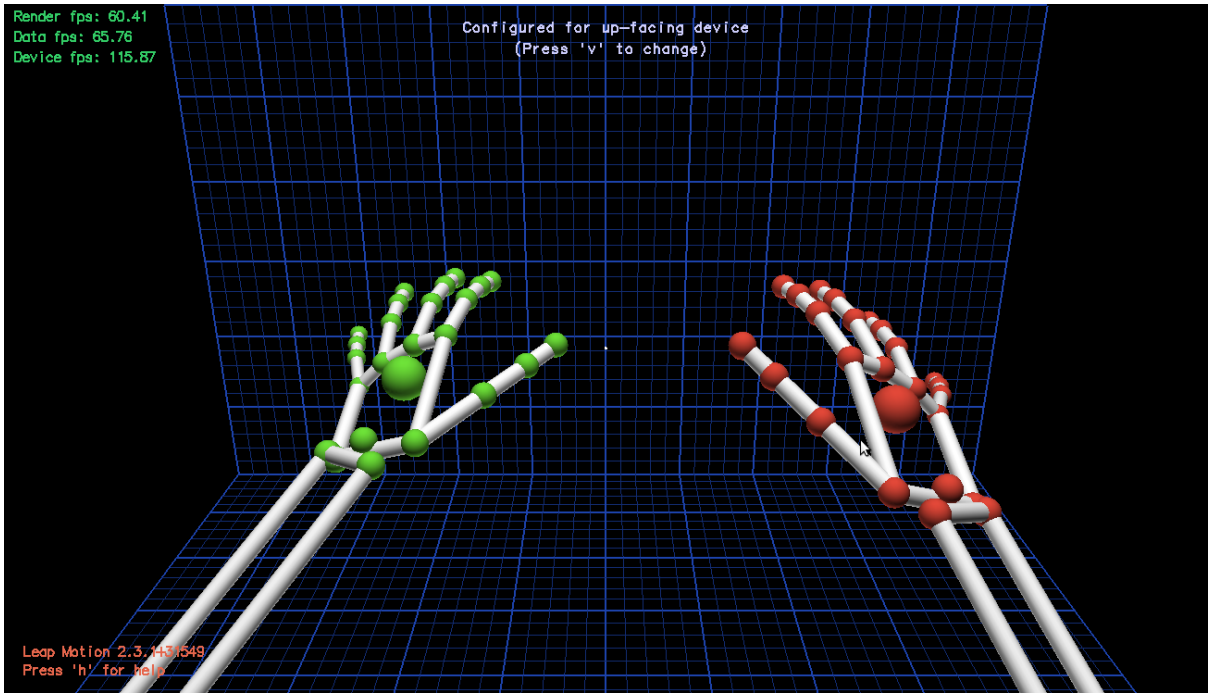


Joonis 12. *Leap Motion kontrolleri poolt kasutatav koordinaatsüsteem [42].*

LeapSDK kasutab parema käe reeglit järgivat Descartes'i koordinaatsüsteemi (lad „*Cartesius*"), kus nullpunkt on defineeritud kui kontrolleri pealmise pinna keskpunkt (joonis 12). Lisaks võimaldab LeapSDK saada järgnevat infot:

1. Käte ja sõrmede asukoht
    a. informatsiooni sõrmede laiuse ja pikkuse kohta
    b. iga individuaalse sõrmelüli asukoht
    c. käelaba orientatsiooni seadme suhtes
2. Käe asukoha muutus erinevate kaadrite vahel
    a. peopesa asukoht, orientatsioon, mõtteline raadius ja joonkiirus
3. Põhiliste käeviibete (nt vajutamine, torkamine ja viipamine) tuvastus
4. Töötlemata kaamerapildi edastus.

Ühtlasi sisaldab LeapSDK graafilist silumistööriista *Diagnostic Visualizer* (joonis 13), mis kuvab seadme poolt tuvastatud käsi ning täiendavaid diagnostilisi andmeid.



Joonis 13. *Kuvatõmmis LeapSDK graafilisest töövahendist Diagnostic Visualizer.*

Juhul kui osa käest pole seadmele nähtav, kasutab tarkvara käe nähtavaid osi, enda sisemist käe mudelit, tõenäosusfunktsioone, käe eelnevaid teadaolevaid positsioone ning arvutab välja labakäe mitte nähtavate osade kõige tõenäolisema asukoha [28], [42]. Käte tuvastuse hetkest käte kujutamiseni arvutiekraanil võtab aega keskmiselt 30 ms, kuid see arv võib olla suurem või väiksem olenevalt kasutatavast USB liidesest, süsteemi graafikakaardist ja üldisest jõudlusest [43], [44].

Seade pakub kaamerapildi edastust sensoritelt kaadrisagedusega üle 100 kaadri sekundis [43]. Töötlemata kaadrid edastatakse koos infrapuna intensiivsuse ja kalibratsiooni andmetega [42]. Viimaseid kasutatakse, et korrigeerida objektiivi moonutust.

5.3. ROS leap_motion pakett

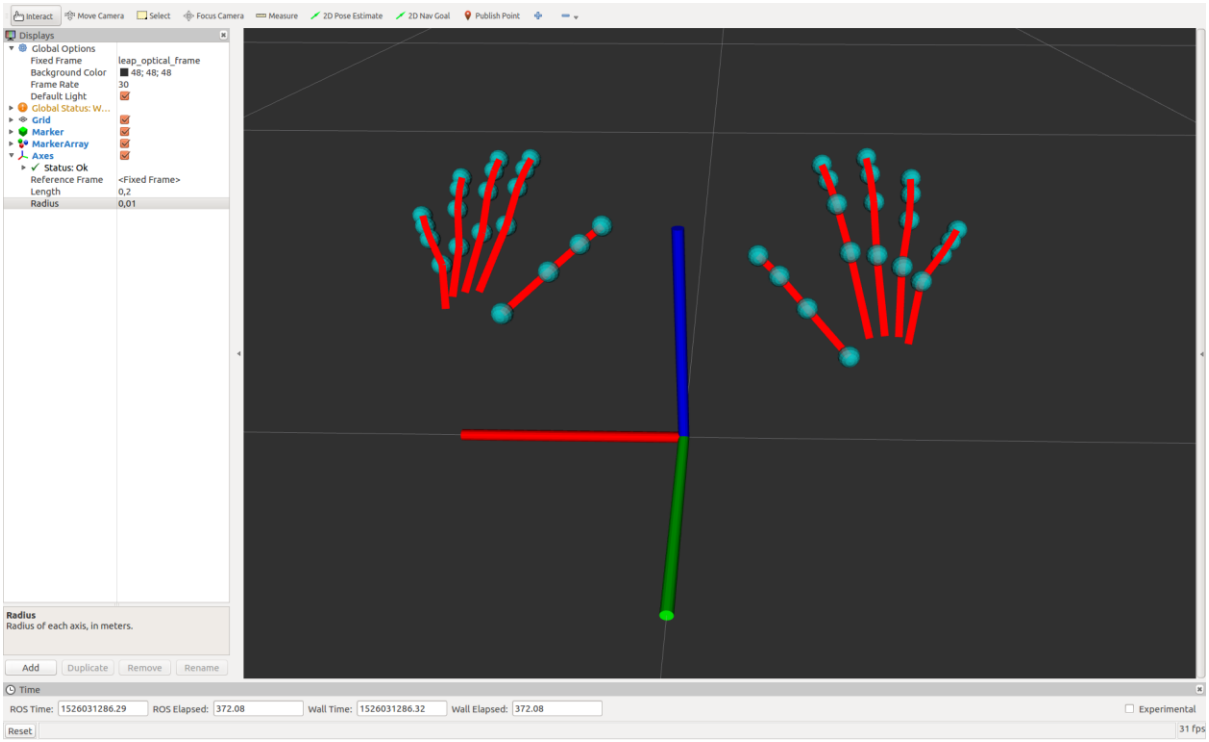
ROSi ametlikus koodihoidlate kogus eksisteerib LM-kontrollerile draiveripakett `leap_motion`, kuid seda tarkvara pole alates 2014. aastast märkimisväärselt arendatud ning koodihoidlal puudub aktiivne hooldaja. Kogu lähtekood on segu Python ja C++ failidest, mis sisaldavad ohtralt välja kommenteeritud koodi, omavahelist funktsionaalset ülekattuvust ning pakuvad ROS arendajatele põhjendamatult piiratud osa LeapSDK võimekusest.

Antud bakalaureusetöö kirjutamise hetkel on leap_motion paketi ametlikul vikilehekülje (https://wiki.ros.org/leap_motion) toodud draiveri lühikirjeldus, loetletud tähtsam funktsionaalsus, kirjeldatud paketis olevate sõlmede eesmärke ning paigaldusjuhend. Paigaldusjuhend kirjeldab lähtekoodi käivitamiseks vajalikke samme ja draiveri installeerimist binaarkujul kasutades Ubuntu Linuxi paketihaldurit käsuga „`sudo apt-get install ros-<väljalaske nimi>-leap-motion`“.

Vikilehekülje väitel võimaldab draiver informatsiooni edastust ainult ühe käe peopesa positsiooni, orientatsiooni ning käe suunavektori kohta 3D ruumis; sõrmelülide algus ja lõpp asukohta ja kaamerapildi edastust seadmest. Draiveri koodihoidla funktsionaalsus põhineb arendajate poolt loodud ROS sõnumitele, mis täideti koordinaate sisaldavate järjenditega või kasutades ROS spetsiifilisi tüüpe nagu „*geometry_msgs/Point*“ ja „*sensor_msgs/Image*“. LeapSDK võimekust arvesse võttes, ei võimaldanud draiver järgnevat informatsiooni edastamist ROSis:

- Žestide tuvastamine
- Tuvastatud käte, sõrmede, žestide arvu.
- Tuvastatud objekide ID nende jälgimiseks kaadrite vahel
- Aega, mil tuvastatud objekti on jälgitud
- Käe haaramis- ja näpistustugevus
- Käte, sõrmede, sõrmelülide pikkust, laiust
- LeapSDK kaadrisagedust

Lähtekoodi lähemal uurimisel ilmnes ka dokumenteerimata võimalus visualiseerida mõlema käe positsiooni ROSi 3D visualisatsioonipaketis RViz (joonis 14). Tarkvara autor oli kasutanud RViz visuaaltähist joon, et kujutada erinevaid sõrmelülisid ning kera, et tähistada iga liigest. Hoolimata taolise graafilise esitluse algelisusest, oli funktsionaalsus piisav, et anda kasutajale arusaam tuvastatud käte asendist. Ent sedavõrd olulise funktsionaalsuse dokumenteerimata jätmine on tõsine puudujääk, eriti esmakordsele kasutajale, kes seda tõenäoliselt kõige rohkem vajaks.



Joonis 14. *Kuvatõmmis olemasoleva draiveripaketi leap_motion dokumenteerimata visualiseerimisvõimekusest.*

Käesoleva töö autoril tuli graafilise kujutamise funktsionaalsuse avastamiseks uurida kõiki lähtekoodi faile ning aru saada, mis võimalusi igaüks neist pakub. Kogu varasem kood oli ka suuresti kommenteerimata ning visualisatsiooni võimaldanud koodifaili nimi, leap_hands.cpp, ei viidanud kuidagi selle tegelikule otstarbele. Peale sõlme käivitamist tuli veel käivitada RViz ning seda kasutaja poolt korrektselt seadistada. Alles peale seda oli võimalik näha pakutavat visualiseerimise võimalust.

5.4. Alternatiivsed LM-kontrolleri draiveripaketid

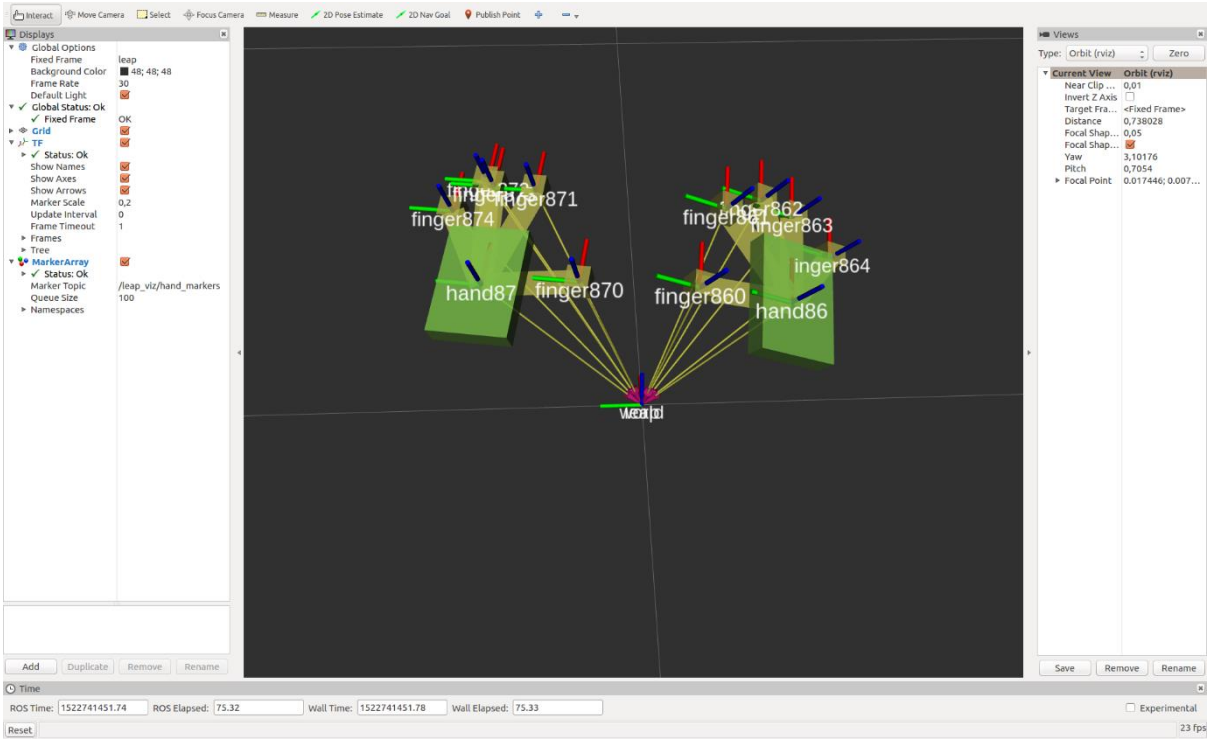
Lisaks ametliku draiveri analüüsile, sai läbi viidud otsing ühes populaarseimas avalikus koodihoidlate keskkonnas GitHub, kus muuhulgas talletatakse ka ROSi erinevate väljalasete lähtekoodi. Uuring teostati märksõnadega „ROS leap“ ning see tagastas 22 tarkvarahoidlat, millest 10 sisaldas tarkvara loodud programmeerimiskeeles C++ ja 6 Pythonis. Ülejäänud koodihoidlad olid tühjad või kasutasid muid vähemlevinud programmeerimiskeeli.

Suurem osa leitud koodihoidlatest oli erinevate arendajate poolt loodud lahendused nende spetsiifilistele probleemidele. Näiteks anduri poolt tuvastatud käe või pliiatsi taolise tööriista kiiruse, nurkkiiruse ja tipu positsiooni kasutamine roboti KUKA youBot manipulaatori kaugjuhtimiseks [45], LM-kontrollerist töötlemata kaamerapildi hõivamine [46] või AR droonide asukoha kontrollimiseks alustatud, kuid kiiresti hüljatud koodihoidla [47]. Osa leitud lahendustest oli loodud erinevatel ühe või kahe päeva pikkustel Leap Motion 3D mängude töötubades [48].

Lisaks ametlikule `leap_motion` paketile, väärib esiletõstmist 2 leitud lahendust:

1. Texase Ülikooli tuuma- ja rakendusrobootika uurimisgrupi koodihoidlas paiknev `leap_motion_controller` [49], kus muuhulgas kasutati käte positsiooni stabiliseerimiseks teise järgu Butterworth madalapääsfiltrit [50].
2. GitHub kasutaja juancamilog koodihoidlas asuv `leap_client` [51]. Antud lahendus võimaldas seadmelt saadud andmeid graafiliselt kujutada (joonis 15) ROSi visualiseerimistöövahendiga RViz. Kood kasutas visualiseerimiseks kõikide sõrmede algus- ja lõpp-punkte, mitte iga sõrmelüli.

Siiski esineb mõlemal lahendusel ka puudusi, nt `leap_motion_controller` ei võimalda käte asendi graafilist visualiseerimist – funktsionaalsust, mis on abiks esmakasutajatele. `leap_client` võimaldab küll RVizi abil visualiseerimist, kuid valitud graafilised elemendid ei võimaldanud nt sõrmede painutamisel korrektset kujutamist.



Joonis 15. *Kuvatõmmis leap_client visualisatsiooni võimekusest.*

6. Loodava draiveripaketi nõuete kirjeldus

Bakalaureusetöö raames valminud draiveripaketi eesmärgiks oli luua ühtne ROSi draiveripakett, mis võtaks õppust olemasolevast, ent puuduliku funktsionaalsuse ja dokumentatsiooniga lahendusest. Seejärel luua selle põhjal uus tarkvara ja dokumentatsioon, millest oleks ROSi kogukonnale kasu. Lähtuvalt LeapSDK funktsionaalsusest ning varasematest lahendustest, sõnastati loodavale LM-kontrolleri draiveripaketile nõuded. Järgnevalt on need välja toodud ja grupeeritud olulisuse põhjal.

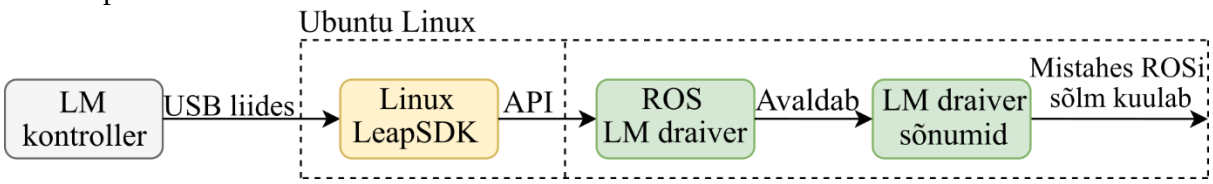
1. Loodav ROSi LM-kontrolleri draiver peab võimaldama informatsiooni edastust kasutaja käte kohta.
    a. Käte asukoht ja käelisus (e parem või vasak)
    b. Sõrmede asukohad ja nimetused
    c. Sõrmelülide asukohad ja nimetused
    d. Tuvastatud käte, sõrmede ja žestide arv
2. Kasutajal peab olema võimalik kontrollida, kas LM-kontroller on tuvastanud žeste.
    a. Ühe käe sirgjooneline liikumine, sõrm välja sirutatud
    b. Ühe sõrme ringikujuline liikumine
    c. Väljasirutatud sõrmega LM-kontrolleri z-teljes (joonis 12) edasi-tagasi liikumine ehk torkamine
    d. Väljasirutatud sõrmega y-teljes üles-alla liikumine ehk vajutamine
3. Infoedastus peab toimuma kasutades selgelt arusaadavaid teemade ja sõnumite nimetusi.
    a. ROS sõnumeid edastatakse teema „*leap_device*” või „*leap_filtered*“ all
4. Kasutajamugavus
    a. Käte asendi visualiseerimine kasutades RVizi
    b. Roslaunch-tüüpi kiirkäivitusskriptid, mille võimaldavad ühe käsuga draiveripaketti käitada
    c. Kõik draiveri pakutavad võimalused on dokumenteeritud
    d. Lähtekood on kommenteeritud ROS C++ stiilinõuete [52] kohaselt
5. ROS distributsioonid ja LM kontrolleri erinevad variatsioonid
    a. Loodav draiver toetab ROS distributsiooni *Kinetic Kame* ja uuemaid väljalaskeid
6. Lisaväärtust pakkuvad võimalused
    a. LM-kontrolleri töötlemata kaamerapildi edastamine
    b. Käte positsiooni koordinaatide filtreerimine
    c. Paketi paigaldamine binaarkujul kasutades Ubuntu Linuxi paketihaldurit käsuga `sudo apt-get install ros-<väljalaske nimi>-leap-motion`
    d. Draiveripaketi paigaldamise ja kasutamise juhend on ROS vikileheküljel

7. Uus draiveripakett: lmc_ros_driver

7.1. Arhitektuur ja disain

Kuna aktiivse kasutaja toega SDK v3.2.1 nimega Leap Motion Orion töötab ainult Windows operatsioonisüsteemil, mida ROS ei toeta, siis tuli käesolevas töös kasutada arendustarkvara LeapSDK versiooni 2.3.1, millele enam ametlikku tuge ei pakuta. SDK sai paigaldatud Ubuntu Linuxi versioonile 16.04, Xenial Xerus, mis valiti laialdase leviku ja hea ROSi toe tõttu.

Bakalaureusetöö valmimise ajal oli valida ROS Kinetic Kame või ROS Lunar Loggerhead väljalasete („*distribution*“) vahel. Esimene neist on välja antud 23. mail 2016 LTS viieaastase elueaga ning viimane normaalse kaheaastase elueaga väljalase. Põhjusel, et ROS Kinetic Kame on toetatud 2021. aasta aprillini, aga Lunar Loggerheadi tugi lõppeb juba 2019. a. mais, valiti arendusplatvormiks ROS Kinetic Kame.



Joonis 16. *Üldine ülevaade draiveri andmevoost.*

Draiveri töö kõrgetasemelise ülevaatena (joonis 16) on LM-kontrollerist kaamerapildi vastuvõtmine LeapSDK poolt üle USB liidese Linux-tüüpi OSi. Leap Motion tarkvara töötleb saadud kaadrit erinevate algoritmidega ning proovib pildi pealt tuvastada käsi. ROSi draiveripakett kasutab LeapSDK APIt, et küsida informatsiooni iga saabunud kaadri kohta ning väljastab seda erinevate teemade all ROSi sõnumitena.

LMC ROS draiveri puhul on tegu ainult kuulutava draiveriga ehk draiver edastab erinevatele teemadele sobivad ROSi sõnumeid, aga teiste sõlmede sõnumeid see ei jälgi. Uue draiveri aluseks võeti juancamilogi `leap_client`, Texase Ülikooli tuuma- ja rakendusrobootika uurimisgrupi `leap_motion_controller` ja ROSi ametlik `leap_motion` pakett, mille eeskujul hakati konstrueerima uut lahendust. Esimesest oli võimalik üle võtta andmete graafiline kujutamine ning seda täiendada. Texase ülikooli lahendusest võeti aluseks madalpääs filter, mida muudeti sobivaks individuaalsete sõrmelülide positsiooni filtreerimiseks. Viimaks, oli võimalik ROSi ametlikust draiver paketist üle tuua töötlemata kaamerapildi edastusega tegeleva sõlme lähtekood ning viia see vastavusse ROSi stiilinõuetega [52]. Kõik muu draiveri funktsionaalsus loodi käesoleva lõputöö autori poolt.

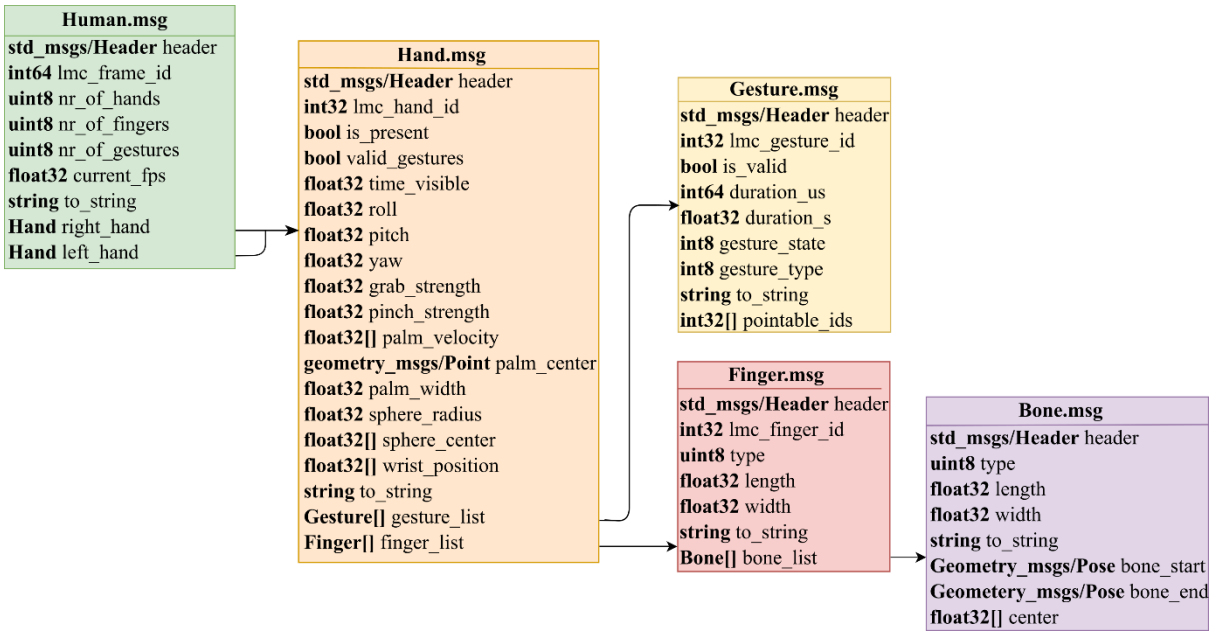
7.2. lmc_ros_driver paketi sõnumid

Sõnumite roll on edastada seadmelt tulnud infot võimalikult standardsel moel ehk kasutades maksimaalselt juba ROSi kogukonna poolt heaks kiidetud sõnumitüüpe [53]. Kõik loodavad

uued sõnumid võiksid ideaalis olla universaalsed ehk kasutatavad ka teistes lahendustes. Seetõttu oli draiveri loomisel üheks alameesmärgiks luua prototüüp-sõnumid laiemaks inimeste esitamiseks ROSis, mitte aga ainult LM-kontrolleri spetsiifilised sõnumid.

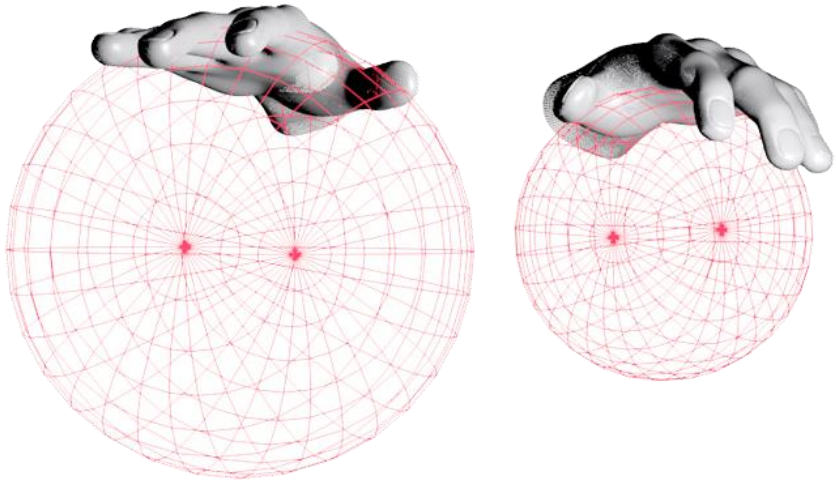
Seega loodi hierarhiline ROS sõnumite struktuur, kus hierarhia kõrgeimal tasemel on Human.msg (joonis 17). Kui LeapSDK API annab infot kaadri-põhiselt, siis LMC ROS draiveri väljundis on see muudetud inimese-keskseks. Kuna LM-kontroller on mõeldud individuaalse kasutaja käte jälgimiseks, siis hetkel on Human-tüüpi sõnumis implementeeritud kuni 2 kätt: parem ja vasak. Sõnum sisaldab endas LM-kontrolleri poolt saadetud informatsiooni ühe kaadri kohta, mida kontroller jäädvustas.

Välja antav sõnum (*lmc_ros_driver/Human*) sisaldab lisaks parema ja vasaku käe infole päist (*std_msgs/Header*) ning tuvastatud käte, sõrmede ja žestide arvu.



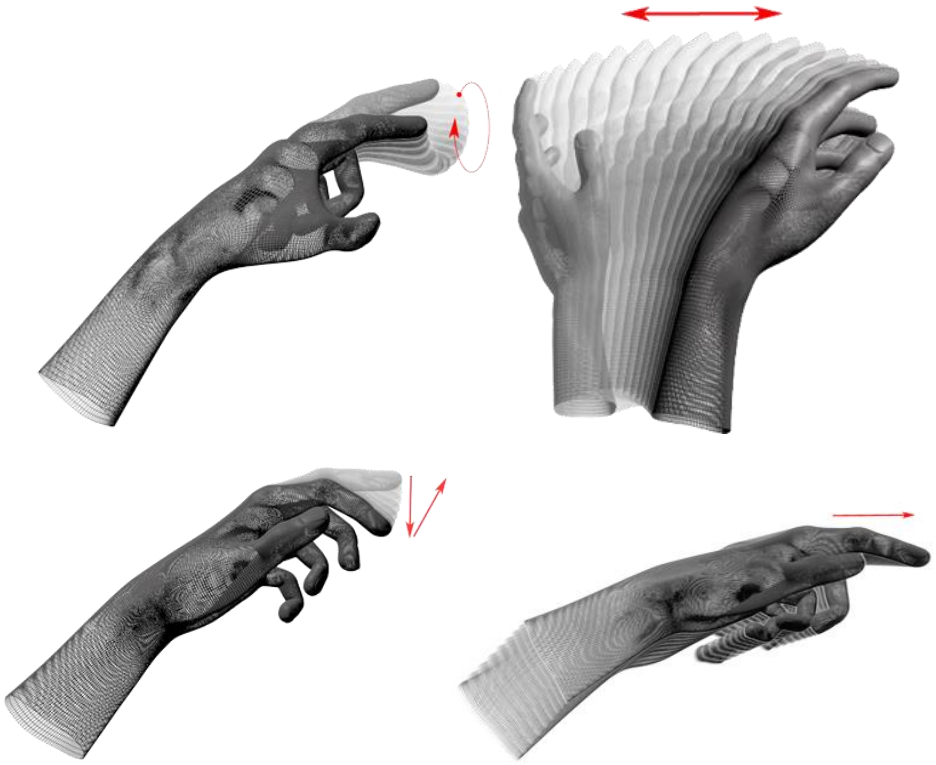
Joonis 17. *Draiveri poolt edastatavate sõnumite hierarhia.*

Hand.msg sisaldab päist, LM-kontrolleri poolt määratud käe identifikaatorit, millega on võimalik üht kätt mitme kaadri vältel jälgida, aega, mille vältel on kätt jälgitud sekundites, ning käe orientatsiooni. Lisaks on võimalik saada infot käe haaramistugevuse kohta, see tähendab nurka käe nelja sõrme ja peopesa vahel vahemikus 0-π radiaani. Avatud käe tulemuseks on 0 rad ning käsi rusikas žesti tuvastamisel on haaramistugevus π radiaani. Näpistustugevuse tööpõhimõte on analoogne, kuid see kirjeldab pöidla ja nimetissõrme vahelist nurka.



Joonis 18. *Käe peopessa mahtuv virtuaalne kera [54].*

Lisaks on võimalik saada informatsiooni virtuaalse sfääri (joonis 18) keskpunkti, raadiuse kohta, mis mahub tuvastatud käe kumerusse. Kera on asetatud kätte nii nagu see oleks käe poolt hoitav pall. Seetõttu väheneb sfääri raadius, kui sõrmed on keerdunud rusikasse. Sõnumi viimasteks väljadeks on käe peopesa hinnatav liikumiskiirus kaadrite vahel (ühik m/s), selle laius, pikkus, asukoht 3D ruumis kontrolleri suhtes meetrites. ning väljad selle käega seonduvate žestide ja sõrmede sõnumite poole.



Joonis 19. *LM-kontrolleri poolt tuvastatavad eelprogrammeeritud žestid [42]. Nimetused alustades ülevalt vasakul: sõrme ringikujuline liikumine, viipamine, vajutamine ja torkamine*

Gesture.msg sisaldab päist, tuvastatud žesti tüüpi, mille abil saab kontrollida, missugust eelprogrammeeritud žesti (joonis 19) on seade tuvastanud, aega selle tuvastuse algusest mikrosekundites ja sekundites, koos selle tõenäolisema olekuga ehk, kas see on just alanud,

pidevas jätkuvas liikumises või hakanud lõppema. Viimaks on võimalik saada iga sellega seotud sõrme ID *pointable_ids* järjendist.

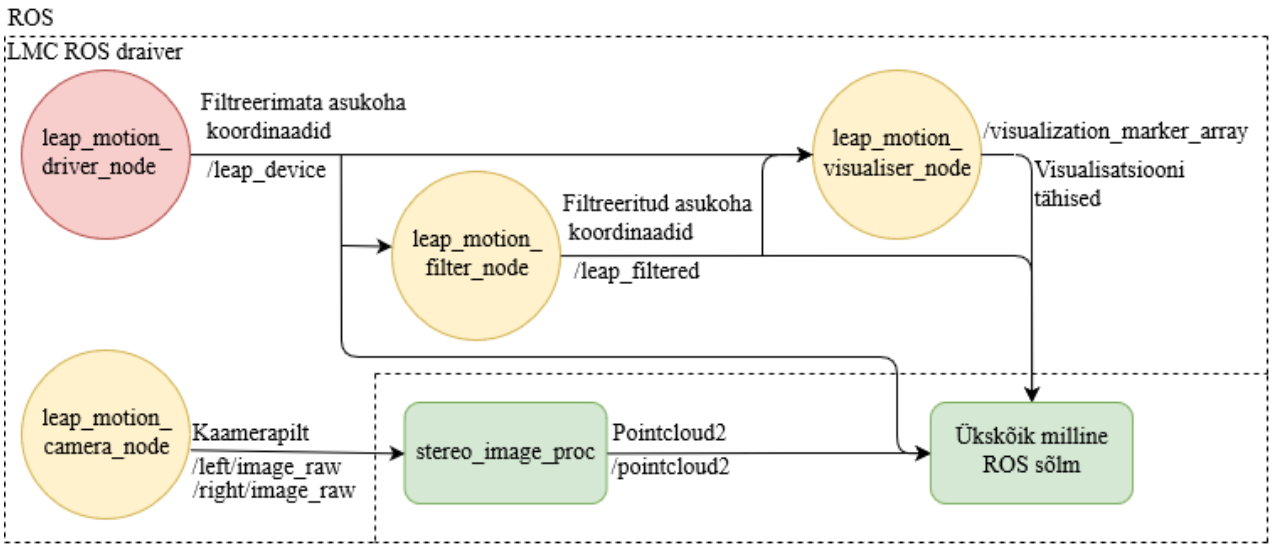
Finger.msg tähtsamateks osadeks on päis, LM-kontrolleri poolt määratud ID sõrme jälgimiseks mitme kaadri vältel ning kaheksabitine märgita täisarv, mis on igal sõrmetüübil erinev. Viimase abil on võimalik teha vahet näiteks pöidla ja nimetissõrme vahel. Lisaks on võimalik veel pärida sõrme laiuse, pikkuse kohta ja väli sõrme moodustavate Bone.msg järjendi poole.

Bone.msg on LM draiveri sõnumite hierarhia madalaim tase sisaldades päist, märgita täisarvu sõrmelüli tüübi määramiseks, infot lüli pikkuse, laiuse, algus-, kesk- ja lõpp punkti kohta 3D ruumis. Asukoha edasi andmiseks ruumis on kasutatud ROSi kogukonna poolt eeldefineeritud „*geometry_msgs/Pose*“ sõnumit, mille abil saab edastada positsiooni XYZ koordinaatidega ja orientatsioon kvaternioniga.

7.3. lmc_ros_driver pakett

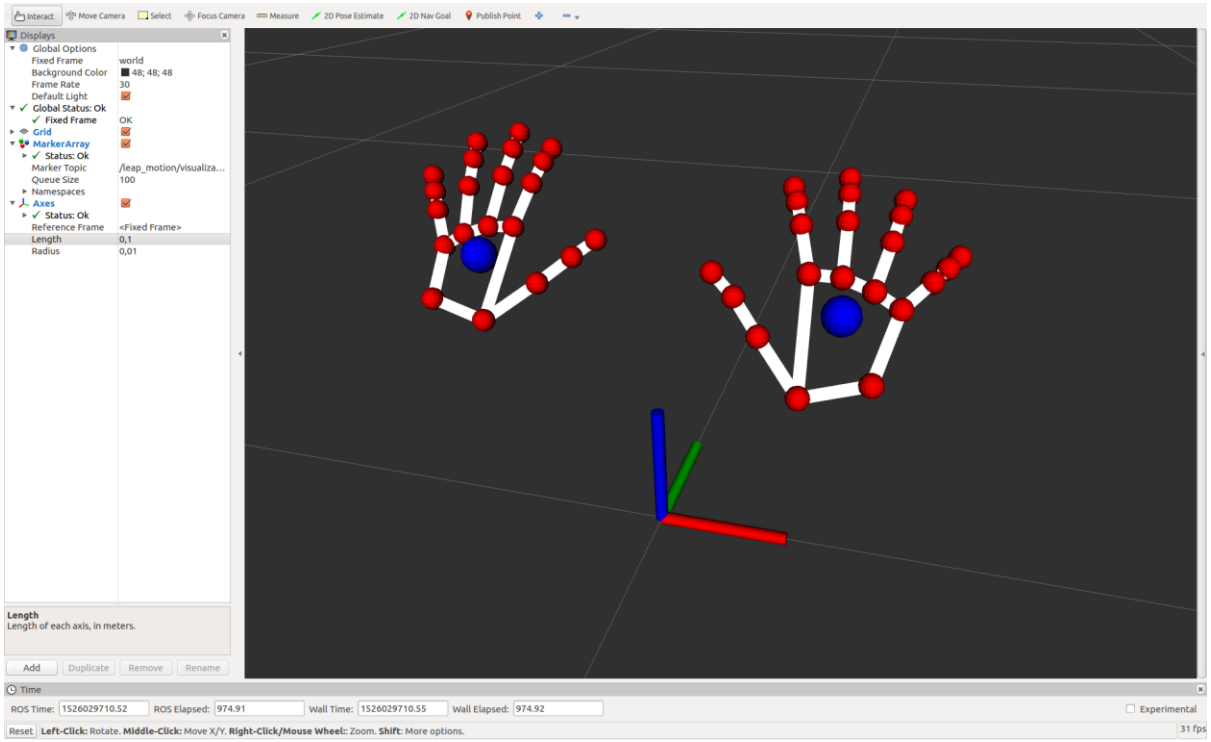
Kuigi ROS toetab tarkvara arenguks mitmeid programmeerimiskeeli nagu C++, Python, Lisp koos eksperimentaalsete teekidega Javas ja Luas, sai loodava draiveri programmeerimiskeeleks valitud C++. Keele tugevateks külgedeks on selle paindlikkus, hea dokumentatsioon, suur kogukond ja ulatuslik levik. Masinkoodi kompileerimine suurendab koodi efektiivsust riistvaral. Enamik ROS draivereid on samuti C++ kirjutatud ning see näis loomuliku valikuna.

Põhifunktsionaalsuse pakkujaks loodud draiveripaketis on *leap_motion_driver_node* (joonis 20). Selle ROS sõlme tööeesmärgiks on suhtlus LM-kontrolleri rakendustarkvaraga ning ilma seda käivitamata pole võimalik draiverit kasutada. Draiverisõlm käitub vahelülina kasutades LeapSDK poolt pakutavat APIt, et töödelda informatsiooni värskeima saabunud kaadri kohta. Seejärel kasutab sõlm saadud informatsiooni, et moodustada sellest `lmc_ros_driver` sõnumid ning väljastab need */leap_device* teema all. Lisaks kontrollib sõlm metainfo (andmed andmete kohta) logimist ROS_INFO abil, LM-kontrolleri poolt ära tuntavate žestide sisse- ja väljalülitamist, `lmc_ros_driver` paketi draiverisõlme andmeedastuse teema pealkirja.



Joonis 20. *Loodud draiveri sõlmede töö loogiline ülevaade. Punaselt on märgitud draiveri tööks hädavajalik sõlm, kollaselt valikulised ning roheliselt draiveri välised sõlmed.*

Draiveris oleva *leap_motion_visualiser_node* ülesandeks on Human.msg põhjal käe kujutise loomine, mida saaks RVizi abil visualiseerida. Kui sõnum näitab, et LM-kontroller on tuvastanud käe, siis luuakse järjend visuaalsetest tähistest („*visualization_msgs/Marker*“), mida suudab graafiliselt esitada RViz. Käte graafiliseks kujutamiseks (joonis 21) loodud lahendus imiteerib võimalikult täpselt LeapSDK töövahendit Dignostic Visualizer (joonis 13) ja seetõttu koosneb käsi valgetest joontest, punastest keradest ning ühest suuremast sinisest kerast, mis tähistab peopesa keskpunkti.

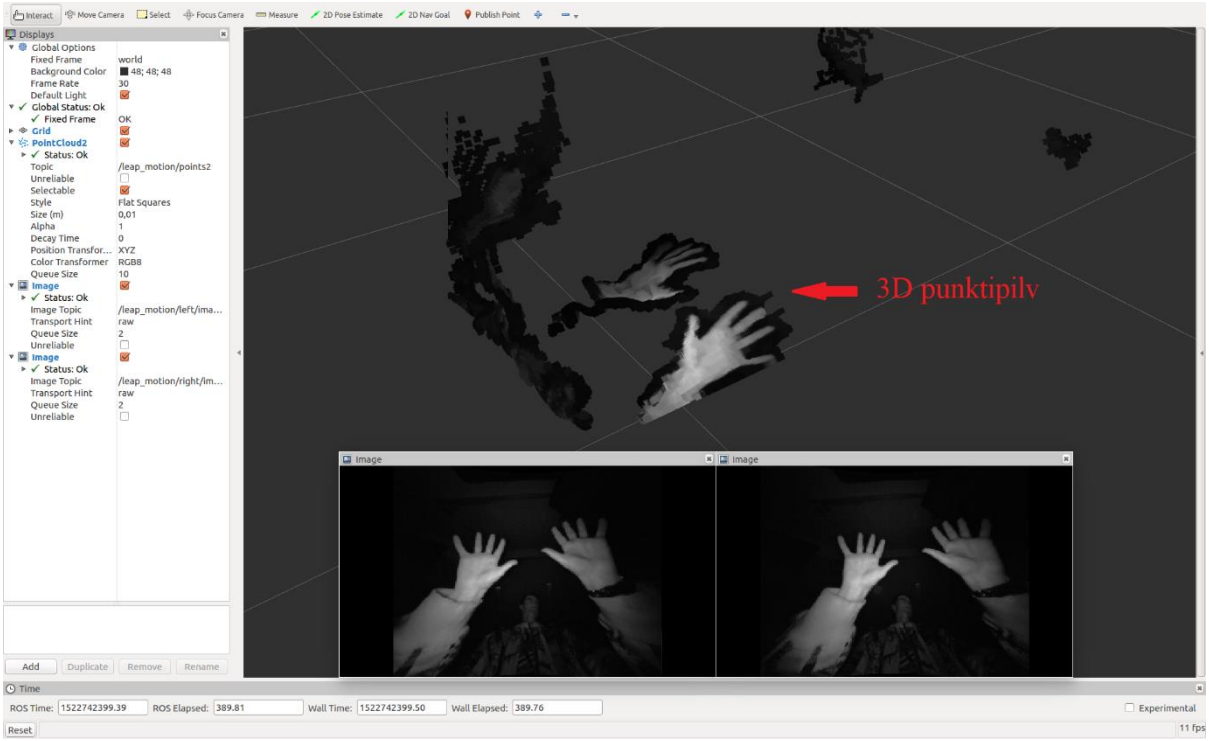


Joonis 21. *RVizi kuvatõmmis käte kujutamisel lmc_ros_driver paketi abil.*

Filtreerimise käigus rakendatakse 2. järgu *Butterworth* madalpääs filtrit, mis võeti kohandatult üle Texase Ülikooli koodihoidlast. Kuna inimene ei suuda kunagi oma kätt täielikult paigal

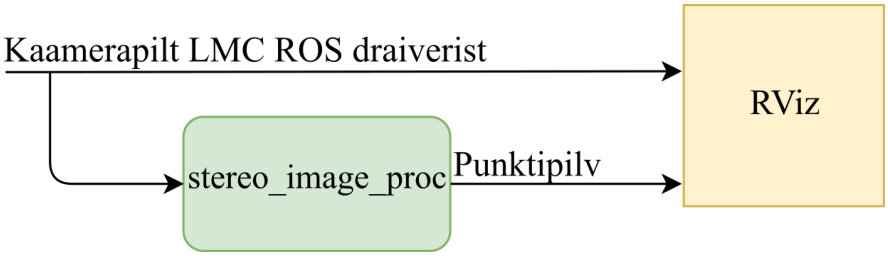
hoida, on filtri eesmärgiks stabiliseerida käe asukohta ning eemaldada käte värinat. Kasutajal on võimalik filtrit sisse-välja lülitada ning muuta lõikesagedust kasutades ROS parameetriserverit.

Töötlemata kaamerapildi edastust võimaldav lähtekood võeti üle `leap_motion` paketist. Sõlm tegeleb seadme töötlemata pildi edastamisega nii paremast kui vasakust kaamerast (joonis 22). Kasutades RVizis ROSi kogukonna poolt eeldefineeritud sõnumit „*sensor_msgs/Image*“ saab kasutaja näha LM-kontrolleri edastatud pilti otseülekandes. Ülevõetud lähtekood oli täielikult dokumenteerimata, seega lisati antud bakalaureusetöö raames tarkvara tööpõhimõtet selgitavaid kommentaare. Lisaks viidi lähtekood vastavusse ROSi stiilinõuetega.



Joonis 22. *Kontrolleri poolt edastatud videokaadrid ning nende põhjal genereeritud punktipilv.*

Kontrolleri kahest kaamera pildist on võimalik genereerida andmepunkte, mis asuvad kolmemõõtmelises koordinaadisüsteemis ehk punktipilve. Taolise ülesande jaoks eksisteerib ROSi pakett `stereo_image_proc` [55], mis võtab sisendiks kahe kaamera pildi ja arvutab nende põhjal 3D punktipilve. Rviz baasvõimekuses on punktipilvede graafiline kuvamine ning selleks tuli RVizis (joonis 22 ja 23) tellida `stereo_image_proc` paketi genereeritud punktipilve sõnumeid (*„sensor_msgs/PointCloud2“*).



Joonis 23. *LM draiverist suunatakse kaamerapilt otse RVizi ning samal ajal ka stereo_image_proc paketti, kus luuakse stereopildi põhjal punktipilv.*

7.4. lmc_ros_driver ja leap_motion paketi võrdlus

Peale töökäigus loodud `lmc_ros_draiver` valmimist sai seda võrreldud ametliku ROS `leap_motion` draiverpaketiga (tabel 1). Mõõtes mõlema draiveri peasõlme infoedastuse sagedust käsuga `rostopic hz /teema_nimi` tuli ametliku draiveri tulemuseks 64 Hz. Teostades analoogset testi loodud `lmc_ros_driver` puhul tuli infoedastuse sageduseks 91 Hz, kuigi paketi poolt edastatav infohulk on suurem.

Lisaks võrreldi kummagi lahenduse poolt pakutavaid lisavõimalusi nagu käte asukoha filtreerimine, draiveri sätete konfigureerimine, žestide tuvastus ning andmete graafiline kujutamine. `lmc_ros_driver` paketis on andmete graafiliseks kujutamiseks loodud erinevaid kiirkäivitusskripte, kuid `leap_motion` paketis esinev lahendus on ilma dokumentatsioonita ning selle kasutamiseks tuleb RViz käsitsi õigete sätetega üles seada.

Tabel 1. *Loodud LMC ROS draiveri ning varasema leap_motion paketi võrdlus.*

	`leap_motion`	`lmc_ros_driver`
Peasõlme infoedastuse keskmine sagedus 1 min jooksul.[1,2]	64 Hz	91 Hz
Programmeerimiskeel	Pythoni ja C++ segu	C++
Toetab žeste	Ei	Jah
Kaamerapildi edastus	Jah, mõlemalt kaameralt.	Jah, mõlemalt kaameralt.
Draiveri sätete konfigureerimine	Argumentide kaudu draiveri käivitamisel.	ROS parameetriserveri kaudu
Käte positsiooni filtreerimine	Ei	Jah
Andmete graafiline kujutamine	Dokumenteerimata võimalus	Dokumenteeritud võimalus koos kiirkäivitusskriptiga
Kiirkäivitusskriptid	Ainult käe positsiooni edastamiseks.	Erinevad skriptid draiveri võimaluste demonstratsiooniks
Väljastatavad sõnumid	Ainult leapros.msg	Spetsiifilised sõnumid iga tuvastatud objekti ja tegevuse kohta
Lähtekoodi kommentaarid	Kohati	Intensiivselt kommenteeritud

Lähtekood järgib ROS stiilinõudeid	C++ failides eiratakse	Jah
Vikileht	Jah, kuid aegunud.	Ei, kuid koodihoidla READMEs kajastub selle tulevane sisu
Kättesaadavus	GitHubi koodihoidlast ja läbi ROSi koosteserveri binaarkujul	IMS Robotics GitHubi koodihoidlast

1. `leap_motion` paketti on lisatud sageduse piiraja, kuid `lmc_ros_driver` paketis see puudub.
2. Peasõlmede poolt edastava info kogus pole võrdne. `lmc_ros_driver` paketi poolt edastav infohulk on suurem.

8. Draiveri tutvustus ROSi kogukonnale

8.1. ROSi GitHub organisatsioonid ja ametlik koosteserver

ROSil eksisteerib mitmeid GitHub organisatsioone nagu *ROS Core*, *ROS Drivers*, *ROS Visualization*, milles hoitakse mitmeid ROSi põhifunktsionaalsust tagavaid pakette. Erinevate organisatsioonide eesmärgiks on samasugust rolli täitvate ROS pakettide grupeerimine kergeks haldamiseks erinevate arendajate poolt. Kuna ROS on kogukonna põhine, siis eeldab ka arendusprotsess läbirääkimisi teiste liikmetega, et leida konsensus.

Lisaks on ROSi arendajatele kättesaadav Open Robotics poolt hallatav ROSi ametlik koosteserver (*„build farm“*). Avalikku koosteserverit kasutatakse ROSi põhifunktsionaalsuse tagavate pakettide ja teiste avatud lähtekoodiga ROS pakettide binaarkujul levitamiseks. OSRF pakub arendajatele võimalust oma poolt loodud ROSi pakett registreerida, koosteserveri virtuaalmasinat paketi spetsiifiliselt seadistada ning seeläbi saada osa Open Robotics serveri poolt pakutavatest teenustest. Koosteserver võimaldab vastuvõetud ROS pakettide levitamist binaarkujul läbi Linuxi *apt* paketihalduri, pidevintegratsiooni, erinevaid standardiseeritud teste ja tööriistu paketi kompileerimise logi analüüsiks [56].

8.2. Arutelud ROS kogukonnas

Kuna LM-kontrolleril eksisteerib puuduliku funktsionaalsuse ja dokumentatsiooniga ROSi draiver, siis on otstarbekas asendada just olemasolev tarkvara, et seeläbi vähendada võimalikku dubleerimist ja kaasnevat segadust. Samuti jõuab sedasi kiiremini draiveri binaarkujul levitamiseni, sest ei pea koosteserveri nimekirja eraldi ligipääsu taotlema ning kõiki sellega kaasnevaid faile seadistama, vaid need on praeguste haldajate poolt juba eelkonfigureeritud. Seega on võimalik suunata uus draiverilahendus ROSi kogukonda tehes koodipakkumine (*„pull request“*) olemasoleva draiveri koodihoidlasse.

2018. aasta veebruaris loodi esimene kontakt *ros-drivers/leap_motion* jälgijate ja haldajatega. Autor esitas draiveri koodihoidlas küsimuse („*issue*“), milles kirjeldas loodud draiveri võimalusi ja, selle loomise motivatsiooni ning kutsus üles kõiki loodud lahendust katsetama (https://github.com/ros-drivers/leap_motion/issues/36). Analoogse sisuga meil saadeti ka kolmele draiveri vikilehel mainitud haldajale, et tagada kõigi osapoolte kaasamine. Kolmest haldajast reageeris saadetud kirjale ainult üks.

Florian Lier oli igati nõus kahe draiveri sulandamiseks, kuid mainis, et küsimust tuleb arutada ka teiste kogukonna liikmetega**.** Kuna hetkel on `leap_motion` draiveri lähtekood GitHubis organisatsiooni ros-drivers koodihoidlas sai grupi teadete tahvlile tehtud vastav postitus (https://wiki.ros.org/sig/Drivers?place=topic%2Fros-sig-drivers%2FNGSihcq_jKI%2Fdiscussion).

ROSi ametlike draiverite kogukonnast vastas päringule Jack O'Quin, kes sisuliselt nõustus uue draiveri kasutuselevõtuga, kuid ei soovitanud vana lahenduse kohest eemaldamist. Tema soovitus oli kasutada nn. tikk-takk mudelit [57] ehk uut draiverit kasutusele võttes säilitatakse ka vana lahendus vähemalt ühe ROS väljalaske jooksul. Peale seda vana draiver eemaldatakse ning kasutusele jääb ainult uus. Sellise arendustsükli kasutamise eelis on vana draiveri versiooni tagasiühilduvuse säilitamine lubades ROSi arendajatel uue draiveri kasutusele võtu ajagraafikus, mis on neile sobilik.

Viimaseks arutelu teemaks oli koodipakkumise avamine `leap_motion` koodihoidlasse. Samuti tehti lõputöö autorile ettepanek hakata uueks *ros-drivers/leap_motion* koodihoidla haldajaks (https://github.com/ros-drivers/leap_motion/issues/37).

8.3. GitHubi koodipakkumine

Kuna nn. tikk-takk mudel eeldab, et esialgu jäävad draiveripaketti alles ka kogu varasem võimekus, ühendati eksisteeriv lahendus käesolevas lõputöös arendatuga. Kuna ühendamisel konflikte ei tuvastatud ning mõlemaid lahendusi sai kasutada teineteisest sõltumatult, siis tehti 7. märtsil 2018 *ros-drivers/leap_motion* koodihoidlasse sisupakkumine, mis sisaldas kogu lõputöö raames arendatud tarkvara (https://github.com/ros-drivers/leap_motion/pull/38).

Koodipakkumisele tuli ühe `leap_motion` paketi koodihaldaja poolt kiire vastus, kus mainiti, et kuna pakutav lahendus lisab olemasolevasse draiverisse suurel hulgal uut lähtekoodi peavad praegused haldajad leidma aega, et kontrollida üle lisatava lähtekoodi kvaliteet. Järgnenud diskusiooni käigus liikus arutelu uue draiveri sulandamiselt `leap_motion` paketi Travis CI integratsioonitesti parandamisele. Praegune `leap_motion` koodihoidla poolt kasutatav pidev integratsiooni (CI) lahendus ei töötanud Leap Motion SDK binaarfailide puudumisel. Integratsioonitesti korda tegemisel oleks võimalik koodihoidla haldajatel hõlpsasti tuvastada lisatava lähtekoodi kompileeruvus, käitada automaatseid teste ning kompileerida lähtekoodi kõigi arendajate jaoks identses virtuaalmasinas.

Töö valmimise ajaks sai autor enda loodud `leap_motion` paketi koopias Travis CI integratsioonitesti tööle, kuid lahendamata probleemiks oli Leap Motion SDK binaarfailidele skriptis krüpteeritud kujul programmaatiliselt viitamine. Vajalikke faile ei tohtinud lihtsalt `leap_motion` koodihoidlasse lisada LeapSDK litsentsitingimuste tõttu. LeapSDK kasutustingimused ei luba selle individuaalsete osade levitamist välja arvatud arendajate lahendusse kompileeritult. Seetõttu peab vajalikke binaarfaile hoidma privaatses koodihoidlas või lisama krüpteeritud viite failiserverile. Kuna Travis CI tasuta versioon ei toeta neid krüpteerimisviise, mis oleks lubanud salastatud viite lisamist privaatsele koodihoidale, siis seetõttu ei jõutud töö valmimise ajaks veel kõigile osapooltele sobiva lahenduseni.

8.4. Loodud lahenduse kasutamine

Hetkel on töö käigus loodud `lmc_ros_driver` pakett kättesaadav UT IMS Robotics organisatoorses GitHub koodihoidlas aadressil: https://github.com/ut-ims-robotics/lmc_ros_driver. Koodihoidla README fail tutvustab draiveri poolt pakutavaid võimalusi, loetleb üles paigaldamiseks ja kasutamiseks vajalikud teegid ning sisaldab kasutusjuhendit draiveripaketi paigaldamiseks lähtekoodist.

Paigaldatud draiveripaketi kasutamiseks on loodud ka mitmeid kiirkäivitusskripte, mille abil saab paketi kasutaja rakendada mugavat *roslaunch* käsku, mis käivitab kõik vajalikud sõlmed, laadib parameetriserverisse vaikeväärtused ning avab õigete sätetega RVizi akna. Näiteks `roslaunch lmc_ros_driver visualization.launch` käivitab LMC ROS draiversõlme, vastavalt ROS parameetriserveri väärtustele filtrisõlme, visualisatsioonisõlme ning eelkonfigureeritud RViz akna. Seeläbi on võimalik demonstreerida kogu draiveri põhifunktsionaalsust kasutades üht kergesti kutsutavat käsku.

Draiverit testiti ROS Kinetic Kame väljalaske ja Ubuntu versiooni 16.04 Xenial Xerus platvormil. Kuna koodipakkumist leap_motion koodihoidlasse (peatükk 8.3) pole seniste arendajate poolt veel vastu võetud, siis pole uuendatud ametliku draiveri ROSi vikilehekülge ega saavutatud seda, et draiver oleks binaarkujul kättesaadav Ubuntu Linuxi paketihalduri käsuga `sudo apt-get install ros-<väljalaske nimi>-leap-motion`.

9. Kokkuvõte

Käesoleva bakalaureusetöö käigus loodi ROS arendusplatvormile uus avatud lähtekoodiga Leap Motion kontrolleri draiveripakett, et laiendada olemasoleva draiveri funktsionaalsust. Paketi peamiseks eesmärgiks on edastada kasutaja käte asukohta ja seotud žeste ROSi sõnumitena ning võimaldada nende sõnumite graafilist kujutamist 3D visualiseerimiskeskkonnas RViz. Kogu infovahetus toimub selgesti pealkirjastatud ROS teemade all ning loodud kiirkäivitusskriptid lubavad loodud lahendust hõlpsasti kasutada ka algajal robootikul, kuid pakub ka maksimaalset LeapSDK tehnilist võimekust keerukamate süsteemide inseneridele.

Hetkel on töö käigus loodud `lmc_ros_driver` pakett kättesaadav UT IMS Robotics organisatoorses GitHub koodihoidlas aadressil: https://github.com/ut-ims-robotics/lmc_ros_driver. Koodihoidla README fail tutvustab draiveri poolt pakutavaid võimalusi, loetleb üles paigaldamiseks ja kasutamiseks vajalikud teegid ning sisaldab kasutusjuhendit draiveripaketi paigaldamiseks lähtekoodist.

Viited

[1] Rethink Robotics, “Baxter Collaborative Robots for Industrial Automation.” [Võrgumaterjal]. Kättesaadav: https://www.rethinkrobotics.com/baxter/. [Külastatud: 19.05.2018].

[2] SoftBank Robotics, “Pepper, the humanoid robot from SoftBank Robotics, a genuine companion.” [Võrgumaterjal]. Kättesaadav: https://www.softbankrobotics.com/emea/en/robots/pepper. [Külastatud: 19.05.2018].

[3] Yoonchang Sung and Woojin Chung, “Human tracking of a mobile robot with an onboard LRF (Laser Range Finder) using human walking motion analysis,” *2011 8th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, 2011, lk. 366–370.

[4] S. V. Tathe and S. P. Narote, “Real-time human detection and tracking,” *2013 Annual IEEE India Conference (INDICON)*, 2013, lk. 1–5.

[5] J. Krüger, T. K. Lien, and A. Verl, “Cooperation of human and machines in assembly lines,” *CIRP Ann. - Manuf. Technol.*, kd. 58, nr. 2, lk. 628–646, jaanuar 2009.

[6] D. Zhang, F. Xia, Z. Yang, L. Yao, and W. Zhao, “Localization Technologies for Indoor Human Tracking,” *2010 5th International Conference on Future Information Technology*, 2010, lk. 1–6.

[7] Z. Zhang, “Microsoft Kinect Sensor and Its Effect,” *IEEE Multimed.*, kd. 19, nr. 2, lk. 4–10, veebruar 2012.

[8] S. Ha, Y. Bai, and C. K. Liu, “Human motion reconstruction from force sensors,” *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation - SCA ’11*, 2011, lk. 129.

[9] R. Das and K. B. S. Kumar, “GeroSim: A simulation framework for gesture driven robotic arm control using Intel RealSense,” *2016 IEEE 1st International Conference on Power Electronics, Intelligent Control and Energy Systems (ICPEICES)*, 2016, lk. 1–5.

[10] H. Aagela, M. Al-Nesf, and V. Holmes, “An Asus_xtion_probased indoor MAPPING using a Raspberry Pi with Turtlebot robot Turtlebot robot,” *2017 23rd International Conference on Automation and Computing (ICAC)*, 2017, lk. 1–5.

[11] S. A. Guomundsson, R. Larsen, H. Aanaes, M. Pardas, and J. R. Casas, “TOF imaging in Smart room environments towards improved people tracking,” *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2008, lk. 1–6.

[12] R. Bogue, “Detecting humans in the robot workspace,” *Ind. Robot An Int. J.*, kd. 44, nr. 6, lk. 689–694, oktoober 2017.

[13] G. Hidalgo, Z. Cao, T. Simon, S.-E. Wei, H. Joo, and Y. Sheikh, “GitHub - CMU-Perceptual-Computing-Lab/openpose: OpenPose: Real-time multi-person keypoint detection library for body, face, and hands estimation.” [Võrgumaterjal]. Kättesaadav: https://github.com/CMU-Perceptual-Computing-Lab/openpose. [Külastatud: 08.04.2018].

[14] Z. Cao, T. Simon, S. E. Wei, and Y. Sheikh, “Realtime multi-person 2D pose estimation

using part affinity fields," *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017, kd. 2017–jaanuar, lk. 1302–1310.

[15] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, lk. 779–788.

[16] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "YOLO: Real Time Object Detection · pjreddie/darknet Wiki · GitHub," 2016. [Võrgumaterjal]. Kättesaadav: https://github.com/pjreddie/darknet/wiki/YOLO:-Real-Time-Object-Detection. [Külastatud: 21.04.2018].

[17] C. Seubert, "How Do Laser Distance Meters Work?," *Sciencing*, 2017. [Võrgumaterjal]. Kättesaadav: https://sciencing.com/do-laser-distance-meters-work-6332366.html. [Külastatud: 11.04.2018].

[18] Mike1024, "File:LIDAR-scanned-SICK-LMS-animation.gif - Wikimedia Commons," *Wikimedia Commons*, 2008. [Võrgumaterjal]. Kättesaadav: https://commons.wikimedia.org/wiki/File:LIDAR-scanned-SICK-LMS-animation.gif. [Külastatud: 18.04.2018].

[19] J. Lee, T. Tsubouchi, K. Yamamoto, and S. Egawa, "People Tracking Using a Robot in Motion with Laser Range Finder," *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006, lk. 2936–2942.

[20] B. Templeton, "Cameras or Lasers?," 2013. [Võrgumaterjal]. Kättesaadav: https://www.templetons.com/brad/robocars/cameras-lasers.html. [Külastatud: 18.04.2018].

[21] E. S. Peter Kaldén and E. Sternå, "Development of a low-cost laser range-finder," M. S. thesis, Chalmers University of Technology, 2015.

[22] Comet Labs Research Team, "Depth sensors are the key to unlocking next level computer vision applications.," *Comet Labs*, 2017. [Võrgumaterjal]. Kättesaadav: https://blog.cometlabs.io/depth-sensors-are-the-key-to-unlocking-next-level-computer-vision-applications-3499533d3246. [Külastatud: 18.04.2018].

[23] A. Bernin, "Kinect chess board meta pattern," *Living Place Hamburg*, 2010. [Võrgumaterjal]. Kättesaadav: http://livingplace.informatik.haw-hamburg.de/blog/?p=229. [Külastatud: 18.04.2018].

[24] I. Corporation, "Intel ® RealSense TM D400 Series (DS5) Product Family Datasheet," andmeleht, jaanuar 2018.

[25] M. Munaro, A. Horn, R. Illum, J. Burke, and R. B. Rusu, "Openptrack: People tracking for heterogeneous networks of color-depth cameras," IAS-13 Workshop Proceedings: 1st Intl. Workshop on 3D Robot Perception with Point Cloud Library, lk. 235–247, 2014.

[26] Leap Motion Inc, "Application Asset and Marketing Guidelines," 2018. [Võrgumaterjal]. Kättesaadav: https://developer.leapmotion.com/documentation/v2/javascript/practices/App_Assets_and_Marketing_Guidelines.html. [Külastatud: 24.02.2018].

[27] W. Shanklin, "Leap Motion teams up with OSVR for motion-controlled virtual reality," *New Atlas*, 2015. [Võrgumaterjal]. Kättesaadav: https://newatlas.com/leap-motion-osvr-

virtual-reality/36714/. [Külastatud: 28.03.2018].

[28] A. Logan, “How Does the Leap Motion Controller Work?,” 2014. [Võrgumaterjal]. Kättesaadav: http://blog.leapmotion.com/hardware-to-software-how-does-the-leap-motion-controller-work/. [Külastatud: 24.02.2018].

[29] Ignitetron, “Leap Motion.” [Võrgumaterjal]. Kättesaadav: http://ignitetron.com/leap-motion/. [Külastatud: 01.05.2018].

[30] Leap Motion Inc, “Frame — Leap Motion C++ SDK v2.3 documentation.” [Võrgumaterjal]. Kättesaadav: https://developer.leapmotion.com/documentation/v2/cpp/api/Leap.Frame.html. [Külastatud: 20.05.2018].

[31] Open Robotics, “Is ROS For Me?” [Võrgumaterjal]. Kättesaadav: http://www.ros.org/is-ros-for-me/. [Külastatud: 08.04.2018].

[32] T. Dirk, “Introduction - What is ROS?,” 2014. [Võrgumaterjal]. Kättesaadav: https://wiki.ros.org/ROS/Introduction. [Külastatud: 24.02.2018].

[33] AaronMR, “ROS Concepts - ROS Wiki,” 2014. [Võrgumaterjal]. Kättesaadav: https://wiki.ros.org/ROS/Concepts. [Külastatud: 21.03.2018].

[34] I. Carpis, “RealSense Camera package.” [Võrgumaterjal]. Kättesaadav: https://wiki.ros.org/realsense_camera. [Külastatud: 18.05.2018].

[35] P. Goebel, “Skeleton Markers - joint markers for viewing in RViz.” [Võrgumaterjal]. Kättesaadav: https://wiki.ros.org/skeleton_markers. [Külastatud: 03.05.2018].

[36] G. Garrat, “Hand interaction package.” [Võrgumaterjal]. Kättesaadav: https://wiki.ros.org/hand_interaction. [Külastatud: 03.05.2018].

[37] R. Uibo, “Projektsiooniline anatoomia Sisukord,” 2006. [Võrgumaterjal]. Kättesaadav: https://kodu.kliinikum.ee/radioloogia/images/stories/oppetoo/residentuur/residentuuri_konspektid/Projektsiooniline_anatoomia.pdf. [Külastatud: 21.03.2018].

[38] M. Ruiz, “File:Scheme human hand bones-en.svg - Wikimedia Commons,” 2007. [Võrgumaterjal]. Kättesaadav: https://commons.wikimedia.org/wiki/File:Scheme_human_hand_bones-en.svg. [Külastatud: 21.03.2018].

[39] R. R. Itkarkar and A. V. Nandi, “A survey of 2D and 3D imaging used in hand gesture recognition for human-computer interaction (HCI),” *2016 IEEE International WIE Conference on Electrical and Computer Engineering (WIECON-ECE)*, 2016, lk. 188–193.

[40] M. Horde, “Leap Motion Teardown - learn.sparkfun.com.” [Võrgumaterjal]. Kättesaadav: https://learn.sparkfun.com/tutorials/leap-motion-teardown. [Külastatud: 08.04.2018].

[41] Leap Motion Inc, “Releases — Leap Motion Developer.” [Võrgumaterjal]. Kättesaadav: https://developer.leapmotion.com/releases/?category=orion. [Külastatud: 21.03.2018].

[42] L. M. Inc, “API Overview — Leap Motion C++ SDK v2.3 documentation.” [Võrgumaterjal]. Kättesaadav: https://developer.leapmotion.com/documentation/v2/cpp/devguide/Leap_Overview.ht

ml. [Külastatud: 24.02.2018].

[43] R. Bedikian, “Understanding Latency: Part 1 - Leap Motion Blog,” *Leap Blog*, 2013. [Võrgumaterjal]. Kättesaadav: http://blog.leapmotion.com/understanding-latency-part-1/. [Külastatud: 08.04.2018].

[44] R. Bedikian, “Understanding Latency: Part 2 - Leap Motion Blog,” *Leap Blog*, 2013. [Võrgumaterjal]. Kättesaadav: http://blog.leapmotion.com/understanding-latency-part-2/. [Külastatud: 08.04.2018].

[45] R. Madaan, “A ROS package for teleoperating a youBot.” [Võrgumaterjal]. Kättesaadav: https://github.com/madratman/youbot_leapmotionteleop. [Külastatud: 18.05.2018].

[46] E. McCann, “ROS_OpenLeap.” [Võrgumaterjal]. Kättesaadav: https://github.com/nuclearmistake/ROS_OpenLeap. [Külastatud: 18.05.2018].

[47] P. Lorefice, “ROS package for controlling an AR.Drone 2.0 using a Leap Motion.” [Võrgumaterjal]. Kättesaadav: https://github.com/plorefice/ardrone-leapjoy. [Külastatud: 18.05.2018].

[48] zmmrose, “Leap3DJam.” [Võrgumaterjal]. Kättesaadav: https://github.com/zmmrose/rose-Leap3DJam. [Külastatud: 18.05.2018].

[49] A. Zelenak, “ROS driver for Leap Motion Controller (TM),” 2017. [Võrgumaterjal]. Kättesaadav: https://github.com/UTNuclearRoboticsPublic/leap_motion_controller. [Külastatud: 10.05.2018].

[50] J. O. (Julius O. Smith, *Introduction to digital filters : with audio applications*. W3K, 2008.

[51] J. Camilog, “A ROS C++ wrapper around the Leap Motion API.” [Võrgumaterjal]. Kättesaadav: https://github.com/juancamilog/leap_client. [Külastatud: 10.05.2018].

[52] P. Bouchier, “CppStyleGuide.” [Võrgumaterjal]. Kättesaadav: https://wiki.ros.org/CppStyleGuide. [Külastatud: 18.05.2018].

[53] D. Thomas, “msg.” [Võrgumaterjal]. Kättesaadav: https://wiki.ros.org/msg. [Külastatud: 18.05.2018].

[54] Leap Motion, “API Reference — Leap Motion C++ SDK v2.3 documentation.” [Võrgumaterjal]. Kättesaadav: https://developer.leapmotion.com/documentation/v2/cpp/api/Leap_Classes.html. [Külastatud: 13.05.2018].

[55] V. Rabaud, “stereo_image_proc.” [Võrgumaterjal]. Kättesaadav: https://wiki.ros.org/stereo_image_proc. [Külastatud: 19.05.2018].

[56] T. Foote, “buildfarm - ROS Wiki,” 2016. [Võrgumaterjal]. Kättesaadav: https://wiki.ros.org/buildfarm. [Külastatud: 14.05.2018].

[57] T. Foote and K. Conley, “ROS Enhancement Proposal 3,” 2010. [Võrgumaterjal]. Kättesaadav: http://www.ros.org/reps/rep-0003.html#architectures. [Külastatud: 20.05.2018].

Lihtlitsents

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina, Kristo Allaje,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose

„DRAIVERIPAKETT KÄTE JÄLGIMISSEADME LEAP MOTION™ KONTROLLER KASUTAMISEKS ROBOOTIKA ARENDUSPLATVORMIL ROS“

mille juhendaja on Karl Kruusamäe

1.1. reprodutseerimiseks säilitamise ja üldsusele kättesaadavaks tegemise eesmärgil, sealhulgas digitaalarhiivi DSpace-is lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;

1.2. üldsusele kättesaadavaks tegemiseks Tartu Ülikooli veebikeskkonna kaudu, sealhulgas digitaalarhiivi DSpace´i kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.

2. olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile.

3. kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest tulenevaid õigusi.

Tartus, **20.05.2018**