TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Artjom Nikokošev IAIB 213104

OBJEKTIDE KAARDISTAMISE RAKENDUS LIIKUMISE REGISTREERIMISE SÜSTEEMI JAOKS

Bakalaureusetöö

Juhendaja: Sven Nõmm
PhD

Tallinn 2024

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Artjom Nikokošev

27.05.2024

Annotatsioon

Käesoleva lõputöö põhieesmärk on välja töötada süsteem, mis tuvastaks ja kaardistaks objektid ruumis, kus tehakse jämedat motoorseid teste. Põhieesmärk on välja töötada süsteemi NVIDIA GPU-ga varustatud arvuti jaoks. Projekti viimases etapis on plaanis hinnata selle portimise võimalust Jetson Orin NANO platvormile. On vaja välja töötada ja demonstreerida üks või mitu arvutinägemise rakendust Jetson Orin NANO'l, nimeks objektide tuvatamine. Eesmärgiks on luua lahendus, mis oleks tõhus nii töökiiruselt kui ka täpsuselt, pidades samal ajal Jetson Orin NANO võimsuse ja mälu piiranguid. Oodatavaks tulemuseks on töötav prototüüp, mis näitab arvutinägemise rakenduste praktilist teostatavust Jetson Orin NANO'l, samuti järeldused ja soovitused arvutinägemuse algoritmide kohandamiseks ja optimeerimiseks sellel platvormil.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 40 leheküljel, 10 peatükki, 40 joonist, 4 tabelit.

Abstract

Object mapping application for a motion registration system

The main goal of this thesis is to develop a system that would detect and map objects in a room where gross motor tests are performed. System should be developed for a computer equipped with an NVIDIA GPU. In the last stage of the project, it is planned to evaluate the possibility of porting it to the Jetson Orin NANO platform. It is necessary to develop and demonstrate one or more computer vision applications for the Jetson Orin NANO, namely object recognition. The aim is to create a solution that is efficient in terms of both speed and accuracy, while staying within the power and memory limitations of the Jetson Orin NANO. The expected result is a working prototype that demonstrates the practical usage of computer vision application on the Jetson Orin NANO, as well as conclusions and recommendations for adapting and optimizing computer vision algorithms on this platform.

The thesis is written in Estonian and is 40 pages long, including 10 chapters, 40 figures and 4 tables.

Lühendite ja mõistete sõnastik

AI	*Artificial Intelligence*, tehisintellekt
API	*Application Programming Interface*, rakendusliides
ASCII	*American Standard Code for Information Interchange*, Ameerika standardkood teabevahetuseks
AWS	*Amazon Web Services*, Amazoni veebiteenused
BGRA	*Blue, Green, Red, Alpha*, sinine, roheline, punane, alpha
CD	*Continuous Delivery/Deployment*, pidev tarnimine/juurutamine
CI	*Continuous Integration*, pidev integreerimine
CLI	*Command Line Interface*, käsurea liides
CNN	*Convolutional Neural Network*, konvolutsioonilene närvivõrk
COCO	*Common Objects in Context*, ühesugused objektid kontekstis
CPU	*Central Processing Unit*, keskseade
CSV	*Comma Separated Values*, komaga eraldatud väärtused
CUDA	*Compute Unified Device Architecture*, arvutatud ühendatud seadme arhitektuur
EMG	*Electromyography*, elektromüograafia
ETAG	Eesti Teadusagentuur
FLOPS	*Floating Point Operations Per Second*, ujuvkomaoperatsioone sekundis
GB	*Gigabyte*, gigabait
GELAN	*Generalized Efficient Layer Aggregation Network*, üldine tõhus kihtide koondamise võrk
GPU	*Graphics Processing Unit*, graafikaprotsessor
HD	*High Definition*, kõrgkvaliteet
IDE	*Integrated Development Environment*, integreeritud programmeerimiskeskkond
IMU	*Inertial Measurement Unit*, inertsiaalmõõteühik
IOT	*Internet Of Things*, asjade internet
IOU	*Intersection over Union*, lõikepindala suhe ühendpindalasse
LPDDR	*Low-Power Double Data Rate*, väikese võimsusega kahekordne andmeedastuskiirus

ML	*Machine Learning*, masinõpe
MM	*Motion Mass*, liikumismass
NumPy	*Numerical Python*, numbriline python
ONNX	*Open Neural Network Exchange*, avatud neuraalvõrkude vahetus
OpenCV	*Open Source Computer Vision Library*, avatud lähtekoodiga arvutinägemise teek
OS	*Operating System*, operatsioonisüsteem
PC	*Personal Computer*, personaalarvuti
PD	*Parkinson's Disease*, Parkinsoni tõve
PGI	*Programmable Gradient Information*, programmeeritav gradiendi teave
PIL	*Python Imaging Library*, Pythoni pilditöötlusraamatukogu
R-CNN	*Region-based Convolutional Neural Networks*, piirkonnapõhised konvolutsioonilised närvivõrgud
RGB	*Red, Green, Blue*, punane, roheline, sinine
ROI	*Region Of Interest*, huvipakkuv piirkond
RPN	*Region Proposal Network*, piirkonnapakkumise võrk
SDK	*Software Development Kit*, tarkvaraarenduskomplekt
SIFT	*Scale Invariant Feature Transform*, skaalainvariantsete funktsioonide teisendus
SOTA	*The State of the Art*, tehnika tase
SSD	*Single Shot Detector*, ühe pildi detektor
TOPS	*Trillions or Tera Operations per Second*, triljoneid või tera operatsioone sekundis
TUG	*Timed Up and Go*, ajastatud üles-tõusmise ja mine-test
USB	*Universal Serial Bus*, universaalne jadaliides
USP	*Unique Selling Proposition*, ainulaadne müügipakkumine
VGA	*Video Graphics Array*, videograafika massiiv
VM	*Virtual Machine*, virtuaalmasin
YAML/YML	*Yet Another Markup Language*, veel üks märgistuskeel
YOLO	*You Only Look Once*, vaatad vaid ühe korra

Sisukord

Jooniste loetelu

Tabelite loetelu

1. Sissejuhatus

Viimaste aastate jooksul on Sven Nõmm'i töörühm saavutanud märkimisväärseid edusamme jäme mootorsete liikumise registreerimise protsessi automatiseerimisel. Neurodegeneratiivsete haiguste, näiteks Parkinsoni tõbi, diagnoosimiseks ja raskusastme hindamiseks kasutatakse jämemotoorseid teste, nagu Tõuse-Ja-Mine (Up-and-Go).

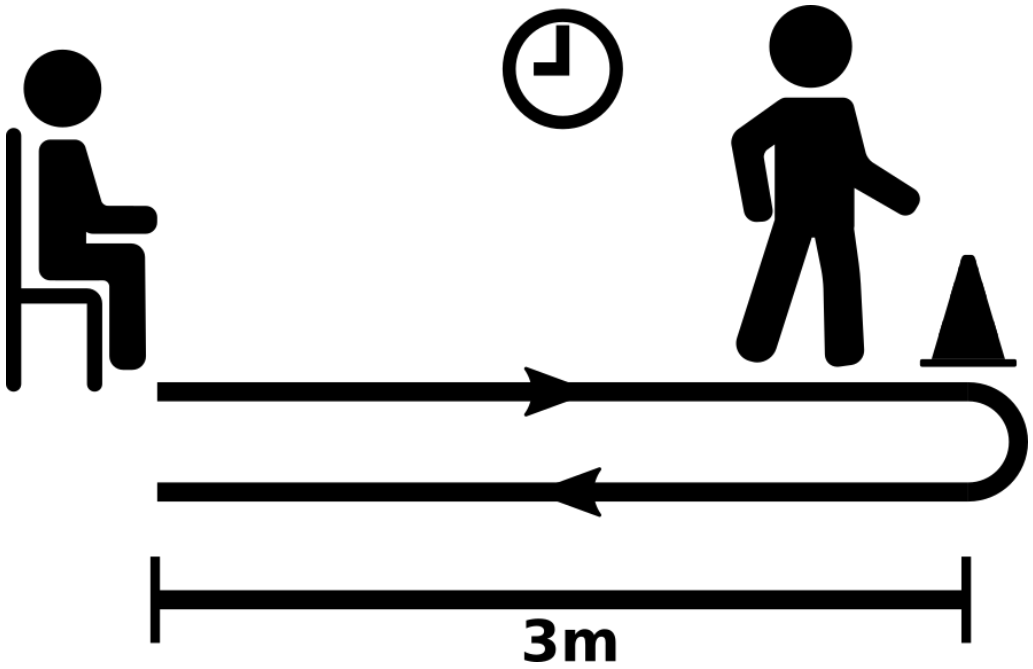
1.1 Taust

On teada, et 1-2% elanikkonnast, kes on vanemad kui 65 aastat, kannatavad Parkinsoni Tõve (PD) all. See on neurodegeneratiivne haigus, mis mõjutab inimese motoorseid funktsioone, põhjustades tahtmatuid liigutusi ja liigeste jäikust. Iseloomulikeks sümptomiteks on kõnnaku stabiilsuse järjepidev langus, suurenev värisemine ja lihaste jäikus. Esineb ka mõningaid mittemotoorseid sümptomeid, nagu unehäired, depressioon ja teised. Arvatakse, et haigus on põhjustatud dopamiini puudusest dopaminergiliste neuronite surma tõttu [1].

Parkinsoni tõve mõjutab tõsiselt inimese liikumist, põhjustades keha värisemist, lihaste jäikust ja kõnnaku ebastabiilsust. See toob kaasa muutlikkuse liikumise soorituse efektiivsuses, liikumiste hulgas ja sujuvuses. Neid kõrvalekaldeid on võimalik tuvastada kõnnianalüüsi abil. Kõnnianalüüs on inimliikumise uuring, mis keskendub keha liikumise ja lihaste tegevuse mõõtmisele. Tavaliselt mõõdetakse mõningaid parameetreid, et tulemusi kvantifitseerida ja paremini tuvastada motoorseid kõrvalekaldeid või vale tehnikat, juhul kui seda kasutatakse spordi hindamiseks [1].

On mitmeid kõnniliikuvuse teste, mida saab kasutada vanemate või vigastatud inimeste tasakaalu hindamiseks ning erinevate füüsiliste häirete tõttu tekkivate kõnnikõrvalekallete tuvastamiseks. Need võivad pakkuda kasulikku ülevaadet isiku seisundist ja aidata kavandada vajalikku taastusravi või valida õige hooldusstrateegia. Mõned näited on istumisest tõusmise test, mine-test, 4-meetri kõnnitest, üles-tõusmise ja mine-test [1].

Üles-tõusmise ja mine-test on üks kõnniliikuvuse teste, mis algab sellega, et inimene istub toolil, seejärel, pärast terapeudi poolt antud käsklust, tõuseb inimene toolilt püsti, kõnnib kolm meetrit, pöördub tagasi, kõnnib tagasi tooli juurde, pöördub ümber ja istub tagasi toolile (Joonis 1). Testi ajastatud versiooni nimetatakse ka ajastatud üles-tõusmise ja mine-testiks [1].



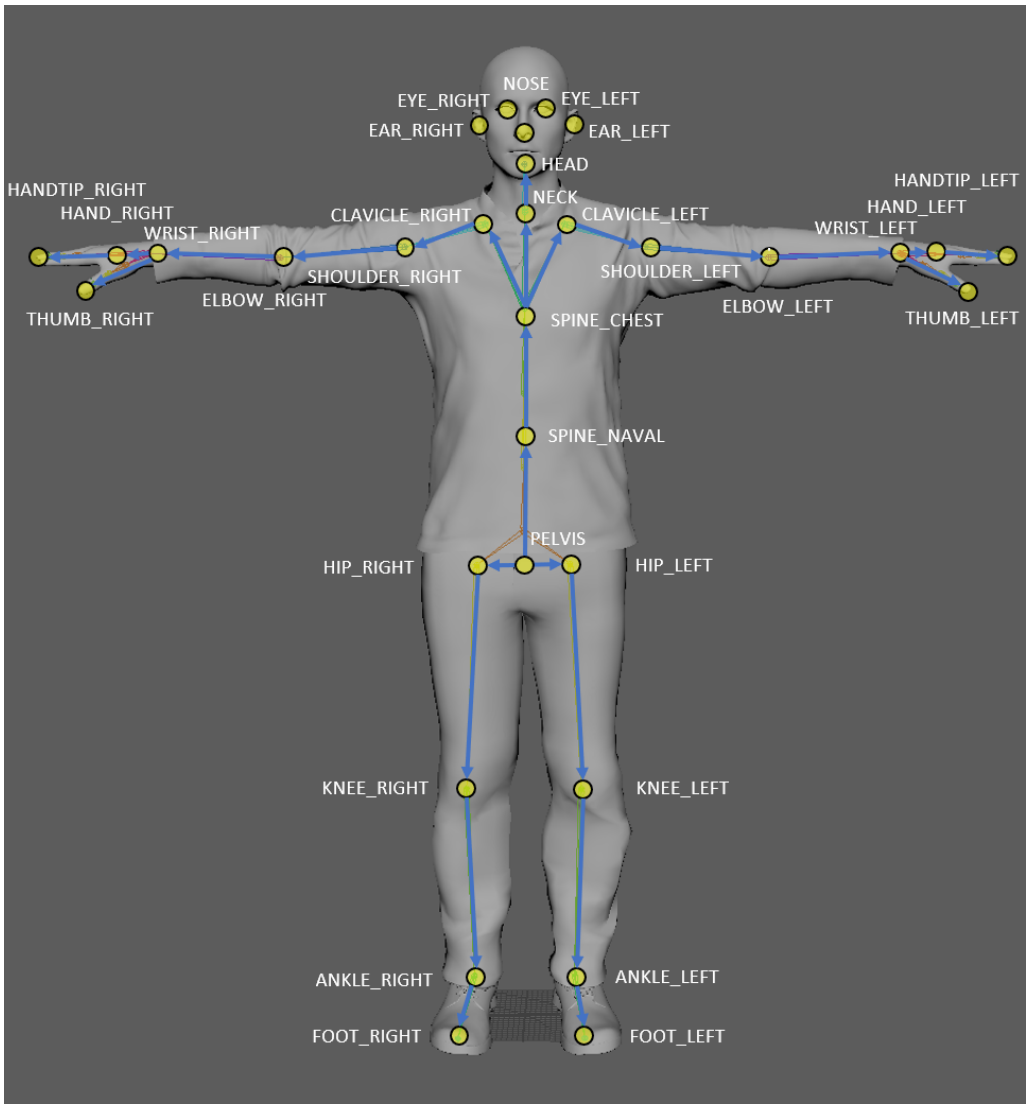
Joonis 1. *Tõuse-ja-mine test [2].*

Seda testi viiakse tavaliselt läbi tervishoiutöötaja, näiteks arsti või füsioterapeudi juuresolekul ruumis.

Kaasaegses kõnnilaboris on liikumise jäädvustamise süsteem, mis koosneb spetsiaalsest arvutist, markeritega kiiretest liikumise jäädvustamise kaameratest, rõhutundlikest plaatidest ja elektromüograafia (EMG) süsteemist. Et saada kogu inimkeha kolmemõõtmelised kineetikad, on vaja umbes 30 markerit. Sellistes laborites läbiviidavate katsete täpsus sõltub markerite paigutuse täpsusest [3].

Nimetatud seadistus on kallis ja seda saab kasutada ainult eriväljaõppe saanud personal. Lisaks on katsed markerite paigutamise tõttu invasiivsed ja aeganõudvad [3].

Aastal 2013 Nõmm jt. tutvustasid liikumismassi (MM) parameetreid, mis arvutatakse liikumise erinevate segmentide jaoks ja mis annavad liikumise kvaliteedist põhjaliku ülevaate. Need hõlmavad aspekte nagu trajektoor, kiirus ja kiirendus. Uuringus kasutati ka Kinecti süsteemi liikumise jäädvustamiseks, mis osutus tõhusaks ja täpseks meditsiinilistes rakendustes, hoolimata selle lihtsusest ja madalast hinnast [4].



Joonis 2. *Kinect'i kehamudel [5].*

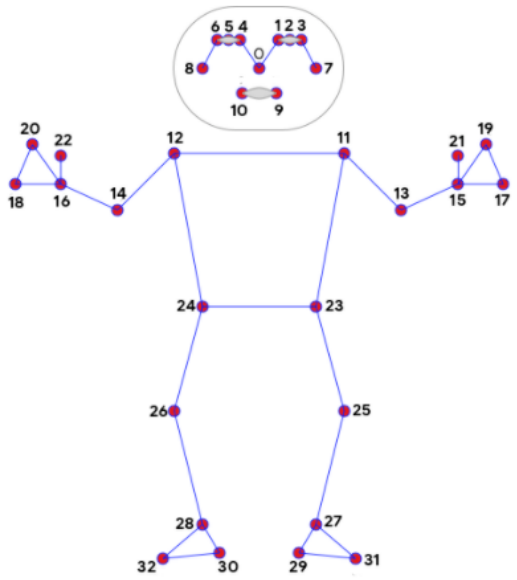
Selle bakalaureusetöö eesmärk on täiustada traditsioonilist ajastatud üles-tõusmise ja mine-testi (TUG), integreerides robottehnoloogiat.

Selles edasiarendatud kohanduses asendab arsti robot, mis on varustatud kaamera ja autonoomse navigatsioonivõimega. Roboti ülesanne on navigeerida iseseisvalt ruumis, kaardistades objekte ja kaugust objektide ja kaamera vahel. Roboti kriitiline funktsionaalsus oleks suutlikkus tuvastada objekte nagu tool ja seejärel määrata, kas objekti ümber on vähemalt kolme meetri suurune takistusteta ruum suvalises suunas. See võimekus on oluline traditsioonilise TUG-testi tingimuste täpseks taasloomiseks automatiseeritud ja loodetavasti täpsemal viisil.

Selline lähenemisviis ei tähtsusta mitte ainult teste automatiseerimist, vähendamaks otsese inimjärelevalve vajadust, vaid suurendab ka testikeskkonna täpsust ja järjepidevust, mis võib viia standardiseeritumate hinnanguteni ja potentsiaalselt suurema skaalatavuseni

erinevates tervishoiuseadetes, võimaldades sagedasi ja süstemaatilisi patsientide liikuvuse hindamisi ilma logistiliste piiranguteta, mis nõuavad iga testi jaoks tervishoiutöötaja kohalolekut. Niisugused tehnoloogilised edusammud võiksid oluliselt mõjutada taastusravi ja geriaatrilise hoolduse valdkonda, kus järjepidevad ja täpsed liikuvushinnangud on hädavajalikud.

Käesolevas lõputöös Kinect'i kaamerat asendab ZED 2 stereokaamera, seega kasutatakse ka MediaPipe teeki, mis asendab Kinect'i sisse ehitatud kehamudeli. Joonis 3 illustreerib teegis kasutatavaid liigeseid. Tabel 4 sisaldab liigese tõlget eesti keelde.



0. nose
1. left_eye_inner
2. left_eye
3. left_eye_outer
4. right_eye_inner
5. right_eye
6. right_eye_outer
7. left_ear
8. right_ear
9. mouth_left
10. mouth_right
11. left_shoulder
12. right_shoulder
13. left_elbow
14. right_elbow
15. left_wrist
16. right_wrist
17. left_pinky
18. right_pinky
19. left_index
20. right_index
21. left_thumb
22. right_thumb
23. left_hip
24. right_hip
25. left_knee
26. right_knee
27. left_ankle
28. right_ankle
29. left_heel
30. right_heel
31. left_foot_index
32. right_foot_index

Joonis 3. *MediaPipe'i kehamudel [6].*

Praegu on olemas liikumislaboris robot, mis suudab ringi sõita. Käesoleva lõputöö eesmärk on arendada komponent, mis annab robotile võimalust:

- Kui inimene on tuvastatud, joonistada inimese peale tema kehamudel;
- Tuvastada ruumis olevaid objekte;
- Vaadata, kas kaardi salvestamine on võimalik;
- Kaardistada ruumis olevaid objekte;
- Salvestada kaardi, kui on võimalik.

2. Probleemi püstitus

Varasemad projektid ja lõputööd [1], [7], [8], [9] on näidanud, et asuvad ruumis objektid (nt toolid, lauad jne) segavad jämemotoorsete teste läbiviimisele, sest nende kohalolu kaadris muudab tulemused ebatäpseks. See asjaolu motiveerib ka käesoleva lõputöö. Põhiidee on kasutada ruumis olevate objektide tuvastamiseks ja võimalusel äratundmiseks kasutusvalmis süvaõppemudeleid, kusjuures pilti või videot jäädvustab ZED-kaamera. Otsida, kas ruumis on inimene, ja hinnata, kas ruumis tuvastatud objektid võivad inimese hetkeasendist 3-meetrise sirge joone (kõnnakuanalüüsiks vajalik kaugus) varjutada. Põhieesmärk on välja töötada süsteemi NVIDIA GPU-ga varustatud arvuti jaoks. Projekti viimases etapis on plaanis hinnata selle portimise võimalust Jetson Orin NANO platvormile.

Kavandatud süsteemi hindamiseks paigaldavad õpilane ja tema juhendaja laboris mööblit ja esemeid ning kasutavad süsteemi ruumis olevate objektide kaardistamiseks. Süsteemi koostatud kaarti võrreldakse algsete asukohtadega. Kui inimese osalus on vajalik, poseerib kaamera ees kas õpilane ise või juhendaja. Seetõttu ei sisalda see uuring inimkatseid ega vaja eetikakomitee luba.

Loodud lahendus peab olema kasulik mitte ainult jäme motoorsete testide valdkonnas, vaid ka mistahes muu valdkonna jaoks, kus on vaja objektide tuvastamist ja kauguse ennustamist.

3. Metoodika

Selle lõputöö raames uuritakse olemasolevaid võimalusi antud probleemi lahendamiseks, millele järgneb tehnoloogiliste vahendite valik edasiseks väljatöötamiseks. Rakenduse nõuete analüüsimisel identifitseeritakse nõudeid, mis peavad olema täidetud käesoleva lõputöö raames.

Sellele järgned rakenduse arendamise osa, kus käsitletakse rakenduse arendust ja käivitamist. Edasi testitakse loodud lahendus, et tagada seda korrekset käitumist erinevate olukordadel. Töö lõpus hinnatakse lõpplahendust kooskõlas eelnevalt määratud nõuetega ning arutletakse võimalused edasiseks arenduseks. Pärast seda koostatakse lõppkokkuvõtte käesoleva bakalauresetööst.

4. Probleemi analüüs

Enne rakendamise arendamist on vaja kindlaks teha, milliseid tehnoloogilisi ja tarkvaralisi võimalusi hakatakse rakenduse arenduse protsessis kasutama. Õigesti valitud tarkvara ja riistvara kombinatsioon tõsiselt lihtsustab rakenduse edasist arendust ning sellepärast on hädavajalik valida tehnoloogiaid, mis ei lähe üksteisega vastuollu.

4.1 Tehnoloogia valik

Selles peatükist vaadeldakse erinevaid riistvara ja tarkvara võimalusi käesoleva lõpitöö eesmärgi saavutamiseks, nimeks objektide tuvastamist ja kauguse määramist. Lisaks sellele, esitatakse selles peatükis teisi kasutatavate tehnoloogiaid.

4.1.1 Programmeerimiskeel

Python on selge ja võimas objektorienteeritud programmeerimiskeel, võrreldav Perl'i, Ruby, Scheme'i või Java'ga [10]. Mõned Pythoni märkimisväärsed omadused on [10]:

- Kasutab elegantset süntaksit, muutes kirjutatud programmid kergemini loetavaks.
- On lihtsalt kasutatav keel, mis muudab programmi tööle saamise lihtsaks. See teeb Pythonist ideaalse keele prototüüpide arendamiseks ja muudeks ad-hoc programmeerimisülesanneteks, säilitades samal ajal hooldatavuse.
- Töötab iga süsteemiga, sealhulgas Mac OS X, Windows, Linux ja Unix süsteemides, mitteametlikud versioonid on saadaval ka Androidile ja iOS-ile.

Mõned Pythoni programmeerimiskeele omadused on [10]:

- Saadaval on mitmesugused põhilised andmetüübid: numbrid (ujuvkomaarvud, kompleksarvud ja piiramatu pikkusega pikad täisarvud), stringid (nii ASCII kui Unicode), loendid ja sõnastikud.
- Python sisaldab arenenud programmeerimisvõimalusi nagu generaatorid ja loendi mõistmised.

- Pythoni automaatne mäluhaldus vabastab teid vajadusest käsitsi mälu eraldada ja vabastada oma koodis.

4.1.2 Operansioonisüsteem

Operatsioonsüsteemiks oli valitud Ubuntu Linux. Ubuntu on iidne Aafrika sõna, mis tähendab "inimlikkust teistele" [11].

Esimese ametliku Ubuntu väljalaske — versioon 4.10, hüüdnimega 'Warty Warthog' — käivitati oktoobris 2004 ja see tekitas tohutut ülemaailmset huvi, kui tuhanded vabavara entusiastid ja eksperdid liitusid Ubuntu kogukonnaga [11].

Täna on Ubuntu'l palju erinevaid maitseid ja kümneid spetsialiseeritud tuletisi. On olemas ka eriväljaanded serveritele, OpenStack pilvedele ja ühendatud seadmetele. Kõik väljaanded jagavad ühist infrastruktuuri ja tarkvara, muutes Ubuntu unikaalseks ühtseks platvormiks, mis ulatub tarbeelektroonikast töölauaarvutiteni ja tõuseb pilve ettevõttearvutusteks [11].

Ubuntu töölaud on maailma kõige laialdasemalt kasutatav Linuxi tööjaamade platvorm, mis toetab inseneride tööd kogu maailmas. Ubuntu Core kehtestab ülimalt turvaliste ühendatud seadmete väikeste tehingutega operatsioonisüsteemide standardi. Ubuntu server on OpenStacki projekti võrdlusoperatsioonisüsteem ja ülipopulaarne külalis-OS AWS-is, Azure'is ja Google Cloudis. Ubuntu on eelinstallitud Delli, HP, Asuse, Lenovo ja teiste ülemaailmsete tarnijate arvutitesse [11].

4.1.3 Arvuti

Arvutiks, mille jaoks rakendust arendada, autor valis Jetson Orin NANO. Jetson Orin NANO arendajakomplekt seab uue standardi algtaseme AI-toega robotite, nutikate droonide ja intelligentsete kaamerate arendamisel, samuti lihtsustab see Jetson Orin Nano seeriaga alustamist [12]. Kompaktne disain, palju pistikuid ja kuni 40 TOPS-i tehisintellekti jõudlust muudavad selle arendajakomplekti ideaalseks visiooniliste ideede reaalsuseks muutmiseks. Eelmise põlvkonna Jetson NANO'ga võrreldes suudab see käitada kõiki kaasaegseid tehisintellekti mudeleid kuni 80-kordse jõudlusega, sealhulgas trafo- ja täiustatud robootikamudeleid [13].

Töö alguses oli otsustatud kasutada Jetson NANO, mis on Jetson Orin NANO eelkäija. Töö

käigus selgus aga ära, et Jetson NANO arendajakomplekt ei ole enam toetatud NVIDIA poolt ning NVIDIA ise soovitab kasutada Orin seeriat [14]. Veel üks põhjus oli sellel, et Jetson NANO vaikimisi Python versioon on 3.6, mis on selle lõputöö kirjutamise hetkel EOL (End Of Life), ja ei ole enam ametlikult toetatud Python'i arendajate poolt [15]. Python'i versioon peaks olema vähem 3.8, sellepärast et mõnede teekide korrektseks ja efektiivsemaks tööks oli vaja uuem Python'i versioon. Jetson Orin NANO vaikimisi Python versioon on 3.8, mis tegi rakenduse arendamist väga otsekoheselt. Samuti töö jooksul oli soovitatud implementeerida inimkeha mudeli joonistamist, ja varasemas lõputöös mil oli sellele pühendatud oli kirjutatud, et Jetson NANO arvutide jõudlus on täpseks 3D-poosi tuvastamiseks liiga madal [9].

Tabel 1. *Jetson Orin NANO vs Jetson NANO [16].*

Parameeter	**Jetson Orin NANO (8GB)**	**Jetson NANO (4GB)**
Python (vaikimisi)	3.8	3.6
Tehisintellekti jõudlus	40 TOPS	umbes 0.5 TOPS
GPU	512-tuumaline NVIDIA Ampere'i arhitektuuriga GPU 16 tensorituumaga	128-tuumaline NVIDIA Maxwell™ arhitektuuriga GPU
CPU	6-tuumaline Arm® Cortex®-A78AE v8.2 64-bit	Quad-Core Arm® Cortex®-A57 MPCore
Energia tarbimine	15W	10W
Mälu	8GB 128-bit LPDDR5, 68 GB/s	4GB 64-bit LPDDR4, 25.6GB/s"

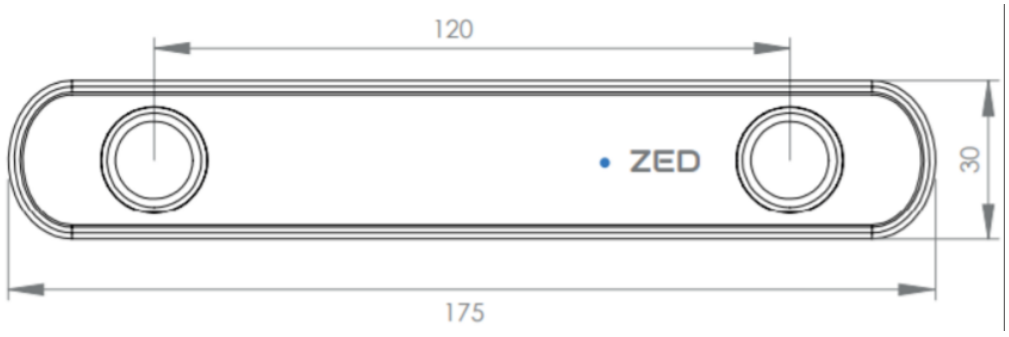
Joonis 4. *Jetson Orin NANO [17].*

Ubuntu Linuxi ja Jetson Orin NANO kombinatsioon pakub mõjuvat lahendust projektidele, mis nõuavad stabiilset, turvalist ja suure jõudlusega andmetöötluskeskkonda. Ubuntu paindlik ja avatud tarkvara ökosüsteem täiendab Orin NANO riistvaravõimekust, pakkudes tugeva aluse uuenduslike rakenduste arendamiseks ja juurutamiseks, mis kasutavad tehisintellekti ja masinõppe tehnoloogiaid. Selline riistvaraplatvormi ja laialdaselt toetatud operatsioonisüsteemi vaheline sünergia on näide tehnoloogiavalikust, mis võib arvutusprojektide arengut ja edu märkimisväärselt soodustada.

4.1.4 Kaamera

Kaameraks oli valitud ZED-kaamera. ZED on passiivne stereokaamera, mis jäljendab inimese nägemise toimimist. Kasutades oma kahte "silmad" ja triangulatsiooni kaudu, loob ZED stseeni, mida ta jälgib, kolmemõõtmelise mudeli, tutvustades esmakordselt sise- ja välistingimustes toimivat pika ulatusega sügavustaju ja positsioneerimise jälgimist [18].

Stereolabsi ZED 2/2i kaamerad on loodud täiustama arvutinägemisrakendusi, pakkudes täpset sügavusteavet. Neil on kaks objektiivi, mis jäädvustavad kõrge eraldusvõimega pilte ja loovad täpseid sügavuskaarte. Need kaamerad võimaldavad objektide täpset tuvastamist. Nende samaaegsed värvi- ja sügavusvõimalused muudavad need mitmekülgseks tehisintellektiga juhitavate ülesannete jaoks, nagu autonoomne navigeerimine, robootika ja liitreaalsus. ZED-kaamerad pakuvad CUDA-kiirendust, võimendades NVIDIA GPU-de võimsust sügavuse arvutamise ja objektide tuvastamise kiirendamiseks, mille tulemuseks on 3D-nägemise ülesannete kiirem ja tõhusam töötlemine. Lihtsalt integreeritavad ja professionaalide poolt laialdaselt kasutatavad ZED-kaamerad on suurepärane valik neile, kes otsivad täiustatud 3D-nägemise ja objektide tuvastamise võimalusi [19]. All on ZED 2 stereokaamera (suurused on millimeetrites).



Joonis 5. *ZED 2 stereokaamera [20].*

ZED kaamera jäädvustab kõrge definitsiooniga 3D videot laia vaateväljaga ja väljastab kaks sünkroniseeritud vasaku ja parema video voogu kõrvuti formaadis USB 3.0 kaudu. Stereolabs pakub mitme sensoriga kaameraid, mis hõlmavad stereonägemist, liikumist,

asendit ja keskkonnasensoreid. ZED API võimaldab madalat juurdepääsu kaamerale ja sensoritele ning hõlbustab kvaliteetse video salvestamist ja voogedastust [21].

ZED sügavuskaamerate pere on mitme sensoriga platvorm. Kaameratel on sisseehitatud sensorid, et lisada sinu rakendusele asendi- ja liikumisabi funktsioone, alates kiirendusmõõtjast ja güroskoobist kuni temperatuuri, baromeetri, magnetomeetri ja rohkemani. Sensoreid saab kasutada kaamera liikumiste tuvastamiseks, kaamera orientatsiooni arvutamiseks vastavalt põhjamagnetpoolusele, suhteliste kõrguse muutuste tuvastamiseks, väliste ilmastikutingimuste analüüsimiseks ja palju muuks [22].

Näiteks üks kasutatavat sensoritest lõputöös on kiirendusmõõtja ja güroskoop (IMU), nimeks andmed `pose` funktsioonist. See funktsioon tagastab IMU asend vabas ruumis, koosneb asendist ja orientatsioonist, ning on saadud kiirendusmõõtja ja güroskoobi sensorite ühendamise teel [22].

ZED-kaamera võimalused, mis olid kasutatud selles bakalaurusetöös:

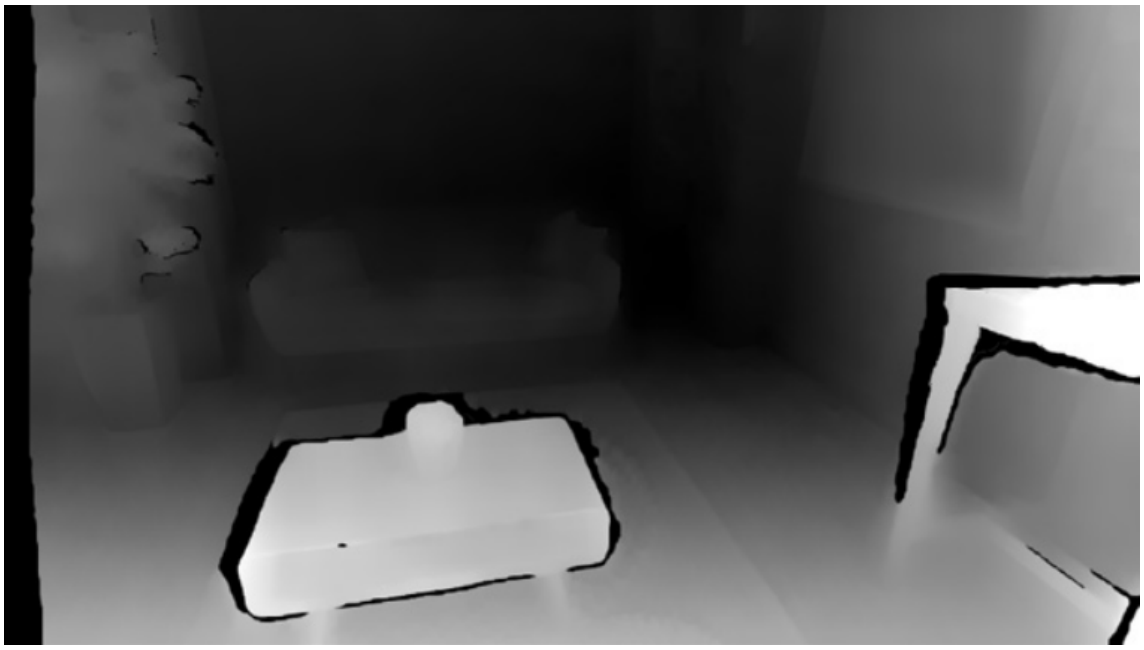
- Sügavuse tajumine: ZED stereokaamera jäljendab, kuidas inimese kahe silma nägemine töötab. Inimese silmad on horisontaalselt eraldatud umbes 65 mm keskmiselt. Seega on igal silmal maailmast veidi erinev vaade. Neid kahte vaadet võrreldes suudab meie aju järeldada mitte ainult sügavust, vaid ka 3D liikumist ruumis. Samamoodi on Stereolabsi stereokaameratel kaks silma, mis on eraldatud 6 kuni 12 cm, võimaldades neil jäädvustada kõrge resolutsiooniga 3D videot stseenist ning hinnata sügavust ja liikumist, võrreldes pikslite nihet vasakul ja paremal pildil [23].

  ZED kasutab sügavuse hindamiseks triangulatsiooni dispariteedipildilt, järgneva valemiga kirjeldades, kuidas sügavuse lahutusvõime muutub kaamera ulatuse piires: $Dr = Z^2 * alpha$, kus Z on kaugus ja alpha konstant.
  Sügavuse täpsus väheneb kvadrantiliselt z-kauguse suhtes, sügavuse täpsusega 1% kaugusest lähivahemikus kuni 9% kaugvahemikus. Sügavuse täpsust võivad mõjutada ka erandid mõõtmistes homogeensetel ja tekstuurita pindadel, nagu valged seinad, rohelised ekraanid ja peegelduvad alad. Need pinnad tekitavad tavaliselt ajutist ebastabiilsust sügavuse mõõtmistes [24].

  Sügavuskaart: ZED poolt püütud sügavuskaardid salvestavad iga pildi piksli (X, Y) jaoks kauguse väärtuse (Z). Kaugus on väljendatud meetermõõdustikus (näiteks meetrites) ja arvutatud kaamera vasaku silma tagant stseeni objektini. Sügavuskaarte ei saa otse kuvada, kuna need on kodeeritud 32 bitile. Sügavuskaardi kuvamiseks on

vajalik monokroomne (halltoonides) 8-bitine esitus väärtustega vahemikus [0, 255], kus 255 esindab lähimat võimalikku sügavusväärtust ja 0 on kõige kaugem võimalik sügavusväärtus [23].

- Objektide tuvastamine: Objektide tuvastamine on võime tuvastada pildil olevaid objekte. Tänu sügavustajule ja 3D informatsioonile suudab ZED kaamera pakkuda stseenis olevate objektide 2D ja 3D asukohti. ZED SDK kasutab tehisintellekti ja närvivõrke, et määrata, millised objektid on olemas nii vasakus kui ka paremas pildis. SDK seejärel arvutab iga objekti 3D asukoha, samuti nende piirdekarbi, kasutades andmeid sügavusmoodulist [25].



Joonis 6. *ZED kaamera sügavuskaarti representatsioon [23].*

4.1.5 Versioonihaldussüsteem

Versioonihaldussüsteemi valikul oli oluline leida lahendus, mis pakub tasuta võimalusi ning on laialdaselt kasutusel, olles toetatud rohkete õppematerjalidega. Lähtudes nendest kriteeriumitest, oli autoril kaks peamist valikut - GitHub ja GitLab.

- GitHub: Seda töötasid 2008. aasta veebruaris välja Chris Wanstrath, Scott Chacon, Tom Preston-Werner ja P. J. Hyett, kasutades Ruby on Rails (RoR). Võttes kasutusele esmaliikujate eelised, sai see paljude avatud lähtekoodi hoidlate pesaks. GitHub on ajalooliselt keskendunud rohkem koodihostimisele ja koostööle, kuid on hiljuti lisanud selliseid funktsioone nagu CI/CD töövood ja projektihaldustööriistad. Olles 2020. aastal maailmas suurim lähtekood, on GitHub Git-põhine hoidlate

hostimisplatvorm, mis koosneb enam kui 40 miljonist kasutajast. GitHubi abil saate oma projektid avalikuks teha. Seega võib iga avalikult jagatud kood olla kõigile avatud [26].

- GitLab: Käivitati hiljem. Selle töötasid välja Ukraina arendajad Valeri Sizov ja Dmitri Zaporožets 2011. aastal. GitHub oli nutikalt loodud projekti koostöötööriistade ja koodihoidla teenuste jaoks. GitLab on kõik-ühes DevOpsi platvorm, mis ei sisalda mitte ainult Giti hoidla hostimist, vaid ka projektihaldust, CI/CD torujuhtmeid, probleemide jälgimist ja palju muud. GitLab on pilvepõhine Giti hoidla ja DevOpsi platvorm, mis muudab arendajatele koodi testimise, jälgimise ja juurutamise sujuvaks. Algselt oli GitLabi peamine USP pilvepõhine Giti hoidla. Järk-järgult on see jõuline arendusplatvorm arenenud palju rohkem kui selle päritolu [26].

Selle bakalaurusetöö raames oli otsustatud kasutada GitLab'i sellepärast, et see on ülikooli peamine versioonihaldussüsteem ning nii juhendajale kui ka tudengile on lihtsam ja tõhusam kasutada GitLab'i. Kuid varundamiseks otsustati ka projekt GitHub'sse üles laadida.

4.1.6 Raamistiku valik

Selle projekti raames oli autoril kaks peamist konkurenti – TensorFlow ja PyTorch.

- PyTorch: Käivitati 2016. aastal ja selle põhimõte on hoida API lihtsana, võimaldades seda kiiresti muuta ja hoida kooskõlas uusimate suundumustega tehisintellekti valdkonnas.
  Seda tehakse kolme põhiprintsiibi järgi, millest esimene on meetod, kuidas selle funktsioonid määratletakse. Raamistik defineerib kõik oma komponendid "pythonic" viisil, eesmärgiga muuta see kergesti kasutatavaks kasutajatele, kes juba tunnevad Python programmeerimiskeelt. Samas lihtsuse kontekstis on komponentide määratlemine liideste kaudu PyTorchi teine põhimõte. Sel viisil peidetakse komponentide määratlemise ja kasutamise keerukus lihtsate initsialiseerimiste ja funktsioonikutsete taha, mis aitab tuvastada põhilisi närvivõrgu kontseptsioone ning lihtsustab seeläbi õppimist. Raamistiku viimane põhimõte on, et lihtne disain on parem, isegi kui osa jõudlusest ohverdatakse, sest keerukam disain võib pikemas perspektiivis muuta integreerimise keerulisemaks ja vähendada jõudlust. Sellel lähenemisel on ka lisakasu: lihtsam hooldus võimaldab kiiremini reageerida võimalikele probleemidele, mis võivad ilmneda nii raamistikus kui ka väljaspool seda [27].

- TensorFlow: Esimene versioon ilmus 2015. aastal ning selle eesmärk oli koondada mitmed teegid ühtseks paketiks, mida saaks kasutada erinevate masinõppe probleemide lahendamiseks. Kogenud kasutaja jaoks pakkus raamistik kõiki vajalikke tööriistu, kuid mitmete teekide ühendamine tähendas ka seda, et mõned koostoimed olid keerulisemad ja raskemini mõistetavad. Sellise olukorra näiteks oli vajadus luua eraldi koodid võrgu treenimiseks, mis muutis vigade leidmise ja parandamise raskemaks. Seetõttu peeti raamistikku algselt mitte eriti kasutajasõbralikuks, kuid versiooniga 2.0, mis ilmus 2019. aastal, lahendati varasemad puudused. Kõigi muudatuste tulemusena sai TensorFlow 2.0-st kogukonna kogenud ja kogenematute liikmete seas üsna populaarne valik. See pakkus väga head jõudlust ja lihtsasti kasutatavat liidest, mis aitas oluliselt vähendada õppimiskõverat [27].

Tabel 2. *Tensorflow vs PyTorch [27]*

Kriteerium	**TensorFlow**	**PyTorch**
Kasutajasõbralikkus	Halvem	Parem
Dokumentatsiooni tugi	Sama	Sama
Treeningu ja soorituse ajad	Halvem	Parem
Täpsus treeningu ja soorituse ajal	Parem	Halvem

PyTorch on suunatud rohkem "disainifaasile" ja sellel on kasutajasõbralikum liides. Lisaks on seda lihtsam lahendustesse integreerida tänu ühele käsule installimiseks, väliste sõltuvuste puudumisele ja veidi madalamale täpsusele treeningu ajal võrreldes TensorFlow'ga, kahe teegi vahelise 1,16% erinevusega. Seetõttu sobib PyTorch paremini õppimiseks või ülesanneteks, kus kiirus on prioriteedis, kuna sellel on lihtsam paigaldusprotsess, kergemini järgitav andmevoog ja üldiselt lihtsam integreerimine, pakkudes samas 25,5% kiiremat treeningukiirust ja 77,7% kiiremat sooritust [27].

Selles lõputöö raames otsustas autor kasutada PyTorch'i sellepärast, et see on rohkem Python-i sõbralik, ja sellel on hea dokumentatsioon ja kogukonna tugi, samuti saab YOLO algoritmi lihtsasti hõlpsastada PyTorch'i abil.

4.1.7 Piltide töötlemise teekid

Kuna video on vaid kaadrite jada ja iga kaader on pilt, on vaja valida sobiva piltide töötlemise teeki. Enimlevinud teekid piltide töötlemiseks on [28]:

- OpenCV (Open Source Computer Vision Library): Avatud lähtekoodiga masinõppe

ja arvutinägemise tarkvara raamatukogu. OpenCV loodi arvutinägemisrakenduste ühise infrastruktuuri pakkumiseks ja masina tajumise kiirendamiseks kommertstoodetes. Kuna OpenCV on Apache 2 litsentsitud toode, teeb see ettevõtetele koodi kasutamise ja muutmise lihtsaks [29].

- Scikit-Image (endine scikits.image): Pythoni programmeerimiskeele avatud lähtekoodiga pilditöötlusteek. See sisaldab algoritme segmenteerimiseks, geomeetrilisteks teisendusteks, värviruumi manipuleerimiseks, analüüsiks, filtreerimiseks, morfoloogiaks, funktsioonide tuvastamiseks ja muuks. See on loodud koostoimima Pythoni numbri- ja teadusraamatukogudega NumPy ja SciPy [30].

- Pillow/PIL: See Pythoni pilditeek lisab Pythoni interpretaatorile pilditöötlusvõimalused. Teek pakub ulatuslikku failivormingu tuge, tõhusat sisemist esitust ja üsna võimsaid pilditöötlusvõimalusi. Põhipilditeek on loodud kiireks juurdepääsuks mõnes põhipikslivormingus salvestatud andmetele. See peaks andma kindla aluse üldisele pilditöötlustööriistale [31].

- NumPy: Pythoni teek, mida kasutatakse massiividega töötamiseks. Sellel on ka funktsioonid lineaaralgebra, Fourier teisenduse ja maatriksite valdkonnas töötamiseks. NumPy lõi 2005. aastal Travis Oliphant. See on avatud lähtekoodiga projekt ja saab seda vabalt kasutada. NumPy tähistab Numerical Python. Pythonis on loendid, mis teenivad massiivide eesmärki, kuid nende töötlemine on aeglane. NumPy eesmärk on pakkuda massiiviobjekti, mis on kuni 50 korda kiirem kui traditsioonilised Pythoni loendid. NumPy massiiviobjekti nimetatakse ndarrayks, see pakub palju tugifunktsioone, mis muudavad ndarrayga töötamise väga lihtsaks. Massiive kasutatakse väga sageli andmeteaduses, kus kiirus ja ressursid on väga olulised [32].

Kokkuvõttes, NumPy on hea alustamiseks ja madala taseme piltide manipuleerimiseks, Scikit-Image on suurepärane teaduslikuks pilditöötluseks, kui on vaja valmis algoritme, OpenCV on parim valik keerukate arvutinägemise projektide jaoks, kus on vajalik kiirus ja lai funktsionaalsus ja Pillow on ideaalne lihtsamateks piltide töötlemise ülesanneteks, kui ei ole vaja laiaulatuslikke arvutinägemise võimalusi.

Autor otsustas kasutada oma bakalaurusetöös OpenCV, sellepärast et see sobib kõige paremeni arvutinägemiseks. Lisaks sellele, OpenCV teegil on suur kogukond, mis teeb materjalide leidmist lihtsamaks. Samuti NumPy oli selles bakalauresetöös kasutatud.

4.1.8 Objektide tuvastamise algoritmid

Järgnevalt on välja toodud raamistikud arvutinägemise teostamiseks ja objektide tuvastamiseks ning sellega kooskõlastavad tehnoloogiad.

Süvaõpe

Süvaõpe on masinõppe alamvaldkond, mis kasutab õppimiseks ja suurte andmemahtude põhjal prognooside tegemiseks mitmekihilisi tehisnärvivõrke ehk ANN-e (Artificial Neural Network). Süvaõppes konstrueeritakse ANN-id mitme kihiga omavahel ühendatud sõlmedest või neuronitest, kus iga neuron teeb oma sisenditega lihtsa matemaatilise toimingu ja edastab tulemuse järgmisele kihile. Kihid on virnastatud üksteise peale, moodustades sisendandmete hierarhilise esituse [33].

Süvaõpe koosneb erinevatest kihtidest ja igal kihil on oma kindel ülesanne [33]:

- Normaliseerimiskiht: kiht skaleerib sisendandmed sobivate intervallidega. Normaliseerimiskihiga muudetakse mudel stabiilsemaks ja parandatakse jõudlust [33].

- Konvolutsioonikiht: kiht rakendab sisendile filtreid. Filtrite abil tõstetakse esile sisendi olulised omadused. Kiht on väga kasulik näiteks kujutiste klassifitseerimises ja objektide tuvastamises [33].

- Koondamiskiht: kiht vähendab funktsioonide tundlikkust ja säilitades kõige silmatorkavamad funktsioonid ja omadused. Selle tulemusel saavutatakse paremad ja üldisemad andmeid. Samuti vähendatakse tunnuskaartide suurust [33].

- Aktiveerimiskiht: kiht rakendab sisendkihile matemaatilisi funktsioone f(x). Samuti tutvustatakse mudelile mittelineaarsust, võimaldades mudelil õppida keerukamaid seoseid sisendite ja väljundite vahel [33].

- Ahenduskiht: kiht on kasutatud andmemahtude ja parameetrite kokkusurumiseks, et vähendada ülesobitumist. Kui sisendiks on pilt, siis koonduva kihi peamine ülesanne on pildi kokkusurumine [34].

All on enim kasutatud objektide tuvastamise algoritmid [35].

YOLO (You Only Look Once)

YOLO on üks populaarsemaid mudeliarhitektuure ja -algoritme objektide tuvastamiseks. Nimetus viitab sellele, et see suudab tuvastusülesande täita võrgustiku ühe läbimisega, erinevalt varasematest lähenemistest mis pidid ühe pildi kohta töötlema sadu või tuhandeid kordi [36]. YOLO-st on mitu versiooni [37]:

1. 2016. aastal välja antud YOLOv2 täiustas algset mudelit, lisades partii normaliseerimise, ankurduskastid ja mõõtmete klastrid.

2. 2018. aastal käivitatud YOLOv3 parandas veelgi mudeli jõudlust, kasutades tõhusamat põhivõrku, mitut ankrut ja ruumilise püramiidi ühendamist.

3. YOLOv4 ilmus 2020. aastal, tutvustades selliseid uuendusi nagu Mosaici andmete suurendamine, uus ankruvaba tuvastuspea ja uus kadufunktsioon.

4. YOLOv5 parandas veelgi mudeli jõudlust ja lisas uusi funktsioone, nagu hüperparameetrite optimeerimine, integreeritud katsete jälgimine ja automaatne eksport populaarsetesse ekspordivormingutesse.

5. YOLOv6 sai avatud lähtekoodiga Meituan 2022. aastal ja seda kasutatakse paljudes ettevõtte autonoomsetes tarnerobotites.

6. YOLOv7 lisas COCO võtmepunktide andmekogus täiendavaid ülesandeid, näiteks positsioneerimise hindamine.

7. YOLOv8 on Ultralyticsi YOLO uusim versioon. Tipptasemel tipptasemel (SOTA) mudelina toetub YOLOv8 eelmiste versioonide edule, tuues sisse uusi funktsioone ja täiustusi, et suurendada jõudlust, paindlikkust ja tõhusust. YOLOv8 toetab kõiki nägemis-AI-ülesandeid, sealhulgas tuvastamist, segmenteerimist, poosi hindamist, jälgimist ja klassifitseerimist. See mitmekülgsus võimaldab kasutajatel kasutada YOLOv8 võimalusi erinevates rakendustes ja domeenides.

8. YOLOv9 tutvustab uuenduslikke meetodeid, nagu programmeeritav gradiendi teave (PGI) ja üldine tõhus kihtide koondamise võrk (GELAN) [37].

Region-based Convolutional Neural Networks (R-CNN)

Aastal 2014 pakkus Ross Girshick välja R-CNN-i, mis kasutab libiseva akna asemel selektiivse otsingu algoritmi, lahendades akna liigse korduvuse probleemi ja vähendades algoritmi ajalise keerukuse. R-CNN mudeli jõudlus on võrreldes traditsiooniliste objektituvastusalgoritmidega oluliselt paranenud, kuid esineb ka palju probleeme. Näiteks genereerib R-CNN umbes 2000 kandidaatregiooni ja omaduste ekstraktsioon võtab liiga palju aega; konvolutsioonilised närvivõrgud nõuavad fikseeritud suurusega sisendit ja pildi kärpimine või venitamine põhjustab pildiinfo kaotust; koolituskiirus on samuti aeglane [34].

Fast R-CNN

Aastal 2015 avaldatud Fast R-CNN'i võrreldes Fast R-CNN'i ja R-CNN'i raamistikutega, võib leida kaks peamist erinevust: esimene on see, et viimasele konvolutsioonikihile on lisatud ROI ahenduskiht ja teine on see, et kahjufunktsioon kasutab mitmeülesandelise kahjufunktsiooni. Fast R-CNN kasutab kogu pildi omaduste väljavõtmiseks esiteks CNN-võrku, mitte ei ekstrakteeri omadusi mitu korda iga pildibloki jaoks. Seejärel saab kandidaatregioonide loomise meetodit rakendada otse ekstraheeritud omaduste kaartidele [34].

Faster R-CNN

Fast R-CNN-i probleemiks on see, et selektiivne otsing kõikide kandidaatkastide leidmiseks on väga aeganõudev. Selleks, et saada neid kandidaatkastid efektiivsemalt, on Faster R-CNN lisanud neuraalvõrgu piirkonnapakkumise võrgu RPN (Region Proposal Network), mis ekstraheerib servi. See aitas vähendada pakutavate piirkondade arvu, kuid säilitada siiski objektituvastuse täpsus [34].

Single Shot Detector (SSD)

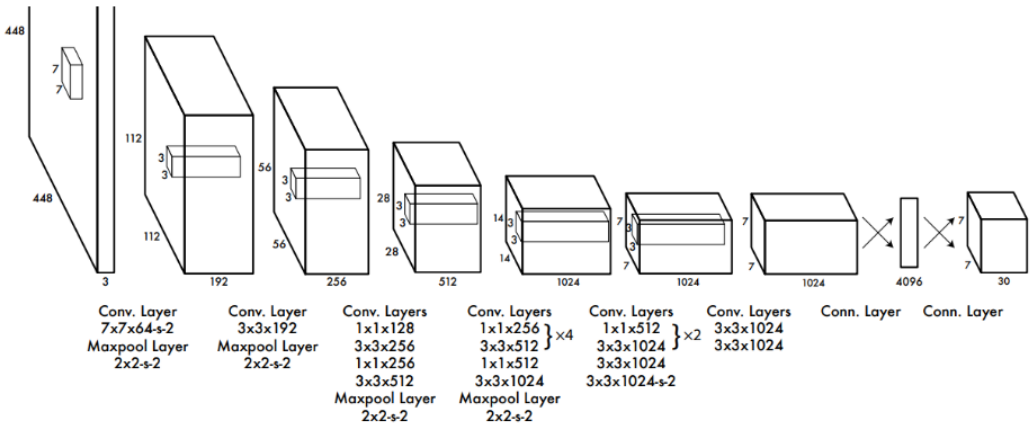
SSD on kiire objektituvastaja mitmele kategooriale, mis läbib närvivõrgu ainult üks kord. Mudeli põhifunktsiooniks on mitmes mõõtkavas konvolutsiooniliste piirdekarbikeste väljundite kasutamine, mis on ühendatud mitme omaduste kaardiga võrgustiku tipus. See esitusviis võimaldab tõhusalt modelleerida võimalike kastikujude ruumi [38].

4.1.9 YOLOv8

Ülaltoodud algoritmitest oli otsustatud kasutada YOLOv8, sest Ultralyticsi YOLO üks uuemaid versiooni tipptasemel (SOTA) mudelina tugineb YOLOv8 eelmiste versioonide edul, tuues sisse uusi funktsioone ja täiustusi, et suurendada jõudlust, paindlikkust ja tõhusust.

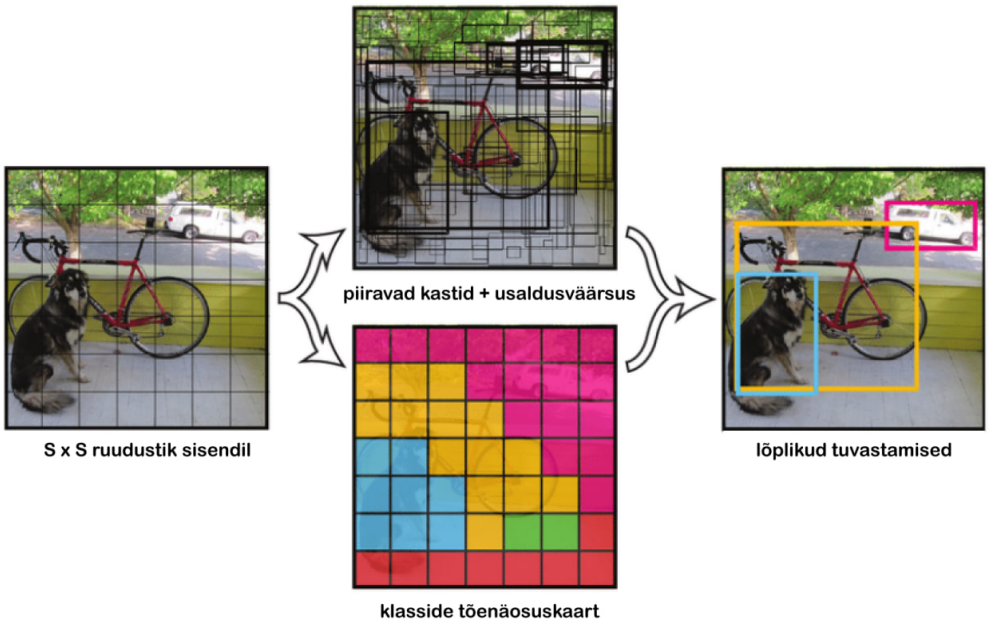
YOLOv8 toetab kõiki nägemis-AI-ülesandeid, sealhulgas tuvastamist, segmenteerimist, poosi hindamist, jälgimist ja klassifitseerimist. See mitmekülgsus võimaldab kasutajatel kasutada YOLOv8 võimalusi erinevates rakendustes ja domeenides [37]. Kõik YOLOv8 mudelid suudavad tuvastada 80 erinevat objektide klasse, mis on toodud COCO datasetis [39], [40].

Ka YOLO kasutab objektide tuvastamiseks ja klassifitseerimiseks ühte närvivõrku. See erineb teistest objektituvastussüsteemidest, mis kasutavad objektide tuvastamiseks ja klassifitseerimiseks tavaliselt mitut närvivõrku. See lähenemisviis võimaldab YOLO süsteemil saavutada head täpsust, muutes selle ideaalseks rakenduste jaoks, kus kiirus on kriitiline [33]. Joonisel 7 tähendab *Conv. Layer* konvolutsionikihti ning *Maxpool Layer* maksimaalset ahenduskihti.



Joonis 7. *YOLO algoritmi arhitektuur [33].*

YOLO algoritm koosneb peamiselt konvolutsioonikihtidest. Konvolutsioonikiht rakendab sisendile filtreid, mille abil tõstetakse esile sisendi olulised omadused. Kiht on väga kasulik, näiteks kujutiste klassifitseerimises ja objektide tuvastamises [33].



Joonis 8. *YOLO algoritmi visuaalne representatsioon [41].*

YOLOv8 mudelid

All on YOLOv8 erinevate mudelite võrdlus.

Tabel 3. *YOLOv8 mudelite võrdlus [42].*

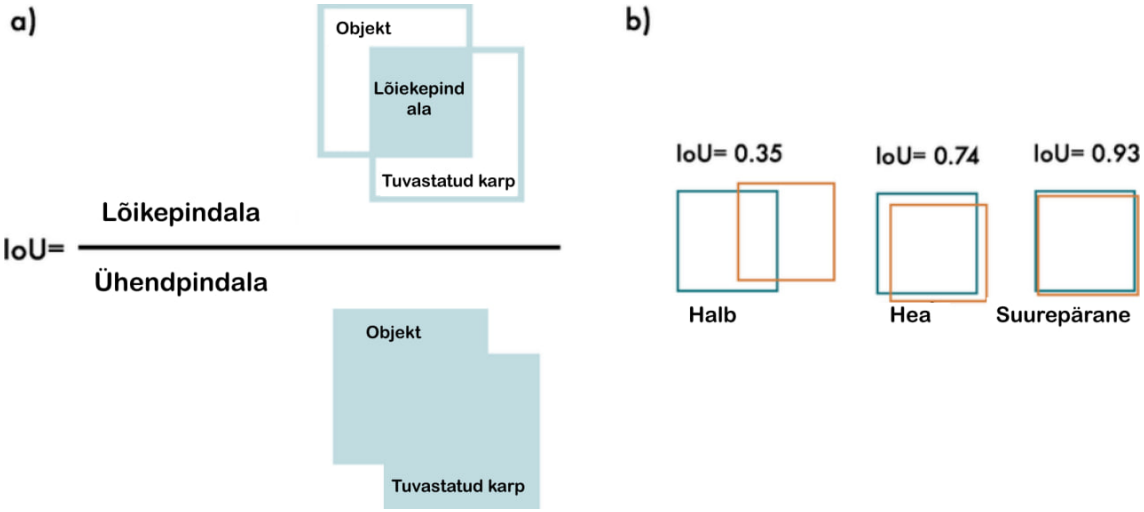
Mudel	**Suurus**	**mAP^{val}, (50-95)**	**Kiirus (CPU ONNX), ms**	**Kiirus (A100 TensorRT), ms**	**params (M)**	**FLOPs (B)**
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

- mAP^{val} (50-95) väärtused on üksikmudeli ühe skaala jaoks COCO val2017 andmekogul. Kordatakse käskuga: `yolo val detect data=coco.yaml device=0` [42].

- Kiirus arvutatud keskmiselt COCO val piltide pealt, kasutades Amazon EC2 P4d instantsi. Kordatakse käskuga: `yolo val detect data=coco128.yaml batch=1 device=0|cpu` [42].

Mõõteväärtus mAP^{val} (50-95) tähistab keskmist täpsust, mis on arvutatud erinevate IoU (Intersection over Union) lävendite järgi vahemikus 0,5 kuni 0,95, sammuga 0,05 (0,5, 0,55, 0,6, 0,65, 0,7, 0,75, 0,8, 0,85, 0,9, 0,95) [43].

IoU on ennustatud piirdekarbi ja tegeliku piirdekarbi lõikepindala ja ühendpindala suhe (Joonis 9). See mõõdab tegelike ja ennustatud piirdekarpide kattuvust. COCO võrdlusalus kasutab mudeli jõudluse hindamiseks mitmeid IoU lävendeid erinevatel lokaliseerimise täpsustasemetel [36].



Joonis 9. *YOLO Lõikepindala suhe ühendpindalasse [36].*

Mõõteväärtuse mAP^{val} (50-95) toimub järmiselt – esiteks määrab täpsus positiivse ennustusväärtuse, mida saab arvutada valemiga [43]:

$$Tapsus = \frac{T_p}{T_p + V_p}$$

Mudelite puhul uuriti lisaks täpsusele ka saagist. Saagis, tuntud ka kui tundlikkus, esindab õigete positiivsete ennustuste suhet ja seda arvutatakse valemiga [43]:

$$Saagis = \frac{T_p}{T_p + V_n},$$

kus T_p tähistab tõest positiivset, T_n viitab tõesele negatiivsele ning V_p ja V_n tähistavad vastavalt valet positiivset ja valet negatiivset.

Selles lõputöös otsustati kasutada YOLOv8m, kuna see on kompromiss kõige täpsema ja kiireima versiooni vahel. YOLOv8x või YOLOv8l kasutamine oleks Jetson Orin NANO jaoks tarbetu koormus.

4.1.10 Teised kasutatud teekid

Selles bakalauruse töös on autor otsustanud kasutada ka järgmisi teeke:

- `pyzed.sl`: On osa Stereolabs ZED kaamera vajalikke tööriste, nimeks ZED Python API. Pythoni API on ZED SDK ümbris, mis on kirjutatud C++ optimeeritud koodis. ZED SDK tehaks juurdepääsetavaks välisest Pythoni koodist, kasutades Cythoni [44]. Cython on optimeeriv staatiline kompilaator nii Pythoni programmeerimiskeele kui ka laiendatud Cythoni programmeerimiskeele jaoks (põhineb Pyrexil). See teeb Pythoni jaoks C-laiendite kirjutamise sama lihtsaks kui Python ise [45].

- `threading.Lock`: Primitiivseid lukuobjekte rakendav klass. Kui lõime on lukustunud, blokeerivad järgmised katsed seda omandada, kuni see vabastatakse; iga lõng võib selle vabastada. Lukuobjektis on kaks peamist meetodit: `acquire()` ja `release()` [46].

    1. `acquire(blocking=True, timeout=-1)`: Hangitakse lukk, blokeeriv või mitteblokeeriv. Kui käivitatakse blokeerimisargumendi väärtuseks `True` (vaikeväärtus), blokeeritakse seni, kuni lukk avatakse, seejärel määratakse see lukustatud ja tagastatakse väärtus `True`. Kui käivitatakse blokeerimisargumendi väärtuseks `False`, seda ei blokeeri. Kui kõne blokeerimise väärtuseks on `True`, blokeeritakse ja tagastatakse kohe `False`; muul juhul seatakse lukk olekusse lukustatud ja tagastatakse `True`. Kui käivitatakse ujukoma ajalõpu argumendiga, mis on seatud positiivsele väärtusele, blokeeritakse maksimaalselt nii palju sekundeid, mis on määratud ajalõpuga, ja seni, kuni lukku pole võimalik hankida. Ajalõpu argument -1 määrab piiramatu ootamise. Kui blokeerimine on `False`, ajalõpu määramine on keelatud. Tagastusväärtus on `True`, kui lukk on edukalt omandatud, `False`, kui mitte (näiteks kui ajalõpp on aegunud) [46].

    2. `release()`: Vabastatakse lukk. Seda saab kutsuda mis tahes lõimest, mitte ainult lukust saanud lõimest. Kui lukk on lukus, lähtestatakse see lukustamata ja pööratakse tagasi. Kui mõni muu lõime on blokeeritud, oodates luku avamist, lastakse täpselt ühel neist jätkata. Kui see käivitatakse lukustamata luku korral, kuvatakse `RuntimeError`. Tagastusväärtust pole [46].

- `threading.Thread()`: Seda konstruktorit tuleks alati kutsuda märksõnaargumentidega. Argumendid on: `group` peaks olema `None`; reserveeritud tulevaseks laiendamiseks, kui `ThreadGroup` klass on rakendatud. `target` on kutsutav

objekt, mida käivitab meetod `run()`. Vaikimisi puudub, mis tähendab, et midagi ei kutsuta. `name` on lõime nimi. `args` on sihtkutse argumentide loend või rida. Vaikimisi (). `kwargs` on märksõnaargumentide sõnastik sihikutse jaoks. Vaikimisi on tühi sõnastik. Kui mitte `None`, määrab daemon selgesõnaliselt, kas lõim on daemonlik. Kui `None` (vaikimisi), päritakse daemon atribuut praegusest lõimest [46]. Arvutustehnikas on daemon (hääldatakse DI-muhn) programm, mis töötab pidevalt taustaprotsessina ja ärkab üles perioodiliste teenusepäringute käitlemiseks, mis sageli tulevad kaugprotsessidest [47].

1. `start()`: Käivitatakse lõime tegevus. Seda tuleb ühe lõimeobjekti kohta kutsuda maksimaalselt üks kord. See korraldab objekti `run()` meetodi käivitamise eraldi juhtlõimes. See meetod tekitab `RuntimeErrors`, kui seda sama lõimeobjekti puhul rohkem kui üks kord kutsutakse [46].

2. `run()`: Lõime tegevust kirjeldav meetod. On võimalik selle meetodi alamklassis alistada. Standardne meetod `run()` kutsub esile kutsutava objekti, mis edastatakse objekti konstruktorile sihtargumendina, kui see on olemas, koos asukoha- ja märksõnaargumentidega, mis on võetud vastavalt argumentidest `args` ja `kwargs`. Sama efekti võib saavutada loendi või korteeži kasutamise arg-argumendina, mis edastati lõimele [46].

- `time`: See moodul pakub erinevaid ajaga seotud funktsioone. Kuigi see moodul on alati saadaval, pole kõik funktsioonid kõigil platvormidel saadaval. Enamik selles moodulis määratletud funktsioone kutsuvad platvormi C teegid sama nimega. Mõnikord võib olla kasulik tutvuda platvormi dokumentatsiooniga, kuna nende funktsioonide semantika on platvormidel erinev [48].

  1. `sleep()`: Peatatakse kutsuva lõime täitmine etteantud sekundite arvuks. Argumendiks võib olla ujukomaarv, mis näitab täpsemat uneaega. Kui unerežiim katkestatakse signaaliga ja signaali töötleja ei tee erandit, alustatakse unerežiimi uuesti arvutatud ajalõpuga. Peatusaeg võib olla suvalise summa võrra pikem, kui nõutud, kuna süsteemis on ajastatud muud tegevused [48].

- `argparse`: Moodul argparse teeb kasutajasõbralike käsurea liideste kirjutamise lihtsaks. Programm määrab, milliseid argumente see nõuab, ja argparse selgitab välja, kuidas need failist `sys.argv` välja sõeluda. Moodul argparse genereerib automaatselt ka abi- ja kasutusteateid. Moodul annab tõrkeid ka siis, kui kasutajad annavad programmile kehtetuid argumente [49].

- `matplotlib.pyplot`: Matplotlib on raamistik Pythonis staatiliste, animeeritud ja interaktiivsete visualisatsioonide loomiseks [50]. `matplotlib.pyplot` on matplotlibi olekupõhine liides. See pakub kaudset, MATLAB-i sarnast joonistusviisi. Samuti avab see teie ekraanil figuurid ja toimib jooniste GUI haldurina. `pyplot` on mõeldud peamiselt interaktiivsete graafikute jaoks ja lihtsate programmiliste süžeede genereerimise juhtumite jaoks [51].

- `mediapipe`: MediaPipe on lihtsaim viis teadlastele ja arendajatele maailmatasemel ML lahenduste ja rakenduste loomiseks mobiilsetele seadmetele, äärealadele, pilvele ja veebile [52]. MediaPipe Pose Landmarker ülesanne võimaldab tuvastada inimkehade maamärke pildil või videos. Seda ülesannet saab kasutada oluliste keha asukohtade tuvastamiseks, kehahoiaku analüüsimiseks ja liigutuste kategoriseerimiseks. See ülesanne kasutab masinõppe (ML) mudeleid, mis töötavad üksikute piltide või videoga. Ülesanne väljastab keha poosi maamärke pildi koordinaatides ja 3-mõõtmelistes maailma koordinaatides [53].

- `pyinstaller`: PyInstaller pakendab Pythoni rakenduse ja kõik selle sõltuvused ühte paketti. Kasutaja saab pakendatud rakendust käivitada ilma Pythoni tõlgendajat või mingeid mooduleid installimata. PyInstaller toetab Python 3.8 ja uuemaid versioone ning pakendab õigesti paljud peamised Pythoni paketid nagu numpy, matplotlib, PyQt, wxPython ja teised. PyInstallerit testiti Windowsi, MacOS X-i ja Linuxi platvormidel. Siiski ei ole see ristkompilaator; Windowsi rakenduse loomiseks käivitate PyInstalleri Windowsis ja Linuxi rakenduse loomiseks käivitate selle Linuxis jne [54].

5. Rakenduse nõuete analüüs

Sellessamas peatükis käsitletakse arendatava süsteemi nii funktsionaalseid kui ka mitte-funktsionaalseid nõudeid.

5.1 Nõuete määramine

Nõuete määramisel lähtuti peamiselt sellest, et loodakse prototüüpi, mille peamine eesmärk on uurida ja demonstreerida parimaid tehnoloogilisi lahendusi.

5.1.1 Funktsionaalsed nõuded

Funktsionaalsed nõuded on järgmised:

- Kaamera suudab tuvastada objekti;
- Kaamera abil suudakse tuvastada distantsi objekti ja kaamera vahel;
- Ekraanis on võimalik näha objekti nimi ja distantsi kaamera ja objekti vahel;
- Kui suvalise objekti ja inimese objekti vahel distants on vähem kui 3 meetrit, näidatakse ekraanil hoiatus;
- Kui kaamera ei tuvastaks inimest enne oma töö lõpetamist, on vaja salvestada kaardi objektidest ruumis;
- Kui kaamera ei tuvastaks inimest enne oma töö lõpetamist, on vaja salvestada objektide nimikirja, kus on iga objekti nimetus ja kooridnaadid;
- Kui inimene on tuvaastatud, peab rakendus tema peale joonistada kehamudeli;
- Kui inimene ilmus kaadris rakenduse käivitamise ajal, peab tema kehamudeli andmed salvestada faili pärast rakenduse töö lõpetamist.

5.1.2 Mittefunktsionaalsed nõuded

Mittefunktsionaalsed nõuded on järgmised:

- Rakendus on lihtsasti kasutatav ja käivitatav;
- Objektide tuvastamine toimiks kiiresti;
- Rakendust saab kasutada Windows'is, Linux'is ja Jetson Orin NANO'l.

5.1.3 Loodav rakendus

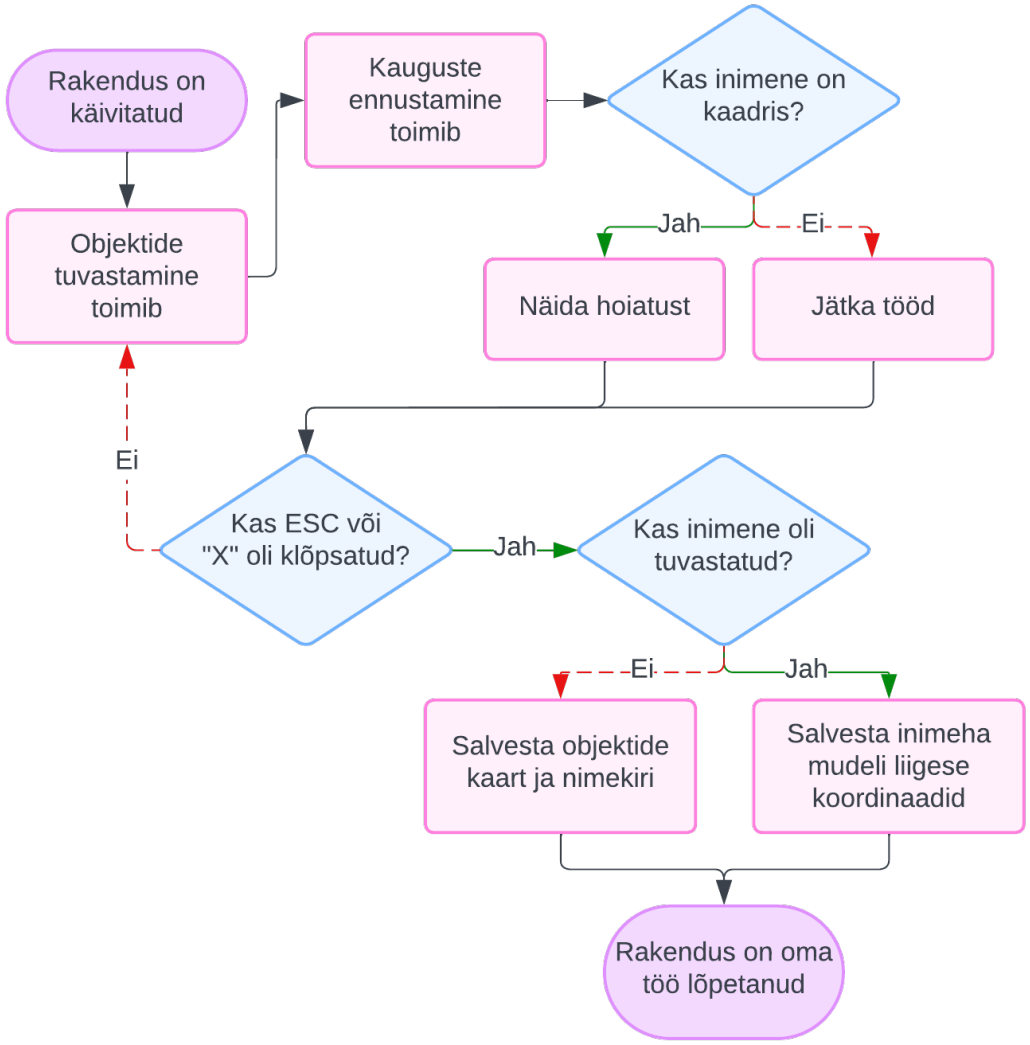
Pärast nõuete määramist, on selge kuidas lõplik rakendus peab välja nägema. Peamine funktsionaalsus koosneb objektide tuvastamisest ja kauguse määramisest reaalajal. Lisaks sellele, kui inimene on kaadris, on soovituslik joonistada inimese peale kehamudel liigeste punktidega, ning pärast rakenduse töö lõpetamist peab vajadusel salvestama kaardi ja nimekirja tuvastatud objektidest ja liigeste asendi nimekirja rakenduse käivitamise ajal.

Kasutajakogemuse paremaks muutumiseks oli otsustatud luua rakendust, mida on võimalik käivitada topeltklõpsades kas Windows'il, Linux'il või Jetson'il.

6. Rakendamise arendamine

Kõigepealt arendatakse rakenduse baasfunktsionaalsust, mis suudab tuvastada objekte ja objektide kauguse kaamera vahel ning kuvada seda kasutajale ekraanil. Pärast seda keskendatakse andmete salvestamisele ja kehamudeli joonistamisele. Lõpuks porditakse rakendus Jetson Orin NANO'le.

Rakenduse arendus ja testimine toimub üldiselt ühes harus, aga kui on vaja midagi tõsisemat testida, siis kasutades versioonihalduse võimalusi luuakse uut haru. See teeb võimalikuks mitmesugused lahendused eraldi testida ja vajadusel tagasi minna.



Joonis 10. *Rakenduse voog.*

6.1 Rakenduse põhi

Teemat uurides leiti Stereolabs'i ametlikus dokumentatsioonis juhendit, kuidas ZED kaamerat ja YOLO algoritmi paika panema ja tööle saama [55], [56]. Oli tehtud otsus seda juhendit kasutada rakenduse põhja arendamiseks.

6.2 Rakenduse käivitamine

Selles peatükis käsitletakse rakenduse käivitamist ja arendamist, tehtud otsuseid, kasutatavaid tehnoloogiaid ja muid tehnoloogilisi küsimusi.

6.2.1 Rakenduse käivitamine kasutajana

Rakenduse käivitamiseks kasutajana on vaja allalaadida kausta `the_component_<nõutava_OS_nimi>` ja käivitada faili `the_component` topeltklõpsades. Kausta saab allalaadida GitHub'i README's oleva lingi abil: `https://github.com/artn01/the_component`.

6.2.2 Rakenduse käivitamine arendajana

Käesolevas lõputöö raames on rakenduse põhjaks valitud Python, mis on üks enamlevinumaid programmeerimiskeele. Rakenduse käivitamiseks on vaja terminalis kasutada käsku `python3 the_component.py`.

Enne seda aga on vaja ühendada ZED-kaamerat arvutiga ja installeerida mõned teegid. Et veenduma, et installitakse kõik vajalikud teegid, on mõistlik käivitada abifail `hello.py`. Selle faili sees on mõned lihtsed operatsioonid, mis aitavad kasutajale käivitada peamist programmi.

Edasi on loetelu teekidest ja programmidest, mis on vaja allalaadida enne rakenduse käivitamist:

- ZED SDK: Olenevalt süsteemist (Windows [57], Linux [58] või Jetson [59]), on vaja allalaadida ZED SDK Stereolabs ametlikult veebilehelt.

- Python: On vaja, et arvutis oleks installeeritud python'i versioon uuem kui 3.8. Seda on võimalik teha kas kasutades ametlikut python veebilehte või via CLI [60].

- Ultralytics: Teek, mis sisaldab YOLO algoritmi ning PyTorch'i. Installeerimiseks kasutakse käsku terminalis: `pip install ultralytics`.

- Mediapipe: Teek inimkeha mudeli joonistamiseks. Installeerimiseks kasutatakse käsku terminalis: `pip install mediapipe`.

- PyTorch koos CUDA toetusega: Kui mis iganes põhjusel ei ole PyTorch installeeritud pärast kõike ülaltoodud käske käivitamisest, siis seda saab allalaadida kasutades järgmist käsku: `pip install torch==1.10.0+cu102 torchvision==0.11.0+cu102 torchaudio==0.10.0 -f https://download.pytorch.org/whl/torch_stable.html`. See on kõige vanem sobiv versioon. Rohkem infot on võimalik leida PyTorch ametlikust veebilehest [61].

- ZED Python API: Kui see API pole veel installitud ZED SDK installeerimise protsessis, siis on vaja jälgida ametlikut Stereolabs dokumentatsiooni ja allalaadida kõik vajalikud teekid [44].

Kui kõik ülaltoodud teekid on allalaaditud, siis on vaja allalaadida failid kaustast `jetson_setup`. Pärast allalaadimist need kaks faili peavad olema samas kaustas. Siis on vaja käivitada faili `hello.py`. Esimestel kordadel, kui seda faili käivitatakse, võib juhtida, et kasutaja saab veateave. See on oodatud ja veateaves öeldakse, milliseid teeki ei ole veel installeeritud. Kasutaja peab järgima veateates antud lahendust ja installeerida vajalik teek kasutades käsku `pip install <teegi_nimi>`. Kui kõik vajalikud teegid on installitud, peaks programm töötama ja kasutaja saab näha, mis CUDA ja ZED SDK versiooni kasutatakse, kas YOLO teek töötab nagu oodatud (YOLO teeb ennustuse samas kaustas olevale pildile, nimelt `room.jpg`), ja kas ZED kaamera on õigesti ühendatud (programm kuvab ZED kaamera seerianumbri).

6.2.3 Rakenduse käivitamine kasutades Miniconda

Üks võimalustest on ka projekti käivitamine kasutades virtuaalkeskkonda, näiteks Anaconda või seda analoogit Miniconda.

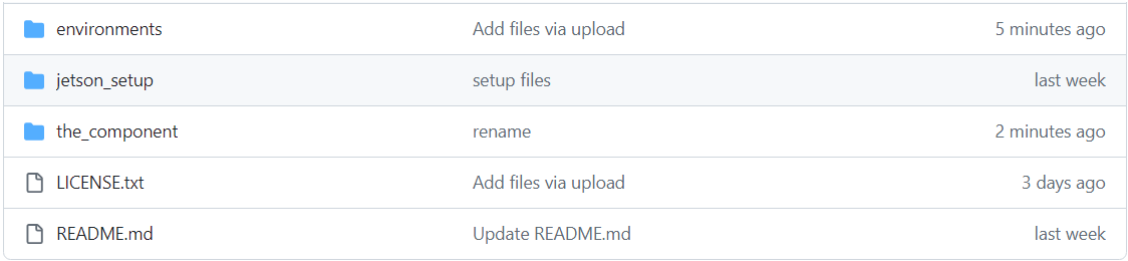
Anaconda.org on Anaconda paketihaldusteenus. Anaconda.org muudab avalike märkmike, keskkondade ning conda ja standardsete Pythoni pakettide leidmise, ligipääsu, salvestamise ja jagamise lihtsaks. Anaconda.org muudab ka pakettidele ja keskkondadele tehtud uuendustega kursis olemise lihtsaks. Anaconda.org hostib sadu kasulikke Pythoni pakette, märkmikke, projekte ja keskkondi mitmesuguste rakenduste jaoks. Avalike pakettide

otsimiseks, allalaadimiseks ja installimiseks ei ole vaja sisse logida ega isegi Anaconda.org kontot omada [62].

Windows'is ja Linuxi's on vaja installida Miniconda't jälgides ametlikut instruktsiooni [63]. Jetson'i jaoks on loodud isegi kergekaalulisem analoog - Archiconda. Seda saab installida jälgides seda juhendit - [64]. Pärast seda peab jälgima Lisas 6 esitatud kasutusjuhendit.

6.3 Rakenduse struktuur

Selles alamtükis vaadeldakse projekti struktuuri. On olemas kolm peamist kausta:



Joonis 11. *Rakenduse struktuur.*

Kaust `environments` sisaldab erinevaid keskkondi rakenduse käivitamiseks.

Kaust `jetson_setup` sisaldab koodi ühenduvuse testimiseks Jetson Orin NANO'l. Kaustas on kaks faili: üks neist on python file, kus testitakse kas kõik teekid töötavad nagu on vaja, teine on pilt ruumist, ka testimiseks. Faili `hello.py` on vaja käivitada kui ZED SDK on installitud.

Järgnevalt on olemas kaust kolmest failist, nimetatud `the_component`, mis on peamine projekti kaust:

- `drawing_utils.py`: See on abifail, mis genereerib erinevaid värvi, otsustab, kas on vaja objekti ekraani joonistada, ja joonestab ka vertikaalseid jooned. Samuti on seal meetod luustiku inimese peale joonistamiseks ning varasemad liigese nimekirjad.

- `cv2viewer.py`: Selle faili peamine meetod on `render2D`, mis sisaldab kogu loogikat objektide ja nende ümber olevate piirdekarbikeste joonistamiseks, ekraanil teksti kuvamiseks, nagu objektide nimed ja hoiatused ning samuti liigeste vaheliste ühenduste joonistamiseks, kui kaamera tuvastab inimese. `cvt` meetod kohandab antud punkti X- ja Y-koordinaate määratud skaalateguritega ja tagastab skaaleeritud punkti. Teised meetodid selles failis on abimeetodid selleks, et pilti paremaks

joonistada. Mõned neist saavad objekti positsiooni ja mõned on välja toodud `render2D` meetodist, et teha koodi lihtsasti lugevamaks. Selle faili alguses on samuti nimekiri vajalikest liigestest ja nende ühendustest, mis on inspireeritud teise tudengi varasema projektiga. Tudengi nimi on Brigite Kerge [65].

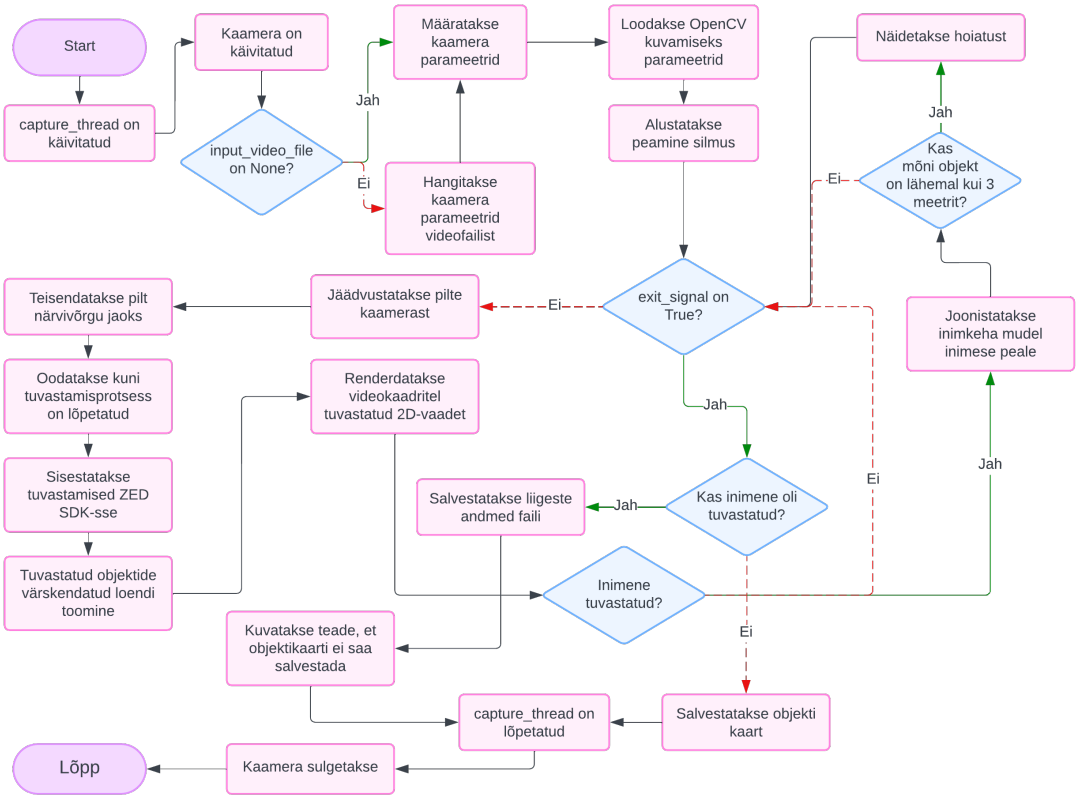
- `the_component.py`: Peamine fail, kust käivitatakse programmi. Sees on meetodid kaardi salvestamiseks pärast kävitamist, meetod piirdekarbi keskpunkti koordinaatide teisendamiseks piirdekarbi serva koordinaatideks, meetod tuvastatud objektide kohandatud piirdekarbi teisendamiseks, meetod lõime loomiseks ja peamine meetod programmi käivitamiseks.

    1. `torch_thread`: See meetod töötab eraldi lõimes ja jätkab piltide töötlemist kuni kästakse peatuda. Kui on olemas pilt töötlemiseks, hangib see lukku tagamiseks, et ükski teine lõim ei saaks samal ajal jagatud ressursse (`image_net`, `detections`) muuta, tagades niimoodi lõimede ohutuse. Meetod võtab pildi globaalsest muutujast `image_net` ja teisendab selle värvi formaadi OpenCV abil BGRA-st RGB-ks. Selline teisendus on vajalik, kuna erinevad pildiallikad ja töötlemisraamatukogud kasutavad erinevaid värvi formaate. Seejärel teisendatakse ennustused PyTorch tensoritest NumPy massiivideks edasiseks töötlemiseks. Pärast töötlemist vabastatakse lukku, võimaldades teistel lõimedel jagatud ressurssele juurde pääseda. Seejärel peatub lõim väga lühikeks ajaks (0,01 sekundit) enne, kui kontrollitakse uuesti, kas on uut tööd teha või kas peaks väljuma. Antud lühike ooteaeg aitab vähendada protsessori kasutust, kuna ei teostata pidevat uute piltide otsingut tsüklis.

    2. `main`: Peamine meetod, mida käivitatakse programmi käivitamisel. Esiteks, funktsioon alustab eraldi lõime (`capture_thread`) loomisega, mis töötleb pilte kasutades neuraalvõrku (läbi `torch_thread` meetodi). See lõim töötab paralleelselt põhifunktsiooniga, et tuvastada objekte ZED kaamerast püütud videokaadritel. Neuraalvõrgu parameetrid nagu kaalud, pildi suurus ja usalduslävend seatakse varasemalt analüüsitud argumentide (`parsed_args`) põhjal.
    Järgmisena toimub kaamera seadistamine. Programm määrab algsed väärtused ZED kaamerale, määrates erinevaid parameetreid nagu sügavusrežiim, resolutsioon ja koordinaatsüsteem. Samuti on võimalik neid parameetreid saada videosalvestuse failist (SVO), kui see on määratud. SVO on Stereolabsi loodud kohandatud failiformaat, et salvestada kaamera andmeid, mida saab kasutada sisendina ZED SDK-le, justkui oleks sama kaamera ühendatud USB-pordi kaudu [66]. Kaamera seadistus hõlmab ühikute määramist, sügavusmõõtmiste

kvaliteeti ja sügavusmõõtmiste maksimaalse kauguse määramist.
Siis toimub funktsioonide käivitamine. Pärast kaamera edukat avamist lubatakse asukoha jälgimise ja objektide tuvastamise. Need seaded võimaldavad kaameral jälgida oma asukohta ruumis ja tuvastada vaateväljas olevaid objekte. Pärast seda funktsioon siseneb põhitsüklisse, mis jätkub kuni `exit_signal` on määratud tõeseks. Selle tsükli jooksul toimub järgmine: jäädvustatakse kaamerast pilte, kui on võimalik (`zed.grab` õnnestub). Saadud pilt teisendatakse neuraalvõrgule sobivaks formaadiks. Pärast seda, määratakse kaamera positsiooni. Kui tuvastused on saadaval, siis neid sisestatakse tagasi ZED SDK-sse edasiseks töötlemiseks. Järgmisena taastab programm uuendatud nimekirja tuvastatud objektidest. Kasutades OpenCV-d, renderdab programm 2D vaate tuvastustest videokaadritel ja kuvab neid.
Tsüklist väljumise järel (kas vea tõttu või käsitsi ESC klahvi vajutamisel) kontrollib funktsioon, kas tuvastati mõni inimene. Kui inimest ei tuvastatud, salvestab see objektide asukohtade skeemi. Kõik tuvastatud objektid salvestatakse samuti CSV-faili.
Lõpuks sulgeb programm kaamera enne töö lõpetamist ning meetodi lõpus salvestatakse ka andmeid inimmudelist CSV-faili, kus on iga liigese nimi, X, Y ja Z koordinaadid.
On väärt märkimist, et `main()` meetodit kasutatakse `torch.no_grad()` funktsiooni abil. `torch.no_grad()` kontekstihaldurit kasutatakse ajutiselt gradientide arvutuse keelamiseks. See on kasulik, kuna gradientide arvutamine võib olla mälumahukas ja pole vajalik, kui mudelit ei treenita, näiteks mudeli hindamise või järelduste tegemise ajal. Gradientide jälgimise keelamisega saab vähendada mälu kasutust ja kiirendada arvutusi [67].

3. `save_plot`: Meetodit käivitakse ainult juhul, kui rakenduse töötamise inimest ei tuvastati. See meetod joonistab skeemile objektide asukohad ning nende kauguse, kasutades selleks joont, mille kohal on meetrites näidatud kaugus. Kui aga on kaardil rohkem kui 5 objekte, siis joont ei joonista. Meetod on loodud tulemuste paremaks mõistmiseks ja loetavuseks inimestele. Samuti selle meetodi sees toimub tuvastatud objektide asukohtade salvestamine CSV-faili.



Joonis 12. *Rakenduse peamise meetodi voog.*

7. Lõpplahenduse testimine

Järgnevas lõigus süveneb autor lõpliku lahenduse testimisse ja veendub samuti, et lahendus vastab algsetele eesmärkidele. Testimist teostab autor oma juhendaja abil.

Esiteks määratakse, kas rakendust saab käitada nii täissuuruses Linuxi arvutis kui ka Nvidia Jetson Orin NANO'l.

Ootuspärane töötamine tähendab rakenduse töötamist ilma tõrkedeta ja rakenduse suvalise väljalülitamiseta, samuti rakendus peab tuvastama objektid ja nende kaugust kaamerast, salvestada inimkeha mudeli kui inimene oli tuvastatud, objektide nimekirja ja kaardi salvestamine kui inimest pole tuvastatud ning hoiatuse näitamine juhul kui suvaline objekti ja inimese vahel on vähem kui 3 meetrit.

- Täissuuruses Linux PC: Kõik töötab ootuspäraselt, videokvaliteeti saab seada HD720-le, kuna täissuuruses arvuti on oluliselt võimsam kui väike Jetson Orin NANO. Pilt on samuti sujuvam, jällegi tänu suuremale võimsusele.

- Jetson Orin NANO: Kõik töötab samuti vastavalt ootustele, aga videokvaliteedi on vaja seada VGA formaadile, sest see on kõige kergekaalulisem videoformaat. Pilt ei ole nii sujuv nagu täissuuruses arvutis, aga see on ka ootuspärane.

Käivitamise protsess on identiline nagu oli kirjeldatud eelnevas lõigus (läbi käsurea). Kõik töötas vastavalt ootustele.

Järgmiseks testiks seati kaamera fikseeritud asendisse ja see pidi tuvastama nähtud objekti.

- Staatiline asend, inimene on kaamera vaateväljas: Esimene test hõlmas inimese olemasolu kaamera vaateväljas. See tähendas, et hoiatus tuli ekraanil kuvada ning tuvastatud objektide kaardi ei olnud vaja salvestada. Tulemus oli kooskõlas ootustega, kõik toimis asjakohaselt.

- Staatiline asend, inimest pole kaamera vaateväljas: Teise testi ootuseks oli, et kaamera vaateväljas polnud inimest. Selles juhus hoiatust ei pea näitama ja objektide kaart peab olema salvestatud joonise kujus. Programm toimus ootuspäraselt. Kaart oli salvestatud, kus oli näha, et kaamera ei ole liikunud.

Käesolevaks testiks pidi kaamera liikuma, jäljendamaks selle paiknemist robotil.

- Dünaamiline asend, inimene ilmus kaadris: Simuleeriti kaamera asendit robotis, liikudes seda ümber oma telje ja edasi, tagasi, paremale ja vasakule. Tulemused olid ootuspärased, ekraanis ilmus hoitaus ja pärast kaamera välja lülitamist kaart ei olnud salvestatud.

- Dünaamiline asend, inimest polnud kaadris: Sarnaselt eelmise testiga simuleeriti kaamera asendit robotis, liikudes seda ümber oma telje ja edasi, tagasi, paremale ja vasakule. See kord inimene ei ilmunud kaamera vaateväljas, mis tähendas, et hoiatust ei olnud vaja näidata ja ka kaart peaks olema salvestatud. Tulemus oli jälle ootusepärane: hoiatus ei ilmunud ekraanis kaamera töö ajal ja kaart oli salvestatud pärast töö lõpetamist. Skeemil olid objektid kaamera ümber igal pool, mis kinnitas, et kaamera liikus ja vaatas ringi. See on asjakohane tulemus.

Kolmaks testiks oli inimkeha mudeli korrektsuse valideerimine. Kuna programm oskab ka inimkeha mudeli ekraanile joonistada, kui kaamera inimest näeb, siis oli nõutav seda ka testida.

- Inimkeha mudeli tuvastamine: Kui inimene ilmub kaadris, siis peab programm üle inimese jonistada nõnda nimetatud "luustiku" või lihtsalt inimkeha mudeli. See peaks joonistama määratud liigesed ja nende ühendused. Tulemus oli ootuspärane: programm joonistas inimese peale liigesed ja nende ühendused.

Neljas test oli stressitest. Seda viidi läbi IT-teaduskonna doktoriõppe avatud uste õhtul, mis toimus 16.04.2024,kus autor ja tema juhendaja tutvustasid tudengitele ja teistele külalistele teemasid, kasutades ka käesoleva bakalaurusetöö. Üritusel pidi programm pidevalt töötama ühe tunni jooksul. Autor ise jälgis seda, et program ei lülitaks välja ja tuvastaks inimesi ja objekte. Programm töötas vastavalt ootustele ning kõrvalekaldeid ei esinenud.

Kokkuvõtteks, kõik testide tulemused olid asjakohased ja kooskõlas autori ja tema juhendaja ootustega. Tulemuste illustratiivsed joonised on Lisas 3.

8. Hinnang lõpptulemusele

Lõpliku tulemuse hindamine on näidanud positiivseid tulemusi, mis ületavad esialgsed ootused ja nõuded. Põhjaliku testimise ja pärismaailmas kasutamisele kaudu on süsteem järjepidevalt näidanud kõrget täpsust ja usaldusväärsust objektide tuvastamisel ja kaardistamisel ruumis.

Rakendus töötab Jetson Orin NANO'l ning käivitamine võtab 13 sekundit ja see tulemus on täielikult sobiv nii autorile kui ka tema juhendajale.

Maksimaalne objekltide tuvastamise kaugus on 7 meetrit. Vajadusel saab seda suurendada või vähendada, aga arendamise protsessis just 7 meetrit peeti kõige sobilikuma kauguse. Täpsed inimkeha mudeli tuvastamised on kaugusel 1 meetrist kuni 3.5 meetrini. Rakendus on suuteline tuvastada inimkeha mudeli ka kaugemal või lähedamal, aga sellistel juhtudel täpsus halveneb.

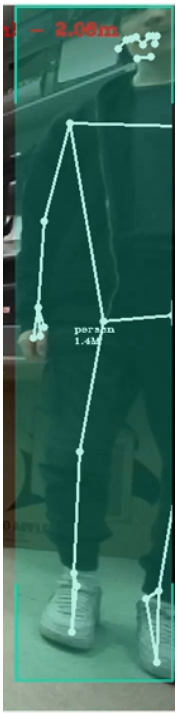
Loodud rakendus suudab tõestada 80 erinevat klassi. Need klassid on määratletud COCO datasetis [68], mida rakenduse pretreenitud mudel kasutab. Tehtud oli 50 erinevat katset ja nende põhjal tuvastas programm kõik objektid õigesti umbes 80% juhtudest. Kuna lõputöös oli kautatud YOLOv8m mudel ilma lisakihtideta, täpsus on sama nagu Nyoman Bogi Aditya Karna ja teistega kirjutatud uuringus Tabeli 2 esimes osas [43]. Programm ei tuvastanud objekte, mis ei olnud selles lõputöös kasutatud mudelis (näiteks väike kast). See tulemus on selle lõputöö jaoks sobiv, kuid vajadusel saab andmestikku vahetada spetsialiseerituma vastu või luua kohandatud andmestik.

Ka 50 erineva testi põhjal oli kauguse hindamine enamasti õige, välja arvatud äärejuhtumid, kus objektist oli kaamerale nähtav ainult väike osa või objekt oli kaamerale liiga lähedal, hõivates üle 80% kaamera vaateväljast.

Joonis 13. *Vale kauguse ennustamine (TV vasakul).*

Lisaks sellele saab rakendus joonistada inimesele luustiku, kasutades MediaPipe baasversiooni, millel on 32 liigesepunkti. Liigeste valideerimine ei ole käesoleva töö eesmärk, aga selle täpsust saab võrrelda järgmisel tööl uuritud täpsusega: Chung Jen-Li, Ong Lee-Yeng ja Leow Meng-Chew "Skelettidel põhineva inimpoosi hindamise võrdlev analüüs" [69].

Inimkeha mudeli saab valideerida kui vähemalt pool inimkehast on kaamera vaateväljas (120°) ja asub kõige rohkem 7 meetri kaugusel ja vähem 1 meetri kaugusel.



Joonis 14. *Pool inimkeha kaamera vaateväljas.*

Üldiselt on rakendus tõestanud oma kasulikkust ja täpsust nii simuleeritud (ülikooli laboris)

kui ka päriselu (doktoriõppe avatud uste õhtu) keskkondades.

Kokkuvõttes saab see rakendus positiivse hinnangu autorist ja tema juhendajast selle tõhususe ja kohandatavuse kohta. Samuti käesolev rakendus loob ka lubava aluse tulevastele täiustustele ja potentsiaalsetele uutele rakendustele.

8.1 Võimalused edasiseks arenduseks

Hoolimata selle bakalaurusetöö saavutustest, on veel võimalusi edasisteks täiustusteks, eriti kaardistamise täpsuses ja süsteemi operatiivses rakendamises.

- Kaardistamise täpsuse edendamine: Kuigi süsteem toimib praegu kiiduväärselt täpselt, on veel ruumi täiendusteks. Täiustatud sensoritehnoloogiate integreerimine ja keerukama objektide tuvastamise mudeli rakendamine võiks vähendada vigu objektide tuvastamisel ja kauguse ennustamisel. Katsetamine erinevat tüüpi anduritega, nagu infrapuna või ultraheli, võib samuti parandada süsteemi reageerimist erinevatele keskkonnatingimustele ja objektidele.

- Täiustatud mobiilne rakendamine: Praegune seadistus, kasutades Jetson Orin NANO mini-PC-d, on osutunud tõhusaks; siiski, selle seadistuse kinnitamine liikuvale robotile kujutab endast isegi rohkem lubavat arengut. Selline mobiilsus võimaldaks süsteemil dünaamiliselt toimida erinevates keskkondades, laiendades sellisel viisil selle kasutusvõimalusi, eriti stsenaariumides nagu autonoomne navigeerimine.

- Stsenaariumipõhine kohandamine: Süsteemi edasine kohandamine konkreetsetele stsenaariumidele võiks oluliselt suurendada selle tõhusust. Näiteks spetsiifiliste profiilide või seadistuste loomine, mis kohandavad tuvastusalgoritme tüüpiliste objektitüüpide ja liikumiste põhjal konkreetses keskkonnas, nagu rahvarohked linnaruumid või struktureeritud siseruumid. Selline lähenemine võiks parandada nii täpsust kui ka efektiivsust.

Need välja pakutud arendused ei püüa mitte ainult laiendada süsteemi saavutuste piire, vaid tagavad ka selle, et see jääks objektide tuvastamise ja liikumise registreerimise tehnoloogiate kooskõlas. Iga täiustus nõuab hoolikat testimist, et see vastaks täpselt praktilistele vajadustele ja tehnoloogilistele edusammudele valdkonnas.

9. Kokkuvõte

Käesolevas lõputöös on arendus ja rakendamine edukalt täitnud kõik eelnevalt määratletud eesmärgid, lisaks projektiga kaasnenud täiendavatele eesmärkidele. Rakendus tuvastab oskuslikult objekte oma visuaalses vaateväljas, mõõdab kaugust nendest objektidest kaamerani ning eristab inimkujusid, kujutades luustiku, mis koosneb liigestest (kujutatud ringadena), mida ühendavad jooned. See võimekus pakkub detailset visuaalset esitust inimliikumisest reaalajas.

Süsteem on kavandatud ka andmehaldusstrateegiaga: see salvestab objektide ja kaamera asukohad koos kaugustega objektideni ainult siis, kui töötamise ajal ei tuvastata inimest. See selektiivne andmehaldus optimeerib salvestus- ja töötlusressursse. Nende omaduste edukas rakendamine näitab vastupidavat rakendust, mis mitte ainult ei täida oma algset ulatust, vaid laiendab ka oma kasulikkust lisafunktsioonide kaudu, tehes olulisi edusamme liikumise registreerimise ja objektide tuvastamise tehnoloogiate valdkonnas.

10. Tunnustus

Käesolev lõputöö on osaliselt toetatud Eesti Teadusagentuuri ETAG grandiga PRG 2100.

Kasutatud kirjandus

[1] Anna Krajuškina. *Modelling Parkinson's Disease with Gait Analysis Approach*. [Kasutatud: 22-04-2024]. 2019. URL: https://digikogu.taltech.ee/et/Item/b2bf1757-1417-4048-837a-61f38c844fa7.

[2] *Timed Up & Go (TUG) – NeurologyToolKit*. [Kasutatud: 10-05-2024]. URL: https://neurotoolkit.com/tug/.

[3] Anna Krajuškina. "An Alternative Approach for Gait Analysis of Parkinson's Disease Patients". [Kasutatud: 29-04-2024]. PhD thesis. Tallinn University of Technology, 2017, p. 13. URL: https://digikogu.taltech.ee/et/Item/a73c9006-006b-457e-a419-3738c4305101.

[4] Sven Nõmm et al. "An Alternative Approach to Distinguish Movements of Parkinson Disease Patients". In: *IFAC-PapersOnLine*. Vol. 19. [Kasutatud: 27-04-2024]. 2016, pp. 272–276. URL: https://www.sciencedirect.com/science/article/pii/S240589631632136X.

[5] *Azure Kinect body tracking joints*. [Kasutatud: 10-05-2024]. URL: https://learn.microsoft.com/en-us/azure/kinect-dk/body-joints#joint-hierarchy.

[6] *Pose landmark detection guide*. [Kasutatud: 10-05-2024]. URL: https://developers.google.com/mediapipe/solutions/vision/pose_landmarker.

[7] Ahmed Abdelhady. *Low cost gait capture during turning motion*. [Kasutatud: 12-05-2024]. 2017. URL: https://digikogu.taltech.ee/et/Item/b013ee9a-1f3f-48ee-95ad-c143c5d26cd8.

[8] Jan-Joonas Bernstein. "Context Based Registration and Analysis of Human Motions". [Kasutatud: 12-05-2024]. PhD thesis. Tallinn University of Technology, 2017. URL: https://digikogu.taltech.ee/et/Item/eb7a16ec-eab2-459d-af30-4741da9124b3.

[9] Artjom Protski and Danila Romanov. "Human Motion Capture and Analysis Component for Mobile Robotics Platform". [Kasutatud: 12-05-2024]. PhD thesis. Tallinn University of Technology, 2023. URL: https://digikogu.taltech.ee/et/Item/be997795-4cc8-4baf-8c36-ed6e84363d72.

[10] *Beginners Guide / Overview*. [Kasutatud: 14-04-2024]. URL: https://wiki.python.org/moin/BeginnersGuide/Overview.

[11] *About the Ubuntu project*. [Kasutatud: 11-03-2024]. URL: https://ubuntu.com/about.

[12] *Jetson Orin for Next-Gen Robotics - NVIDIA*. [Kasutatud: 11-03-2024]. URL: https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-orin/#:~:text=The%20NVIDIA%20Jetson%20Orin%20Nano,on%20Jetson%20Orin%20Developer%20Kits.

[13] *NVIDIA JETSON ORIN NANO 8GB DEVELOPMENT KIT*. [Kasutatud: 11-03-2024]. URL: https://www.siliconhighwaydirect.com/product-p/945-13766-0005-000.htm.

[14] *Jetson Nano Developer Kit EOL*. [Kasutatud: 22-05-2024]. URL: https://forums.developer.nvidia.com/t/jetson-nano-developer-kit-eol/276729.

[15] *Status of Python versions*. [Kasutatud: 22-05-2024]. URL: https://devguide.python.org/versions/.

[16] *Jetson Modules*. [Kasutatud: 25-05-2024]. URL: https://developer.nvidia.com/embedded/jetson-modules.

[17] *Yahboom Jetson Orin Nano*. [Kasutatud: 10-05-2024]. URL: https://www.amazon.com/Yahboom-Artificial-Intelligence-Development-Programming/dp/B0C5LTSQNN?th=1.

[18] *Getting Started with your ZED camera*. [Kasutatud: 04-04-2024]. URL: https://support.stereolabs.com/hc/en-us/articles/207616785-Getting-Started-with-your-ZED-camera.

[19] *Stereolabs - ZED 2/2i*. [Kasutatud: 11-03-2024]. URL: https://husarion.com/tutorials/ros-equipment/zed/#:~:text=ZED%202%2F2i%20cameras%20by,accurate%20object%20detection%20and%20recognition..

[20] *ZED Camera dimensions*. [Kasutatud: 10-05-2024]. URL: https://github.com/stereolabs/zed-ros-wrapper/issues/797.

[21] *ZED Camera Overview*. [Kasutatud: 04-04-2024]. URL: https://www.stereolabs.com/docs/video.

[22] *ZED Camera Sensors Overview*. [Kasutatud: 04-04-2024]. URL: https://www.stereolabs.com/docs/sensors.

[23] *ZED Camera Depth Sensing Overview*. [Kasutatud: 04-04-2024]. URL: https://www.stereolabs.com/docs/depth-sensing.

[24] *How does the ZED work?* [Kasutatud: 16-04-2024]. URL: https://support.stereolabs.com/hc/en-us/articles/206953039-How-does-the-ZED-work.

[25] *3D Object Detection Overview*. [Kasutatud: 04-04-2024]. URL: https://www.stereolabs.com/docs/object-detection.

[26] Jigar Shah. *GitHub vs GitLab: Which is the Best in 2024?* [Kasutatud: 12-03-2024]. URL: https://radixweb.com/blog/github-vs-gitlab.

[27] Ovidiu-Constantin Novac et al. "Analysis of the Application Efficiency of TensorFlow and PyTorch in Convolutional Neural Network". In: *Sensors* 22 (2022). [Kasutatud: 06-05-2024]. URL: https://www.mdpi.com/1424-8220/22/22/8872.

[28] Renas Rajab Asaad et al. "Image Processing with Python Libraries". In: *Academic Journal of Nawroz University* 12.2 (2023). [Kasutatud: 12-05-2024]. URL: https://www.scirp.org/journal/paperinformation?paperid=115011.

[29] *About OpenCV*. [Kasutatud: 10-03-2024]. URL: https://opencv.org/about/.

[30] *Scikit-Image*. [Kasutatud: 10-03-2024]. URL: https://en.wikipedia.org/wiki/Scikit-image.

[31] *Pillow*. [Kasutatud: 10-03-2024]. URL: https://pillow.readthedocs.io/en/stable/.

[32] *NumPy*. [Kasutatud: 10-03-2024]. URL: https://www.w3schools.com/python/numpy/numpy_intro.asp#:~:text=NumPy%20is%20a%20Python%20library,you%20can%20use%20it%20freely..

[33] Henri Kübe. *Car detection system based on Tartu college parking lot*. [Kasutatud: 04-05-2024]. 2023. URL: https://digikogu.taltech.ee/et/Item/f5175927-8199-4192-abfe-76e3bfe6a790.

[34] Junsong Ren and Yi Wang. "Overview of Object Detection Algorithms Using Convolutional Neural Networks". In: [Kasutatud: 12-05-2024]. 2022. URL: https://www.scirp.org/journal/paperinformation?paperid=115011.

[35] Shengyu Lu et al. "A real-time object detection algorithm for video". In: *Computers & Electrical Engineering* 77 (2019). [Kasutatud: 12-05-2024], pp. 398–408. ISSN: 0045-7906. DOI: https://doi.org/10.1016/j.compeleceng.2019.05.009. URL: https://www.sciencedirect.com/science/article/pii/S0045790618319682.

[36] Juan Terven and Diana-Margarita Cordova-Esparza. “A Comprehensive Review of YOLO: From YOLOv1 to YOLOv8 and Beyond”. In: (2023). [Kasutatud: 10-05-2024], p. 3. URL: https://arxiv.org/abs/2304.00501.

[37] *Ultralytics YOLOv8 Docs*. [Kasutatud: 11-03-2024]. URL: https://docs.ultralytics.com/#yolo-a-brief-history.

[38] Wei Liu et al. “SSD: Single Shot MultiBox Detector”. In: *Computer Vision – ECCV 2016*. Cham: Springer International Publishing, 2016, pp. 21–37. ISBN: 978-3-319-46448-0.

[39] *COCO dataset*. [Kasutatud: 18-04-2024]. URL: https://github.com/ultralytics/ultralytics/blob/main/ultralytics/cfg/datasets/coco.yaml.

[40] Tsung-Yi Lin et al. *Microsoft COCO: Common Objects in Context*. [Kasutatud: 10-05-2024]. 2015. arXiv: 1405.0312 [cs.CV].

[41] Joseph Redmon et al. *You Only Look Once: Unified, Real-Time Object Detection*. [Kasutatud: 04-05-2024]. 2016. URL: https://arxiv.org/abs/1506.02640.

[42] *Ultralytics Object Detection*. [Kasutatud: 17-04-2024]. URL: https://docs.ultralytics.com/tasks/detect/.

[43] Nyoman Bogi Aditya Karna et al. “Toward Accurate Fused Deposition Modeling 3D Printer Fault Detection Using Improved YOLOv8 With Hyperparameter Optimization”. In: *IEEE Access* 11 (2023). [Kasutatud: 10-05-2024], pp. 74251–74262. DOI: 10.1109/ACCESS.2023.3293056.

[44] *Install the ZED Python API*. [Kasutatud: 26-03-2024]. URL: https://www.stereolabs.com/docs/app-development/python/install.

[45] *About Cython*. [Kasutatud: 26-03-2024]. URL: https://cython.org/.

[46] *threading — Thread-based parallelism*. [Kasutatud: 26-03-2024]. URL: https://docs.python.org/3/library/threading.html.

[47] Pat Brans. *What is a daemon?* [Kasutatud: 14-04-2024]. URL: https://www.techtarget.com/whatis/definition/daemon.

[48] *time — Time access and conversions*. [Kasutatud: 30-03-2024]. URL: https://docs.python.org/3/library/time.html.

[49] *argparse — Parser for command-line options, arguments and sub-commands*. [Kasutatud: 30-03-2024]. URL: https://docs.python.org/3/library/argparse.html.

[50] *Matplotlib: Visualization with Python*. [Kasutatud: 30-03-2024]. URL: https://matplotlib.org/.

[51] *matplotlib.pyplot*. [Kasutatud: 30-03-2024]. URL: https://matplotlib.org/stable/api/pyplot_summary.html.

[52] *mediapipe 0.10.11*. [Kasutatud: 04-04-2024]. URL: https://pypi.org/project/mediapipe/.

[53] *Pose landmark detection guide*. [Kasutatud: 04-04-2024]. URL: https://developers.google.com/mediapipe/solutions/vision/pose_landmarker.

[54] *PyInstaller Manual*. [Kasutatud: 01-05-2024]. URL: https://pyinstaller.org/en/stable/.

[55] *How to Use YOLO with ZED*. [Kasutatud: 11-05-2024]. URL: https://www.stereolabs.com/docs/yolo#introduction.

[56] *Using YOLO and the ZED*. [Kasutatud: 11-05-2024]. URL: https://github.com/stereolabs/zed-yolo.

[57] *How to Install ZED SDK on Windows*. [Kasutatud: 04-04-2024]. URL: https://www.stereolabs.com/docs/installation/windows.

[58] *How to Install ZED SDK on Linux*. [Kasutatud: 04-04-2024]. URL: https://www.stereolabs.com/docs/installation/linux.

[59] *How to Install ZED SDK on NVIDIA Jetson*. [Kasutatud: 04-05-2024]. URL: https://www.stereolabs.com/docs/installation/jetson.

[60] *Python*. [Kasutatud: 04-04-2024]. URL: https://www.python.org/downloads/.

[61] *INSTALLING PREVIOUS VERSIONS OF PYTORCH*. [Kasutatud: 04-04-2024]. URL: https://pytorch.org/get-started/previous-versions/#osx-27.

[62] *Anaconda.org*. [Kasutatud: 13-05-2024]. URL: https://docs.anaconda.com/free/anacondaorg/.

[63] *Installing Miniconda*. [Kasutatud: 07-04-2024]. URL: https://docs.anaconda.com/free/miniconda/miniconda-install/.

[64] *Install Archiconda3*. [Kasutatud: 12-05-2024]. URL: https://gist.github.com/SaschaDittmann/687218eca54f74654e9c83d41937eade.

[65] Brigite Kerge. *Mediasense*. [Kasutatud: 16-04-2024]. URL: https://gitlab.cs.ttu.ee/brkerg/iti0303-2023.

[66] *What is SVO and why it is used for*. [Kasutatud: 07-04-2024]. URL: https://community.stereolabs.com/t/what-is-svo-and-why-it-is-used-for/1071/2.

[67] *no_grad*. [Kasutatud: 13-05-2024]. URL: https://pytorch.org/docs/stable/generated/torch.no_grad.html.

[68] *COCO dataset*. [Kasutatud: 26-05-2024]. URL: https://github.com/ultralytics/ultralytics/blob/main/ultralytics/cfg/datasets/coco.yaml.

[69] Jen-Li Chung, Lee-Yeng Ong, and Meng-Chew Leow. "Comparative Analysis of Skeleton-Based Human Pose Estimation". In: *Future Internet* 14.12 (2022). [Kasutatud: 10-05-2024]. ISSN: 1999-5903. DOI: 10.3390/fi14120380. URL: https://www.mdpi.com/1999-5903/14/12/380.

Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks[1]

Mina, Artjom Nikokošev

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "Objektide kaardistamise rakendus liikumise registreerimise süsteemi jaoks", mille juhendaja on Sven Nõmm
   1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

27.05.2024

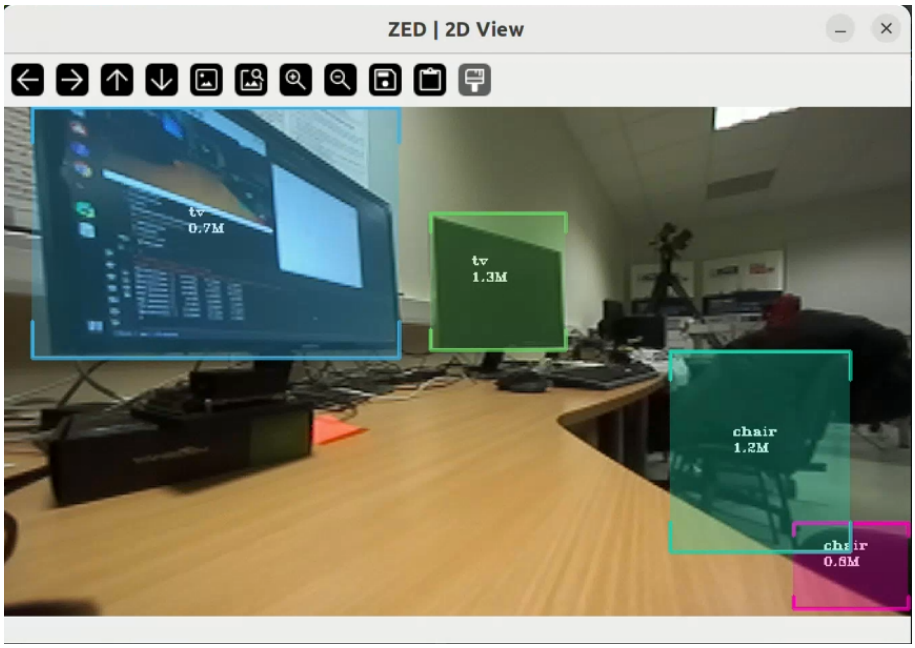
[1] Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või enam isikut oma ühise loomingulise tegevusega ning lõputöö kaas- või ühisautor(id) ei ole andnud lõputööd kaitsvale üliõpilasele kindlaksmääratud tähtajaks nõusolekut lõputöö reprodutseerimiseks ja avalikustamiseks vastavalt lihtlitsentsi punktidele 1.1. ja 1.2, siis lihtlitsents nimetatud tähtaja jooksul ei kehti.

Lisa 2 – Rakenduse failid

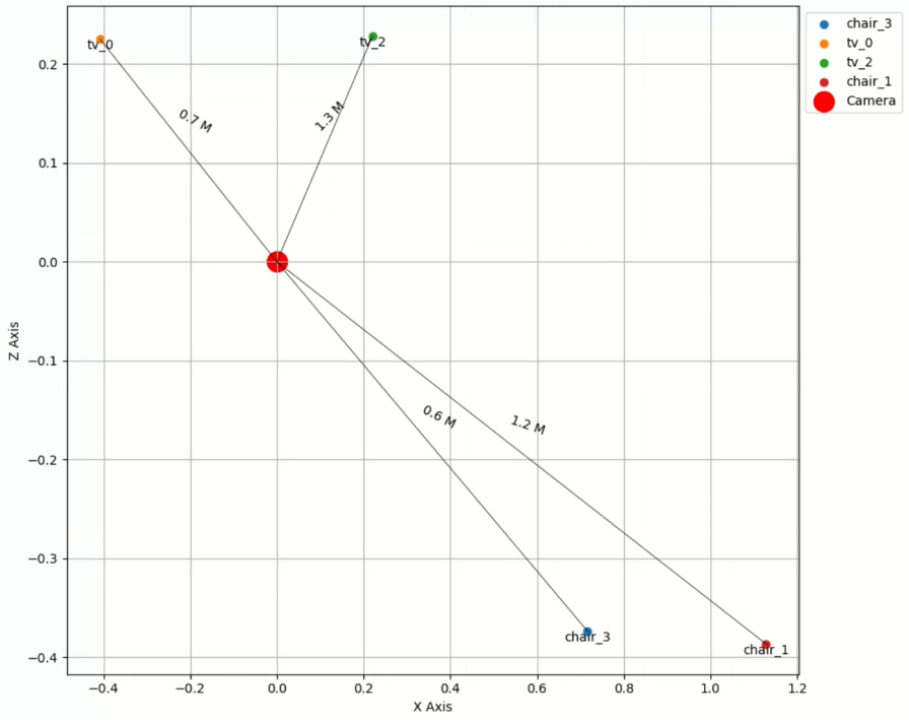
Lähtekood:

- `https://gitlab.cs.ttu.ee/anikok/thesis`
- `https://github.com/artn01/the_component`

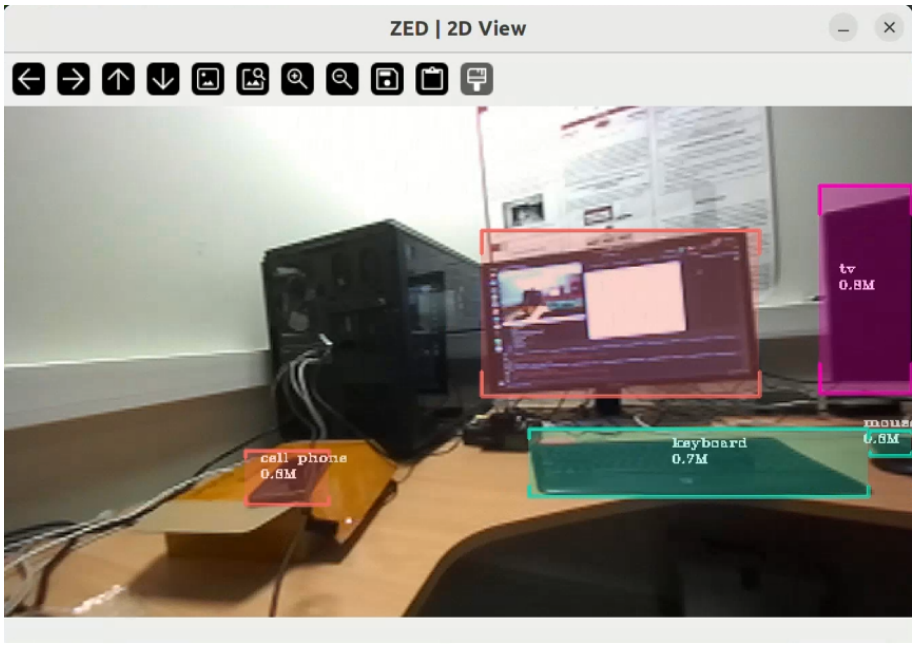
Lisa 3 – Testimise tulemused



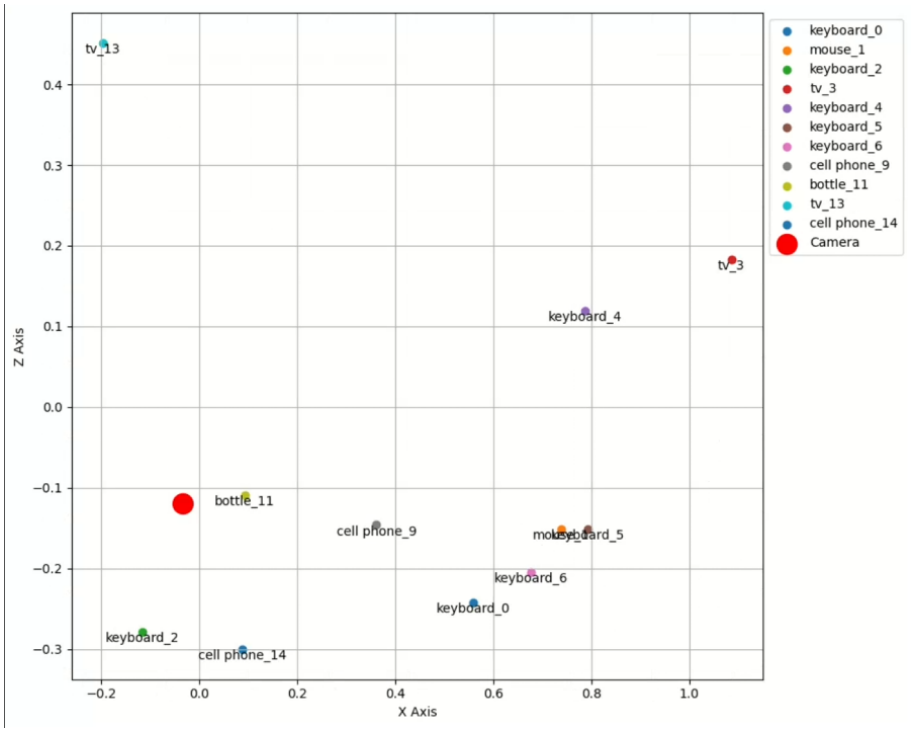
Joonis 15. *Staatiline asend ilma inimeseta.*



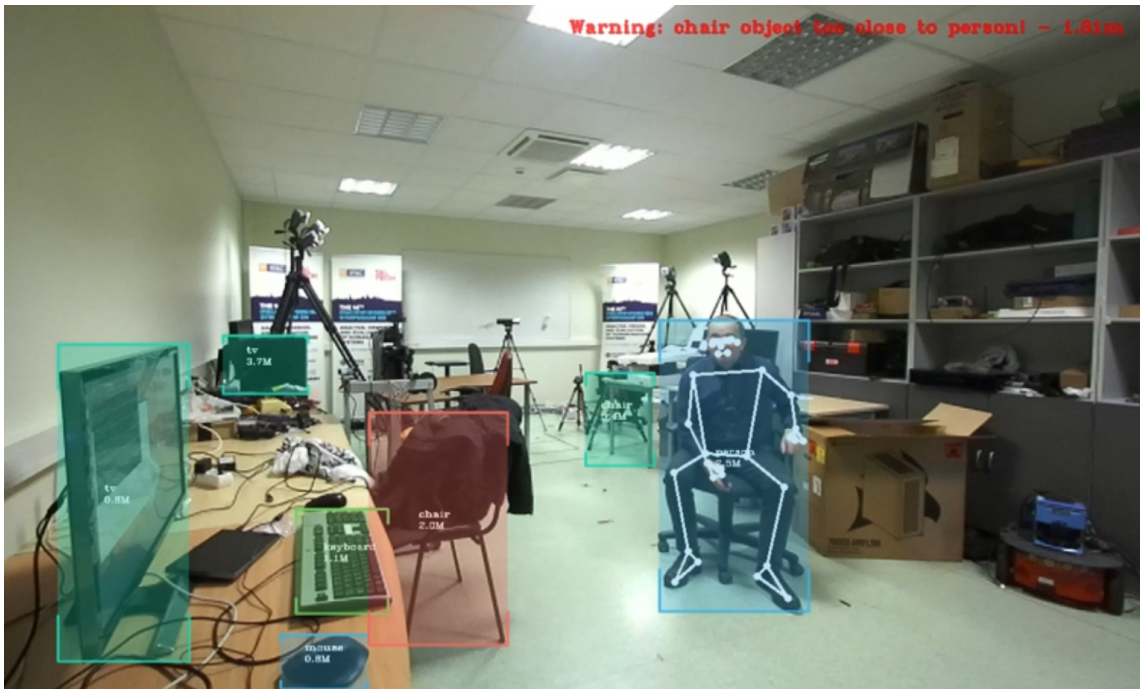
Joonis 16. *Staatiline asend ilma inimeseta, kaart.*



Joonis 17. *Dünaamiline asend ilma inimeseta.*



Joonis 18. *Dünaamiline asend ilma inimeseta, kaart.*



Joonis 19. *Inimene kaadris, hoiatus ja keha mudel.*

Joonis 20. *Inimene kaadris, hoiatust pole.*

Lisa 4 – Mediasense projekt

```
joints = [
    {"name": "right wrist", "connectedJoints": [20, 16, 14]},
    {"name": "left wrist", "connectedJoints": [19, 15, 13]},
    {"name": "right ankle", "connectedJoints": [32, 28, 26]},
    {"name": "left ankle", "connectedJoints": [31, 27, 25]},
    {"name": "right elbow", "connectedJoints": [16, 14, 12]},
    {"name": "left elbow", "connectedJoints": [11, 13, 15]},
    {"name": "right knee", "connectedJoints": [24, 26, 28]},
    {"name": "left knee", "connectedJoints": [23, 25, 27]},
    {"name": "right shoulder", "connectedJoints": [14, 12, 24]},
    {"name": "left shoulder", "connectedJoints": [13, 11, 23]},
    {"name": "right hip", "connectedJoints": [12, 24, 26]},
    {"name": "left hip", "connectedJoints": [11, 23, 25]},
```

Joonis 21. *Liigese nimekiri, originaalne.*

```
joints_full_no_self_index = [
    {"name": "nose", "connectedJoints": [4, 1], "id": 0},

    {"name": "left eye(inner)", "connectedJoints": [0, 2], "id": 1},
    {"name": "left eye", "connectedJoints": [1, 3], "id": 2},
    {"name": "left eye(outer)", "connectedJoints": [2, 7], "id": 3},

    {"name": "right eye(inner)", "connectedJoints": [0, 5], "id": 4},
    {"name": "right eye", "connectedJoints": [4, 6], "id": 5},
    {"name": "right eye(outer)", "connectedJoints": [5, 8], "id": 6},

    {"name": "left ear", "connectedJoints": [3], "id": 7},
    {"name": "right ear", "connectedJoints": [6], "id": 8},
    {"name": "mouth(left)", "connectedJoints": [10], "id": 9},
    {"name": "mouth(right)", "connectedJoints": [9], "id": 10},
    {"name": "left shoulder", "connectedJoints": [12, 23, 13], "id": 11},
    {"name": "right shoulder", "connectedJoints": [11, 24, 14], "id": 12},

    {"name": "left elbow", "connectedJoints": [11, 15], "id": 13},
    {"name": "right elbow", "connectedJoints": [12, 16], "id": 14},
    {"name": "left wrist", "connectedJoints": [13, 17, 19, 21], "id": 15},
    {"name": "right wrist", "connectedJoints": [14, 18, 20, 22], "id": 16},

    {"name": "left pinky", "connectedJoints": [15, 19], "id": 17},
    {"name": "right pinky", "connectedJoints": [16, 20], "id": 18},
    {"name": "left index", "connectedJoints": [15, 17], "id": 19},
    {"name": "right index", "connectedJoints": [18, 16], "id": 20},

    {"name": "left thumb", "connectedJoints": [15], "id": 21},
    {"name": "right thumb", "connectedJoints": [16], "id": 22},
    {"name": "left hip", "connectedJoints": [11, 25, 24], "id": 23},
    {"name": "right hip", "connectedJoints": [12, 26, 23], "id": 24},

    {"name": "left knee", "connectedJoints": [23, 27], "id": 25},
    {"name": "right knee", "connectedJoints": [24, 28], "id": 26},
    {"name": "left ankle", "connectedJoints": [25, 29, 31], "id": 27},
    {"name": "right ankle", "connectedJoints": [26, 30, 32], "id": 28},

    {"name": "left heel", "connectedJoints": [27, 31], "id": 29},
    {"name": "right heel", "connectedJoints": [28, 32], "id": 30},
    {"name": "left foot index", "connectedJoints": [27, 29], "id": 31},
    {"name": "right foot index", "connectedJoints": [28, 30], "id": 32},
]
```

Joonis 22. *Liigese nimekiri, täiendatud.*

Tabel 4. *Liigese eesti keelsed nimetused.*

Inglise keeles	**Eesti keeles**
nose	nina
left eye (inner)	vasak silm (sisemine)

Jätkub...

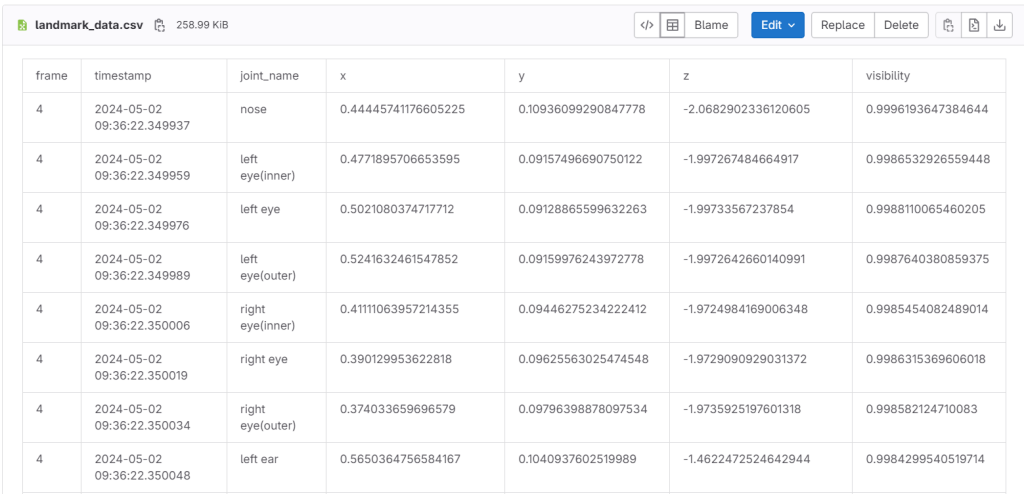
Tabel 4 – *Jätkub...*

Inglise keeles	**Eesti keeles**
left eye	vasak silm
left eye (outer)	vasak silm (välimine)
right eye (inner)	parem silm (sisemine)
right eye	parem silm
right eye (outer)	parem silm (välimine)
left ear	vasak kõrv
right ear	parem kõrv
mouth (left)	suu (vasak)
mouth (right)	suu (parem)
left pinky	vasak väike sõrm
right pinky	parem väike sõrm
left index	vasak nimetissõrm
right index	parem nimetissõrm
left thumb	vasak pöial
right thumb	parem pöial
left heel	vasak kand
right heel	parem kand
left foot index	vasaku jala nimetissõrm
right foot index	parema jala nimetissõrm
right wrist	parem ranne
left wrist	vasak ranne
right ankle	parem hüppeliiges
left ankle	vasak hüppeliides
right elbow	parem küünarliiges
left elbow	vasak küünarliiges
right knee	parem põlv
left knee	vasak põlv
right shoulder	parem õlg
left shoulder	vasak õlg
right hip	parem puus
left hip	vasak puus

Lisa 5 – Tekitatud failid

	A	B	C	D
1	object	x	y	distance
2	tv_0	0.844496	0.349507	1.559229
3	laptop_1	-1.42552	0.232081	2.05513
4	keyboard_3	1.156424	-0.25695	1.858292
5	camera_1	0	0	

Joonis 23. `objects.csv`



frame	timestamp	joint_name	x	y	z	visibility
4	2024-05-02 09:36:22.349937	nose	0.44445741176605225	0.10936099290847778	-2.0682902336120605	0.9996193647384644
4	2024-05-02 09:36:22.349959	left eye(inner)	0.4771895706653595	0.09157496690750122	-1.997267484664917	0.9986532926559448
4	2024-05-02 09:36:22.349976	left eye	0.5021080374717712	0.09128865599632263	-1.99733567237854	0.998811006546020 5
4	2024-05-02 09:36:22.349989	left eye(outer)	0.5241632461547852	0.09159976243972778	-1.9972642660140991	0.9987640380859375
4	2024-05-02 09:36:22.350006	right eye(inner)	0.41111063957214355	0.09446275234222412	-1.9724984169006348	0.9985454082489014
4	2024-05-02 09:36:22.350019	right eye	0.390129953622818	0.09625563025474548	-1.9729090929031372	0.9986315369606018
4	2024-05-02 09:36:22.350034	right eye(outer)	0.374033659696579	0.09796398878097534	-1.9735925197601318	0.998582124710083
4	2024-05-02 09:36:22.350048	left ear	0.5650364756584167	0.1040937602519989	-1.4622472524642944	0.9984299540519714

Joonis 24. `landmark_data.csv`

Lisa 6 – Kasutusjuhend

Nüüd on vaja tuvastada, et conda töötab õigesti. Seda saab teha käskuga: `conda`. **NB!** Näidis illustreerib Windows'il süsteemi paigaldust.

```
C:\Users\Artjom>conda
usage: conda-script.py [-h] [-v] [--no-plugins] [-V] COMMAND ...

conda is a tool for managing and deploying applications, environments and packages.
```

Joonis 25. *`conda` käsku käivitamine.*

Seejärel on vaja luua uut keskkonda. Seda tehakse käsuga `conda create -n <env-name> python=3.8`, ja kui on vaja valida *([y]/n)* valida *y*:

```
C:\Users\Artjom>conda create -n py38 python=3.8
Channels:
 - defaults
 - conda-forge
Platform: win-64
Collecting package metadata (repodata.json): done
Solving environment: done
```

Joonis 26. *Keskkonda loomine.*

```
The following NEW packages will be INSTALLED:

  ca-certificates    pkgs/main/win-64::ca-certificates-2024.3.11-haa95532_0
  libffi             pkgs/main/win-64::libffi-3.4.4-hd77b12b_1
  openssl            pkgs/main/win-64::openssl-3.0.13-h2bbff1b_1
  pip                pkgs/main/win-64::pip-24.0-py38haa95532_0
  python             pkgs/main/win-64::python-3.8.19-h1aa4202_0
  setuptools         pkgs/main/win-64::setuptools-69.5.1-py38haa95532_0
  sqlite             pkgs/main/win-64::sqlite-3.45.3-h2bbff1b_0
  vc                 pkgs/main/win-64::vc-14.2-h21ff451_1
  vs2015_runtime     pkgs/main/win-64::vs2015_runtime-14.27.29016-h5e58377_2
  wheel              pkgs/main/win-64::wheel-0.43.0-py38haa95532_0


Proceed ([y]/n)? y
```

Joonis 27. *Jah/Ei valik.*

Nüüd on vaja aktiveerida loodud keskkonda ja installida vajalikud teegid:

```
C:\Users\Artjom>conda activate py38

(py38) C:\Users\Artjom>
```

Joonis 28. `conda activate <env-name>` *käivitamine.*

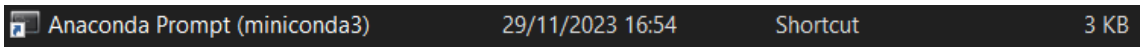
```
(py38) C:\Users\Artjom>pip install ultralytics
```

Joonis 29. `pip install ultralytics` *käivitamine.*

```
(py38) C:\Users\Artjom>pip install mediapipe
```

Joonis 30. `pip install mediapipe` *käivitamine.*

Pärast seda peab kasutaja sulgeda terminali aken ja leida oma arvutis järgmise rakenduse:



Joonis 31. *Miniconda käsurealiides.*

Siis kasutaja peab paremklõpsuga valida *Run as administrator* ja käivitada miniconda käsurealiidese. Peab ilmuma järgmine:

```
(base) C:\Windows\system32>
```

Joonis 32. *Administraatorina käivitamine.*

Nüüd peab aktiveerima keskkonda ja navigeerima ZED kaustale:

```
Administrator: Anaconda Prompt (miniconda3)

(base) C:\Windows\system32>conda activate py38

(py38) C:\Windows\system32>
```

Joonis 33. *Keskkonna aktiveerimine.*

```
(py38) C:\Windows\system32>cd C:\Program Files (x86)\ZED SDK

(py38) C:\Program Files (x86)\ZED SDK>
```

Joonis 34. *Navigeerimine ZED kausta.*

Seejärel on vaja käivitada käsku `python get_python_api.py`.

```
(py38) C:\Program Files (x86)\ZED SDK>python get_python_api.py
-> Downloading to 'C:\Program Files (x86)\ZED SDK'
Detected platform:
        win_amd64
        Python 3.8
        ZED SDK 4.1
-> Checking if https://download.stereolabs.com/zedsdk/4.1/whl/win_amd64/pyzed-4.1-cp3
8-cp38-win_amd64.whl exists and is available
-> Found ! Downloading python package into C:\Program Files (x86)\ZED SDK\pyzed-4.1-c
```

Joonis 35. *`python get_python_api.py` käsku rakendamine.*

Pärast seda on vaja sulgeda miniconda käsurealiides ja avada seda uuesti, nüüd seda saab teha tavakasutajana. Siis kasutaja peab minema kausta, kus allalaaditud rakendus asub, ja seejärel käivitada terminalis järgmise käsu: `conda activate <env-name> cd <kausta nimi>`.

Näides autoril on fail `the_component.py` kaustas `Desktop\the_component`.

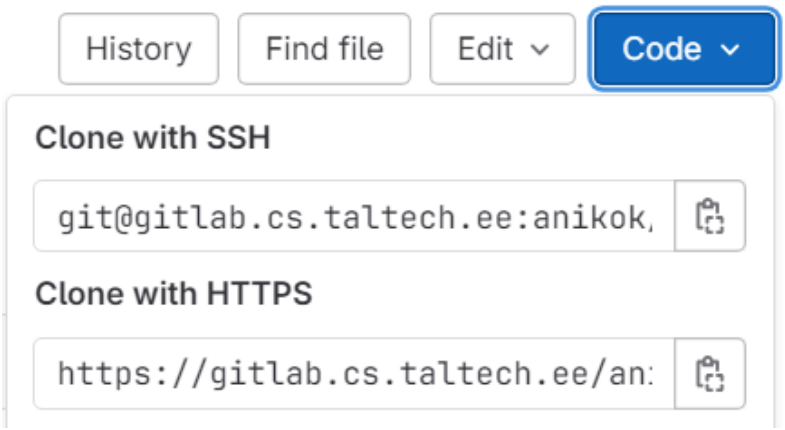
```
C:\Users\Artjom>conda activate py38

(py38) C:\Users\Artjom>cd Desktop/the_component

(py38) C:\Users\Artjom\Desktop\the_component>python the_component.py
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
Starting Network...
Starting camera...
```

Joonis 36. *Rakenduse käivitamine.*

Pärast seda saab loodud keskkonda kasutada erinevates IDE's (näiteks PyCharm või VSCode) rakenduse arendamiseks.

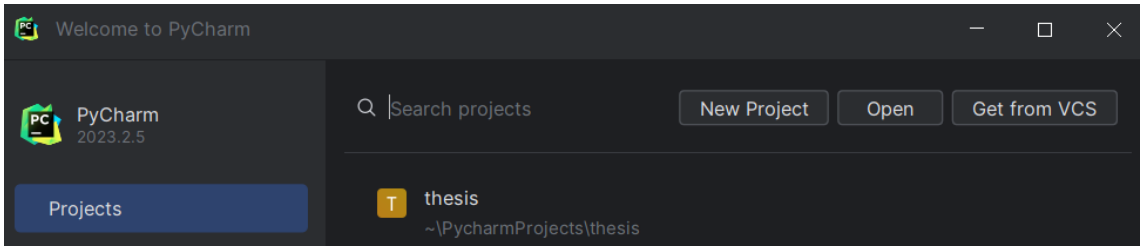
Rakenduse käivitamine PyCharm'is

PyCharm on üks populaarsemaid IDE Pyhon'i rakenduste arendamiseks. Selleks, et paigaldada käesoleva bakalurusetöö PyCharmi, on vaja teha järgmised sammud. Esiteks, on vaja navigeerida GitLabi ja kopeerida HTTPS lingi projektile:



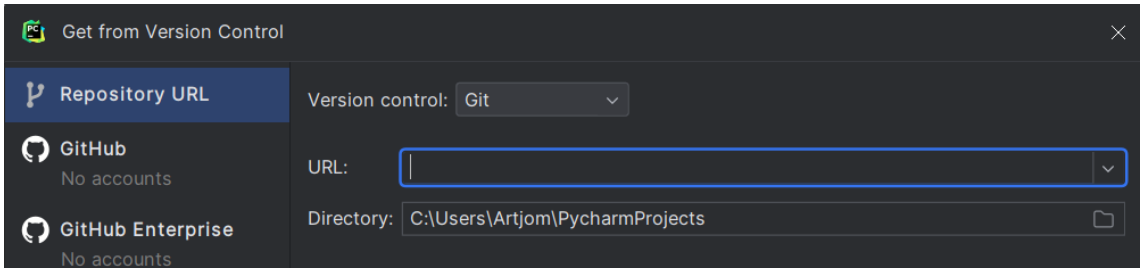
Joonis 37. *Lingi kopeerimine.*

Pärast seda on vaja luua PyCharm'is uue projekti, klõpsades nuppule *Get from VCS*:



Joonis 38. *Uue projekti loomine.*

Seejärel on vaja kleepida kopeeritud lingi URL alale:



Joonis 39. *Lingi kleepimine.*

Kui projekt on allalaaditud, peab klõpsama *File* ülevases vasakus nurgas, siis valida *Settings -> Project: <project-name> -> Python Interpreter -> Add interpreter -> Add local interpreter*, valida vasakus *Conda Environment*, valida paremal *Load Environments* ja endiselt valida rippmenüüst oma conda keskkonda ja klõpsada *OK*. Pärast seda on vaja klõpsada rohelisele ringile et see muutuks selliseks:

Joonis 40. *Conda Package Manager'i väljalülitamine.*

Lõpuks on vaja klõpsada *Apply* siis klõpsada *OK*, ja rakendus on arendamiseks valmis.

Lisa 7 – Rakenduse demo

Rakenduse demovideod on siin:

- Jetson Orin NANO demo monitoril - `https://youtu.be/cWGXhZJVi5A`
- Jetson Orin NANO demo kasutades väikset puuteekraani - `https://youtu.be/ghl9ySHBq2g`
- Linux PC demo kasutades monitori - `https://youtu.be/aftpTiHgzuc`