TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Tarkvaratehnika instituut

Jaanus Piip 121840IAPM

Sünesteesiasimulaatori arendamine

magistritöö

Juhendaja: PhD Aleksei Tepljakov

Tallinn 2019

TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies
Department of Software Sciences

Jaanus Piip 121840IAPM

Developing the Synesthesia Simulator

Master's thesis

Supervisor: PhD Aleksei Tepljakov

Tallinn 2019

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: Jaanus Piip

06.05.2019

Annotatsioon

Käesolev magistritöö implementeerib populaarses skriptimis- ja üldotstarbelises programmeerimiskeeles sünesteesiasimulaatori prototüübi uue iteratsiooni kandidaadi. Selle sünesteesiasimulaatori eesmärk on visualiseerida helide omadusi. Kandidaadi peamiseks kvaliteedikriteeriumiks seatakse viide heliallikaga seotud sündmusest selle kuvamiseni simulaatori kasutajale. Eesmärk üritatakse saavutada, kasutades olemasolevat riistvara. Programmeerimise käigus implementeeritakse eelnevat iteratsiooni dokumenteeriva uurimustöö soovitusi ning tulenevalt kasutatava lisarakenduse võimekusest lisavõimalusi samaaegselt visualiseerida mitut heliallikat. Kahes voorus korraldatavate eksperimentide käigus selgub varasema iteratsiooniga võrreldes nii edasiminek heliallika omaduste visualiseerimise viites kui ka regressioon heliallika liikumise viivituses. Heliallika liikumise trajektoori jälgimise täpsuse hindamisel selgub kasutatava lisarakenduse nõrkus, mis raskendab oluliselt jälgimise täpsuse hindamist. Arendusprotsessi ja eksperimenteerimise käigus tehtud tähelepanekutest tulenevalt soovitatakse mitut uut potentsiaalset suunda prototüübi järgmise iteratsiooni kasutatavuse parendamiseks.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 48 leheküljel, 5 peatükki, 21 joonist, 8 tabelit.

Abstract

Developing the Synesthesia Simulator

This thesis implements a candidate for the new iteration of a synesthesia simulator using a popular scripting and programming language. The purpose of this synesthesia simulator is to visualise the features of a sound source. The main quality criteria is set to be the latency between an event regarding the sound source and it being displayed to the user. The new candidate is implemented using already existing hardware. Some recommended features from the whitepaper detailing the previous iteration are implemented, the used tracking application also allows tracking multiple sources to be implemented. Two rounds of experiments reveal both an improvement in the latency of displaying audio features' changes and a regression in displaying of audio source's movement latency. While trying to quantify the accuracy of movement trajectory tracking the main weakness of the used tracking application reveals itself making assessing its accuracy very hard. Due to this several possible new development directions are discussed for the next iteration of the prototype.

This thesis is written in Estonian and contains 48 pages of text, 5 chapters, 21 figures, 8 tables.

Lühendite ja mõistete sõnastik

ALSA	*Advanced Linux Sound Architecture*
dB	detsibell
DTW	*Dynamic Time warping*
HMD	*Head-Mounted Display*, pähekinnitatav ekraanisüsteem
JSON	*JavaScript Object Notation*, javascripti objektinotatsioon
MFCC	*Mel-frequency cepstral coefficients*, Meli-skaalal kepstri kordajad
ms	millisekund
MSWAC	*Maximum Sliding Window Automatic Calibration*
ODAS	*Open embeddeD Audition System*
OLED	*Organic Light Emitting Diode*, orgaaniline valgustemiteeriv diood
OpenVR	*Open Virtual Reality*
SRP-PHAT	*Steered-Response Power Phase Transform*
UDP	*User Datagram Protocol*, kasutajadatagrammi protokoll
UE	*Unreal Engine*
VR	*Virtual Reality*, virtuaalreaalsus

Sisukord

Joonised

Tabelid

1 Sissejuhatus

Aastat 2016 võib lugeda virtuaalreaalsuse uueks tulemiseks - selle aasta märtsis ja juunis jõudsid kodukasutajate kätte kahe suure tegija Oculuse ja HTC virtuaalreaalsuskomplektid. Need kättesaadavad ja eelnevate põlvkondadega võrreldes suhteliselt odavad seadmed tõid virtuaalreaalsuse laiematesse massidesse kui kunagi varem. Kui varasemalt võis HMD, *Head-Mounted Display*, pähekinnitatav ekranisüsteem, maksta kuni 49,000$ [1], siis HTC Vive sai osta vaid 800$ [2] eest. Lihvitum tarkvara koos avatud lähtekoodiga rakendusliidestega on aidanud rakendada neid seadmeid peamiselt meelelahutuslike seadmetena, kuid on samas avanud ka uusi piltlike uksi nende tööriistade ja teadusetegemise vahenditena kasutamiseks. Virtuaalreaalsusel on potentsiaal olla kasutatud meditsiinivahendina, näiteks on juba aastal 2001 näidatud selle mõju valuaistingu vähendamisel põletusravi patsientide puhul [3]. Iroonilisel kombel aitab virtuaalreaalsuse püsimajäämisele kaasa suuresti hoopis pornograafia [4]. Võrreldes varasemate seadmetega toimunud kvaliteedihüpe on oluliselt parendanud kasutajakogemust ja pannud kasutajaid mõtlema, mida veel peale mängude saab virtuaalreaalsuses implementeerida. Teaduslikuks ja tehniliseks tööks toodetakse juba inimese nägemise eraldusvõimele lähenevaid seadmeid [5]. Võimalus näha maailma läbi kellegi teise pilgu on seni olnud käega katsutamatus kauguses, tõepoolest on see seda veel siiski. Küll aga on sellised tehnoloogilised arengud meid liigutanud lähemale võimalusele seda olukorda simuleerida. Ühe sellise maailma erinevalt tajumise põhjuse — sünesteesia, simuleerimiseks on TalTechis arendamisel prototüüp. Käsolev töö keskendub TalTechi Arukate Süsteemide keskuse virtuaalreaalsuse laboris arendatud sünesteesia-simulatsiooni prototüübi arendamisele läbi valmisteekide ja labori käsutuses oleva riistvara kasutamise.

Suurim eesmärk on uurida uue ODASe ehk *Open embeddeD Audition System*-nimelise mitme samaaegse heliallika asukoha jälgimist võimaldava rakenduse sobivust, parendamaks eelneva prototüübi kvaliteeti ja implementeerimaks varasemalt välja toodud ideid. Käesolev töö on jaotatud viieks. Esimeses peatükis seletatakse lahti prototüübiga seotud olulisemad terminid, selle käigus selgub ka üks peamisi headusekriteeriumeid, mille alusel valminud rakendust hinnata. Teises peatükis tehakse terviklik ülevaade nii prototüübi eelmisest iteratsioonist kui ka käesoleva töö raames uurimuslikult valminud rakenduse sisust, põhjalikumalt seletatakse lahti ka tehtud valikud. Kolmandas peatükis kirjeldatakse prototüübi praktilist külge, seatakse üles eksperimendid prototüübi sobivuse hindamiseks, tuuakse välja mõõtmiste tulemused. Neljas peatükk analüüsib loodut, ja käib välja ideid mida saaks nende parendamiseks teha ja kuhu suunda seda prototüüpi tulevikus arendada võiks. Viies peatükk pakub lühikese kokkuvõtte tehtust. Lisade alt leiab tehtud konfiguratsiooni, näite kasutatud virtuaalreaalsussüsteemiga eksperimenteerimise lihtsusest ja mõõtmiste tulemused.

1.1 Taustainfo

Järgnevalt seletatakse lahti olulisemad terminid ja algoritmid ning määratakse üks olulisematest näitajatest, mida prototüübikandidaadi puhul mõõta.

1.1.1 Sünesteesia

Sünesteesia (*syn*, *aisthēsis* - koos, taju) on fenomen, mille puhul ühe meele stimulatsiooni korral kogeb sünesteet tahtmatult ja järjepidevalt mõne ülejäänud meele aistingut [6].

Sünesteesia jaguneb projektiivseks sünesteesiaks ja assotsiatiivseks sünesteesiaks. Neist esimese puhul aistab sünesteet teise meelega midagi esimese stimulaatoriga seotut (näiteks: auto tüüpilise signaali heli kuulmise korral näeb sünesteet lillat värvi) ning teise puhul põhjustab seosega stimulatsioon sünesteedis tugeva tunde

(näiteks: sama signaali kuulmise peale seostab sünesteet kuuldu muutumatult lilla värviga) [6].

Sünesteesia tüüpe on dokumenteeritud vähemalt 80 [7], mille hulgast on käesoleva töö kontekstis olulisim kroomsteesia, mille korral seonduvad spetsiifilised helid või sagedused spetsiifiliste värvitoonidega [7].

Käesolevas töös on sünesteesiaaspektiks heliallika valjususe kuvamine virtuaalmaailmas helimarkeri suuruse muutumise läbi ja emiteeritud heli dominantse sagedussegmendi visualiseerimine läbi sama markeri pinnavärvi muutmise. Seos sageduse ja värvi vahel on fikseeritud ning ei sõltu kasutajast või kasutaja sünesteet olemisest.

1.1.2 Virtuaalreaalsus

Virtuaalreaalsus, VR, on realistlik spetsialiseerunud tarkvara ja riistvara kombinatsioonina loodud interaktiivne simulatsioon, mida kasutaja kontrollib läbi oma kehaliigutuste [8], [9]. Lihtsamal juhul jälgitakse kasutaja pea ja käte asendit. Keerulisematel juhtudel on võimalik lisada kasutaja kehale markereid, mille positsiooni HMD ja pultidega samadel tingimustel jälgitakse [8]. Virtuaalreaalsuse puhul võib tarkvaralist külge kõrvale jättes suureks osaks lugeda just kasutatava seadme tehnoloogilist võimekust ja füüsilisi omadusi nagu kaal ning kuju. Iga aspekt, mis kasutajale meelde tuletab, et ta pole kohas mida ta näeb, vähendab immersiooni, usutavust – kuivõrd suudab see kogemusena kasutaja reaalsest maailmast lahutada [10]. Sellist endasse neelavust võib täheldada paljude tegevuste juures: raamatute lugemise, filmide vaatamise ning arvutimängude puhul. Seega paistab, et mainitud omadus ei piirdu puhtalt tehnoloogilise võimekusega ning selle definitsioon sõltub kontekstist [10]. HMDde puhul on seepärast olulised näitajad ekraanide resolutsioon, värskendussagedus ja vaateväli. Siia lisanduvad veel läätsede erinevad optilised omadused ja HMD kaal. Rasket seadet, mis optiliste moonutuse ja kitsa vaateväljaga end konstantselt meelde tuletab, on raske unustada. Olulisim aspekt, mis VR puhul kasutaja kogemusest välja toob või lausa merehaiguse-laadseid sümptomeid põhjustab, [11] on latentsus pärismaailmas

toimuva ja virtuaalses maailmas toimuva vahel. Mitmes uurimustöös on demonstreeritud, et suurem viide liigutuse ja simulatsioonikuva vahel vähendab tunnetatud immersiooni ning ka näiteks motoorset võimekust ülesannete täitmisel virtuaalmaailmas [12], [11]. Käesoleva töö kontekstis loetakse oluliseks kasutatava virtuaalreaalsussüsteemi visuaalse külje immersioonivõime: peamiselt kuva täpsus ja latentsus reaalse maailma sündmuse ning kuvamuutuse vahel.

1.1.3 MFCC

MFCC ehk *Mel-frequency cepstral coefficients* ehk Meli-skaalal kepstrikordajad on populaarne viis leida helisignaali dominantsed sagedused. Näiteks on seda juba kaua kasutatud kõnetuvastuses.

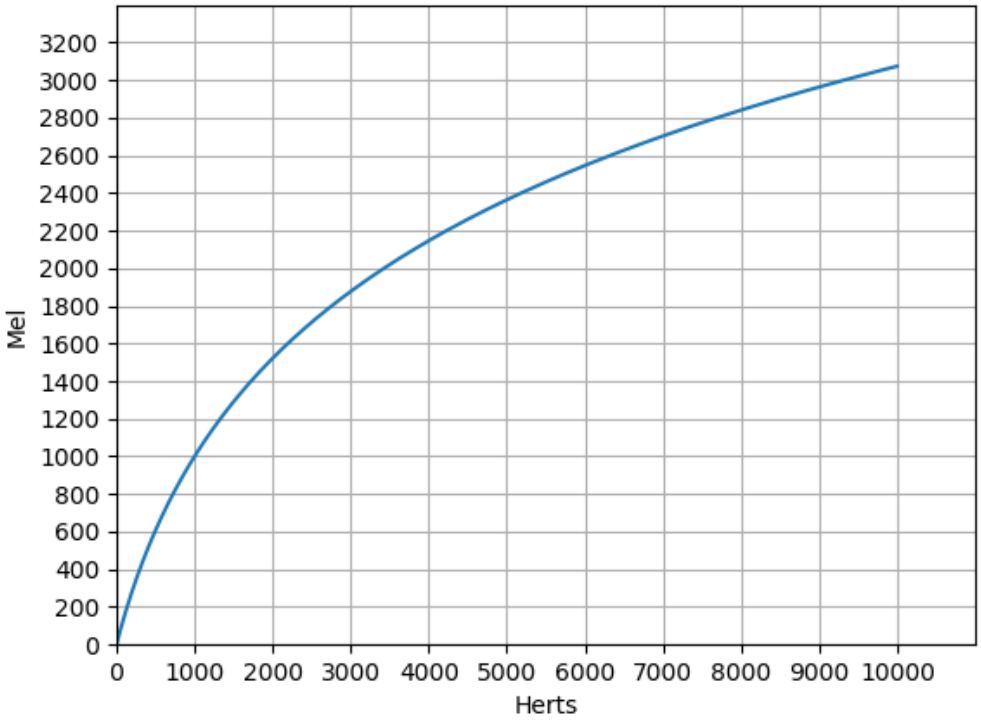
Meli-sagedusskaala on erinevalt lineaarsest Hertzi-skaalast mittelineaarne sagedusskaala, mille eesmärk on paremini sobida kokku inimkõrva logaritmilise kuulmisega. Skaaladevaheline seos on kujutatud joonisel 1. Levinum valem hertsidest melidesse teisendamiseks on $m = 2595log_{10}(1 + \frac{f}{100})$, kus m on sagedus melides ja f on sagedus hertsides.

Algoritmiliselt leitakse MFCCd järgnevalt:

1. Jaota signaal diskreetseteks kaadriteks.
2. Tee kaadrile diskreetne Fourier' teisendus.
3. Teisenda saadud väärtused Meli skaalasse.
4. Leia saadud väärtuste logaritmid.
5. Tee logaritmidele diskreetne koosinusteisendus.

1.1.4 SRP-PHAT

SRP-PHAT, ehk *Steered-Response Power Phase Transform* on populaarne heliallika suuna leidmise algoritm.



Joonis 1: Meli ja Hertzi skaala seos

Olgu M hulk mikrofone, m üks mikrofon hulgast $\{1, ..., M\}$ ning diskreetse ajahetke signaali väärtus ühest mikrofonist $s_m(n)$.

Diskreetse ajasammu ruumipunktist x mikrofonimassiivi suunas lähtuv helienergia on avaldatav kui

$$P(x) \triangleq \sum_{n \in \mathbb{Z}} \left| \sum_{m=1}^{M} s_m(n - \tau_m(x)) \right|^2 \tag{1}$$

kus $\tau_m(x)$ on ruumipunktist x lähtuva heli kohale jõudmiseks kuluv aeg ja $\mathbb{Z}$ on täisarvude hulk. Heliallika tõenäoline suund leitakse, arvutades otsinguruumi kõikides punktides maksimaalsed väärtused. Üks viis selle algoritmi keerukuse vähendamiseks on otsinguruumi piiramine.

1.1.5 DTW

DTW, ehk *dynamic time warping* on algoritm, mis võimaldab võrrelda kahe aegrea omavahelist sarnasust. DTW eeliseks on aegridade erineva tempo, faasi või pikkuse lubamine.

Charu C. Aggarwal defineerib oma andmekaevandamise õpikus DTW algoritmi rekursiivselt järgnevalt [13]: olgu $DTW(i,j)$ kahe aegrea $\overline{X} = (x_1, ..., x_m)$ ja $\overline{Y} = (y_1, ..., y_n)$ viimaste vastavalt i ja j elemendi vaheline optimaalne distants.

$$DTW(i,j) = distance(x_i, y_j) + min \begin{cases} DTW(i, j-1) & repeat\ x_i \\ DTW(i-1, j) & repeat\ y_j \\ DTW(i-1, j-1) & repeat\ neither \end{cases} \quad (2)$$

Funktsioon $distance(x_i, y_j)$ on mõni valitud distantsiarvutusmeetod, näiteks Manhattani kaugus või eukleidiline kaugus.

1.2 Seotud tööd

Tsinghua ülikooli psühholoogiateaduskonna teadlased uurisid meeltevahelise infovahetuse mõju, sünesteesiat, kuidas katsealused raporteerisid erinevate teede maitset, kui selle välimust mõjutati virtuaalreaalsust kasutades [14]. Püstitati kolm küsimust: kas tee päris värvi nägemine virtuaalreaalsuses mõjutab katsealuse maitsehinnangut teele? Kas subjekti poolt tee maitsega seostatava värvi simuleerimine virtuaalreaalsuses mõjutab katsealuse hinnangut tee maitsele? Kas laiemalt levinud konkreetse maitsega seostatava värvuse simuleerimine virtuaalreaalsuses mõjutab subjekti hinnangut tee maitsele? Töö tulemustest saab lühidalt välja tuua, et nähtud tee värvus mõjutab subjekti hinnangut tee maitsele juhul, kui nähtav värvus on subjekti jaoks seostatud orienteeruva maitsega - assotsiatiivne sünesteesia [14].

Michigani ülikoolis uuriti heliefektide mõju keskkonna usutavusele lihtsate geomeetrilisete objektide liigutamisel virtuaalreaalsuses [15]. Täpsemalt uuriti heli-

ja visuaalsete efektide mõju taktiilsete efektide võimendamisel. Katsealustele anti ülesandeks virtuaalses keskkonnas liigutada käsi kasutades geomeetrilisi primitiive. Visuaalse efektina eksisteeris objekti küljes vedru, millest haarates objekti liigutamiseks venis alustuseks vedru ja seejärel hüppas liigutatav objekt liigutusele järele. Heliefektidena kasutati tüüpilist vedru venitamise häält ja objekti järele hüppamisel kostuvat klõpsu. Eksperimendid jagati neljaks, kus iga efekt kas eksisteeris või ei eksisteerinud. Osa katsealustest raporteeris visuaalse efekti olemasolul objekte liigutades suurenenud tunnet, nagu omaks liigutatav objekt massi ja heliefekti olemasolul suurenenud immersiooni komponenti - kohalolekutunnet [15].

BurnAR autorid kasutasid liitreaalsussüsteemi tekitamaks illusioon peakomplekti kandja käte põlemisest [16]. Kasutatud virtuaalreaalsusplatvorm koosnes kaameratega HMDst, tarkvarakihist, mis videopildi alusel rekonstrueeris kasutaja käte pinnad kolmemõõtmelise punktipilvena ja mängumootorist, mis tekitas sisendiks saadud käte-punktipilvedele suitsu ning tuleefekti. Viimase kihi tulemusega kaeti esialgne videopilt, mis siis kasutajale ekraanidel tagasi mängiti. Illusiooni tugevdas kasutajale mängitav põlemise heli, mille valjusust reguleeriti vastavalt käte kaugusele kasutaja peast. Avalikul demonstratsioonil kinnitas pea viiendik rohkem kui sajast proovijast, et tundsid illusiooni nägemise ajal kätes mainimisväärset soojaaistingut. Sellest lähtuvalt korraldati eetikakomiteelt nõusolek saaduna formaalne eksperiment eelnevalt liitreaalsuse uudsuse suhtes desensibiliseeritud 20 vabatahtliku katsealuse peal. 5-minutise liitreaalsuskogemuse järel raporteeris katsealustest 6 tahtmatut soojatunnet oma kätes. Avaliku demonstratsiooni anekdotaalne viiendik ja formaalse katse 30% viitavad, et mainimisväärses hulgas inimestes võiks olla võimalik usutava sisendi abil tekitada sünteetiline sünesteesia [16].

Töös [17] visualiseeriti heli varieerivas suuruses keradena, pannes kera värvi sõltuma heli sagedusest. Töö [18] keskendus pigem psühhedeelikale, selles töös lisati visualiseerimispoolele kehaülene aktuaatoritega ülikond, stimuleerimaks visualisatsioonikogeja erinevaid kehapiirkondi Mõlemal juhul kasutati sageduste

esitamiseks kogu värvispektrit. Kuna kummaski töös pole katsed eksperimendina formaliseeritud ega välja toodud katsealuste kohta lisainformatsiooni, peab võtma mõlemas välja toodud tagasisidet anekdotaalselt, kuid välja on toodud positiivsed kommentaarid visualisatsioonis viibimise järel tekkiva sünesteesiatunde ja vibratsiooniülikonna poolt pakutav positiivne lisakogemuse kohta [17], [18].

Ross [19] analüüsis nelja "kogemus"-mängu ja kahe 360° filmi näitel elemente, mis suurendavad vastava kogemuse sünesteetilist kogemust ja immersiooni. Siduv element oli pea alati interaktiivsus, detailid olid aga erinevad. Veealuseid stseene pakkuva *TheBlu* puhul toodi oluliste lisadetailidena välja vee rõhumistunde tekitajatena heljuvaid osakesi ja õhumulle. *Richie's Plank Experience*, mis viib kogeja kõrgele plangule kõndima, peeti oluliseks kõrgusesse tõusmise erilist rõhutamist, mis suutvat ka kindlalt reaalsuses viibivas inimeses plangult maha astumise puhul tugeva kiire langemisega seostatava tundumuse tekitada [19].

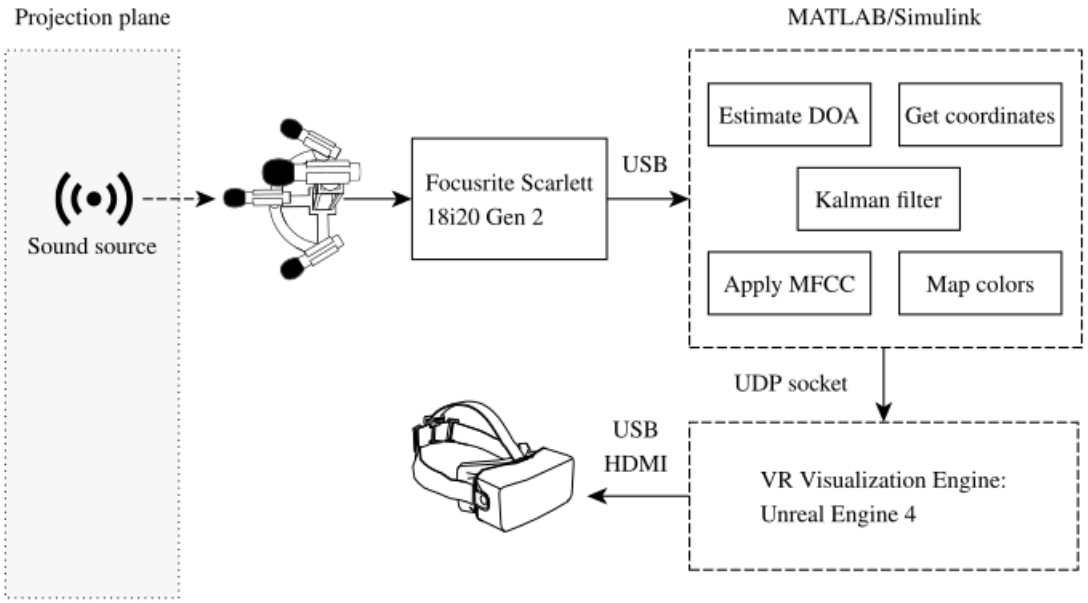
TalTechis töötati välja autorile teadaolevalt esimene sünesteesiasimulaator, protsess on avaldatud töödes [20], [21] ja [22]. Erinevate algoritmide hulgast implementeeritakse SRP-PHAT ja valitakse heli dominantsete sageduste määramiseks MFCC algoritm lihtsama kiire Fourier' teisenduse asemel. Seda põhjusel, et Meli skaala väljendab täpsemalt inimeste helitaju, mis pole kuuldavas sagedusalas lineaarne. Mikrofonisisendite töötlemine ning väljundi filtreerimine Kalmani filtri abil implementeeriti Matlabis. Visualiseerimiseks valiti välja UE, Unreal Engine, selle lihtsa kuid efektiivse objektide skriptimise süsteemi tõttu. See võimaldas UEs lihtsasti UDP, *User Datagram Protocol*, võrguliiklust vastu võtta võimaldava pistikprogrammi abil saavutada info vastuvõtmise sagedust $f_s = 1kHz$. Kasutatav mikrofonimassiiv tegi läbi ühe iteratsiooni ringikujulisest joonisel 5 nähtavaks koonuseliseks [20], [21], [22].

1.2.1 Eksisteeriv prototüüp

Eksisteeriv prototüüp on antud ajahetkeks läbinud mõned iteratsioonid ning koosneb ühest nelja mikrofoniga mikrofonimassiivist, Matlabi programmist, mis tegeleb heliallika positsioneerimise, liikumise trajektoori silumise, helianalüüsi ja

väljundi vormindamisega ning Unreal Engines jooksvast visualiseerimismootorist [22].

Eelneva prototüübi tarvis oli Matlabis implementeeritud SRP-PHAT algoritm. SRP-PHAT-i kirjelduse leiab peatükist 1.1.4. Prototüübi üldloogika on kujutatud joonisel 2.

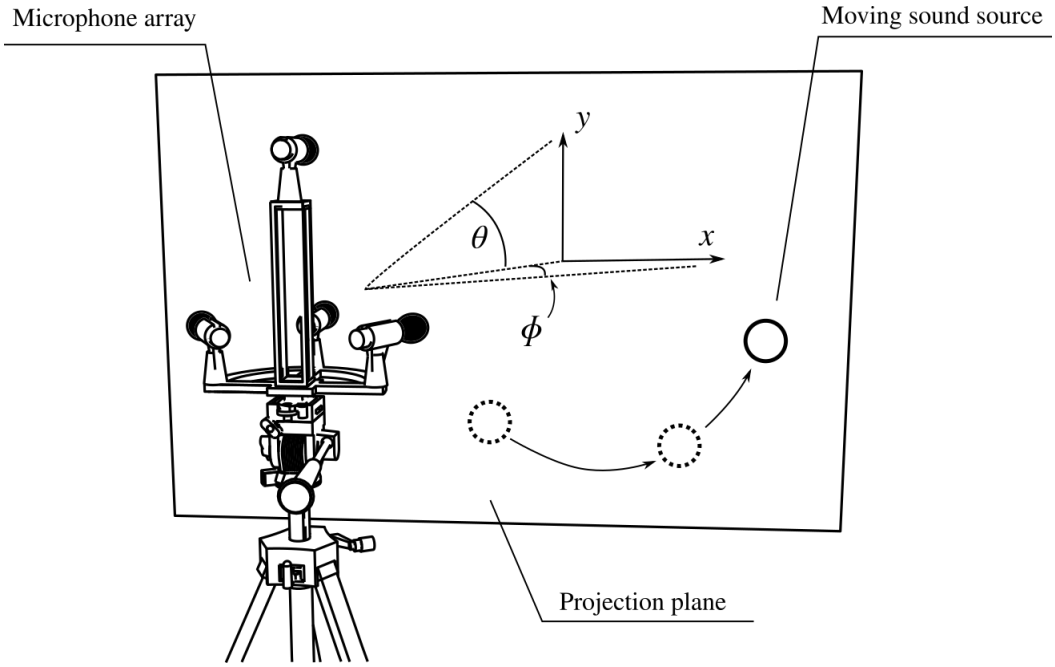


Joonis 2: Eelmise prototüübiversiooni protsess [22]

Ülesande lihtsustamiseks eeldati, et heliallikas asub mikrofonimassiivist fikseeritud kaugusel ja Matlabi programmi väljund asimuudi ϕ ja elevatsiooni θ näol projitseeriti tasapinnale, saamaks Unreal Engine jaoks sobilikud kahemõõtmelise pinna koordinaadid x ja y. Positsiooni väljund on kujutatud joonisel 3. Heliallika markerina virtuaalmaailmas kasutati kera, mille suurus sõltus heli valjususest ja värv leiti dominantse sagedussegmendi järgi vabalt valitud värvikaardilt [22].

Eelmise prototüübi iteratsiooniga katseid läbi viies jõudsid Köse, Tepljakov jt. järgmiste tulemusteni:

- Prototüüp positsioneerib rääkija korrektselt, kuid inimkõne ja näiteks ka moodne muusika ei erine oma sagedusvahemikus piisavalt virtuaalreaalsuses kuvatud markeri värvi oluliseks varieerumiseks [22].



Joonis 3: Eelmise prototüübiversiooni väljund [22]

- Prototüüp omab umbes 500-millisekundilist viidet reaalse maailma olukorra muutuse ja HMDs kuvatud pildi muutumise vahel [22].

Mainitud prototüübiga katsetades käidi välja järgnevad parendusettepanekud:

- Implementeerida lihtne kasutajaliides simulatsiooniparameetrite muutmiseks (värvikaart, värviskaala) [22].
- Implementeerida võimekus leida heliallika kaugus mikrofonimassiivist [22].

2 Prototüübi iteratsiooni komponendid

Uue prototüübi loomisel kasutati maksimaalselt ära juba labori valduses olevat riistvara. Sellest eesmärgist erinevad potentsiaalsed arendussuunad tuuakse välja töö lõpus.

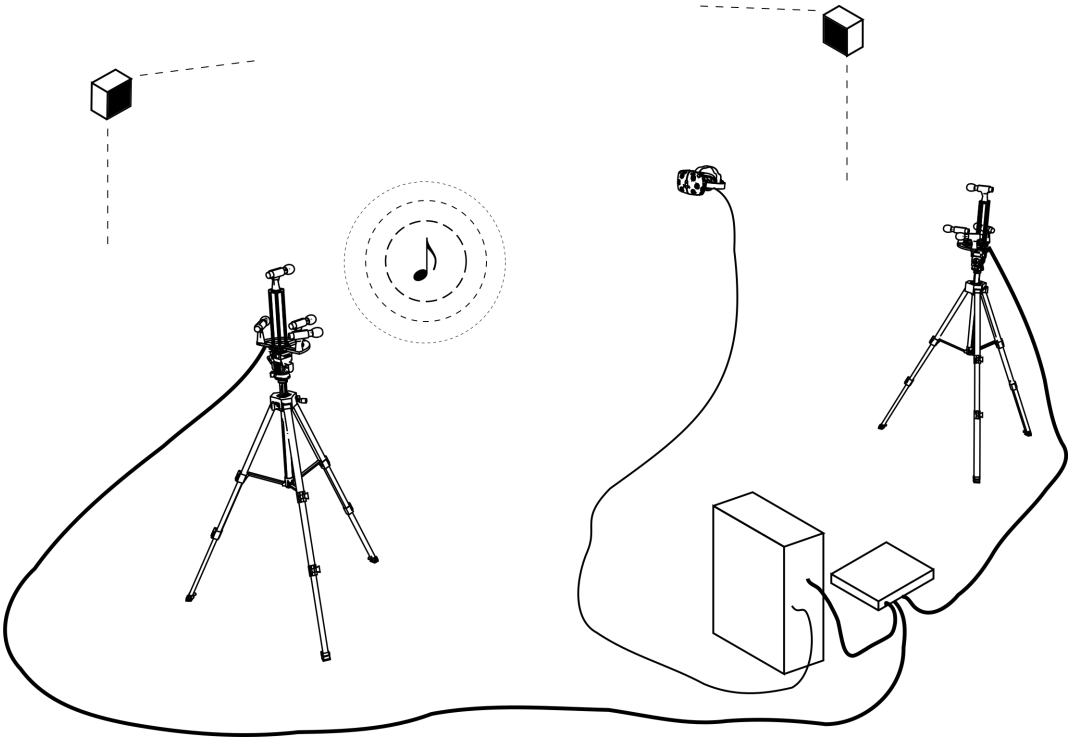
Terviklik süsteem on kujutatud joonisel 4:

- 2 teineteisest sõltumatut ODASe protsessi
  - 2 mikrofonimassiivi:
    - statiiv
    - 3d-prinditud raam mikrofonide hoidmiseks
    - 4 mikrofoni Behringer C-2 [23]
  - helikaart Scarlett 18i20 [24]
- antud magistritöö raames loodud trianguleerija
- Unreal Engine 4 jaoks Köse jt. loodud lihtne UDP info vastuvõtja [25]
- autori loodud Unreal Engine 4 jooksev tulemust renderdav lihtne testmaailm

Kogu arendatud programm koosneb kuuest erinevast protsessist, millest kolme puhul jookseb paralleelselt kuni neli koopiat.

- põhiprotsess
- heliinfot vastu võttev lõim
- 2 positsiooninfot vastu võtvat lõime
- 4 heli analüüsimise protsessi

- trianguleerimisprotsess
- 4 tulemusi kompileerivat ja UEle edastavat protsessi



Joonis 4: Prototüübikandidaadi lihtsustatud riistvaraline skeem

2.1 Riistvara

Andmaks prototüübist terviklik ülevaade tuuakse järgnevalt lühidalt välja kasutatud riistvaralised komponendid ja nende eesmärgid.

Mikrofonimassiivid

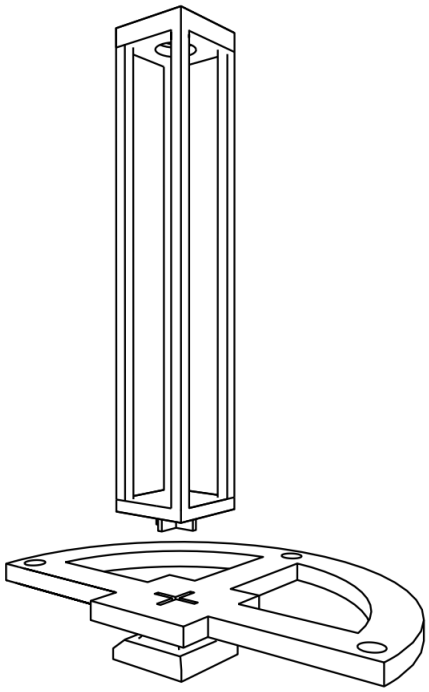
Kasutatavad mikrofonimassiivid koosnevad kolmest komponendist:

- kiirkinnitusega statiiv
- 3d-prinditud raam mikrofonide hoidmiseks

- 4 mikrofoni Behringer C-2

Massiivi raam

Mikrofone hoidvad raamid joonisel 5 loodi eelmise iteratsiooni käigus kasutades sulatatud plastiklõnga deponeerimise võimalusi. See protsess on üks mitmest, mida saab kokkuvõtvalt nimetada 3d-printimiseks. Raami sisse on projekteeritud avad, kuhu kinnitatakse mikrofonidega komplekteeritud hoidikud, mis võimaldavad muuta nende pööret ning kallet.



Joonis 5: Mikrofonimassiivi raam

Kasutatavad raamid modelleeriti eelneva prototüübi loomise käigus vabavaralises modelleerimisrakenduses Blender, printimiseks kasutati printerit Ultimaker 3.

Focusrite Scarlett 18i20

Scarlett 18i20 on stuudiohelikaart, mida kasutatakse muuhulgas selle arvukate mikrofonisisendite tõttu. Kokku omab Scarlett 18i20 kaheksat mikrofonisisendit,

millest esimest nelikut kasutab peamine ODASe instants ja viimast nelikut teine ODASe instants. Helikaardi mikrofonisisendi karakteristikud on välja toodud tabelis 1. Selleks, et oleks võimalik kasutada üht helikaarti kahe iseseisva rakenduse poolt, lisati ALSA ehk *Advanced Linux Sound Architecture* konfiguratsioonifaili .asroundrc virtuaalne heliseade nimega *"dsnooper"*. Tehtud konfiguratsioon on välja toodud lisas A. Mõlemad ODASe instantsid kasutavad seadet *dsnooper*, kuid kaardistavad end kuulama erinevaid mikrofone.

Tabel 1: Scarlett 18i20 mikrofonisisendi karakteristikud

Sageduskarakteristik	20 Hz - 20 kHz +0.5/-1.5 dB
Dünaamiline diapasoon	109 dB (A-Weighted)
Mittelineaarmoonutus	<0.002%
Müralävi	-127 dBu
Maksimaalne sisend	+8.5 dBu
Võimendus	50 dB

HTC Vive

HTC Vive on üks esimesi moodsaid masstootena tarbijale suunatud virtuaalreaalsuse komplekte. Komplekt koosneb jälgimiseks tarvilikest majakatest, pultidest, pähe käivast ekraanist ja adapterkarbist, mis sisaldab majakatega suhtlemiseks tarvilikku IEEE 802.15 raadioseadet. Vive HMD sisaldab kahte eraldiseisvat OLED, orgaanilise valgustemiteerivate dioodide ekraani resolutsiooniga 1080×1200 värskendussagedusega 90Hz [26]. Läbi seadme läätsede avaneb kasutajale 110-kraadine vaateväli pikslitihedusega 447 pikslit tolli kohta [26]. Pultide ja peaseadme liikumist jälgivate majakate latentsus objekti liikumisest liikumise kuvani on <11 ms ja objekti positsiooni jälgimise täpsus on 2 millimeetrit [27].

Üldotstarbelised arvutid

Kuna ODAS ja selle monitoorimisrakendus ODAS Studio on mõlemad ette nähtud UNIXilaadsetel operatsioonisüsteemidel kasutamiseks, kasutati nii arendamisel kui eksperimenteerimisel vähemalt kaht üldotstarbelist arvutit. Esimesel neist jooksis Ubuntu Linux ja teisel Windows 10. Kõik kolm arvutit lisati ruuteri abil privaatsesse kohtvõrku.

Kasutatav heliallikate suunda määrav rakendus on suhteliselt vähenõudlik: i7-2640M protsessoriga (2 füüsilist tuuma, 4 paralleelset lõime) püsis protsessorikasutus summaarses režiimis 10 protsendi kandis, mis jätab ruumi objektide jälgimise algoritmi keerukamaks seadistamiseks. ODASe instantse jooksutav arvuti ei pidanud omama graafilist võimekust virtuaalreaalsuse komplekti ja Unreal Engine jooksutamiseks. UEd jooksutav masin oli jõulisem protsessoriga i5-7600 (füüsilist tuuma, 4 parallelset lõime).

Suhtlus ODASeid jooksutava masina ja VR-keskkonda ning arendatud tarkvaratükki jooksutava masina vahel toimus üle kohtvõrgu TCP protokolli kasutades. Trianguleerimise tulemus saadeti UEd töös hoidvasse masinasse UDP protokolli kasutades.

2.2 Tarkvara

Järgnevalt tuuakse välja arendatud prototüübikandidaadi tarkvaralised osad. Siia kuuluvad kasutatud rakendus nimega ODAS ja arendatud trianguleerija komponendid. Lahti seletatakse trianguleerimisloogika ja lühidalt tuuakse välja ka tarkvara üles seadmise sammud.

2.2.1 ODAS

ODAS ehk Open Embedded Audio System on GPL 3.0 litsensiga avatud lähtekoodiga rakendus, mis on kirjutatud keeles C, võimaldamaks selle kasutamist

lihtsal odaval riistvaral. Rakenduse peamine kasutusvaldkond on robootika ja inimese-roboti interaktsioonide hõlbustamine [28]. Rakendusse on lisatud võimalused mitme samaaegse heliallika positsiooni jälgimiseks, nende emiteeritud helide eristamiseks ja taustamüra väljafiltreerimiseks.

Algoritmina heliallikate lokaliseerimiseks kasutab ODAS sarnaselt esialgse prototüübiga SRP-PHAT algoritmi. Võrreldes tavapärase SRP-PHAT-iga on seda täiendatud ja nimetatakse SRP-PHAT-HSDA-ks.

HSDA ehk *Hierarchical Search with Directivity model and Automatic calibration* lisab siia järgneva:

- Mikrofonide suunsus: vältimaks mikrofonide arvu M kasvuga kaasnevat keerukuse kasvu $O(M^2)$ kasutatakse seadistatud mikrofonide suunsust, ignoreerimaks arvutustsüklis ebaolulisi mikrofone [29].
- MSWAC, *Maximum Sliding Window Automatic Calibration*: heliallika suuna otsingu aknasuurus arvutatakse automaatselt enne rakenduse käivitamist, sõltuvalt konfiguratsioonis määratud mikrofoni asukoha ja suuna positsiooni veast.
- Hierarhiline otsing: initsialisatsioonisammuna luuakse konfiguratsioonijärgsed progresseeruva tihedusega võrestikud ning nendevahelise kaardistuse maatriksid. Potentsiaalseid heliallikaid otsides tehakse seda alustuseks madalama eraldusvõimega võrestikus. Potentsiaalse heliallika suuna leidmisel kasutatakse eelgenereeritud kaardistusmaatrikseid sama piirkonna rafineeritumas otsinguvõrestikus välja otsimiseks. Samme korratakse, kuni eksisteerib rafineeritum võrestik [29].

ODASe kasutatud versioon võimaldab kirjutada positsioneerimisinfot JSON-ina faili või edastada seda üle võrgu. Heliinfo edastatakse baidijadana kas faili või samuti üle võrgu. Arendustöö lihtsustamiseks kasutame üle võrgu edastamise võimalusi. See võimaldab lõpptarbijana kasutada ka Windowsi-põhist arvutit.

Oma orientatsioonilt on ODAS ette nähtud tegelema inimkõnega, see tähendab, et vaikimisi seadistused näevad ette helisignaali diskreetimissagedust 16 kilohertsi.

Seega on maksimaalne sagedus, mida saab käesoleva töö raames tuvastada, 8 kilohertsi. Katsetades ODASe sisemise diskreetimissageduse tõstmisega jõuti tulemuseni: selle tõstmine kolm korda suurendas latentsust kaks korda. Puhvrite ja töösagedust prooviti tõsta just kolm korda, kuna see oli esimene tüüpiline kasutav sagedus, mis lisaks jagus jäägita eelneva diskreetimissagedusega.

2.2.2 Arendatud rakenduse kirjeldus

Käesoleva töö suurim eesmärk oli implementeerida Pythonis horisontaalselt skaleeruv ja minimaalset viivitust tekitav programm, mis tõlgiks ODASe väljundi eksisteerivale UE UDP pluginale sobivaks. Lisandusid parendused nelja objekti paralleelse jälgimise ja sügavusinfo välja arvutamise ODASe sekundaarse instantsi abil. Arhitektuurimärksõna käesoleva rakenduse arendamisel oli "parallelsus", kasutamaks maksimaalselt ära tänapäevaste protsessorite rööprähklemise võimekust. Kätte jõudnud pooljuhtide tootmise litograafia kahanemise hetk (TSMC 7nm) toob kaasa uue tuumade arvu kasvu protsessoris. See kombineerituna juba tavapäraseks saanud mitme lõime üheaegse jooksutamise võimekusega soosib parallelseid protsesse. Kirjutusstiili üritati hoida maksimaalselt PEP 8-lähedane.

Prototüübi kasutamiseks on vaja seadistada järgnev:

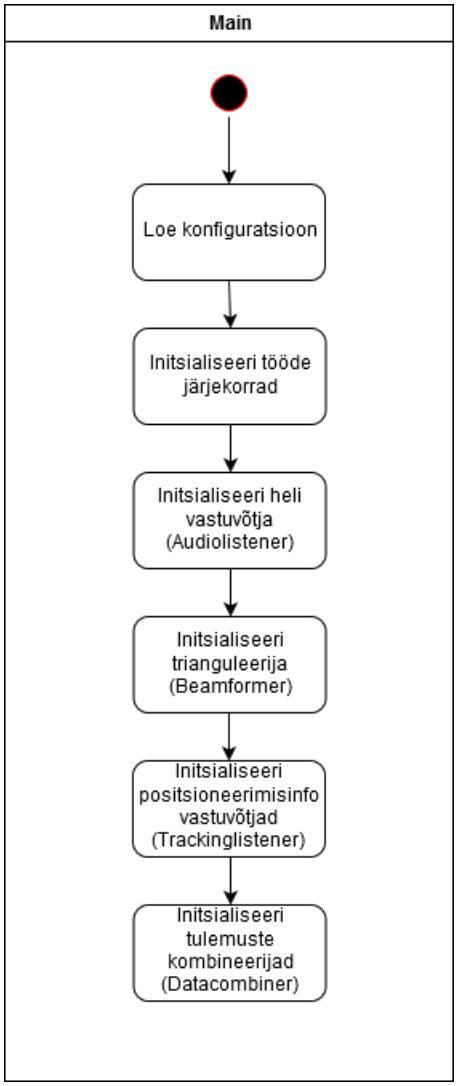
1. Seadistada ODASeid jookutavas Linuxi-masinas ALSA heliseade, mis lubaks mitmel erineval rakendusel samaaegselt kasutada üht ja sama sisendseadet.
2. Seadistada kaks ODASe instantsi saatma erinevatel portidel jälgitud heliallikate positsiooniinfot.
3. Seadistada ainult primaarne ODASe instants saatma allikate eraldatud heliribasid.
4. Seadistada mõlema ODASe instantsi jaoks vastava mikrofonimassiivi mikrofonide asukohad, suunad ja asendi määramatus.
5. Seadistada trinaguleerijas mikrofonimassiivide mõõdetud asukohad ja suunad

UE koordinaadistikus.

6. Seadistada ODASe instantsides, trianguleerijas ja UEs siht- ja lähtevõrguaadressid.

Main

Arendatud rakenduse main-failis initsialiseerime töödejärjekorrad ja käivitame kõik töölisprotsessid. See võimaldab hoida rakenduse mitteblokeeruvana, kui on vaja mingi sisend seal anda. Põhitsükli loogika on kujutatud joonisel 6.



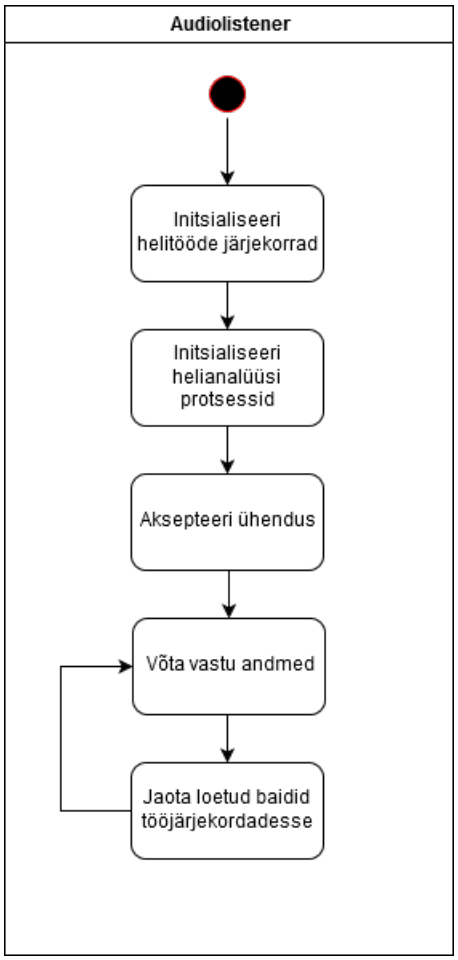
Joonis 6: Põhitsükli loogika

Audiolistener

Audiolistener on funktsioon, mida jooksutatakse ODASe põhiinstantsi poolt tuleva eraldatud heliinfo vastu võtmiseks. Selles lõimes kuulatakse võrgust tulevat baidijada, jaotatakse see neljaks eraldiseisvaks heliribaks ja söödetakse 640 baidi kaupa (20 millisekundit) samas lõimes viidatud eraldiseisvale audioworker-protsessi töödejärjekorda.

Neljaks heliribaks jaotamise loogika leiab ODAS web failist recordings.js

Heliinfo vastu võtmise loogika on kujutatud joonisel 7.



Joonis 7: Heliinfo vastu võtmine ja jaotamine

Audioworker

Audiolistener tekitab neli töölisprotsessi - iga ODASes jälgitava objekti kohta üks tööline. Eraldi protsessina eksisteerimine tahab tavaliste järjekordade asemel *multiprocessing* järjekordade kasutamist, nendeks on üks sisend- ja üks väljundjärjekord info tagasi põhiprotsessi liigutamiseks. Lühidalt on töölise ülesanne lugeda oma tööjärjekorrast kvant baite, lahendada oma nihkuv aken, määrata helivõimsus, tuvastada helijupis dominantne sagedussektsioon ja leida sellele vastav värv. Kõik töölises kasutatavad funktsioonid on eraldi moodulis nimega "audiofunctions.py".

Prototüübi eelmise iteratsiooni lahenduste järgselt leitakse heli dominantne sagedussegment MFCC abil ja helivõimsuse ruutkeskmise võimsusena.

Heli valjususe ja dominantse sageduse leidmise loogika on kujutatud joonisel 8.

Trackinglistener

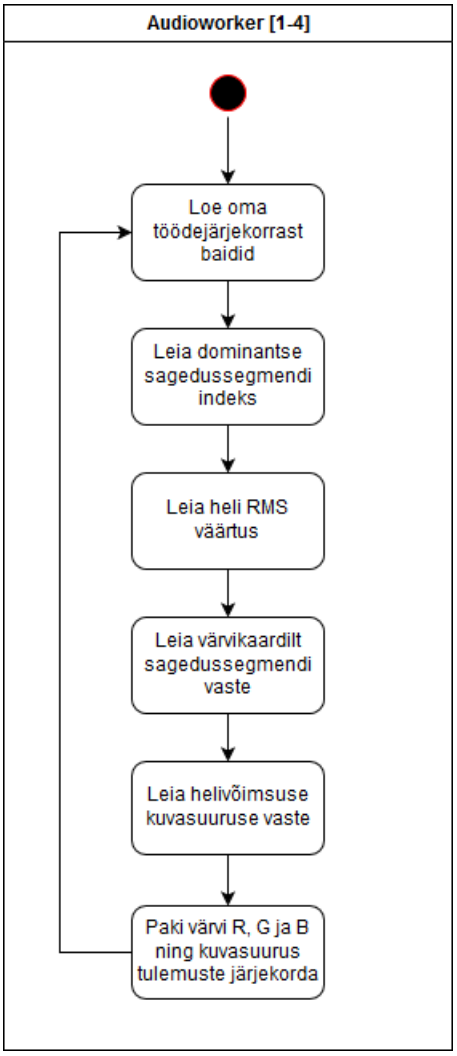
Trackinglistener on funktsioon, mida jooksutatakse eraldi lõimena ning mille eesmärk on vastu võtta ODASelt jälgitavate objektide positsiooniinfot, dekodeerida see stringiks ja muuta parsimiseks legaalseks JSON-objektiks. Saadud objektist küsitakse välja kõigi nelja jälgitud objekti suund, mis pakitakse 1D 12-liikmeliseks numpy massiiviks ning pannakse trianguleerija tööjärjekorda.

Neid lõimesid tekitatakse kaks: lõim indeksiga 0 on põhilõim (kuulab primaarse ODASe infot pordil 8000), lõim indeksiga 1 on sekundaarne lõim (kuulab sekundaarse ODASe infot pordil 8001).

Objektide positsiooniinfo vastu võtmise loogika on kujutatud joonisel 9.

Beamformer

Beamformer on käesoleva töö üks olulisemaid funktsioone. Korrektsem oleks seda nimetada trianguleerijaks. Kõik kasutatavad abifunktsioonid asuvad failis

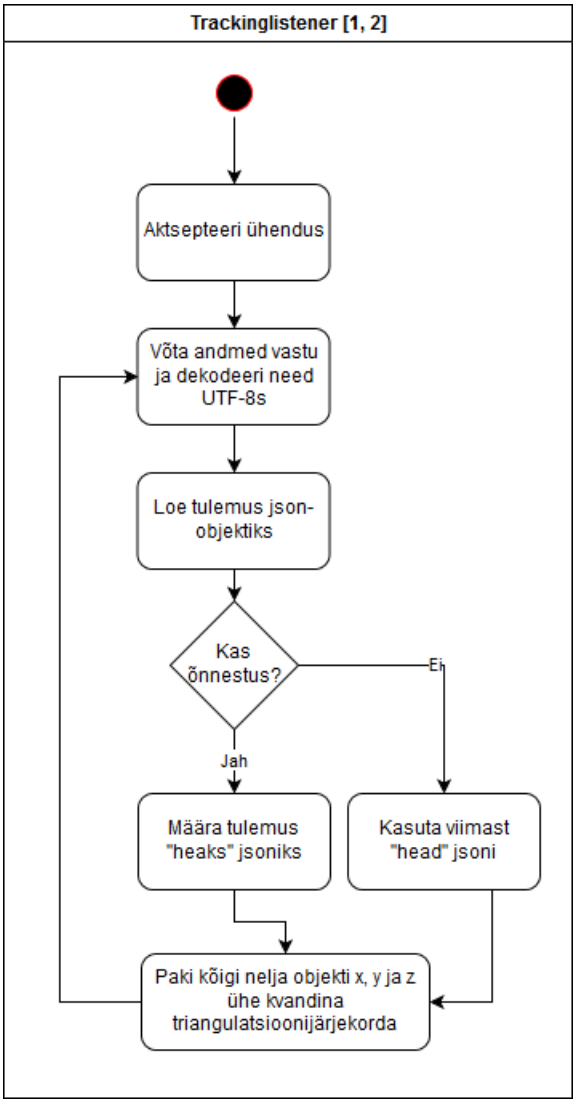


Joonis 8: Heli analüüsitsükkel

"positionfunctions.py". Loogikas kasutatakse kiiruse huvides maksimaalselt numpy funktsionaalsust ning lihtsustatult on see kujutatud joonisel 10.

Töötsükli algul küsitakse mõlemast sisendjärjekorrast üks 12-liikmeline ühemõõtmeline numpy massiiv. Mõlemad massiivid sisaldavad endas nelja jälgitava objekti suunavektoreid arvestusega, et vastava massiivi keskpunkt on vastava koordinaadistiku nullpunkt. Sekundaarse instantsi info esialgse instantsi koordinaadistikku paigutamiseks tehakse sekundaarse järjekorra elementidele ning vastavale "nullpunktile" enne edasist töötlemist pööre ja nihe, mille väärtused arvutatakse välja massiivide paiknemise konfiguratsioonist.

Kuna me teame iga objekti suunavektori alguspunkti, milleks on seda jälgiva



Joonis 9: Objektide positsiooniinfo kuulamine

massiivi nullpunkt, saame tõmmata läbi nende kahe punkti lõpmatu pikkusega sirgjoone. Trianguleerija konstrueerib iga objekti jaoks sirgjoone, leiab nendevahelise minimaalse distantsi, valib lühimate distantsidega paarid ja kui nende paaride distants on väiksem kasutaja valitud seadistatavast lävendist, siis leitakse lisaks keskpunkt, mis on neil kahel sirgjoonel asuvate lühima distantsi moodustavate punktide geomeetrilises keskpunktis. Leitud keskpunkt on jälgitud objekti oletatav asukoht. Leitud geomeetriline keskpunkt pannakse positsioonitulemuste järjekorda datacombinerile edasiseks töötlemiseks.

Juhul kui üks järjekord on tühi, tühjendatakse ka teine desünkroniseerumise

vältimiseks (eksperimendid näitavad, et isegi resursside jagumisel kipub üks ODASe instants rohkem jälgimisinfot saatma).

Triangulatsiooniloogika

Iga sirgjoone saab avaldada läbi selle läbitava punkti ja suunavektori võrrandiga

$$p = r + te \tag{3}$$

kus p on suvaline koordinaat vaadeldaval sirgjoonel, r on teadaolev punkt, mida vaadeldav joon läbib, t on suvaline skalaarväärtus ja e on suunavektor.

Kuna ODASe väljund on punkt ühikvektori kaugusel mikrofonimassiivi nullpunktist, saame konstrueerida selle nullpunkti ja jälgitava objekti suunavektori eelmist võrrandit kasutades lõpmatu sirgjoone.

Sama saame teha sekundaarse ODASe väljundiga pärast tolle nullpunkti ja jälgitava objekti suunda tähistava punkti roteerimist ja transleerimist primaarse ODASe koordinaadistikku. Rotatsiooni ja translatsiooni viime läbi teadmisega sekundaarse massiivi asukohast ja suunavektorist. Pärast rotatsiooni leiame teise sirgjoone tekitamiseks teise joone suunavektori transleeritud sekundaarse massiivi nullpunktist roteeritud ja transleeritud sekundaarse objektile osutava punkti vahel lahutades viimasest transleeritud nullpunkti.

Rotatsioonimaatriksi saab leida, teades mikrofonimassiivide suunavektoreid. Need kolmemõõtmelised ühikvektorid määratakse mikrofonimassiivide asukoha konfigureerimisel.

Rotatsioonimaatriksi R saab avaldada kui

$$R = I + [v]_\times + [v]_\times^2 \frac{1-c}{s^2} \tag{4}$$

kus

$$v = a \times b \tag{5}$$

$$s = \|v\| \tag{6}$$

$$c = a \cdot b \tag{7}$$

$$[v]_\times \stackrel{\text{def}}{=} \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix} \tag{8}$$

Seega oleme leidnud kahte sirgjoont väljendavad punktid ja suunavektorid ning saame avaldada suvalise punkti neil joontel 9 ja 10 nende kummagi peal võrrandi 3 abil.

$$p_1 = r_1 + t_1 e_1 \tag{9}$$

$$p_2 = r_2 + t_2 e_2 \tag{10}$$

Eeldusel, et mõlemad massiivid jälgivad sama objekti, peavad need kaks sirgjoont kas lõikuma või sattuma teineteisele suhteliselt lähedale. Seega on vaja leida nende kahe joone vaheline minimaalne distants.

Teame, et lühima distansi moodustavate punktide vahel asuv sirgjoon on risti kummagi suunavektoriga. Seega saame tekitada nende punktide vahelise joone suunavektori:

$$n = e_1 \times e_2 \tag{11}$$

Nüüd saame nende joonte vahelise distantsi leida võrrandiga

$$d = \frac{n \cdot p_1}{\|n\|} - \frac{n \cdot p_2}{\|n\|} = \frac{n \cdot (p_1 - p_2)}{\|n\|} = \frac{n \cdot (r_1 - r_2 + t_1 e_1 - t_2 e_2)}{\|n\|} \tag{12}$$

Kuna

$$n \cdot e_1 = n \cdot e_2 = 0 \tag{13}$$

siis

$$d = \frac{n \cdot (r_1 - r_2)}{\|n\|} \tag{14}$$

Kui leitud distants jääb alla valitud distantsilävendi, saame oletada, et nende kahe sirgjoone vahelise lühima joone keskpunkti lähedal asub meie jälgitav objekt.

Selle keskpunkti leidmiseks peame avaldama punktid p_1 ja p_2, mille vahel mainitud lühim distants asub. Transleerides teist sirgjoont mööda n-i moodustub tasand, millel asub p_2 ja see on risti n_2-ga.

$$n_2 = e_2 \times (e_1 \times e_2) \tag{15}$$

Seega avaldub:

$$p_1 = r_1 + \frac{(r_2 - r_1) \cdot n_2}{e_1 \cdot n_2} \cdot e_1 \tag{16}$$

Sarnaselt:

$$n_1 = e_2 \times (e_2 \times e_1) \tag{17}$$

ja seega

$$p_2 = r_2 + \frac{(r_1 - r_2) \cdot n_1}{e_2 \cdot n_1} \cdot e_2 \tag{18}$$

p_1 ja p_2 vaheline keskpunkt p_m on avaldatav võrrandist

$$p_m = \frac{p_1 + p_2}{2} \tag{19}$$

Datacombiner

Datacombiner on funktsioon, mis käivitatakse eraldi protsessina ning mille eesmärk on pärida oma heliallika positsioonijärjekorrast asukohainfo, välimusejärjekorrast värvi ja suuruseinfo, need kombineerida ning need käivitamisel määratud porti saata. Selle funktsiooni tööprotsess on lihtne: küsi oma mõlemast järjekorrast infot, kui seda kohe saada pole, kasuta eelnevat. Sellele järgnevalt maga minimaalne aeg ja korda protsessi. Selline lähenemine ei ole efektiivne protsessorikasutus, kuid on osa andmete töötlemisele kuluva aja minimiseerimisest.

Datacombineri teisejärguline funktsioon on sundida markerina kasutatavad kerad kahanema läbi eelneva väärtuse kasutamisel selle skalaariga 0,5 läbikorrutamise.

Andmete kombineerimise ja edastamise loogika on kujutatud joonisel 11

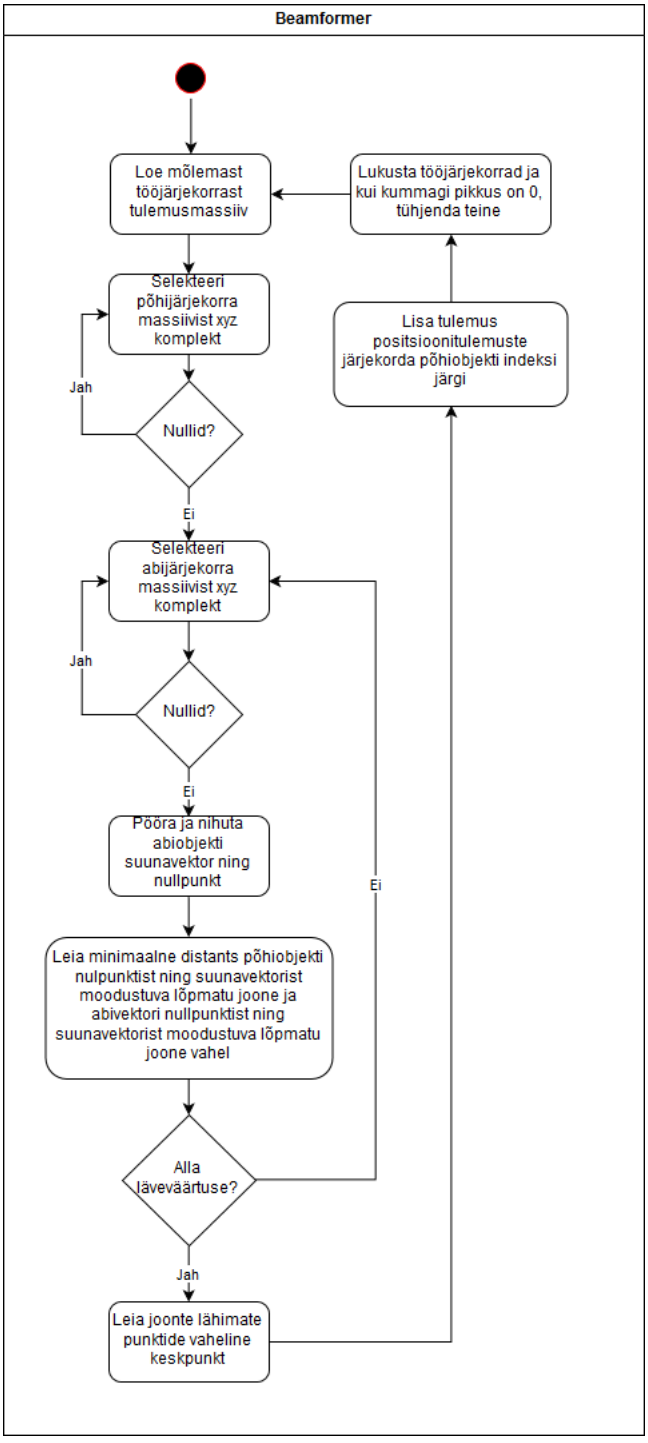
Unreal Engine testmaailm

Projekti katsetamiseks lõi käesoleva töö autor ka lihtsa testmaailma Unreal Engine abil. Sünteesiakuva on sarnane esimese prototüübiga: heliallika tuvastatud asukohta kuvatakse kera, mille pinna värv sõltub allika tekitatud heli dominantsest toonist ning suurus heli valjususest. Erinevuseks eelnevast prototüübist on see, et kera ei liigu kasutaja suunas, vaid muudab oma suurust.

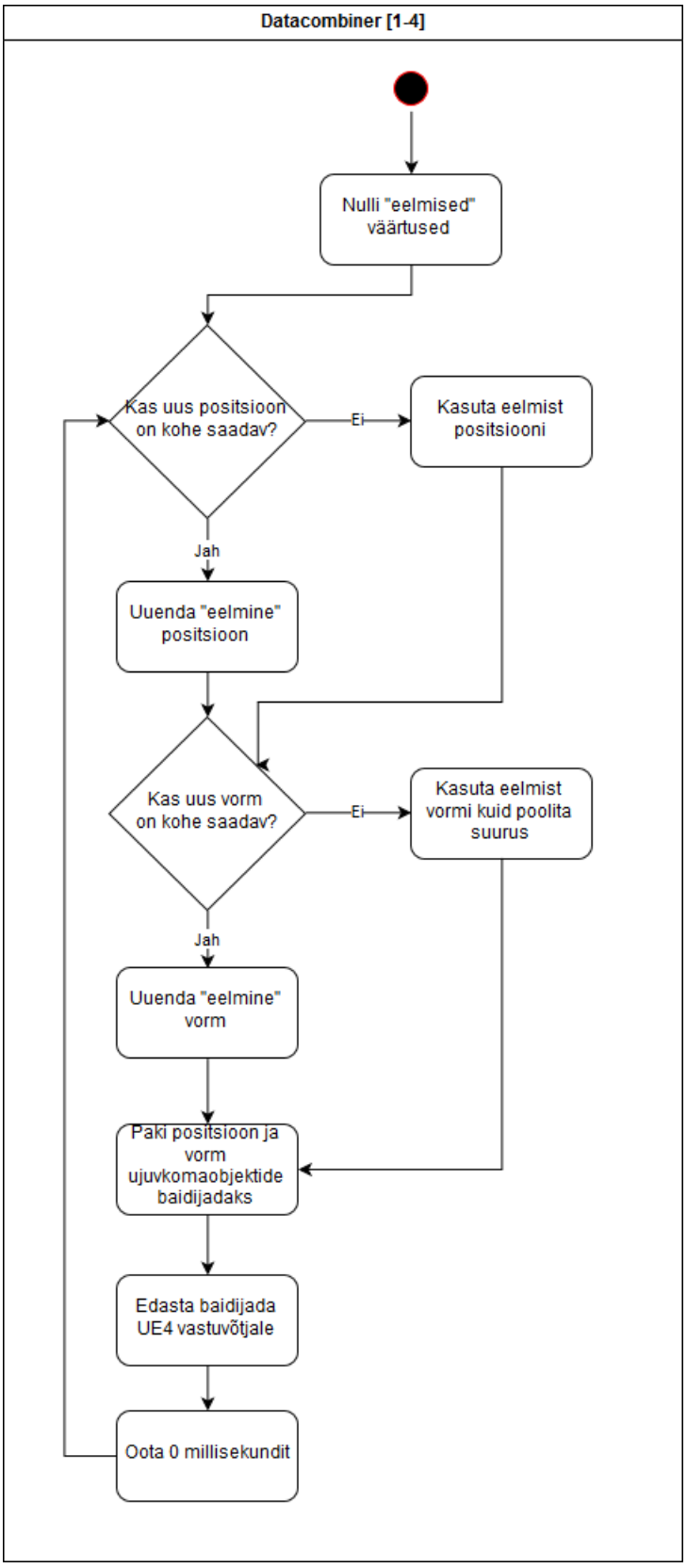
Selleks lisati tühja maailma põrand, instantsieeriti 4 UDP vastuvõtja objekti [25] portidel 11111 kuni 11114 ja lisati maailma positsiooni lihtsamaks visualiseerimiseks ka mikrofonimassiivide asukohad.

Kera värvi R, G ja B väärtused leitakse käesoleva projekti Pythonis kirjutatud osas helianalüüsi ajal. UDP liiklust kuulava pistikprogrammi abil võetakse ujuvkomaarvudeks kokku pakitud väärtused vastu ja muudetakse vastava objekti x, y ja z koordinaati, skaalat ning dünaamilise materjali pinnavärvi.

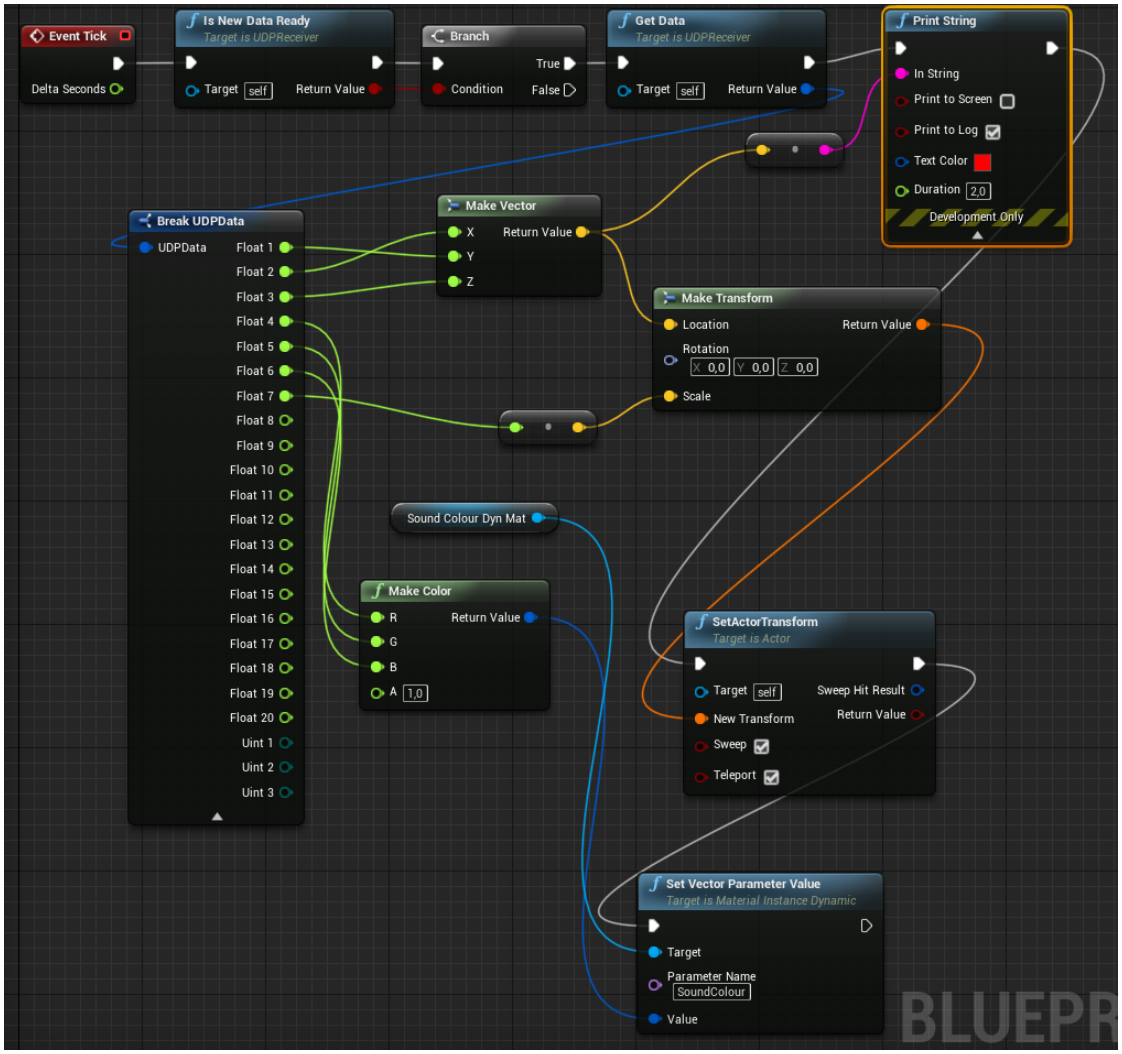
Esimese vastuvõtja loogikaskeem on kujutatud joonisel 12.



Joonis 10: Trianguleerimisloogika



Joonis 11: Positsiooni- ja vormiinfo kombineerimine ning edastamine



Joonis 12: UDP vastuvõtja loogikaskeem

3 Eksperimendid ja tulemused

Eksperimendid toimusid kahes voorus: prototüübikandidaat laborisse üles seatuna mõõdeti sellega väike hulk viiteid ja liikumistäpsust, sellele järgnes koodi optimeerimine, muudatuste külmutamine ja primaarne mõõtmisvoor. Järgnevalt kirjeldatakse prototüübi laboris üles seadmist, esimest mõõtmistevooru ja peamist mõõtmisvooru.

3.1 Mõõtmiste üles seadmine

Mõõtmistulemuste saamiseks seati laboris üles 2,5-meetrise küljepikkusega VR-ala, mille äärde seati üles pea- ja sekundaarne mikrofonimassiiv. Mõõtmisala on kujutatud fotol 13. Ideaalsel juhul oleks selle alal ümbrus mööblist vaba. Seda nii põhjusel, et isegi mänguala piiri meelde tuletava virtuaalseinaga kiputakse mööblisse sisse kõndima kui ka etteruttavalt põhjusel, et varajased katsed näitasid ODASe tundlikkust kajale ja helipeegeldustele. Näiteks tekitas massiivist mööda kõndimine visalt kaduva jälgitavate heliallikate raja. Kuna helikaardi iga mikrofonisisendi väljundvõimsus on kontrollitav omaette nupuga, siis keerati see kõikide mikrofonide jaoks maksimaalsesse asendisse ja lülitati mikrofonil asuv filtrivalik asendisse -10dB. Mikrofoni väljundi vähendamine 10dB võrra on oluline vähendamaks nii tubaste kui ka hooneväliste müraallikate mõju mõõtmistulemustele. Sellest hoolimata, nagu juba teises lokatsioonis seda rakendust arendades selgeks sai, kippus prototüüp tänu oma ülitundlikele mikrofonidele ja tasemel helikaardile üles korjama kajasid ja ka näiteks liiklust tänaval läbi nii avatud kui ka suletud akende.

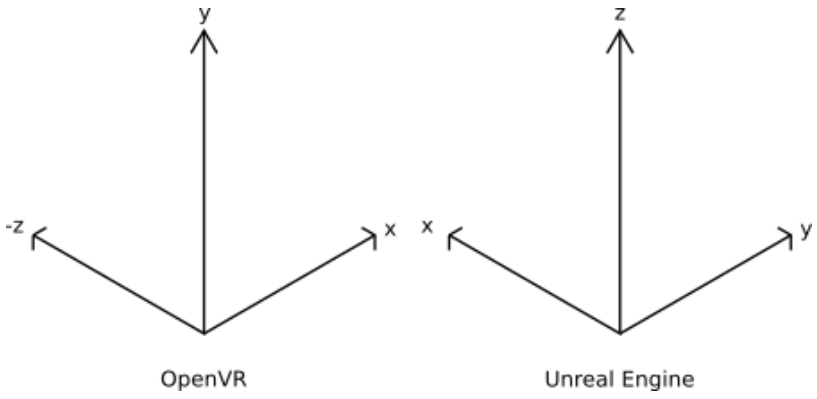
Joonis 13: Vaade eksperimendialale

Kuna samaaegselt läbi Vive HMD läätse selle ekraani ja mõõduala üheaegne filmimine pole HMDd lammutamata või spetsiaalset riistvara kasutamata võimalik, pidi käesoleva töö raames piirduma sündmusest kuva muutumiseni kuluva aja hindamiseks piirduma mõõteala ja kuvamootorit jooksuvatava arvuti monitori pildi samaaegse filmimisega. Sündmus on siinkohal kas järsk heli või heliallika liikumise algus. Filmimiseks kasutati GoPro kaamerat Hero 3, mis suutis salvestada videot 240 kaadrit sekundis lahutusvõimega 768 × 480 pikslit.

Tuvastatud trajektoori täpsuse hindamiseks kasutati heliallikana väikest portatiivset kõlarit, mille peale kinnitati takjaribaga Vive Tracker - eraldiseisev moodul, mis toimib HTC Vive majakasüsteemis ja edastab ülejäänud süsteemile enda positsioneerimisinfot. Heliallikas on kujutatud fotol 14. Sarnaselt teiste Vive ökosüsteemi osadega võib Vive Trackeri puhul arvestada täpsusega 2 millimeetrit [27]. Vive Trackeri positsiooniinfo salvestamiseks kasutati modifitseeritud Triad Semiconductori *openvri* pakendajat, kus lisaks väljundi konsooli kirjutamisele kirjutati see reavahetuste lisamise järel tekstifaili. Modifitseeritud kood on välja

Joonis 14: Komplekteeritud heliallikas

toodud lisas B. Unreal Engine positsiooniväljundi saamiseks piisas selle printimisest logisse. DTWd arvutati Rjadnev-Meristo magistritöö [30] raames kogutud andmete töötlemiseks kirjutatud lihtsa rakendusega. Mainitud töö autor modifitseeris eelnevalt käesoleva töö autori poolt instrueerituna oma rakendust vastu võtma käesoleva töö väljundit.



Joonis 15: Teljestike erinevused

3.1.1 Koordinaadistike vahel teisendamine

Kuna OpenVR ja Unreal Engine kasutavad erinevaid koordinaadistikke on vaja mõõtmistulemusi ette valmistada. Koordinaadistike erinevused on kujutatud joonisel 15. Kuna virtuaalreaalsuse mänguala ja mõõtmisala/UE teljed pole suure tõenäosusega paralleelsed, on vaja kõikide UE väljundkoordinaatide peal rakendada rotatsioon ja nihe, teisendamaks nad OpenVR koordinaadistikku. Ka koordinaadistike skaalad on erinevad: OpenVR puhul on ühik meeter, Unreal Engine puhul sentimeeter - seega on mõõtmistulemuste ettevalmistamisel vaja läbi viia ka vastavad teisendused.

Rotatsiooni kahe koordinaadistiku vahel saab leida, kasutades Kabschi algoritmi [31]. Mainitud algoritmi mõte on leida optimaalne punktipilve rotatsioon samas koordinaadistikus, kuid teades kolme punkti koordinaate kummaski koordinaadistikus saame leitava rotatsioonimaatriksi näol maatriksi, mille rotatsioon näitab teljestike rotatsiooni teineteise suhtes. Need fikseeritud kolm ruumipunkti on: peamassiivi nullpunkt, abimassiivi nullpunkt ja massiivide üles seadmisel maha märgitud massiivide suunavektorite ristumispunkt. Kuna mõlemad teljestikud on paralleelsed põrandapinnaga, võime neid punkte vaadelda ilma kõrguseta kahemõõtmelisel tasandil.

3.2 Esialgsed katsed ja tulemused

Saamaks ettekujutus prototüübikandidaadi sobivusest selle laboris esmakordsel ülesseadmisel viis autor läbi lühikesed esialgsed mõõtmised, saamaks esmased viiteväärtused ja süsteemi täpsuse orienteeruva hinnangu. Esialgsed viitemõõtmised viidi läbi väikese $n = 5$ valimiga. Liikumise täpsuse hindamiseks liigutati heliallikat kolmes mustris kolm korda, andes koguvalimiks $n = 9$.

3.2.1 Heli ja liikumise viide

Kui video kaadrisagedus sekundis on $f_s = 240$, siis kaadritevaheline ajaaken on $\Delta_s = 4,17$ millisekundit. Liikumisest selle kuvamiseni kuluva aja mõõtmiseks sätiti samaaegselt kaadrisse tool juhtmevaba kõlariga ja mängumootorit jooksutava arvuti monitor. Tooli liigutati mõõduka kiirusega 5 korda. Viiteväärtuse saamiseks loeti kaadreid esimesest kaadrist, kui tooli nihkumine muutub arusaadavaks, kaadrini, kus ekraanil on näha heliallika nihkumist. Tulemustes oletame, et puudub eksperimenteerijapoolne inimlik viga. Tulemused on välja toodud tabelis 2.

Tabel 2: Liikumisest kuva muutuseni esimene mõõtmine

	Kaadreid	**Millisekundeid**
	381	1589
	215	896.6
	165	688.1
	260	1084
	345	1439
Keskmine	273.2	1139
Mediaan	266.6	1112
75%	345.0	1439
Standardhälve	89.51	373.4
Keskmine absoluuthälve	71.84	299.7

Heliviivituse mõõtmiseks sätiti kaadrisse mõõtmisala ja mängumootorit jooksutava arvuti monitor. Samaaegselt mängiva heliga tehti mõõtmisalas 5 plaksu. Kaadreid loeti käte kokkupuutumise hetkest kaadrini kui monitorilt paistis jälgitud heliallika kasvamine järsu valju heli tõttu. Tulemused on välja toodud tabelis 3.

3.2.2 Liikumise jälgimise täpsus

Liikumise täpsuse hindamiseks kasutati kolme mustrit: vertikaalne ring, vertikaalne rist ja horisontaalne romb. Kõikide liigutuste ajal seisis eksperimenteerija

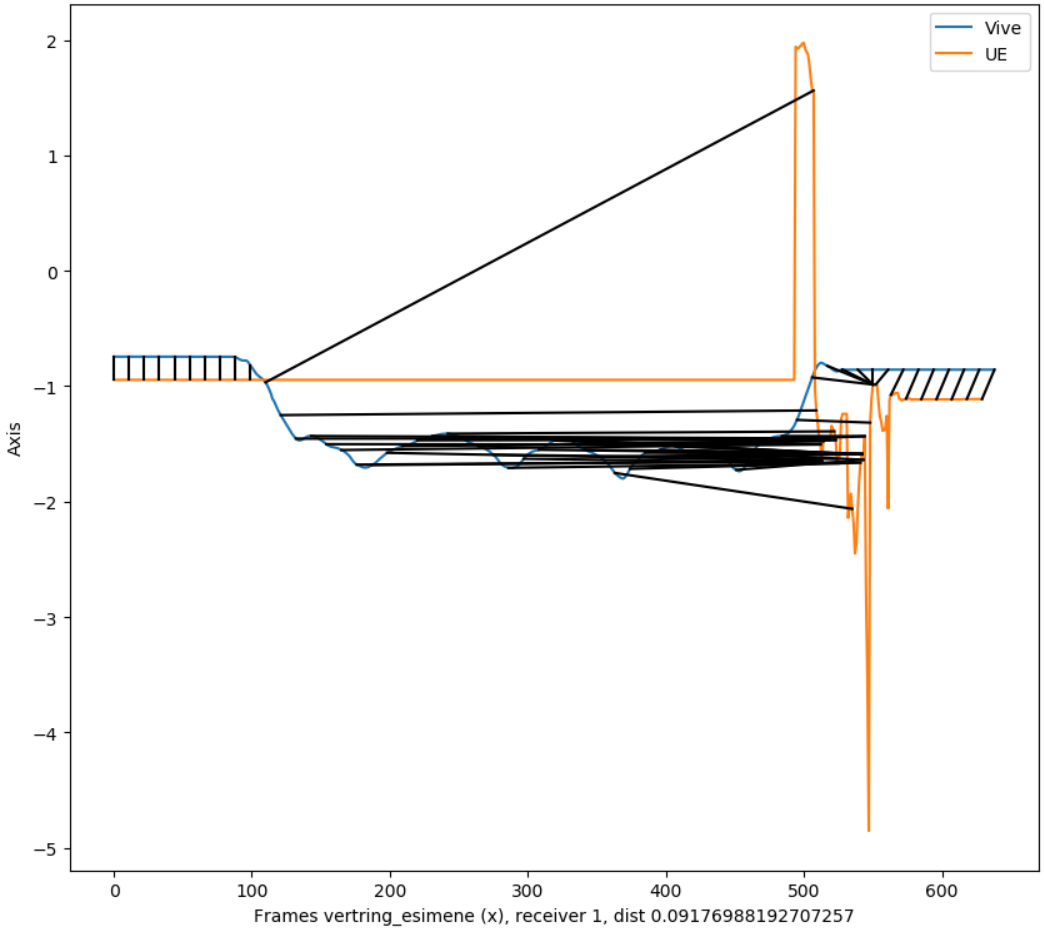
Tabel 3: Plaksust kuva muutuseni esimene mõõtmine

	Kaadreid	**Millisekundeid**
	26	108.4
	23	95.91
	25	104.3
	23	95.91
	20	83.40
Keskmine	23.40	97.58
Mediaan	23.20	96.75
75%	25.00	104.3
Standardhälve	2.302	9.603
Keskmine absoluuthälve	1.680	7.013

mõõtmisala keskpunktist pool sammu kaugemal, orienteeruvalt 45° nurga all kummagi mikrofonimassiivi suhtes, näoga nende vahel asuva nurga suunas. Heliallikat liigutati mõõduka konstantse tempoga. Trajektoori täpsuse planeeritud hindamine jäeti lühikeseks mõõtmise ebakorrektse läbiviimise tõttu: mõõdetud andmetest jäid puudu katsete algusajad, mis muutis DTW arvutamise raskeks. Eelnev kombineeritud graafikutel paistva asukoha trianguleerimise kehva täpsusega näitas, et kogu mõõdetu läbi töötamine annaks tulemuseks peamiselt müra. Puudulik tulemus on demonstreeritud joonistel 16, 17 ja 18, kus on võrreldud OpenVR teljestikus telje kaupa vertikaalse ringi UEst teisendatud infot ja Vive Trackeriga salvestatud OpenVR koordinaadijada. Nagu joonistelt paistab, on UE logitud aegreast palju positsiooniinfot puudu. Sellest tulenevalt annab DTW anomaalseid tulemusi.

3.3 Primaarsed mõõtmised

Viimase mõõtemistevooru eel toimus koodi optimeerimine suunaga peamiselt vähendada liikumisele reageerimise viivitust. Selleks optimeeriti oluliselt naiivselt

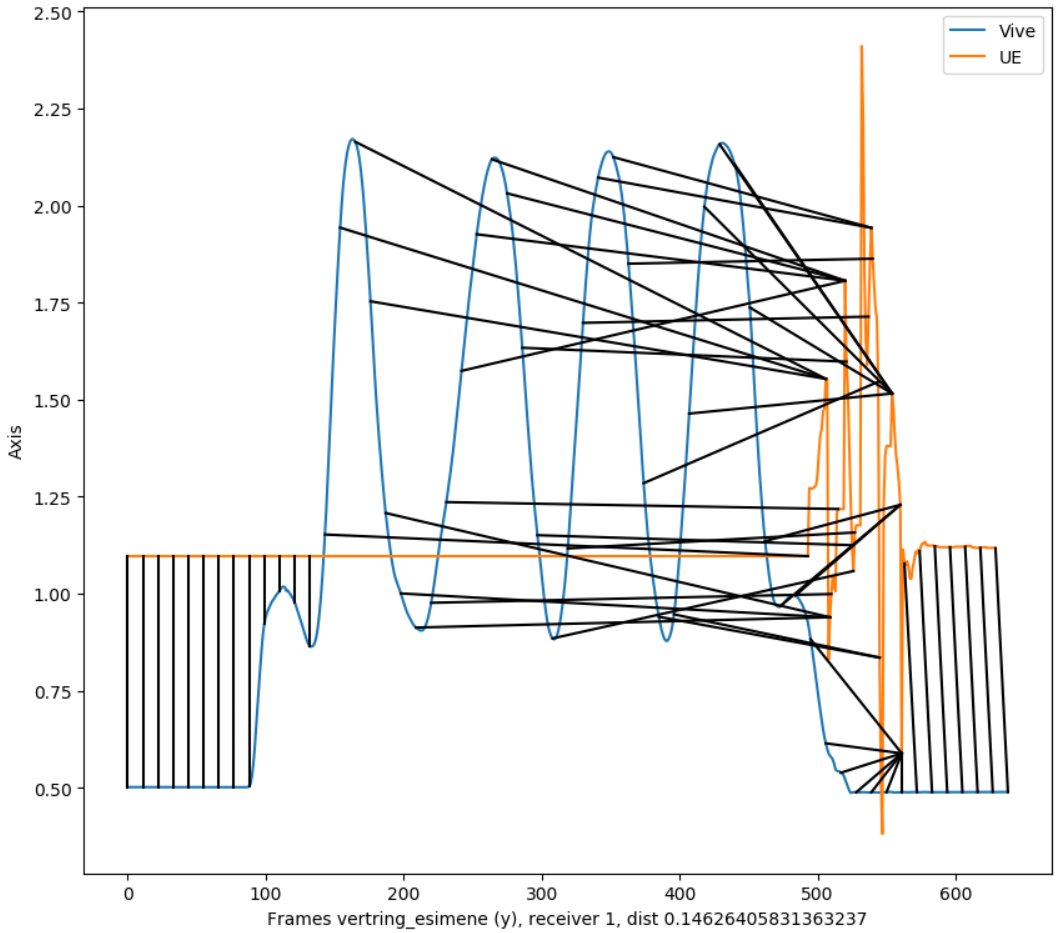


Joonis 16: Halva näite x-telg

implementeeritud trianguleerimisfunktsioone korduvate kulukate operatsioonide eemaldamise abil. Samuti tõsteti põhiprotsessist välja omaette protsessideks andmeid UEle edastavad eelnevalt peaprotsessis lõimedena jooksnud funktsioonid. Mainitud optimisatsioonitööde käigus sai katsetatud ODASe sisemise diskreetimissageduse ja seotud puhvrite kolmekordistamist. Kuna see kahekordistas helile reageerimise viited, otsustati jääda esialgse sätte juurde.

Viite määramine toimus preliminaarsete eksperimentidega samal alusel: nii arvutimonitor kui ka eksperimendiala püüti kaamera kaadrisse, filmiti kaadrisagedusega 240 kaadrit sekundis, salvestised töödeldi *ffmpeg*iga kirjutamaks sellele kaadrite järjekorranumbrid ja loeti kaadreid sündmusest ekraanil nähtava muudatuseni.

Liikumise täpsuse hindamiseks kasutati esialgsete mõõtmistega sama asendit:

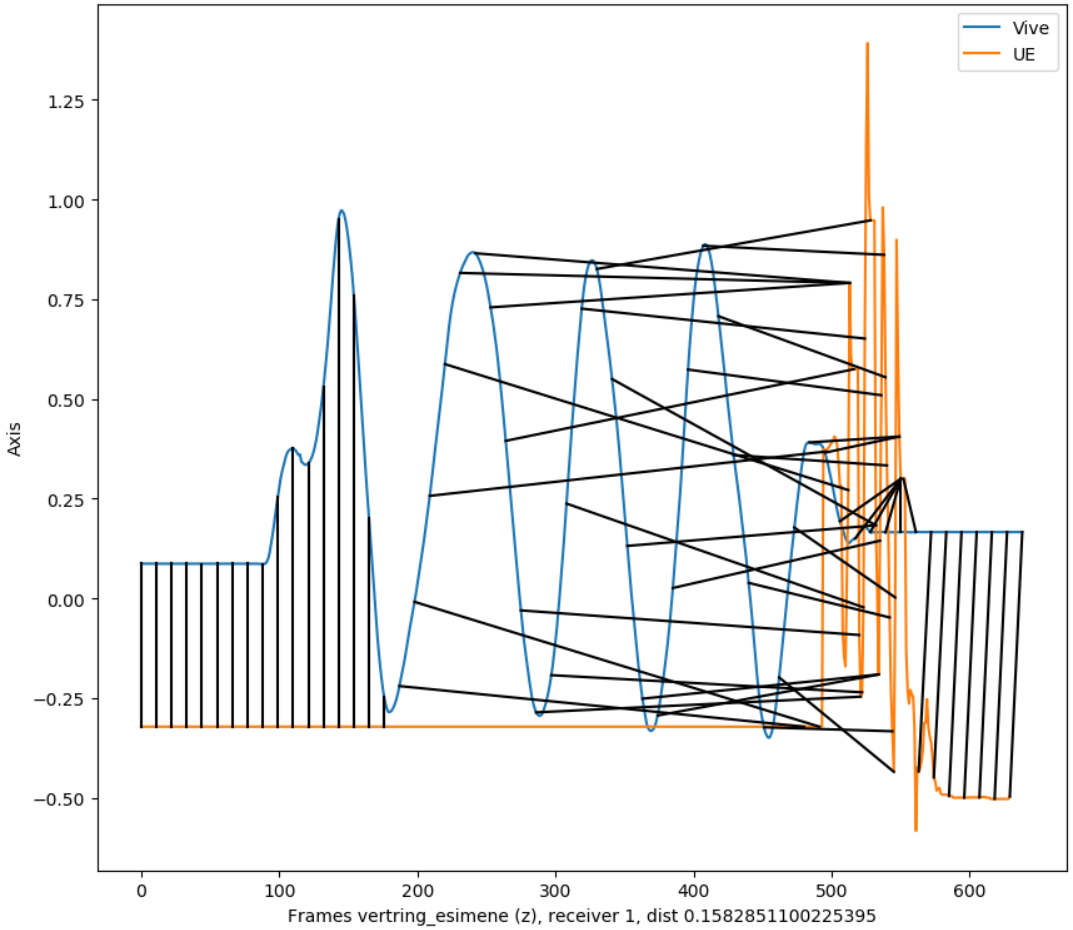


Joonis 17: Halva näite y-telg

mõõtmisala keskpunktist mikrofonimassiivide suhtes kaugemal, nägu mikrofonimassiivide vahelise nurga suunas. Mõõdeti kolme mustrit, iga mustrit liigutati viies grupis, iga grupp sisaldas viit liigutust. Kasutatud mustrid: vertikaalne ring, vertikaalne sirgjoon ja horisontaalne sikksakk. Mustrite lihtsustamine tulenes preliminaarsetest tulemustest.

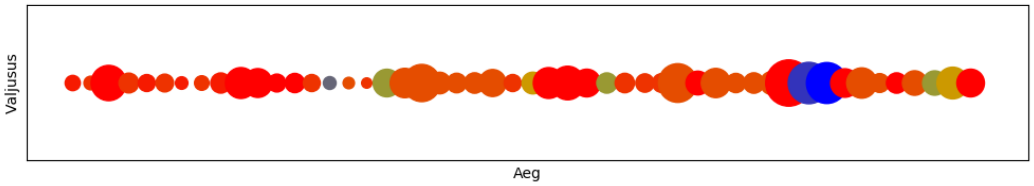
3.3.1 Helivaljusus ja sagedus

Heli valjususe ja sagedusest sõltuva värvi loogika jäi sarnaseks varasema tööga, kus kasutati värvikaardina Matlabi kaarti "jet" [22]. Käesolevas töös pöörati värvikaart teistpidi: madalamate sagedustega seostati punakamaid värve ja kõrgemate toonidega sinisemaid. Kahte ekstreemumit eraldab kollane. Lühike näide selle



Joonis 18: Halva näite z-telg

toimimisest on välja toodud joonisel 19. Selle visualisatsiooni tekitamiseks pidi küll logitud infot hõrendama, kuid siiski paistab jooniselt välja The White Stripes loo Seven Nation Army katkendi ikooniline trummirütm ja tugevam taldrikulöök.



Joonis 19: The White Stripes - Seven Nation Army

3.3.2 Heli ja liikumise viivitus

Helile reageerimise viite ja ja heliallika liikumisele reageerimise viite mõõtmisel suurendati katsete arv ja seega ka valim väärtuseni $n = 50$. Järsule valjule helile reageerimise viite mõõdetud väärtused on välja toodud lisas C, kokkuvõttev statistika on kujutatud tabelis 4.

Tabel 4: Plaksust kuva muutuseni mõõtmiste statistika

	Kaadreid	**Millisekundeid**
Keskmine	27.2	113
Mediaan	26.0	108
75%	29.0	121
Standardhälve	4.70	19.6
Keskmine absoluuthälve	2.81	11.7

Heliallika liikumisele reageerimise viite mõõdetud väärtused on välja toodud lisas C, kokkuvõttev statistika on kujutatud tabelis 5.

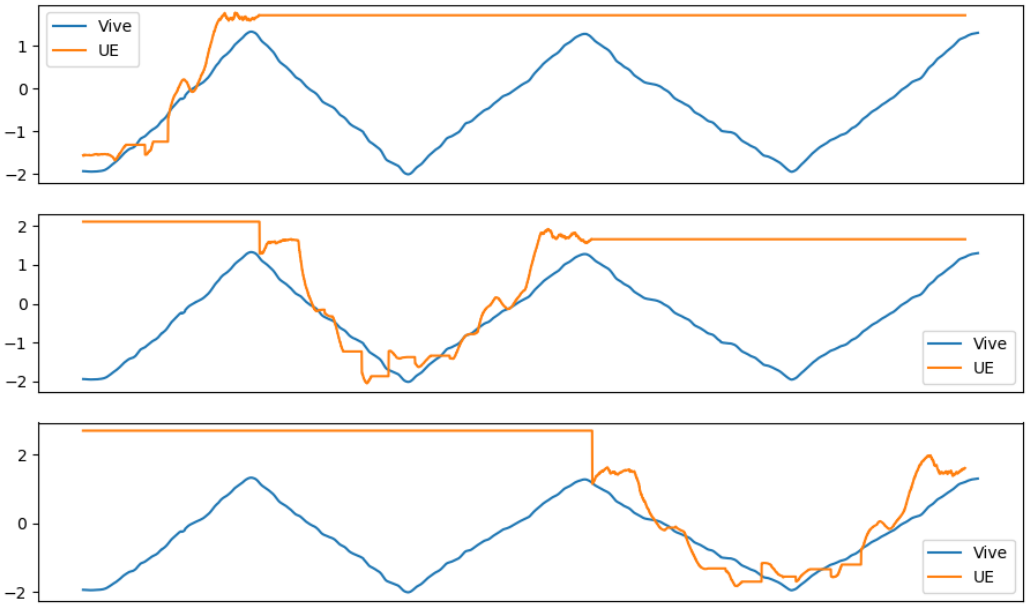
Tabel 5: Heliallika liikumisest kuva muutuseni mõõtmiste statistika

	Kaadreid	**Millisekundeid**
Keskmine	244.1	1018
Mediaan	249.0	1038
75%	298.0	1243
Standardhälve	71.11	296.5
Keskmine absoluuthälve	58.73	244.9

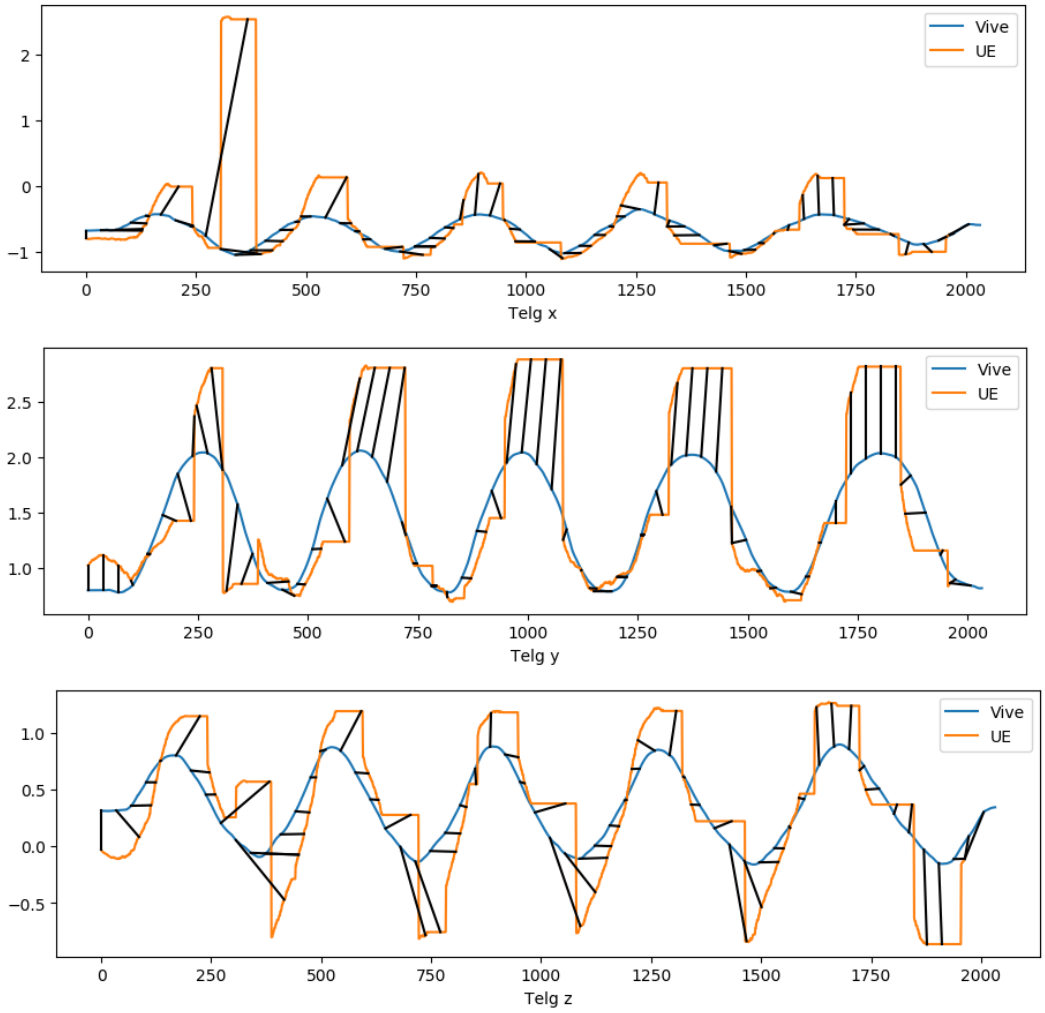
3.3.3 Liikumise jälgimise täpsus

Preliminaarse mõõtmisega võrreldes muudeti liikumise täpsuse hindamise ajal UEs logitavat info kogust. Kui esimeses voorus logiti vaid esimese objekti koordinaate, siis põhivooru ajal lülitati sisse kõigi nelja objekti vastuvõetud positsiooniinfo logimine.

Tänu optimisatsioonidele ja kõigi nelja potentsiaalse objekti asukoha logimisele saadi primaarsete mõõtmiste tulemused paremad. Tänu viimasele saab illustreerida esialgsetes eksperimentides nähtud objektidevahelise "üleandmise" efekti. Efekt on näidatud joonistel 20, kus on välja toodu ülevalt alla liikumine y-teljel objektidele 1,2 ja 4. Samas sai tänu optimisatsioonidele põhivoorus ka häid tulemusi, nagu on kujutatud joonisel 21. Tabelis 6 on välja toodud tulemused objektidest, mille puhul on kogu positsiooniinfo jäänud ühe objekti piiresse, vältimaks müra kajastamist. Kuna teame, et OpenVR koordinaadistikus on ühikuks meeter ja DTW distants leiti eukleidilise kaugusena, võiks leitud väärtusi vaadelda samuti kaugusena ideaaltrajektoorist.



Joonis 20: Jälgitava objekti üleandmise näide x-teljel



Joonis 21: Kogu jälgimine on jäänud ühe objekti piiresse

Tabel 6: DTW maksumused telje kaupa testides, kus ei toimunud objektivahetust

Muster	**Objekt**	**DTW X**	**DTW Y**	**DTW Z**
ulesalla_esimene	1	0,089	0,249	0,157
ulesalla_kolmas	3	0,098	0,154	0,154
ulesalla_neljas	2	0,078	0,095	0,212
ulesalla_teine	3	0,079	0,130	0,175
ulesalla_viies	2	0,075	0,089	0,214
vertring_esimene	1	0,078	0,147	0,092
vertring_kolmas	1	0,105	0,170	0,130
vertring_neljas	1	0,155	0,168	0,131
vertring_teine	1	0,097	0,161	0,113
vertring_viies	1	0,062	0,210	0,091

4 Analüüs ja arutelu

Varasemalt defineeritud üks olulisemaid näitajaid — viivitus pärismaailma sündmuse ja virtuaalmaailmas kuvatava vahel on segakott: vaadates kolmandat kvartiili (75% väärtustest valimis on väiksemad kui kvartiilväärtus), toimub heliallika liikumise (tabelid 2 ja 5) osas kolmanda kvartiili paranemine 196 ms võrra (13,6%), kuid heliallika helisündmusele reageerimise viivituse kolmas kvartiil (tabelid 3 ja 4) teeb läbi regressiooni 17 ms (16,3%). Vaadates vastavaid standardhälbeid, paistab silma huvitav tendents: heliallika liikumise standardhälve vähenes, samal ajal kui heli omadustele reageerimise standardhälve suurenes. Ka võrdluses eelneva prototüübiiteratsiooniga on tulemused segatud: üks näitaja on 4-5 korda parem ja teine 2-3 korda halvem kui eelmise iteratsiooni 500 ms [22]. Eeldati, et optimaalsem triangulatsiooniarvutus kombineerituna ebaoptimaalselt viisil protsessorit koormava andmete kombineerijaga on selle olukorra põhjus. Olukorra parendamiseks tuleks tulevikus mõelda välja parem viis positsiooni- ja vormiinfo kombineerimiseks ja edastamiseks. Kõige lihtsam viis tuvastada, kas tekitatud ebaoptimaalsus on süü allikas, on seda rakendust jooksutada protsessoriga, mille füüsiliste tuumade arv oleks >4.

Joonisel 21 näidatu, kus heliallikat liigutati vertikaalset ringjoont mööda, on mõõdetutest üks parimaid tulemusi. Mis seda joonist vaadates silma hakkab, on kummagi ODASe väljundi filter - kui ODAS ei saa päris täpset tulemust, siis kohalik Kalmani filter tekitab objekti trajektoori teatava triivimise hetkeni, kuni üks kahest instantsist objekti asukoha uuesti kätte saab, triivi peatab ja seejärel uude asukohta viskab. Eeldusel, et saame tabelis 6 vaadeldavaid DTW väärtusi võtta pärismaailma distantsides, võiks selle tabeli põhjal öelda, et asjaolud pole

kõige halvemad. Kahjuks ebaõnnestus trajektoori täpsuse mõõtmine esimeses mõõtmisvoorus ning seega pole võimalik mõõta ka progressiooni voorudevahelise optimeerimise tulemusena.

4.1 Potentsiaalsed arendussuunad

Kokkuvõtlikult võiks arendussuunad välja tuua järgmiste variantidena:

- ODASe *Phoenix*-haru
- Liitreaalsus koos HMD külge kinnitatud uue mikrofonimassiiviga
- Liitreaalsus koos käesoleva prototüübi heliseadmetega
- Mikrofonimasiivide suuna ja positsiooni konfiguratsiooni automaatne genereerimine
- ODASe muutmine eemaldamaks neist väljundi filter, trianguleerijas filtri implementeerimine

Käesoleva töö alustamise hetkel eksisteeris ODASest kaks haru: stabiilne haru ja Föönіksi-nimeline rohkem eksperimentaalne haru. Prototüübi itereerimise ajal on just mainitud eksperimentaalne haru saanud olulisi täiendusi võrreldes stabiilse peaharuga. Aja- ja arhitektuurilistel kaalutlustel ei hakanud käesolev töö autor kasutatavat haru vahetama, kuid üks viis proovida selle prototüübikandidaadi võimekust parandada oleks katsed just mainitud eksperimentaalse haruga.

4.1.1 Suund liitreaalsusesse

Rõhk liitreaalsusele tuleneb faktist, et ajast, kui käesolevas töös käsitletav prototüüp mängumootorina Unreal Engine-t kasutama hakkas, on möödas vähemalt 5 kasutatava mootori versiooni ja üks neist — UE versioon 4.20 lisas toe liitreaalsusele [32]. Liitreaalsuse kasutamine suurendaks oluliselt prototüübi portatiivsust, kaotades vajaduse seda uues kohas üles seades luua ka vastava uue ruumi kolmemõõtmeline mudel. Piltlikult öeldes muudetaks süsteem varasemalt

välja toodud BurnAR [16] sarnaseks: kas läbi HMD kaamerate püütud videole või Microsoft Hololensi sarnase seadme ekraanile asetame Unreal Engines loodud ülekatte helimarkeritest.

Suurima lihtsustuse tooks mikrofonimassiivi muutmine hetkel kasutatavast statsionaarsest koonusekujulisest massiivist tsirkulaarseks HMD peale kinnitatavaks. Sellised massiivid on juba hetkel kättesaadavad valmistootena [33] ja ka vabavaraliste skeemidena [34] ise ehitamiseks. Kasutades sellise paigutusega mikrofonimassiivi, kaob sisuliselt vajadus omada sügavusinfot: ODASe väljund on heliallika suund massiivi ümbritseva kera pinnal, mis tähendab, et me teame heliallika elevatsiooni ja asimuuti mikrofonimassiivi asendi (seega ka HMD asendi) suhtes. See võimaldab kuvada heliallika marker kasutaja perspektiivist õiges suunas ning võimaldab ka ruumis liikudes heliallikat erinevates suundadest "näha". Kuna allika markeri suurus sõltub mikrofonide poolt püütud energiast, toimub viimase suhteline suurenemine ja vähenemine loomulikul teel, sõltuvalt distansist. Kirjeldatud muutus on tõenäoliselt lihtsaim viis suurendada prototüübi praktilist väärtust: kaob probleem trianguleerimisvigadega, kaob vajadus ringi liigutada kaht suurt mikrofonimassiivi ja kaob vajadus konstantselt uusi UE ruume luua.

4.1.2 Mikrofonimassiivide seadistuse lihtsustamine

Käesoleva prototüübikandidaadi üks suurimaid probleeme on portatiivsuse puudumine. Selle laborisse üles seadmine oli ühe tööpäeva ülesanne. ODAS on väga tundlik mikrofonide konfigureeritud suuna ja reaalse suuna erinevustele; olukorda ei lihtsusta kasutatava mikrofonimassiiviraamiga kasutatavate mikrofonihoidikute liikuvus (pööre ja kalle). Esimene lihtsaim samm oleks kõikide hoidikute suuna ja kalde fikseerimine optimaalses suunas läbi spetsiifiliste hoidikelementide disainimise ja printimise.

Järgmine aeganõudev ja mõõtmistulemuste inimlikku viga suurendav aspekt on mikrofonimassiivide omavahelise paiknemise konfigureerimine. Vead selles konfiguratsioonis halvendasid oluliselt trianguleerimise täpsust. Väiksemas ruumis prototüüpi arendades saavutati lõpuks rutiinselt massiividest lähtuvate

kiirevaheline distants < 10 sentimeetrit, pärast prototüübi ülesseadmist suuremas laboris oli tüüpiline kiirtevaheline distants < 25 sentimeetrit. Vea kasv tulenes siinkohal sellest, et massiivide omavahelist nurka ja distantsi ning mikrofonide suundi ning nurkasid on raske pärast prototüübi osadeks võtmist uuesti paika saada. Üks viis massiivide üldist positsiooni seadistada on nende külge teadaoleva suunaga Vive Trackeri lisamine. Kasutades kaht Vive Trackerit, kinnitatuna sirgjoones ühe massiivi külge, saaks empiiriliselt kätte mikrofonimassiivide suunad ja paigutuse teineteise suhtes. See aitaks oluliselt parandada trianguleerimise täpsust läbi osalise inimliku vea kõrvaldamise.

4.1.3 Immersiooni suurendamine

Lähtudes Rossi poolt [19] välja toodust, teeb käesoleva töö autor ettepaneku lisada sõltumata kasutatavast tehnoloogiast UEs joonistatavale maailmale atmosfäär hõreda heljuva tolmu näol. Eeldusel, et heliallika asukoha markeriks jääb kera, peaks lisama markeri ajalises mõttes kohalikust maksimumist lähtuva moonutuslaine selles virtuaalses atmosfääris. Mainitud laine, mis võiks paista heliallikast eralduva moonutuslainena, võiks suurendada tunnet, et simulatsiooniprototüübi kasutaja "näeb heli". Toetudes Biocca jt. [15] ja Outrami jt. [18] poolt varasemalt esitatud töödele, teeb autor ettepaneku lisada simulatsiooniprototüübile uue mõõtme "heli katsumise" näol. Näiteks võiks kasutada vibratsiooni abil puudutust simuleerivaid kindaid, mis vibreeriksid hetkel kui kasutaja käsi läbib markerist lähtuva moonutuslaine. Sellised kindad on saadaval valmistootena [35]. Alternatiivina saaks kasutada vibreerivaid pulte.

5 Kokkuvõte

Käesoleva töö peamine eesmärk oli implementeerida uus sünesteesiasimulaatori prototüübikandidaat. Arendustöö toimus piiranguga kasutada olemasolevat riistvara ja keeles Python. Kood üritati hoida lihtne ja seega tulevikus muutuvate vajaduste tarvis hõlpsasti modifitseeritav. Samuti oli eesmärk asendada Matlabis implementeeritud positsioneerimine sama teha võimaldava valmisrakendusega nimega ODAS. Käesoleva töö tulemuseks on mõlema ülesandepüstituse täitmine kuid ka ODASe ühe suurima nõrkuse välja toomine: mikrofonimassiivide äärealadel, kus suuna leidmine muutub raskeks, kipub samaaegselt nelja objekti jälgiv ODAS asendama ühe objekti teisega. Teine objektimarker võib jätkata liikumist heliallika liikumise trajektooril samal ajal kui esimese objekti trajektoori logis tekib katkestus. Mõõtmistulemuste hulgas oli ka katseid, kus selline "üleandmine" toimus lühikese 30-sekundise sessiooni jooksul kolm korda, mis tähendas, et ühe trajektoorilogi raames muutus DTW näit mõttetuks. VR ühe suurima olulise näitaja latentsuse osas on tulemused kahetised: helisündmuse, st. liikumise või toonimuutuse latsentsusaegades saavutati võrreldes varasema prototüübiga vastavalt kerge regressioon ja pea 5-kordne progressioon. Õnnetuseks on isegi helisündmuse kuvalatentsus 110 ms veel kaugel soovituslikust maksimumist 15 ms. Uue prototüübikandidaadi arendamise ja selle väljundi mõõtmise protsess pakkus võimaluse genereerida ka arvestatava hulga praktilisemaid ja vähempraktilisemaid suundi, kuhu prototüüp viia.

Conclusion

The main objective of the thesis at hand was to implement a new candidate for the syesthesia simulator. The development work was limited to the Python language and hardware that already existed in the lab. The developed code was kept as simple as possible for it to be easy to understand and modifiable for the future. One of the set targets was to implement the candidate using a third-party open source tracking application named ODAS instead of Matlab code. The set objectives were met but also the main weakness of ODAS was discovered: on the edges of microphone arrays where finding the sound's direction of arrival becomes harder, ODAS seems to substitute one tracked object with another. While the new object keeps moving along the sound source's path, the old one will see empty spots in it's tracking log. There were short tracking sessions where this kind of substitution was observed to take place three times in 30 seconds which made calculating DTW costs on that specific path pointless. The results of one of the most important metrics of VR - the latency between real-world action and VR-reaction are a mixed bag: while tracking the movement of the sound source there was a regression in comparison with the previous iteration of the prototype but there also was almost a 5-times progression while considering the latency between a sound event and it being displayed in VR. Nevertheless - even the achieved 110 ms is far from VR's recommended maximum of 15 ms. The process of developing the new prototype and experimenting it offered plenty of ideas on how to make it more practical, all of which were proposed.

Kasutatud kirjandus

[1] *Emerging markets for virtual reality*. IGI Consulting, 1992.

[2] Paul James. *The HTC Vive Costs $799 and Ships April 1st – Road to VR*. Juuli 2016. URL: https://www.roadtovr.com/htc-vive-price-799-and-release-date-april-1st/ (vaadatud 04. 05. 2019).

[3] Hunter G. Hoffman et al. "Effectiveness of Virtual Reality–Based Pain Control With Multiple Treatments". *The Clinical journal of pain* 17 (oktoober 2001), l. 229–35. DOI: 10.1097/00002508-200109000-00007. (Vaadatud 27. 04. 2019).

[4] Dean Takahashi. *Porn and games are the biggest drivers of VR revenues*. August 2018. URL: https://venturebeat.com/2018/07/23/porn-and-games-are-the-biggest-drivers-of-vr-revenues/ (vaadatud 04. 05. 2019).

[5] *VR–1 – Varjo.com*. URL: https://varjo.com/vr-1/ (vaadatud 04. 05. 2019).

[6] *What is Synaesthesia?* http://www.cogsci.mq.edu.au/research/projects/synaesthesia/. (Vaadatud 24. 07. 2018).

[7] *Demographic aspects of synesthesia*. http://www.daysyn.com/Types-of-Syn.html. (Vaadatud 14. 04. 2018).

[8] *Thesaurus: Virtual Reality*. http://www.dictionary.com/browse/virtual-reality. (Vaadatud 24. 07. 2018).

[9] Mark R. McMinn. "2.31 - Technology in Practice". Teoses: *Comprehensive Clinical Psychology*. Toim. Alan S. Bellack ja Michel Hersen. Oxford: Pergamon, 1998, l. 363–375. ISBN: 978-0-08-042707-2. DOI:

https://doi.org/10.1016/B0080-4270(73)00056-0. URL: http://www.sciencedirect.com/science/article/pii/B0080427073000560.

[10] Charlene Jennett et al. "Measuring and defining the experience of immersion in games". *International Journal of Human-Computer Studies* 66.9 (2008), l. 641–661. ISSN: 1071-5819. DOI: https://doi.org/10.1016/j.ijhcs.2008.04.004. URL: http://www.sciencedirect.com/science/article/pii/S1071581908000499.

[11] C. Anthes et al. "State of the art of virtual reality technology". Teoses: *2016 IEEE Aerospace Conference*. Märts 2016, l. 1–19. DOI: 10.1109/AERO.2016.7500674.

[12] Thomas Waltemate et al. "The Impact of Latency on Perceptual Judgments and Motor Performance in Closed-loop Interaction in Virtual Reality". Teoses: *Proceedings of the 22Nd ACM Conference on Virtual Reality Software and Technology*. VRST '16. Munich, Germany: ACM, 2016, l. 27–35. ISBN: 978-1-4503-4491-3. DOI: 10.1145/2993369.2993381. URL: http://doi.acm.org/10.1145/2993369.2993381.

[13] Charu C. Aggarwal. *Data mining: the textbook*. Springer, 2016.

[14] Fuxing Huang, Jianping Huang ja Xiaoang Wan. "Influence of virtual color on taste: Multisensory integration between virtual and real worlds". *Computers in Human Behavior* 95 (jaanuar 2019). DOI: 10.1016/j.chb.2019.01.027.

[15] Frank Biocca et al. "Visual cues and virtual touch: Role of visual stimuli and intersensory integration in cross-modal haptic illusions and the sense of presence" (jaanuar 2002).

[16] P. Weir et al. "Burnar: Involuntary heat sensations in augmented reality". Teoses: *2013 IEEE Virtual Reality (VR)*. Märts 2013, l. 43–46. DOI: 10.1109/VR.2013.6549357.

[17] B. I. Outram. "Synesthesia audio-visual interactive-sound and music visualization in virtual reality with orbital observation and navigation". Teoses: *2016 IEEE International Workshop on Mixed Reality Art (MRA)*. Märts 2016, l. 7–8. DOI: 10.1109/MIXRA.2016.7858997.

[18] B. Outram et al. “Crystal Vibes feat. Ott: A psychedelic musical virtual reality experience utilising the full-body vibrotactile haptic synesthesia suit”. Teoses: *2017 23rd International Conference on Virtual System Multimedia (VSMM)*. Oktoober 2017, l. 1–4. DOI: 10.1109/VSMM.2017.8346269.

[19] Miriam Ross. “Virtual Reality’s New Synesthetic Possibilities”. *Television & New Media* 0.0 (0), l. 1527476418805240. DOI: 10.1177/1527476418805240. eprint: https://doi.org/10.1177/1527476418805240. URL: https://doi.org/10.1177/1527476418805240.

[20] A. Tepljakov et al. “Sound localization and processing for inducing synesthetic experiences in Virtual Reality”. Teoses: *2016 15th Biennial Baltic Electronics Conference (BEC)*. Oktoober 2016, l. 159–162. DOI: 10.1109/BEC.2016.7743753.

[21] Ahmet Kose, Aleksei Tepljakov ja Sergei Astapov. “Real-time localization and visualization of a sound source for virtual reality applications”. *2017 25th International Conference on Software, Telecommunications and Computer Networks (SoftCOM)* (2017), l. 1–6.

[22] Ahmet Kose et al. “Towards a Synesthesia Laboratory: Real-time Localization and Visualization of a Sound Source for Virtual Reality Applications”. *Journal of Communications Software and Systems* 14.1 (2018), l. 112–120. ISSN: 1846-6079. DOI: 10.24138/jcomss.v14i1.410. URL: https://jcomss.fesb.unist.hr/index.php/jcomss/article/view/410.

[23] *C-2—Behringer—P0263*. URL: https://www.musictribe.com/Categories/Behringer/Microphones/Condenser/C-2/p/P0263 (vaadatud 20. 04. 2019).

[24] *Scarlett 18i20 — Focusrite*. URL: https://focusrite.com/usb-audio-interface/scarlett/scarlett-18i20.

[25] *UDPCommunication for UE4*. https://github.com/is-centre/udp-ue4-plugin-win64. (Vaadatud 02. 02. 2019).

[26] *VIVE™ — VIVE Virtual Reality System*. URL: https://www.vive.com/us/product/vive-virtual-reality-system/ (vaadatud 04. 05. 2019).

[27] Oliver Kreylos. *Lighthouse tracking examined*. Mai 2016. URL: http://doc-ok.org/?p=1478.

[28] *ODAS: Open embeddeD Audition System*. https://github.com/introlab/odas. (Vaadatud 24. 07. 2018).

[29] François Grondin ja François Michaud. "Lightweight and Optimized Sound Source Localization and Tracking Methods for Open and Closed Microphone Array Configurations". *Robotics and Autonomous Systems* 113 (jaanuar 2018). DOI: 10.1016/j.robot.2019.01.002.

[30] Rahel Rjadnev-Meristo. "Fine motor analysis for school success and habits modelling of schoolchildren". Magistritöö. 2018.

[31] Nghia Ho. *Finding optimal rotation and translation between corresponding 3D points*. URL: http://nghiaho.com/?page_id=671 (vaadatud 04. 05. 2019).

[32] Ben Lang. *Unreal Engine 4.20 Ships With Native Mixed Reality Compositing for VR, New AR/VR SDKs*. Juuli 2018. URL: https://www.roadtovr.com/unreal-engine-4-20-ships-with-native-mixed-reality-compositing-for-vr-new-ar-vr-sdks/ (vaadatud 04. 05. 2019).

[33] *xCORE Array Microphone*. URL: https://www.xmos.com/developer/products/voice-user-interfaces/xcore-microphone-array-platform (vaadatud 04. 05. 2019).

[34] *8 Sounds USB*. URL: https://sourceforge.net/projects/eightsoundsusb/files/Schematics/ (vaadatud 04. 05. 2019).

[35] *Haptic Gloves*. URL: https://manus-vr.com/haptics/ (vaadatud 04. 05. 2019).

[36] Triad Semiconductor TriadSemi. *TriadSemi/triad_openvr*. Märts 2019. URL: https://github.com/TriadSemi/triad%5C_openvr (vaadatud 27. 04. 2019).

A ALSA konfiguratsioon

```
pcm.snd_card {
    type hw
    card 2
}
pcm.dsnooper {
    type dsnoop
    ipc_key 2048
    ipc_perm 0666
    slave.pcm "snd_card"
    slave
    {
        rate 96000
        period_time 0
        period_size 1024
        buffer_size 4096
        channels 10
    }
    bindings {
        0 0
        1 1
        2 2
        3 3
        4 4
        5 5
        6 6
        7 7
        8 8
        9 9
    }
}
```

B Vive Trackeri positsiooniinfo salvestamine

```
1  import triad_openvr
2  import time
3  import datetime
4  import sys
5
6  v = triad_openvr.triad_openvr()
7  v.print_discovered_objects()
8  outfile = open('output.txt','w')
9
10 if len(sys.argv) == 1:
11     interval = 1/250
12 elif len(sys.argv) == 2:
13     interval = 1/float(sys.argv[1])
14 else:
15     print("Invalid number of arguments")
16     interval = False
17
18 if interval:
19     while(True):
20         start = time.time()
21         txt = ""
22         for each in v.devices["tracker_0"].get_pose_euler():
23             txt += "%.4f" % each
24             txt += " "
25         txt = str(datetime.datetime.now().time()) + " " + txt + "\n"
26         outfile.write(txt)
27         print("\r" + txt, end="")
28         sleep_time = interval-(time.time()-start)
29         if sleep_time>0:
```

```
30                  time.sleep(sleep_time)
```

Kood pärineb allika [36] näitest trackertest.py. Käesoleva töö autori lisatud read on 19, 24, 41 ja 42. Lisa mõte on illustreerida Vive Trackerite juba eksisteerivate teekide abil kasutamise lihtsust positsiooniinfo kätte saamisel.

C Helile reageerimise viited

Tabelis 7 on välja toodud kaadrite loendamise tulemused järsu heli korral. Märgitud on sündmuse kaader, reaktsiooni kaader, nende vahe ja nende vahe korrutatuna kaadritevahelise aja 4,17 millisekundiga. Mõõtmised teostati kahe videoklipiga, sellest tulenevalt on teises tabelis kaadrite nummerdus alanud uuesti nullist.

Tabelis 8 on välja toodud kaadrite loendamise tulemused heliallika liigutamisel. Märgitud on sündmuse kaader, reaktsiooni kaader, nende vahe ja nende vahe korrutatuna kaadritevahelise aja 4,17 millisekundiga.

Tabel 7: Plaksu helile reageerimise andmestik

Heli	**Reaktsioon**	**Kaadreid**	**Aeg**	**Heli**	**Reaktsioon**	**Kaadreid**	**Aeg**
# kaader	**# kaader**	**#**	**ms**	**# kaader**	**# kaader**	**#**	**ms**
557	588	31	129	281	307	26	108
678	702	24	100	558	583	25	104
814	842	28	117	838	864	26	108
977	1007	30	125	1105	1127	22	92
1156	1181	25	104	1380	1408	28	117
1327	1351	24	100	1665	1692	27	113
1508	1538	30	125	1943	1967	24	100
1681	1707	26	108	2213	2239	26	108
1853	1882	29	121	2483	2506	23	96
2024	2050	26	108	2738	2762	24	100
2205	2235	30	125	2999	3023	24	100
2388	2418	30	125	3262	3291	29	121
2580	2610	30	125	3536	3566	30	125
2769	2799	30	125	3801	3827	26	108
2956	2978	22	92	4070	4095	25	104
3146	3174	28	117	4335	4359	24	100
3348	3378	30	125	4606	4631	25	104
3534	3588	54	225	4883	4907	24	100
3729	3754	25	104	5153	5174	21	88
3912	3938	26	108	5429	5455	26	108
4114	4142	28	117	5692	5722	30	125
4312	4338	26	108	5965	5996	31	129
4497	4526	29	121	6237	6262	25	104
4686	4711	25	104	6533	6560	27	113
4891	4918	27	113	6809	6839	30	125

Tabel 8: Heliallika liikumisele reageerimise viite andmestik

Liigutus # kaader	**Reaktsioon # kaader**	**Kaadreid #**	**Aeg ms**	**Liigutus # kaader**	**Reaktsioon # kaader**	**Kaadreid #**	**Aeg ms**
460	610	150	626	14647	14881	234	976
880	1205	325	1355	15270	15583	313	1305
1396	1655	259	1080	15865	16213	348	1451
1922	2255	333	1389	16436	16717	281	1172
2467	2679	212	884	17078	17284	206	859
2968	3140	172	717	17667	17936	269	1122
3445	3718	273	1138	18289	18510	221	922
3932	4149	217	905	18822	19145	323	1347
4495	4795	300	1251	19429	19617	188	784
5169	5288	119	496	19969	20136	167	696
5766	6072	306	1276	20560	20732	172	717
6489	6629	140	584	21014	21170	156	651
7167	7408	241	1005	21517	21681	164	684
7794	7978	184	767	22058	22310	252	1051
8352	8501	149	621	22529	22948	419	1747
8975	9123	148	617	23196	23478	282	1176
9616	9747	131	546	23808	24124	316	1318
10137	10301	164	684	24382	24680	298	1243
10685	10937	252	1051	24943	25225	282	1176
11225	11416	191	796	25427	25640	213	888
11746	11995	249	1038	25930	26199	269	1122
12302	12538	236	984	26504	26762	258	1076
12880	13167	287	1197	27079	27445	366	1526
13455	13702	247	1030	27660	28026	366	1526
14048	14360	312	1301	28270	28544	274	1143