Projektitöö

Object tracking and motion analysis

Mihkel Heidelberg
Karl Tarbe

Tartu 2011

Sisukord

1 Probleemi kirjeldus

1.1 Robot

Meie kasutusel olev robot nimega Mall on ehitatud mängima jalgpalli golfipallidega 2m * 3m väljakul vastavalt Robotex 2010 reeglitele. Korraga on väljakul kaks võistlevat robotit ja kuni 9 golfipalli. Mall saab infot maailma kohta läbi kaamera, mis on suunatud hüperboolpeeglile. Kaamerast saadud kaadrite jaoks on implementeeritud pilditöötlus, mis leiab sealt pallid ja väravad, tehes seda iga kaadri kohta iseseisvalt. Lisaks on anduriteks veel magnetandurid, mis loevad rataste pöördenurkasid ning sellest saab täpse suhtelise positsioonimuudu ja kiiruse. Selleks, et saada täpset positsiooni ja kiirust väljaku mõistes absoluutsetes koordinaatides tuleb magnetanduritelt saadud suhtelist positsiooni pilditöötluselt saadud absoluutse positsiooni abil parandada.

Nõuded filtreerimisele

AI jaoks on vaja, et pilditöötluselt saadud info väljakul asuvate objektide kohta oleks kvaliteetne.

- Pallide asukoht peab olema püsiv ja täpne. Sellisel juhul saab võimalikult kiiresti neid püüda.
- Pallid ei tohi liiga tihti ilmuda ja kaduda.
- Ei tohi näha viirastusi. Ei kuskilt ilmuvad olematud pallid häirivad kindlasti robotil parima trajektoori valimist.
- Peab eristama liikuvat ja seisvat palli. Ründes ei ole mõtet järgneda eest ära veerevale pallile ja samas kaitses peab ette sõitma värava poole liikuvale pallile.

Pilditöötlusest tulev info neid tingimusi ei rahulda, sest pilditöötlust tehakse üksiku kaadri põhjal, millelt saab teada ainult pallide asukohad, mitte kiirused. Ühe kaadri info ei ole täpne, sest halb on juba hüperboolpeeglist saadav väikese resolutsiooniline pilt, millel kaugemad pallid on diameetriga suurusjärgus 5-10 pikslit, mürased ja ei asu fookuses, sest peegeldudes hüperbooli pinnalt on kujutis radiaalsuunas ja tangentsiaalsuunas erinevatel kaugustel. Segavad ka muutuvad valgusolud, läiked ning ka pilditöötlus pole päris täiuslik. Kogu selle tulemusena on andmete seas näha viirastusi ja alati ei ole näha kõiki palle. Pildil olev müra ja suuremal määral ka liikumisel tekkiv vibratsioon muudab pallide asukohtade info müraseks, sest juba ühe pikseli suurune muudatus võib tähendada mitme sentimeetrist erinevust väljakul. Lisaks sellele leitakse pallide asukohad roboti suhtes ning pallide asukohtade saamiseks väljaku suhtes tuleb eelnevalt ka roboti positsioon väljakul täpselt määrata.

1.2 Raskused filtreerimisel

Põhiprobleemid olidki selles, et enamus filtreid eeldab konkreetset vastavust mõõtmistulemuste ja hüpoteeside vahel, ehk on mõeldud enamasti ühe objekti jälgimiseks. Seega pidi ise nuputama, kuidas siduda ja mida teha üleliiksete mõõtmiste või objektidega. Samuti on meil väga töömahukas kõiki mudeli parameetreid korrektselt määrata. Seetõttu pidime palju improviseerima.

2 Lahendusmeetodid

Meie probleem on väga sarnane radariga, kus samuti ei ole algsignaal puhas ning on olemas vajadus objektide jälgimiseks. Radari jaoks on wikipedia andmetel mitmeid erinevaid lahendusi.

2.1 Alfa-Beta filter

Lihtsustadud filter, Kalmani filtri kauge sugulane, milles süsteemi mudel koosneb kahest muutujast, millest esimene on teise integraal. Meie jaoks oleks need palli või roboti mingi suunaline koordinaat ja kiirus. Filter eeldab, et süsteemi vead on fikseeritud dispersiooniga Gaussi jaotused. Eraldi peab lahendama mõõtmistulemuste ja jälgitavate hüpoteeside vastavusse seadmise.

2.2 Kalmani filter

Matemaatiliselt kõvasti korrektsem filter eelmisest. Süsteemi mudel on kirjeldatud rohkemate muutujatega. Eeldusteks on see, et süsteemi oleku muutust ja mõõtmist kirjeldavad lineaarsed funktsioonid ja süsteemi müra ning mõõtmistulemuste määramatud on Gaussi jaotusega. Korrektselt rakendatuna annab süsteemi peidetud muutujatele ka määramatuse. Matemaatilisest seisukohast annab see lineaarsete süsteemide korral süsteemi olekumuutujatele optimaalse hinnangu. Samuti peab eraldi lahendama mõõtmistulemuste ja jälgitavate hüpoteeside vastavusse seadmise.

2.3 Multiple hypothesis tracker (MHT)

Siin kaalutakse kõiki võimalikke liikumisi ja nii tekitatakse iga uuendusega suur hulk võimalikke liikumisharusid juurde. Sobib hästi, kui objekti liikumist on väga raske ennustada. Filtri ülesandeks on eemaldada väga ebatõenäolised harud.

2.4 Interacting multiple model (IMM)

Siin kasutatakse mitut erinevate parameetritega Kalmani filtrit, ning nende tulemused kombineeritakse mudeli tulemuseks. Seda võib kasutada näiteks JPDAF-i või MHT-i ennustamise faasis.

2.5 Extended Kalman filter (EKF)

Filter kujutab endast Kalmani filtri laiendust nendele juhtudele kui mõõtmismudel või süsteemi olekumuutuse mudel ei ole lineaarne, nagu on vaja tavalise Kalmani filtri jaoks. EKF üritab lineariseerida neid mudeleid kasutades selleks Taylor-i jada esimest järku liikmeid ja siis läheneb probleemile samamoodi nagu seda teeb tavaline Kalmani filter.

2.6 Particle filter

Filter, mis kasutab tõenäosuse tihedusfunktsiooni esitamiseks suurt hulka osakesi. Eeliseks on olenevalt süsteemist suhteliselt lihtne implementatsioon. Ei eelda, et süsteemi on lineaarne või et mõõtmisvead või süsteemi müra on Gaussi jaotusega. Puudusena sõltub tõenäosustiheduse

esituse täpsus ja sellega ka filtri efektiivsus simuleeritud osakeste arvust. Sõltuvalt süsteemist võib sellest tingitud arvututsnõudlus üle jõu käia. Samuti peab eraldi lahendama mõõtmistulemuste ja jälgitavate hüpoteeside vastavusse seadmise.

2.7 Joint Probabilistic Data Association Filter (JPDAF)

Filter, mis konkreetses näites lisaks Particle filtri peale ehitatud hüpoteeside jälgimisele proovib matemaatiliselt korrektselt seada neid mõõtmistele vastavusse, arvestades erineva kaaluga kõiki või kõiki mõistlikke sidumise võimalusi. Saab hakkama juhtudega kus hüpoteese ei ole võrdselt mõõtmistulemustega. Siiski hüpoteeride tekkimise ja kadumise peab filtri väliselt implementeerima. See oli ilmselt meie jaoks kõige korretsem lahendus, kuid kuna see oli üsna keeruline ja aeg projekti valmimiseks oli piiratud, siis sai seda põhjalikult uuritud, kuid implementatsiooniga valmis ei jõutud.

3 Eksperimendid

3.1 Andmete saamine

Andmete tekitamiseks kirjutasime Malli tarkvara salvestama magnetandurite abil arvutatud positsiooni, pildilt leitud pallide ja väravate asukohti ja ka pilti ennast. Et viivis magnetandurite ja pildi vahel oleks võimalikult ühtlane ja väike salvestasime video toores YUYV formaadis 30 kaadrit sekundis otse RAM-i. Niimoodi saime salvestada korraga kuni umbes 1000 kaadrit ~ 35 sekundit materjali, mis hiljem kirjutati faili, mida sai pärast ette mängida või uuesti pilditöötluse sisendiks saata.

Saadud andmeid kui ka filtreeritud tulemit oli vaja kuidagi kontrollida. Selle lahendasime lõpuks kinnitades väljaku kohale lakke sülearvuti, mis salvestas statsionaarsest asendist pealtvaates videot. Selle jaoks tuli pilditöötlus ümber kirjutada, kuid sealt videolt leitud pallide asukohad on palju täpsemad ja stabiilsemad kui robotilt vaadatuna. Laest saadud pilti oleks olnud võimalik ka, erinevalt meie hüperboolpeeglist saadud vaatest, täiesti standardsete võtetega pealtvaate perspektiivi viia. Selleks on näiteks opencv-s funktsioon getPerspectiveTransform, millele saab anda ette näiteks väljaku nurkade koordinaadid. Kahjuks seda realiseerida ei jõudnud ning me viisime pildi nii pilditöötluse kui ka tagasi esituse jaoks lihtsamate teisendustega “käsitsi timmides” perspektiivi.

3.2 Katse stsenaariumid

3.2.1 Paigalseisev robot

Selle katse käigus robot seisis terve katse aja paigal, ning palle panime me ise liikuma. Katse eesmärgiks oli eemaldada katse andmete seast roboti odomeetriast tulenevad roboti positsiooni vead, mis omakorda mõjutavad pallide koordinaate, sest need leitakse roboti asukoha suhtes.

3.2.2 Liikuv robot

Selle katse käigus sõitis robot ettemääratud trajektoori, milleks oli kummulikeeratud kaheksa ehk lõpmatuse märk. Pallid seisid paigal või liikusid ainult juhul, kui nad roboti trajektoorile jäid. Katse on oma olemuselt kõige sarnasem Robotexi väljakul aset leidvale, sest liiguvad ainult üksikud pallid, millele robot ise otsa sõidab, ning enamik palle on paigal.

3.2.3 Liikuv robot ja liikuvad pallid

Selle katse käigus sõitis robot samasugust trajektoori nagu eelmises katses, kuid seekord veeretasime palle väljakul ka käsitsi. Testi eesmärgiks oli näha filtrite tööd väga dünaamilise keskkonna puhul.

3.3 Andmete taasesitamine

Võimalus salvestatud andmete korrektsust kontrollida on väga oluline abivahend tarkvara silumisel. Lisaks sellele oli meil vaja testida filtreid ning iga kord reaalajas tulevate andmetega oleks testimine väga vaevaline olnud. Kirjutasime andmete taasesitamiseks ja filtreerimiseks eraldi programmi. See jaguneb laias laastus kolmeks.

3.3.1 Andmete sünkroniseerimine

Et andmeid õiges ajalises järjestuses ja sünkroniseeritult esitada olid nad salvestatud kuute erinevasse faili: magnetandurite positsioon, robotilt leitud väravate asukoht, robotilt leitud pallide asukoht, robotilt salvestatud kaadrid, laest leitud pallide asukoht, laest salvestatud kaadrid. Kõik andmed olid varustatud ajatemplitega, mille alguspunktid tuli käsitsi taasesitust vaadates katse-eksitus meetodil paika timmida.

3.3.2 Visualiseerimine

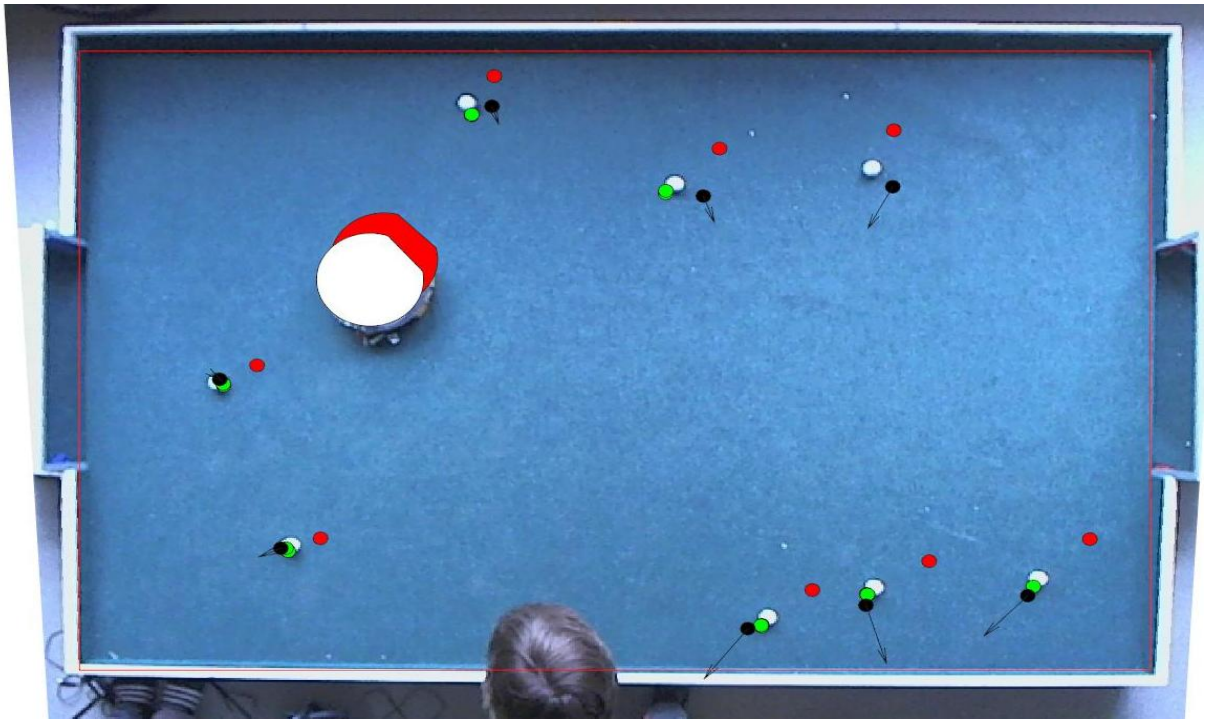
Et veenduda andmete korrektsuses on hea neid visuaalselt kuvada. Selleks lõime algselt väljaku kujutise kuhu peale märkimise ära leitud pallide ja roboti asukohad õiges mõõtkavas. Hiljem panime taustale mängima ka laest võetud video ning lisasime ka laest võetud video pealt leitud pallide visualiseerimise.

3.3.3 Andmete filtreerimine

Filtreerimine on tegelikult selle projekti tuum, kuid selle tegemiseks oli vaja palju eeltööd teha. Filtreerimiseks kasutasime Alfa-Beta filtrit ning Particle filtrit.

3.4 Alfa-Beta filtri implementatsioon

Filtri alustalaks oli lihtne filter, mis tihti esinevad mõõtmistulemused omistas lähimale hüpoteesile. Ja kui mõõtmisi kogunes piisavalt tihti tegi selle hüpoteesi nähtavaks. Nähtavate hüpoteeside peal töötas edasi juba Alfa-Beta filter, mille ülesandeks jäi eelneva filtri tulemusi parandada. Ta tegi seda korrigeerides pallide positsiooni ja kiirust vastavalt määratud parameetritele. Roboti positsiooni uuendamiseks kasutati ainult alfa kordajat, sest roboti positsiooni parandati videotöötluse abil väravate asukohast, aga roboti kiirus oli tänu magnetanduritele teada suure täpsusega. Selline lähenemine osutus lihtsaks ning efektiivseks.



Joonis 1. Rohelised ringid on laest olevalt pildilt tehtud videotöötlus tulem. Punased on roboti enda andmed ning mustad on filtreeritud andmed. (Valge suur ring on roboti filtreeritud asukoht).

3.5 Particle filtri implementatsioon

Tahtsime implementeerida ka ühe keerulisema filtri. Me valisime selleks Particle filtri. Filtri implementeerimine võttis omajagu aega. Alguses proovisime kogu väljaku seisu esitada ühe osakese pilvena, aga selle dimensionaalsus (~40) oli nii suur, et ei õnnestunud simuleerida piisavalt osakesi filtri toimimiseks. Seetõttu pidime igal pallil eraldi osakeste kogumit kasutama. Pallide osakesi liigutati edasi vastavalt nende kiirusele. Nende kiirust vähendati eksponentsiaalselt vastavalt ajasammule.

- Müra pallide ja roboti positsioonile ja kiirusele oli implementeeritud Gaussi jaotusega juhusliku arvuga mille dispersioon oli võrdeline ruutjuurega ajasammust.
- Pallidele seati vastavusse suurimate kaaludega osakeste põhjal leitud keskmine.
- Pallide osakeste kaale hinnati ümber vastavalt nende määratud nähtud palli nurgaveale ja kaugusveale, kasutades Gaussi jaotust.
- Osakest, mille pallil ei olnud vastavuses nähtud palli, kaaluti ümber lähima vastavuseta nähtud palli abil.

- Roboti osakesi kaaluti ümber kõigi pallide järgi.
- Nähtavaks tehti pallid, mille kaal kaalumisel väheneb keskmiselt alla mingi piiri – see tähendab, et enamus osakesi on kogunenud vastava mõõtmistulemuse juurde.
- Osakesi tekitati ja kaotati, kui nende effektiivne hulk läks alla piiri kasutades algortimi “Select with Replacement”.

Tulemuseks oli filter, mis hoidis nähtaval ainult neid palle, mille osakesed olid palli ümber koondunud. Ehk kui kusagilt ilmus nähtavale pall, siis hakkasid vabad osakesed selle ümber koonduma ja kui piisavalt paljud suutsid seda järgida, siis filter muutis vastava palli nähtavaks. Antud lahendus muutuva arvu pallide jaoks on meie arvates üsnagi elegante, kuigi osakese nähtavale ilmumise tingimust peab ehk veel muutma. Siiski kui simuleeritud osakeste pilvi on rohkem või võrdselt pallide arvuga väljakul, ilmuvad ja kaovad filtri väljundis pallid nii nagu peab.

Probleem oli parameetrite paika saamiseks ja ikkagi piisava arvu osakeste tegemine. Kuna osakeste liikumine on veidi kaootiline (osad lüüakse, osad ilmuvad nähtavale, osad jäävad seisma. osad lähevad väravasse). Me saine kesise tulemuse 1000 osakesega palli kohta. Samas see ei olnud roboti riistvaral jooksutatav. Tuleb kas kõvasti optimiseerida ja parameetreid timmida või midagi kavalamat teha.

Mõõtmistulemuste töötlemine

Kuidagi oli vaja ka filtrite tööd mõõta. Nähtavaid palle ja nähtud palle on muutuv arv ja mõlemad filtrid said ka natuke konservatiivsemad kui toored andmed. Seetõttu sai kaua kaalutud sobivat hindamiskriteeriumit. Lihtsalt ruutkeskmine kaugus ei sobi, sest olenevalt arvestatud ja tegelikust pallide arvust ja sidumisest, tuleb see oluliselt erinev. Lõpuks otsustasime jääda kahe suuruse juurde.

1. Keskmine nähtud pallide arv - see on seda parem, mida lähemal on see tegelikule pallide arvule, kuid ei tohi kindlasti olla suurem. Mida suurem see on, seda ahnem on filter.
2. Iga hetk kolme nähtud palli ruutkeskmine kaugus neile lähimast pallist + trahv, kui mitmele on lähimaks sama pall - see iseloomustab filtri väljundi kvaliteeti.

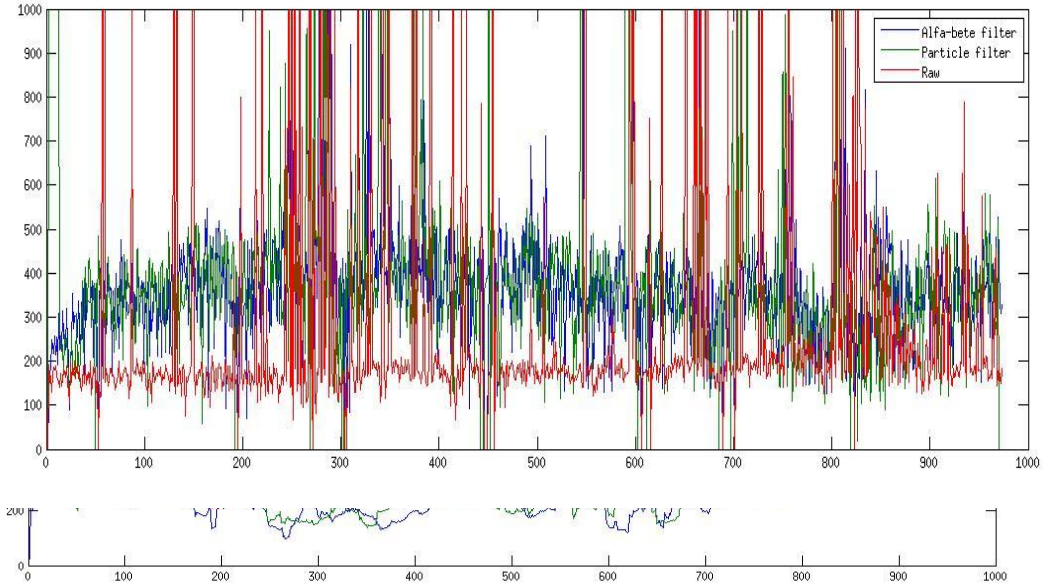
Vastavad suurused sai Alfa-Beta filtri, Particle filtri ja filtrita välja toodud ja Matlab-iga graafikud joonestatud.

4 Eksperimentide tulemused

4.1 Alfa-beta ja Particle filtri võrdlus

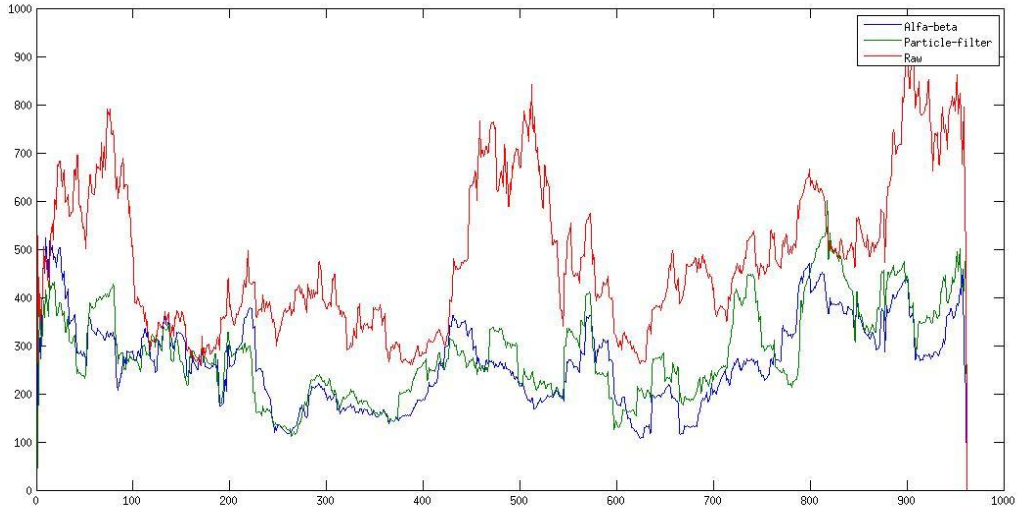
Joonis 2. erinevate filtrite kolme suvalise palli ruutkeskmine kaugus.

Joonisel 2 on näha, et üksikute kaadrite kolme suvalise oletatava palli ruutkeskmine kaugus

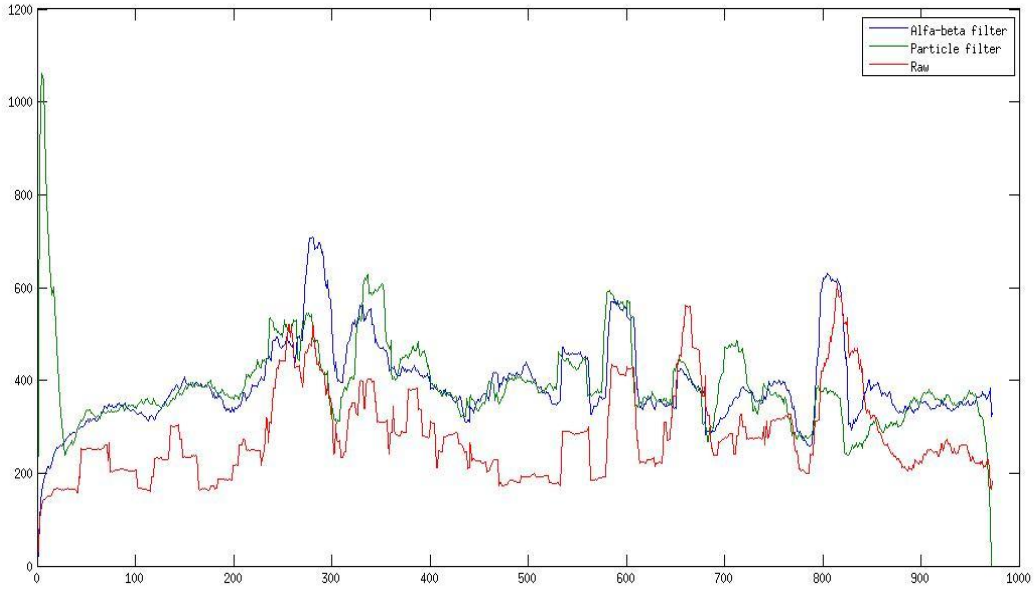


tegelikest pallidest on üsnagi mürane, et saada selgemat pilti teen samad graafikud silutuna 30 kaadrise ~ 1 sekundilise liikuva keskmise filtriga.

Joonis.3 Robot seisab paigal, osasid palle liigutatakse. Kolme suvalise oletatava palli ruutkeskmine kaugus kolmest tegelikust pallist silutuna liikuva keskmisega.



Joonis.4 Robot liigub, osasid palle liigutatakse. Kolme suvalise oletatava palli ruutkeskmine kaugus kolmest tegelikust pallist silutuna liikuva keskmisega.



Joonis.5 Robot liigub, pallid on paigal. Kolme suvalise oletatava palli ruutkeskmine kaugus kolmest tegelikust pallist silutuna liikuva keskmisega.

- Esimesel juhul (joonis 3) on toorete andmete korral kolme juhusliku palli ruutkeskmine kaugus keskmiselt isegi väiksem kui filtreeritud andmete korral - kuid siis ongi robot paigal ja müra pallide asukohtade määramisel seetõttu vähem. Palju suurem viga tuli filtreeritud andmete korral sellest, et roboti positsiooni ei saa väravate abil piisavalt täpselt määrata.
- Teisel ja kolmandal juhul (joonis 4 ja 5) on see tooretel andmetel küll suurem, aga seda paljuski jälle sellepärast, et liikudel läheb positsioon rohkem paigast ära, kui seda väravate pealt ei korrigeerida

Kokku tulid kolme palli ruutkeskmised kaugused ja keskmised arvestatud pallide arvud teisel ja kolmandal juhul järgmised.

	Palle teine	Palle kolmas	RMSE3 teine (mm)	RMSE3 kolmas (mm)
Alfa-beta filter	7.0074	6.9657	258.5076	266.6798
Particle filter	6.5117	6.4875	274.1797	242.4299
toored andmed	7.1367	7.1362	427.4533	417.8451

Arvestades roboti positsiooni viga saab sellest järeldada ainult filtrite suhtelist efektiivsust, mitte võrrelda neid toorete andmetega
Tabelist on näha, et käesolevas implementatsioonis töötasid Alfa-beta filter ja Particle filter peaaegu sama hästi - Particle filter siiski natuke kehvemini.

Mis senises implementatsioonis puudu jäi

Puudulikud olid mõlema videotöötluse geomeetria andmed. Ehk siis pallide asukohad pole päris need, mis nad olema peaksid. Filtrites kasutatakse mõõtmistulemuste ja hüpoteeside sidumiseks mitte väga korrektseid meetodeid. Puudu on Particle filtris osakese kiiruse arvestamine osakesele kaalu määramisel. Kasutades pealtvaate pilti saaks leida mudelisse katsetulemuste põhjal sobivad nurga ja kauguse mõõtemääramatused, mis peaksid parandama oluliselt filtri tulemusi. Puudulik oli roboti positsiooni määramise täpsus, antud testide jaoks oleks pidanud seda kah tegema laest võetud pildilt. Loodaks ikka saada Particle filter saada välja andma paremaid tulemusi kui Alfa-Beta filter.

4.2 Mida tulevikus veel proovida

Kindlasti tahaks ära proovida korrektset JPDAF meetodit. Täpsemalt on meie arust sobiv Monte Carlo JPDAF algoritm, mis tundub olevat täpselt see, mida meil vaja oleks. Omaette küsimus on muidugi selles, kas seda ka reaalajas piisava täpsuse juures rakendada saab.

5 Kokkuvõte

Tohutult aega kulus erinevate olemasolevate algoritmide uurimiseks ning nende matemaatilise tausta tundmaõppimiseks.

5.1 Karl:

Minu arust me ei saavutanud seda tulemust, mida oleksime tahtnud. Particle filter on antud implementatsiooni puhul liiga aeglane, et seda reaalajas robotil kasutada, ning meie implementatsiooni matemaatiline korrektsus on küsitav. Kogu projekt võttis oodatust kordades kauem aega. See meenutab natuke Robotexiks valmistumise kohta käivat reeglit, et kõik ajahinnangud korruta Pii-ga ning, kui tegemist on keerulise asjaga, siis korruta Pii-ruuduga.

5.1.1 Mihkel:

Aruande kirjutamiseks ja mõõtmistulemuste paremaks analüüsiks jäi kindlasti liiga vähe aega. Eelnev töö: katseandmete saamine, laest video õige võtmine, andmete tagasimängija ja visualiseeria(!) kirjutamine, erinevate filtrite uurimine ja katsetamine võttis tohutult aega. Seega on kahju, lõpptulemus natuke kesine on. Praegusel kujul on Particle filter robotil kasutamatu, seda tuleks kindlasti edasi arendada.

5.2 Lingid

http://en.wikipedia.org/wiki/Radar_tracker
http://en.wikipedia.org/wiki/Alpha_beta_filter
http://en.wikipedia.org/wiki/Particle_filter
http://ceit.aut.ac.ir/~shiry/publications/JPDAF%20algorithm.pdf
http://www.cim.mcgill.ca/~yiannis/particletutorial.pdf
http://www-sigproc.eng.cam.ac.uk/~sjg/papers/04/multi_object_revI.pdf