



CHALMERS



GÖTEBORGS UNIVERSITET



Felhantering och prestandabevakning av GulliView som inomhuslokaliseringssystem

Kandidatarbete inom Data- och informationsteknik

Loura Alammour, Oskar Andersson, Samuel Henningsson Kling,
Hampus Johansson, Nadir Morabet och Dalxa Yuusuf

INSTITUTIONEN FÖR DATA- OCH INFORMATIONSTEKNIK

CHALMERS TEKNISKA HÖGSKOLA
Göteborg 2026
www.chalmers.se

KANDIDATARBETE 2026

Felhantering och prestandabevakning av GulliView som inomhuslokaliseringssystem

Loura Alammour
Oskar Andersson
Samuel Henningsson Kling
Hampus Johansson
Nadir Morabet
Dalxa Yuusuf



GÖTEBORGS
UNIVERSITET



CHALMERS

INSTITUTIONEN FÖR DATA- OCH INFORMATIONSTEKNIK
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET
Göteborg 2026

Felhantering och prestandabevakning av GulliView som inomhuslokaliseringssystem

© Loura Alammour 2026.
© Oskar Andersson 2026.
© Samuel Henningsson Kling 2026.
© Hampus Johansson 2026.
© Nadir Morabet 2026.
© Dalxa Yuusuf 2026.

Handledare: Elad Schiller, Institutionen för Data- och informationsteknik
Examinator: Arne Linde, Institutionen för Data- och informationsteknik

Kandidatarbete 2026
Institutionen för Data- och informationsteknik
Chalmers Tekniska Högskola
SE-412 96 Göteborg
Telefon +46 31 772 1000

Skriven i L^AT_EX
Göteborg 2026

Abstract

This bachelor's thesis presents the evaluation and development of GulliView, a camera based indoor localization system for autonomous and cooperative robotic systems. The purpose of the project is to evaluate the localization accuracy of the system and develop methods for monitoring the performance and the uncertainty in positional data.

In the evaluation, the ground-truth coordinates are used as a reference to analyze how accurately the pose estimation method estimates global positions. The evaluation is performed using a dotted map, where RMSE, mean error and variance is used in order to analyze the localization accuracy. In addition, an angle based analysis is conducted in order to see how well the geometric structure of the dotted map is preserved after distortion correction and perspective transformation. The project also introduces multiple methods for error handling, security and long term planing through ROS messages. This is done by having ROS messages that contains information such as age of information (AoI), timestamps and different predictions, such as position and when the next message will arrive to the robot. Furthermore a dynamic motion bounding boxes are developed by combining the robots physical dimensions with the predicted errors and safety margins.

The results in this project show that the system's accuracy varies across the test area and that the distortion as well as communication delays affect the reliability of the positional information. The work in this project provides a foundation for future development of robust and safe localization solutions for autonomous robotic systems.

Sammandrag

Detta kandidatarbete presenterar utvärderingen och utvecklingen av GulliView, ett kamerabaserat inomhuslokaliseringssystem för autonoma och kooperativa robotsystem. Syftet med detta projekt är att utvärdera lokaliseringsnoggrannheten av systemet samt att utveckla metoder för övervakning av systemets prestanda och positionsdata.

I utvärderingsdelen används marksanningskoordinater som referens för att analysera hur noggrant poseuppskattningsmetoden uppskattar globala positioner. Utvärderingen genomförs med hjälp av en punktmatta där uträkning av RMSE, medelfel och varians används för att analysera lokaliseringsnoggrannheten. Dessutom genomförs en vinkelanalys för att undersöka hur väl den geometriska strukturen på punktmattan bevaras efter distorsionskorrigering och perspektivtransformering. Projektet introducerar även metoder med fokus på felhantering, säkerhet och framförhållning för autonoma robotar via ROS meddelanden. Detta görs genom att ROS meddelanden innehåller information så som ålder på informationen (AoI), tidstämplar samt en del olika uppskattningar så som positionsuppskattning och när nästa meddelande anländer till roboten. Vidare har det även utvecklats dynamiska avgränsningsboxar som kombinerar robotens fysiska storlek med prediktionsfel och säkerhetsmarginaler.

Resultaten i projektet visar att systemets noggrannhet varierar över testområdet samt att distorsion och kommunikationsfördröjningar påverkar tillförlitligheten hos positionsinformationen. Arbetet i detta projekt utgör en grund för framtida utveckling för robust och säker lokaliseringslösning för autonoma robotsystem.

Tackord

Vi vill börja med att tack vår handledare Elad Schiller för upplyftande stöd under projektet gång. Hans vägledning har varit till stor hjälp och har lett till att vi kunnat arbeta mer effektivt.

Vi vill även tacka Yen-an Wang för hans värdefulla feedback som har starkt höjt kvalitén på vår rapport.

Slutligen vill vi också tacka Keyvan för hans otroliga stöd med GulliView Noetic koden.

Ordlista

Nedanför finns en lista över beteckningar och ord som har används under projekts gång och som tillkommer i rapporten, orden är sorterade i alfabetisk ordning:

ACK	Står för acknowledgement och är ett svar på ett ROS meddelande
AoI	Age of Information. Mått på hur gammal informationen är.
AprilTag	Visuell markör med binärt mönster för positionsbestämning med kamera.
Avförvrängning	Korrigerig av geometriska bildfel orsakade av linsdistorsion.
EWMA	Exponential Weighted Moving Average, matematisk metod för att generera medelvärde och varians som beroende på tidigare resultat där nyare data har högre prioritet.
Fågelperspektiv	Bildvy rakt uppifrån erhållen via perspektivtransformation.
GulliView	Kamerabaserat inomhuslokaliseringssystem i labbsalen ED5355 på Chalmers Tekniska Högskola.
GulliView Matei	Tidigare enkeltrådad version av GulliView.
GulliView Noetic	Version av GulliView som användas för utvärdering av positioneringsnoggrannhet och distorsionsfel.
Hamming-avstånd	Antal bitar som måste vara fel innan systemet förväxlar två Apriltaggar.
Homografitransformation	Transformation som transformerar punkter från ett plan till ett annat.
Marksanningskoordinater	Referenskoordinater som används som korrekt jämförelsevärde vid utvärdering av ett systems positionsnoggrannhet.

NTP	Network Time Protocol. Protokoll för klocksynkronisering i ett distribuerat system.
Offlinefunktionalitet	Funktioner som körs när hela systemet inte är aktivt
Onlinefunktionalitet	Funktioner som körs när hela systemet är aktivt
OpenCV	Open Source Computer Vision Library, programbibliotek som används i programkod för kamerabaserad datorseende.
Perspektivvy	Ett perspektiv som liknar hur mänskliga ögon och kameror ser. Innebär att objekt som är längre bort är mindre.
Pixelkoordinat	En punkts position i en digital bild.
PnP	Perspective-n-Point. Algoritm för att beräkna objektets pose från kamerabild.
Pose	Ett objekts position och orientering i ett koordinatsystem.
Punktmatta	Matta med markörer på kända avstånd för noggrannhetsutvärdering.
RMSE	Root Mean Square Error. Det statistiska felmått som beskriver den genomsnittliga avvikelsen mellan uppskattade värden och referensvärden.
Robot	System som navigerar i testmiljön med hjälp av Gulliview via ROS.
ROS	Robot Operating System. Ett protokoll som säkerställer trådlös kommunikation mellan system.
Taggfamilj	Grupp AprilTags med samma kodningsschema och rutnätsstorlek.
Testbädd	Yta som täcker Gulliviews synfält och används för diverse tester och provkörningar av systemet.

Innehåll

Ordlista	vii
Figurer	xiv
1 Inledning	1
1.1 Syfte	1
1.2 Problemdefinition	2
1.3 Avgränsningar	2
1.4 Relaterat arbete	3
1.5 Vårt bidrag	3
1.6 Validering	4
2 Teknisk Bakgrund	5
2.1 Kamerabaserat datorseende	5
2.2 AprilTag	5
2.3 Global koordinering	6
2.4 Robot Operativ System	7
2.4.1 Noder	7
2.4.2 Topics	7
2.4.3 TCPROS och UDPROS	8
2.5 Gulliview	8
2.6 Punktbaserad Detektion med Gulliview Noetic	8
2.6.1 Punktmattan	8
2.6.2 USB Camera CV	9
2.6.3 Dot-based detection	9
2.6.4 Functionality	11
2.7 Exponentially Weighted Moving Average	11
2.8 Network Time Protocol	12
2.9 Age of Information	13
3 Metod	15
3.1 Analys av distorsionsfel och geometrisk struktur	15
3.2 Statisk utvärdering av globalt koordinatsystem	16
3.2.1 Bestämmande av marksanningskoordinater	16
3.2.2 Positionering av punkter med poseuppskattning	16
3.2.3 System för evaluering av globala koordinater	17
3.3 Gulliviews meddelandestruktur	18
3.3.1 Hantering av informationsålder	19
3.3.2 Filtrering och stabilisering med EWMA	19

3.3.3	Unik identifiering av meddelande	20
3.3.4	Uppskattning av nästa position	20
3.3.5	Rumslig osäkerhet och säkerhetsmarginaler	20
3.3.6	Nätverksinformation och framtida uppdateringar	22
3.3.7	Verifiering av meddelandets innehåll	22
3.4	Robotens ACK-mekanism	22
3.4.1	Unik identifiering av ack	23
3.4.2	Kompakt kvittering med bitfält	23
3.4.3	Tidstämplar för nätverksanalys	23
4	Evaluering	25
4.1	Forskningsfrågor	25
4.2	Evalueringskriterier	25
4.2.1	Geometrisk noggrannhet	26
4.2.2	Positioneringsfel	26
4.2.3	Informationsålder	26
4.2.4	Estimerad information	27
4.2.5	Skalningsfaktor för avgränsningsbox	27
4.3	Evalueringsmiljö	28
4.4	Experiment	28
4.4.1	Experiment 1: Lokaliseringsfel över testytan	28
4.4.2	Experiment 2: Distorsionsfel och representation av fysiska miljön i bilder	29
4.4.3	Experiment 3: Uppmätt informationsålder	30
4.4.4	Experiment 4: Estimerad information	30
4.4.5	Experiment 5: Skalningsfaktor för avgränsningsbox	31
5	Resultat	33
5.1	Resultat av experiment 1	33
5.2	Resultat av experiment 2	35
5.3	Resultat av experiment 3	36
5.4	Resultat av experiment 4	36
5.5	Resultat av experiment 5	38
6	Diskussion	39
6.1	Diskussion av experiment 1	39
6.2	Diskussion av experiment 2	39
6.3	Diskussion av experiment 3	40
6.4	Diskussion av experiment 4	40
6.5	Diskussion av experiment 5	41
6.6	Framtida arbete	41
6.7	Sammanfattning	42
7	Etiska aspekter	43
	Litteraturförteckning	45
A	Appendix	I
B	Appendix	III

C Appendix	V
D Appendix	VII

Figurer

2.1	Exempel på tagg i taggfamilj 25h9	6
2.2	Exempel på tagg i taggfamilj 36h11	6
2.3	Exempel på ett ROS nätverk	7
2.4	Punktmattan som Gulliview Noetic använder.	9
2.5	Råbild av punktmattan.	10
2.6	Detektion av punkter i fågelperspektivvy.	11
2.7	Hur tidstämplas sätt över tid	13
3.1	Illustration av hur vektorerna beräknas i koden.	15
3.2	Visualisering om avgränsningsbox mellan bilder (till vänster) och avgränsningsbox av nuvarande bild (till höger)	21
5.1	RMSE av 2D euklidiska avståndsfelet över x-axel för kamera 2	34
5.2	Rumslig fördelning av 2D euklidiska avståndsfelet över det utvärderade punktmönstret för kamera 2 högre sidan av testbädden	35
5.3	Låddiagram på uppmätta vinklar över alla fyra kameror	35
5.4	Sammanställd stapeldiagram för representation av informationens ålder	36
5.5	Stapeldiagram över felet i estimering av position	37
5.6	Stapeldiagram över felet i estimering av ankomsttid	37
5.7	Stapeldiagram över felet i estimering av nästa bild	37
D.1	RMSE av 2D euklidiska avståndsfelet över x-axel för kamera 0	VII
D.2	RMSE av 2D euklidiska avståndsfelet över x-axel för kamera 1	VII
D.3	RMSE av 2D euklidiska avståndsfelet över x-axel för kamera 3	VII

1

Inledning

Autonoma och kooperativa robotsystem ställer höga krav på tillförlitlig positionsbestämning, särskilt i miljöer där flera autonoma fordon ska röra sig säkert inom samma område. I inomhusmiljöer är GPS ofta otillräckligt, vilket gör att alternativa lokaliseringssystem behövs. På Chalmers tekniska högskola används testbädden i labbsalen ED5355 för att undersöka hur kamerabaserad inomhuslokalisering kan stödja säkra kooperativa fordonsmanövrar.

Detta arbete utgår från GulliView, ett befintligt kamerabaserat inomhuslokaliseringssystem som använder takmonterade kameror och visuella markörer för att uppskatta robotars positioner. För att systemet ska kunna användas som stöd för säkra manövrar krävs inte endast positionsdata, utan även kunskap om hur noggrann och aktuell informationen är. Lokaliseringsfel, fördröjningar och paketförluster kan påverka möjligheten att förutsäga robotars framtida rörelser och därmed försvåra säkerhetsbedömningar.

Rapporten behandlar därför både utvärdering av GulliViews positioneringsnoggrannhet och vidareutveckling av systemet för att tydligare beskriva osäkerheter i positionsdata. Arbetet omfattar analys av geometriska fel, utvärdering av metod för positionering samt framtagning av information som kan hjälpa robotar och användare av testbädden att resonera kring positioneringssäkerhet i realtid.

1.1 Syfte

Syftet med projektet är att vidareutveckla GulliView som ett inomhuslokaliseringssystem för att möjliggöra säkra kooperativa fordonsmanövrar genom noggrann och tillförlitlig positionsbestämning. Projektet fokuserar på att utvärdera och analysera positionsfel samt hur dessa påverkar systemets förmåga att förutsäga robotars framtida rörelser.

Projektet syftar även till att utveckla metoder och designkoncept för att övervaka, visualisera och resonera kring positioneringsosäkerheter, exempelvis genom offline- och onlineinformation om fel samt konceptet avgränsningsbox. Resultaten förväntas möjliggöra vidareutveckling av robusta positioneringslösningar för autonoma och kooperativa robotsystem.

1.2 Problemdefinition

För att autonoma och kooperativa robotsystem ska kunna framföras säkert krävs ett lokaliseringssystem som levererar noggrann och tillförlitlig positionsinformation i realtid. Positionsfel, föråldrad information, paketförluster och processeringsfördröjningar kan påverka systemets förmåga att förutsäga robotars rörelser och därmed försvåra säkerhetsbedömningar.

Det är därför viktigt att undersöka både GulliViews noggrannhet och hur dess prestanda påverkas av olika felkällor. Projektet kommer därför att utgå från följande två övergripande frågeställningar:

F-1 *Hur noggrann är GulliView vid positionering i inomhusmiljöer?*

F-2 *Hur kan mekanismer designas och implementeras för att övervaka prestandan hos GulliView?*

Dessa frågeställningar är relevanta för att bedöma systemets tillförlitlighet och förstå hur osäkerheter påverkar säkerheten vid kooperativa fordonsmanövrar.

1.3 Avgränsningar

Projektet fokuserar på vidareutveckling av inomhuslokaliseringssystemet GulliView. Eftersom projekt genomförs på en begränsad men kontrollerbar testmiljö så omfattar inte detta projekt en evaluering i andra typer av miljöer eller lokaler än den som finns i labbsal ED5355 på Chalmers Tekniska Högskola.

Under utvärderingen av lokaliseringsnoggrannhet så genomförs endast stationära tester på en punktmatta, och inkluderar därför inte dynamiska eller rörliga lokaliseringsnoggrannhet med rörliga robotar. Projektet fokuserar främst på en tvådimensionell positionering relativt till golvet, jämfört med en fullständig rumslig tredimensionell utvärdering. Därav tar projektet inte hänsyn till parametrar som höjdvariationer under utvärderingen för lokaliseringsnoggrannhet. Lokaliseringsnoggrannhet fokuserar inte heller på utvecklingen utav nya algoritmiska metoder. Därav bidrar inte projekt med en förbättrad noggrannhet utan fokuserar endast på att utvärdera och jämföra de befintliga metoderna för poseuppskattning med dess verkliga positionen, istället för att ersätta den med alternativa lösningar.

Slutligen inkluderar projekt en design av meddelandetyper till autonom navigering, kollisionsutveckling och relevansuppskattning av informationen för robotar. De metoderna som används i denna delen av projektet används därav som stöd för framtida säkerhets- och navigeringssystem. Projektet försöker inte att utveckla nya metoder för navigering, kollision eller uppskattning utan applicerar kända metoder som kan vara relevant för framtida projekt. Vidare är nätverksanalysen begränsad, det existerar i meddelandets delar för störnings- och latensuppskattning, men undersöks inte ingående då det sträcker sig utanför projektets ramar.

1.4 Relaterat arbete

Arbetet som presenteras i denna rapport bygger vidare på tidigare utvecklade system i labbsal ED5355. GulliView har utvecklats iterativt över flera projektgrupper, där en tidigare grupp implementerade ett flertrådigt system med stöd för global koordinering [16]. En annan grupp reducerade fördröjningen i systemet samt implementerade ett globalt koordinatsystem som gjorde GulliView mer användbart för labbsalens robotar[17].

Det punktbaseade utvärderingssystemet som används i detta projekt bygger på GulliView Noetic[16], ett tidigare utvecklat system som använder en punktmatta med cirkulära markörer för att utvärdera positioneringsnoggrannheten. Bildbehandlingen stöds av OpenCVs bibliotek, och kommunikationen mellan systemets komponenter sker via ROS som transportprotokoll.

Vinkelanalysmetoden som introduceras i detta projekt relaterar till principer inom kamerabaserat datorseende och kamerakalibrering, specifikt Zhangs kalibreringsmetod [14] och användningen av AprilTags för positionsbestämning [13].

Evalueringen av det globala koordinatsystemet utförs på en poseuppskattningsmetod baserad på Perspective-n-Point [15]. Riskkriterier för säkerhetsmarginaler i autonoma system har tidigare studerats i labbsal ED5355 och utgör grunden för evalueringen av avgränsningsboxens skalningsfaktor i detta projekt[17].

1.5 Vårt bidrag

Projektet bidrar med ett felövervakningssystem med offline- och onlinefunktionalitet som ger både robotar och operatörer möjlighet att resonera kring positioneringssäkerheter i realtid.

Genom systematisk evaluering av systemets positionsnoggrannhet identifierades ett genomsnittligt positionsfel på upp till 0.298 m beroende på kamera, med kamera 3 som mest noggrann med ett RMSE på 0.039 m. Felen uppvisar en systematisk karaktär med tydliga riktningsavvikelser, vilket indikerar strukturella brister i koordinattransformationen för vissa kameror. En ny vinkelanalysmetod introduceras för att mäta hur väl bildbehandlingen bevarar den geometriska strukturen hos miljön, eftersom geometriska fel i bildplanet propagerar direkt till fel i det globala koordinatsystemet och därmed försämrar säkerheten vid navigering.

För att utöka säkerheten i labbsal ED5355 introduceras ett utökat ROS-meddelandeformat med informationsålder (AoI), tur-och-returtid (RTT), predikterad nästa position och EWMA-filtrerad statistik. Informationsåldern uppmättes till i genomsnitt 112 ms, vilket ger robotarna ett kvantitativt underlag för att bedöma hur aktuell den mottagna informationen är. Ett ACK-protokoll implementerades som möjliggör beräkning av paketförlust. Dynamiska rörelseavgränsningsboxar introduceras för att kombinera robotens fysiska storlek med positionsosäkerheten från EWMA-prediktionsfelet, vilket möjliggör en högre grad av säkerhet vid kooperativa manövrar.

Resultaten utgör ett konkret underlag för framtida grupper att designa manövrar som är

medvetna om positioneringsfel samt undersöka hur förbättringar av lokaliseringssystemet påverkar noggrannheten. Systemet kan även testas i nya miljöer och användas för att utveckla säkrare kooperativa rörelsebeslut för autonoma fordon i labbsal ED5355. Källkoden för detta projekt hittas i Appendix A.

1.6 Validering

Systemets prestanda valideras genom praktiska experiment i labbsal ED5355 för att undersöka om projektets syfte har uppnåtts. Valideringen fokuserar på GulliViews förmåga att leverera noggrann och tillförlitlig positionsinformation som kan användas vid säkra kooperativa fordonsmanövrar.

Positioneringsnoggrannheten undersöks med hjälp av punktmattan, där systemets uppskattade positioner jämförs med referenspunkter. Samtidigt studeras systemets stabilitet, uppdateringstid och påverkan av felkällor såsom föråldrad information, paketförluster och processeringsfördröjningar. Utöver positionsnoggrannheten valideras även informationens aktualitet och tillförlitlighet över tid. Detta används för att bedöma systemets förmåga att hantera osäkerheter vid kooperativa manövrar.

2

Teknisk Bakgrund

I detta kapitel presenteras den kunskapen som GulliView är baserat på. Samt de teorier som bygger på vidareutvecklingen utav GulliView som inomhuslokaliseringssystem. Den första delen i kapitlet har fokus på positioneringsnoggrannheten medans den senare delen introducerar de metoder som används för säkerheten av autonoma fordon.

2.1 Kamerabaserat datorseende

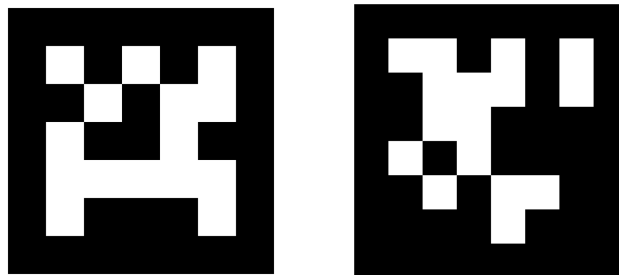
Datorseende är ett område inom datavetenskap som syftar till att möjliggöra för datorer att uppfatta och tolka visuell information på ett sätt som liknar människans visuella förmåga. Precis som människor använder sina ögon för att samla in information och hjärnan för att analysera och tolka den, handlar datorseende om hur datorer kan analysera bilder och videoströmmar med hjälp av algoritmer [10].

Ett vanligt verktyg inom datorseende är OpenCV (Open Source Computer Vision Library). Det är ett kraftfullt bibliotek med öppen källkod som erbjuder ett brett utbud av funktioner för effektiv bildanalys. Genom funktioner såsom kantdetektering, objektidentifiering och bildinläsning kan OpenCV användas för att analysera bilder som tagits med kameror [11]. Detta möjliggör extraktion av relevant information, exempelvis pixelkoordinater på objekten.

Pixelkoordinater beskriver en punkts position i en digital bild och definieras vanligtvis i ett tvådimensionellt koordinatsystem (u,v), där u representerar den horisontella axeln (kolumnindex) och v den vertikala axeln (radindex). Origo är i regel placerat i bildens övre vänstra hörn.

2.2 AprilTag

AprilTags är ett visuellt markörsystem [13]. Systemet består av fysiska markörer med ett unikt binärt mönster samt en mjukvarualgoritm som detekterar och avkodar dessa i kamerabilder. Markörerna organiseras i taggfamiljer, där varje familj definierar rutnätets storlek, antalet databitar samt systemets robusthet mot detekteringsfel uttryckt som Hamming-avstånd. I detta projekt används två familjer: `tag25h9` för kalibreringstaggar och `tag36h11` för robotstaggar.



Figur 2.1: Exempel på tagg i taggfamilj 25h9 **Figur 2.2:** Exempel på tagg i taggfamilj 36h11

Detekteringen sker genom att bilden omvandlas till gråskala och därefter identifieras det binära datamönstret med kantdetektering för att sedan avkodas. Med hjälp av taggens kända fysiska storlek och kamerans intrinsiska parametrar beräknas sedan taggens position och orientering (pose) via PnP (Perspective-n-Point) algoritmen [15]. En fördel med AprilTags är att de möjliggör estimering av pose från en enda kamera. En begränsning är att taggen måste vara synlig för kameran och att noggrannheten påverkas av taggens storlek i förhållande till kamerans upplösning och avstånd. När en Apriltag detekteras kan dess pose bestämmas i förhållande till kameran. För att flera kameror och objekt ska kunna relateras till varandra så omvandlas denna information vidare till ett gemensamt globalt koordinatsystem.

2.3 Global koordinering

Global koordinering innebär att alla objekt och system använder ett gemensamt koordinatsystem i verkliga måttenheter, exempelvis meter, i stället för separata lokala koordinatsystem. Detta möjliggör att positioner och orienteringar (pose) kan uttryckas i samma referensram oberoende av vilken kamera eller vilket delsystem som används. För att åstadkomma detta definieras ett globalt origo på testbädden, varefter samtliga objekt och system refererar till detta koordinatsystem.

För att varje kamera ska kunna relatera sina observationer till det globala koordinatsystemet krävs att kamerans pose, bestäms i förhållande till testbädden. Detta görs med hjälp av statistiska kalibreringstaggar med kända positioner i miljön. Genom att detektera dessa taggar i kamerabilden kan kamerans relation till testbädden beräknas, vilket gör att alla kameror kan uttrycka observerade objekt i samma globala referensram [16].

Poseuppskattningen bygger på en metod där kända tredimensionella punkter på en April-Tag relateras till motsvarande tvådimensionella punkter i bildplanet. Kärnan i denna metod utgörs av OpenCV-funktionen `solvePnP` [16], som beräknar en rotationsvektor och en translationsvektor utifrån dessa punktkorrespondenser samt kamerans inre parametrar. Resultatet beskriver objektets pose relativt till kameran.

Den beräknade transformationen kombineras därefter med tidigare bestämda transformationer mellan kamera och kalibreringstaggar för att uttrycka objektets pose i det globala koordinatsystemet [?]. När flera kalibreringstaggar används kan flera uppskattningar av objektets globala position och orientering erhållas. Dessa kombineras sedan för att ge en

mer robust slutlig uppskattning av objektets pose i den gemensamma globala referensramen.

2.4 Robot Operativ System

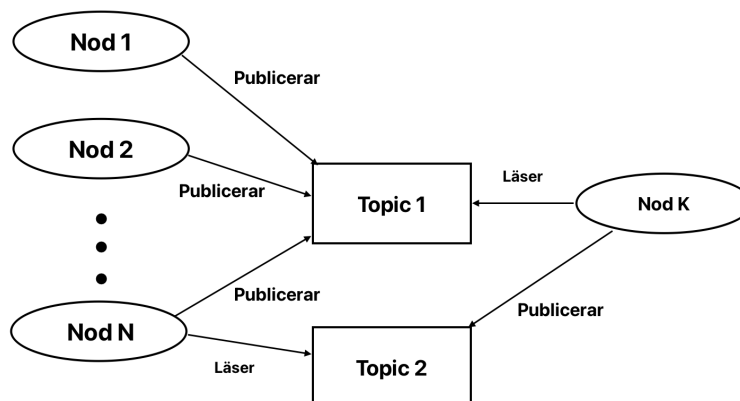
ROS eller Robot Operativ System (jmf. eng. Robot Operating System) är en typ av öppet källkodat mellanlager (jmf. eng. open source middleware). ROS är menat att fungera som en kommunikationslänk mellan olika ROS system och möjliggör att de kan kommunicera med varandra på olika typer utav hårdvara [1]. ROS är i grunden en samling bibliotek av funktioner och variabler som skapar möjligheten för robotar och diverse system att dela information från en mängd olika sensorer, motorer och kameror m.m [1]. ROS appliceras på applikationsnivå och en samling utav ROS system som talar med varandra kallas för ett ROS nätverk.

2.4.1 Noder

För att visa upp vilka system som existerar på ett ROS nätverk så har ROS ett fenomen som kallas noder (jmf. eng. nodes) vilket är ett samlingsord för en process som utför någon form av beräkning. Noder kan kopplas samman i ett nätverk och dela information med andra noder från så kallade Topics se avsnitt 2.4.2 [2]. En robot behöver inte bestå utav endast en nod, den kan också bestå av flera noder, som möjliggör för olika delar av samma system att ta del av all information samtidigt. Det är också med hjälp av dessa noder som gör att kommunikation mellan robotar är möjligt och att de dela information mellan sig själva.

2.4.2 Topics

För att noder ska kunna dela information mellan sig används något som kallas Topics [3]. På Topics kan noderna publicera information men också välja att läsa information från en specifik Topic. Varje nod har möjligheten att publicera eller läsa från ett stort antal olika Topics som andra noder väljer att publicera till. När noder publicerar eller läser från Topics så sker det anonymt [3]. Med andra ord, informationen som existerar på en specifik Topic delas utan publicistens medvetande, det samma gäller när noder läser från en Topic, då vet inte publicisten vilken nod på nätverket som läser den.



Figur 2.3: Exempel på ett ROS nätverk

2.4.3 TCPROS och UDPROS

För att skicka meddelanden till och från Topics så använder sig ROS utav ett TCP (Transmission Control Protocol) liknande protokoll som standard ofta kallat TCPROS [4]. TCP säkerställer tillförlitlig dataöverföring genom att garantera att alla paket levereras i rätt ordning och utan förlust. Detta kan dock innebära något högre latens.

Det är även möjligt att konfigurera ROS att använda UDP (User Datagram Protocol), ofta kallat UDPROS [5]. UDP erbjuder snabbare överföring med lägre latens, men saknar garantier för att alla paket levereras eller att de kommer fram i rätt ordning.

2.5 GulliView

GulliView är namnet på det befintliga inomhuslokaliseringssystemet som används för att tillhandahålla positionsdata till objekt, som i detta system utgörs av robotar. Systemet använder fyra kameror för att samla in bilddata och beräkna positioner baserat på AprilTags. GulliView är en vidareutveckling av ett tidigare projekt, GulliView Matei.

Den senaste versionen av GulliView är flertrådad och använder poseuppskattning för att beräkna globala positioner, medan Matei-versionen är enkeltrådad och baseras på en homografimetod.

Det finns även en tredje version, GulliView Noetic, som används för att utvärdera systemets noggrannhet. Denna version beskrivs mer ingående i ett senare avsnitt (se Avsnitt 2.6).

2.6 Punktbaserad Detektion med GulliView Noetic

GulliView Noetic är ett system som analyserar en punktmatta och beräknar pixelkoordinater för samtliga punkter. Systemet består av fyra delsystem: USB Camera CV, Dot-Based Detection, Functionality och AprilTag ROS. Tillsammans bildar dessa en pipeline som omvandlar en råbild av punktmattan till en lista av pixelkoordinater för samtliga punkter, uttryckta i råbildens koordinatsystem.

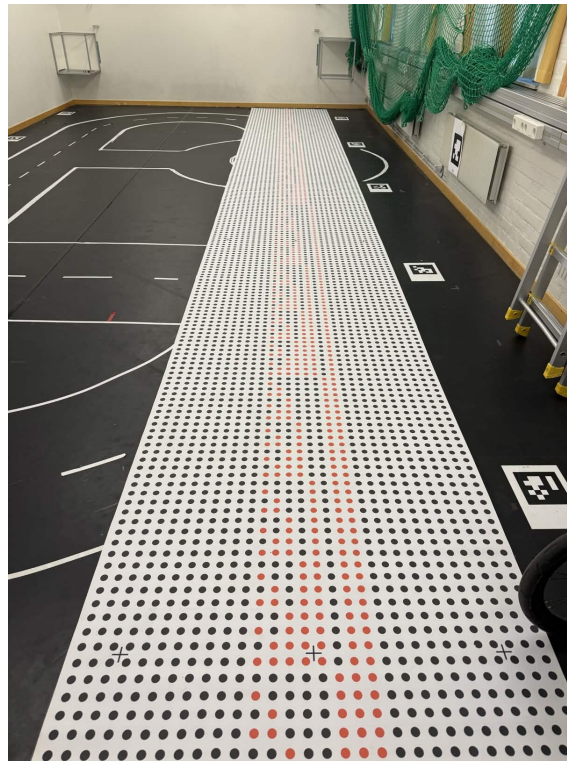
2.6.1 Punktmattan

För att möjliggöra en noggrann och reproducerbar utvärdering av systemets mätprecision används en punktmatta, se Figur 2.4. Punktmattan utgörs av ett regelbundet tvådimensionellt rutnät bestående av cirkulära markörer (hädanefters benämnda punkter), vars geometriska egenskaper är kända med hög precision. Avståndet mellan centrum för två intilliggande punkter är konstant och uppgår till 42 mm, vilket ger en tydlig och väldefinierad referensstruktur.

Varje rad i punktmattan innehåller totalt 30 punkter, vilket innebär att mattan täcker en relativt stor yta och därmed möjliggör analys över ett brett mätområde. Den regelbundna strukturen gör det möjligt att använda punktmattan som referens vid både geometri- och felanalys. Genom att jämföra de uppmätta punktpositionerna från systemet med de kända, ideala positionerna kan avvikelser i exempelvis skala, rotation och translation identifieras.

Vidare möjliggör punktmattan även analys av lokala distorsioner och icke-linjära fel i systemet. Eftersom punkterna är jämnt fördelade över hela ytan kan variationer i noggrannhet

beroende på position studeras. Detta är särskilt relevant för optiska eller bildbaserade mätsystem där linsdistorsion eller perspektivfel kan påverka resultatet.



Figur 2.4: Punktmattan som GulliView Noetic använder.

2.6.2 USB Camera CV

USB Camera CV ansvarar för att hämta bilder från kamerorna med hjälp av OpenCV och publicera dessa som råbilder på en Topic som heter `/cameraX/image_raw`, där X identifierar den aktuella kameran. Till exempel motsvarar `/camera3/image_raw` bildströmmen från kamera 3. Detta möjliggör att andra systemkomponenter kan läsa från Topicen och använda bilddatan för vidare bearbetning. Ett exempel på en råbild av punktmattan visas i Figur 2.5.

2.6.3 Dot-based detection

Dot-based detection består av flera delsteg. Först tas en råbild emot från `/cameraX/image_raw`. Denna bild avförvrängs med hjälp av OpenCV och kamerans kalibreringsparametrar. Detta steg utförs av noden `image_undistortion_node`.

Därefter transformeras den avförvrängda bilden till en fågelperspektivvy med hjälp av en homografitransformation. Transformationen görs med funktionen `cv2.warpPerspective` från OpenCV, där homografimatrisen $\mathbf{H}$ ges som indata tillsammans med bildens storlek och används för att projicera varje pixel till en ny position i målplanet. Homografin är konstruerad så att upplösningen i fågelperspektivbilden motsvarar 1 pixel per millimeter i verkligheten, vilket innebär att avstånd i bilden direkt kan tolkas som millimeter i det globala koordinatsystemet [18]. Detta steg utförs också i `image_undistortion_node`. Den re-

sulterande fågelperspektivbilden publiceras därefter på Topic `/cameraX/image_top_view`.

Den geometriska transformationen beskrivs av homografimatrisen $\mathbf{H}$, som mappar bildkoordinater i pixelplanet till koordinater på testbäddens plan:

$$\begin{bmatrix} x' \\ y' \\ w \end{bmatrix} = \mathbf{H} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (2.1)$$

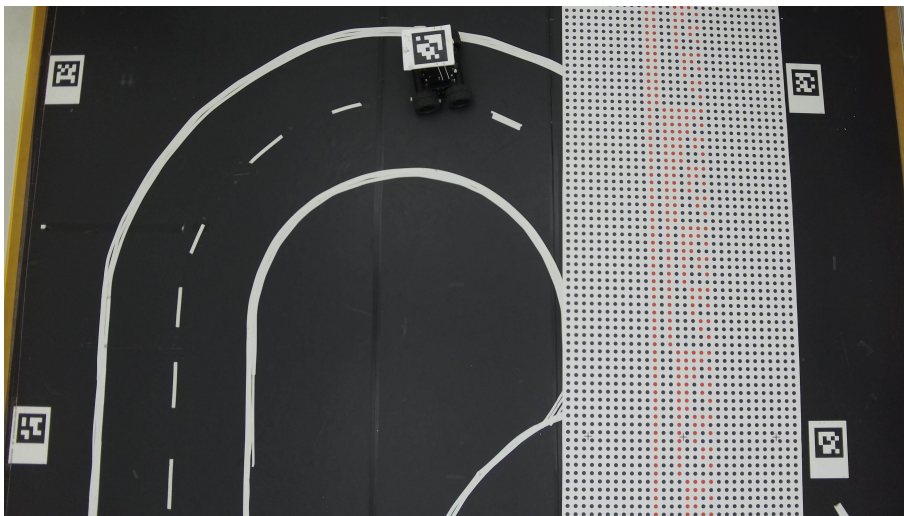
De transformerade koordinaterna erhålls därefter genom homogen normalisering:

$$x' = \frac{x'}{w}, \quad y' = \frac{y'}{w} \quad (2.2)$$

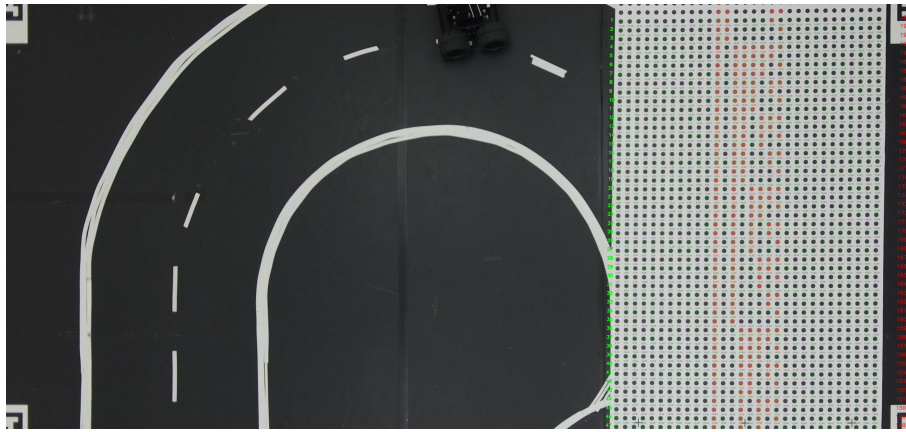
Homografimatrisen $\mathbf{H}$ är förberäknad och unik för varje kamera. Den estimeras genom att använda kalibreringstaggar med kända positioner på testbäddens verkliga koordinater som sitter utmed bildens kanter (se Figur 2.5). Positionernas motsvarande pixelkoordinater extraheras från en avförvrängd bild, och därefter beräknas $\mathbf{H}$ genom att lösa en perspektivtransform mellan punktmängderna i bildplanet och i testbäddens plan.

Transformationen innebär att bilden projiceras till ett plan där effekter av perspektivvyn, det vill säga objekt som är längre bort uppfattas som mindre, elimineras. Det är dock viktigt att notera att avstånd endast blir metriskt korrekta om homografin är korrekt kalibrerad mot ett verkligt plan. Resultatet är en bild där objekt på detta plan representeras som om de observeras rakt uppifrån så kallad fågelperspektiv, vilket förenklar vidare bildbehandling och detektion av punkter.

Punktdetektionen utförs på fågelperspektivbilden med hjälp av OpenCV-funktionen `cv2.findContours`, vilket används för att identifiera punkter i bilden. Detta steg utförs av noden `circle_grid_auto_pattern`, som läser från Topic `/cameraX/image_top_view` och därmed hämtar fågelperspektivbilden för vidare bearbetning. Ett exempel på detektionen visas i Figur 2.6. För varje detekterad punkt beräknas dess centrum, och dessa positioner extraheras som pixelkoordinater. De resulterande koordinaterna lagras därefter i en fil för vidare bearbetning och analys.



Figur 2.5: Råbild av punktmattan.



Figur 2.6: Detektion av punkter i fågelperspektivvy.

2.6.4 Functionality

Delssystemet *Functionality* används för att transformera pixelkoordinater för punkter i fågelperspektivvyn tillbaka till motsvarande pixelkoordinater i den avförvängda bilden. Detta åstadkoms genom att applicera inversen av homografimatrisen, $\mathbf{H}^{-1}$, på samtliga detekterade punkter, det vill säga

$$\mathbf{x} = \mathbf{H}^{-1}\mathbf{x}',$$

där varje punkt transformeras individuellt tillbaka till den ursprungliga bildens koordinatsystem, vilket i detta fall motsvarar den avförvängda bilden.

2.7 Exponentially Weighted Moving Average

Exponentiellt viktat glidande medelvärde (EWMA) (jmf. eng. Exponentially Weighted Moving Average) är en rekursiv statistisk metod som beräknar ett glidande medelvärde där alla tidigare observationer påverkar värdet men olika mycket [6]. Metoden använder en faktor, α , för att successivt minska betydelsen av äldre data, vilket gör att EWMA både bevarar historisk information och anpassar sig till nya trender.

EWMA för en tidsserie y_t vid tidpunkt t beräknas som:

$$S_t = \alpha y_t + (1 - \alpha)S_{t-1} \quad (2.3)$$

där S_t är EWMA-värdet, y_t är den aktuella datapunkten, ($0 < \alpha < 1$) är glömfaktor.

EWMA minskar variationen i tidsseriedata genom att inkludera äldre observationer med successivt minskad relevans, vilket reducerar effekten av plötsliga förändringar och slumpmässigt buller. Samtidigt bevarar metoden historisk information, till skillnad från enkla glidande medelvärden som endast beaktar data inom ett fast fönster. Detta gör att EWMA kan fånga långsiktiga trender utan att förlora relevant information från tidigare tidpunkter. Glömfaktorn α kan dessutom justeras för att styra hur snabbt EWMA reagerar på ny data, vilket ger flexibilitet beroende på om man önskar en stabilare, långsammare anpassning eller en mer känslig, snabbare reaktion på nya trender.

Förutom att beräkna ett glidande medelvärde kan EWMA även användas för att uppskatta variansen i en tidsserie. Detta görs genom att först beräkna EWMA för medelvärdet och

sedan använda detta för att uppskatta variansen. Den rekursiva formeln för EWMA-variens ges av:

$$\sigma_t^2 = \beta(y_t - \mu_{t-1})^2 + (1 - \beta)\sigma_{t-1}^2 \quad (2.4)$$

där:

σ_t^2 är EWMA-variensen vid tidpunkt t ,

y_t är den aktuella observationen,

μ_{t-1} är EWMA-medelvärdet vid tidpunkt $t - 1$,

β är en glömfaktor för variansen ($0 < \beta < 1$),

σ_{t-1}^2 är EWMA-variensen vid tidpunkt $t - 1$.

Glömfaktorn β styr hur snabbt variansuppskattningen anpassar sig till ny data. Ett lågt β -värde ger en stabilare uppskattning, medan ett högt β -värde gör att variansen snabbt anpassar sig till förändringar.

2.8 Network Time Protocol

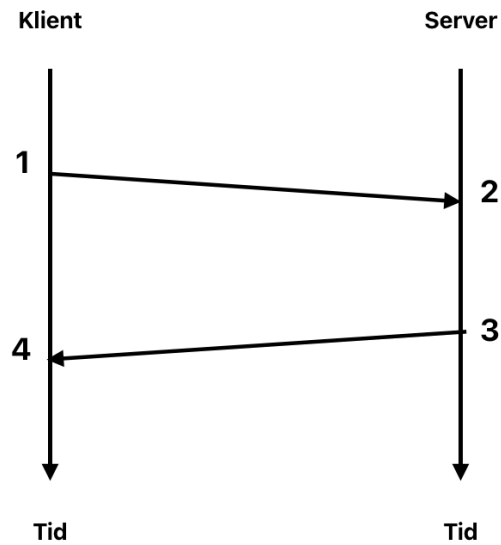
För distribuerade system där flera system kommunicerar med varandra över ett nätverk är det viktigt att de har en synkroniserad klocka som fungerar mellan varandra. När de kommunicerar med information som tidsstämplar är det viktigt att de har samma uppfattning av vad tiden är, minsta fel i klockan kan introducera betydande fel i beräkningar [8]. I verkligheten är de sällan perfekt synkroniserade, utan de kan skilja gradvis i tid beroende på olika faktorer, som hårdvara eller omgivningens temperatur.

Network Time Protocol (NTP) (jmf. sv. nätverksprotokoll för tid) är ett sätt att synkronisera tidsdifferensen på ett nätverk. Den använder en hierarkimodell utav klienter och servrar [8]. Klienten byter ut meddelanden med en server med jämna mellanrum, vid varje utbyte samlas fyra tidsstämplar vilket möjliggör beräkning av tur-och-retur-tid (RTT) och offset enligt:

$$RTT = (T_4 - T_1) - (T_3 - T_2), \quad (2.5)$$

$$offset = \frac{(T_2 - T_1) + (T_3 - T_4)}{2}. \quad (2.6)$$

Där T_1 är tiden då klienten skickade meddelandet, T_2 då servern tar emot meddelandet, T_3 då server skickar tillbaka ett meddelande och T_4 då klienten slutligen får tillbaka serverns skickade meddelande se figur 2.7. Denna metod möjliggör att klienten kan justera sin lokala klocka utefter vad servern meddelar är klientens tidsförskjutning gentemot server [8].



Figur 2.7: Hur tidstämplas sätt över tid

2.9 Age of Information

I ett realtidssystem är det viktigt att dess mottagare tar emot information så tidigt som möjligt. I ett scenario där ett objekt ska motta en positionering, är det viktigt att informationen är aktuell, då objektet rör sig kontinuerligt förlitar den sig på information som är så aktuell som möjligt. Desto mer tid som går blir informationen som systemet har skickat mindre aktuell för mottagaren. Denna skillnad som skapas mellan hur gammal mottagarens senaste uppdatering är kan beskrivas av begreppet Age of Information (AoI) (jmf. sv. ålder av information).

AoI definieras som den tid som har förflutit sedan sändaren skickade statusuppdateringen tills att den mottogs utav mottagaren [7]. Om informationen skapades vid tidpunkten $u(t)$, och t är tiden informationen mottogs:

$$\Delta t = t - u(t). \quad (2.7)$$

Värdet Δt beskriver hur informationen åldras i realtid [7]. Vid tiden då mottagaren tar emot informationen minskar Δt till skillnaden mellan ankomsttid och sändningstid mellan mottagare och sändare. Anledningen varför man inte fokuserar på att optimera latensen mellan sändare och mottagare är att om information skickas för snabbt till mottagaren kan den bli överlastad med information och inte få nytta av den snabba uppdateringstiden [7].

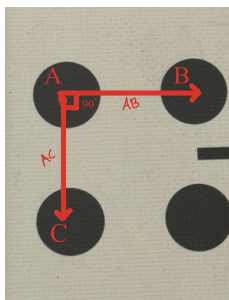
3

Metod

Detta kapitel beskriver de metoder som genomförs under projektets gång. Kapitlet behandlar analys av geometriska fel, statisk utvärdering av det globala koordinatsystemet samt de implementationer som krävs för GulliViews meddelandestruktur och robotens ACK-mekanism. Syftet är att tydliggöra hur systemets positionsnoggrannhet, kommunikation och osäkerhet hanteras och utvärderas.

3.1 Analys av distorsionsfel och geometrisk struktur

Den fysiska miljön i labbsalen förändras i bilderna tagna av kameran på grund av resterande distorsionsfel och för att se hur mycket av det som förvrängas så utvecklades en metod som tar reda på vinklarna mellan punkterna på punktmattan. Punkterna är ortogonala mot varandra i den verkliga se fig 3.1. Men de är inte helt ortogonala i de tagna bilderna av kamerorna. För varje punkt **A** i rutnätet så definieras två andra närliggande punkter, en horisontell närliggande punkt **B** och vertikal närliggande punkt **C**. Dessa punkter definieras sedan som vektorer som genom att subtrahera koordinaterna:



Figur 3.1: Illustration av hur vektorerna beräknas i koden.

$$\overrightarrow{AB} = B - A \quad (3.1)$$

$$\overrightarrow{AC} = C - A \quad (3.2)$$

Sedan används dessa två vektorer till att beräkna vinkeln och se ifall de punkterna verkligen är ortogonala mot varandra. För vinkelberäkning så används skalärprodukten:

$$\overrightarrow{AB} \cdot \overrightarrow{AC} = AB_x \cdot AC_x + AB_y \cdot AC_y \quad (3.3)$$

Denna produkt relateras sedan till vinkeln via

$$\frac{\vec{AB} \cdot \vec{AC}}{|\vec{AB}| \cdot |\vec{AC}|} = \cos\theta \quad (3.4)$$

där $|\vec{AB}|$ och $|\vec{AC}|$ är magnituden av vektorerna.

Genom att ta arccosinus får vinkeln θ som beskriver vinkelrelationen mellan dessa tre punkter. Detta upprepas sedan för alla punkter för ge en vinkelrelationen mellan alla punkter på punktmattan i de tagna bilderna. Algoritmen för samtliga punkter kan sammanfattas som:

Algorithm 1 Algoritm för vinkelrelation mellan punkter

```

1: for each dot A in the grid:
2:   B ← nearest neighbor to the right
3:   C ← nearest neighbor below
4:   AB ← B − A
5:   AC ← C − A
6:   theta ← arccos( $\frac{\text{dot}(AB, AC)}{|AB| \times |AC|}$ )
7:   store theta
8: endfor
9:
10: deviation ←  $|mean(theta) - 90^\circ|$ 

```

3.2 Statisk utvärdering av globalt koordinatsystem

Den statistiska utvärderingen av det globala koordinatsystemet genomförs med hjälp av GullView Noetic, vilket beskrivs i den tekniska bakgrunden (se Avsnitt 2.6). Samtliga implementationer som behandlas i detta delkapitel har utförts inom detta system.

3.2.1 Bestämmande av marksanningskoordinater

Marksanningskoordinaterna definierades utifrån de kända verkliga positionerna för kalibreringstaggarna som placerades på golvet. De detekterade pixelpositionerna för dessa taggar matchades med deras uppmätta koordinater i golvplanet. Dessa motsvarande punktpar användes därefter för att beräkna en homografimatrix, som används för att göra en perspektivtransformation mellan kamerabilden och golvplanet. Transformationen omvandlar kamerabilden till en fågelperspektivsvy av golvet; för de bakomliggande beräkningarna, se Avsnitt 2.6.3. Homografimatriken justerades så att en pixel motsvarar en millimeter i verkligheten efter perspektivtransformationen. Därmed kunde positionerna i den transformerade fågelperspektivsbilden uttryckas direkt som koordinater i golvplanet, angivna i millimeter. Dessa koordinater sparades därefter i en CSV-fil och användes som referenskoordinater vid utvärdering av andra metoder för uppskattning av globala koordinater.

3.2.2 Positionering av punkter med poseuppskattning

Metoden syftar till att bestämma globala koordinater för varje detekterad punkt på punktmattan. Dessa beräknas med hjälp av en poseuppskattningsmetod, som nu är implementerad i den nuvarande versionen av GullView. En mer detaljerad beskrivning av den underliggande logiken för denna metod och de tillhörande beräkningarna presenteras i teknisk

bakgrund (se Avsnitt 2.3). Poseuppskattningsmetoden beaktar både kamerans och objektens position och orientering i tre dimensioner, inklusive translation och rotation. Detta möjliggör koordinatberäkning i 3D och innebär att objekten inte behöver finnas sig i samma höjdplan.

Som beskrivs i tekniska bakgrunden (se Avsnitt 2.6.3) detekterar `circle_grid_auto_pattern` ursprungligen endast punkternas centrum. Logiken för beräkning av globala koordinater kräver fyra hörnpunkter. För att kunna utföra positionsmappningen, blir första steget därför att anpassa `circle_grid_auto_pattern` så att den, utöver centrum, även identifierar fyra hörnpunkter för varje punkt.

För att möjliggöra mappning av punkternas positioner översattes logiken för beräkning av globala koordinater från C++ till Python. Logiken därefter implementeras i filen `coordinate_processor` därefter ändras kvadraten sidlängd från 0.16 cm till 0.02 cm eftersom kvadraten runt punkterna har sidlängd 20 mm. De erhållna koordinaterna beräknade med poseuppskattningsmetoden sparas därefter i en CSV-fil, tillsammans med marksanningskoordinaterna, beskrivna i Avsnitt 3.2.1, för vidare analys.

3.2.3 System för evaluering av globala koordinater

Ett system utvecklades i Python för att möjliggöra en kvantitativ utvärdering av metoden för beräkning av globala koordinater, beskriven i Avsnitt 3.2.2, genom att jämföra dessa koordinater med marksanningskoordinater, beskrivna i Avsnitt 3.2.1. Systemet tar som indata en fil innehållande koordinater beräknade med poseuppskattningsmetoden samt motsvarande marksanningskoordinater för varje punkt på punktmattan i en bild.

För att utvärdera noggrannheten hos poseuppskattningsmetoden beräknas ett positionsfel mellan de uppskattade koordinaterna och motsvarande marksanningskoordinater. Låt de uppskattade koordinaterna för ett givet exempel i betecknas som (x_i^{est}, y_i^{est}) , och motsvarande marksanningskoordinater som (x_i^{gt}, y_i^{gt}) . Positionsfelet i respektive koordinatriktning definieras som

$$e_{x,i} = x_i^{est} - x_i^{gt} \quad (3.5)$$

$$e_{y,i} = y_i^{est} - y_i^{gt} \quad (3.6)$$

För att erhålla ett sammanfattande mått på positionsfelet i planet beräknas det euklidiska avståndsfelet enligt

$$\|e_i\|_{xy} = \sqrt{e_{x,i}^2 + e_{y,i}^2} \quad (3.7)$$

Detta mått representerar storleken på lokaliseringsfelet för varje enskilt prov, oberoende av riktning. Resultaten används därefter för att analysera systemets noggrannhet och jämföra de två metodernas prestanda.

För att sammanfatta den totala noggrannheten över samtliga prover beräknas även Root Mean Square Error (RMSE). Denna beräkning är implementerad i det utvecklade utvärderingssystemet och definieras som

$$RMSE_{xy} = \sqrt{\frac{1}{N} \sum_{i=1}^N \|e_i\|_{xy}^2} \quad (3.8)$$

där N är det totala antalet utvärderade prover. RMSE ger ett sammanfattande mått på den totala lokaliseringssnoggrannheten, där större fel ges högre vikt jämfört med mindre fel.

För att undersöka om systemet uppvisar systematiska fel i någon riktning beräknas även medelfelet separat för x - och y -led. Även dessa beräkningar utförs automatiskt av det implementerade systemet enligt

$$\mu_x = \frac{1}{N} \sum_{i=1}^N e_{x,i} \quad (3.9)$$

$$\mu_y = \frac{1}{N} \sum_{i=1}^N e_{y,i} \quad (3.10)$$

Om något av dessa medelvärden avviker från noll indikerar detta en systematisk förskjutning i den aktuella riktningen.

För att analysera spridningen i positionsfelen kring medelvärdet beräknar systemet även variansen i respektive riktning enligt

$$\sigma_x^2 = \frac{1}{N} \sum_{i=1}^N (e_{x,i} - \mu_x)^2 \quad (3.11)$$

$$\sigma_y^2 = \frac{1}{N} \sum_{i=1}^N (e_{y,i} - \mu_y)^2 \quad (3.12)$$

En låg varians indikerar att systemet ger stabila och konsekventa uppskattningar, medan en hög varians tyder på att positioneringssnoggrannheten varierar mellan olika prover eller positioner.

För att erhålla koordinater från de två metoderna bearbetas den insamlade bilddatan i två steg. Först körs filen *point_reverse.py*, följt av *coordinate_processor.py*. Dessa steg genererar en CSV-fil som innehåller de uppskattade globala koordinaterna från poseuppskattningsmetoden samt marksanningskoordinaterna.

De kvantitativa felmått beräknas därefter automatiskt med hjälp av utvärderingsskriptet *evaluation.py*. Skriptet beräknar bland annat RMSE, medelfel och varians, resultaten sparas i textfiler för vidare analys. För varje bild från respektive kamera genereras även en felkarta (eng. *error map*), vilken visualiserar hur positionsfelet är spatialt fördelat över det utvärderade området. Detta möjliggör identifiering av områden där felet är systematiskt större.

Utöver detta genererar *evaluation.py* histogram för varje kamera. Histogrammen sammanställer felvärden från samtliga placeringar av punktmatta inom samma kamera, vilket möjliggör en direkt jämförelse av hur felets fördelning varierar mellan olika placeringar.

3.3 GulliViews meddelandestruktur

För att en robot ska kunna navigera säkert behöver den inte enbart känna till sin aktuella position utan andra aspekter som tas upp i detta avsnitt. Roboten behöver information

om hur gammal informationen den tar emot är, se Avsnitt 2.9, samt hur tillförlitligt informationen roboten tar emot är m.m. Detta ställer krav på vad Gulliview publicerar till Topics och vad de ska innehålla samt utformat efter det behov roboten har. I de kommande delavsnitt diskuteras de metoder och designval som används i den fullständiga meddelandestrukturen.

3.3.1 Hantering av informationsålder

Eftersom Gulliview är byggd utefter att funka med distribuerade system och kommunicera med flera system på samma ROS nätverk kan det uppstå en klockdrift mellan systemen när de ska dela tidstämplar. För att motverka avvikelse mellan de synkroniserade klockorna har ett Network Time Protocol (NTP) implementeras, se Avsnitt 2.8. Robotar som tar emot Gulliviews meddelande ska kunna ha samma uppfattning om vad tiden är i en specifik punkt i tiden. Genom att kombinera och spara tidstämplar vid olika vitala delar under skapandet av meddelandet så kan roboterna få en uppskattning hur gammal informationen är och avgöra informationens användbarhet i realtid.

$$time_offset = \frac{(T_2 - T_1) + (T_3 - T_4)}{2} \quad (3.13)$$

där

T_1 : tidstämpel då roboten skickar ACK

T_2 : tidstämpel då Gulliview mottar ACK

T_3 : tidstämpel då Gulliview skickar meddelandet

T_4 : tidstämpel då roboten mottar meddelandet

Ekvationen ovan beräknar hur mycket klockorna på roboten avviker från Gulliviews klocka. Detta värde `time_offset` ger en uppskattning om tiden som skiljer mellan Gulliview och roboten som läser från meddelande på dess Topic. Är värdet på `time_offset` positivt så är robotens klocka före Gulliview, annars är robotens klocka efter. Med andra ord, värdet på `time_offset` är hur tiden varierar jämfört med Gulliview, på detta ROS nätverk antar vi att Gulliview har den rätta tiden och roboten ska ändra sin tid utefter den.

3.3.2 Filtrering och stabilisering med EWMA

För att hantera variationer i ROS nätverket, till exempel om ett meddelande skulle ta lite längre tid på sig eller att ett meddelande tappas används exponentiellt viktat glidande medelvärde (EWMA) se Avsnitt 2.7. Denna metod appliceras på flera variabler i meddelandet, så som RTT och publiceringsfrekvens. Tre variabler skickas med, nuvarande värdet, medianvärdet och variationsvärdet. Detta görs för att en mottagande robot ska få en så bra uppfattning om vad det nuvarande värdet är och dess korrekthet.

$$mean = \alpha \times sample + (1 - \alpha) \times previous_mean \quad (3.14)$$

$$variance = \beta \times (sample - mean)^2 + (1 - \beta) \times previous_variance \quad (3.15)$$

Variabeln `sample` skiljer sig åt beroende på vilken typ av information som mäts och dessa är:

$$\begin{aligned}
coordinate_error &= \sqrt{dx^2 + dy^2} \\
RTT &= time_now - message_sent \\
time_offset &= \frac{robot_receive_time - gulliview_message_time + robot_send_time - time_now}{2} \\
success_rate &= \frac{bits_received}{total_possible_bits} \\
publish_rate &= \frac{1.0}{gulliview_message_time - previous_gulliview_message_time}
\end{aligned}$$

Anledningen till att meddelandet innehåller EWMA är för att vissa parametrar som mäter meddelandes stabilitet kan variera kraftigt över tid. Genom att använda äldre värden kan roboten få en uppskattning om vad värdet kan vara, vilket gör den responsiv mot stora förändringar i värden.

3.3.3 Unik identifiering av meddelande

Meddelandet som Gulliview skickar ska innehålla ett sekvensnummer av flera anledningar. Ett av dessa är för att kunna organisera och felsöka informationen som skickas. Sekvensnummer ska även skapa en dynamiska feluppskattningar inom systemet, läs mer om detta i Avsnitt 3.4.1.

I meddelandet ska även det specifika tag_id som informationen tillhör deklarerar. Detta gör att roboten vet vem informationen handlar om. Då alla robotar ska få tillgång till alla meddelanden så ska roboten själv kunna avgöra om informationen är relevant att läsa.

3.3.4 Uppskattning av nästa position

Utifrån den information som Gulliview publicerar, om hur ofta uppdateringar sker, hastighet, acceleration och orientation med mera, kan Gulliview skapa en kvalificerad gissning vart roboten kan befinna sig i nästa uppdatering. Detta förmedlas till roboten så att den kan förbereda sig för ny information. Det Gulliview rapporterar blir sällan helt korrekt. Därför genomförs en utvärdering av hur bra uppskattningen av robotens position var och rapporterar hur mycket positionsvärdet kan avvika i ett nytt meddelande (se Avsnitt 3.3.2).

3.3.5 Rumslig osäkerhet och säkerhetsmarginaler

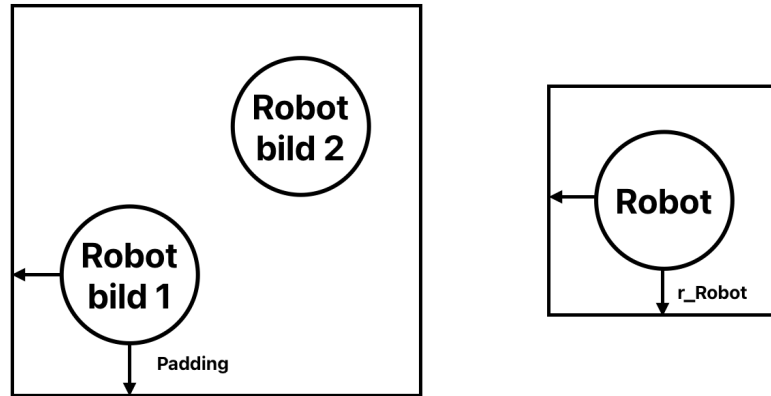
För att hantera osäkerhet i Gulliviews positionsbestämning så ska en säkerhetsmarginal användas. Denna säkerhetsmarginal beräknas utifrån prediktionsfelet från nästa position jämfört med den rätta positionen i nästkommande meddelande, och kan beräknas med följande:

$$m = \mu_e + k * \sqrt{\sigma_e^2}. \quad (3.16)$$

Där μ_e är den estimerade exponentiellt viktade felet och σ_e^2 är den estimerade variansen, se Avsnitt 2.7. Värdet k är en skalningsfaktor som används för att bestämma hur viktig variansen ska vara för säkerhetsmarginalen och där med kommer m automatiskt skalas när osäkerheten av Gulliview blir större. Med säkerhetsmarginalen kan en samlad utfyllnad definieras, som täcker varje robot:

$$padding = r_{Robot} + m \quad (3.17)$$

där r_{Robot} är den uppmätta fysiska längden av roboten från mitten till dess längsta sida. r_{Robot} används för att mäta ut storleken på hela roboten i den nuvarande bilden. Padding-värdet används istället som en storlek för att mäta positionssäkerhet till avgränsningsboxen mellan två tagna bilder, där den fysiska storleken och säkerhetsmarginalen är medräknad.



Figur 3.2: Visualisering om avgränsningsbox mellan bilder (till vänster) och avgränsningsbox av nuvarande bild (till höger)

Avgränsningsboxen är menad för att visa det område som roboten befinner sig i den nuvarande bilden. Utöver avgränsningsbox för den nuvarande bilden definieras ytterligare en avgränsningsbox mellan två tagna bilder. Då Gulliview har en uppskattning på var roboten ska befinna sig vid nästa uppdatering (se Avsnitt 3.3.4) skapas en större avgränsningsbox om var roboten kan befinna sig mellan två tagna bilder. För att garantera att roboten i bild 1 med koordinater (x_1, y_1) och samma robot i en senare bild, bild 2, med koordinater (x_2, y_2) befinner sig inom avgränsningsboxen, används min- och maxvärden för de två positionerna.

Algoritm 2 Bygga två avgränsningsboxar

- 1: $\text{padding} \leftarrow \text{radius}_{\text{robot}} + m$
 - 2:
 - 3: $\triangleright$ *Avgränsningsbox centrerad kring aktuell position*
 - 4: $x_1 \leftarrow p_x - \text{radius}_{\text{robot}}$
 - 5: $y_1 \leftarrow p_y - \text{radius}_{\text{robot}}$
 - 6: $x_2 \leftarrow p_x + \text{radius}_{\text{robot}}$
 - 7: $y_2 \leftarrow p_y + \text{radius}_{\text{robot}}$
 - 8: $\text{object_box} \leftarrow (x_1, y_1, x_2, y_2)$
 - 9:
 - 10: $\triangleright$ *Avgränsningsbox som täcker intervallet mellan nuvarande och nästa position*
 - 11: $x_1 \leftarrow \min(p_x, p_{\text{next},x}) - \text{padding}$
 - 12: $y_1 \leftarrow \min(p_y, p_{\text{next},y}) - \text{padding}$
 - 13: $x_2 \leftarrow \max(p_x, p_{\text{next},x}) + \text{padding}$
 - 14: $y_2 \leftarrow \max(p_y, p_{\text{next},y}) + \text{padding}$
 - 15: $\text{motion_box} \leftarrow (x_1, y_1, x_2, y_2)$
-

3.3.6 Nätverksinformation och framtida uppdateringar

För att roboten ska veta när nästa meddelande ska skapas av GulliView och när nästa uppdatering anländer behövs viss information.

Det måste finnas en uppskattning om hur lång tid det tar från att ett meddelande skickas tills dess att GulliView får ett ack. Denna tiden kallas för tur-och-retur-tid (RTT) och för enkelhetens skull ska det antas att från att ett meddelande skickas så är det $RTT/2$ tills dess att meddelandet anländer till roboten.

För att veta när nästa troliga uppdatering skapas så används två olika variabler. Uppdateringshastigheten som säger hur många meddelanden som skapas, i kombination med procenten lyckade meddelanden, det vill säga antalet meddelanden som får ett ack tillbaka jämfört med det totala antalet skickade meddelanden. Genom att multiplicera dessa variabler skapas den faktiska lyckade uppdateringshastigheten, vilket ska användas för att få fram när nästa troliga uppdatering kommer att skapas.

Denna tidsuppskattning ska även användas i andra beräkningar så som uppskattning av nästa position.

3.3.7 Verifiering av meddelandets innehåll

För att informationen GulliView publicerar till Topics ska kunna säkerställas vara korrekta, ska också ett bitfält med meddelandet publiceras. Syftet med detta bitfält är att verifiera att resterande variabler i meddelandet innehåller korrekt och nödvändig information.

Detta bitfält består utav n bitars fält, där varje bit i meddelandet representerar en variabel i informationen. Är informationen korrekt så ska den biten som representerar den variabeln sättas till 1. Skulle det exempelvis GulliView skapa en RTT som är negativ eller success rate som är över 100% kommer den representerande biten sättas till 0 för att indikera att roboten inte ska använda just det värdet, alla förutsättningar kan ses i Appnedix C. Slutligen finns det en dedikerat bit för alla variabler. Sätts denna biten till 1 så kan roboten anta att all information i meddelandet är korrekt, om den är 0 så indikerar den att någon information i meddelandet är felaktigt.

Anledningen till denna design är att möjliggöra selektiv användning av information i ett system där tillförlitlig leverans inte kan garanteras. I stället för att kassera hela meddelandet ges roboten möjlighet att använda de delar av informationen som fortfarande anses som tillförlitliga.

3.4 Robotens ACK-mekanism

För att kunna uppskatta hur många meddelande som når roboten måste GulliView på något sätt få information om vilka meddelanden som har tagits emot. Om kommunikationen hade skett över ROSTCP hade detta inte varit något problem, eftersom tillförlitlig leverans då hanteras av transportprotokollet. På grund av höga krav på prestanda valdes dock i stället ROSUDP som kommunikationsprotokoll. Detta innebär att tillförlitligheten måste hanteras på applikationsnivå, och att kommunikationsprotokollet inte kan garantera att meddelanden levereras till mottagaren eller i rätt ordning. Ett eget protokoll behöver därför implementeras för att avgöra vilka meddelanden som har nått fram respektive gått förlorade. Utöver hantering utav mottagna meddelanden så innehåller denna del tidstämplat och unika identifierare som används för att få information om ROS nätverkets prestanda.

3.4.1 Unik identifiering av ack

Roboten ska kopiera det sekvensnummer som existerar i meddelandet. På detta sätt säkerställs att informationen som jämförs är relevant. Utan sekvensnummer vet inte datorn vilket ack som ska jämföras med vilket meddelande. Detta är viktigt då den jämförda informationen ska skapa en dynamisk felmarginal både inom uppskattning för framtida positioner av roboten men även synkronisering av lokala klockor.

3.4.2 Kompakt kvittering med bitfält

Roboten skickar tillbaka så kallade ack-meddelanden. Varje sådant meddelande innehåller dels sekvensnumret för det senast mottagna meddelandet, dels ett 32-bitarsfält som anger huruvida de 32 föregående meddelandena har tagits emot eller inte. Med denna lösning gör det inget om ett enskilt ack-meddelande skulle gå förlorat, eftersom efterföljande ack-meddelanden fortfarande innehåller information om tidigare mottagna meddelanden. Algoritm 3 visar pseudokod för hur ett ack-meddelande skapas.

Algoritm 3 Bygga ett ack bitfält

```

1: received_acks                                ▷ array of received packet sequence numbers
2: ack                                           ▷ latest received sequence number
3: ack_bitfield ← 0
4: for  $i = 0$  to 31 do
5:   if received_acks[ack -  $i$  - 1] then
6:     set bit  $i$  in ack_bitfield to 1
7:   else
8:     set bit  $i$  in ack_bitfield to 0
9:   end if
10: end for
11: send (ack, ack_bitfield) to GulliView

```

3.4.3 Tidstämplar för nätverksanalys

För att möjliggöra att GulliView ska kunna analysera ROS nätverkets prestanda så skickas flera tidstämplar med varje ACK-meddelande. Dessa tidstämplar används för att beräkna tur-och-retur-tid (RTT) samt avvikelser i de lokala klockorna, dessa tidstämplar används i Avsnitt 3.3.1 samt Avsnitt 3.3.2. Denna information är vital för att förstå ROS nätverkets beteende och beräkningar hos GulliView.

4

Evaluering

Detta kapitel presenterar utvärderingen av Gulliview med fokus på positioneringsnoggrannhet, geometriska avvikelser, informationsålder, predikterad information och säkerhetsmarginaler. Utvärderingen utgår från definierade forskningsfrågor och evalueringskriterier, följt av en beskrivning av testmiljön och de experiment som används för att analysera systemets prestanda.

4.1 Forskningsfrågor

F-1 *Hur noggrant är Gulliview vid positionering i inomhusmiljöer?*

F-1.1 *Hur påverkar kameradistorsionen representationen av den fysiska miljön och dess struktur i bilderna?*

F-1.2 *Hur stort är positioneringsfelet vid användning av poseuppskattningsmetoden?*

F-1.3 *Hur varierar positioneringsfelet över testytan, finns det specifika områden där felet är systematiskt större?*

F-2 *Hur kan mekanismer designas och implementeras för att mäta prestandan av Gulliview?*

F-2.1 *Vad är medelvärdet och variansen för åldern av informationen som observeras av Roboten?*

F-2.2 *Hur stort är felet på Gulliviews estimerad information med avseende till tid och position?*

F-2.3 *Hur stor är sannolikheten att Roboten befinner sig innanför den uppmätta avgränsningsboxen med avseende till skalningsfaktorn i säkerhetsmarginalen?*

4.2 Evalueringskriterier

Detta avsnitt beskriver de kriterier som används för att bedöma resultaten från utvärderingen. Kriterierna är kopplade till forskningsfrågorna och omfattar både positionsrelaterade mått, såsom RMSE och medelfel, mått för geometrisk noggrannhet, informationsålder, predikterad information och avgränsningsboxens säkerhetsmarginal. Syftet är att tydliggöra vilka mätvärden som används för att analysera systemets prestanda och hur dessa

relaterar till respektive forskningsfråga.

4.2.1 Geometrisk noggrannhet

Punktmattans rutnät är i verkligheten perfekt ortogonalt, vilket innebär att varje hörn bildar exakt 90° . Avvikelse från 90° efter distorsionskorrigering och perspektivtransform används som mått på kvarvarande distorsion, enligt metoden beskriven i avsnitt 3.1.

Kriterium	Motivering	Forskningsfråga
Medelvärde vinkelavvikelse	Anger den genomsnittliga avvikelsen från 90° över samtliga punkter i rutnätet.	F-1.1
Varians vinkelavvikelse	Anger hur mycket avvikelsen varierar över synfältet, vilket indikerar om distorsionen är jämnt fördelad eller lokalt koncentrerad.	F-1.1
Spatial fördelning	Spridningsdiagram som visar var i synfältet avvikelserna är störst, för att identifiera systematiska geometriska fel.	F-1.1

Tabell 4.1: Evalueringskriterier för geometrisk noggrannhet.

4.2.2 Positioneringssfel

För att utvärdera lokaliseringssfelet används RMSE samt medelfel i x - och y -led. RMSE används för att besvara F-1.2, eftersom måttet sammanfattar hur stort felet är för de poseuppskattade koordinaterna i förhållande till marksanningskoordinaterna. Medelfel i x - och y -led används för att identifiera om det finns en systematisk förskjutning i någon riktning.

Kriterierna används även för att besvara F-1.3 genom att jämföra resultaten över olika delar av testytan. På så sätt kan det undersökas om lokaliseringssfelet varierar spatialt eller om vissa områden uppvisar större avvikelser.

Kriterium	Motivering	Forskningsfråga
RMSE	Sammanfattar det totala lokaliseringssfelet mot marksanningskoordinaterna.	F-1.2, F-1.3
Medelfel i x -led	Visar eventuell systematisk avvikelse i horisontell riktning.	F-1.2, F-1.3
Medelfel i y -led	Visar eventuell systematisk avvikelse i vertikal riktning.	F-1.2, F-1.3

Tabell 4.2: Evalueringskriterier för lokaliseringssfel över testytan.

4.2.3 Informationsålder

För att utvärdera informationsåldern används dess medelvärde, varians och värdet på informationsåldern över experimentets genomförande. Dessa kriterier kopplas till forskningsfråga F-2.1 där medelvärdet visar den genomsnittliga ålder över experimentet och

variansen visar hur mycket informationsålder varierar över tid. De sanna värdena är till för att se ifall de finns någon tydlig korrelation till varför medelvärdet eller variansen ger just de värdena de visar.

Resultaten analyseras numeriskt i relation till systemets uppdateringsfrekvens. På så sätt kan informationsålderns värden och stabilitet bedömas utan att använda kvalitativa etiketter som exempelvis bra eller otillräckligt.

Kriterium	Motivering	Forskningsfråga
Medelvärde AoI	Anger den genomsnittliga åldern på informationen som roboten tar emot.	F-2.1
Varians AoI	Anger hur mycket informationsåldern varierar över tid.	F-2.1

Tabell 4.3: Evalueringskriterier för informationsålder.

4.2.4 Estimerad information

För att utvärdera GulliViews estimerade information används positionsfel och tidsfel. Positionsfelet mäter skillnaden mellan GulliViews estimerade nästa position och robotens sanna uppmätta position i nästa bild. Tidsfelet mäter skillnaden mellan den estimerade tiden för nästa bild och den faktiska tiden då nästa bild skapas.

Dessa kriterier kopplas direkt till F-2.2, eftersom forskningsfrågan behandlar felet i GulliViews estimerade information med avseende på både position och tid. Resultaten redovisas som numeriska värden för att kunna analysera prediktionens noggrannhet.

Kriterium	Motivering	Forskningsfråga
Positionsfel	Mäter skillnaden mellan estimerad nästa position och uppmätt position i nästa bild.	F-2.2
Tidsfel	Mäter skillnaden mellan estimerad tid för nästa bild och faktisk tid då nästa bild genereras.	F-2.2

Tabell 4.4: Evalueringskriterier estimerad information.

4.2.5 Skalningsfaktor för avgränsningsbox

För att utvärdera skalningsfaktorn för avgränsningsboxen används andelen prover där roboten befinner sig utanför den beräknade avgränsningsboxen. Detta kriterium visar hur sannolikheten att roboten hamnar innanför säkerhetsmarginalen förändras när skalningsfaktorn varierar.

Kriteriet kopplas direkt till F-2.3, eftersom forskningsfrågan handlar om sannolikheten att roboten befinner sig innanför avgränsningsboxen med avseende på skalningsfaktorn. Resultaten redovisas numeriskt för varje värde på en skalningsfaktor, vilket gör det möjligt att jämföra hur säkerhetsmarginalen påverkas av olika skalningsfaktorer.

Kriterium	Motivering	Forskningsfråga
Andel innanför avgränsningsbox	Mäter hur ofta roboten befinner sig innanför den beräknade avgränsningsboxen för en given skalningsfaktor.	F-2.3

Tabell 4.5: Evalueringskriterier skalningsfaktor för avgränsningsbox.

4.3 Evalueringsmiljö

Utvärderingen genomförs i labbsal ED5355 på Chalmers Tekniska Högskola. Testmiljön består av en testbädd med ett vägnätverk och fyra takmonterade Elgato Facecam Pro-kameror som täcker hela testytan. Utvärderingen utförs med hjälp av två separata kod-baser: *Gulliview Noetic* och *Gulliview Matei* för mer information av dessa system se Avsnitt 2.5.

Följande specifikationer används under samtliga experiment:

Komponent	Specifikation
Kameror	4× Elgato Facecam Pro mk.2
Upplösning	4K (3840×2160)
Operativsystem	Ubuntu 20.04.6 LTS (Focal Fossa)
CPU	Intel® Core™ Ultra 7 155H
RAM	32GB DDR5 3200MHz
ROS-version	ROS Noetic
Referensobjekt	Punktmatta och AprilTag
Robot	WiFiBot

Punktmattan placeras plant på testbäddens golv inom kamerornas synfält. Datainsamling sker under kontrollerade förhållanden med konstant belysning och utan störande objekt i synfältet.

4.4 Experiment

Detta avsnitt beskriver de experimenten som genomförs för att besvara forskningsfrågorna och utvärdera GulliViews prestanda. Experimenten omfattar både statiska tester, där punktmattan används för att analysera positioneringssfel och geometriska avvikelser, samt dynamiska tester med roboten, där informationsålder, predikterad information och avgränsningsboxens säkerhetsmarginal undersöks. För varje experiment beskrivs syfte, genomförande och vilka delar av systemet som utvärderas.

4.4.1 Experiment 1: Lokaliseringsfel över testytan

Syftet med experimentet är att undersöka hur stor positioneringssfelet blir vid användning av poseuppskattning vid olika placeringar av punktmattan. Experimentet är utformat för att besvara forskningsfrågorna:

F-1.2: Hur stort är positioneringssfelet vid användning av poseuppskattningsmetoden?

och

F-1.3: Hur varierar positioneringssfelet över testytan, finns det specifika områden där felet är systematiskt större?

Punktmattan placeras på tre positioner över testytan, och för varje placering jämförs de uppskattade koordinaterna med marksanningskoordinaterna. I experimentet varierar punktmattans horisontella placering, definierad som avståndet från origo till mattans vänsterkant. De tre placeringarna var 0.1 m, 1.485 m och 2.87 m. För varje placering samlas en bild in per kamera, vilket ger ett mättillfälle för varje kombination av kamera och positionen på punktmattan.

Hypotesen är att de poseuppskattade koordinaterna förväntas avvika från marksanningen med ett fel i storleksordningen några centimeter. Detta motiveras av att båda metoderna utgår från samma testyta och kalibrerade kamerabilder, vilket bör ge små skillnader mellan koordinatsystemen.

- **Genomförande:**

1. Punktmattan placeras på den valda positionen på testytan.
2. Bilder för denna placering samlas in.
3. Poseuppskattade koordinaterna och marksanningskoordinaterna beräknas för bilderna.
4. Poseuppskattade koordinaterna jämförs med marksanningskoordinaterna.
5. Steg 1–4 upprepas för samtliga valda placeringar av punktmattan.

4.4.2 Experiment 2: Distorsionsfel och representation av fysiska miljön i bilder

För att undersöka hur distorsionen i kamerorna påverkar representationen av den verkliga miljön används metoden beskriven i Avsnitt 3.1. Denna metod är baserad på att räkna vinklarna mellan punkterna på punktmattan från de tagna bilderna. I en ideal värld är även punkterna i de tagna bilderna ortogonala mot varandra, vilket innebär att avvikelser från det kan tolkas som kvarvarande distorsionsfel.

Denna utvärdering syftar på att undersöka hur mycket av det korrekta geometriska strukturen som är bevarad efter bildbehandlingen samt ifall det finns lokala avvikelser. För denna utvärdering används följande forskningsfråga:

F-1.3 Hur påverkar kamera distorsionen representationen av den fysiska miljön och dess struktur i bilderna?

Utvärderingen baseras på en uppsättning punktdata från samtliga fyra kameror, där inga externa variabler varierar. Hypotesen är att majoriteten av de uppmätta vinklarna förväntas ligga nära 90° med små avvikelser efter bildbehandlingen, eftersom distorsionskorrigering och perspektivtransformation tillämpas innan analysen.

- **Genomförande:**

1. Punktkoordinater laddas från CSV filer genererade vid detektionen av punkterna
2. För varje punkt i rutnätet identifieras närliggande punkt i horisontell och vertikal riktning.

3. Vektorer skapas mellan dessa punkter
4. Vinkeln räknas ut med hjälp av skalärprodukten
5. För samtliga punkter så sammanställs vinkelvärdena och statistiska värden så som median, varians medelvärde, standardavvikelse och minimum och maximum värden räknas ut.
6. Resultaten från samtliga kameror sammanställs i laddiagram för enkel jämförelse mellan kamerorna

4.4.3 Experiment 3: Uppmätt informationsålder

Meddelanden som GulliView publicerar innehåller bland annat tidstämplor. Det är en viktig del då alla delar av GulliView använder sig av lokala klockor. Då klockorna inte är synkroniserade skickas tidstämplor för att jämför hur mycket som skiljer och räkna med detta i uppskattningar för att minimera felmarginaler.

Bilderna från kamerorna behöver gå igenom många steg, såsom analysering, skapandet av meddelandet samt publicering för att nämna några. Detta gör att informationen som roboten tar emot alltid kommer vara en representation av dåtiden. På grund av detta är det viktigt att ha koll på åldern av informationen då det påverkar hur relevant den är (se Avsnitt 3.3.1). Experimentet är därför utformat för att besvara forskningsfrågan:

F-2.1: Vad är medelvärdet och variansen för åldern av informationen som observeras av roboten?

I detta experiment varierades inga externa eller konfigurerbara variabler. Utvärderingen baserades i stället på 5000 insamlade prover av den behövda informationen, vilket gav ett underlag för att beräkna ålder av information och tidstämplornas offset. Hypotesen för detta experiment är att informationsåldern ska vara stabil under hela testet. Detta eftersom att experimentet genomförs utan förändrade variabler och att varje prov går igenom samma steg, från bildinsamling och analys till publicering av meddelandet. Därför förväntas informationsåldern vara relativt stabil.

• Genomförande:

1. Sätt parametrar i datainsamlingsprogrammet:
 - (a) Antalet prover (`MAX_SAMPLES`) sätts till 5000.
 - (b) Antalet mätta parametrar (`NUM_METRICS`) sätts till 4.
 - (c) Specificera mätta parametrar, `recv_time`, `frame_time`, `time_offset`, `publish_rate`.
2. Ladda över källkod för att kunna läsa och publicera till Topics från Roboten med minimal källkod för att kunna kommunicera med GulliView.
3. Starta GulliView systemet.
4. Placera Robot inom GulliViews synfält för att den ska detekteras.
5. Starta datainsamlingsprogrammet med satta parametrar på fristående system på nätverket.

4.4.4 Experiment 4: Estimerad information

Meddelandet innehåller GullViews estimering för var roboten kommer vara i nästa bild samt när nästa bild kommer att skapas (se Avsnitt 3.3.4). Detta är en säkerhetsaspekt då utan denna information har roboten ingen chans att planera sitt körande, vilket kan leda till farliga situationer såsom kollision. Denna estimerade information jämförs med nästa

bild för att utvärdera hur nära sanningen den är. Experimentet är utformat för att besvara forskningsfrågan

F-2.2: Hur stort är felet på GulliViews estimerade information med avseende på tid och position?

Utvärderingen baseras på 5 000 insamlade prover, där GulliViews prediktion jämförs med motsvarande sanna informationen från nästa bild. Vi kan sedan skapa hypotesen att GulliViews estimerade nästa position och bildtid avviker i en normalkurva från motsvarande uppmätta värden i nästa bild. Detta eftersom roboten kör med varierande hastigheter, vilket gör den nästkommande positionen och tidpunkten ändras med olika stor marginal.

- **Genomförande:**

1. Sätt parametrar i datainsamlingsprogrammet:
 - (a) Antalet prover (MAX_SAMPLES) sätts till 5000.
 - (b) Antalet mätta parametrar (NUM_METRICS) sätts till 5.
 - (c) Specificera mätta parametrar, next_coordinate_error.sample, frame_time, next_frame_time, recive_time, next_recive_time.
2. Ladda över källkod för att kunna läsa och publicera till Topics från roboten med minimal källkod för att kunna kommunicera med GulliView.
3. Starta GulliView systemet.
4. Placera roboten inom GulliViews synfält för att den ska detekteras.
5. När roboten detekteras av GulliView, starta programkod på roboten.
6. Kör roboten med varierande hastighet och i en icke bestämt bana.
7. Starta datainsamlingsprogrammet med satta parametrar på fristående system på nätverket.

4.4.5 Experiment 5: Skalningsfaktor för avgränsningsbox

Meddelandet innehåller koordinater för en avgränsningsbox (se Avsnitt 3.3.5) där roboten förutsätts befinna sig fram till nästa meddelande, likt en säkerhetszon som ej bör beträdas. Denna avgränsningsbox räknas ut genom att ta nuvarande position till estimerade nästa position och addera den estimerade felmarginalen som multipliceras med en skalningsfaktor.

*Avgränsningsbox = Estimerad_position - Nuvarande_position + Estimerat_felmarginal * Skalningsfaktor*

Genom att sätta skalningsfaktorn till ett stort tal kan det garantera att roboten kör säkert. Detta blir däremot problematiskt om flera robotar ska använda testbädden samtidigt, eftersom en för stor avgränsningsbox begränsar det tillgängliga utrymmet. För att undersöka hur skalningsfaktorn påverkar säkerhetsmarginalen är experimentet utformat för att besvara forskningsfrågan:

F-2.3 Hur stor är sannolikheten att Roboten kan befinna sig innanför den uppmätta avgränsningsboxen med avseende till skalningsfaktorn i säkerhetsmarginalen?

I detta experiment varierar skalningsfaktorn $m \in \{0,0; 0,1; 0,2; 0,3; 0,4; 0,5; 0,6; 0,7; 0,8; 0,9; 1,0\}$. Utvärderingen baseras på 5 000 prover för varje värde på m . Hypotesen är att för en större

skalningsfaktor minskar risken för att roboten hamnar utanför avgränsningsboxen, eftersom säkerhetsmarginalen blir större.

- **Genomförande:**

1. Sätt parametrar i datainsamlingsprogrammet:
 - (a) Antalet prover (`MAX_SAMPLES`) sätts till 5000.
 - (b) Antalet mätta parametrar (`NUM_METRICS`) sätts till 2.
 - (c) Specificera mätta parametrar `object_bounding_box`, `between_bounding_box`.
2. Ladda över källkod för att kunna läsa och publicera till Topics från roboten med minimal källkod för att kunna kommunicera med Gulliview.
3. Sätt variabeln `multiplier_safety_margin` till den valda skalningsfaktorn, bygg och starta Gulliview systemet.
4. Placera roboten inom Gulliviews synfält för att den ska detekteras.
5. När roboten detekteras av Gulliview, starta programkod på roboten.
6. Kör roboten med varierande hastighet och i en icke bestämd bana.
7. Starta datainsamlingsprogrammet med satta parametrar på fristående system på nätverket.
8. Upprepa steg 3-7 för varje element i m .

5

Resultat

Detta kapitel redovisar resultaten från projektets genomförda experiment. Resultaten presenteras i relation till forskningsfrågorna. Syftet är att ge en samlad bild av systemets prestanda utifrån de evalueringskriterier som definierades i föregående kapitel.

5.1 Resultat av experiment 1

Resultaten av experimentet (se Avsnitt 4.4.1) för positioneringsnoggrannheten i globala koordinater presenteras i relation till forskningsfråga F-1.2 och F-1.3, där RMSE-felen för de olika placeringarna av punktmattan presenteras i Figur 5.1. I figuren varierar felen mellan kamerorna i relativt stor utsträckning, där exempelvis kamera 0 uppvisar en RMSE på 0.236 m på vänster sida medan kamera 3 har ett motsvarande värde på 0.087 m, vilket innebär att vissa mätvärden är mer avvikande än andra. Vidare kan det observeras att RMSE-värdena generellt är lägre för kameror med högre kamera-id.

Kamera-id	Vänster [m]	Mitten [m]	Höger [m]
0	0.236	0.205	0.298
1	0.239	0.190	0.159
2	0.169	0.140	0.135
3	0.087	0.039	0.078

Tabell 5.1: RMSE på xy

I Tabell 5.2 presenteras medelfelen i position i riktningarna (x,y) för respektive kamera och placering (vänster, mitten och höger). Från tabellen framgår det att medelfelen varierar både mellan kameror och mellan olika positioner inom samma kamera. För kamera 0 observeras exempelvis negativa värden i både vänster och mittenpositionerna, medan högerpositionen uppvisar ett positivt värde i x-led. Kamera 1 och kamera 2 visar genomgående större positiva avvikelser i vissa positioner, särskilt i y-led, där exempelvis kamera 1 har ett värde på 0.154 m i vänsterpositionen och kamera 2 når upp till -0.136 m. Samtidigt uppvisar kamera 3 generellt mindre avvikelser i både x- och y-led jämfört med de övriga kamerorna, även om vissa enskilda värden fortfarande avviker från noll. Sammantaget visar tabellen att medelfelen varierar i både storlek och riktning beroende på kamera och position.

Kamera-id	Vänster [m]	Mitten [m]	Höger [m]
0	(0.144, 0.098)	(0.194, 0.015)	(0.272, -0.047)
1	(0.171, 0.154)	(0.114, 0.147)	(0.052, 0.144)
2	(0.079, -0.136)	(0.040, -0.122)	(-0.020, -0.110)
3	(0.028, -0.007)	(0.027, -0.003)	(-0.002, 0.0186)

Tabell 5.2: Medelfel på (x, y)

Figur 5.1 visar RMSE för det euklidiska avståndsfelet i två dimensioner är uppdelat i intervall längs x-axeln för kamera 2. Staplarna visar det beräknade RMSE-värdet för varje x-intervall, medan den röda kurvan visar en anpassad trendkurva. I figuren kan RMSE-värdena avläsas för de olika x-intervallen. Värdena är något större i början av x-området jämfört med mitten, och en lokal ökning syns även nära slutet av x-området. y-axeln är fixerad till 0.6 m, vilket gör att figuren kan jämföras med motsvarande RMSE-plottar för andra kameror med samma skala. För tabeller på RMSE för samtliga kameror, se Appendix D

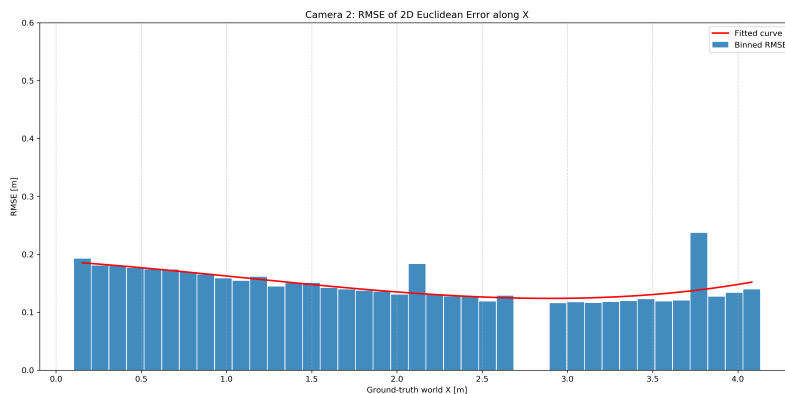
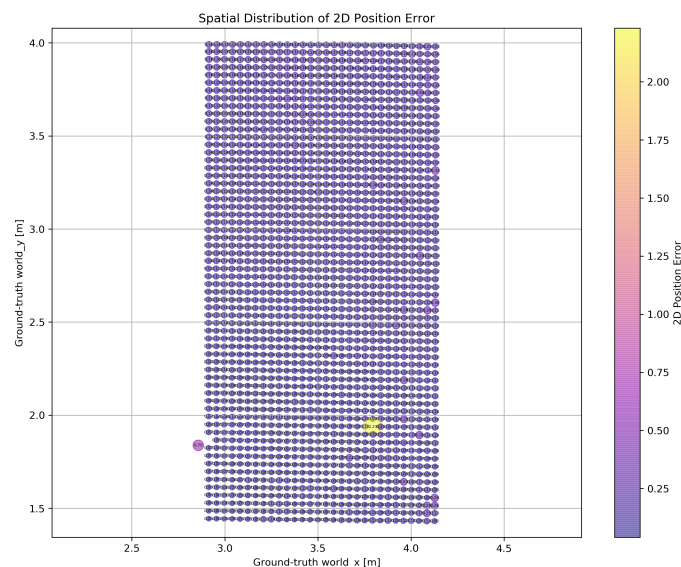


Figure 5.1: RMSE av 2D euklidiska avståndsfelet över x-axel för kamera 2

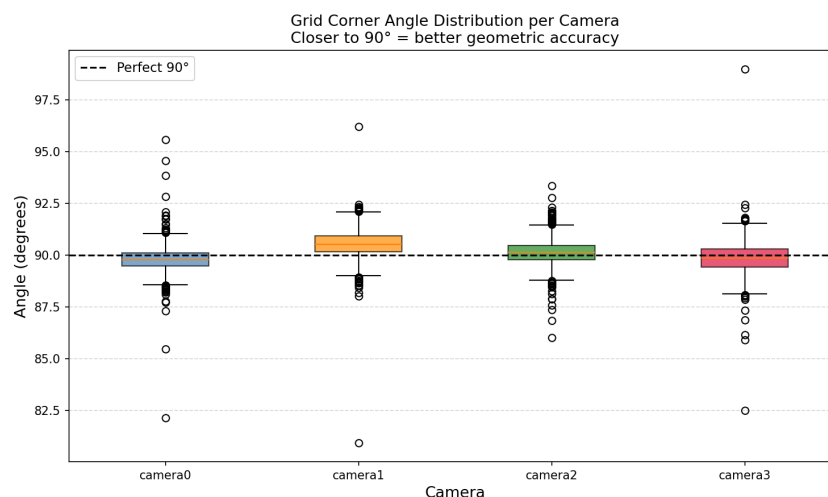
Figur 5.2 visar den rumsliga fördelningen av 2D-positionsfelet över det utvärderade punktmönstret. Varje markör representerar en detekterad punkt uttryckt i marksanningskoordinater, där både färg och markörstorlek motsvarar storleken på det euklidiska positionsfelet. Mörkare/lila markörer indikerar lägre fel, medan ljusare/gulne markörer visar högre felvärden.



Figur 5.2: Rumslig fördelning av 2D euklidiska avståndsfelet över det utvärderade punktmönstret för kamera 2 högre sidan av testbadden

5.2 Resultat av experiment 2

Resultaten från experimentet för vinkelanalys visar hur väl den geometriska strukturen från punktmattan har bevarats efter bildbehandlingen. Låddiagrammet 5.3 illustrerar fördelningen av de uppmätta vinklarna per kamera. I ett perfekt system är alla vinklar exakt 90° . Kamerorna 0, 2 och 3 ligger tätt kring 90° , medan kamera 1 uppvisar en systematisk förskjutning uppåt.



Figur 5.3: Låddiagram på uppmätta vinklar över alla fyra kameror

Tabell 5.3 visar en sammanställning av de statistiska värdena, medelvärde, medianvärde samt varians för varje kamera. För samtliga kameror så ligger medianen och medelvärdet nära det ideala värdet 90° . Variansen i tabellen ger en beskrivning på hur spridning på mätningarna och hur vinklarna varierar kring medelvärdet. Lägre varians ger en indikation på mer konsekventa mätningar medan en högre ger en indikation på en större spridning.

Kamera-id	Medelvärde [°]	Medianvärde [°]	Varians [° ²]
0	89.7935	89.7879	0.3698
1	90.5270	90.5297	0.4432
2	90.1228	90.1226	0.3523
3	89.8759	89.8434	0.5458

Tabell 5.3: Statistiska värden över uppmätta vinklar från alla fyra kameror

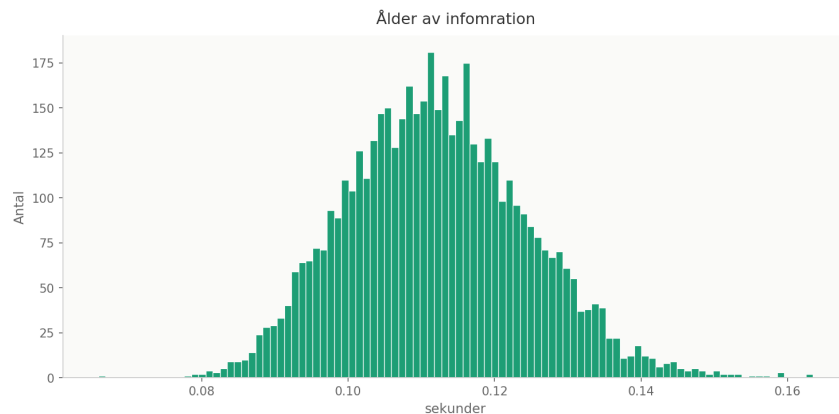
5.3 Resultat av experiment 3

Resultaten av experimentet för informationens ålder används i relation till forskningsfråga F-2.1 som efterfrågar medelvärdet samt variansen av informationsåldern. Figur 5.4 visar en normalfördelad kurva för åldern av informationen. Utifrån denna data beräknas värdena.

Medelvärdet av ålder på informationen blev: 0.11215 s.

Variansen av ålder på informationen blev: 0.00015 s².

En låg varians innebär att resultaten är stabila och jämna eftersom skillnaden mellan mätningarna är liten.



Figur 5.4: Sammanställt stapeldiagram för representation av informationens ålder

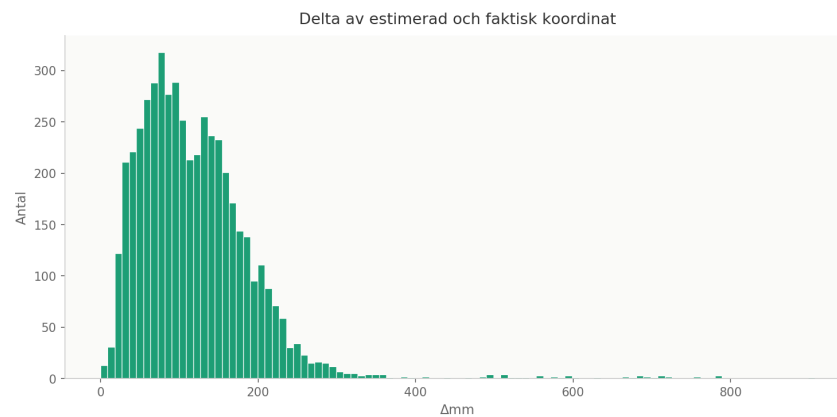
5.4 Resultat av experiment 4

Resultaten av experimentet för estimerad information används i relation till forskningsfråga F-2.2 som frågar hur noggrann är den estimerade informationen med avseende på tid och position.

Felet som presenteras i Figur 5.5 beräknas genom hypotenusan mellan den estimerade och den uppmätta positionen. Eftersom avståndet beräknas som ett absolutbelopp kan felet inte anta negativa värden. Figuren visar att positionsfelet är relativt stort, då felet ofta uppgår till 120 mm och i vissa fall överstiger även detta värde.

Medelvärdet av positionen på informationen blev: 120.32 mm.

Variansen av positionen på informationen blev: 6246.65 mm².



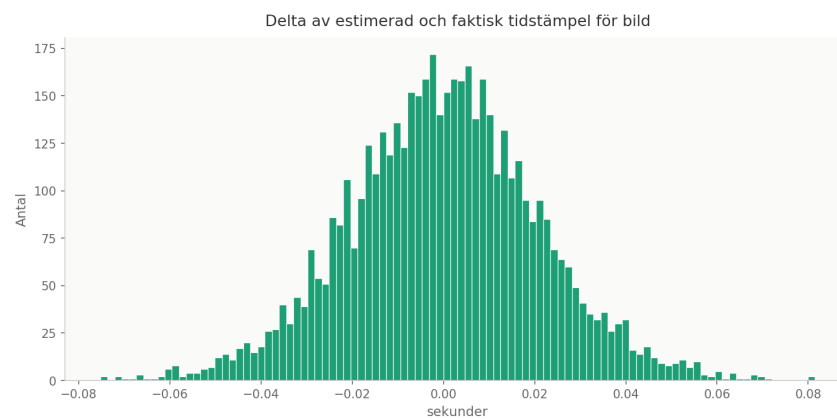
Figur 5.5: Stapeldiagram över felet i estimering av position

I Figur 5.6 kan vi se att informationen anländer innan den estimerats varje gång. Där det största felet beräknas till -0.032 sekunder.



Figur 5.6: Stapeldiagram över felet i estimering av ankomsttid

Deltan av den estimerade nästa informationsinhämtning jämfört med nästa tidstämpel ger Figur 5.7. I figuren ser vi att estimeringen är jämn då den avviker lika mycket åt höger som vänster.



Figur 5.7: Stapeldiagram över felet i estimering av nästa bild

5.5 Resultat av experiment 5

Resultaten av experimentet gällande skalningsfaktor för avgränsningsbox används i relation till forskningsfråga F-2.3 som frågar hur stor är sannolikheten att roboten finns utanför området beroende på vilken skalningsfaktor som används.

Tabell 5.4 visar hur sannolikt det är att roboten befinner sig inom det uträknade området beroende på vilken skalningsfaktor som utnyttjas. Detta räknas ut genom att ta antalet tillfällen den är inom områden jämfört med totala antalet bilder, d.v.s. 5000.

Skalningsfaktor	Innanför avgränsningsboxen [%]	Innanför avgränsningsboxen [antal]
0.0	64.84	3242
0.1	67.40	3370
0.2	70.22	3511
0.3	72.62	3631
0.4	75.26	3763
0.5	77.98	3899
0.6	80.30	4015
0.7	82.54	4127
0.8	84.66	4233
0.9	86.34	4317
1.0	87.92	4396

Tabell 5.4: Procent av bilder då roboten befinner sig inom avgränsningsboxen beroende på vilken skalningsfaktor som används

6

Diskussion

Detta kapitel diskuterar resultaten från experimenten i relation till forskningsfrågorna och de tidigare formulerade hypoteserna. Fokus ligger på att tolka systemets noggrannhet, begränsningar och möjliga felkällor samt att identifiera förbättringsområden för framtida arbete.

6.1 Diskussion av experiment 1

Resultaten visar att positioneringsnoggrannheten varierar tydligt mellan kamerorna och mellan punktmattans olika placeringar. Den generella trenden är att kameror med högre kamera-id uppvisar lägre RMSE-värden, där kamera 3 ger de mest stabila resultaten medan kamera 0 har de största felen. Detta tyder på att kamerornas placering, synfält, kalibrering och perspektivförhållanden har en betydande påverkan på den globala koordinatskattningen.

RMSE-värdena visar att systemet kan uppnå relativt god noggrannhet i vissa delar av testbädden, särskilt för kamera 3 där felet ligger under 0.1 m för samtliga placeringar. Samtidigt visar kamera 0 och kamera 1 större fel, framför allt i höger respektive vänster position. Detta indikerar att noggrannheten inte är jämnt fördelad över hela arbetsytan. En möjlig förklaring är att vissa områden ligger längre från kamerans optimala synfält eller påverkas mer av kvarvarande linsdistorsion och perspektivfel efter transformationen.

Medelfelen i x- och y-led visar dessutom att felen inte enbart är slumpmässiga, utan i vissa fall har en systematisk riktning. Exempelvis har kamera 0 ett tydligt positivt medelfel i x-led vid mitten- och högerpositionen, medan kamera 2 visar negativa avvikelser i y-led. Detta kan tyda på en förskjutning i koordinattransformationen eller en mindre feljustering mellan kamerans uppskattade koordinatsystem och referenssystemet. Kamera 3 har däremot medelfel nära noll, vilket stärker resultatet från RMSE-tabellen och visar att denna kamera ger den mest konsekventa lokaliseringen.

6.2 Diskussion av experiment 2

Resultaten från vinkelmanalysen visar att den geometriska strukturen i punktmattan i huvudsak bevaras efter bildbehandlingen, eftersom både medelvärdet och medianen ligger nära det ideala värdet 90° för samtliga kameror. Kamera 2 har lägst varians och därmed mest konsekventa vinkelmätningar, medan kamera 3 har högst varians och därmed något större spridning. Kamera 1 uppvisar en mindre systematisk förskjutning uppåt, med ett medelvärde på 90.5270° , vilket tyder på att den transformerade geometrin för denna ka-

mera är något förvrängd jämfört med det ideala rutmönstret.

Utöver algoritmiska och kalibreringsrelaterade fel kan även fysiska förhållanden i labbsal ED5355 ha påverkat resultaten för både positions- och vinkelanalysen. Fönster på ena sidan av lokalen kan ha orsakat varierande ljusförhållanden, reflektioner och skuggor över punktmattan, vilket kan försämma kontrasten och påverka detektionen av punkter. Även ojämnheter, små veck eller förskjutningar i punktmattan samt osäkerheter vid manuell inmatning av referenspunkter kan ha bidragit till felen. Eftersom både koordinattransformationen och vinkelanalysen bygger på att punktmattan ligger plant och att rutmönstrets geometri bevaras, kan sådana fysiska avvikelser leda till lokala och systematiska fel i positionsuppskattningen samt avvikelser från den ideala vinkeln 90° .

6.3 Diskussion av experiment 3

Resultatet för experiment 3 tyder på att hypotesen stämmer. Då inga variabler förändrades under experimentets gång var tiden för att gå igenom alla steg, såsom bildinsamling, analys och stabil publicering i tid med minimal varians.

Resultatet (se Avsnitt 5.3) visar på att GulliView har god prestanda, dock existerar variabler med förbättringsarbete med mer modern och snabbare hårdvara skapar potential med att programkoden kan exekveras snabbare och därtill meddelandena skapas snabbare till sin omgivning. Då skapandet utav meddelanden skulle det vara en förbättrings möjlighet att generera meddelanden med en flertrådad struktur för att nå en högre prestanda. Detta lämnas dock som en framtida utveckling och anses inte vara lika viktigt att åtgärda som andra problem i systemet då medelvärdet för informationstiden är acceptabelt.

6.4 Diskussion av experiment 4

Resultaten av experimentet (se Avsnitt 5.4) för estimerad information tyder på att den antagna hypotesen var delvis korrekt. Den korrekt föreslog att tiden för att skapa nästa bild skulle variera inom en normalkurva, se Figur 5.7, där estimeringen nästan lika ofta är för låg eller hög jämfört med den faktiska tiden att generera nästa bild. Vi kan också se från Figur 5.6 att ankomsttiden för bilderna att anlända till roboten alltid är lite innan uppskattad tid.

Hypotesen stämmer dock inte när det gäller positionsuppskattning, (se Figur 5.5) där är det vanligaste inte att uppskattningen är korrekt utan oftast skiljer det en del. Det är dock inte märkligt att positionsuppskattningen inte är exakt, experiment 1 visade på att delar av systemet lider av distorsionsfel samt perspektivfel vilket påverkar dessa uppskattningar då testet utfördes över hela testbadden. Övergången mellan kameror skapar även förvirring inom de globala koordinatsystemet vilket kan öka felen i uppskattningarna. Då testet utfördes med varierande hastighet och GulliView bygger sina uppskattningar på senaste bild-informationen skapar detta ett möjligt problem om accelerationen ökar kraftigt mellan två bilder, vilket gör att uppskattningen kan vara felaktig. Det samma gäller om dess positionering skulle ändras drastiskt mellan två bilder, exempelvis om roboten skulle svängt skarpt och hamnat utanför dess förväntade nästa position.

Med detta i åtanke anser vi detta experiment ändå vara lyckat. Att systemet lider av en del distorsions- och koordinatfel har varit känt sen tidigare men inte hur detta påverkar körningen av robotar samt hur mycket det är. Skillnaden mellan en konstant hastighet och en kraftig acceleration är drastisk. När systemet körs med konstant hastighet märks bytandet mellan kameror samt viss positioneringsfel, medans med kraftig accelerationen blir felen mycket större. Hade denna meddelandetyp appliceras till ett efterkommande projekt skulle GulliView prestera betydligt bättre under faktiska förhållanden. Robotar utrustade med mer avancerade manövrar än i detta test kan sällan uppnå samma ogynnsamma förhållanden än de som uppstod under detta test. Med detta i åtanke vill vi ändå trycka på att positionsfelen i systemet är den största begränsande faktorn i dagsläget och bör fokuseras på i framtida förbättringar. Detta på grund av att det inte går att lita på de uppskattningar som görs då grunddaten är så pass dålig.

6.5 Diskussion av experiment 5

Resultatet av experimentet (se Tabell 5.4) visar på att hypotesen är korrekt inom det utvalda intervallet. Experimentet visar en tydlig linjär koppling mellan skalningsfaktorns storlek och robotens andel tid innanför avgränsningsboxen. Målet med detta experimentet var att hitta en skalningsfaktor där roboten alltid befinner sig innanför utan att den är onödigt stor, det bästa resultatet uppgav till 87.92%. Med kunskapen om positionsproblematiken som har nämnts tidigare, ser vi detta resultat som positivt, även om den inte är 100% som förhoppningen var. I ett vidare arbete hade det varit intressant att testa skalningsfaktorer i intervallet 1.0 - 2.0 för att åstadkomma högre procent men vi anser att det troligtvis inte kommer bidra till en särskild ökad säkerhet om inte positionsavläsningarna blir mer noggranna. Eftersom när positionsfelet minskar kommer skalningsfaktorerna blir mer gynnsamma, då GulliView blir mer säker om den är inom eller utanför avgränsningsboxen eller inte.

6.6 Framtida arbete

GulliView genererar och distribuerar nu en utökad uppsättning data, men denna information utnyttjas för närvarande inte läsare på ROS nätverket, det vill säga robotarna. Eftersom detta arbete inte ingått i projektets omfattning återstår det att utveckla och implementera algoritmer som möjliggör för robotarna att autonomt bedöma risk för kollision, och vid behov initiera stopp. Utvecklingen av dessa beräkningsmodeller utgör en central del av det fortsatta arbetet för att säkerställa ett robust och säkert samspel mellan robotarna. Utöver beräkningsmodeller så återstår att framutveckla nya metoder för att förbättra den förvrängda bilden så att GulliViews positioneringssystem kan bli noggrannare, och nå en mer exakt rapportering av robotarna. Till sist vill vi upplysa om att det finns en version utav GulliView som har flertrådig egenskap, för att öka hastigheten och effektiviteten utav systemet hade ett framtida projekt varit att sammanfoga de framkallade meddelandet i detta projekt med den flertrådig versionen utav GulliView.

6.7 Sammanfattning

Sammanfattningsvis visar resultaten att systemet kan uppnå god positioneringsnoggrannhet, men att noggrannheten varierar beroende på kamera och placering. Kamera 3 gav mest stabila positionsresultat, medan vinkelmanalysen visar att rutmönstrets geometri i huvudsak bevarades nära 90° . Resultaten påverkas dock av både tekniska faktorer, såsom kalibrering och perspektivtransformation, samt fysiska förhållanden i labbsalen ED5355, exempelvis ljusvariationer, reflektioner och ojämnheter i punktmattan.

Vi kan slutligen också konstatera att det framkallade meddelandet (se Appendix B) under detta projekt har en god struktur och använder sig utav metoder som är relevanta för en framtida utveckling utav autonoma robotar med prioriterad säkerhet och framförhållning. Trots att resultaten visar på ett delvis lyckat projekt enligt hypoteserna så ser vi att dessa avvikelser i prediktion och felmarginal kan nå gynnsammare värden när meddelandet väl appliceras i en autonom robot med algoritmer.

7

Etiska aspekter

Säkerhet är ett stort fokus i detta projekt eftersom fel i lokalisering eller kommunikation kan leda till att robotarna fattar beslut baserade på felaktig data, och därmed kolliderar med varandra, utrustning eller i värsta fall människor. Det måste säkerställas att den information som produceras är tillförlitlig. Ur ett etiskt perspektiv innebär detta att systemet inte bara behöver fungera tekniskt, utan även vara tillräckligt pålitligt för den miljö där det används.

Om ett liknande system skulle implementeras i verklig trafik med autonoma fordon skulle konsekvenserna av fel i lokalisering och kommunikation vara avsevärt större än i en kontrollerad miljö. Även små fel i positionsuppskattning eller tidsfördröjningar kan leda till felaktiga beslut i säkerhetskritiska situationer, vilket kan sätta liv i fara.

En tredje etisk utmaning kan vara vem som bär ansvaret vid kollision när fel beslut har tagits baserat på felaktig positionsdata. I sådana situationer kan det vara svårt att avgöra vem som bär ansvaret för kollisionen. Ansvaret kan exempelvis ligga hos utvecklarna av lokaliseringssystemet, de som implementerat navigeringsalgoritmerna eller operatören som använder systemet. Därför är det speciellt viktigt att lokaliseringssystemet rapporterar korrekt data, speciellt när det kan existera andra system som tar till sig informationen. Trots att detta projekt har utvecklats i en kontrollerad testmiljö så kan det finnas en illustrerad risk om hur detta system skulle agera om den hade tagits i bruk under verkliga förhållanden.

Litteraturförteckning

- [1] M. Quigley, B. Gerkey, and W. D. Smart, Programming Robots with ROS: A Practical Introduction to the Robot Operating System, 1st ed. Sebastopol, CA: O'Reilly Media, Dec. 2015, ISBN: 978-1-4493-2389-9
- [2] Open Robotics Foundation. Nodes - ROS Wiki". ros.org. Hämtad 2026-04-09. [Online.] Tillgänglig: <https://wiki.ros.org/Nodes>
- [3] Open Robotics Foundation. Topics - ROS Wiki". ros.org. Hämtad 2026-04-12. [Online.] Tillgänglig: <https://wiki.ros.org/Topics>
- [4] Open Robotics Foundation. ROS/TCPROS - ROS Wiki". ros.org. Hämtad 2026-04-17. [Online.] Tillgänglig: <https://wiki.ros.org/ROS/TCPROS>
- [5] Open Robotics Foundation. ROS/UDPROS - ROS Wiki". ros.org. Hämtad 2026-04-17. [Online.] Tillgänglig: <https://wiki.ros.org/ROS/UDPROS>
- [6] L. Albinsson och A. Olofsson, *Failure Detection Using Time-Series Machine Learning Approaches*, Institutionen för data- och informationsteknik, Chalmers tekniska högskola och Göteborgs universitet, Göteborg, Sverige, 2026.
- [7] R. D. Yates, Y. Sun, D. R. Brown, S. K. Kaul, E. Modiano and S. Ulukus, "Age of Information: An Introduction and Survey" in IEEE Journal on Selected Areas in Communications, vol. 39, no. 5, pp. 1183-1210, May 2021, doi: 10.1109/JSAC.2021.3065072.
- [8] D. Chefrou, "Evolution of network time synchronization towards nanoseconds accuracy: A survey," Computer Communications, vol. 191, pp. 26–35, Apr. 2022, doi: 10.1016/j.comcom.2022.04.023.
(<https://www.sciencedirect.com/science/article/pii/S0140366422001359>)
- [9] S. Thrun, W. Burgard, and D. Fox, Probabilistic Robotics. Cambridge, MA, USA: MIT Press, 2005.
- [10] A. Amoriya, C. Kumar and S. Kumari, "Counting Number of Objects in Images Using Machine Learning," 2024 IEEE International Conference on Computing, Power and Communication Technologies (IC2PCT), Greater Noida, India, 2024, pp. 852-856, doi: 10.1109/IC2PCT60090.2024.1048672.
- [11] J. Yang, Real Time Object Tracking Using OpenCV, "2023 IEEE 3rd International Conference on Data Science and Computer Application (ICDSCA), Dalian, China, 2023, pp. 1472-1475, doi: 10.1109/ICDSCA59871.2023.1039283.
- [12] J. M. Maragh, Dynamic tracking with AprilTags for robotic education, Bachelor's thesis, Department of Mechanical Engineering, Massachusetts Institute of Technology, 2013. [Online.] Tillgänglig: <http://hdl.handle.net/1721.1/83725>
- [13] E. Olson, AprilTag: A robust and flexible visual fiducial system, "in Proc. IEEE International Conference on Robotics and Automation (ICRA), 2011, pp. 3400–3407.
- [14] Z. Zhang, A Flexible New Technique for Camera Calibration, "IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no. 11, pp. 1330–1334, 2000.
- [15] G. Bradski, The OpenCV Library, "Dr. Dobbs's Journal of Software Tools, 2000. [Online.] Tillgänglig: <https://opencv.org>

- [16] T. Malmentun, E. Nylander, T. Reinerson, E. Samuelsson, E. Svensson, and J. Viberud, "Utveckling samt evaluering av lokalisering och kooperativa styrsystem," Kandidatarbete, Institutionen för data- och informationsteknik, Chalmers tekniska högskola och Göteborgs universitet, Göteborg, Sverige, 2025.
- [17] E. Blohm, E. Diatchkov, E. Stenmark Tullberg, K. Nguyen, O. Wir, och M. Ygdell, *Globala Koordinatsystem och Autonom Navigering med SLAM och TurtleBot*, Kandidatarbete, Institutionen för data- och informationsteknik, Chalmers tekniska högskola och Göteborgs universitet, Göteborg, Sverige, 2025.
- [18] H. Tan, Dot Based Evaluation, GitHub repository *DIT669-gulliview_noetic*, 2026. Hämtad 2026-05-18. [Online.] Tillgänglig: https://github.com/HyuanTan/DIT669-gulliview_noetic/blob/main/dots_ucs/5.3%20Dot%20Based%20Evaluation.pdf

A

Appendix

Gitrepo

Länkt till projekets Github repository:

<https://github.com/CrOwn24/DATX11-Project-Scaled-Vehicles>

Bitbucket

Länkt till projekets Bitbucket repository:

ROS Integration:

<https://bitbucket.org/automationarticulatedvehicles/gulliview-ros-integration/src/main/>

Global koordinat evaluering:

https://bitbucket.org/automationarticulatedvehicles/global_coordination_evaluation/src/main/

B

Appendix

Meddelandestruktur

Nedan listats det meddelande som används för att Gulliview ska kunna ta emot likaså skicka information till och från robotar i dess omgivning.

`object_data.msg`

Detta meddelande skickas från Gulliview till robotarna, och består delvis av ett annat meddelande `ewma.msg` som visas i avsnittet efter.

```
# identity
uint32 tag_id
uint32 camera_id
uint64 sequence_number

# timestamps of different core parts
float64 message_time
float64 frame_time
float64 next_frame_time
float64 next_recive_time

# information of object
int32[2] local_coordinate

int32[2] global_coordinate
float32 theta
float32 velocity
float32 acceleraton

# gulliviews prediction
int32[2] next_coordinate
int32[4] between_bounding_box
int32[4] object_bounding_box

# margin of error
ewma next_coordinate_error
ewma time_offset
ewma residual_offset

# message information
```

```
ewma RTT
ewma success_rate
ewma publish_rate

uint64 complete_msg
```

ewma.msg

Delstruktur av meddelandet som Gulliview skickar till en robot.

```
float64 sample
float64 mean
float64 variance
```

ack.msg

Struktur på meddelande som roboterna skickar till Gulliview.

```
# identity
uint32 tag_id
uint64 sequence_number

# timestamps
float64 gulliview_message_time
float64 receive_time
float64 send_time

# bitfield
uint32 ack_bitfield
```

C

Appendix

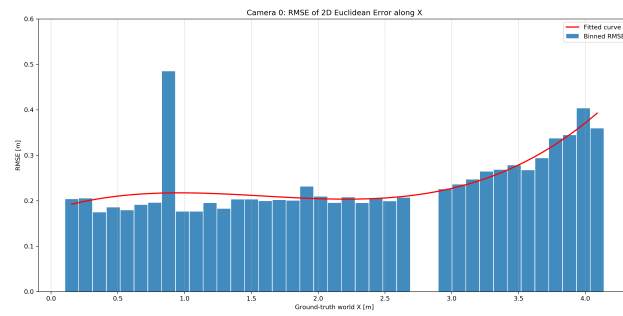
Bit nummer	Variabel namn	Gräns för korrekthet
0	all_fields	Alla fält är korrekta (1)
1	message_time	> 0 och $\in \mathbb{R}$ och $\neq \pm\infty$
2	frame_time	> 0 och $\in \mathbb{R}$ och $\neq \pm\infty$
3	next_frame_time	> 0 och $\in \mathbb{R}$ och $\neq \pm\infty$
4	next_recive_time	> 0 och $\in \mathbb{R}$ och $\neq \pm\infty$
5	theta	$\in \mathbb{R}$ och $\neq \pm\infty$
6	velocity	> 0 och $\in \mathbb{R}$ och $\neq \pm\infty$
7	acceleration	> 0 och $\in \mathbb{R}$ och $\neq \pm\infty$
8	next_coordinate_x	> 0 och < 4250
9	next_coordinate_y	> 0 och < 8000
10	between_bounding_box_x1	> 0 och < 4250
11	between_bounding_box_y1	> 0 och < 8000
12	between_bounding_box_x2	> 0 och < 4250
13	between_bounding_box_y2	> 0 och < 8000
14	object_bounding_box_x1	> 0 och < 4250
15	object_bounding_box_y1	> 0 och < 8000
16	object_bounding_box_x2	> 0 och < 4250
17	object_bounding_box_y2	> 0 och < 8000
18	next_coordinate_error.sample	$\in \mathbb{R}$ och $\neq \pm\infty$
19	next_coordinate_error.mean	$\in \mathbb{R}$ och $\neq \pm\infty$
20	next_coordinate_error.variance	$\in \mathbb{R}$ och $\neq \pm\infty$
21	time_offset.sample	$\in \mathbb{R}$ och $\neq \pm\infty$
22	time_offset.mean	$\in \mathbb{R}$ och $\neq \pm\infty$
23	time_offset.variance	$\in \mathbb{R}$ och $\neq \pm\infty$
24	RTT.sample	$\in \mathbb{R}$ och $\neq \pm\infty$
25	RTT.mean	$\in \mathbb{R}$ och $\neq \pm\infty$
26	RTT.variance	$\in \mathbb{R}$ och $\neq \pm\infty$
27	publish_rate.sample	$\in \mathbb{R}$ och $\neq \pm\infty$
28	publish_rate.mean	$\in \mathbb{R}$ och $\neq \pm\infty$
29	publish_rate.variance	$\in \mathbb{R}$ och $\neq \pm\infty$
30	success_rate.sample	> 0 och < 1
31	success_rate.mean	> 0 och < 1
32	success_rate.variance	$\in \mathbb{R}$ och $\neq \pm\infty$

Tabell C.1: Visualisering över bitfältet för `complete_msg`

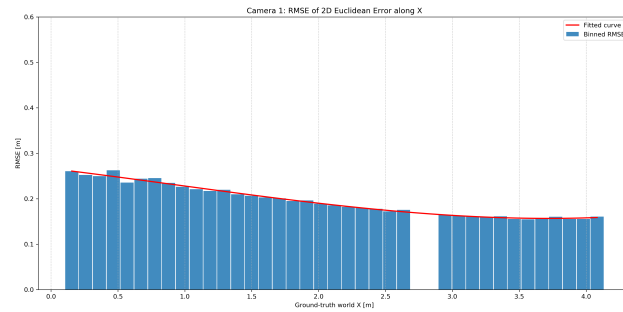
D

Appendix

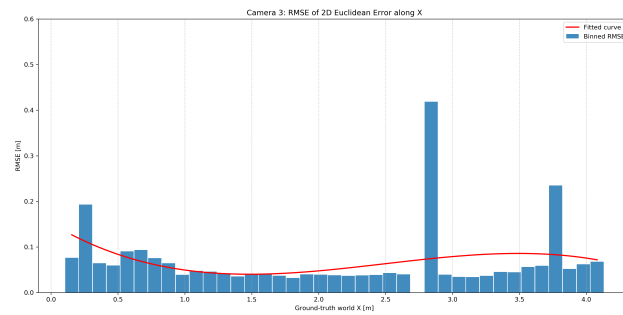
Positionsnoggrannhet



Figur D.1: RMSE av 2D euklidiska avståndsfelet över x-axel för kamera 0



Figur D.2: RMSE av 2D euklidiska avståndsfelet över x-axel för kamera 1



Figur D.3: RMSE av 2D euklidiska avståndsfelet över x-axel för kamera 3

INSTITUTIONEN FÖR DATA- OCH INFORMATIONSTEKNIK
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige
www.chalmers.se



GÖTEBORGS
UNIVERSITET



CHALMERS