Forsvaret

Bacheloroppgave

OPG3301

Predefinert informasjon

Startdato:	05-12-2022 09:00 CET	**Termin:**	2022 HØST
Sluttdato:	19-12-2022 08:00 CET	**Vurderingsform:**	Norsk 6-trinns skala (A-F)
Eksamensform:	Oppgave		
Flowkode:	1627 OPG3301 1 O 2022 HØST SKSK		
Intern sensor:	Alexander Sauter		
Intern sensor:	Christophe Massacand		

Deltaker

Navn:	Stian Algrøy
Kandidatnr.:	
FHS-id:	salgroy@mil.no

Gruppe

Gruppenavn:	Bachelor - GPS uavhengig posisjonering
Gruppenummer:	9
Andre medlemmer i gruppen:	Martin Nikolai Opdahl

Sjøkrigsskolen

Bacheloroppgave

GPS uavhengig posisjonering

– Utforsking av datamaskinens synsevne i bildeanalyse –

av

Stian Algrøy og Martin Opdahl

Levert som en del av kravet til graden:

BACHELOR I MILITÆRE STUDIER MED FORDYPNING I LEDELSE - MARINEINGENIØR VÅPENSYSTEMER, ELEKTRONIKK OG DATA

Innlevert: desember 2022

Godkjent for offentlig publiserig

Publiseringsavtale
En avtale om elektronisk publisering av bachelor/prosjektoppgave

Kadetten(ene) har opphavsrett til oppgaven, inkludert rettighetene til å publisere den.

Alle oppgaver som oppfyller kravene til publisering vil bli registrert og publisert i Bibsys Brage når kadetten(ene) har godkjent publisering.

Oppgaver som er graderte eller begrenset av en inngått avtale vil ikke bli publisert.

Vi gir herved Sjøkrigsskolen rett til å gjøre denne oppgaven tilgjengelig elektronisk, gratis og uten kostnader	[X] Ja	[ ] Nei
Finnes det en avtale om forsinket eller kun intern publisering? (Utfyllende opplysninger må fylles ut)	[ ] Ja	[ ] Nei
Hvis ja: kan oppgaven publiseres elektronisk når embargoperioden utløper?	[X] Ja	[ ] Nei

Plagiaterklæring

Vi erklærer herved at oppgaven er vårt eget arbeid og med bruk av riktig kildehenvisning.

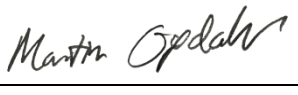
Vi har ikke nyttet annen hjelp enn det som er beskrevet i oppgaven.

Vi er klar over at brudd på dette vil føre til avvisning av oppgaven.

Dato: 18 – 12- 2022

Stian Algrøy
Kadett navn

Martin Opdahl
Kadett navn



Forord

Oppgaven er skrevet i perioden september til desember 2022 av Stian Algrøy og Martin Opdahl som et krav til utdanning fra FHS Sjøkrigsskolen.

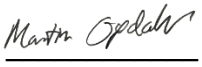
I løpet av perioden har vi fulgt et kurs for selvkjørende biler gjennom Udacity sine nettkurs. Kurset tar for seg alt som skal til for at en bil skal bli selvkjørende, og gir studentene i oppgave å løse problemer underveis i kurset. Gjennom dette kurset tilnærmet vi oss grunnlaget for å kunne forstå lignende systemer og metoder for visuell posisjonering, slik at vi kunne gjøre et begrunnet valg av metode i oppgaven. SLAM og Odometry er to posisjoneringsalgoritmer som har mulighet til å integrere flere sensorer, inkludert visuelle sensorer, for å bestemme posisjonen til et system. Vi begrenset metodene til kun de visuelle sensorene, ettersom et slikt system ikke er implementert i marinen.

Vi ønsker å gi en stor takk til førsteamanuensis Alexander Sauter for god veiledning og gode diskusjoner under arbeidet av bacheloroppgaven. Videre ønsker vi å takke førsteamanuensis Cristophe Massacand for godt samarbeid og gode samtaler.

En spesiell takk rettes til Roy Hvaara for faglig veiledning innenfor området SLAM og posisjonering.

Bergen, Sjøkrigsskolen, 18.12.2022

Stian Algrøy



Martin Opdahl

Sammendrag

GPS jamming er en stadig økende trussel i Norge. I Finnmark ble GPS-nettet jammet tre ganger i løpet av 2018, der hver periode varte i flere uker.

Metoden å jamme GPS signaler vises igjen i krigen mellom Russland og Ukraina, hvor jamming benyttes som elektronisk krigføring. Dette viser hvorfor det i større grad er viktig å benytte posisjoneringsmetoder som selvstendig kan estimere sin egen posisjon.

På sjøen finnes det metoder for å estimere egen posisjon ved hjelp av synet. Sekstanten måler vertikale og horisontale vinkler, og benyttes til å måle himmellegemers høyde over horisonten. Brukeren av sekstanten benytter synet sitt gjennom et instrument til å definere markante kjennetegn på himmelen. Kan et datasystem prosessere visuelle data og ut ifra dette bestemme egen posisjon?

Målene som var satt for oppgaven var å innhente informasjon og teste ulike metoder for visuell posisjonering. Det ble satt fire krav et posisjoneringssystem må har for å utfordre GPS posisjonering:

1. Nøyaktighet - Lav feil fra faktisk posisjon.
2. Robusthet - Systemet bør fungere i vanskelige forhold og klare å håndtere uforutsigbarheter.
3. Selvstendighet - Systemet skal fungere med en type sensor, som i denne oppgaven er visuelle sensorer.
4. Kartlegging - Systemet skal grafisk vise hvordan forflytning gjennom miljøet har vært.

Oppgaven redegjør for to metoder for selvstendig posisjonering. Visual Odometry er metoden å vite sin egen posisjon basert på kjennetegn algoritmen finner i miljøet systemet beveger seg gjennom. Deretter kalkulerer algoritmen bevegelse til disse punktene for å beregne systemets forflytning i forhold til disse. Visual SLAM er metoden å danne et lokalt kart av miljøet systemet befinner seg i, og benytte dette til å beregne egen posisjon. Algoritmen finner kjennetegn og triangulerer disse prinsipielt likt som visual odometry, men benytter en feature matching algoritme for å sammenligne punktene og kartlegge disse i et 3D kart som systemet forflytter seg i.

Innholdsfortegnelse

Figurer

Nomenklatur / Forkortelser / Symboler

Algoritme - trinnvis utførelse av prosesser for å løse problem.

Bildestrøm - En rekke sammenhengende bilder.

Datastrøm - informasjonen delt fra sensor til prosesseringselement.

Feature - Særtrekk eller kontraster i bildet.

FPS - Frames per second / bilder per sekund.

Maskinlæring - Finne mønster i en mengde data som maskinen er trent opp til å identifisere ved hjelp av kunstig intelligens.

Miljø - Geografisk område.

Motion blur - Uskarphet i bildet som følge av raske bevegelser eller dårlig fokus.

Pipeline - En rekke sammenhengende prosesser som behandler data.

Posisjonering - Person eller objekts plassering i forhold til omgivelser.

Posisjoneringsalgoritme – Algoritme som setter sammen sensordata, prosesserer denne og leverer et estimat på hvor systemet befinner seg.

Redundans - Dimensjonert for å sikre feiltoleranse, stabilitet, sikkerhet eller oppetid.

Sanntid - Varians i tid mellom registrering av sensor og prosessering av data er tilnærmet null.

Sensor fusion – Sette sammen data fra forskjellige sensorer.

Software - Betegnelse for programmer, apper og dataprogrammer som brukes på datamaskinen for å utføre bestemte oppgaver.

System - Sammensetning av komponenter som jobber sammen for å nå et felles mål.

FAST – Features from accelerated segment test.

ORB – Oriented FAST and Rotated BRIEF.

1 Innledning eller introduksjon

1.1 Bakgrunn

GPS jamming er en stadig økende trussel i verden, også i Norge. I Finnmark ble GPS-nettet jammet tre ganger i løpet av 2018. Hver jamming varte i flere uker før signalene fungerte som normalt igjen. (Johansen & Bentzrød, 2019)

"Jamming er en bevisst utsending av radiostøysignaler, for å forstyrre, stenge eller hindre mottak av signaler fra annen stasjon, og for å påvirke posisjoneringstjenester (GPS)." - (Kristiansen, 2021)

Metoden å jamme GPS signaler sees igjen i krigen mellom Russland og Ukraina. I krigen benyttes jamming som elektronisk krigføring, og dette viser hvorfor det i større grad er viktig å benytte posisjoneringsmetoder som selvstendig kan estimere sin egen posisjon for å forhindre jamming.

To kadetter ved sjøkrigsskolen har tidligere utarbeidet en bachelor som tok for seg posisjonering ved hjelp av radar, uavhengig av GPS. (Carboni & Meyer-Vikaskag, 2022) Oppgaven beskriver i korthet å jamme GPS signaler og hindre nøyaktig posisjonering på sjøen. GPS sender ut signaler med lav intensitet som enkelt kan forstyrres ved å broadcaste signaler med samme frekvens i området som skal rammes.

Oppgaven fanget vår interesse for å undersøke hvorvidt et system kan benytte seg av andre sensorer for posisjonering. Hvor passivt kan systemet være, samtidig som det er robust?

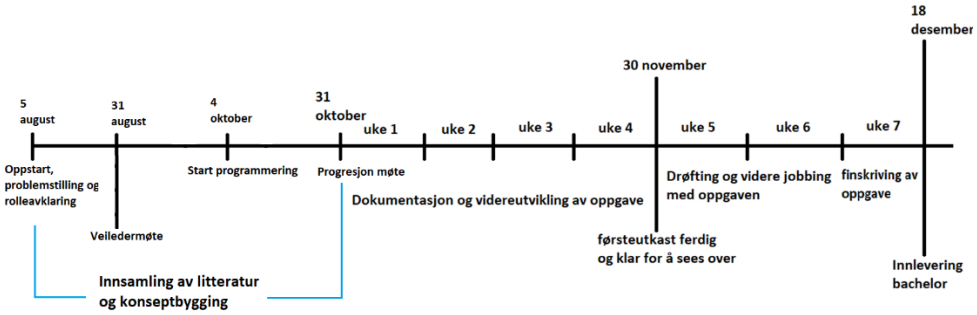
Det finnes metoder for å lokalisere seg på sjøen ved hjelp av synet. Sekstanten måler vertikale og horisontale vinkler, og benyttes til å måle himmellegemers høyde over horisonten for å bestemme sin posisjon. Dette var et av momentene som fanget vår interesse. Brukeren av sekstanten benytter synet sitt gjennom et instrument til å definere markante kjennetegn på himmelen. Kan et datasystem prosessere visuelle data og ut ifra dette bestemme egen posisjon?

1.2 Oppaveformulering

Er det mulig å lage et nytt system for lokalisering av egen posisjon som er robust og presist ved hjelp av visuelle sensorer?

1.3 Arbeidsprosess og målsetting

Ved oppstart av bacheloren ble det satt opp en tidsplan for hvordan oppgaven skulle løses. Første oppstartsmøte markerer oppstart på prosjektet hvor problemstillingen var klar, og rolleavklaring og tidslinje for arbeid gjennom perioden frem til innlevering ble satt.



Figur 1.1 – Tidslinje over arbeidsprosess

Tidslinjen inneholder fire faser, innsamling av litteratur og konseptbygging, dokumentasjon og videreutvikling av oppgaven, drøfting, og til slutt korrektur arbeid.

Første fase var "Innsamling av litteratur og konseptbygging", hvor det ble fokusert på å samle litteratur til å bygge konseptet og skrive teoridelen av oppgaven. I løpet av perioden skulle også eventuell programmering utføres og teknisk utstyr innhentes og gjøres klart til bruk. Det ble også gjennomført en veiledersamtale om konsept. Samarbeidet med veileder har bestått av 3-4 offisielle møter, samt mindre møter for veiledning. Veilederen har ofte vært tilgjengelig på skolen og har hatt åpen dør ved behov. Det ble gjennomført et progresjonsmøte 31 oktober for å sammenfatte funn, status på oppgaven, planlegge videre arbeid og oppdatere tidslinje vist i **Figur 1.1**. Etter dette møtet var to metoder valgt, visual odometry og visual SLAM, og mål ble satt. Andre metoder som ble vurdert for oppgaven var YOLO. Dette er en prediksjonsalgoritme som klassifiserer objekter i et bilde. Grunnen til at denne metoden ikke ble valgt, er fordi den vektlegger identifisering av objekter i

større grad enn systemets relative posisjon til objektet. Visual odometry og visual SLAM er derimot to metoder for posisjonering, som passet bedre til problemstillingen.

Hver mandag ble det gjennomført gruppemøte for statusrapport. Gruppemøtene besto av oppdatering på fremdrift på det fysiske produktet og arbeidsmål for uken. Første forsinkelse oppsto ved innlevering av førsteutkast. Oppgaven var fullført til og med konsept, og delvis fullført for kapitlene implementering og resultater. Førsteutkastet av implementering og resultater skulle vært levert samtidig med teori, men ble sendt 9. desember, samtidig som det ble jobbet med drøfting og korrigering av oppgaven frem til og med teori. Årsaken til forsinkelsen kom av komplikasjoner ved kalibrering og programmering av kodene.

Målene som var satt for oppgaven var å innhente informasjon og teste ulike metoder for visuell posisjonering. Deretter drøfte om metodene hadde potensiale til å implementeres på sjø og i marinen. Suksesskriteriene for en vellykket oppgave var når en eller flere metoder besto kravene om robusthet, nøyaktighet, kartlegging og selvstendighet.

1.4 Begrensninger

Oppgaven er begrenset til anvendelse av de forskjellige egenskapene til visual odometry og visual SLAM. Følgende begrensninger er gjort for denne oppgaven:

1. Systemet skal kunne være helt uavhengig av utstråling og fungere som et selvstendig system.
2. I oppgaven benyttes rimelige kamera for å utforske hvor kosteffektivt systemet kan bli.
3. Sluttproduktet skal fungere som et lokaliseringsverktøy, ikke et styresystem. Dette betyr at oppgaven ikke tar for seg autonomi i den grad fartøyet skal operere selvstendig, men undersøker hvorvidt SLAM og odometry kan bistå til mer effektiv posisjonering.
4. Systemet testes ut over et begrenset geografisk område for å teste de ulike aspektene ved oppgaven
5. SLAM og odometry har mulighet til å integrere flere typer sensorer inn i samme kart for en mer robust og presis lokalisering. Oppgaven er avgrenset til kun optiske sensorer.

1.5 Struktur

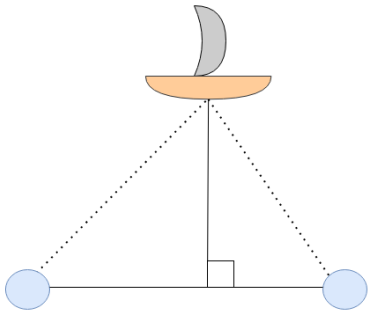
Oppgaven starter med en teoridel der Visual Odometry og SLAM som begreper blir redegjort for. Undervegs i teoridelen dykkes det dypere i hver av de to posisjoneringsalgoritmene, der disse blir brutt ned og forklart gjennom teori og matematiske modeller. Konseptutviklingen følger teoridelen, som forklarer hvorfor det velges å fokusere på de to forskjellige posisjoneringsalgoritmene. Konseptutvikling tar for seg krav og avgrensninger som har blitt gjort for å binde teori med praktisk utførelse. Videre implementeres systemet med sensorer og algoritmer, før resultater fra systemene blir vist. Avslutningsvis drøftes resultatene og testene for hvorvidt disse ble suksessfulle, og det oppsummeres hvilke læringsmoment som har oppstått. Til slutt kommer forslag til videre arbeid med oppgaven, hvilken algoritme det anbefales å jobbe videre med og hvordan denne metoden kan være med på å løse marinens behov for GPS uavhengig posisjonering.

2 Teori

Nå vil relevant teori som er benyttet for å forklare de to valgte metodene redegjøres for. Oppgaven starter med generell teori som gjelder for begge metodene, deretter redegjøres det for teorien bak de to ulike metodene.

2.1 Triangulering

Triangulering har lenge blitt brukt som en metode for å kartlegge landemerker. Ved å sammenbinde enkelte punkter får man et nett som kalles triangelnett. Deretter måles vinkler i dette triangelnettet og punktenes innbyrdes beliggenhet regnes ut. Triangulering utføres ved å krysspeile et punkt fra to posisjoner med kjent avstand mellom hverandre, og gjennom bruk av trigonometri regne ut avstanden normalt fra linjen mellom de to punktene, og ut til det tredje punktet som vist i **Figur 2.1.** (Mæhlum, 2020)



Figur 2.1 – Illustrasjon av hvordan triangulering fungerer

Krysspeiling er en måte å triangulere egen posisjon på sjøen ved hjelp av et kompass, et kart og synet. Triangulering starter ved å lete etter kjente landemerker, ta kompasskursen til landemerket og finne dette igjen på kartet. Når man har funnet et landemerke på kartet lager man en linje fra dette landemerket med den kursen som ble funnet fra sin posisjon. Deretter finner man et nytt landemerke og gjentar prosessen. Dette er egentlig tilstrekkelig for å lage et kryss for hvor man befinner seg, men for å øke nøyaktigheten gjentar man prosessen med et tredje landemerke. Resultatet blir et kart med tre linjer som møtes et sted på kartet og gir et grovt estimat på egen posisjon. (Kjerstad, 2021)

2.2 Detektere kjennetegn

Feature detection er måten en algoritme legger merke til kjennetegn, eller features, i et bilde. En feature er små kjennetegn som en sensor får fra miljøet systemet beveger seg i. Eksempel på en feature kan være et hjørne, en kant, en prikk, en sterk farge eller andre egenskaper miljøet har som skiller seg fra resten.

Feature extraction er en metode for å gjøre om data innhentet fra feature detection til numeriske features som kan prosesseres, samtidig som man tar vare på informasjonen fra opprinnelig data. (MathWorks, 2022)

Ofte blir bildet algoritmen skal detektere features fra, gjort om til et gråskala bilde. Ved å bruke gråskalering så fjernes de ekstra lagene med farger, som gjør at bildet kun består av intensiteten til hver piksel med en verdi mellom 0-255. Dette gjør prosesserings belastningen lavere. Deretter vil algoritmen detektere kanter og hjørner ved å se på kontrasten mellom partikler med lav og høy verdi.

2.3 Kamera kalibrering

Det finnes flere metoder å benytte kamera som sensor. De to metodene som er benyttet i oppgaven er monocular vision og stereo vision. En av de viktigste operasjonene som må bli gjennomført for å få presise målinger med disse metodene, er kalibrering. Kamera har tilnærmet lik struktur som et menneskeøye ved at lys går gjennom en konveks linse og deretter gjennom et lite nåløye, før lyset treffer netthinnen. Nåløyet i sammenhengen med kamera blir fokalpunktet til linsen. Utfordringer med dette er linsen, fordi en konkav linse samler lysstråler og forårsaker en forvrengning. Konsekvensen blir at bildet blir mer og mer forvrengt desto lenger fra sentrum. For å rette opp effekten til forvrengningen, blir det behov for å gjøre en kamerakalibrering. I tillegg blir det nødvendig å kalibrere kameraets posisjon og perspektiv til hvordan et 2D-bilde kan leses i en 3D verden. For stereovision med to eller flere kameraer, må disse også kalibreres i forhold til hverandre slik at de ser punktene likt. (MathWorks, 2022)

Det er to sett med parametre som trengs for å bevege seg fra 3D perspektiv til 2D-perspektiv. De ytre parametrene beskriver kameraets orientering og relative posisjon sammenlignet med den globale 3D orienteringen. Matrisen for ytre parametere blir referert som extrinsic matrix. For å finne denne, kreves en rotasjonsmatrise som forteller om kameraets egen rotasjon i sitt 3D rom. I tillegg kreves en transformasjonsmatrise for å rette opp kameraets 3D rom med det globale 3D rommet.

$$R = \begin{bmatrix} r11 & r12 & r13 \\ r21 & r22 & r23 \\ r31 & r32 & r33 \end{bmatrix} \quad t = \begin{bmatrix} tx \\ ty \\ tz \end{bmatrix} \quad Mext = \begin{bmatrix} r11 & r12 & r13 & tx \\ r21 & r22 & r23 & ty \\ r31 & r32 & r33 & tz \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.1}$$

De indre parametrene består av en matrise med brennvidde, brennpunkt og det optiske senter, som er det geometriske midtpunktet til linsen. Interne parametre blir satt inn i en matrise, som blir referert til intrinsic matrix.

$$Mint = \begin{bmatrix} fx & 0 & ox & 0 \\ 0 & fy & oy & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{1.2}$$

Matrisen forklarer hvordan punkt i 3D verdenen blir oversatt til punkt i et 2D-plan eller bilde. I mange tilfeller holder det å vite intrinsic matrisen for å kalibrere et enkelt kamera. For å kalibrere et stereovision system trengs projeksjons matrisen. Denne matrisen er en 3x4 matrise som tar hensyn til kameraets oppfattelse av hvilke lysstråler som tilhører hvilke piksler, og er et produkt mellom intrinsic matrix og extrinsic matrix.

$$P = \begin{bmatrix} p11 & p12 & p13 & p14 \\ p21 & p22 & p23 & p24 \\ p31 & p32 & p33 & p34 \end{bmatrix} = \begin{bmatrix} fx & 0 & ox & 0 \\ 0 & fy & oy & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r11 & r12 & r13 & tx \\ r21 & r22 & r23 & ty \\ r31 & r32 & r33 & tz \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.3}$$

2.4 Posisjoneringsalgoritmene

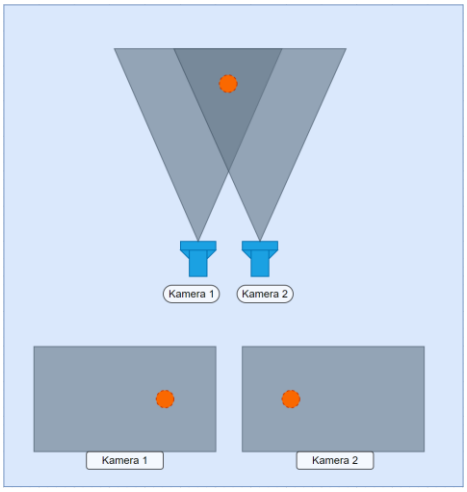
Det er valgt to posisjoneringsalgoritmer for denne oppgaven; Visual Odometry og Visual SLAM. Det redegjøres nå for algoritmene, og for teorien bak de to forskjellige metodene for lokalisering.

2.4.1 Visual odometry

Visual Odometry tar for seg hvordan systemet beveger seg fra bilde til bilde, og ikke hvor det har vært. Dette er grunnprinsippet til odometry, og som benytter seg av ulike verktøy for å oppnå denne forflytningen. De to metodene som forteller systemet om forflytning, er feature matching og optical flow. Ved å finne features i et bilde, er det mulig for systemet å få et overblikk over miljøet det befinner seg i. Eksempler kan være å bruke en deteksjonsalgortime, som FAST algoritmen, for å finne hjørner eller kontraster. Når features blir funnet over flere bilder, kan optical flow lage tilknyttede bevegelsesvektorer for punktene. Optical flow har som oppgave å estimere systemets forflytning ved å sammenligne en feature i et bilde i sanntid med en i et tidligere bilde. Disse verdiene blir midlertidig lagret og brukes til å beregne forflytning mellom hvert bilde. Hver lille forflytning blir kartlagt i et 2D kart for å lage den totale ruten posisjoneringsalgoritmen oppfatter. Ettersom Visual odometry kun lagrer de små forflytningene og ikke spesifikke features, blir ruten mindre presis desto lengre systemet er i drift.

Bildeserie som datakilde

Et godt input til visual odometry er essensielt for at programmet skal operere på best mulig måte. Det er to metoder denne bildestrømmen blir oppfattet på i en kode. Den første er ved hjelp av et sanntids opptak gjennom kameraer, og den andre er ved hjelp av et datasett. Datasett kan bli fremstilt på flere ulike måter ut fra hvilke resultater som er ønsket. Kvaliteten på bildene er utslagsgivende for feature deteksjon, hvor flere piksler gir flere målbare features. Kompileringshastigheten til visual odometry er i større grad avhengig av maskinvare og optimalisering av denne, enn det er avhengig av et godt kamera. (Gul, et al., 2021)



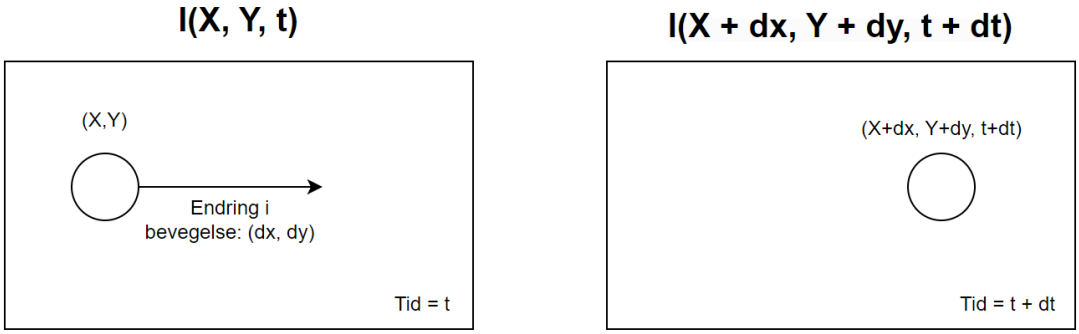
Figur 2.2 – Illustrasjon av stereo vision

Stereo vision benytter datastrøm fra to kamera slik at systemet får en dybdeforståelse av miljøet. For at kameraene skal kunne samhandle, er de nødt til å gjennomgå kalibrering slik at bildeforvrengning av linsene blir minst mulig. Etter at kalibrering er utført og forvrengning har blitt korrigert, er det mulig å måle dybde ved å se på differansen mellom posisjonen til et objekt i bildene fra begge kameraene.

Estimering av bevegelse

Optical flow er bevegelsen til et objekt mellom bilder i sekvens, forårsaket av den relative bevegelsen mellom kamera og objektet. Et objekt har posisjon x og y i det første bildet. Deretter beveger objektet eller kameraet får en dx og dy verdi som forteller systemet om endring av objektets posisjon. Dette er illustrert i **Figur 2.3**. (Lin, 2018)

Ved å følge objektet gjennom konsekvente bilder i sekvens, kan formelen utledes for intensiteten til pikselen (I) ved å se på posisjonen avhengig av tid. Piksel intensitet er sammenhengen mellom hvor lys eller mørk en piksel er ved bevegelse gjennom flere bilder. Den nye intensitet til objektet er gitt som summen av x, y, og t med korresponderende delta verdier.



Figur 2.3 – Objektets forflytning i tid

Først antas at intensitet for pikslene er bevart for hvert bilde, og setter opp likningen for intensitet før lik intensiteten etter:

$$I(x, y, t) = I(x + \delta x, y + \delta y, t + \delta t) \quad (1.4)$$

Deretter forenkles høyre side og fjerner fellesbetegnelser:

$$I(x+\delta x, y+\ \delta y, t+\ \delta t) =\ I(x,y,t) +\ \frac{dI}{dx}\delta x + \frac{dI}{dy}\delta y +\ \frac{dI}{dt}\delta t +\ \ldots$$
$$=>\ \frac{dI}{dx}\delta x +\ \frac{dI}{dy}\delta y\ \frac{dI}{dt}\delta t = 0 \tag{1.5}$$

Så deles endringen i tid for å utlede likningen for optical flow som er:

$$\frac{dI}{dx}u +\ \frac{dI}{dy}v +\ \frac{dI}{dt} = 0 \tag{1.6}$$

$\frac{dI}{dx}$, $\frac{dI}{dy}$ og $\frac{dI}{dt}$ er bilde gradientene langs den horisontale-, vertikale- og tidsaksen.

Dermed kan likningene løses for $u(\frac{dx}{dt})$ og $v\left(\frac{dy}{dt}\right)$ for å bestemme bevegelse over tid.

Optimalisering

Siste punkt i visual odometry er local optimization. Essensen i dette punktet handler om at variablene i de tidligere punktene i pipeline blir kalibrert til det miljøet og hastigheten som systemet skal fungere i. Dette er valg som tas og som definerer ytelsen til algoritmen og det er målet med dette punktet, å øke ytelsen og kalibrere etter ønsket effekt. I tillegg er det også viktig å forstå at systemets maskinvare og software utgjør en like viktig rolle for ytelsen til programmet som valg av metodene visual odomety benytter. En kjappere feature deteksjonsmetode er ikke nødvendigvis mer presis som en tregere metode. En SSD er kjappere enn en harddisk men kostnaden er høyere. Den begrensede faktoren er hvordan datamaskinen som visual odometry skal kjøres på er satt sammen.

2.4.2 Visual SLAM

SLAM er en lokaliserings- og kartleggings algoritme som gjennom observasjoner fra sensorer danner et estimat av hvor systemet tror det befinner seg i et miljø. SLAM står for "Simultaneous Localization and Mapping" og benytter seg av flere forskjellige sensorer for å lage et system som både lokaliserer et system i sanntid, og kartlegger denne informasjonen. På denne måten kan dataene benyttes senere når systemet går over samme område på nytt.

Visual SLAM benytter seg av visuelle sensorer for å bestemme egen posisjon. Bildeserien som hentes fra den visuelle sensoren, sendes til en feature detector som finner de beste kjennetegnene fra bildet og lagrer disse. Prosessen for å hente ut features er lik for Visual Odometry og Visual SLAM. Kjennetegnene blir deretter sendt til et kart og plottes som 3D punkter relativt til hvor de har blitt observert av systemet. Deretter blir kjennetegnene sammenlignet med korresponderende kjennetegn i det neste bildet i bildeserien for å estimere bevegelse til systemet. Posisjonen til systemet og kameravinkel blir oppdatert i kartet. Resultatet fra kartdata og posisjonen SLAM produserer er svært nøyaktig, fordi et fullstendig system bruker flere sensorer til å danne et felles kart for å oppdatere posisjon. Denne oppgaven baseres på Visual SLAM og vil derfor ikke gå mer i dybden på integrasjon av andre sensorer.

Visuell sensor

Systemet benytter monocular vision for å motta bildestrøm. Monocular vision skiller seg fra stereo vision ved at det kun benytter seg av ett kamera for å motta bildestrømen. Forskjellen på de to ulike anvendelsene av sensordata er at stereo vision sammenligner to bilder fra sanntid med hverandre, mens monocular vision tar sanntidsbildet og sammenligner dette med det forrige bildet for å triangulere features i de to bildene og finne sin relative posisjon til disse featurene.

Det finnes i utgangspunktet to forskjellige måter å håndtere data fra monocular vision. Den ene er feature based som oppgaven fokuserer på, mens den andre er direct. At monocular vision er direct betyr at algoritmen håndterer rådata til forskjell fra feature basert som håndterer kjennetegn algoritmen finner i bildet. (Krishnan & Sahin, 2019)

Oppdatere systemets posisjon

Visual SLAM benytter feature matching for å oppdatere posisjonen til systemet. Feature matching er en teknikk som er brukt for å identifisere og sammenligne ulike features fra et bilde til et annet. Metoden handler om å sammenligne to bilder med hverandre for å finne likheter.

For å gjennomføre feature matching, blir kjennetegn detektert som beskrevet i **Kapittel 2.2**. Disse kjennetegnene blir deretter beskrevet matematisk, ofte som en vektor, som fanger opp og lagrer karakteristikk ved hver feature. Når metoden har funnet korresponderende features i forskjellige bilder, beregner feature matching forflytningen denne featuren har hatt fra det ene bildet til de andre. Dette forteller systemet om egen forflytning relativt til de punktene som har blitt identifisert i rommet. (Tyagi, 2019)

Kartlegging

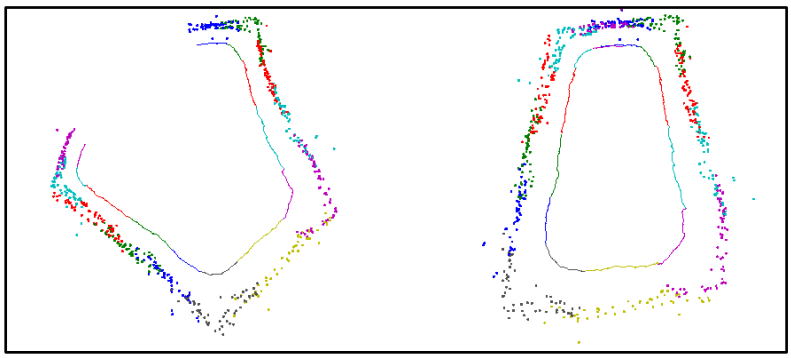
Kartlegging er en metode for hvordan man kan lagre informasjon om landskap for å senere kunne bevege seg enklere, tryggere og hurtigere. Kartene kan være enkle fremstillinger av landskapet, eller tyngre og mer detaljerte.

Visual SLAM fremstiller features som er sammenlignet gjennom flere bilder som 3D punkter i et lokalt kart. Forskjellen på visual Odometry og visual SLAM er at visual Odometry beregner forflytning basert på features forflytning i bildet. Visual SLAM lagrer først punktene i et kart og bruker kartet til å beregne forflytningen. Det er dette kartet som er hovedforskjellen mellom de to posisjoneringsalgoritmene. (Mathworks inc., 2022) Det lokale 3D kartet kan deretter lagres og anvendes senere dersom systemet skal forflytte seg i det samme området en annen gang. Det vil si at første gangen systemet beveger seg gjennom et område må det både kartlegge området og posisjonere seg etter dette kartet. Når kartet er laget, trenger kun systemet å posisjonere seg etter dette.

Optimalisering

Visual SLAM optimaliserer posisjonen sin ved å gjennomføre loop closing. Loop closing er en metode for å bestemme om et system har returnert til sin opprinnelige posisjon. Ved loop closing oppdateres geometrien til rutem systemet har gått og dette brukes til å optimalisere posisjonen til systemet.

Det er flere faktorer som kan forårsake feil i løpet av en forflytning. Loop closing korrigerer denne feilen ved å optimalisere posisjonen når systemet kommer tilbake til utgangspunktet**.** Loop closing hjelper også systemet å forstå hvordan geometrien til miljøet er utformet. Loop closing gjennomføres når tilstrekkelig mange features blir gjenkjent fra tidligere bilder. Dette forteller systemet at det har gjennomført en runde og er tilbake igjen ved utgangspunktet. Dette kan over tid bli mye prosessering med mange hundre frames og tusenvis av features. (Williams, et al., 2008)



Figur 2.4 – Loop closure før og etter systemet har funnet utgangspunktet (Williams, et al., 2008)

Figur 2.4 viser fra venstre mot høyre hvordan systemet beveger seg i et miljø. Over tid vil usikkerheten for egen posisjon øke, som vist til venstre i figuren. Derimot, når systemet returnerer tilbake til utgangspunktet, så vil loop closure registrere kjennetegn fra tidligere og oppdatere posisjonen. Dette er illustrert til høyre i figuren.

3 Konseptutvikling

I teoridelen ble to ulike metoder for posisjonering redegjort for; Visual Odometry og Visual SLAM. Odometry og SLAM har sine kvaliteter og bruksområder, og noen ganger blir deler ved Odometry også benyttet i SLAM som en del av systemet. Teorien danner grunnlag for konseptutviklingen. Før teorien ble implementerert ble det gjort valg for krav og avgrensninger av oppgaven.

3.1 Krav

For at et lokaliseringssystem skal fungere hensiktsmessig, er det viktig å forstå noen krav et slikt system er nødt til å oppfylle for at det skal lønne seg å forske videre på et alternativt lokaliseringssystem til GPS. Kravene er listet opp nedenfor etter viktighet:

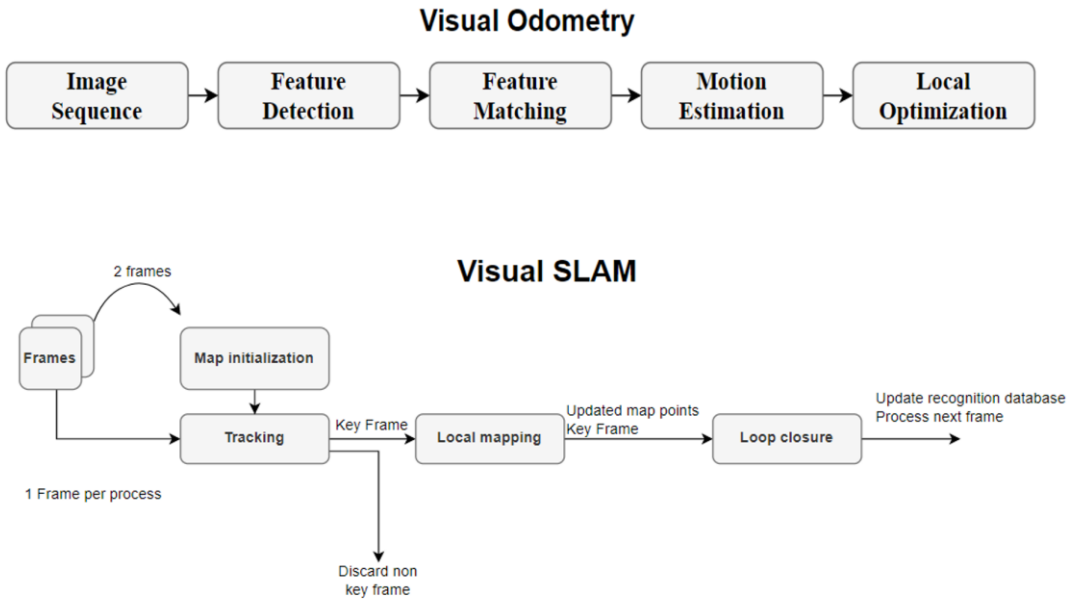
Krav	**Beskrivelse av krav**	**Ønsket måloppnåelse**
Nøyaktighet	Systemet har lav feil fra faktisk posisjon.	Systemet klarer å posisjonere seg med lavere feilmargin enn andre posisjoneringssystemer.
Robusthet	Systemet bør fungere i vanskelige forhold og klare å håndtere uforutsigbarheter	Presisjon påvirkes ikke av endring i miljø.
Selvstendighet	Systemet skal fungere med en type sensor, som i denne oppgaven er visuelle sensorer.	Levere presis og robust posisjon ved hjelp av kun en sensor.
Kartlegging	Systemet skal vise grafisk hvordan forflytning gjennom miljøet har vært.	Skal kunne kartlegge områder det tidligere har vært for å enklere og mer intuitivt klare å lokalisere seg på samme plassen senere.

3.2 Avgrensninger

Oppgaven avgrenses ved en rekke valg som er tatt fra teori til implementasjon. Avgrensningene listet nedenfor er konkrete valg gjort undervegs i realiseringen av prosjektet.

Avgrensning	**Beskrivelse av avgrensning**
Sensor	Systemene som testes ut i praksis, tar utelukkende for seg kamera. Oppgaven skal utforske hvor godt et system kan fungere kun med visuelle sensorer. Fordeler ved å benytte kamera er at dette er en tilgjengelig og lett anvendelig sensor.
Algoritme	I oppgaven utforsker to ulike posisjoneringsalgoritmer, visual odometry og visual SLAM. Disse har blitt valgt da de har fått mye oppmerksomhet i sammenheng med lokalisering og posisjonering, og det finnes mye tilgjengelig kode å undersøke.
På land kontra i sjøen	Testene gjennomføres i områder med lite friksjon for å lettere kunne teste ut algoritmene uten å være avhengig av båt. Dette gjør det også mulig å teste posisjoneringsalgoritmene oftere dersom det oppstår problemer.
GPS	Resultatene fra posisjoneringsalgoritmene blir ikke sammenlignet med GPS posisjon, fordi majoriteten av datasettene gjennomføres innendørs og i lukkede områder der GPS signaler er upresise. GPS har en feilmargin opp mot 7 meter for hver posisjon. Istedenfor blir resultatene sammenlignet med hverandre og oppmålte modeller over enkelte av miljøene.

4 Implementering



Figur 4.1 – Pipeline til metodene

Oppgaven fortsetter nå til implementering av metodene. Med bakgrunn i teori og i henhold til krav og avgrensninger, har oppgaven undersøkt de to ulike metodene å posisjonere. I implementering er funksjonaliteten og oppbygningen til posisjoneringsalgoritmene forklart.

4.1 Maskinvare

Maskin

Datamaskinen brukt for prosjektet er en Lenovo Thinkpad med følgende spesifikasjoner:

Prosessor	Intel(R) Core(™) i7-7500 CPU @ 2.70GHz
RAM (Random Access Memory)	8,00 GB Single-Channel DDR4 @ 1063 MHz
Hovedkort	LENOVO 20H1004VMX (U3E1)
Grafikkort	2047 MB NVIDIA Geforce 940MX
Lagringsplass	238 GB SanDisk SATA SSD

Kamera

I oppgaven benyttes to Logitech C930e ultra wide Angle 1080p webkamera for å kjøre Visual Odometry. Disse kameraene har USB overføring av både strøm og data. Kamera leverer data opp til 1920x1080 HD oppløsning.

Figur 4.2 – Logitech C930 webkamera

I tillegg brukes et GoPro hero 7 kamera for testing til Monocular SLAM. Kamera er utstyrt med en intern stabilisering og har mulighet til å filme med 1920x1440 oppløsning i 60 FPS.

Figur 4.3 – GoPro Hero 7 kamera

4.2 Dataset

I oppgaven er det samlet inn data fra tre ulike miljøer på skolens område i den hensikt å teste algoritmene. Områder som har blitt testet ut er skolens aula, bibliotek og kaiområde. Dette for å teste systemet i et åpent miljø, teste systemet i et trangt og vanskelig miljø og for å teste systemet utendørs.

Visual Odometry benytter et datasett med bildefiler fra to logitech kameraer. Oppløsningen på bildene er nedgradert til 480p og bildene er tatt i en hastighet på 10 bilder per sekund. I tillegg blir kalibreringen av kamera lagt ved som en tekstfil. I denne oppgaven er det ikke brukt GPS posisjon. Posisjoneringsalgoritmen har en innebygd funksjon for å sammenligne resultat med GPS posisjon som er erstattet med oppmålte områder. GPS blir ikke benyttet til å sammenligne resultatene med, da presisjonsnivået til GPS ikke egner seg for miljøene til testene.

Visual SLAM har samlet inn videodata fra de tre miljøene med et GoPro kamera. Oppløsningen på videoene har blitt redusert til 540p for å minske belastningen på prosessoren. Deretter er videoene sendt gjennom linux kommandoen:

(ffmpeg -i ../(Navn_på_video) -vf fps=(Antall_frames) (Navn_på_dataset_%06d.png))

Kommandoen bryter ned videoene i bilder, der første bilde i dette eksempelet blir kalt Navn_på_dataset_000001.png, andre blir kalt Navn_på_dataset_000002.png og resten av bildene i datasettet følger dette formatet. Datasettene har en bildehastighet på 24 FPS, som er den optimale frameraten til monocular SLAM algoritmen.

4.3 Posisjoneringsalgoritmen - Visual Odometry

Anvendelse av teorien til oppgaven, startet med at systemet måtte klargjøres til bruk med maskinvare og software. To kamera ble plassert på datamaskinen slik at avstanden mellom sentrum av linsene var 10 cm. Deretter ble et utvalg av programvarer og bibliotek lagt til for å kjøre og prosessere visuell

Figur 4.4 - Maskin

data. For å strukturere systemet ble Anaconda Navigator installert, og det ble etablert et eget arbeidsmiljø. Anaconda er en miljøbehandler som er open source og enkel å bruke. Miljøbehandleren gjør at installasjonene ikke blandes med andre bibliotek eller pakker som finnes i root. (Anaconda inc., 2022)

Python

Programmeringsspråket som benyttes for Visual Odometry er Python. De viktigste bibliotekene som er benyttet for å støtte python er numpy, OpenCV og matplotlib.

Numpy

Numpy er et matematisk bibliotek som implementerer funksjoner for å løse matematiske problemer. (NumPy, 2022)

Matplotlib

Matplotlib er et visualiserings bibliotek som hjelper med å plotte resultatene ut fra de matematiske beregningene. (The Matplotlib development team, 2022)

OpenCV

OpenCV er et *Open source Computer Vision* bibliotek spesiallaget for å prosessere visuelle data, enten video eller bilder. Biblioteket er utviklet for å lettere kunne gjennomføre bildeanalyser ved å bistå med verktøy for bildebehandling. Biblioteket er tett knyttet til maskinlæring og resultatene fra verktøyene i OpenCV har et format som enkelt kan leses av for maskiner. Det vil både kunne behandle en bildestrøm i sanntid eller fra et datasett med video eller bilder. (OpenCV team, 2022)

Det ble nødvendig med to koder for at visual odometry skullel fungere. Den første koden ble brukt til å kalibrere kameraene slik at linseforvregningen og kameraposisjon ble riktig korrigert. Posisjoneringsalgoritmen prosesserte bildene fra kameraene og produserte resultatet grafiskt. Oppgaven tar utgangspunkt i Github brukeren Nicolai Nielsens programmering, og anvender koden for kalibrering og posisjoneringsalgoritmen som finnes i brukerens repository. Denne koden har blitt konfigurerer slik at den kunne brukes til oppsettet og miljøet oppgaven har testet ut. (Nielsen, 2022)

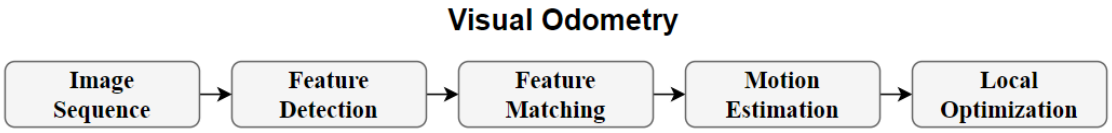
Stereo kamera kalibrering

Stereo kamera kalibrering handler om å kalibrere begge kamera slik at forvrengninger og ujevnheter utlignes. Første mappe i filstrukturen inneholder kodene med mål om å korrigere forvrengninger i bildet og gjør at målingene tatt i prosesseringsalgoritmen blir mest mulig eksakt. Det er tre koder i mappen. **"calibration_image.py"** koden tar bilder med begge kameraene samtidig. **"stereovision_calibration.py"** kalibrerer bildene og regner ut projection matrix. I tillegg lager den en fil med verdier for kalibrering i en xml fil. "**stereovision.py"** leser av verdiene fra xml filen og viser effekten av kalibreringen over kameraene.

For å kalibrere kameraene ble flere bilder tatt av et geometrisk kjent objekt fra flere forskjellige vinkler. Det mest anvendte objektet er et sjakkbrett, ettersom sjakkbrett består av et svart/hvit mønster med 10x10 ruter som tilsvarer 9x9 hjørner. Avstanden mellom hjørnene vil alltid være konstant, og eventuell fortrenging kan beregnes og korrigeres. For å hjelpe algoritmen ble den nederste raden fjernet, slik at brettet fikk flere hjørner i horisontal retning enn i vertikal. Dette hjalp kalibreringsalgoritmen å forstå hvilken rotasjon sjakkbrettet har i forhold til kameraet. Etter at målingene var tatt, ble disse overført til koden for kalibrering, som ved hjelp av OpenCV biblioteket skrev ut projection matrix.

Metoden benyttet for å kalibrere bildene tilstrekkelig, var å ta utgangspunkt i en projection matrix fra et lignende system. Derretter ble flere bilder av sjakkbrettet prosessert i kalibreringskoden som produserer en egen procjetion matrix for Logitech kameraene. Denne projection matrix ble justert opp eller ned ut fra hvilke verdier som varierte fra projection matrix til det lignende systemet. På denne måten ble kalibreringen mer presis, og enkelte manuelle endringer ble gjort for å oppnå best mulig resultat. Den manuelle justeringen kom av at metoden benyttet i kalibreringsalgoritmen ikke selvstendig klarte å finne en perfekt matrise.

4.3.1 Pipeline Visual Odometry



Figur 4.5 – Pipeline for visual odometry

Filstrukturen for posisjoneringsalgoritmen inneholder en mappe for datasettet sammen med prosesseringsalgoritmen. I tillegg inneholder filstrukturen et bibliotek for visualisering som tar for seg plotting, video, bilder og kamera. Oppsettet for Visual Odometry består av funksjoner som gjennomfører hvert steg av prosessen til pipeline. De første funksjonene hentet data fra datasettet. Deretter ble bildene sendt gjennom kalibrering, feature detection og optical flow. De resterende funksjonene er matematiske funksjoner som behandlet matrisene og produktene av de ulike prosessene. Hver funksjon er beskrevet i detalj i **Vedlegg B**.

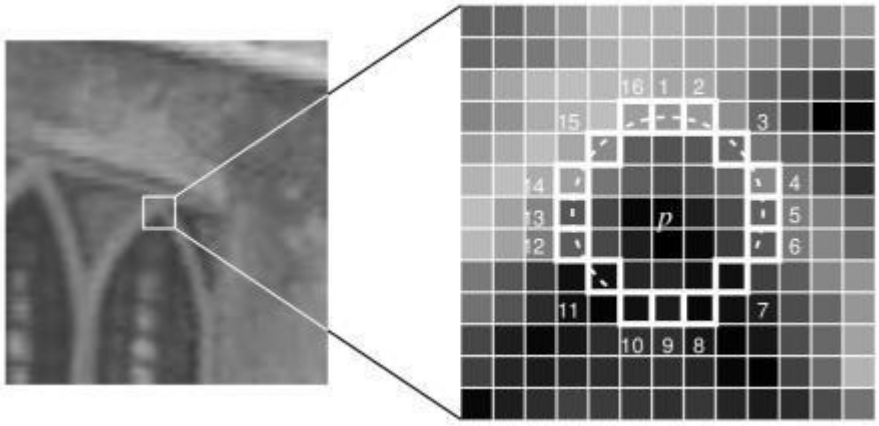
Første steget

Det første steget i pipeline er *image sequence*. Det første steget laster kalibreringen til kameraene opp fra tekstfilen som kalibreringskoden har laget. Koden prosesserer ett bilde av gangen som vil si at bilde per sekund er bestemt av hastigheten datamaskinen klarer å gjennomføre de ulike prosessene.

FAST

FAST har blitt benyttet til feature detection. Algoritmen er designet for hurtig deteksjon av features som gjør det mulig å detektere i sanntid. Før feature detection kan gjennomføres må bildet først omdannes til et svart/hvit bilde, hvor hver piksel får en oversatt verdi fra 0-255. Denne algoritmen finner hjørner ved å ta et mørkt interessepunkt, gir dette en lav pikselverdi, og lager en sirkel på 16 piksler rundt punktet. Hvis flere av pikslene i sirkelen har en sammenhengende høy pikselverdi, vil punktet bli identifisert som et hjørne. For å fordele flere features over hele bildet, blir bildet delt inn 32x48 blokker, der

FAST algoritmen blir kjørt i hver av blokkene. Resultatet blir at færre features blir konsentrert i en del av bildet. Konsekvensen av å ha alle features konsentrert, er at ved en forflytning så kan mange features forsvinne samtidig. (OpenCV, 2016)



Figur 4.6 – FAST klassifisering av punkter (OpenCV, 2016)

Optical Flow

Måten optical flow har blitt implementert er ved bruk av funksjoner fra OpenCV og støttefunksjoner i koden. Metoden har gjennomført beregninger for forflytning som definert i teori tidligere i **Kapittel 2.4.1**. Systemet vil på denne måten forstå forflytning ved å se på endring i posisjon for hver feature FAST detekterer. Deretter finner FAST sammenhengen mellom punktene og danner bevegelsesvektorer. Feature matching behandles av optical flow. Funksjonen for optical flow benytter teorien beskrevet i **Kapittel 2.4.1**, til å estimere bevegelsen til features. På en grafisk fremstilling av resultatet fra optical flow vil de punktene i et bilde som står stille, ikke ha en verdi. De punktene som har en detektert bevegelse, vil ha en vektor som viser distanse og retning punktet har forflyttet seg. Hvis alle punkt får utslag på en forflytning, blir det mulig å finne rotasjonen og bevegelsen til kameraet. (OpenCV, 2016)

Optimalisering

Local optimization benyttes for at programmet skal kompilere raskere ved å påvirke funksjoner i koden eller endre fysiske elementer. Et eksempel er å nedgradere resolusjonen til bildet slik at posisjoneringsalgoritmen detekterer færre features. Dette gjør at algoritmen bruker mindre tid på å prosessere bildene, men fører også til en reduksjon i nøyaktigheten. Det finnes også andre metoder for å øke hastigheten. Ved å oppgradere prosessor eller grafikkort kan det øke hastigheten for deteksjon og sammenligning av features, uten å ofre kvaliteten på bildene. Økning i prosessorkraft gjør at posisjoneringsalgoritmen kan gjennomføre flere antall målinger per sekund.

4.4 Posisjoneringsalgoritmen - Visual SLAM

MatLab

MatLab er et program som er særlig utviklet for forskere og ingeniører med fokus på analysering og design av systemer. MatLab er et matrisebasert programmeringsspråk som gjør det mulig å visualisere matematiske beregninger. (MathWorks inc., 2022)

I denne delen av oppgaven har monocular visual SLAM posisjoneringsalgoritme fra MatLab blitt testet og anvendt i de forskjellige miljøene på skolen. Algoritmen tar for seg de viktigste begrepene for visuell posisjonering, og har vist seg å fungere godt for datasettene. Posisjoneringsalgoritmen har blitt modifisert for å passe til miljøene datasettene er hentet fra. Se **Vedlegg B** for flere detaljer om disse modifikasjonene.

Kalibrering GoPro

Alle datasett benyttet til monocular SLAM har blitt kalibrert. Dette har også blitt gjort gjennom MatLab, som har en egen applikasjon for å kalibrere datasett. Resultatet blir lagret som en .mat fil og gir posisjoneringsalgoritmen tilstrekkelig informasjon om verdier for fokalpunkt (fx, fy) og prinsipal punkt (cx, cy) til å håndtere forvrengninger i bildet. Kalibreringen har blitt gjennomført med et kalibreringsdatasett. Dette er et A4 ark med svarte og hvite ruter skrevet ut på et papir og festet til et bord for å sørge for at det

ligger helt flatt. At arket ligger helt flatt uten bølger er essensielt for å sikre at kalibreringen blir så god som mulig. Bølger, uskarpheter eller skjerming av rutene i kalibreringssettet vil gi en stor feil når posisjoneringsalgoritmen kjører gjennom bildestrømmen fra datasettet. Bildene ble tatt med samme kamera og har samme oppløsning som datasettene. Se **Vedlegg B** for flere tekniske detaljer vedrørende kalibrering.

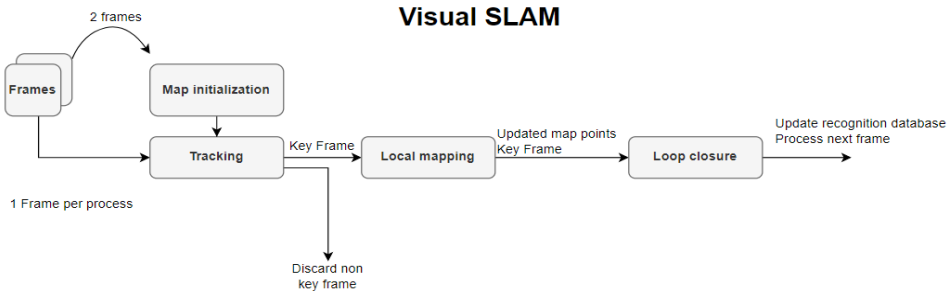
ORB feature matching

ORB feature matching har blitt implementert i visual SLAM for å kartlegge forflyttelse. ORB er en feature matching algoritme som ble utviklet som et raskere og mer robust alternativ til SIFT(Scale-Invariant Feature Transform) og SURF (Speeded-up Robust Features). (Rublee, et al., 2011)

Metoden er utviklet med inspirasjon fra to kjente feature matching algoritmer, FAST og BRIEF. ORB henter inspirasjon fra hjørnedetektoren til FAST og benytter denne til å hurtig identifisere interessepunkter i et bilde. ORB benytter også descriptoren til BRIEF som brukes til å beskrive utseende til featurene på en kompakt binær form. ORB er særlig populær på grunn av hastigheten, nøyaktigheten og robustheten. Dette har gjort den godt egnet til bruk i datasettene fra skolens miljøer.

4.4.1 Pipeline Visual SLAM

For å enkelt forklare hvordan MatLab algoritmen er bygget opp, har pipeline for algoritmen blitt brukt.



Figur 4.7 – Pipeline visual SLAM (Mathworks inc., 2022)

Initialisering av kartet

Programmet starter ved å initialisere et kart av 3D punkter hentet fra 2 bilder av videoen. 3D punktene som oppstår og kamera sin relative posisjon til disse punktene, er kalkulert ved hjelp av triangulering basert på feature matchingen ORB gjennomfører av 2D features fra de initielle bildene. Initialisering av kartet er et svært viktig steg da det påvirker nøyaktigheten til det endelige SLAM resultatet. De initielle ORB feature punktene blir funnet gjennom å matche features mellom de to første bildene som blir tatt. Etter at korresponderende features er definert, blir de punktene med flest descripotrer plottet som 3D punkter i det lokale kartet.

Forflytning i systemet

Tracking går gjennom hvert bilde i datasettet for å finne key frames. Key frames er bilder som inneholder features med gode descriptors. Dette er kjennetegn med mange beskrivelser av hvordan punktet har blitt observert av feature detektoren. For at et bilde skal klassifiseres som et key frame, er det nødt til å følge to kriterier. Det første kriteriet er at bildet må inneholde nok visuell data om features som finnes i bildet sammenlignet med tidligere key frames. Det andre kriteriet er at det nye bildet som blir tatt er nødt til å ha nok korresponderende features med nabobildene i bildeserien. De bildene som ikke tilfredsstiller kriteriene, blir forkastet. (Mathworks inc., 2022)

Det er key frames som blir brukt til å kalkulere et estimat om hvor systemet befinner seg. Hvert bilde blir gjennomgått av ORB som finner nye features i bildet og sammenligner disse gjennom feature matching med korresponderende features i forrige bilde som har kjente map-points. Deretter blir camera pose estimert, som er vinkelen kamera har relativt til objektet som har blitt detektert. Så finner systemet nye features, og disse sammenlignes på nytt for at camera pose skal gjennomføre en bevegelse. Algoritmen sender deretter map points fra det lokale mappet og for å lete etter ytterligere likheter og oppdaterer igjen camera pose. Til slutt bestemmer Tracking om bildet er et key frame, hvor dette blir sendt videre til Local Mapping, eller om tracking skal gå videre til neste frame og gjenta prosessen.

Det lokale kartet

Local mapping er satt sammen av alle features som er hentet fra key frames. Når et nytt key frame er funnet, vil attributtene fra dette bildet legges til og oppdatere features i det lokale kartet. Nye map points blir definert ved å triangulere ORB features fra et key frame til de andre. For hver feature som ikke får en match, søker feature matching gjennom de andre key frames etter andre features som ikke får en match. Punktene blir plottet som 3D punkter relativt til der systemet oppdaget disse i rommet.

Optimalisering

Loop detection blir gjennomført av bags-of-words tilnærming. Dette innebærer at de beste kjennetegnene blir samlet i en "bag", lagret og brukt for å sammenligne kjennetegn ved andre bilder for å identifisere om dette settet med kjennetegn har blitt identifisert tidligere. (Mishra, 2020)

Resultatet etter implementasjonen av loop closure lagres i en database. I databasen lagres også visuell beskrivelse av bildet i kartet basert på samlingen av ORB features. Loop Closure fungerer ved at algoritmen tar et key frame og forsøker å benytte dette for å lokke loopen. Loop closing gjøres ved å identifisere andre key frames som er visuelt like nåværende key frame. Bildet blir merket som en key frame candidate hvis det ikke er tilknyttet forrige key frame, og hvis nabobildene også er loop closure kandidater. Når en loop candidate er funnet, blir den relative posisjonen mellom loop kandidater kalkulert.

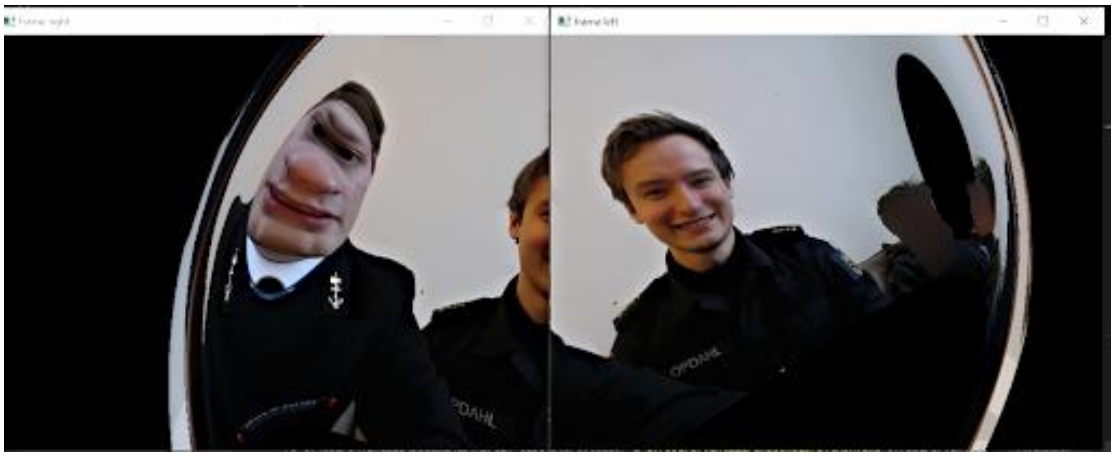
5 Resultater

For å undersøke om metodene oppnår kravene til et nytt posisjoneringssystem, har det blitt gjennomført en rekke tester fra ulike miljø ved skolens område. Resultatene viser frem de ulike stegene for begge metodene, før resultater fra gjennomføring på forskjellige områder blir diskutert.

5.1 Resultater fra Visual Odometry

5.1.1 Kalibrering Logitech

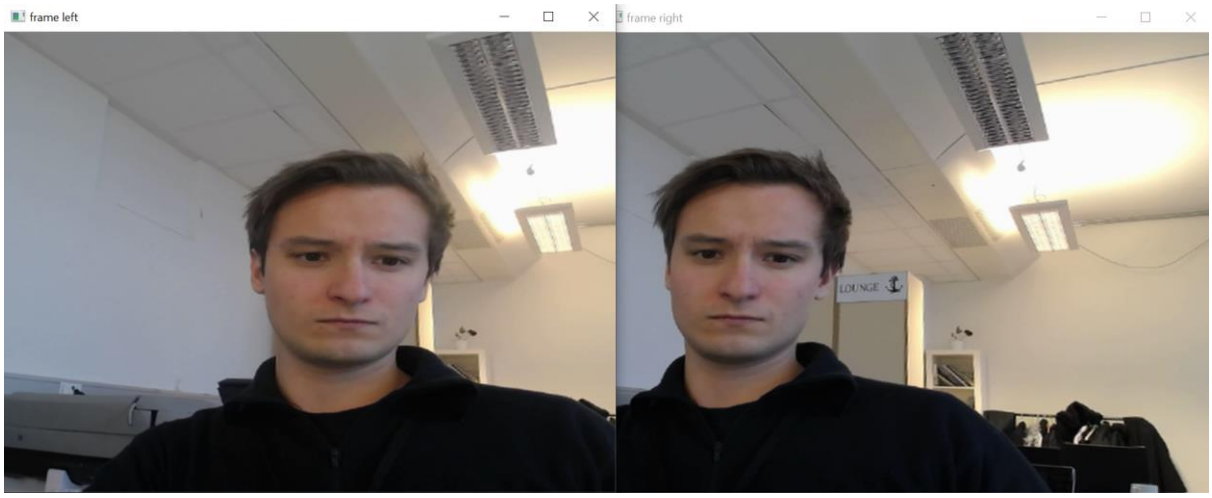
For å oppnå nøyaktig kalibrering av datasettet, var det viktig å benytte mange kallibreringsbilder fra ulike vinkler med god oppløsning. **Figur 5.1** viser resultatet fra et datasett der disse parameterene ikke har blitt tatt hensyn til.



Figur 5.1 – Resultat fra dårlig kalibrering

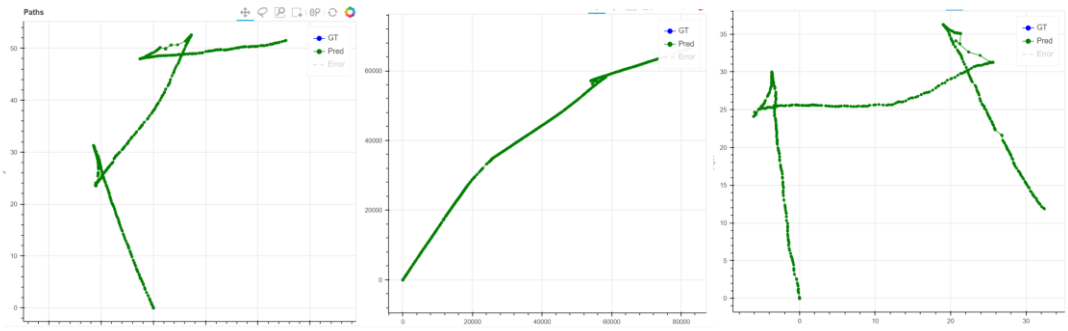
Resultatet viser hvordan en dårlig gjennomføring av en kamerakalibrering kan bli. Det er tydelige forvrengninger og kantene på bildet er brutt og vrengt. I denne testen ble kun

ett bilde benyttet til å kalibrere etter. Dette fungerte destruktivt mot posisjoneringsalgoritmen og skapte mer forvrengning enn et bilde som ikke ble kalibrert.



Figur 5.2 – Resultat fra tilstrekkelig kalibrering

Figur 5.2 viser resultatet fra en bedre kalibrering hvor flere bilder ble brukt. Forvrengningen ble mindre og kameraene var bedre korrigert i forhold til hverandre.

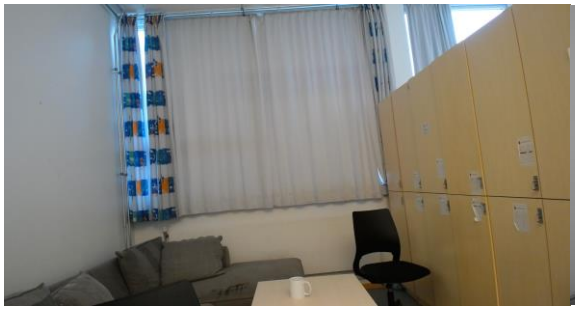


Figur 5.3 – Resultat fra kalibrering

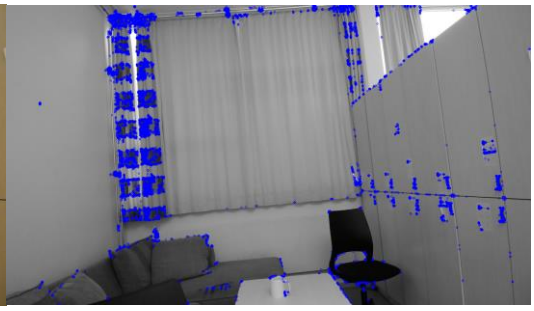
For å vise effekten av kalibrering, ble det benyttet et datasett fra samme lokasjon og eneste endringen mellom målingene var kalibreringen. Til venstre i **Figur 5.3**, er testsettet som kom sammen med koden illustrert. Ruten i midten i **Figur 5.3** kommer fra kalibreringen i kalibreringskoden i oppgaven. Til høyre i **Figur 5.3** er den beste kalibreringen oppnådd med manuell justering i henhold til **Kapittel 4.3**.

5.1.2 Feature detection

For å demonstrere feature deteksjonen, har et bilde av klasserommet blitt prosessert av FAST algoritmen. De blå sirklene er features som ble detektert og godkjent som gode nok punkt. Disse punktene vil bli brukt senere i koden til å beregne systemets bevegelse i forhold til omgivelsene.



Figur 5.4 – Resultat før FAST og grayscale



Figur 5.5 – Resultat etter FAST

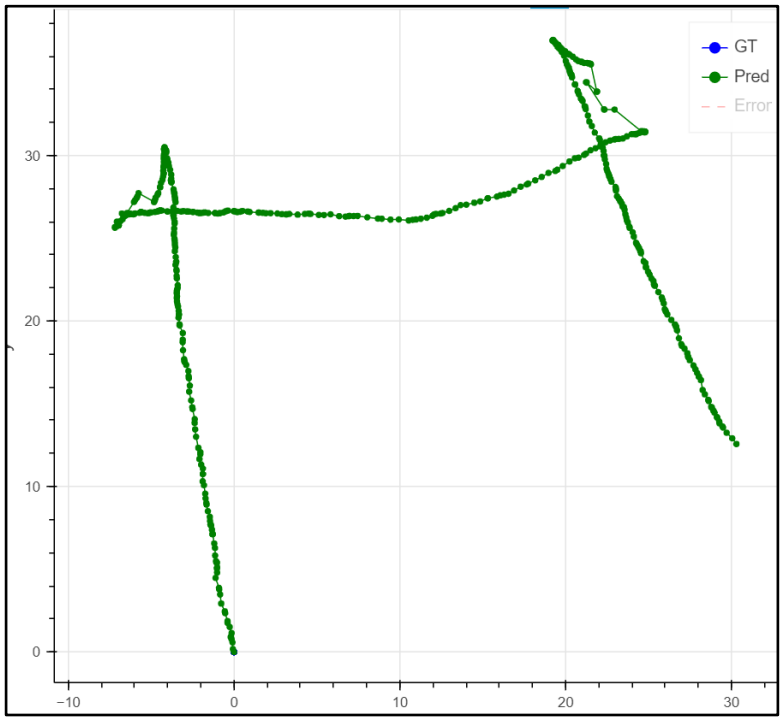
5.1.3 Forflytning

Et estimat av hvor mye systemet beveger seg i forhold til miljøet, blir gjennomført med hjelp av optical flow. **Figur 5.4** demonstrerer hvordan optical flow prosesserer bilder. Det legges til mange punkter i et bilde og hvert punkt har en tilknyttet vektor som tilsvarer mengden bevegelse som oppfattes. I **Figur 5.5** beveges kun armen frem og tilbake, mens resten av kroppen står i ro. Resultatet viser at rundt armen ble utslaget til bevegelsesvektorene stor, mens resten av kroppen har ingen utslag. Mellom hodet og armen fikk punktene en verdi, fordi det er der armen har vært før den ble flyttet. Disse vektorene kan også brukes til å estimere kameraets rotasjon slik at det kan måle hvilken retning kameraene peker ut fra start.

Figur 5.6 – Resultat Optical flow

5.1.4 Aula

Ruten som ble gjennomført i dette datasettet kommer av å følge de tre av veggene inne i aulaen. Vendingene ved hvert hjørne i denne testen ble gjennomført ved 90 graders rotasjon på samme posisjon i rommet.

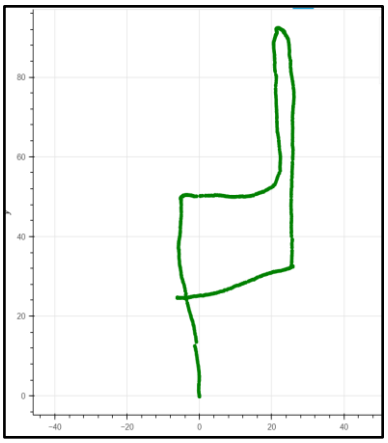


Figur 5.7 – Visual odometry gjennomføring i aula med de 100 beste features for hvert bilde.

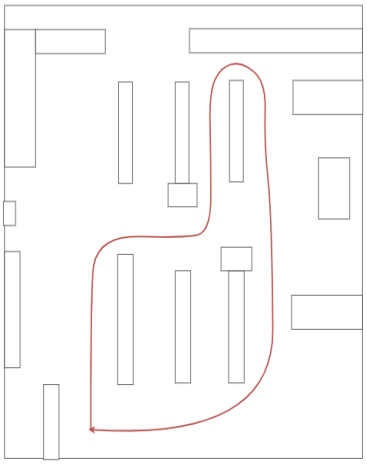
Ved å se på strekningene så ligger ikke posisjoneringspunktene perfekt etter hverandre, men har små hopp frem og tilbake. En kombinasjon av mangel av features og større avstander, forårsakte at posisjoneringsalgoritmen ikke klarte å beregne eksakte forflytninger mellom bildene. Konsekvensen endte med at strekningene får en konstant bevegelse mot venstre eller høyre. I svingene hoppet kartleggingen veldig, som er en effekt som skapes av stereo kameraet. I svingen beveget ikke kamera én på seg, mens kamera to får en bak-over pendelbevegelse.

5.1.5 Bibliotek

Biblioteket er et smalt område med mange særtrekk som bokhyller, kontorstoler og bord. Den planlagte ruten gjennom biblioteket inneholdt flere svinger og en blanding av korte og lange strekninger. Ruten fra visual odometry er visualisert i **Figur 5.8** og sammenlignet med illustrert rute i **Figur 5.9**. I motsetning til aula ble bevegelsene gjennom svingene gjennomført i denne testen var roligere og hadde bedre flyt. Resultatet inneholder flere tettere punkt og har en tilnærmet lik geometri som sann posisjon.



Figur 5.8 – Visual odometry gjennomføring på biblioteket med de 100 beste features fra hvert bilde



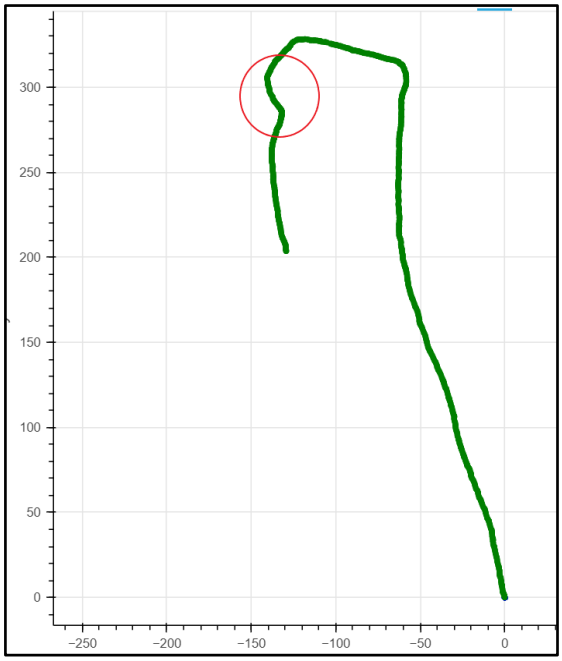
Figur 5.9 – Illustrasjon av geometrien til biblioteket. Rød stripe er forflytning.

Startpunkt og sluttpunkt skal være på samme plass, men algoritmen oppfattet første strekning som lengre enn den var og den parallelle strekningen på vei tilbake til startpunktet ble kortere. I starten av ruten passeres en bokhylle til venstre for systemet, og når den

passeres kan mange features byttes ut samtidig. Dermed oppfattet optical flow at posisjoen endret hurtig. I svingen vist øverst i **Figur 5.9** gjorde systemet en raskere rotasjon som førte til motion blur i bildene. Dette forårsaket at strekningen etter denne svingen fremsto som kortere enn den faktisk var. Det har også oppstått motion blur i den siste svingen nederst i **Figur 5.9**. Dette gjorde at siste strekningen ble forortet enda mer på vei tilbake til utgangspunktet. Gjennom disse to svingene så gikk venstre kamera ut av fokus og høyre kamera opplevde motion blur. Dette medførte at svingene ikke ble kalkulert riktig. Resten av kartleggingen opplevde ikke motion blur eller andre forvrengninger, og er derfor mer presis.

5.1.6 Kaiområdet

Figur 5.10 illustrerer ruten som ble dannet av datasettet fra kaiområdet. Området i den røde sirkelen i figuren er vist igjen i **Figur 5.11**. Uteområdet var dekket av snø, og avstand mellom objekter som kan benyttes som ankerpunkt er stort. Bevegelsen systemet fulgte er kaikanten, for deretter å ta en sving rundt noen grønne konteinere, og så gått samme strekning tilbake. Resultatet viser at posisjoneringsalgoritmen ikke slutter der den startet, men slutter like etter svingen på vei tilbake til utgangspunktet. Dette er også hvordan sann posisjon avslutter.

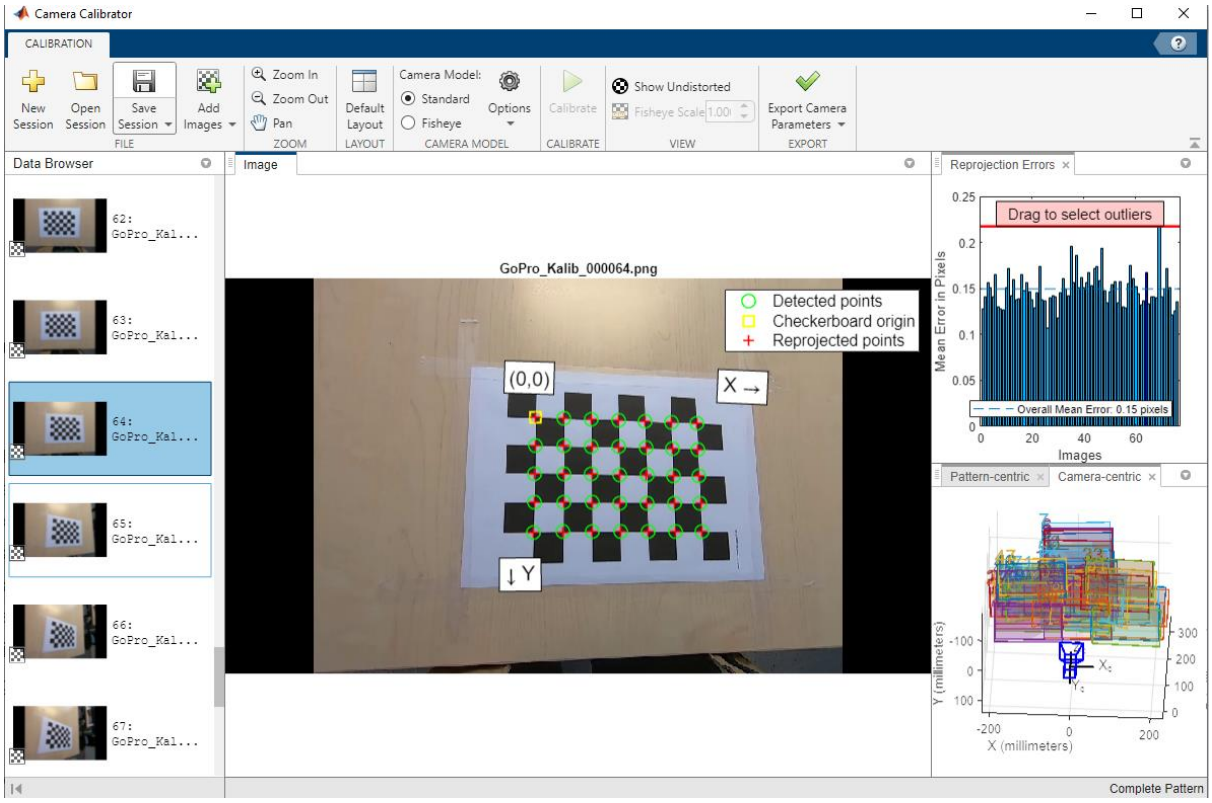


Figur 5.10 – Gjennomføring på kaiområdet med de 100 beste features fra hvert bilde.

Figur 5.11 – Referansebilde fra området med rød sirkel i Figur 5.10.

I **Figur 5.10** var det få kontraster og særtrekk som gjorde at deteksjon algoritmen ikke klarte å finne gode features å følge. Strekningen systemet forflyttet seg over var egentlig lenger, og skulle til slutt danne en firkant. I sirkelen var distansen mellom de to svingene kortere enn det den var i virkeligheten, noe som kan komme av reduksjonen av antall features FAST klarte å detektere. Visual odometry klarer heller ikke å kartlegge svingen like skarp som den skulle vært.

5.2 Resultater fra Visual SLAM



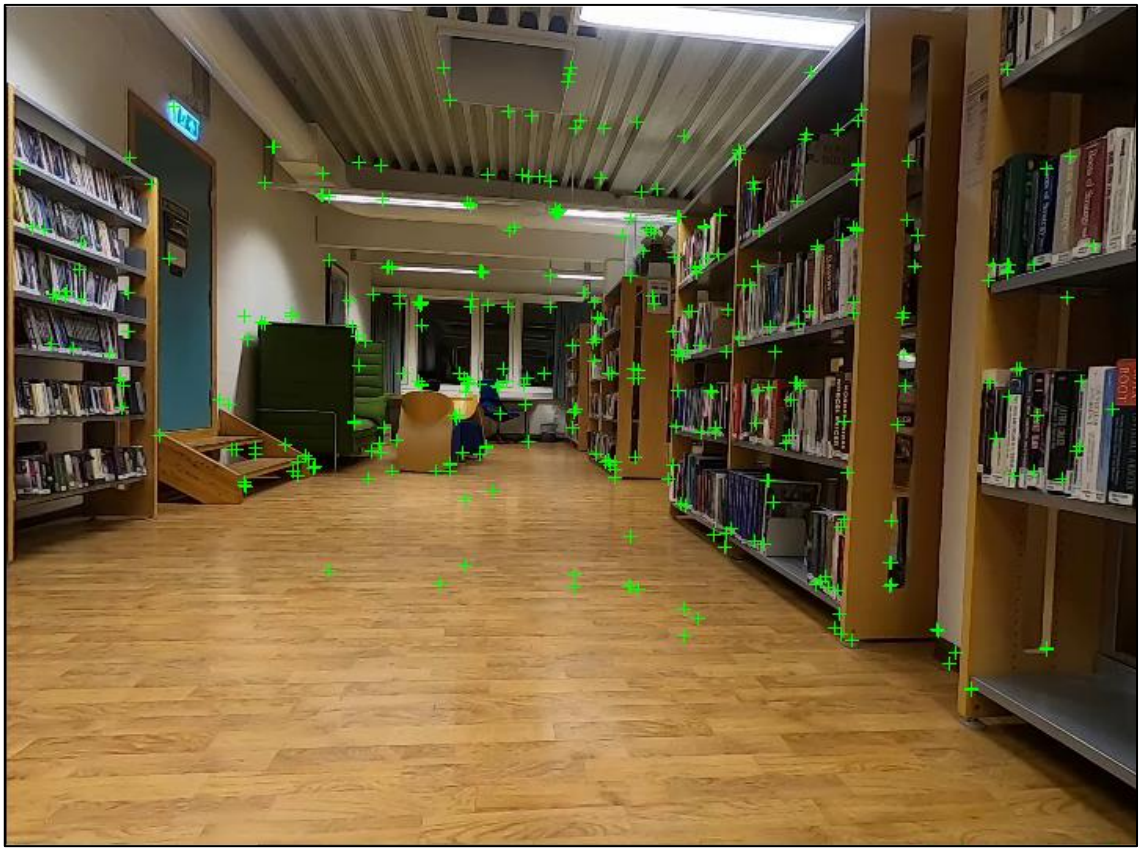
Figur 5.12 – Kalibreringsverktøy for monocular vision.

Kalibreringsverktøyet gjorde det mulig å se hvilke bilder som inneholdt mest feil, slik at nye bilder kan bli tatt fra den vinkelen som forårsaket feilen. Verktøyet gjør det også mulig å zoome inn på bildet for å se hvor godt punktene blir plassert i hvert hjørne av kalibreringsbildet.

Resultatene fra kalibrering viser at «**Chessboard»** kalibreringssettet har en gjennomsnittlig feilmargin på 0.103 piksler og en maks feil på 0.401 piksler.

5.2.1 Feature detection

Som beskrevet i **Kapittel 4.4** så benytter ORB feature detection seg av egenskaper fra FAST og BRIEF for å detektere features i et frame. Vedlagt er testresultater gjennomført i biblioteket som viser hvordan disse featurene så ut i miljøet datasetet ble tatt. Algoritmen var innstilt på å lete etter 2000 feature points i hvert frame og fremmet interessante resultater



Figur 5.13 – Feature detection med ORB

Først så viser resultatet at ORB klarte å gjenkjenne kanter og linjer godt. Helt til venstre i bildet viser resultatet hvordan ORB har tatt tak i alle kantene algoritmen fant på bokhyllen. En annen observasjon er at linjer på gulvet i nærheten av linsen har blitt fanget opp av algoritmen. Posisjoneringsalgoritmen har en funksjon for å justere avstanden ORB leter etter features i. Denne funksjonen stilles inn etter hvor store avstander miljøet har. I figuren har det blitt valgt å ikke filtrere bort disse punktene for å vise frem poenget.

5.2.2 Forflytning

Posisjoneringsalgoritmen viser features som har blitt matchet med hverandre ved å trekke en linje fra en feature med descriptor i første bildet til den korresponderende featuren med lik descriptor i det neste bildet. ORB illustrerer dette ved å trekke en linje mellom de feature punktene algoritmen har sammenlignet.

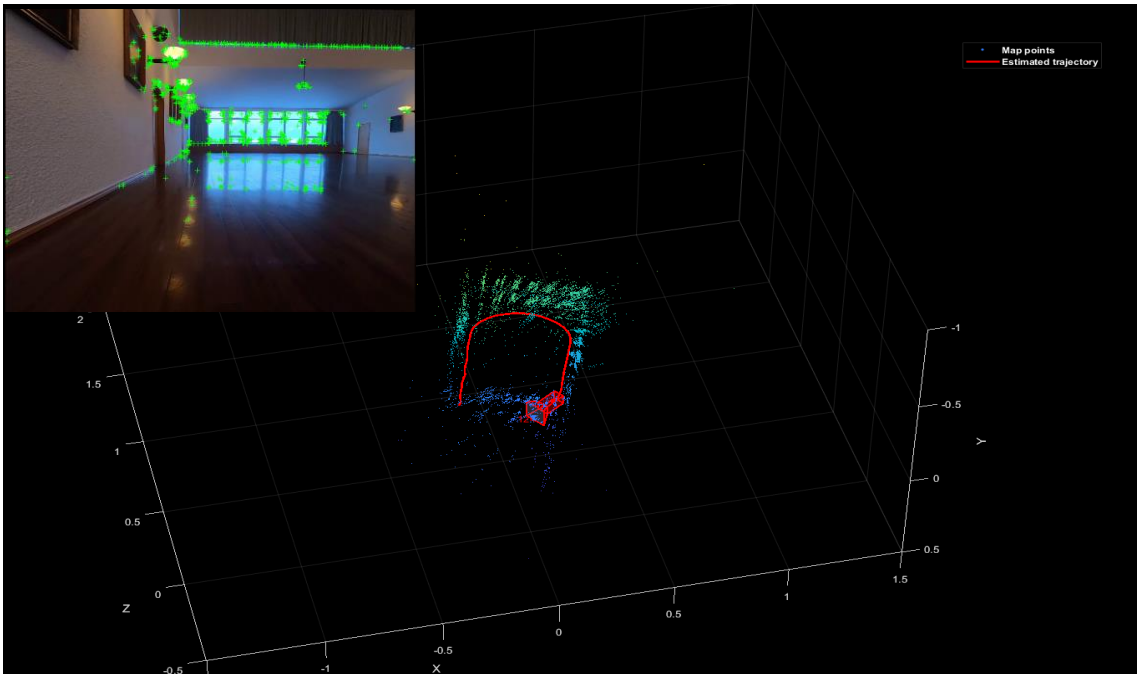
Dersom noen linjer krysser andre linjer, er det en indikasjon på ugyldig data, og parameterne er nødt til å bli justert og testen må gjennomføres på nytt. Det eneste unntaket fra denne regelen er dersom bildet er rotert 180 grader. Da skal alle linjene krysse gjennom samme punkt for at resultatet skal aksepteres som gyldig. Testen var gjennomført med 2000 numPoints for å tydeliggjøre streker mellom features.

Figur 5.14 – Feature matching med ORB

Resultatet i **Figur 5.14** viser feature matchingen. Det er ingen tydelige kryss mellom linjene, noe som tyder på at feature matchingen fungerte optimalt. ORB ser ut til å ha detektert punkter langs venstre veggen og i enden av rommet tidlig i gjennomføringen. Dette er ønskelig, da feature matchingen tidlig har identifisert stort sett hele rommet og kan oppdatere det lokale kartet gjennom hele gjennomføringen.

5.2.3 Aula

Den første testen ble gjennomført inne i skolens Aula. Området er åpent, og systemet kan nesten se hele rommet fra første måling. Derimot har rommet relativt få detaljer posisjoneringsalgoritmen kan finne feature points fra. Miljøet har noen svinger som er interessante å teste. Systemet vil gjennomføre rolige svinger for å sørge for at data som samles inn er tilstrekkelig for nøyaktig posisjonering i området. Testen passer godt til å vise hvordan et åpent miljø med store avstander påvirker posisjoneringsalgoritmen. Rommet er 21 meter langt og 10 meter bredt.

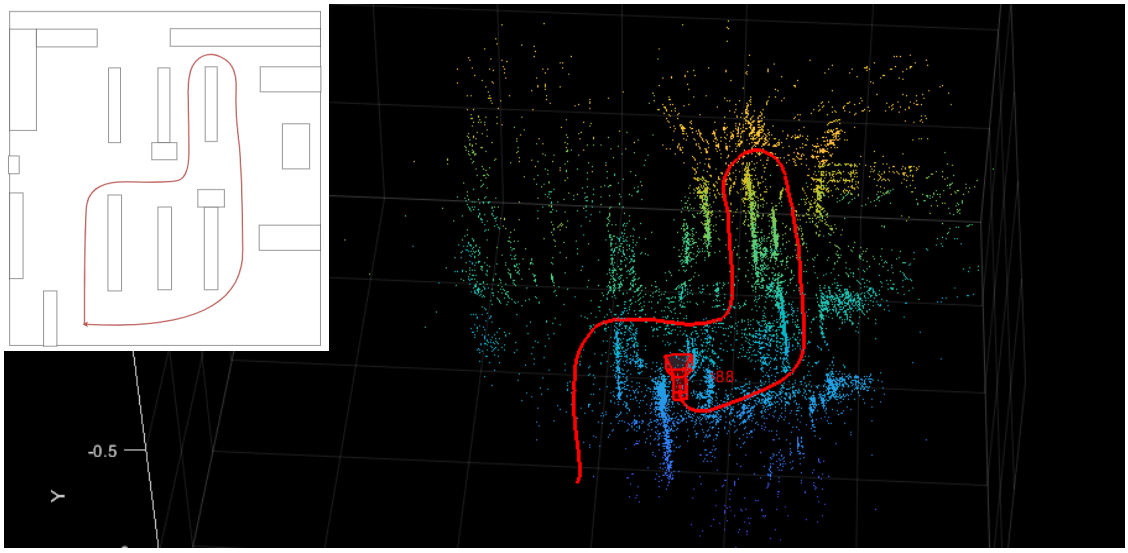


Figur 5.15 – Visual SLAM resultat fra aula

Resultatet viser tydelig at avstander opp til 21 meter ikke var en utfordring. Resultatet ble gjennomført med rolige svinger slik at posisjoneringsalgoritmen kunne oppdage nye features over lengere tid for å sammenligne disse bedre. De rolige svingene ser man igjen i resultatet, og markerer tydelig hvordan systemet har bevegde seg gjennom miljøet. Systemet klarte også å forstå geometrien til området. Resultatet viser at sideveggene er litt lenger enn endeveggen og det stemmer overens med rommets faktiske geometri.

5.2.4 Bibliotek

Den andre testen har blitt gjennomført på skolens bibliotek. Området er trangt og inneholder veldig mange detaljer, som posisjoneringsalgoritmen kan ta utgangspunkt i. Miljøet har flere svinger og passer spesielt til testing av robusthet, i et terreng hvor systemet ikke kan se hele området fra første måling. Systemet måtte derfor kartlegge nye map points undervegs som ikke kunne sammenlignes med tidligere map points. Systemet har gjennomført ruten med rolige, slake svinger hvor posisjoneringsalgoritmen har fått tid til å detekter features i det nye miljøet.

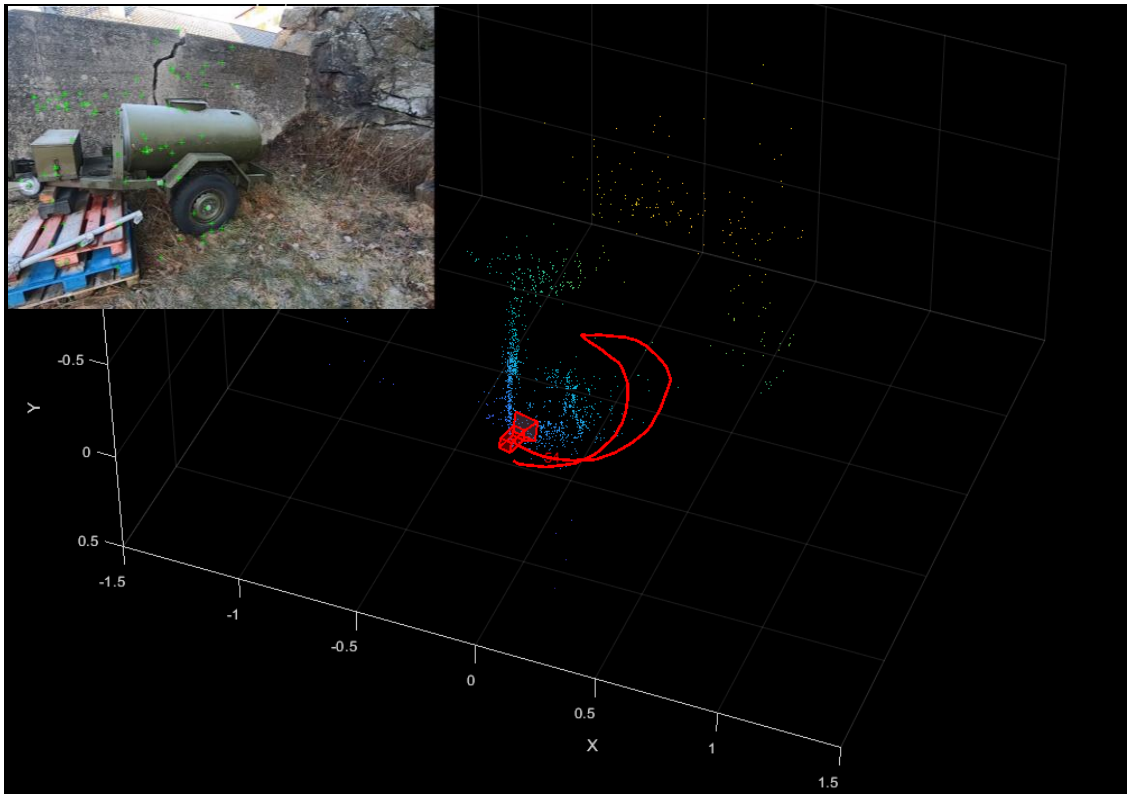


Figur 5.16 – Visual SLAM resultat fra biblioteket med referansebilde

Resultatet viser i stor grad at systemet viste hvor det befant seg gjennom majoriteten av målingen. Samtidig viser resultatet også et interessant moment ved at systemet ikke klarte å finne tilbake til start. Feature matching i områder som inneholdt få map points, vil resultere i at visual SLAM bruker lenger tid på å identifisere feature points og key frames. Ved brå bevegelser ville systemet også få en større grad av motion blur. Dette er bilder som ikke inneholder tilstrekkelig mange features som gjorde at posisjoneringsalgortimen forkastet bildene. Dette påvirket nøyaktigheten, og gjorde at systemet trodde det befant seg lenger bak i ruten enn det den faktisk var. Med andre ord så ville ikke systemet finne tilbake til startposisjon. Bilder som skulle fortalt systemet om forflytning inneholdt ikke god nok kvalitet, og kunne derfor ikke benyttes til å kalkulere forflytning. Dette gjorde også at systemet ikke kunne gjennomføre loop closure for å optimalisere posisjonen sin. Resultatet viser at bokhyllene ble markert som tydelig rette linjer med samlinger av features. Dette er en forventning da bokhyllene har mange kanter og linjer som feature matching kunne finne kjennetegn fra.

5.2.5 Kaiområdet

Testen ble gjennomført på skolens kaiområde. Området er åpent og inneholder flere forskjellige større objekter Visual SLAM kan undersøke som en murvegg, fjellvegg og en militær vanntank. Kamera ble pekt på den militære vanntanken gjennom hele ruten, for å teste hvor godt systemet kunne beregne posisjonen sin med utgangspunkt i et annet objekt.



Figur 5.17 – Visual SLAM resultat fra kaiområdet med referansebilde

Resultatet viser at Visual SLAM på en god måte klarte å følge vanntanken. Posisjoneringsalgoritmen startet med å definere punkter på fjellveggen. Disse punktene ble utgangspunktet for forflytningen. Etterhvert som systemet begynte svingen, ble vanntanken og murveggen bak vanntanken mer interessant, og features ble kartlagt fra ulike vinkler. Tydelig endring i resultatet skjedde når systemet økte avstanden til vanntanken. Da ble features kartlagt med en annen avstand og fikk flere beskrivelser av punktet. Ved å øke avstanden til vanntanken oppdaterte posisjoneringsalgoritmen geometrien til kartet basert på triangulering. Resultatet viser at murveggen ble definert som en tydelig rett strek bak objektet, noe som var å forvente. Resultatet viser også at fremsiden av vanntanken fikk tydelige 3D punkter, mens baksiden ikke får det. Dette er også å forvente, da systemet ikke fikk tilstrekkelig tid til å observere vanntanken ovenfra eller fra baksiden.

6 Drøfting

I denne delen av oppgaven drøftes resultater og andre relevante temaer. Drøftingen tar for seg valg av sensorer, maskinvare, hensyn for datasettene og forskjellen i resultater fra de forskjellige posisjoneringsalgoritmene.

6.1 Maskinvare

I oppgaven har det blitt benyttet to kamera og en lab datamaskin som er utlevert labutstyr fra SKSK. I tillegg har det blitt benyttet et GoPro kamera for visual SLAM. Den største begrensningen for ytelsen av posisjoneringsalgoritmene har vært datamaskinen. For at posisjoneringsalgoritmene skal kunne fungere optimalt, ville oppgaven trengt høyere prosessorkraft enn det maskinen i oppgaven har tilgjengelig. Det tok betraktelig lengre tid å prosessere på lab datamaskinen enn på en datamaskin med moderne prosessor.

Et forsøk ble gjennomført på å kjøre koden til visual odometry i sanntid, noe som ga resultatene en bildehastighet på 3 FPS. Maskinen er laget for å prosessere enkle simuleringer og prosesser, men store prosesser som posisjoneringsalgoritmene krever en bedre prosessorkraft og lagringsplass.

To Logitech webkamera ble brukt i visual odometry. Begrunnelsen for å bruke kamera med lavere kvalitet uten stabilisering, var for å vise at dersom oppgaven kan løses med simple kamera, kan det mest sannsynlig også løses med et avansert kamera. Visual SLAM har tatt i bruk et GoPro kamera, ettersom GoPro har en innebygd stabiliseringsfunksjon. Årsaken til at GoPro ble benyttet i visual SLAM og ikke i visual Odometry, var fordi algoritmen til visual SLAM krevde høyere grad av stabilitet i bildene for nøyaktig posisjonering enn hva algoritmen til visual odometry krevde. Gjennom testfasen ble det tydelig at både Visual Odometry og visual SLAM måtte senke kvaliteten på bildene for at datamaskinen skulle klare å prosessere datasettene. Dermed ble de største forskjellene på Logitech webkamera og GoPro, at stabiliseringsfunksjonen til GoPro gjorde at flere features kunne bli oppdaget i bildene. I tillegg ble det mindre grad av motion blur i datasettet.

6.2 Monocular og stereo vision

Monocular Vision

Visuell posisjonering krever at det blir tatt flere bilder per sekund og benyttes langsomme bevegelser. Som eksempel er visual SLAM algoritmen kalibrert for 24 FPS. Det betyr at datastrømmen som sendes fra sensor til posisjoneringsalgoritmen ofte tenderer til å bli svært tung å prosessere. Ved å kun benytte seg av ett kamera i stedet for to, så ofret monocular SLAM til dels dybdeforståelse for å redusere prosesseringsbelastningen. Resultatene fra posisjoneringsalgoritmen viser dessuten at monocular vision klarte å triangulere feature points som algoritmen fant ved å sammenligne bildet før i tid med sanntidsbildet på en god måte.

En erstatning til monocular ville vært Stereo SLAM. Ved Stereo SLAM ville systemet fått en bedre følelse av miljøet det befinner seg i, men dette ville kostet ekstra prosesseringskraft. Datamaskinen benyttet i denne oppgaven, har verken nok lagringsplass eller prosesseringskraft til å gjennomføre disse testene på en tilstrekkelig måte. For å teste SLAM med stereo vision vil det være nødvendig å benytte en kraftigere prosessor, bedre lagringsplass og økt lokalt minne (RAM). Dette vil øke hastigheten systemet kan kjøre gjennom datasettet, og vil være helt nødvendig dersom SLAM skal klare å prosessere i sanntid.

Stereo vision

En stor fordel med å velge stereo vision har vært dybdeinformasjonen som kunne bli anvendt i posisjoneringsalgoritmene. Det største problemet med å trekke ut features fra et 2D bilde er å vite posisjonen til punktet i 3D verdenen. Ved å benytte to kameraer ble det enklere å regne ut den relative 3D posisjonen til features, som førte til nøyaktigere målinger. Et monokulært kamerasystem har en stor utfordring med features som er nært sentrum av bildet, fordi forflytning midt i bildet ofte er relativt liten på rette strekninger.

Ulempen med stereovision har vist seg å være den ekstra mengden med bilder skal prosesseres. I oppgavens tilfelle, ble antall bilder doblet for stereo vision i forhold til monocular vision. Dette er også grunnen til at bruk av flere kameraer er uaktuelt, ettersom det ikke bidrar nok til resultatet i forhold til mengden ekstra data. Eneste grunnen til å vurdere

bruk av flere kameraer vil være å bygge et fartøy som har flere stereo vision oppsett rundt seg fartøyet.

6.3 Kalibrering

I løpet av testingen av posisjoneringsalgoritmene ble det gjort erfaringer om at kalibrering har stor innflytelse på resultatene. Ved for mye forvrengning blir det utfordrende for posisjoneringsalgoritmene å følge forflytningen systemet har gjennom miljøet. Når kalibreringen var tilpasset systemet, presterte algoritmene bedre. En tendens som ble erfart med systemene var at de med dårlig kalibrerte data, antok at rommet ble mindre for hver sving, helt til systemet sluttet å kartlegge bevegelse. Problemet oppsto som følge av at kamera benyttet seg av andre fokalverdier og prinsipalverdier enn hva som var innført i posisjoneringsalgoritmene. I tillegg til dette inneholdt referansebildene, som ble brukt til kalibreringen, motion blur som førte til stor usikkerhet i gjennomføringene. Erfaringer som ble gjort var at referansebildene som blir tatt er nødt til å være så nøyaktige som mulig for å minske feilmarginen fra en perfekt måling til en turbulent måling. Det viste seg at referansebildene trenger å være av høyest kvalitet for at kalibreringen skal kunne oppfatte de rette parametrene. Dersom det er dårlig kvalitet på referansebildene, vil testbildene bli ubrukelig.

6.4 Klimatiske forhold

Majoriteten av testing som ble utført på posisjoneringsalgoritmene ble gjennomført innendørs. Fordelen med dette er at klimatiske varianser som lys, tåke og skygger ikke vil være utslagsgivende for resultatet. I løpet av testperioden var været varierende med sol, regn, overskyet og snø. Med færre klimatiske faktorer kan testing gjennomføres konsistent over flere dager uten å måtte ta hensyn til disse faktorene.

En negativ side med å teste systemet i et miljø som forholder seg uendret, er at testene blir ensidige. Dette gjør at metodene ikke testes i miljøer som kan påvirke systemet i stor grad, som tåke og regn. Dersom testene hadde blitt gjennomført i flere forhold, kunne robustheten til systemet i større grad blitt utforsket.

6.5 Resultater Visual Odometry

Resultatene fra visual odometry viser hvordan posisjonerigsalgoritmen behandlet ulike miljø. Robusthet og presisjon er de viktigste egenskapene systemet må beherske for å kunne være et pålitelig posisjoneringssystem. Systemet har kun benyttet visuelle sensorer for å teste selvstendighet og et kart over forflytning har oppfylt kravet om kartlegging.

Aulaen utfordret systemet som et monotont miljø som inneholder få særtrekk og store avstander mellom features. På kartleggingen blir ikke banen helt rett og det er en liten knekk etter første sving, en strekning med nesten kun vinduskarmer til venstre for kamera. I tillegg viser resultatet at det var få kjennetegn å detektere rett foran kameraet, som antyder at for få features med god kvalitet ble funnet. Dette skaper avvik fra den faktiske forflyttelsen. Denne observasjonen gjaldt for hele gjennomføringen der resultatet fra **Figur 5.7** viser at kartleggingen har en bølgende form langs strekningene. I hjørnene av aulaen blir det en veldig skarp sving, og dette skaper en spesiell bane. Det antas at denne effekten kommer av stereo vision oppsettet. Under rotasjonen så sto det ene kameraet stille, mens det andre roterte bakover med en pendelbevegelse i forhold til det andre. Dermed trodde systemet at banen går bakover samtidig som det roterer. Dette var en adferd som spesielt ble vist i datasettet fra aula ettersom rotasjonen på systemet i de andre ikke er like skarp.

I biblioteket ble ruten gjennomført med en langsommere flyt, som ga flere bilder, og svingene var jevnere slik at kameraene ikke fikk samme rotasjon som i aula. Miljøet inneholdt veldig mange flere særtrekk som ga store muligheter for FAST algoritmen til å detektere features. I motsetning til aulaen ble ruten mer presis og ruten driftet ikke like mye. Fremdeles er det områder som ikke er korrekt i forhold til den sanne forflytningen. Starten og slutten skulle vært på samme punkt, men systemet oppfattet den første strekningen som mye lengre enn det den egentlig var. I tillegg ble det et lite hopp i kartleggingen som kan forklares av mange features som ble byttet ut samtidig. Startområdet startet smalt ved bokhyllen veldig nært kameraene og gikk fort over til et relativt åpent område. Dette kan ha forårsaket at mange av de beste featurene ble byttet ut.

Kaiområdet ved skolen viste i resultatet noen av de samme utfordringene som gjennomføringen i aula. Det var få særtrekk langs kaiområdet og det var store avstander. Bygninger, kraner og båter var også tildekket av snø under innsamlingen av datasettet. Likevel klarte systemet å lage en presis rute frem til rød sirkel som vist i **Figur 5.10**. I henvist område så ser kameraet kun kanten av kaien og havnebassenget til skolen, og fikk dermed vansker for å detektere features. Konsekvensen ble en kortere forflytting enn sann forflytning. Alle objekter og området forbi kaikanten ble mer usikre enn objekter identifisert tidligere i gjennomføringen, fordi pikselstørrelsen generaliserer mer av miljøet på større avstander enn for miljøet på kortere avstander. Dette betyr at særtrekk blir vanskeligere å detektere på avstand. En løsning for denne utfordringen kunne vært å øke bildekvaliteten. Dette ville ført til at objektene på avstand ville inneholdt flere piksler, og FAST kunne funnet flere features ut ifra disse pikslene.

Avstand er den største utfordringen til systemet ettersom det er den mest avgjørende faktoren for presisjon. Spesielt merkbart var problemet på kaiområdet og i aula hvor systemet estimerte bevegelsen dårlig når features var langt fra kameraene.

6.6 Resultater Visual SLAM

Gjennomføringene i de tre forskjellige miljøene gir oppgaven dybde til å drøfte hvordan forskjellige forhold påvirker visual SLAM. Det er tydelig at noen faktorer påvirker posisjoneringsalgoritmen tyngre enn andre. Eksempler på slike faktorer er *avstand, lys, areal* og *bevegelse*.

Eksempelet fra **Kapittel 5.2.3** hvordan systemet reagerer på et åpent areal med få kjennetegn å posisjonere seg etter. Dette resulterte i et kart der vegger og hindringer er delvis diffuse på hvor disse starter og slutter. Det var ikke nok features på de hvite veggene til at algoritmen klarer å kartlegge disse like godt som i de andre gjennomføringene. Areal hadde derimot mindre å si i et åpent område, da systemet kunne se nesten hele miljøet allerede fra første bilde. Dette gjorde at map points som har ble funnet tidligere i tid, kan detekteres senere i tid og oppdateres, istedenfor å finne mange nye map points.

Til forskjell fra Aulaen så hadde biblioteket veldig mange kjennetegn, men var i gjengjeld et veldig trangt miljø å posisjonere seg i. **Figur 5.16** viser dette på en god måte, da kartet visual SLAM lagde var veldig detaljert, og bokhyller ble markert med tydelige, tette samlinger av features. Derimot mistet systemet posisjonen sin over tid, da posisjoneringsalgoritmen hele tiden måtte kartlegge nye områder systemet tidligere ikke har befunnet seg i. Dette gjorde at systemet ikke fant tilbake til start, som antyder på at systemet trodde det er lenger tilbake i tid enn det faktisk var.

Lys og bevegelse er to helt essensielle parametere som er nødt til å finne sted dersom visuell posisjonering skal være mulig. Posisjoneringsalgoritmene leter etter kanter og konturer i miljøet. Dersom kamera er ute av fokus gjør dette at kanter blir diffuse og til dels smelter sammen med hverandre. Dårlig lys sammen med redusert bildekvalitet vil i tillegg forårsake at noen kjennetegn i miljøet blir uleselig, og presisjonsnivået på posisjoneringsalgoritmene reduseres.

Det er tydelig at posisjonering blir mer nøyaktig desto lenger tid systemet får til å kartlegge features fra et objekt. Det mest nøyaktige resultatet fra SLAM gjennomføringene var objektfølgingen av den militære vanntanken i **Kapittel 5.2.5**.Det som var interessant med dette eksempelet er at systemet fikk lov til å kartlegge miljøet veldig rolig og fra en rekke forskjellige vinkler og avstander. Når miljøet kan observeres på denne måten så vil algoritmen kunne gjenkjenne de samme featurene fra forskjellige vinkler, som gjør at kartet blir mer presist relativt til kamera. Resultatet i **Figur 5.17** viste at dette gjorde posisjoneringen veldig presis. Å følge et objekt og vite avstanden til dette er overførbart til å vite egen avstand relativt til et annet fartøy eller et kjent landemerke. Testen viste at ved å følge et objekt fikk systemet bedre tid til å kartlegge features ved dette objektet, som igjen økte nøyaktigheten.

6.7 Kartlegging

Kartlegging er en av de mest spennende funksjonene til visual SLAM. Ved å kartlegge miljøet systemet har besøkt, kan kartet lagres i en database som fartøyet selv kan benytte seg av senere. Det som er interessant med denne funksjonen er dersom flere fartøy benytter seg av samme måte for å kartlegge havområdene. På denne måten kan kartene deles mellom fartøy og øke robustheten til hvert system. Dette kan gjøres på flere måter. Kartene kan lastes opp på en skyløsning som alle fartøyene kan hente data fra i sanntid. En annen måte å dele kartene på er å laste over alle kartene på en felles database ved kai.

Fordelene ved å laste opp kartet i en skyløsning er at kartet oppdateres kontinuerlig hvis flere fartøyer benytter SLAM. Eksempelvis hvis en bro raser sammen så vil et fartøy som observerer dette oppdatere denne endringen i kartet. Farer ved å benytte en skyløsning er at data som kartlegges over havområdene er tilgjengelig for alle og sårbare for manipulasjon. Derfor kan et bedre alternativ være å lagre kartet frem til fartøyet kommer til kai, før dette kartet lastes over i en database. Denne databasen kan være lokal på marinens base, slik at fartøyer tilknyttet marinebasen kan få tilgang til dataene og skjerme data fra resten av verden. Dette vil være en robust måte ta vare på sensitiv data, men det vil ikke nødvendigvis være like effektivt som en skyløsning.

Dersom et slikt kartleggingsverktøy implementeres som en del av en metode for posisjonering, vil dette gi mulighet for et 3D kart av alle havner, fjorder og havområder et fartøy besøker. Dette kartet vil også oppdateres når et fartøy besøker samme område på nytt ved en senere anledning.

7 Konklusjon

Denne oppgaven hadde som mål å innhente informasjon og teste ulike metoder for visuell posisjonering. For at metodene skulle svare på problemstillingen, ble det satt krav til egenskaper de måtte inneha. Både visual odometry og visual SLAM er gjennomført utelukkende for å prosessere visuell data. I tillegg har metodene kartlagt egen forflytning i form av en linje som forteller om systemets tidligere posisjon. Ulikhetene mellom systemene kommer tydelig frem i hvordan metodene presterer i nøyaktighet og robusthet.

Visual Odometry er presis i miljøer som har mange særtrekk, som kanter og konturer. Resultatene viser derimot at miljø med færre konturer og kjennetegn gjør posisjonen mindre nøyaktig. I tillegg har ikke posisjoneringsalgoritmen en måte å korrigere feil på, slik at ruten får større avvik fra sann rute over tid. Visual SLAM har derimot denne muligheten til å korrigere posisjon over tid ved hjelp av loop closing. Dette optimaliserer posisjonen til systemet og gir systemet kjennskap til geometrien i miljøet. Det er også viktig å poengtere at visual odometry ikke har mulighet til å kartlegge og gjenkjenne områder fra tidligere posisjoner, noe visual SLAM har.

Når det kommer til robusthet, så har resultater fra oppgaven vist at visuelle sensorer ikke vil supplere tilstrekkelig robusthet alene i et miljø med større avstander, større bevegelser eller varierende lysforhold. Derimot har visual odometry og visual SLAM vist at posisjoneringsalgoritmene klarer å selvstendig detektere og forflytte seg uten hjelp av andre lokaliseringsmetoder eller sensorer. Gitt at området har tilstrekkelig lys, begrensede avstander og rolige bevegelser.

For å svare på problemstillingen, så har visual SLAM utfylt kravene til lokalisering bedre enn visual odometry, og er overlegen med mulighet for kartlegging og loop closing. Visual SLAM er derimot ikke tilstrekkelig for å erstatte GPS posisjonering i marinen. Selv med relativt perfekte datasett så strevde visual SLAM med robusthet i forbindelse med faktorer som motion blur og tilstrekkelig lys. I et miljø med mer friksjon og ytre påvirkninger, vil sannsynligheten for feil øke. I oppgaven utelukkes derimot ikke potensiale visual SLAM har i et system sammen med andre sensorer som utfyller svakhetene til systemet anvendt i oppgaven. Ved å benytte flere sensorer integrert i samme kart vil det

gi systemet tilstrekkelig robusthet og nøyaktighet til å utfordre den tradisjonelle GPS posisjoneringen.

8 Videre forskning

I løpet av arbeidet med oppgaven har det blitt identifisert temaer som er aktuelt å forske videre med, og som bygger videre på SLAM. Tre aktuelle tema for videre arbeid, er optimalisering, global kartlegging og sensor fusion.

Optimalisering

En anbefaling for videre arbeid av oppgaven er å modifisere og optimalisere visual SLAM til å kunne prestere bedre. Det er fremdeles mye arbeid rundt kamerakalibrering og optimalisering av prosesser som kan utbedres. Kamerakalibreringen har ikke blitt perfekt, noe som kan være utslagsgivende for nøyaktigheten til systemet. Derfor anbefales det å undersøke om det finnes en bedre løsning for å kalibrere kamera. Maskinvare har også vært en stor utfordring for yteevne av prosesseringen av data. Dette har ført til at bildekvaliteten var nødt til å bli nedgradert. Bedre prosessorkraft vil gjøre det mulig å teste visual SLAM med bilder i sanntid.

Metodevalg for feature detection og matching er et vidt tema som er i konstant utvikling. Metodene benyttet i denne oppgaven er to av mange som benyttes i moderne anvendelse av computer vision. Disse er også de mest oppdaterte metodene som er gratis å bruke. I tillegg er det mulig å optimalisere og justere disse prosessene slik at de passer bedre til et kystnært miljø.

Loop closing funksjonen til visual SLAM ble ikke testet ut. Denne funksjonen vil kunne bistå til å optimalisere posisjonen til systemet og oppdatere geometrien til området. Dersom loop closing fungerer, vil det bidra til å øke robustheten til systemet og posisjonen i testene vil bli mer nøyaktig. Loop closing er også vitalt for å løfte SLAM fra et lokalt kart til å bidra til et globalt kart med tidligere loggførte features.

Global kartlegging

Gjennom lokalisering og mapping av større områder kan havområdene kartlegges på en bedre måte. Dersom flere fartøy benytter seg av samme måte for å kartlegge havområdene, kan disse kartene deles mellom fartøy, og øke robustheten og nøyaktigheten til hvert system som drøftet i **Kapittel 6.7**. Ved å lage et digitalt kart som posisjoneringsalgoritmen kan kjenne seg igjen i, vil dette bidra til at fremkomstmiddelet hurtigere og mer presist kan oppdatere posisjonen sin. På mange måter vil denne automatiske oppdateringen av posisjon fungere som dagens manuelle triangulering av kjente landemerker.

På mange måter ligner disse kartene på puslespill eller en grid, der hver brikke definerer en bestemt plass i verden. Når fartøyet vet sin posisjon på denne gridden, så vet systemet nøyaktig hvor det befinner seg på det lokale kartet samtidig som det vet sin posisjon i verden.

Sensor fusion

I oppgaven har visual SLAM blitt gjort rede for. Denne metoden for posisjonering utelukker alle andre sensorer enn visuelle sensorer. SLAM er et system som har mulighet til å integrere flere sensortyper i samme kartet. Dette konseptet kalles sensor fusion og blir mye brukt til å utvikle selvkjørende biler og andre autonome fremkomstmidler. For selvkjørende biler kan optiske sensorer kombineres sammen med laser og GPS for å gi systemet mest mulig presis posisjon. Visuell sensor holder følge med miljøet rett rundt bilen og kan gi den selvkjørende bilen tilstrekkelig informasjon om eksempelvis linjer i veien. En lasersensor vil så gi systemet informasjon om terrenget på større avstander som bygninger og trær.

Hvis en sensor blir koblet ut, eller nøyaktigheten forsvinner, kan fortsatt de andre sensorene gi tilstrekkelig informasjon om posisjonen til fremkomstmiddelet, spesielt hvis systemet beveger seg i et tidligere kartlagt område.

Måten dette kan anvendes på i sjøforsvaret er ved eksempelvis operasjoner som krever lav utstråling. Muligheten til å koble ut alle aktive sensorer og kun operere med passive sensorer for å posisjonere fartøyet. En annen fordel med et flersensor system er redundans

som gir robusthet mot jamming. Hvis GPS blir utsatt for jamming eller gir en usikkerhet større enn de andre sensorene, er det fremdeles mulig å benytte radar, optiske eller andre typer sensorer til å overta posisjoneringen.

9 Referanseliste

Anaconda inc., 2022. *Anaconda Distribution.* [Internett]
Available at: https://www.anaconda.com/products/distribution

Carboni, R. & Meyer-Vikaskag, T., 2022. *Automatisk radarposisjonering. Et alternativ til GNSS.* [Internett]
Available at: https://fhs.brage.unit.no/fhs-xmlui/handle/11250/3010713

Fisher, B., 1997. *3x4 Projection Matrix.* [Internett]
Available at: https://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/EPSRC_SSAZ/node3.html

Gul, S. et al., 2021. *Stereovision.* [Internett]
Available at: https://www.sciencedirect.com/topics/engineering/stereovision

Johansen, P. A. & Bentzrød, S. B., 2019. *Så ofte slår Russland ut GPS-nettet i Norge. Nå er russerne blitt en trussel mot sivile fly.*. [Internett]
Available at: https://www.aftenposten.no/norge/i/P3bAG6/saa-ofte-slaar-russland-ut-gps-nettet-i-norge-naa-er-russerne-blitt-en-trussel-mot-sivile-fly

Kjerstad, N., 2021. *Krysspeiling.* [Internett]
Available at: https://snl.no/krysspeiling
[Funnet 4 desember 2022].

Krishnan, K. S. & Sahin, F., 2019. *A Feature-Based Deep Learning Approach to Monocular Visual Odometry.* [Internett]
Available at: A Feature-Based Deep Learning Approach to Monocular Visual Odometry
[Funnet 22 november 2022].

Kristiansen, J. R., 2021. *Jamming.* [Internett]
Available at: https://snl.no/jamming

Lin, C.-e., 2018. *Introduction to Motion Estimation with Optical Flow.* [Internett]
Available at: https://nanonets.com/blog/optical-flow/

Mann, A. & Mysore, S., 2020. *Visual Odometry with Real-World Data.* [Internett]
Available at: https://cs-people.bu.edu/ainezm/cs585-FinalProj-sidmys-ainezm.html

Mathworks inc., 2022. *Monocular Visual Simultaneous Localization and Mapping.* [Internett]

Available at: https://se.mathworks.com/help/vision/ug/monocular-visual-simultaneous-localization-and-mapping.html

MathWorks inc., 2022. *What Is MATLAB?.* [Internett]
Available at: https://se.mathworks.com/discovery/what-is-matlab.html

MathWorks, 2022. *Feature extraction for machine learning and deep learning.* [Internett]
Available at: https://se.mathworks.com/discovery/feature-extraction.html

MathWorks, 2022. *Visual Simultaneous Localization and Mapping (vSLAM).* [Internett]
Available at: https://se.mathworks.com/help/vision/visual-simultaneous-localization-and-mapping-slam.html

MathWorks, 2022. *What Is Camera Calibration?.* [Internett]
Available at: https://se.mathworks.com/help/vision/ug/camera-calibration.html

Mishra, V., 2020. *Bag Of Visual Words.* [Internett]
Available at: https://medium.com/analytics-vidhya/bag-of-visual-words-bag-of-features-9a2f7aec7866

Mæhlum, L., 2020. *Triangulering.* [Internett]
Available at: https://snl.no/triangulering

Nayar, S. K., 2021. *Linear Camera Model | Camera Calibration.* [Internett]
Available at: https://www.youtube.com/watch?v=qByYk6JggQU&t=2s

Nielsen, N., 2021. *stereoVisionCalibration - Github repository.* [Internett]
Available at: https://github.com/niconielsen32/ComputerVision/tree/master/stereoVisionCalibration
[Funnet 21 oktober 2022].

Nielsen, N., 2022. *Visual Odometry with a Stereo Camera - Project in OpenCV with Code and KITTI Dataset.* [Internett]
Available at: https://www.youtube.com/watch?v=WV3ZiPqd2G4&t=33s

Nielsen, N. H., 2022. *Visual Odometry - Github repository.* [Internett]
Available at: https://github.com/niconielsen32/ComputerVision/tree/master/VisualOdometry
[Funnet 1 november 2022].

NumPy, 2022. [Internett]
Available at: https://numpy.org/

OpenCV team, 2022. [Internett]
Available at: https://opencv.org/

OpenCV, 2016. *FAST Algorithm for Corner Detection.* [Internett]
Available at: https://docs.opencv.org/3.4/df/d0c/tutorial_py_fast.html
[Funnet 28 november 2022].

OpenCV, 2016. *Object Tracking.* [Internett]
Available at: https://docs.opencv.org/3.4/df/d0c/tutorial_py_fast.html
[Funnet 2022].

Rublee, E., Rabaud, V., Konolige, K. & Bradski, G., 2011. *ORB: An efficient alternative to SIFT or SURF.* [Internett]
Available at: https://ieeexplore.ieee.org/abstract/document/6126544

The Matplotlib development team, 2022. *Matplotlib: Visualization with Python.* [Internett]
Available at: https://matplotlib.org/

Tyagi, D., 2019. *Introduction To Feature Detection And Matching.* [Internett]
Available at: https://medium.com/data-breach/introduction-to-feature-detection-and-matching-65e27179885d

Tyagi, D., 2019. *Introduction to ORB (Oriented FAST and Rotated BRIEF).* [Internett]
Available at: https://medium.com/data-breach/introduction-to-orb-oriented-fast-and-rotated-brief-4220e8ec40cf

Williams, B. et al., 2008. *An image-to-map loop closing method for monocular SLAM.* [Internett]
Available at: https://www.semanticscholar.org/paper/An-image-to-map-loop-closing-method-for-monocular-Williams-Cummins/2668c0211baea09c86582c60fc4d59ff7c88e2b3
[Funnet 12 desember 2022].

Vedlegg

Vedlegg A - Arbeidslogg

Vedlegg B - Brukermanual

Vedlegg C - Kildekode