Høgskulen på Vestlandet

Bacheloroppgave Nautikk (NAB3030)

NAB3030-PRO-1-2021-VÅR-FLOWassign

Predefinert informasjon

Startdato:	10-03-2021 13:56	**Termin:**	2021 VÅR
Sluttdato:	05-05-2021 14:00	**Vurderingsform:**	Norsk 6-trinns skala (A-F + Bestått)
Eksamensform:	Bacheloroppgave		
SIS-kode:	203 NAB3030 1 PRO-1 2021 VÅR		
Intern sensor:	(Anonymisert)		

Deltaker

Kandidatnr.:	217

Informasjon fra deltaker

Tittel *:	Algoritmer og sensorer som en drone trenger for å delta i AutoDrone 2020
Antall ord *:	15177
Engelsk tittel *:	Algorithms and sensors that a drone needs to participate in AutoDrone 2020

Sett hake dersom besvarelsen kan brukes som eksempel i undervisning?:	Ja	**Egenerklæring *:**	Ja
		Inneholder besvarelsen konfidensielt materiale?:	Nei
Jeg bekrefter at jeg har registrert oppgavetittelen på norsk og engelsk i StudentWeb og vet at denne vil stå på vitnemålet mitt *:	Ja		

Gruppe

Gruppenavn:	(Anonymisert)
Gruppenummer:	6
Andre medlemmer i gruppen:	220

Jeg godkjenner avtalen om publisering av bacheloroppgaven min *

Ja

Er bacheloroppgaven skrevet som del av et større forskningsprosjekt ved HVL? *

Nei

Er bacheloroppgaven skrevet ved bedrift/virksomhet i næringsliv eller offentlig sektor? *

Nei

Høgskulen på Vestlandet

BACHELOROPPGAVE

Algoritmer og sensorer som en drone trenger for å delta i AutoDrone 2020

Algorithms and sensors that a drone needs to participate in AutoDrone 2020

Jarle Stavland **217**
Sander Davidsen Raa **220**

Bachelor Nautikk
Høgskulen Vestlandet, campus Haugesund
Torkel Bjarte Larsson
05.05.2021

Jeg bekrefter at arbeidet er selvstendig utarbeidet, og at referanser/kildehenvisninger til alle kilder som er brukt i arbeidet er oppgitt, *jf. Forskrift om studium og eksamen ved Høgskulen på Vestlandet, § 12-1.*

Forord

Motivasjon

Da vi skulle velge hva vi skulle skrive om i vår bacheloroppgave så vi at det var kommet en e-post hvor han som skulle bli vår veileder, kom med et veldig spennende prosjekt som vi fattet interesse for. Vi rådførte oss med han og etter en oppløftende samtale fant vi ut av at dette var noe vi kunne tenke oss å skrive om.
Dette var jo et tema som er veldig relevant generelt i dagens samfunn, da selvkjørende biler er et hett tema for tiden og vi var nysgjerrig på hva som foregikk innen temaet til sjøs. Vi mente at vi kunne finne mye informasjon om dette og syntes at det kunne bli et veldig bra prosjekt om vi la ned nok innsats.

Som fremtidige styrmenn er vi veldig interessert i hvordan vi kan være med på å utvikle og forutse hvordan den maritime sektoren kommer til å se ut i fremtiden.
Med dette som grunnlag syntes vi at oppgaven vi har valgt passer ypperlig for å bli kjent med hva som foregår på et felt som vil ha stor innvirkning på vårt arbeidsområde.

Vi vil takke vår veileder Torkel Bjarte Larsson, som har vært til uvurderlig hjelp og som alltid har vært tilgjengelig hvis det var noe vi hadde spørsmål om.
Vi vil også takke Leif Ole Dreyer og Jone Abotnes for gode innspill til hvordan vi skulle løse denne oppgaven.

Sammendrag

AutoDrone konkurransen er en event som holdes av Universitetet i sør-øst Norge i Horten, og er en konkurranse åpen for droner/ ASV som er fullt ut autonome, og tilfredsstiller kravene satt av arrangør.
Som nautikkstudenter var vår innsikt i temaet autonome skip veldig begrenset, så for å sette oss inn i temaet måtte vi lese oss opp på grunnleggende begreper og teknologier. Vi bestemte oss for å gjøre et narrativt litteratursøk, og ved hjelp av *snowballing* metoden søke oss frem til litteratur på området.
Vår oppgave er ment som et design forslag, og vi har fokusert på hvilke utviklingsverktøy, algoritmer, sensorer og deres fusjoner som kan være aktuelle for å løse de fire oppdragene i konkurransen.

Abstrakt

The AutoDrone competition which is an event held by the University of Southeast Norway in Horten, and is a competition open to drones / ASV that is fully autonomous, and satisfies the requirements set by the organizer.
As nautical students, our insight into the topic of autonomous ships was very limited, so to get acquainted with the topic, we had to read up on basic concepts and technologies. We decided to do a narrative literature search and using the snowballing method search for literature in the field.
Our thesis is intended as a design proposal, and we have. focused on which development tools, algorithms, sensors and their mergers may be relevant to solve the assignments in competitions.

Innholdsfortegnelse

1. Innledning

1.1 Introduksjon

Autonome fartøy både til lands, vanns og i luften er noe som er veldig aktuelt for tiden. Teslas selvkjørende biler, Amazons flygende droner for pakkelevering er eksempler som har vært mye omtalt i media de siste årene. Autonome skip er nok ikke like mye omtalt, men siden 80-90% av verdens varer transporteres på skip i dag(Fiskeridepartementet, 2018) vil besparelsene på miljø og økonomi være særs store når man slipper å ha areal for mannskap ombord og kan prioritere lasterom og kunne frakte mer cargo i forhold til den totale størrelsen på skipet.

Automatisering av system på skip allerede i økende bruk, og Norge er ledende på feltet med flere opprettede testområder langs kysten.
Automatisering og fjernstyring av autonome skip kan gi økt sikkerhet, bedre miljø og høyere effektivitet i skipsfarten. (Fiskeridepartementet, 2018)
"AutoDrone 2020" er en konkurranse der skip i mindre og enklere skala (Droner) blir testet og satt på store prøver. "AutoDrone 2020" og liknende konkurranser fungerer som utprøvings tester for systemer som senere kan bli iverksatt større skip.
Ved å etterligne situasjoner som man kan komme ut for i virkeligheten vil man få mye kunnskap man igjen kan bruke i utviklingen av autonome skip, store skip.

1.2 Faguttrykk

I temaet vi har valgt å ta for oss finnes det svært mange engelske faguttrykk. Vi har valgt å ta for oss disse og skrive de på engelsk i denne oppgaven da det til tider ikke blir så lett å formulere disse på Norsk. De engelske faguttrykkene som er brukt i denne teksten er blitt lagt inn med kursiv skrift.

1.3 Problemstilling

Hvilke teknologier trenger man for å kunne utføre oppgaver som er beskrevet i reglene for AutoDrone 2020. (*AutoDrone 2020 Rules and Task Description.pdf*, u.å.)

Oppgavene dronen skal utføre:

1) Manøvrere i en kompleks led mellom røde og grønne bøyer hvor det vil være hindringer i leden som man må unngå.

2) Unngå kollisjon med kryssende trafikk

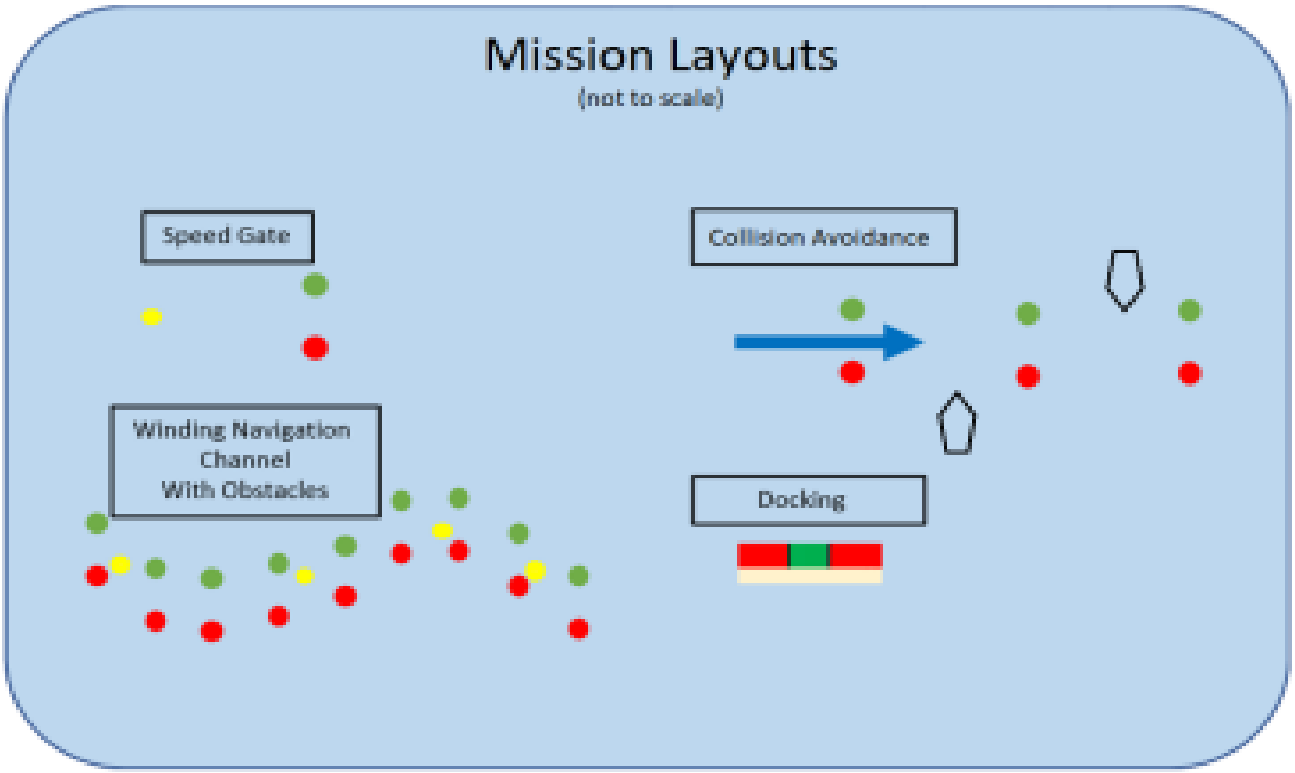
3) Visuell *docking* hvor den skal kunne lokalisere hvor den skal *docke* i et definert område.

4) *Speedgate* test hvor dronen skal manøvrere gjennom en port, finne og runde en markør og gå ut samme port.

For at oppgaven ikke skal bli for stor og overfladisk vil i denne oppgaven fokusere mest på hvilke algoritmer og sensorer som blir brukt for å løse disse oppdragene, og ikke gå så mye inn på hardware, batterier, trøstere og operativsystemer.

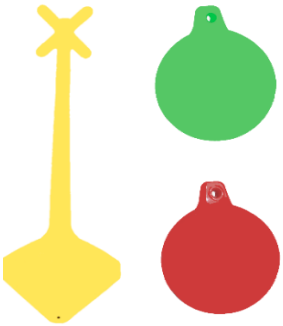
Mission tasks

Man skal demonstrere avansert autonom drone oppførsel gjennom å utføre oppdragene. Oppdragene vil bli utført etter konkurransens tidsplan. Et oppdrag om gangen med et lag om gangen i en gitt rekkefølge. En GPS-posisjon, som angir senter til oppdragets «inngang» vil bli gitt på konkurransen i desimal grad format.



Figur 1: Viser et oversiktsbilde over de fire oppdragene som dronen skal utføre i denne konkurransen. oppdragene er *speed gate, winding navigation channel with obstacles, collision avoidance og docking.*

Bøyer brukt i konkurransen



Red buoy: Art.nr. 25-017 from Biltema
Green buoy: Art.nr. 25-017 from Biltema spraypainted green (Art.nr. 36-577 from Biltema)
Yellow buoy: from Sommerbutikken

Figur 2: Oversikt Viser den røde, grønne og gule bøyen som dronen skal bruke som utgangspunkt for å navigere i konkurransen.

2. Metode

I dette kapittelet vil vi ta for oss hva metode er, hvilken metode vi har valgt å bruke i oppgaven og begrunne hvorfor vi valgte denne metoden. Vi vil også ta for oss hvilke begrensninger det er i å bruke metoden.

2.1 Hva er metode

I all hovedsak skilles det mellom kvalitativ og kvantitativ metode hvor den kvalitative metoden baserer seg mer på forståelse, oppfatning og mening, mens den kvantitative er best til å svare ved bruk av tall og statistikk. hvilken metode som bør brukes er avhengig av hvordan problemstillingen skal besvares.

2.2 Valg av metode

En del av oppgaven er å skaffe oss et ordforråd og kunnskap til å kunne utføre og finne frem til de ordene vi skal bruke for å søke etter de rette tingene (som algoritmer og forklaringer av konsepter). For å besvare problemstillingene i denne bacheloroppgaven valgte vi å bruke litteraturstudie. Det finnes flere typer litteratursøk så vi vil her gi en liten forklaring på hva de forskjellige litteratursøkene går ut på.

***Systematic Review*:** Systematiske litteratursøk gir en strukturert analyse av kjente evidens som kan fastslå helse og medisinsk praksis, standarder og offentlig politikk. Systematiske litteratursøk definerer et emne, identifiserer problemstillinger og besvarer problemstillingen ved å oppsummere og evaluerer funnene på emnet som er rapportert i litteraturen. Denne søkemetoden bruker strenge kriterier laget for å begrense bias og fremheve vitenskapelig validitet med mål om å produsere en upartisk analyse. Systematiske litteraturstudier er den foretrukne metoden for en streng litteraturstudie. En systematisk litteraturstudie krever et klart mål for studiet, definerte kriterier for hvilken litteratur som skal inkluderes og hvilken litteratur som skal ekskluderes, en søkestreng og søke-prosedyre for å kunne repetere søkingen, et strukturert litteratursøk for å finne alle studier og rapporter, en vitenskapelig evaluering av den inkluderte litteraturen, og en systematisk syntese av validiteten, svakhetene og funnene av litteraturen som er studert. En systematisk litteraturstudie bør alltid begynne med en erklært forskningsprotokoll. Protokollen beskriver begrunnelsen, hypotesen og

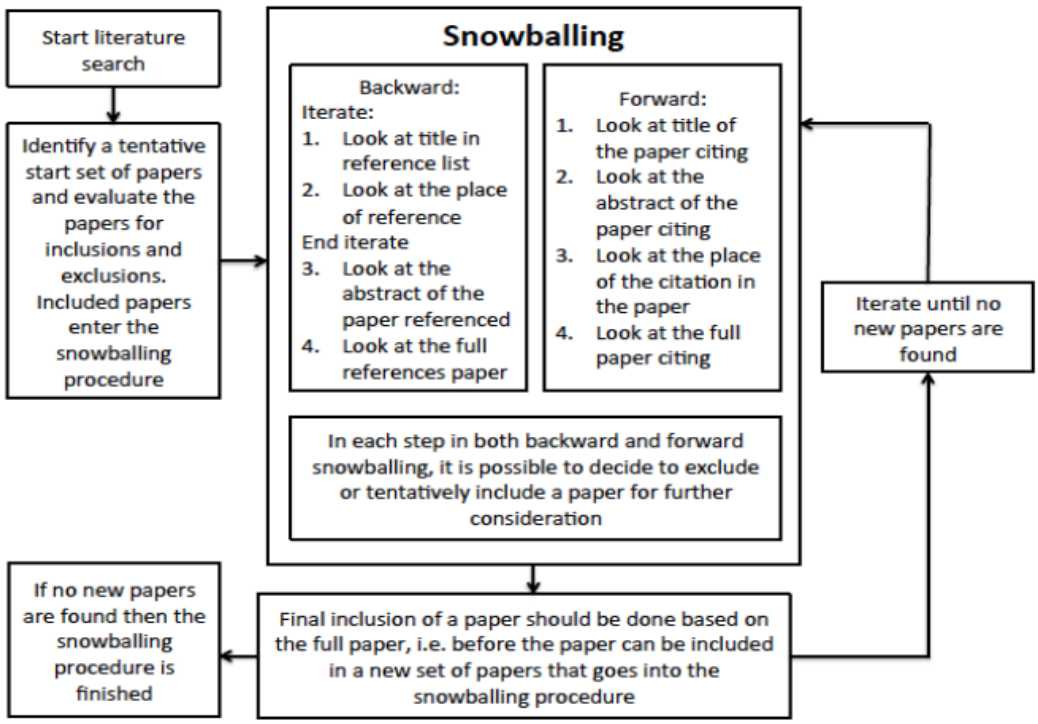
planlagte metoder for studiet. Den bør utarbeides før studiet starter og brukes som veiledning for gjennomføring av studiet. Protokollen bør bli skrevet eksplisitt i det endelige manuskriptet av reviewet. Før man setter i gang med en slik studie er det viktig at man er kjent med den tilgjengelige litteraturen for emnet og sikre at det er en stor nok mengde litteratur for studiets omfang og kvalitet. En systematisk litteraturstudie skal lages og beskrives slik at fremtidige forskere skal kunne reprodusere studiet. For systematiske litteratursøk finnes de retningslinjer som for eksempel i (Moher et al., 2009)

Snowballing literature review: Man starter litteraturreviewet med et nøkkeldokument. Man søker så i referansene i nøkkeldokumentet og så søker man i referanser i de dokumentene man fant etter søket i nøkkeldokumentet. Litteraturen som inngår i litteratur reviewet bygger på seg, derav navnet *snowballing*. Det finnes foreslåtte retningslinjer for hvordan *Snowballing Reviews* skal utføres. Se for eksempel (Wohlin, 2014)

Snowballing refererer til det å bruke referanselisten i en artikkel eller sitering av en artikkel til å identifisere nye artikler. Men *snowballing* kan også tjene på å ikke bare se på referanseliste og sitering, men og komplettere det med en systematisk måte å se på hvor artiklene egentlig er referert og hvor de er sitert. Å bruke referanseliste og sitering hver for seg blir referert som henholdsvis *backward* og f*orward snowballing*. Når man skal bruke snowballing metoden er den første utfordringen å identifisere et sett med artikler å starte med. En god start kan være å bruke for eksempel *Google scholar*. Det er et godt alternativ for å unngå bias i favør av en eller annen forlegger. En mulighet er å identifisere en artikkel som har hatt stor innflytelse på fagområdet, og som er høyt sitert.

Backward snowballing betyr å bruke referanselisten til å identifisere nye artikler som skal inkluderes. Det første steget er å gå igjennom referanselisten og ekskludere artikler som ikke innfrir de grunnleggende kriteriene som for eksempel språk, publikasjons år, eller type publikasjon. Neste steg er å fjerne artikler fra listen som man allerede har undersøkt basert på tidligere *backward* eller *forward snowballing* søk i dette eller tidligere iterasjoner. Når disse er fjernet er de resterende artiklene reelle kandidater for inkludering.(Wohlin, 2014)

Forward snowballing refererer til å identifisere nye artikler basert på artikler som siterer artikkelen som blir undersøkt. Siteringen av artikkelen som blir undersøkt kan bli studert ved å bruke *Google Scholar*. Artiklene som den utvalgte artikkelen siterer, blir fjernet i *Google Scholar* og bare artikler som siterer den utvalgte artikkelen blir brukt. Alle kandidater som siterer artikkelen, blir undersøkt. Den første undersøkelsen blir gjort basert på informasjon gitt av *Google Scholar*. Hvis denne informasjonen ikke er tilstrekkelig for en avgjørelse blir den siterende artikkelen studert i større detalj. Først blir abstraktet studert og hvis det ikke er tilstrekkelig, blir stedet i den siterende artikkelen hvor artikkelen som allerede er inkludert blir undersøkt. Hvis ikke dette er tilstrekkelig, blir hele teksten studert for å bestemme seg om den nye artikkelen skal inkluderes eller ikke. Tilnærmingen til å gå gjennom artiklene er lik den artikkel-gjennomgangen for *backward snowballing*. Se figur under.(Wohlin, 2014)



Figur 3: Viser fremgangsmåten i en *Snowballing* prosedyre. figur 1 (Wohlin, 2014)

***Narrative Literature Review*:** *Narrative literature review* er ustrukturerte søk i litteratur brukt for å etablere en teori, eller konteksten for, eller for å rettferdiggjøre et forskningsfokus. Narrative litteratursøk blir vanligvis funnet i introduksjonen eller bakgrunns delen i et formelt manuskript og brukt for å introdusere og gjennomføre forskning beskrevet i det manuskriptet. *Narrative reviews* kan også bli publisert som frittstående artikler fokusert på teori og tanke rammer angående litteraturen av et bestemt emne *(Theoretical Literature Review). Narrative review* kan også skissere metoder eller forskningsdesign som blir brukt for et bestemt forskningsemne *(Methodical Literature Review),* eller gi en historisk evaluering av utviklingen av teori eller konsepter for et emne (*Historical Literature Review*). *Narrative reviews* kan utføres ved å bruke søkeord i standard litteratur databaser, men beskriver ikke de spesifikke metodene for utvelgelse og gjennomgang av litteraturen man har hentet. *Narrative reviews* inkluderer ofte forfatterens antagelser og partiskhet. ser og kan generelt ikke gjenskapes som med *systematic literature review*. Tre av de vanligste underkategoriene er:

***Critical Review*:** *Critical reviews* inkluderer ikke det strenge studiedesignet til et *systematic review* og er en form av *narrative review*. målet med et *critical review* er å utvikle perspektiver på et forskningsemne ved å bruke tilgjengelig litteratur. Ofte er denne typen review begrenset til ny litteratur, men kan også bli brukt til å vise forandringer i perspektiv over tidsperioder. (Stratton, 2019)

***Conceptual Review*:** *Conceptual reviews* inkluderer ikke en litteratursøk-protokoll eller systematisk datautvinnings teknikker. Denne typen review er vanlig i ledere og konsept artikler. *Conceptual reviews* evaluerer den generelle konsensusen av litteraturen på et gitt forskningsemne og undersøke hvordan denne forståelsen ble nådd. De er laget for å vise den nåværende forståelsen av et emne og for å foreslå om en bedre forståelse eller konsensus trengs. *conceptual reviews* har også et potensial for å vise huller i kunnskapsbasen for et spesifisert forskningsområde. (Stratton, 2019)

***State-Of-the Art Review*:** Denne underkategorien av *narrative review* blir brukt til å fokusere på ny forskning og beskriver hva som for tiden kjent og man er enige om for *review* emnet. Vanlig ved denne typen *review* er en diskusjon om områdene av enighet og uenighet i *review* emnet. *Narrative reviews* som også blir kalt *traditional reviews* kan bli gitt andre navn en de identifisert i denne diskusjonen. *Standard review, comprehensive review, snow-ball review, bibliographic review,* og andre lignende terminologier blir noen ganger brukt for å identifisere det som egentlig er et grunnleggende *narrative review*. (Stratton, 2019)

Type	Advantage(s)	Disadvantage	Application(s)	Guidelines
Systematic Review	1. Minimized bias 2. A-priori protocol 3. Defined search and evaluation methods 4. Reproducible 5. High validity of review conclusions	1. Must adhere to established guidelines 2. Valid literature base required 3. Robust (enough) literature to review 4. Variation in study methods within reviewed literature may affect results	1. Identify relevant evidence 2. Assess quality of evidence 3. Non-biased synthesis of literature 4. Interpret evidence in an impartial manner 5. Applicable for establishing standards and health policy	PRISMA Guidelines[2]
Meta-Analyses - Quantitative	1. Same as systematic review 2. Determine a single estimate of the effect of treatment or management of an illness or event	1. Data in literature must be homogeneous and available for pooled analysis 2. Reliability of literature designs may affect results	1. Same as systematic review 2. Determine best practice for defined topic or event. 3. Narrow variations in known data sets.	PRISMA Guidelines[2]
Meta-Analyses - Qualitative	1. Same as systematic review 2. Determine major themes or experiences for an event or issue	1. Variable sampling errors in original literature leads to bias 2. Variation in qualitative tools used for original research	1. Same as systematic review 2. Define primary themes and priorities 3. Refine future research objectives	PRISMA Guidelines[2]
Cochrane Review	1. Form of systematic review method 2. Well defined methodology 3. Indexed in the Cochrane Library (open source)	1. Same as for Systematic Reviews	1. Same as systematic review 2. Determine support for specific treatment 3. Determine if evidence exists for defined concept	Cochrane Manual[5]
Scoping Review	1. Use of fluid literature search strategy 2. Broader review topics 3. May include literature of varied methodologies	1. Risk of bias due to lack of defined evaluation methods 2. Non-specific objectives 3. Heterogeneity in literature included	1. Map available literature in a review field or area 2. Literature gap analysis 3. Clarification of concept or theory	PRISMA SrR[7]
Narrative Review	1. Researcher determines literature to include 2. Less time intensive 3. May include literature of varied methodologies 4. Interpretive objectives (not structured analysis)	1. Risk of multiple forms of bias and error 2. Unstructured, not reproducible 3. May not include all appropriate literature 4. Lacks systematic synthesis of literature	1. Identify theory and frames of thought on a topic 2. Summarize a particular study topic 3. Justify a research topic	
Critical Review	Same as Narrative Review	Same as Narrative Review	1. Develop perspectives on a topic	
Conceptual Review	Same as Narrative Review	Same as Narrative Review	1. Evaluate general consensus on a topic 2. Show gaps of knowledge in literature	
State-of-the Art Review	Same as Narrative Review	Same as Narrative Review	1. Describe current beliefs on a topic	

Figur 4: Viser forskjellige typer litteratursøks fordeler, ulemper og under hvilke forutsetninger de egner seg.(Stratton, 2019)

Vi har valgt å bruke narrative *literature search* for å løse denne bacheloroppgaven siden problemstillingen er veldig vid og vi ikke vil prøve å gi en helhetlig oversikt over hva som er gjort på fagfeltet, men komme med et designforslag for hvordan oppgavene i AutoDrone kan løses.

2.3 Begrensninger i metode

Når det kommer til begrensninger så kan vi se ut ifra figur 4 i (Stratton, 2019) at risikoen for bias og feil, at den ikke kan reproduseres, at litteraturen ikke nødvendigvis er den riktige og at det mangler systematisk struktur er høyere ved denne type søk, gjør at metoden har en del svakheter.

3. Teori

I denne delen vil vi beskrive utviklingsverktøyene, maskinvaren, algoritmene og begrepene innen navigasjon som trenges for at dronen skal kunne løse oppgavene i konkurransen.

3.1 UTVIKLINGS SOFTWARE

ROS: (Robot Operating System) er en samling av Software rammeverk og verktøy som gir funksjonalitet til et operativsystem.(*ROS.Org | Powering the World's Robots*, u.å.)

MATLAB er et høyt fungerende språk for teknisk databehandling. Det integrerer beregning, visualisering og programmering i et lettfattelig miljø hvor problemer og løsninger er uttrykt i kjente matematiske notasjoner. Det er enkelt å bruke og dedikert matematisk og teknisk beregning mens **Python** er et programmeringsspråk brukt for mer generelle formål.(*What is Matlab*, u.å.)

GAZEBO er et program som kjører robot simuleringer. Det gjør det mulig å raskt teste algoritmer, designe roboter, trene kunstig intelligens systemer ved å bruke realistiske scenarioer.(*Gazebo*, u. å.)

LabVIEW (Laboratory virtual instrument engineering workbench) kan brukes for innsamling av data fra måleinstrument med en PC, og for analyse og presentasjon av disse dataene.(*What Is LabVIEW?*, u.å.)

OpenCV (Open source vision library) bibliotek er en samling prosesser og algoritmer som gir en ny funksjonalitet til det totale OpenCV-biblioteket. vanligvis er en OpenCV-biblioteks pakke sentrert om et tema og hver av funksjonene støtter hele temaet. De fleste bibliotekene er skrevet i programmeringsspråket C. OpenCV brukes ofte til datavisjon og grafisk gjengivelse.(«About», u.å.)

Python

Python er et objekt orientert programmeringsspråk, og tillater for eksempel polymorfisme. I motsetning til mange andre objekt orienterte språk, som for eksempel java og *Smalltalk,* er det likevel fullt mulig å skrive et prosedyre-drevet program. Fra 2016 har python oftest vært å finne blant de fem øverste i kåringer over verdens mest brukte programmeringsspråk.(«Python», 2021)

C++

C++ er et programmeringsspråk for generelle formål. Det er en forlengelse av C programmeringsspråk og har et objekt orientert, generisk og funksjonell funksjon i tillegg til fasiliteter for manipulering av minne på lavt nivå. Det er nesten alltid implementert som et kompilert språk og mange leverandører tilbyr C++ kompilatorer, inkludert Free software foundation, LLMV, Microsoft, Intel, Oracle og IBM så det er tilgjengelig på mange plattformer.(«C++», 2021)

3.2 HARDWARE

Radar: (Radio Detection and Ranging) Er et gjenkjenningssystem som bruker radiobølger til å detektere, og til å måle avstand, vinkel og hastighet til andre objekter. Dette systemet er opprinnelig oppfunnet for militært bruk og består av en sender som sender elektromagnetiske bølger, og en mottakerantenne som tar imot de reflekterte elektromagnetiske bølgene. I tillegg består systemet av et filter som tar ned og filtrerer signalet som bestemmer egenskapene til objektet som er detektert. Radar er et av de viktigste utstyrene for sikker ferdsel til sjøs. («Radar», 2021)

AIS: (Automatic Identification System) er et automatisk identifikasjonssystem og antikollisjonshjelpemiddel for å øke sikkerheten for skip og miljø, samt forbedre trafikkovervåkning og sjøtrafikktjenester.

En AIS-transponder skal automatisk forsyne andre skip og offentlige myndigheter med informasjon om skipets identitet, fart og kurs..

Dette skal også ta imot andre skips informasjon. Ved hjelp av dette kan alle skip med AIS ombord danne seg et bilde av trafikksituasjonen i sitt nærområde.

I tillegg til AIS transpondere ombord på skip er det også utviklet AIS-transpondere for bruk på land, på fyr, på sjømerker, andre små lyst-fartøy samt i luftfartøy og fly.

IMO (International Maritime Organization) har satt som krav at alle fartøy over 300 bruttotonn som seiler i internasjonalt farvann og eller fører farlig eller forurensende last, skal ha utstyr for sending og mottak av AIS-signaler. («Automatic Identification System», 2021)

LIDAR:(Light/Radar) En LIDAR er en teknologi som gir oss presis og nøyaktig posisjonsinformasjon i sanntid av bevegelige objekter og infrastruktur. Dette er noe som brukes til å kartlegge området rundt et fartøy. LIDAR`en er nesten uforstyrret av refleksjoner fra sjøen -noe som gjør den svært egnet for til å oppfatte gjenstander i vannet. Dette fører til at den kan gjenkjenne to forskjellige skip som er svært nær hverandre, og ikke minst oppfatte små fartøy.
Ved hjelp av en laser lyser den rundt i fartøyets omgivelser og analyserer de reflekterte pulsene ved hjelp av en sensor. LIDAR`en er en ideell sensorfusjon i samarbeid med RADAR, kameraer, elektroniske kart og AIS.(Hebert et al., 1997)

IMU (Inertial Measurement Unit) er en sensor som brukes til å måle kreftene som virker på et fartøy, samt fartøyets og retning. Dette oppnås ved å bruke tre sensorer: gyroskop, magnetometer og akselerometer.(«Inertial Measurement Unit», 2021)

GPS (Global Positioning System) er et nettverk bestående av omtrent 30 satellitter som er plassert i bane rundt jorden av det amerikanske forsvaret. Systemet gjør det mulig for en mottaker å fastsette sin egen posisjon med svært stor nøyaktighet overalt i verden, under nær sagt alle værforhold.(«Global Positioning System», 2020)

3.3 ALGORITMER

CNN (Convolutional Neural Network): ConvNet/CNN er en deep learning algoritme som kan ta imot et input bilde, tildele viktighet til forskjellige aspekter/objekter i bildet og skille et bilde fra et annet. Det brukes til å analysere et bilde. Blir brukt i et utvalg av 2D og3D data. (*Convolutional Neural Networks*, u.å.)

Multiple Color Detection in real time using Python-Opencv

For at en robot skal kunne visualisere miljøet rundt seg, sammen med objekt deteksjon, er deteksjon av farge i sann tid veldig viktig. Det er viktig for både selvkjørende biler og autonome fartøy på sjøen når det kommer til trafikklys, og farge på staker og lanterner.

Arbeidsflyts beskrivelse:

Steg 1. input: ta opp video med kamera.

Steg 2. lese videostrømmen i bilderammer.

Steg 3. konverter bilderamme i BGR (RGB *color space* representert som tre matriser av rød, grønn og blå med heltallsverdier fra 0 til 255) til HSV *(hue- saturation- value) color space.*

Hue: beskriver en farge i form av..

Saturation: representerer mengden av gråfarge i fargen og

Value: beskriver *brightness* eller intensitet av fargen. Dette kan bli representert som tre matriser i området henholdsvis 0-179, 0-255 og 0-255.

Steg 4. definer området for hver farge og lag den korresponderende maske.

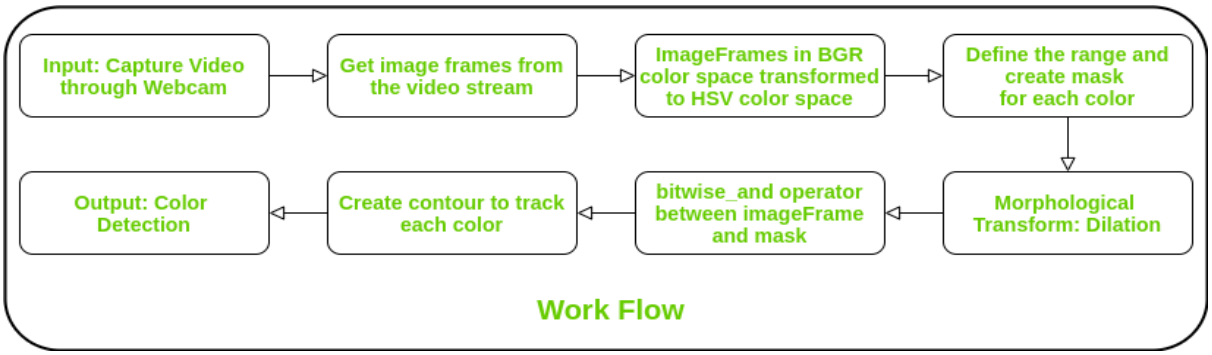
Step 5. *morphological transform: Dilation*, for å fjerne støy i bildene.

Step 6. *bitwise and* blir utført mellom bilderammen og maske for å spesifikt detektere den bestemte fargen og fjerne andre.

Step 7. lag kontur for de enkelte fargene for å vise det oppdagede fargede området adskilt.

Step 8. output: deteksjon av fargene i sann tid.

(«Multiple Color Detection in Real-Time Using Python-OpenCV», 2020)



Figur 5: Viser stegene i algoritmen for farge deteksjon.

(«Multiple Color Detection in Real-Time Using Python-OpenCV», 2020)

Maske

I bildebehandling er en kjerne, konvolusjons matrise eller maske en liten matrise. Den brukes til blurring, sliping, preging, kantdeteksjon og mer. Dette oppnås ved å gjøre en konvolusjon mellom en kjerne og et bilde.

(«Kernel (Image Processing)», 2021)

A* algorithm bruker beste-først søk og finner stien med minst kostnad gitt en start node til en mål-node. A* følger stien med minst forventet total kostnad. Den holder en prioritetsliste av alternative stier og om kostnaden på den nåværende stien overstiger den første i listen kan den skifte alternativ.

Det som gjør at A* er så effektiv er at den tar i bruk heuristikk for å finne ut hvilke av nodene i treet den bør søke gjennom først. Dette blir representert som en funksjon f(x) som er en sum av to funksjoner:

- en funksjon som er distansen fra start-noden til den nåværende noden x (vanligvis *g(x))*
- En heuristikk funksjon som er en estimering av distansen fra x til målet (vanligvis *h(x))*

For at A* skal være gunstig må *h(x)* aldri overestimere kostnaden for å nå målet. Det vil si at den er tillatelig (engelsk: admissible). A* er en search-based method algoritme. («A*», u.å.)

RRT *(Rapidly-exploring random tree)* premisset for en RRT er at punkter blir tilfeldig generert og tilkoblet nærmeste tilgjengelige node. Hver gang et toppunkt blir laget, må det sjekkes at toppunktet ligger utenfor en hindring. Og videre så må lenken mellom toppunkt og nærmeste node også unngå hindringer. Algoritmen avsluttes når en node blir generert i

målområdet eller når en grense er nådd. RRT er en *sampling- based method* algoritme. (Chin, 2019)

RRT* er en optimalisert versjon av RRT. Når antallet noder nærmer seg uendelig, vil RTT* angi den kortest mulige stien til målet.(Chin, 2019)

DBSCAN (Density based spatial clustering of applications with noise)

Formålet med DBSCAN algoritmen er å finne vilkårlig formede *clusters* (klynger) og *clusters* med støy (utliggere)

hovedideen er at et punkt tilhører en cluster hvis den er nær mange punkt i en klynge.

det er to nøkkelparametre ved DBSCAN:

eps: Distansen som spesifiserer nabo områdene. to punkt blir definert som naboer hvis distansen mellom dem er mindre en eller lik eps.

minPts: Et minimum antall datapunkt for å definere en cluster.(Yıldırım, 2020)

Clustering: Density based- den underliggende distribusjonen av data, og beregner hvordan områder med høy tetthet av data korresponderer med topper i distribusjonen.

Clustering: Distance based -bruker en avstandsmåling for å bestemme likheten mellom dataobjekter. Avstandsmålingene måler avstanden mellom faktiske tilfeller i klyngen og det prototypiske tilfelle for klyngen som er kjent som *centroid*.(«Cluster Analysis», 2021)

Cluster analysis

Cluster analysis eller *clustering* er oppgaven å med å gruppere et sett med objekter på en slik måte at objekter i samme gruppe er mer like, i en viss forstand, hverandre enn de i andre grupper. Det er en hovedoppgave med utforskende dataanalyse, og en vanlig teknikk for statistisk dataanalyse, brukt på mange felt, inkludert mønstergjenkjenning, bildeanalyse, informasjonsinnhenting, bioinformatikk, datakomprimering, datagrafikk og maskinlæring. *Cluster analysis* i seg selv er ikke en spesifikk algoritme, men den generelle oppgaven som skal løses. Det kan oppnås ved hjelp av forskjellige algoritmer som skiller seg betydelig ut i deres forståelse av hva som utgjør en klynge og hvordan de effektivt kan finne dem. Populære forestillinger om *clusters* inkluderer grupper med små avstander mellom klynge medlemmer, tette områder, intervaller eller bestemte statistiske fordelinger. Clustering kan derfor formuleres som et multi-objekt-optimaliseringsproblem. Den passende *cluster a*lgoritmen og

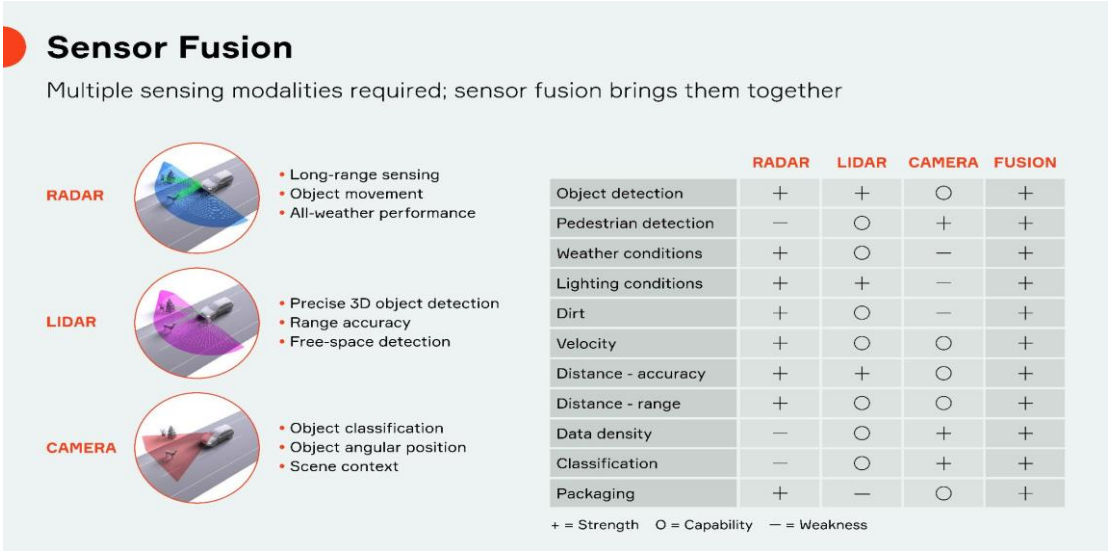
parameterinnstillingene, inkludert parametere som avstandsfunksjonen som skal brukes, en tetthets terskel eller antall forventede *clusters*, avhenger av det individuelle datasettet og tiltenkt bruk av resultatene. *Cluster analysis* som sådan er ikke en automatisk oppgave, men en iterativ prosess med kunnskaps oppdagelse eller interaktiv multi-objektiv optimalisering som involverer prøving og svikt. Det er ofte nødvendig å modifisere forbehandling av data og modellparametere til resultatet oppnår det ønskede egenskapene.
Centroid-modeller som for eksempel *k-means* algoritmen, representerer hver *cluster* med en enkelt gjennomsnitts vektor.
Density baserte modeller som DBSCAN og OPTICS definerer *clusters* som tette tilkoblede data regioner i datarommet. («Cluster Analysis», 2021)

***SLAM (Simultaneous localization and mapping)*:** I beregningsmessig geometri og robotics er *SLAM* det beregningsmessige problemet å lage eller oppdatere et kart med ukjent miljø samtidig som man holder kontroll på hvor fartøyet er på det. det finnes flere algoritmer til å løse det. Populære løsningsmetoder inkluderer partikkelfilter, utvidet Kalman Filter, kovarians kryss og *GraphSLAM*. *SLAM* algoritmer blir brukt i navigasjon, r*obotic mapping*. *SLAM* algoritmer er skreddersydd til de tilgjengelige kildene, og sikter ikke mot perfeksjon, men operasjonell bruk.(«Simultaneous Localization and Mapping», 2021)

Pose graph optimization
Man bruker *pose graph optimization* for å eliminere feil i Lidar og odometer. Ved å sette begrensninger i avstand mellom sann og falsk posisjon. Roboten kjører så rundt i omgivelsene og bygger et kart av området som kan være veldig lite nøyaktig. første og siste node som skal være den samme og observerer de samme omgivelsene. Ved å finne ut hvor de to sammenfaller og drar de sammen, vil man over tid optimalisere modellen til å bli bedre og bedre.(MATLAB, 2020c)

Sensor fusion er evnen til å bringe sammen inputs fra flere sensorer som radarer, Lidarer og kameraer for å lage en enkel modell eller bilde av miljøet rundt et fartøy. Den resulterende modellen er mye mer nøyaktig fordi den balanserer styrken til de forskjellige sensorene. Fartøyenes systemer kan så bruke informasjonen gitt ved *sensor fusion* til å støtte mer intelligente handlinger.

Hver sensor type eller modalitet har sine styrker og svakheter. Radar er veldig sterk på å nøyaktig bestemme distanse og hastighet, til og med i utfordrende værforhold, men kan ikke lese trafikkskilt eller se farger. Kameraer gjør det veldig bra når det kommer til å klassifisere objekter som for eksempel fotgjengere, syklister eller andre fartøy. Men de kan lett bli blindet av skitt, solen, regn, snø eller mørke. Lidarer kan nøyaktig detektere objekter, men har ikke rekkevidden eller samme overkommelige pris som kameraer eller radar.
Sensor fusion bringer data fra hver av disse sensor typene sammen ved å bruke *software* algoritmer til å gi de mest omfattende, og derfor nøyaktige, miljø modellen mulig. (Aptiv - Mobility Insider, 2020)



Figur 6: Viser evne, styrke og svakhet til radar, lidar og kamera i sensor fusion som enkelt sensorer, samt styrker for en fusjon mellom sensorene.

Stereo vision (Stereoscopic vision)

I r*obotic autonomous navigation* er en av de viktigste funksjonene 3D sansing og terreng rekonstruksjon for *path planning* og navigasjon. *Stereoscopic vision* er en mekanisme for å skaffe seg dybde eller rekkevidde data basert på bilder. Denne prosessen er lik menneskets stereoskopiske dybdesyn og vår intuitive oppfattelse av dybde. Desto lenger vekk objekter er i synsfeltet, desto mindre forandrer de posisjon når vi har et øye åpent og så bytter og har det andre åpent. I *stereo vision* er i prinsippet det samme. Objekter som ligger langt borte har liten forskjell, eller misforhold, mellom stereo bildene. Det klassiske problemet ved stereo analyse består i hovedsak av følgende steg: *image acquisition, camera modeling, feature extraction, image matching og depth determination.* Av desse er *matching* prosessen et nøkkelsteg. For å

gjøre det består systemet av to kamera skilt fra hverandre med en gitt grunnlinje, så de to forskjellige synene på en scene er oppnådd likt menneskesyn. Dette paret med bilder er inputen til matching prosessen. Det er prosessen med å identifisere trekk i begge bildene og regne ut forskjellen i posisjon på de trekkene eller pikslene fra det ene bildet relativt til det andre, vanligvis langs den horisontale aksen og skaffer seg et sett med likheter og ulikheter. Ulikhet er definert som subtraksjonen, fra venstre bilde til det høyre bildet, av 2D koordinatene av tilsvarende punkt i bildet.

Som et resultat av *matching* prosessen blir en relativ dybde informasjon oppnådd, som er omvendt proporsjonal med distansen til objektene. Siden dybde er omvendt proporsjonal med ulikhet er det åpenbart en non lineær relasjon mellom disse to begrepene. Når ulikheten er nær 0, vil små ulikheter føre til store dybdeforskjeller. Når ulikheten er stor, vil små ulikheter ikke forandre dybde noe særlig. Konsekvensen er at *stereo vision* systemer har høy dybde oppløsning bare for objekter relativt nærme kamera. Dybde kan etableres ved triangulering av de oppnådde ulikhetene, gitt at posisjonen til sentrene av projeksjonen, brennvidden og orienteringen av den optiske aksen er kjent. Brennvidden, vanligvis representert i millimeter, er den grunnleggende beskrivelsen av en fotografisk linse. Det er ikke målingen av lengden på linsen, men utregningen av den optiske distansen fra det punktet hvor lysstråler konvergerer til å skape et skarpt bilde av et objekt til den digitale sensoren eller 35mm filmen i brennplanet i kameraet. Brennvidden i en linse blir bestemt når linsen er fokusert i uendelig. Brennvidden bestemmer synsvinkelen og hvor mye av en scene blir fanget, og magnifisering, hvor store individuelle element blir. Dess lengre brennvidde, jo smalere synsvinkel og større magnifisering. Dess kortere brennvidde, dess videre synsvinkel og lavere magnifisering. Linjen fra kamera senter, vinkelrett til bildeplanet kalles hovedaksen eller optiske aksen til kamera. planet parallelt med bildeplanet som inneholder det optiske senteret kalles brennvidden til kamera.

Forholdet mellom 3D koordinatene av en scene punkt og koordinatene av dets projeksjon på Y bildeplanet er beskrevet av de sentral eller perspektiv projeksjon.

derfor kan punkt koordinatene i kameraets system referanse bli regnet som (X´,Y´,Z) for det første kamera og (X, Y, Z) for det andre. de kan beregnes hvor langt borte det matchende punktet er (dybde Z) ved derivering av neste uttrykk:

$$\gamma \cdot x' = f\frac{X'}{Z} \qquad \gamma \cdot x = f\frac{X}{Z}$$

$$Z = \frac{f(X'-X)}{\gamma(x'-x)}$$

Ved å kjenne kameraets iboende parametere: brennvidde (f), kameras grunnlinje (b),og piksel dimensjon □.
(Tezanos, 2015)

Occupancy Grids: Brukes til å vise en robots arbeidsområde som et rutenett. Informasjonen om dette området kan hentes fra sensorer i sann tid. Vanlige sensorer er Lasere, overflate sensorer, dybde sensorer og kameraer til å hente inn informasjonen om området. Vi kan forenkle dette med å si at *occupancy grids* er robotens kart hvor hindringene er tegnet inn. kartet bruker robotens vei i planleggingen.
(*Occupancy Grids - MATLAB & Simulink - MathWorks Nordic*, u.å.)

Occupancy Grid Mapping: Hensikten med *occupancy grid mapping* er å lage et kart slik at roboten har kunnskaper om hindringe**r** i området den skal bevege seg i.
Kartet skal også kunne representere tilstedeværelsen av et hinder i miljøet skipet beveger seg i ved hjelp av ulike variabler. Hindringene blir modellert som tilfeldige variabler i *Occupancy Grid Mapping.*
(«Occupancy Grid Mapping», 2021)

Hu moments: Eller HU moment invariants er et sett med 7 størrelser som er beregnet ved å bruke sentrale momenter. De første 6 momentene har vist seg å være invariant for *translation, scale, rotation* og refleksjon. Hu moments blir brukt til å beskrive, karakterisere og tallfeste formen eller konturen av et objekt i et bilde. Blir brukt i OpenCV med f.eks Python eller C++.(*Shape Matching Using Hu Moments (C++ / Python)* | *Learn OpenCV*, 2018)

Scale Invariant
I fysikk, matematikk og statistikk er skala-invarians et trekk ved objekter eller lover som ikke endres hvis skalaer av lengde, energi eller andre variabler multipliseres med en felles faktor, og dermed representerer en universalitet.(«Scale Invariance», 2021)

Rotation invariant

I matematikk sies det at en funksjon definert på et indreproduktrom har rotasjons invarians hvis verdien ikke endres når vilkårlige rotasjoner brukes på argumentet.(«Rotational Invariance», 2020)

SIFT *(The scale-invariant feature transform)* er en *feature detection* algoritme innen datasyn som brukes til å finne og beskrive features i bilder.(«Scale-Invariant Feature Transform», 2021)

Morphology er et bredt sett med bildebehandlings operasjoner som prosesserer bilder basert på former. De vanligste morfologiske operasjonene er dilation og erosion. Dilation legger til piksler i yttergrensene av et objekt i et bilde mens erosion fjerner piksler. Opening er når man først bruker erosion, for å krympe bildet så dilation for å utvide. Closing er å gjøre det motsatte.

Morfologisk opening er nyttig for å fjerne små objekter fra et bilde, men bevarer form og størrelse av større objekter i bildet. Morfologisk closing er nyttig for å fylle små hull i bildet mens form og størrelse bevarer. (*Types of Morphological Operations - MATLAB & Simulink - MathWorks Nordic*, u.å.)

***OSPA** metric* kan bli betraktet som en statistisk avstand mellom flere spor og sannheter. For å beregne OSPA beregninger, tildeler algoritmen eksisterende spor og sannheter til hverandre ved å bruke en Global Nearest Neighbor (GNN) algoritme. Når oppgaven er beregnet, deler OSPA hele distansen inn i to underkomponenter. lokalisering og kardinalitet- misforhold. Lokaliseringskomponenten fanger opp feil som oppstår i nøyaktigheten av tilstands estimeringen, mens kardinalitets- misforhold komponenten fanger opp effekten av overflødige spor, falske spor og savnede sannheter. Den tradisjonelle OSPA beregningen tar ikke med den tidsmessige historien til spor, oppgavene fra tidligere steg har ingen effekt på beregningen av nåværende steg.(*Introduction to Tracking Metrics - MATLAB & Simulink - MathWorks Nordic*, u.å.)

ROI (*Region of interest*) er eksempler i et datasett som er identifisert for et bestemt formål. Konseptet med en ROI brukes ofte i mange bruksområder. For eksempel, i medisinsk bildebehandling, kan grensene for en svulst defineres på et bilde eller i et volum for å måle størrelsen. I geografiske informasjonssystemer (GIS) kan en ROI tas bokstavelig som et mangekantet utvalg fra et 2D-kart. I *computer vision* og optisk tegngjenkjenning definerer ROI

grensene til et objekt som vurderes. I mange applikasjoner legges symbolske (tekstlige) etiketter til en ROI for å beskrive innholdet på en kompakt måte. Eksempler på regions of interest:

1D datasett: et tids- eller frekvensintervall på en bølgeform
2D datasett: grensene til et objekt på et bilde
3D-datasett: konturene eller overflatene som skisserer et objekt (noen ganger kjent som *Volume of interest* (VOI)) i et volum
4D datasett: omrisset av et objekt ved eller i løpet av et bestemt tidsintervall i et tidsvolum. («Region of Interest», 2020)

GOSPA metric måten å beregne *GOSPA* er lik *OSPA.* Ved å bruke en litt annerledes matematisk formulering, beregner *GOSPA* i tillegg underkomponenter som tapte mål komponenter og falske spor komponenter. Likt tradisjonell *OSPA* tar heller ikke *GOSPA* med den tidsmessige historien til spor.(*Introduction to Tracking Metrics - MATLAB & Simulink - MathWorks Nordic*, u.å.)

RANSAC *(Random sample consensus)* er en iterativ metode for å estimere parametrene til en matematisk modell fra et sett med observerte data som inneholder utliggere, når utliggere ikke skal gis noen innflytelse på verdiene til estimatene. Derfor kan det også tolkes som en metode for deteksjon avdekking. Det er en ikke-deterministisk algoritme i den forstand at den produserer et rimelig resultat bare med en viss sannsynlighet, med denne sannsynligheten økende ettersom flere iterasjoner er tillatt. Algoritmen ble først publisert av Fischler og Bolles hos SRI International i 1981. De brukte RANSAC for å løse *Location Determination Problem* (LDP), hvor målet er å bestemme punktene i rommet som projiserer på et bilde til et sett med landemerker med kjente steder.
En grunnleggende antagelse er at dataene består av "inliers", dvs. data hvis distribusjon kan forklares med noen sett med modellparametere, men kan være utsatt for støy, og "outliers" som er data som ikke passer til modellen. De avvikende kan for eksempel komme fra ekstreme verdier av støy eller feilaktige målinger eller uriktige hypoteser om tolkning av data. RANSAC antar også at det, gitt et (vanligvis lite) sett med *inliers,* finnes en prosedyre som kan estimere parameterne til en modell som optimalt forklarer eller passer til disse dataene.(«Random Sample Consensus», 2020)

ICP algoritme (Iterative closest point) er en algoritme som brukes for å minimere forskjellen mellom to point clouds. ICP brukes ofte til å rekonstruere 2D- eller 3D-overflater

fra forskjellige skanninger, for å lokalisere Roboter og oppnå optimal *path planning* (spesielt når odometri er upålitelig på grunn av glatt terreng). («Iterative Closest Point», 2021)

Kalman filter

I statistikk og kontrollteori er Kalman-filtering, også kjent som lineær kvadratisk estimering (LQE), en algoritme som bruker en serie målinger observert over tid, som inneholder statistisk støy og andre unøyaktigheter, og produserer estimater av ukjente variabler som pleier å være mer nøyaktig enn de som er basert på en enkelt måling alene, ved å estimere en felles sannsynlighetsfordeling over variablene for hver tidsramme. Filteret er oppkalt etter Rudolf E. Kálmán, en av de viktigste utviklerne av teorien.(«Kalman Filter», 2021)

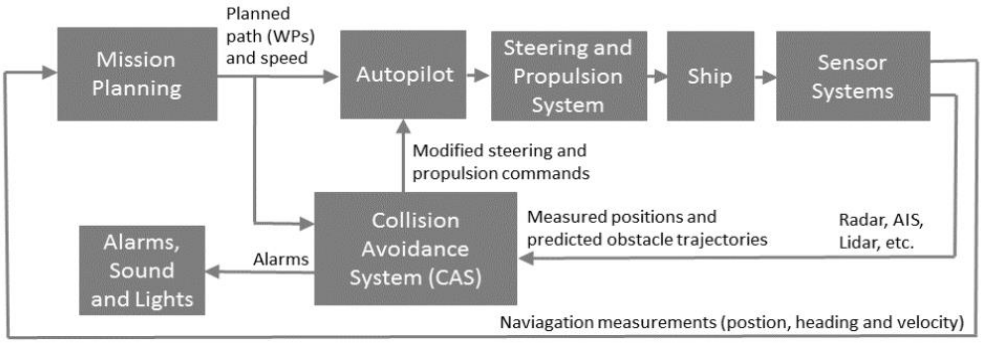
***Extended kalman filter* (EKF)**

I estimeringsteorien er det utvidede Kalman-filteret (EKF) den ikke-lineære versjonen av Kalman-filteret som linjerer rundt et estimat av gjeldende gjennomsnitt og kovarians. Når det gjelder veldefinerte overgangsmodeller, har EKF blitt ansett som de facto-standarden i teorien om ikke-lineær tilstandsestimering, navigasjonssystemer og GPS. («Extended Kalman Filter», 2021)

Partikkelfilter

Roboten bruker Lidaren til å å måle opp og beregne posisjon og avstand til forskjellige objekter ved hjelp av fart, tid og ved å danne seg et omgivelses-bilde og bruker dette ved hjelp av innlagt kunnskap. Noisy Lidar og noisy odometer gjør at man trenger et kalman filter som gjør en serie av målinger over tid mens partikkelen beveger seg rundt og beregner sannsynligheten over hvor man er i omgivelsene. Man fjerner så de med lav sannsynlighet og over tid så vil man sitte igjen med det mest sannsynlige posisjonen og stien den har tatt.
Man kan så bruke monte carlo localization ved å plassere flere partikler i områder med høy sannsynlighet og færre ved lav sannsynlighet. (MATLAB, 2020b)

CAS *(Collision avoidance system)*



Figur 7:

Figur 7 illustrerer det overordnede konseptet med de ulike undersystemer og informasjonsflyten mellom dem. Den nominelle inngangen til skipets autopilot fra *Mission Planner* systemet antas å være fremdrift eller *speed over ground,* og ønsket bane gitt som en sekvens av veipunkter. *Collision Avoidance System* (CAS) tar hensyn til hindringene sine posisjoner og forventede baner, og finner kollisjonsfrie baner nær skipets nominelle bane som tilfredsstiller COLREGS reglene. Dersom det er et hinder i veien, sender CAS ut en modifisert kurs og en fremdrifts kommando til autopiloten. Legg merke til at CAS må vurdere baner med eksplisitt fremstilling av tid (dvs. hvilken posisjon dronen har med tiden) mens autopiloten bare sørger for at banen blir fulgt og ikke ved hvilket tidspunkt dronen er i en bestemt posisjon, Hastigheten til dronen holdes normalt nær en nominell cruisehastighet, men kan reduseres, settes til null eller reverseres, etter kommando fra kollisjonsunngåelsessystemet (CAS). Dronens innebygde navigasjonssystem gir målinger fra et globalt navigasjonssatellitt-system (GNSS) av posisjon og hastighet. Nøyaktigheten av GNSS-posisjonsmålinger er vanligvis 10 meter eller bedre, dette er ikke tilstrekkelig for å utføre oppgavene i AutoDrone 2021 GNSS målingene vil derfor inngå i et *Inertial Measurement System* (IMS) og eventuelt også en *sensor fusion* med *SLAM* (som bruker Lidar og video).

For å kunne beregne baner slik at vi unngår kollisjoner trenger kollisjonsunngåelse systemet CAS følgende informasjon:

• Liste over hindringens posisjoner og hastigheter, fra radar, Lidar, AIS, kamera eller infrarødt varmekamera, eller lignende sensorer og sporingssystemer. • Kartlagte farer fra et elektronisk kart.

• En ønsket nominell sti til mål destinasjonen.

• Matematisk modell av skip for prediksjon av fremtiden

bane for å beregne effekten av styre- og fremdrifts kommandoer, samt vind og havstrømmer.

• Sanntidsmåling av skipets posisjon, hastighet, *heading* og *yaw rate.*

• Estimater av vind- og havstrøm styrker på skipet.

Den foreslåtte arkitekturen innebærer at funksjonen for å unngå kollisjon er atskilt fra funksjonen for oppdragsplanlegging, og kommandoene fra begge disse systemene utføres av skipets autopilot. Dette fører til en svært modulær arkitektur som lar et *collision avoidance system* CAS legges til på toppen av eksisterende funksjonalitet, og slik at pålitelighet og sikkerhet kan sikres gjennom ekstra uavhengige og redundans systemer og funksjoner. (Johansen et al., 2016)

Figur 8:

En oversikt over den foreslåtte CAS-styrings algoritmen er gitt i figur 8. *Collision avoidance* funksjonaliteten blir realisert ved at man beregner den fremtidige banen til eget skip og til andre skip en gitt tidshorisont fremover. Denne beregningen gjøres flere ganger, der man mellom hver gang endrer på styringsvariablene hastighet og kurs. Basert på alle disse utregningene finner man hvilken av banene som minimerer en kostnadsfunksjon. Kostnadsfunksjonen er en funksjon som øker med skaden man forventer dersom en kollisjon inntreffer. En enkel kostnadsfunksjon kan være proporsjonal med kvadratet av den relative hastigheten mellom objektene som kolliderer (dvs. proporsjonal med kinetisk energi). Optimaliseringsproblemet løses på nytt etter hvert som det kommer inn nye målinger av posisjon og hastighet på eget og andre fartøyer. F.eks. kan en oppdatering skje hvert 5 sekund. Faren forbundet med skipets bane som følge av en gitt styrings input (valgt kurs og fart) blir evaluert ved hjelp av en skipssimulator for å lage estimater som tar hensyn til dynamikken i skipet, styrings- og fremdriftssystem, den nåværende posisjonen og hastigheten, styrings

input, samt vind- og havstrøm. Robusthet oppnås ved å sette en passende sikkerhetsmargin og muligens ved å evaluere ytterligere scenarier som følge av forstyrrelse av inngangsdataene for å representere usikkerhet i hindringens fremtidige baner. En kostnadsfunksjon måler de forutsagte farene for grunnstøting og kollisjon, og overholdelse av reglene til COLREGS, ved å bruke hastighets- og *line of sight* vektorer for å uttrykke COLREGS-reglene. Den foreslåtte optimaliseringen er deterministisk og garanterer at det globale minimumet blir funnet etter et kjent begrenset antall kostnadsfunksjon evalueringer. (Johansen et al., 2016)

YOLO ***(You Only Look Once)*** er en effektiv sanntid objekt gjenkjennings algoritme. ***Image classification*** **(**bilde klassifisering) er en av mange spennende applikasjoner i ***convolutional neural networks*** **(CNN).** Foruten enkel bilde klassifisering er det mange andre fascinerende problemer innen computer vision, med *object detection* som en av de mest interessante. Det blir vanligvis assosiert med selvkjørende biler hvor systemer blander *computer vision*, Lidar og andre teknologier for å lage en multidimensjonal representasjon av veien og alle dens deltagere.

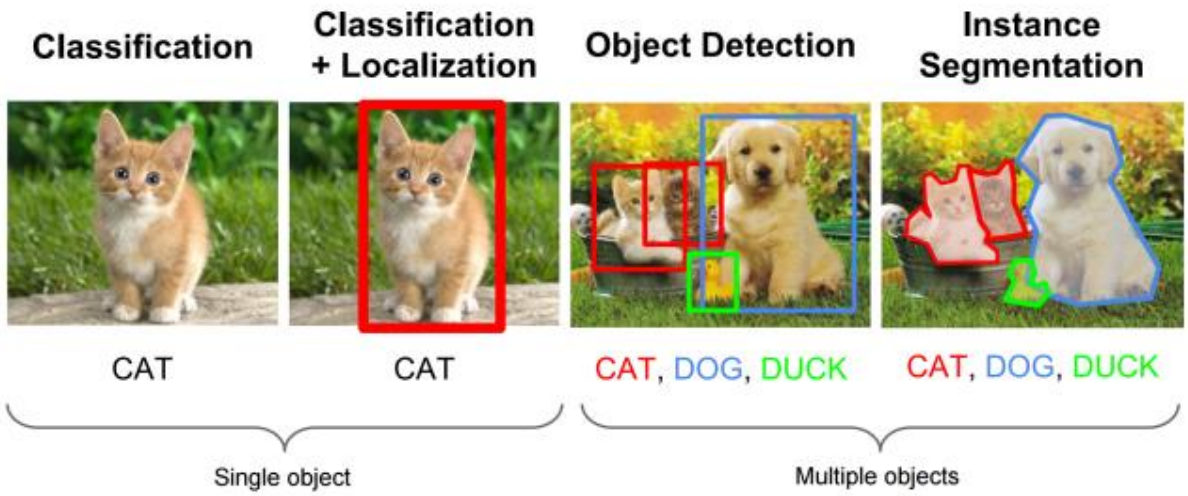
For å utforske konseptet *object detection* er det nyttig å starte med *image classification*. Bilde Klassifisering går igjennom nivå av inkrementell kompleksitet.

Image classification **(1)** sikter på å tilegne et bilde til en av et nummer av forskjellige kategorier (f.eks. en bil, hund, katt, menneske osv.), og i hovedsak svare på spørsmålet «Hva er i dette bildet?». Et bilde har bare en kategori tilegnet seg.

Object localization **(2)** gjør dermed slik at vi kan lokalisere objektet vårt i bildet, slik at spørsmålet blir forandret til «Hva er det og hvor er det?».

I virkeligheten må vi gå lenger enn å lokalisere bare et objekt, men heller mange objekter i et bilde. For eksempel må en selvkjørende bil finne ut hvor plasseringen av andre biler er, trafikklys, skilter, myke trafikanter og gjøre nødvendige handlinger basert på denne informasjonen.

Object detection **(3)** gir verktøyene man trenger for å gjøre akkurat det - finner alle objektene i et bilde og tegner såkalte *bounding boxes* rundt dem.



Figur 9: Viser eksempler på hvordan *bounding* boksene kan se ut.

Det er noen forskjellige algoritmer for *object detection* og de kan deles inn i to grupper:

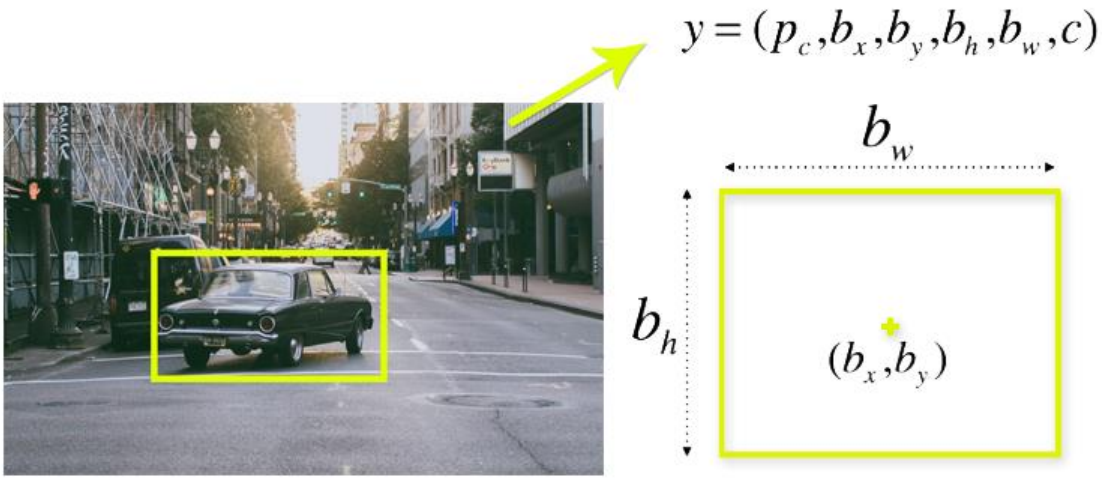
1. Algoritmer basert på klassifisering. De blir implementert i to trinn. Først velger de ut *regions of interest* (ROI) i bildet, så klassifiserer de dem ved å bruke *convolutional neural networks*. Denne løsningen kan være treg fordi man må kjøre predictions for hver valgt region. Velkjente typer av denne type algoritme er *Region-based convolutional network* (RCNN) og slektningene Fast-RCNN og Faster-RCNN.
2. Algoritmer basert på regresjon- i stedet for å velge ut interessante deler av et bilde, forutser de klasser og *bounding boxes* for hele bildet ved å bare kjøre algoritmen en gang. De to best kjente eksemplene i denne gruppen er YOLO *(You Only Look Once)* algoritme familien og SSD *(Single Shot Multibox Detector)*. De blir ofte brukt i sann tid *object detection* da de generelt ofrer litt nøyaktighet for store forbedringer i hastighet.

For å forstå YOLO algoritmen er det nødvendig å etablere hva som egentlig blir forutsett. Til syvende og sist sikter vi på å forutse en type/klasse av et objekt og *bounding* boksen som spesifiserer objektets plassering.

Hver *bounding box* kan bli beskrevet ved å bruke fire deskriptorer:

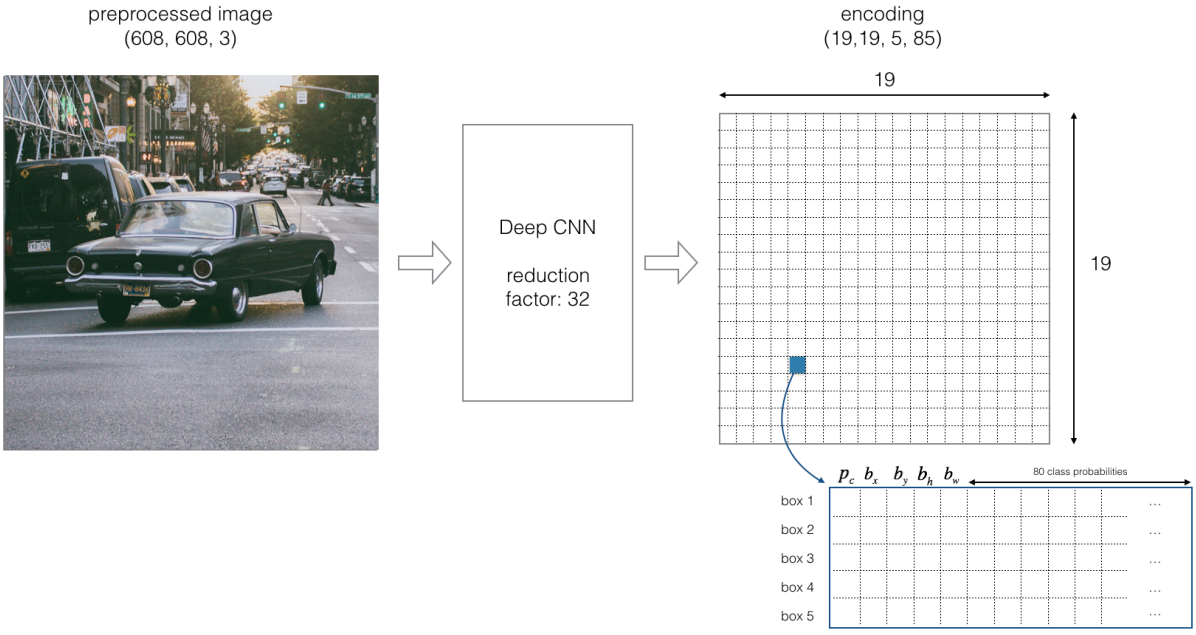
1. senteret av en *bounding box* (bxby)
2. Bredde (bw)
3. Høyde (bh)
4. Verdi c som korresponderer til klassen objekt (bil, trafikklys, osv.)

I tillegg må vi forutse pc verdi som er sannsynligheten for at det er et objekt i *bounding boksen.*



Figur 10: viser formel på hvordan man regner ut pc verdi.

Som nevnt tidligere så søker man ikke etter interessante regioner i bildet som kan inneholde objekter med YOLO algoritmen. I steden deler man bildet opp i celler, typisk et 19x19 nett. Hver celle er ansvarlig for å forutse 5 bounding boxes (i tilfelle det er flere en et objekt i denne cellen). Derfor kommer vi til et stort antall på 1805 *bounding boxes* for et bilde. Se figur 11.



Figur 11:

De fleste av desse cellene og *bounding boksene* vil ikke inneholde et objekt. Derfor forutser vi verdien pc som benyttes til å fjerne bokser med lav objekt sannsynlighet og *bounding bokser* med høyeste delte område i en prosess kalt *non-max suppression.*

Figur 12: Viser hvordan non-max suppression fjerner bokser med lav objekt sannsynlighet og ender opp med en bounding box for objektet..

(*YOLO Algorithm and YOLO Object Detection*, 2020)

Object tracking

I et dynamisk trafikkbilde må man estimere et objekts posisjon, kinematikk, og et objekts størrelse, form og orientering ved bruk av Lidar. Man må prøve å assosiere data til objekter. Man må ta klynger av datapunkter å sannsynliggjøre om de tilhører samme objekt. Dette kalles *partitioning.* Man kan så bruke *Cluster Analysis* som for eksempel *DBSCAN* til å gruppere de sammen. Ved å bruke innstillings parametere bestemmer man maks distanse mellom punktgrupper, og minimum antall punkt i en gruppe. Ved å oppdatere *object trackeren* kontinuerlig vil man kunne skille ut et objekt som ikke hører hjemme i det objektet du først hadde. Klynger som er veldig usannsynlige blir fjernet og man beholder bare de mest sannsynlige.

Extended object tracking estimerer et objekts tilstand, form, posisjonen og hvordan den beveger seg. Den kan da forutsi objektets bevegelse og hvor det vil være fram i tid. Den matematiske modellen (kalman filteret) i *the extended object tracker* vil da kunne forutsi hva

som kommer tilbake av data fra Lidaren forutsatt den vet hvor Lidaren er relativt til hvor den tror objektet er. Når vi så får en virkelig måling og partisjonert deteksjonene inn i grupper kan vi oppdatere tilstanden til det sporede objektet basert på uoverensstemmelsene mellom forutsigelsene og deteksjonene. Man bruker Usikkerhetene i dette til å bestemme hvordan man oppdaterer tilstand og å bestemme hvilken hypotese som er mest sannsynlig.(MATLAB, 2020e, s. 5)

2.4 BEGREPER OG FORKORTELSER

Autonomt system

Et autonomt system består av følgende funksjoner:

Sense - Samle data om området

Perceive - Tolke data, spore objekter, bygge modeller, estimere tilstand, lokalisering.

Plan - lage en plan for å nå målet

Act - Kontrollere kjøretøyet til å følge stien.

En robot kan for eksempel bruke en Lidar for sensing, lage seg et 3D-bilde av rommet med hindringer, og så bevege seg gjennom rommet fra en startposisjon til en målposisjon. (MATLAB, 2020a, s. 1)

Situasjonsforståelse: For at et fartøy skal kunne operere med et nivå av er det avgjørende med en nøyaktig forståelse av omgivelsene rundt fartøyet. Situasjonsforståelse kan sies å være å ha en god oppfatning av ditt fartøys omgivelser til enhver tid, forstå og forutsi hvordan omgivelsene kan virke inn på fartøy. Et skips omgivelse er et bilde som er under konstant utvikling. Derfor er det essensielle parametere for god situasjonsforståelse som kunnskap om andre fartøy i området, kommunikasjon mellom skipstrafikk tjenester og andre mindre båter, en forståelse av vær, bølgehøyde, dybde, flo og fjære, og strøm også avgjørende. å oppnå en forståelse av alle omliggende parametere kan være utfordrende. Derfor er det nødvendig med sensorer som gir korrekt og nøyaktig informasjon av fartøyets omgivelser. (Thunes, 2019)

GNSS: *Global "Navigate Satellite System"* bruker signal fra satellitter i verdensrommet for å bestemme en posisjon. Dette gjør de ved hjelp av tid og posisjonsdata til forskjellige satellitter.

GNSS har global dekning og er det mest brukte navigasjonssystemet for marine fartøyer. For at vi skal oppnå et pålitelig resultat i en GPS-posisjon må minst 4 satellitter være synlig. Dersom fartøyet går inn i et skyggelagt område, vil posisjonen gå tapt så lenge det ikke finnes noen overflødige signaler tilgjengelig. (*What Is GNSS?*, 2016)

Navigation: Robot navigasjon er robotens evne til å finne sin egen posisjon i sin referanseramme og så planlegge en sti mot et lokalitets mål. For å kunne navigere i omgivelsene, trenger roboten eller enhver annen mobil enhet representasjon, et kart av omgivelsene og evnen til å tolke den representasjonen.

Navigation kan defineres av tre fundamentale kompetanser

1. *Self- localisation*
2. *Path planning*
3. *Map building and map interpretation*

(«Robot Navigation», 2021)

Pathfinding er å plotte ved bruk av en computer applikasjon den korteste ruten mellom to punkt ved å bruke en algoritme. Det går ut på å starte på et toppunkt i en graf og søke i nodene ved siden av til destinasjons noden er nådd. Som regel brukt for å finne den "billigste" ruten. A* er en av de mest populære algoritmene.(«Pathfinding», 2021)

Path Planning and motion planning (også kjent som navigasjonsproblemet eller pianostrømmerens problem) er et beregnings problem for å finne en sekvens av gyldige konfigurasjoner som flytter objektet fra kilden til destinasjonen. Begrepet brukes i beregnings geometri, dataanimasjon, robotikk og dataspill. Et grunnleggende bevegelsesplanlegging problem er å beregne en kontinuerlig bane som forbinder en start konfigurasjon S og en må konfigurasjon G, mens du unngår kollisjon med kjente hindringer. Roboten og hindringsgeometrien er beskrevet i et 2D- eller 3D-arbeidsområde, mens bevegelsen er representert som en bane i (muligens høyere dimensjonal) konfigurasjons rom.(«Motion Planning», 2021)

Waypoint er et punkt eller en koordinat som angir et fysisk referansepunkt brukt for navigasjon ved å bruke koordinatene i breddegrad og lengdegrad på land og sjø. ble først

vanlig å bruke i navigasjon etter at avanserte navigasjonssystemer som f.eks *Global Positioning System* (GPS) ble innført. (Fossen, 2002)

Waypoint tracking er det transformere waypointene til en gjennomførbar kurs eller bane. Det er generelt en ikke-lineært optimaliseringsproblem.
(Fossen, 2002)

Waypoint Guidance Systems
Systemer for w*aypoint guidance* blir brukt av både skip og undervannsfarkoster. Disse systemene består av en *waypoint*-generator med menneskelig grensesnitt. De utvalgte *waypointene* blir lagret i en *waypoint* database og bruk for generering av en bane eller kurs for fartøyet å følge. Både *trajectory* og manøvrerings styresystemer kan bli laget for dette formålet. ((Fossen, 2002)

Waypoint Representation
Ruten til et skip eller undervannsfartøy blir vanligvis spesifiser i form av *waypoints.* Hvert *waypoint* blir definert av kartesiske koordinater. For overflatefartøy blir bare to koordinater brukt (x, y). I tillegg kan man legge inn andre *waypoint* verdier som fart, heading osv. *Waypoint* databasen kan bli generert ved å bruke mange kriterier.
Disse er vanligvis basert på:
•***Mission***: fartøyet skal bevege seg fra et startpunkt (x0, y0, z0) til endepunktet (xn, yn, zn) via waypointene (xi,yi,zi)
•***Environmental data:*** informasjon om vind, bølger, og strømmer kan bli brukt for energy optimal routing (eller unngå dårlig vær av sikkerhets hensyn)
•***Geographical data:*** informasjon om grunne farvann,øyer osv. bør inkluderes.
•***Obstacles:*** flytende konstruksjoner og andre hindringer må unngås.
•***Collision avoidance:*** unngå fartøy som beveger seg nær din egen rute ved å legge inn sikkerhetsmarginer
•***Feasibility:*** hvert waypoint må være gjennomførbart, som at det må være mulig å manøvrere til neste *waypoint* uten å overgå maks fart, svinghastighet etc.
(Fossen, 2002)

Output Maneuvering problem

Output maneuvering problemet involverer to oppgaver. Det første, som kalles geometric task, er å tvinge system output til å konvergere til en foretrukket sti parameterisert ved en kontinuerlig skalar variabel θ. Den andre oppgaven, som kalles *dynamic task*, er å tilfredsstille en foretrukket dynamisk oppførsel gjennom stien. Denne dynamiske oppførselen er videre spesifisert via et tids, fart og akselerasjons oppdrag. Mens hovedoppgaven er å tilfredsstille det geometriske oppdraget, sikrer den dynamiske oppgaven at systemets output følger stien med den foretrukne hastighet. En robust tilbakevendende design teknikk blir utviklet for usikre ikke lineære verk i vektor streng *feedback* form. Den geometriske delen av problemet blir løst først. Så blir det konstruert en oppdaterings lov som binder sammen det geometriske designet med farts oppdraget.

(Skjetne et al., 2004)

GNC *(Guidance,Navigation and Control)*

Et bevegelses styresystem er vanligvis konstruert av tre uavhengige blokker betegnet som *guidance,navigation and control systems.* Disse systemene samhandler med hverandre gjennom data og signal overføringer.

Oppgaven til de underliggende systemene er klassifisert som:

Guidance er handlingen eller systemet som kontinuerlig beregner den foretrukne referanse posisjon, hastighet og akselerasjon til et marint fartøy til å bli brukt i bevegelses styresystemet. Disse dataene er vanligvis gitt av en menneskelig operatør eller navigasjonssystemet. De grunnleggende komponentene i et *guidance* system er motion sensors, eksterne data som værdata (vindhastighet og retning, bølgehøyde og helling, strømhastighet og retning) og en computer. Computeren samler og prosesserer informasjonen og mater resultatene til *motion control* systemet. I mange tilfeller blir det brukt avanserte optimaliserings teknikker for å beregne den optimale banen eller kursen for fartøyet å følge. Dette kan inkludere sofistikerte funksjoner som drivstoff optimalisering, minimum tid navigasjon, *weather routing*, kollisjons unngåelse, formasjon kontroll og synkronisering.

Navigation er kunnskapen å styre et fartøy ved å bestemme dets posisjon, kurs og distanse tilbakelagt. I noen tilfeller blir også hastighet og akselerasjon bestemt. Dette blir som regel gjort ved å bruke et *global navigation satellite system* **(**GNSS) kombinert med motion sensorer som akselerometer og gyro. Det mest avanserte navigasjonssystemet for marine applikasjoner er *inertial navigation system* (INS).

Control eller mer spesifikk *motion control* er handlingen å avgjøre den nødvendige kontroll kreftene og momentene som blir gitt av fartøyet for å tilfredsstille et visst kontrollmål. De foretrukne styre målet ses vanligvis i sammenheng med *guidance* systemet. Eksempler på kontroll mål er minimum energi, *setpoint regulation, trajectory-tracking, path-following,* og manøvrerings kontroll. Å konstruere en *control* algoritme inneholder å lage feedback og *feedforward* styrings lover. *Output* fra navigasjonssystemet, posisjon, hastighet og akselerasjon blir brukt som *feedback* kontroll mens *feedforward* kontroll er implementert ved å bruke signaler tilgjengelig i *guidance* systemet og andre eksterne sensorer. (Fossen, 2002)

ASV *(Automonus surface vessel)* V USV *(Unmanned surface vehicle)*

Bruken av begreper som ASV og USV på tvers av litteraturen er ikke alltid enhetlig, og i noen tilfeller er det til og med forvirrende. For eksempel kan et overflatefartøy defineres som et" ikke-lineært underaktuert kinodynamisk system ofte med stor treghet" som opererer i kontinuerlig kontakt med vannoverflaten. Imidlertid er mange moderne fartøy fullstendig aktuert, og tregheten deres trenger ikke nødvendigvis å være stor.

Et autonomt overflatekjøretøy, eller ASV, derimot, er et fartøy som kan ta beslutninger og operere alene, uten menneskelig veiledning, navigering og kontroll. ASV-er brukes vanligvis i militære operasjoner, maritime overvåkings seilaser, marine miljøovervåking applikasjoner og i nær fremtid, sannsynligvis også for transport av varer og mennesker.

Mange artikler i feltet refererer til begrepene ASV og USV (ubemannede overflatefartøyer) som synonymer, og skiller ikke metodikken for disse to typer fartøyer. En USV er et ubemannet kjøretøy som ikke har et menneske om bord for å kontrollere driften, men som vanligvis fjernstyres av en menneskelig operatør. Avgjørende er at en ASV også kan være ubemannet, men det viktige skillet sammenlignet med en USV er at den opererer uten direkte inngrep. (Vagale et al., 2021)

4. Kravspesifikasjon og løsningsstrategier

4.1 Krav til sjø-dronen

- **Autonom** Dronen skal være fullstendig autonom, og alle autonome valg skal være gjort om bord.

- **Kommunikasjon** Dronen skal ikke sende noen styresignaler til eller motta noen styresignaler fra kontrollsentralen når den er i autonom modus.

- **Utplasserbar** Dronen må kunne plasseres manuelt ut i havneområdet der konkurransen skal finne sted.

- **Energikilde** Dronen må være batteridrevet og batteriene skal være forseglet for å redusere risikoen for lekkasje av syre eller kaustiske elektrolytter. Det skal også være mulighet for å koble ut hvert enkelt batteri.

- **Nødstopp** Dronen må ha minst en rød knapp som fungerer som nødstopp plassert på seg, slik at når den aktiveres så vil den øyeblikkelig kutte strømmen til motorer og aktuatorer

- **Trådløs nødstopp** I en nødssituasjon må operatørens kontrollstasjon kunne aktivere nødstopp-knappen om bord på dronen.

- **Fremdrift** Ethvert fremdriftsmiddel kan brukes, men alle bevegelige deler må ha beskyttelse. For eksempel må propellen være innkapslet.

- **Fjernkontrollert** Dronen må kunne fjernstyres fra kontrollstasjonen på land.

.

- **Sleping** Dronen må kunne slepes

- **Visuell tilbakemelding** dronen må ha et visuelt tilbakemeldingssystem som viser dens status, dvs operasjonelle modus. Her skal rødt lys indikere at nødstopp er aktivert, gult lys at dronen styres manuelt, og grønt eller blått lys at dronen er i autonom modus.

- **Vekt** dronen med alt utstyr må veie mindre enn 70 kg.

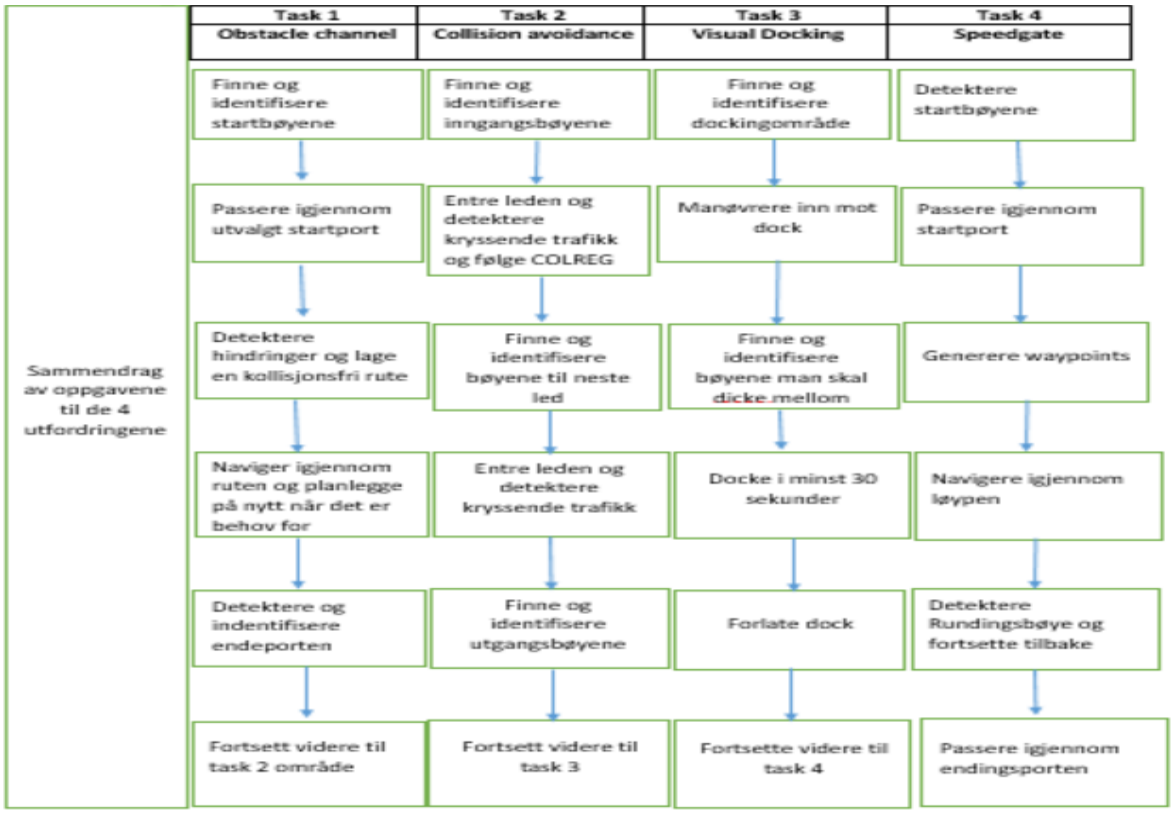
- **Nyttelast** Dronen må ha et sted for å montere et action kamera med uhindret utsikt fremover.

4.2 Funksjonalitet for å løse de forskjellige oppdragene

	Funksjon	**1.** ***Obstacle channel***	**2.** ***collision avoidance***	**3.** ***visual docking***	**4.** ***speedgate***	***Manual***
	Figur deteksjon	x	x	x	x	0
	Farge Deteksjon	x	x	x	x	0
	Relativ Posisjons deteksjon	x	x	x	x	0
Navigasjon	GPS	x	x	x	x	0
Navigasjon	*Waypoint generator*	x	x	x	x	0
kommunikasjon	*WIFI*	x	x	x	x	x
Nødstopp	*Emergency Stop (Manual)*	x	x	x	x	x
Nødstopp	*Emergency Stop (Remote)*	x	x	x	x	x

Figur 13: Er en matrise som viser hvilke evner som trengs for å løse oppdragene. Dette er en modifikasjon av tabell 1 i (Ming & Velamala, 2018)

Mission planner



Figur 14: Viser en oversikt av misjons planleggeren, og beskriver prosedyren for å løse de fire oppdragene. I tillegg til de fire oppgavene må alle overgangene mellom oppgavene utføres autonomt. Derfor må det legges inn flere unntaks og feilbehandlings rutiner slik at mission planneren ikke går seg fast ved mislykkede forsøk. (Dette er en modifikasjon av figur 8 i (Park et al., 2016)

Navigasjon og kontrollsystem.

Det er viktig med et integrert navigasjonssystem som kombinerer en *inertial measurement unit* (IMU) som vil gi informasjon om fartøyets bevegelse og et *global positioning system* (GPS) som gir posisjonen og vil gi høy grad av presisjon til navigeringen av fartøyet. I tillegg er det nødvendig med optiske sensorer som Lidar (vi bruker 2D Lidar med *nodding* mekanisme for å få 3D sansing), kameraer for *stereo vision* og prosessorer som trengs for å tolke sensor data for å kunne utføre relativ navigasjon i henhold til omliggende strukturer som bøyene i leden. Det er og viktig med riktig fremdrift/aktiveringssystem og kontroll algoritmer. Spesielt så kan det være utfordrende å kontrollere fartøyet med vind og bølger ute i sjø.

***Navigation, Guidance and control* til konkurransen**

Vi har her tatt utgangspunkt i hvordan (Kang et al., 2015) har valgt å løse oppdragene i RoboX konkurransen i 2014.

En Kalman-filter-algoritme kan brukes for å lage et navigasjons-filter. Filteret vil har tre frihetsgrader (3 grader av frihet (DOF)) for bevegelsen til dronen. De tre frihetsgradene er *surge, sway og yaw.* Dronens bevegelse blir estimert av filteret basert på målinger av posisjon og vinkelposisjon fra GPS og IMU. Tilstandsvektoren for å representere denne 3- DOF-bevegelsen kan uttrykkes som

$x = [x\ y\ \psi\ u\ v\ \psi]$

hvor x, y og ψ representerer posisjonen og kursen til dronen i det globale referansesystemet. u og v er dronens hastighet i en *body fixed frame.* (se figur 15)

Erfaringer fra Kang (Kang et al., 2015) viser at dronen vil være i stand til å lokalisere seg selv og fastsette sin kurs på en pålitelig måte med dette 3-DOF-baserte GPS-IMU integrerte navigasjonssystemet.

Imidertlig, er nøyaktig sensor måling som lidars punktsky-data viktig for å utføre oppgavene. Fordi lidaren gir punktsky-dataene i dronens body-fixed frame, er det nødvendig å kompensere roll- og pitch bevegelser av dronen fra disse sensor målingene. Av denne grunn må disse ekstra tilstandene i tillegg målt ved hjelp av IMU, og legges til tilstandsvektoren slik at den blir:

$x = [x\ y\ \psi\ \varphi\ \theta\ u\ v\ \psi\ \varphi\ \theta]$

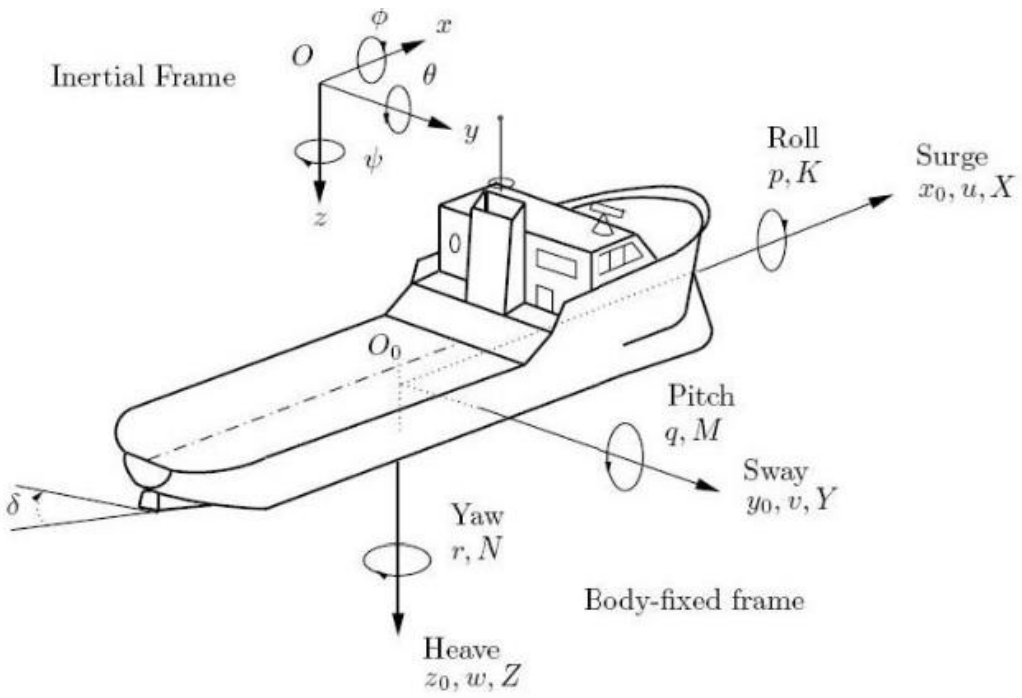
For navigasjon og styring brukte (Kang et al,. 2015) tre manøvrerings modi *(waypoint tracking, weathervaning, position holding).* For Autodrone konkurransen trenger vi bare to av disse: *waypoint tracking og position holding.*

Waypoint tracking er en viktig manøvreringsevne for alle oppgavene. *Waypoints* ble forhåndsdefinert i henhold til kursinformasjonen eller generert av båtens planlegger. Veiledning for waypoint tracking ble brukt for å minimere *line of sight errors* og *cross track errors* foreslår vi at det brukes en *waypoint tracking* algoritme som beskrevet i (Fossen et al. (2003)). Styrings Kreftene som minimerte disse feilene ble bestemt i retning av surge og yaw, og styring kreftene realiseres ved å betjene dronens to hoved propellere. Kang (Kang et al., 2015) rapporterer at dronen effektivt lar seg styre over et bredt spekter av hastigheter i denne

modusen, mens sway bevegelsen ikke kan kontrolleres på grunn av dronens underaktuerte egenskaper i denne modusen.

***Position keeping**:* I *position keeping* modus må to baugpropeller brukes i tillegg til de to hoved propellene, ble to baugpropellere også brukt til å kontrollere drone-bevegelsen i posisjon og orientering samtidig. Dette gjorde det mulig for båten å opprettholde den nåværende posisjonen (dvs. posisjon og retning) og / eller spore en sekvens med pose kommandoer. Denne egenskapen vil være spesielt nyttig for *docking* oppgaven.

I Kang (Kang et al., 2015) ble *Thrust* alokeringsproblemet forenklet til et lineært-ubegrenset optimaliseringsproblem, så kontroll allokering ble oppnådd ved enkel matriseberegning. Ved bestemmelse av styrker på styringssignaler ble en generell PD- styrings lov brukt. Hensikten var å minimere avvik i manøvrerings modusene.

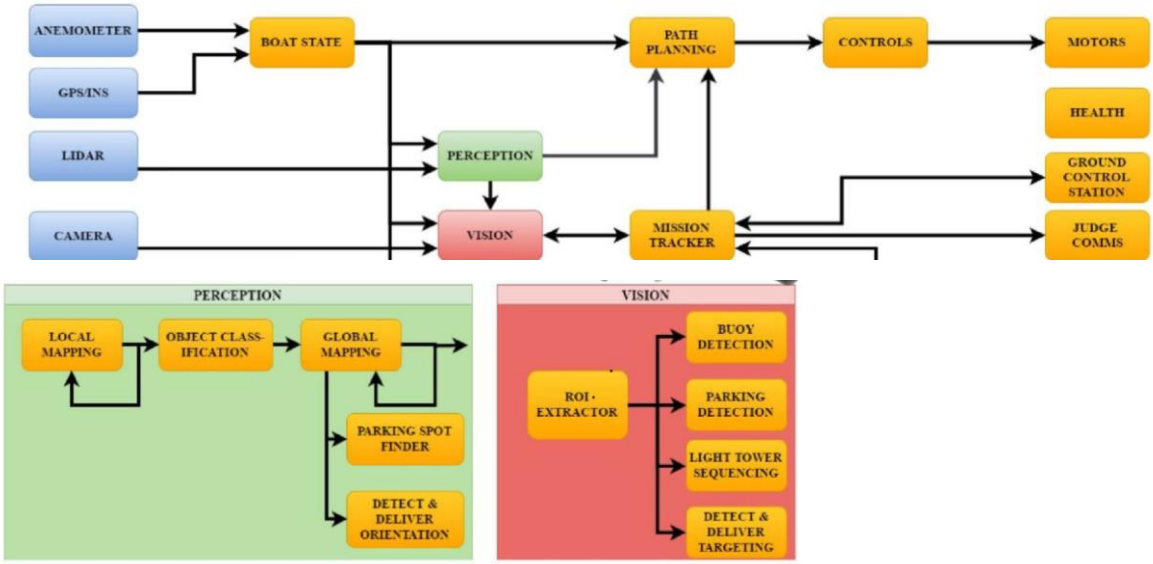


Figur 15: Viser et skips body fixed frame og dets 6 bevegelsesfriheter (6 DOF). Hentet fra fig.2 i (Li et al., 2009).

Vi trenger også en *object detection* algoritme og vi vil foreslå å bruke YOLO.

For å trene opp YOLO algoritmen vil vi skaffe bilder av bøyene, kaien og fartøyene som skal brukes som kryssende trafikk i *collision avoidance* oppgaven. Ved å kjøre 7000 iterasjoner i algoritmen vil den ha opp mot 90 prosent suksess med å gjenkjenne farge og bøyer i konkurransen ifølge (Øksne et al., 2020)

For *path planning* velger vi å bruke A* som er en *search based method* algoritme. Fordelen med den er at den ikke velger laveste verdi på hver node (slik for eksempel en grådighetsalgoritme ville gjort) men kan ta en med høyere verdi hvis det endelige resultatet blir lavere. Den trenger heller ikke å legge inn noder i hele rutenettet.
De passer best i mindre områder da det er kostnadskrevende.
I større rutenett som kan inneholde millioner ruter er *sampling based methods* mer nyttige. RRT er som et tre med greiner som brer seg ut og dekker hele området. Ved å plassere tilfeldige noder ut i kartet. Det gjør at en gren alltid vil utvikle en vei mot målet. RRT* vil så optimalisere grenen ved å plassere ut noder for å rette ut banen og gjøre distansen mindre.(MATLAB, 2020d, s. 4)



Figur 16: Viser et forslag på en software modell for dronen modifisert for å passe oppdragene i Autodrone 2020 tatt fra (Ash et al., u.å.)

.

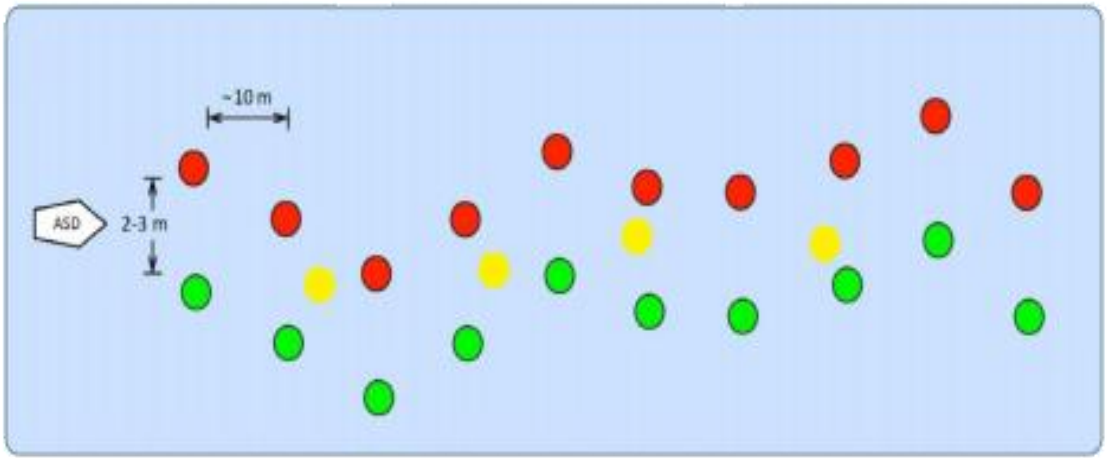
4.3 Oppdragene

Docking og Colreg er nye øvelser i denne type konkurranse.

Våre forslag for løsningsstrategier og algoritmer til disse oppgavene er derfor i stor grad våre egne selvstendige bidrag til hvordan oppgaver av denne typen skal løses.

Under *Visual Docking* har vi valgt å ta med to forskjellige løsninger siden *docking* oppgavene i de artiklene vi fant var en annen type *docking*-øvelse enn det som er oppgitt i vår oppgave.

4.3.1 Obstacle channel



Figur 17: Illustrasjon av *obstacle channel*

Points		
Gate	**Obstacle**	**Time**
+5	-10	100 - 10/min

Tabell 1: Viser poengene for hver port og straffepoengene for hver kollisjon med hinder.

Oppdragskrav

I en suksessfull gjennomføring av *obstacle channel* demonstreres dronens evne til å sanse og manøvrere gjennom en kompleks led, holde seg innenfor den oppmerkede løypen, og unngå å komme i kontakt med hinder langs ruten.

Dronen passerer mellom flere sett med porter bestående av røde og grønne bøyer. Hele dronen skal passere gjennom alle portene uten å berøre bøyene. For å score poeng på en port må den passeres korrekt en gang, man får ikke flere poeng ved å passere en port flere ganger. Dronen må også unngå gule bøyer utlagt tilfeldig i løypen. kontakt med en gul bøye gir straffepoeng. dette er en oppgave som går på tid og hvert minutt en bruker på oppgaven blir trukket fra en start poengsum.

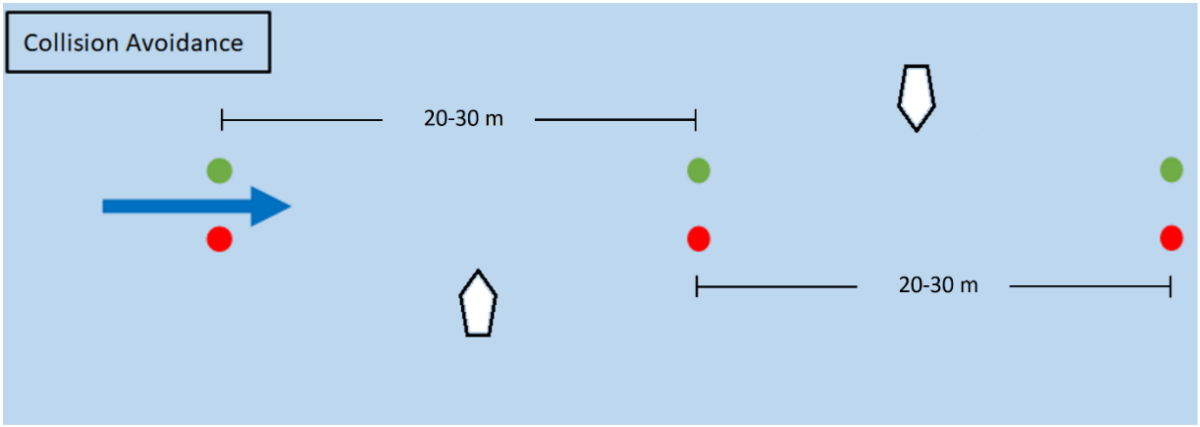
Målet for Task 1. er å krysse en led som er merket med sett av røde og grønne bøyer med gule bøyer satt ut som hindringer i. For å navigere mellom portene autonomt må dronen generere *waypoints* ved å å bruke den relative distansen mellom portene og headingen til portene. En *waypoint tracking control* algoritme kan brukes til å holde kursen. Inngangs og utgangs portene kan finnes ved å bruke Lidar og/eller kamera ved å kombinere sensor målingene og tidligere gitt informasjon om posisjonen for start og endepunkt for oppgaven. Fartøyet må også oppdage og unngå tilfeldig utsatte gule bøyer som er hindringer mens den seiler til endepunktet i oppgaven. Lidar vil da være hoved sensoren til å oppdage hindringer. Ved å bruke en *point distance based clustering* algoritme (PDBS) for å samle klynger med datapunkt til hindringer. En viktig del av denne oppgaven er å fjerne falske eller støyende Lidar målinger i klyngene med datapunkt. For å minimere forstyrrende effekter kan tidligere informasjon om hinderets størrelse og statistiske kjennetegn av Lidar målingen bli brukt. Denne oppgaven går på tid og hvert minutt man bruker blir trukket fra en poengsum man starter med.

Strategi og algoritme

Denne oppgaven går ut på å følge leden ved å passere riktig inngangs og utgangs port, unngå alle hindringer uten å kollidere med bøyene i leden. For å fullføre oppgaven må dronen gjenkjenne start og mål portene og velge inngangs bøyene og de neste bøyene i ruten og i tillegg til å oppdage og unngå hindringer. Vi har valgt å løse denne oppgaven ved å se på hvordan (*KAIST_RX14_Paper.pdf*, u.å.) har løst et lignende problem. Strategien for å gjøre dette er å manøvrere mellom portene ved å generere *waypoints* og utføre *waypoint tracking control*.

Dronen starter å forflytte seg fra et startpunkt ved å bruke en gitt GPS posisjon og utfører *heading control* mot neste *waypoint.* Ved å bruke en *nodding lidar system* samles *point cloud data* og en *point based clustering* algoritme brukes for å detektere port bøyene ved startpunktet og hindrings bøyene. For *path planning* og *obstacle avoidance* blir A* algoritmen brukt. *Path* planneren ser etter en rute som har den korteste avstanden fra start til målet uten å kollidere med noen av hindringene. Kart med hindringer blir ikke gitt før oppgaven og deteksjon prosessen har målefeil og usikkerheter, noe som gjør at man trenger en *online path planning* funksjonalitet. Derfor blir kartet med hindringer oppdatert hver gang dronen oppdager nye hindringer i synsfeltet. Ruten blir kontinuerlig planlagt på nytt basert på det oppdaterte hindrings kartet for å ta hensyn til feilaktige sensor målinger og miljømessige usikkerheter.

4.3.2 *Collision avoidance*



figur 18: Illustrasjon av *collision avoidance* oppgaven. Oppgaven starter med at dronen passerer en grønn og en rød bøye som utgjør startporten. Det vil komme en kryssende båt fra styrbord eller babord og dronen skal benytte forkjørsrett eller vike i henhold til COLREG. Dronen passerer så en ny port bestående av en rød og en grønn bøye, møter igjen en kryssende båt hvor dronen skal følge COLREG. Til slutt passeres målporten som består av en rød og en grønn bøye.

Points		
Gate	**Collision**	**COLREGs**
+10	-30	+30/rule followed

Tabell 2: viser poeng for hver port og for å følge COLREG og minuspoeng for kollisjon.

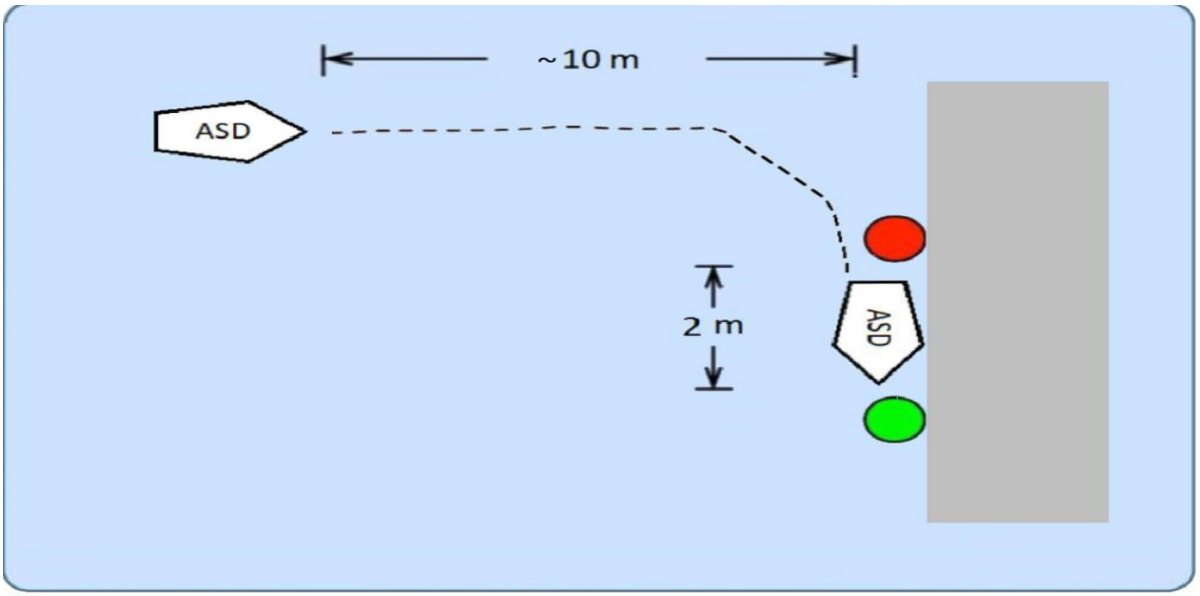
Oppdragskrav

Målet med *collision avoidance* oppgaven er å demonstrere kompleks *path planning.* Dronen må forflytte seg gjennom et område med kryssende trafikk fra både styrbord og babord. Dronen skal så oppføre seg etter COLREGS. Retningen på den kryssende trafikken vil være helt tilfeldig. Dronen må holde en fart på under fire knop og fullføre oppgaven innen 10 minutter. Til å løse denne oppgaven trengs det et *Guidance, Navigation and Control* system (GNC) som har lagt inn COLREG og parametere for at man ikke skal overstige max hastigheten.

Strategi og algoritme

I denne oppgaven skal dronen gå gjennom to kanaler med 20-30 meters bredde gjennom porter består av en rød og en grønn bøye. Dronen har fått oppgitt en GPS-posisjon som den skal starte fra og må så identifisere start porten inn til kanalen på samme måte som i oppgave 1. Dronen vil så starte *path planning* ved bruk av *waypoints* gjennom kanalene med hjelp av A* algoritmen. For å finne og gjenkjenne bøyer og kryssende trafikk vil vi bruke en *sensor fusion* av en *nodding lidar* og stereokamera. *Lidaren* vil egne seg til å finne at vi har et objekt og hvilken farge objektet har. For å ta hensyn til kryssende trafikk vil vi bruke Yolo algoritmen. Yolo vil estimere et objekts tilstand, form,dets posisjon og og hvordan det beveger seg. Den vil da kunne forutsi objekters bevegelse og hvor de vil være frem i tid. Den foretrukne ruten vil kontinuerlig bli planlagt på nytt basert på hindringer som blir lagt til i *obstacle map,* noe som krever *online path planning* evne. Vi vil trenge et GNC system for å utføre oppgaven og for å unngå kryssende trafikk vil vi bruke et *Collision Avoidance System* (CAS) som er COLREG kompatibelt. Den nominelle inputen i autopiloten fra *mission planneren* er tenkt å være fremdriften eller *speed-over-ground* (SOG) kommandoer. Den foretrukne ruten vil være gitt som en serie *waypoints*. CAS vil da søke etter baner som er kollisjonsfrie og i samsvar med COLREG. Banene som CAS finner vil ligge nær dronens nominelle bane, men ta hensyn til de målte posisjonene og sannsynlige banene til hindringene. basert på banen som CAS finner vil det bli gitt nye styring og fremdrifts kommandoer til autopiloten. Vil så bruke *waypoint tracking* for å transformere *waypointene* til en gjennomførbar kurs.

4.3.3 Visual Docking



Figur 19: Illustrasjon av *visual docking* hvor dronen skal lokalisere kaiområde og en rød og en grønn bøye en skal docke imellom. Den skal så ligge til kai ved hjelp av *thrusterene* i minst 30 sekunder før den forlater kai og går videre til neste oppgave.

Points		
Dock Reached	***Docking correct***	***Docking time***
+10	+30	+ 2 /sec docked

Tabell 3: Viser poeng gitt for å nå *docken*, *docke* riktig og for hvert sekund den ligger inntil *docken*.

Oppdragskrav

I *visual docking* oppgaven skal dronen demonstrere evnen til å lokalisere docking området og manøvrere inn til kaien i et definert område. Området på kaien man skal dokke i, vil være definert av en grønn og en rød bøye to meter fra hverandre. Dronen skal naviger til området mellom bøyene og legge til med babord eller styrbord side til kaien. Den skal ligge til kaien i minst 30 sekunder og kan bruke *thrustere* for å holde seg inntil kaien. Etter 30 sekunder skal

den forlate kaien. Som tabell 3 viser vil poeng blir gitt for å ankomme *docking* området, korrekt *docking* og for hvert sekund dronen holder posisjonen ved kaien.
Oppdraget må fullføres på mindre en 10 minutter.

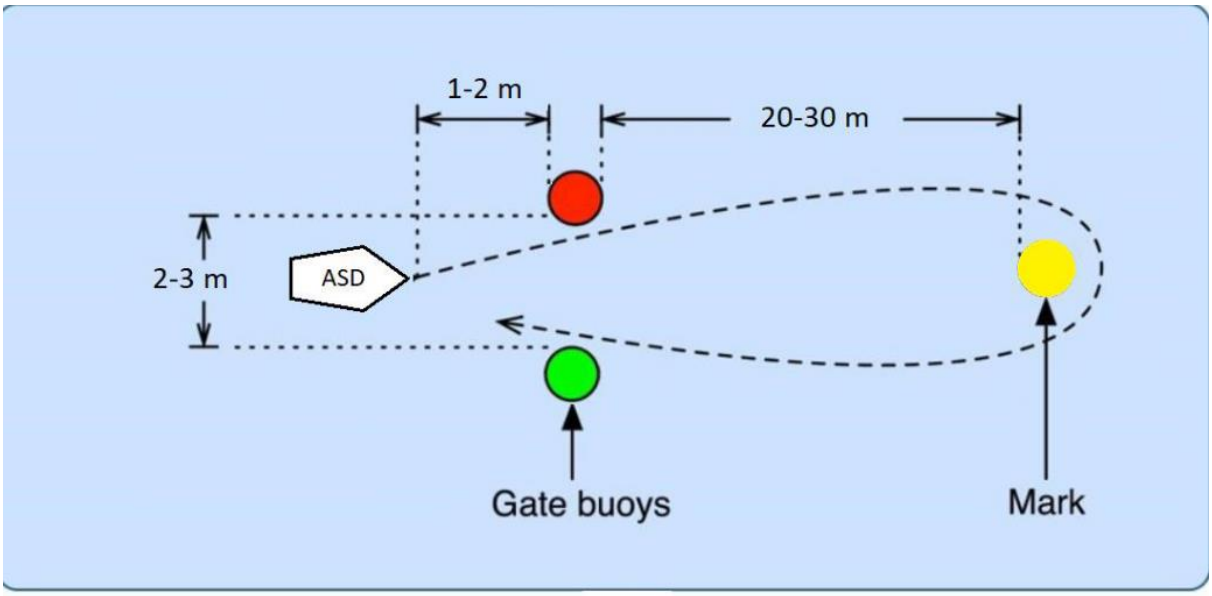
Strategi og algoritme

Alternativ 1:
I denne oppgaven må dronen manøvrere inn til *docking* området og finne en rød og en grønn bøye som den så seiler på langs imellom og inn til kaien. For at dronen skal kunne gjennomføre denne oppgaven må den kunne identifisere den røde og grønne bøyen samt finne selve kaien den skal legge inntil.
Til dette vil dronen bruke *Point-Cloud segmentering* og *Sensor Fusion* mellom kamera og Lidar målinger. Ved hjelp av YOLO algoritmen vil den kunne identifisere form og farge på bøyene og formen av kaien.
Dronen vil så generere *waypoints* og deretter seile mot kaiområdet og ved hjelp av *waypoint tracking* utføre en gjennomførbar bane inn mot *docken*, den vil detektere bøyene og gå sidelengs inn mot kaien mellom de to bøyene.
Enkelt forklart blir strategien som følger - Seile inn mot *docking* området, svinge inn på langs av dokken, stoppe opp ved de to bøyene slik at dronen har lik avstand fra hekken til dronen og bak til bøyen og fra baugen til dronen og frem til bøyen. Deretter skal dronen gå sidelengs inn mot dokken og holde posisjonen i 30 sekunder før den forlater den. Dronen skal så manøvrere mot neste oppgave.

Alternativ 2:
Vi valgte i denne oppgaven å bruke Leite(Leite et al., 2019) for å lage en strategi. Deres tilnærming til autonom *docking*, som er å ha en integrert arkitektur bestående av en *situational awareness modul* og en manøvrerings modul. Ved å behandle data fra Lidar og kamera som blir lagt fusjonert inn i en *occupancy grid* og behandlet som et binært bilde. De geometriske trekkene til *docken* blir så beregnet med data fra kamera brukt for å definere *region of interest (ROI)*. A* algoritmen generer så en rute for operasjonen. Vi vil bruke en *learning based* algoritme som Yolo for å identifisere *docken* ved å projisere sensor informasjonen på en *grid map*. Dess nærmere *docken* dronen kommer, dess mer pålitelig og presis blir dataen som blir mottatt og ved å bruke ICP algoritmen vil dronen kontinuerlig vurdere malene den mottar og beregner ved hjelp av GPS data, UTM koordinater, og tidligere innsamlet kunnskap om

docking området (kai, bøyer) et optimalt docking punkt. Dette blir så overført til manøvrerings modulen som sjekker om målet er gyldig eller usikkert. Den vil så oppdatere seg til den er gyldig etter forhåndsatte parametre. For å navigere dronen mot sitt foretrukne *docking* område blir en kontrollmodul implementert ved å ta retning og avstand til ønsket mål. Denne banen blir diskretisert i små seksjoner og når den er nådd blir den endelige retningen korrigert. Denne modulen er også ansvarlig for beregning av lineære og vinkel hastigheter, og mater dem til dronens *velocity manager*. Disse tar hensyn til fartøyets kinematiske begrensninger og sørger for at maksimale hastigheter respekteres.

4.3.4 Speedgate



Figur 20: Illustrasjon av *speedgate* løypen hvor dronen skal detektere start bøyene, kjøre inn i banen, detektere og runde en markeringsbøye og så kjøre ut av banen gjennom de samme bøyene den ankom.

Scoring	
Time	***Points***
Fastest	80
2nd	50
3rd	20

Tabell 4: Viser antall poeng gitt for de tre raskeste lagene i konkurransen.

Oppdragskrav

En suksessfull gjennomføring av speedgate oppgaven demonstrerer effektiviteten av dronens skrog sammen med fremdriftssystemet, og den resulterende manøvrerbarheten. videre så demonstrerer det evnen til *object detection* og beslutningstaking i henhold til sensing av oppgave elementene

Dronen må ankomme gjennom bøye porten, gå rundt rundingsbøyen (med eller mot klokken) og gå ut den samme porten den ankom så snart som mulig. Port-bøyene er fortøyd 2-3 meter fra hverandre, og rundingsbøyen er plassert 20-30 meter fra port-bøyene.

Dette er en oppgave som går på tid og tiden starter når baugen på dronen krysser porten og stopper når akterenden på dronen krysser porten på vei ut.

Å fullføre oppgaven på mindre en 10 minutt gir 20 poeng, ekstra poeng gis til de tre raskeste dronene.

Strategi og algoritme

For å kunne gjennomføre denne oppgaven må dronen kunne gjenkjenne startbøyene, rundingsbøyen og målpunktet igjen som er det samme som start bøyene.

Denne oppgaven er svært lik som *Obstacle channel* oppgaven, og vi tar også her utgangspunkt i hvordan (*KAIST_RX14_Paper.pdf,* u.å.) løser navigation and control oppgaven. Oppgavene er ganske like bare at man runder en bøye og forlater løypen ut samme port, i stedet for å gå inn en startport og ut en målport. Vi vil bruke samme strategi og algoritme som vi har planlagt i den.

Dronen begynner slik som tidligere ved å forflytte seg fra et startpunkt ved å bruke en gitt GPS posisjon og utfører heading control frem til neste waypoint. Ved hjelp av et *Nodding Lidar System* samles *point cloud data* og en *Point Based Clustering* algoritme som sammen med stereo kamera og Yolo algoritmen som leser fargen på bøyene. *Path Planning* og *Obstacle Avoidance* blir også tatt i bruk i denne oppgaven, og vi bruker også her A* algoritmen.
Siden oppgaven går ut på at den skal utføres på så kort tid som mulig blir *Path Planneren* en viktig del av denne oppgaven. Den leter kontinuerlig etter den raskeste og korteste veien gjennom løypen.

5. Diskusjon og konklusjon

5.1 Diskusjon av metode

Vår oppgave har hovedsakelig vært å lage et designforslag, for en AutoDrone som kan delta i konkurransen AutoDrone 2021.

Fra veilederen vår fikk vi et forslag om å ta utgangspunkt i artikkelen "Team NaviGator AMS - University of Florida: Team NaviGator AMS", og ut fra den gjøre et *"review"*.

Denne artikkelen var (veldig) nyttig for oss, spesielt for at vi skulle bygge opp et fag vokabular for elementene, algoritmene, og prosedyrene som trenges for å løse oppgaver som inngår i AutoDrone 2021. Siden vi er nautikkstudenter, og dermed ikke har hatt ingeniør fag som: *advanced calculus*, lineær algebra, statistikk, algoritmer og datastrukturer, *system engineering* har mye av tiden gått med til å lære det mest grunnleggende innen vokabularet og konseptene innen disse fagfeltene. Neste års studenter vil ikke unngå denne utfordringen. Vi vil likevel basert på vår forståelse foreslå hvordan ideer fra snow-balling og *systematic literature review* kan akselerere inngangen til fagfeltet.

Hvis vi skulle gjort denne oppgaven på nytt ville vi etter å ha lest "Team NaviGator AMS - University of Florida: Team NaviGator AMS" sin artikkel gått gjennom nettsiden https://robotx.org/. Der ville vi funnet fanene *"Programs"* og «*Past programs*".

Basert på artiklene som finnes på denne nettsiden ville vi foreslått en artikkel-lese-liste som består av de tre journal papers som vinner først, andre og tredje pris i konkurransen og av den/de artiklene som vinner beste paper konkurransen i 2014,2016, 2018, 1019. For de oppgavene vi skal løse er 2014 konkurransen mest aktuell:

Vinnerne av 2014 konkurransen:

1st Place ($20,000): Olin College / MIT

TEAM MIT – OLIN RobotX Journal Paper

(*MIT_RX14_Paper.pdf*, u.å.)

2nd Place ($15,000): KAIST

Development of KAIST Grey-Duck USV for 2014 Maritime RobotX Challenge

(*KAIST_RX14_Paper.pdf*, u.å.)

3rd Place ($12,000): Queensland University of Technology
The Endeavour ASV: Hardware Sensor & Software Overview
(Lamont et al., u.å.)

5.2 Valg av algoritmer

Når vi skulle velge *path planner* så fant vi ut at siden konkurransen gikk over et mindre område så ville en *search based* metode være mest gunstig.
Search based methods går ut på å plassere noder ut i et rutenett og gi de en verdi. Start noden har en verdi på null og ved å gi rutene på sidene 10 og de på skrå 14 ville man fylle rutenettet med noder og beregne hvilken sti som har den laveste verdien og være den optimale ruten. A* er en *search based method* algoritme. Fordelen med den er at den ikke velger laveste verdi på hver node, men kan ta en med høyere verdi hvis det endelige resultatet blir lavere. Den trenger heller ikke å legge inn noder i hele rutenettet.
De passer best i mindre områder da det er kostnadskrevende i henhold til datakapasitet.
I større rutenett som kan inneholde millioner ruter er *sampling based methods* mer nyttige. RRT er som et tre med greiner som brer seg ut og dekker hele området. Ved å plassere tilfeldige noder ut i kartet. Det gjør at en gren alltid vil utvikle en vei mot målet. RRT* vil så optimalisere grenen ved å plassere ut noder for å rette ut banen og gjøre distansen mindre.(MATLAB, 2020b)

For *object detection* valgte vi å ofre litt nøyaktighet og å prioritere hastighet. Vi bestemte oss derfor for å bruke YOLO algoritmen fremfor SSD algoritmen i og med at noen av oppdragene i Autodrone konkurransen går på tid.

5.3 Diskusjon fordeler og ulemper med ASV

For å trekke frem fordeler og utfordringer med ASVer gjentar vi de viktigste delene av listen som er satt opp i (Vagale et al., 2021). ASV har potensiale til å prestere bedre enn tradisjonelle fartøy med henhold til sikkerhet. En økt innpass av ASV`er kan føre til reduksjon i ulykker forårsaket av menneskelig svikt. Ulykker forårsaket av menneskelig svikt bidrar som kjent til en stor del av verdens skipsulykker.

Noen av fordelene er:

- Å redusere eller eliminere menneskelig styring og da også menneskelige feil.

- Å gjennomføre lengre og farligere oppdrag som kanskje av sikkerhetsmessige grunner ikke hadde blitt gjort av et bemannet fartøy.

- Forlenge operasjonelle evner, funksjonalitet og presisjon. Dette vil føre til at ASV`er vil bli mer etterspurt på mange felt, for eksempel vitenskapelig forskning, militæroperasjoner, frakt og andre segmenter.

- Redusere personalkostnader og forbedre personell sikkerhet siden det ikke er noe mannskap ombord, og kollisjonsunngåelse basert på kunstig intelligens er implementert

- Redusere fare for piratangrep og for eventuell eliminering og kidnapping av mannskap.

- Øke kapasiteten til å kunne ta mer last om bord i skipene. Dette vil også føre til at ASV`er blir mer etterspurt, spesielt i fraktmarkedet.

ASV`er har ennå flere utfordringer før de kan tas i bruk i nasjonalt farvann verden over. Noen av disse ulempene er:

- **Lovgivning** som regulerer ASV`er er fortsatt uklart.

- Betydelig internasjonalt samarbeid er påkrevd for å lage navigasjon og sikkerhetsforskrifter samt design standarder.

- **Ansvarlighet**. Det er mange juridiske utfordringer som oppstår når det ikke er en kaptein om bord, for eksempel hvem som har ansvaret for handlinger som utføres.

- **Cybersikkerhet** er en stor bekymring for alle autonome systemer. Cybersikkerhet er av avgjørende betydning. En feil i programvaren kan gi uautorisert tilgang til hackere som har som formål å ta over kontrollen av skip.

- **Sikkerhet i navigasjonen**. Et fartøy som seiler i åpent farvann står overfor mange risikoer, inkludert tøffe værforhold, hindringer, spesielt dynamisk eller under vann, eller til og med risiko knyttet til tredjeparter. Spesiell oppmerksomhet bør rettes mot hindringer som ikke kan oppdages av det automatiske identifikasjonssystemet (AIS), for eksempel mennesker i sjøen, mindre fritidsfartøy, små objekter i sjøen eller sjødyr. Et autonomt skip må være i stand til å håndtere slike utfordringer av seg selv uten menneskelig kontroll.

- **Pålitelighet og vedlikehold**. For å operere på sjøen i lengre perioder er det avgjørende å ha systemer for overvåking av skipets tilstand og vedlikeholdsplaner. Hvis det ikke er ingeniører om bord, må det planlagte vedlikeholdet skje i havnen. Dette kan kreve lengre opphold i havnen, og havnekostnader er dyrt. Videre, for å oppnå tilfredsstillende pålitelighet, kan det være nødvendig å redesigne mange av skipssystemene for å forbedre gjennomsnittstiden mellom systemfeil om bord.

- **Tilkobling**. Selv om det er et økende antall satellitter i bane, er det en varierende grad av dekning og båndbredde avhengig av fartøyenes beliggenhet. Områder med høye breddegrader har dårlig dekning og er spesielt utfordrende siden de fleste satellitter er

geostasjonære over ekvator. I tillegg kan et fartøy miste tilkobling på grunn av vær, skade på viktig utstyr (som antenner) og interferens.

- **Piratangrep**. Selv om ASV er ubemannet, har lasten og selve skipet en høy verdi og er gjenstand for kapring. Et ubemannet skip kan også være lettere å angripe.

Den økte populariteten til ASV vises tydelig av at skipsbyggere har forsøkt å innføre autonomi til sjøs, og av suksessen til disse prosjektene. Basert på bransjens spådommer forventes helt autonome overflate fartøy innen 2025. En positiv forsterkning her er de mange samarbeidsprosjektene mellom selskaper som fører til kunnskapsdeling og raskere utvikling av teknologien.

Et annet tema for diskusjon er sikkerhetskonvensjoner som COLREGs. Disse forskriftene, som først ble utviklet i 1972, var tydelig utviklet for menneske kontrollerte bemannede fartøyer. I en tid hvor styringen over fartøy bevisst overføres til datamaskiner, er det åpne spørsmålet hvor godt ASV-er skal følge regler skrevet for mennesker. Videre er det mulig å se for seg eksepsjonelle situasjoner når en ASV ikke skal følge de definerte kollisjons unngåelses reglene for å unngå kollisjon i siste liten. Derfor er det klart at COLREGs regel 2 (b) skal implementeres i GNC-systemet til en ASV. Spørsmålet om når man skal følge forskrifter og når ikke, er et tema for fremtidig forskning. Alternativt bør det utvikles en forbedret versjon av disse forskriftene for å ta hensyn til både USV, ASV og tradisjonelt bemannede overflatefartøy på vann.

For å oppsummere forskning om path plannings algoritmer for ASV-er, finnes det et bredt utvalg av forskjellige metoder for path planning som allerede har blitt brukt for USV- eller ASV-applikasjoner. Videre kan noen av de ikke-anvendte path planning algoritmene som er implementert for kjøretøyer på land som navigerer på ustrukturerte veier eller AUVer, sannsynligvis også tilpasses ASV.

Bibliografi.

https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2663584

About. (u.å.). *OpenCV*. Hentet 3. mars 2021, fra https://opencv.org/about/

Aptiv - Mobility Insider. (2020, mars 3). *What Is Sensor Fusion?* Aptiv. https://www.aptiv.com/en/newsroom/article/what-is-sensor-fusion

Ash, C. M., Bloom, N. D., Brown, J. D., Butka, A. N., Cronin, S. P., Delp, G. C., Goring, R. T., Hockley, C. J., Joswick, Z. R., Lodato, D., Middlebrooks, N. R., Mathews, B. F., Pletz, J. L., Romney, J. S., Schoener, M. A., Schultz, N. C., Thompson, A. M., Thompson, D. J., Wyble, J. K., … Currier, D. P. N. (u.å.). *Design of the Minion Research Platform for the 2016 Maritime RobotX Challenge*. 12.

AutoDrone 2020 Rules and Task Description.pdf. (u.å.). Google Docs. Hentet 3. mars 2021, fra https://drive.google.com/file/d/1znsjKdCcPNmfPysDiKEFOY0PmX390RNk/view?usp=drive_open&usp=embed_facebook

Automatic identification system. (2021). I *Wikipedia*. https://en.wikipedia.org/w/index.php?title=Automatic_identification_system&oldid=1015709608

C++. (2021). I *Wikipedia*. https://en.wikipedia.org/w/index.php?title=C%2B%2B&oldid=1009467233

Chin, T. (2019, februar 26). *Robotic Path Planning: RRT and RRT**. Medium. https://theclassytim.medium.com/robotic-path-planning-rrt-and-rrt-212319121378

Cluster analysis. (2021). I *Wikipedia*. https://en.wikipedia.org/w/index.php?title=Cluster_analysis&oldid=1014086807

Convolutional Neural Networks: The Theory. (u.å.). Bouvet Norge. Hentet 3. mars 2021, fra

https://www.bouvet.no/bouvet-deler/understanding-convolutional-neural-networks-part-1

Extended Kalman filter. (2021). I *Wikipedia*. https://en.wikipedia.org/w/index.php?title=Extended_Kalman_filter&oldid=1004599218

Fiskeridepartementet, N. (2018, juli 20). *Autonome skip* [Innhold]. Regjeringen.no; regjeringen.no. https://www.regjeringen.no/no/tema/naringsliv/maritime-naringer/ny-temaside/forste-kolonne/markedsadgang-og-regelverk/id2589230/

Fossen, T. (2002). Marine control systems: Guidance, navigation and control of ships, rigs and underwater vehicles. I *Journal of Guidance Control and Dynamics—J GUID CONTROL DYNAM* (Bd. 28).

Gazebo. (u.å.). Hentet 3. mars 2021, fra http://gazebosim.org/

Global Positioning System. (2020). I *Wikipedia*. https://nn.wikipedia.org/w/index.php?title=Global_Positioning_System&oldid=3278183

Hebert, M., Thorpe, C., & Stentz, A. (1997). *Intelligent unmanned ground vehicles: Autonomous navigation research at Carnegie Mellon*. Kluwer Academic Publ.

Inertial measurement unit. (2021). I *Wikipedia*. https://en.wikipedia.org/w/index.php?title=Inertial_measurement_unit&oldid=1007636024

Introduction to Tracking Metrics—MATLAB & Simulink—MathWorks Nordic. (u.å.). Hentet 3. mai 2021, fra https://se.mathworks.com/help/fusion/ug/introduction-to-tracking-metrics.html

Iterative closest point. (2021). I *Wikipedia*. https://en.wikipedia.org/w/index.php?title=Iterative_closest_point&oldid=1010395485

Johansen, T. A., Perez, T., & Cristofaro, A. (2016). Ship Collision Avoidance and COLREGS Compliance Using Simulation-Based Control Behavior Selection With Predictive Hazard Assessment. *IEEE Transactions on Intelligent Transportation Systems*,

17(12), 3407–3422. https://doi.org/10.1109/TITS.2016.2551780

KAIST_RX14_Paper.pdf. (u.å.). Hentet 3. mai 2021, fra https://robonation.org/app/uploads/sites/2/2019/09/KAIST_RX14_Paper.pdf

Kalman filter. (2021). I *Wikipedia*. https://en.wikipedia.org/w/index.php?title=Kalman_filter&oldid=1020055616

Kang, M., Kwon, S., Park, J., Kim, T., Han, J., Wang, J., Hong, S., Shim, Y., Yoon, S., Yoo, B., & Kim, J. (2015). Development of USV Autonomy for the 2014 Maritime RobotX Challenge. I *IFAC-PapersOnLine* (Bd. 48, s. 18). https://doi.org/10.1016/j.ifacol.2015.10.251

Kernel (image processing). (2021). I *Wikipedia*. https://en.wikipedia.org/w/index.php?title=Kernel_(image_processing)&oldid=1012990368

Lamont, R., Nicholson, S., Renando, Z., Dirkis, C., Smith, P., Jakes, D., Vanmali, J., Veitch, S., Bang, A. C., & Dunbabin, D. M. (u.å.). *The Endeavour ASV: Hardware, Sensor & Software Overview*. 15.

Leite, P., Silva, R., Matos, A., & Pinto, A. (2019). *An Hierarchical Architecture for Docking Autonomous Surface Vehicles*. https://doi.org/10.1109/ICARSC.2019.8733620

Li, Z., Sun, J., & Oh, S. (2009). Path following for marine surface vessels with rudder and roll constraints: An MPC approach. *2009 American Control Conference*, 3611–3616. https://doi.org/10.1109/ACC.2009.5160302

MATLAB. (2020a, juni 24). *Autonomous Navigation, Part 1: What Is Autonomous Navigation?* https://www.youtube.com/watch?v=Fw8JQ5Q-ZwU&list=PLn8PRpmsu08rLRGrnF-S6TyGrmcA2X7kg&index=1

MATLAB. (2020b, juli 1). *Autonomous Navigation, Part 2: Understanding the Particle Filter*. https://www.youtube.com/watch?v=NrzmH_yerBU&list=PLn8PRpmsu08rLRGrnF-S6TyGrmcA2X7kg&index=2

MATLAB. (2020c, juli 8). *Autonomous Navigation, Part 3: Understanding SLAM Using Pose Graph Optimization*.

https://www.youtube.com/watch?v=saVZtgPyyJQ&list=PLn8PRpmsu08rLRGrnF-S6TyGrmcA2X7kg&index=3

MATLAB. (2020d, juli 15). *Autonomous Navigation, Part 4: Path Planning with A* and RRT*. https://www.youtube.com/watch?v=QR3U1dgc5RE&list=PLn8PRpmsu08rLRGrnF-S6TyGrmcA2X7kg&index=4

MATLAB. (2020e, juli 27). *Autonomous Navigation, Part 5: What Is Extended Object Tracking?* https://www.youtube.com/watch?v=jbfpwpqVDI8&list=PLn8PRpmsu08rLRGrnF-S6TyGrmcA2X7kg&index=5

Ming, X., & Velamala, S. S. (2018). *Maritime Autonomous Surface Vessels: A Review of RobotX Challenge's Works*. 8.

MIT_RX14_Paper.pdf. (u.å.). Hentet 3. mai 2021, fra https://robonation.org/app/uploads/sites/2/2019/09/MIT_RX14_Paper.pdf

Moher, D., Liberati, A., Tetzlaff, J., & Altman, D. G. (2009). Preferred reporting items for systematic reviews and meta-analyses: The PRISMA statement. *BMJ (Clinical Research Ed.)*, *339*, b2535. https://doi.org/10.1136/bmj.b2535

Motion planning. (2021). I *Wikipedia*. https://en.wikipedia.org/w/index.php?title=Motion_planning&oldid=1021190068

Multiple Color Detection in Real-Time using Python-OpenCV. (2020, april 24). *GeeksforGeeks*. https://www.geeksforgeeks.org/multiple-color-detection-in-real-time-using-python-opencv/

Occupancy grid mapping. (2021). I *Wikipedia*. https://en.wikipedia.org/w/index.php?title=Occupancy_grid_mapping&oldid=997884369

Occupancy Grids—MATLAB & Simulink—MathWorks Nordic. (u.å.). Hentet 15. mars 2021, fra https://se.mathworks.com/help/robotics/ug/occupancy-grids.html

Park, J., Kang, M., Kim, T., Kwon, S., Han, J., Wang, J., Yoon, S., Yoo, B., Hong, S., Shim, Y., Park, J., & Kim, J. (2016). Development of an Unmanned Surface Vehicle System

for the 2014 Maritime RobotX Challenge: Development of an Unmanned Surface Vehicle System for the 2014 Maritime RobotX Challenge. *Journal of Field Robotics*, *34*. https://doi.org/10.1002/rob.21659

Pathfinding. (2021). I *Wikipedia*. https://en.wikipedia.org/w/index.php?title=Pathfinding&oldid=1010719335

Python. (2021). I *Wikipedia*. https://no.wikipedia.org/w/index.php?title=Python&oldid=21250308

Radar. (2021). I *Wikipedia*. https://en.wikipedia.org/w/index.php?title=Radar&oldid=1019453377

Random sample consensus. (2020). I *Wikipedia*. https://en.wikipedia.org/w/index.php?title=Random_sample_consensus&oldid=983497063

Region of interest. (2020). I *Wikipedia*. https://en.wikipedia.org/w/index.php?title=Region_of_interest&oldid=962514676

Robot navigation. (2021). I *Wikipedia*. https://en.wikipedia.org/w/index.php?title=Robot_navigation&oldid=1005684102

ROS.org | Powering the world's robots. (u.å.). Hentet 3. mars 2021, fra https://www.ros.org/

Rotational invariance. (2020). I *Wikipedia*. https://en.wikipedia.org/w/index.php?title=Rotational_invariance&oldid=980199096

Scale invariance. (2021). I *Wikipedia*. https://en.wikipedia.org/w/index.php?title=Scale_invariance&oldid=1011978734

Scale-invariant feature transform. (2021). I *Wikipedia*. https://en.wikipedia.org/w/index.php?title=Scale-invariant_feature_transform&oldid=1011461449

Shape Matching using Hu Moments (C++ / Python) | Learn OpenCV. (2018, desember 11). https://learnopencv.com/shape-matching-using-hu-moments-c-python/

Simultaneous localization and mapping. (2021). I *Wikipedia*. https://en.wikipedia.org/w/index.php?title=Simultaneous_localization_and_mapping&o

ldid=1007538558

Skjetne, R., Fossen, T. I., & Kokotović, P. V. (2004). Robust output maneuvering for a class of nonlinear systems. *Automatica*, *40*(3), 373–383. https://doi.org/10.1016/j.automatica.2003.10.010

Stratton, S. J. (2019). Literature Reviews: Methods and Applications. *Prehospital and Disaster Medicine*, *34*(4), 347–349. https://doi.org/10.1017/S1049023X19004588

Tezanos, D. R. C. (2015). *STEREO VISION-BASED PERCEPTION, PATH PLANNING AND NAVIGATION STRATEGIES FOR AUTONOMOUS ROBOTIC EXPLORATION*. 160–162.

Thunes, E. (2019). *Online risk modeling for autonomous ships with experimental results*. https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2622969

Types of Morphological Operations—MATLAB & Simulink—MathWorks Nordic. (u.å.). Hentet 15. mars 2021, fra https://se.mathworks.com/help/images/morphological-dilation-and-erosion.html

Vagale, A., Oucheikh, R., Bye, R. T., Osen, O. L., & Fossen, T. I. (2021). Path planning and collision avoidance for autonomous surface vehicles I: A review. *Journal of Marine Science and Technology*. https://doi.org/10.1007/s00773-020-00787-6

What is GNSS? (2016, mars 1). https://www.gsa.europa.eu/european-gnss/what-gnss

What is LabVIEW? (u.å.). Hentet 3. mars 2021, fra https://www.ni.com/en-no/shop/labview.html

What is Matlab. (u.å.). Hentet 3. mars 2021, fra https://cimss.ssec.wisc.edu/wxwise/class/aos340/spr00/whatismatlab.htm

Wohlin, C. (2014). Guidelines for snowballing in systematic literature studies and a replication in software engineering. *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering - EASE '14*, 1–10. https://doi.org/10.1145/2601248.2601268

Yıldırım, S. (2020, april 22). *DBSCAN Clustering—Explained*. Medium. https://towardsdatascience.com/dbscan-clustering-explained-97556a2ad556

YOLO Algorithm and YOLO Object Detection: An Introduction. (2020, mai 22). Appsilon | End- To- End Data Science Solutions. https://appsilon.com/object-detection-yolo-algorithm/

Øksne, Y. H., Aanning, M., Odden, S., & Holt, M. (2020). *Graph-based slam and navigation in a simulated environment and buoy classification using YOLO for an Autonomous Surface Vessel*. https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2663584