UNIVERSITETET I AGDER

Forside

IS-304: 2022

Tittel: Pålitelighetsmål av bygnings-deteksjoner i KartAi

Emnekode	IS-304 / vår 2022
Emnenavn	Bacheloroppgave i informasjonssystemer
Emneansvarlig	Hallgeir Nilsen
Veileder	Lucia Castro Herrera
Oppdragsgiver	Norkart

Studenter:

Etternavn	Fornavn
Birkeland	Johannes
Hjelleseth	Thomas Låg
Svagård	Eirik Thilesen
Vagle	Bendix Melkeraaen

Jeg/vi bekrefter at vi ikke siterer eller på annen måte bruker andres arbeider uten at dette er oppgitt, og at alle referanser er oppgitt i litteraturlisten.	**JA**	NEI
Kan besvarelsen brukes til undervisningsformål?	**JA**	NEI
Vi bekrefter at alle i gruppa har bidratt til besvarelsen	**JA**	NEI

Forord

Denne rapporten er skrevet som en del av bachelorprosjektet som ble gjennomført vårsemesteret 2022. Den fungerer både som en rapport for UiA hvor det rapporteres om gjennomføringen av prosjektet og som en rapport for Norkart hvor det er dokumentert funn gjennom eksperimentering. Vi i 4Bit har lært utrolig mye dette semesteret: om fag, teknologi, arbeid og oss selv som gruppe. Etter et suksessfullt gjennomført prosjekt har vi noen personer å takke.

Først og fremst en takk til veileder og produkteier Alexander Salveson Nossum, som tok interesse i oss som gruppe. Han har gjennom semesteret vært aktiv og engasjert i vår fremdrift og gitt oss gode tilbakemeldinger, inspirasjon og motivasjon. Andre takk går til vår tekniske veileder, Ivar Oveland hos Kartverket, som alltid hatt tid til å ta et møte når vi har sittet fast, eller for å diskutere problemer og gi forslag. Tusen takk for rask respons og deltakelse på våre sprint-reviews.

En takk må også rettes til Lucia Herrera som har kommet med gode tips til rapporten, deltatt på gruppestyringsmøter og generelt sett hatt en stor interesse for å hjelpe oss til suksess. Og en siste takk til Hallgeir Nilsen som har stått på for å hjelpe alle med forberedelse til bachelorprosjektet, og alle andre forelesere vi har lært av gjennom årene.

Kristiansand, Universitetet i Agder. 16. mai 2022.

Birkeland, Johannes

Hjelleseth, Thomas Låg

Svagård, Eirik Thilesen

Vagle, Bendix Melkeraaen

Sammendrag

Denne rapporten er skrevet og utformet av gruppen 4Bit, våren 2022, som dokumentasjon på fullført bachelorprosjekt i emnet IS-304, samt bachelorstudiet vårt. Rapporten er også skrevet for Norkart. Oppgaven ble utformet sammen med produkteier i starten av semesteret og går ut på å skape et pålitelighetsmål for et prosjekt Norkart deltar på «KartAi» som skal automatisk finne bygg som mangler i den kommunale registeret basert på flere forskjellige observasjonskilder. Pålitelighetsmålet er ment å være en del av en større prosess kalt «endringsanalyse». Her er pålitelighetsmålets jobb å kunne bekrefte om en observasjon er pålitelig nok til at en innbygger kan kontaktes angående endringer på deres tomt.

Oppgaven har vært et forskningsprosjekt hvor eksperimentering og research er blitt vektlagt for å prøve å finne en måte å måle pålitelighet. Vi har under hele prosjektet vært frie til å velge verktøy selv. Gjennom semesteret har vi vært gjennom flere teknologier og teknikker, her blant annet PostGIS, diverse python-biblioteker, BIoU, GIoU og mer.

Som arbeidsmetodikk startet vi med tradisjonell scrum. Da vi kom lengre inn i prosjektet innså vi at dette passet dårlig med vår type oppgave. Vi måtte ofte sitte å gjøre research frem til vi fant noe nyttig, og så begynne å arbeide med det. Da fungerer det dårlig med statiske arbeidsoppgaver. Vi valgte å bytte til en sammensetning av scrum og kanban som lot oss legge til arbeidsoppgaver ved behov midt i sprinter. Arbeidsflyten fungerte langt bedre med denne tilpassede arbeidsmetodikken.

Vårt resultat er ikke et ferdig produkt, og er ment som et utgangspunkt for videre utvikling av pålitelighetsmål. Utgangspunktet er definert i kapittel 4 «Eksperimentering» pluss et tilhørende git-repository (https://github.com/4bit-UiA/Bachelor-2022-KartAi). Det kapittelet tar for seg alt av teknologier, ideer og algoritmer som vi har vært innom, og det er hva som er av interesse for Norkart. Arbeidet vårt gjør det enklere for et nytt team å sette seg inn i hvordan pålitelighet kan bli målt og hvilke sammenligningsmetoder som er gunstige. Det neste steget blir å teste algoritmene våre på reell data fra AI-en, og begynne å finjustere konfigurasjonene tilsvarende.

Innholdsfortegnelse

Figurliste

Tabelliste

1. Innledning

I bachelor-emnet IS-304 er oppgaven å kjøre et IT relatert prosjekt selvstendig i et semester, ved å ta i bruk etablerte metoder og teknikker for planlegging, implementering, estimering og dokumentering av prosess og produkt. Målet er å kunne bruke lærdom fra tidligere emner i bacheloren til å utføre et reelt prosjekt for en ekte bedrift. Prosjektet gjøres med bedriften Norkart som produkteier, som gruppen kom i kontakt med gjennom deres presentasjoner på universitetet. I tillegg var et av medlemmene i gruppen i praksis hos Norkart semesteret før, så valget falt ganske naturlig på dem. Vårt prosjekt var å finne en måte å skape et pålitelighetsmål for prosjektet KartAi, som Norkart deltar på, som har som mål å automatisk detektere uregistrerte bygg fra forskjellige datakilder (flyfoto, lasermålinger, og flere).

I kapittel 1 vil vi presentere bakgrunnen for vårt prosjekt. Her kommer vi inn på hvem vår produkteier er, en introduksjon til KartAi og hva vårt prosjekt omhandler. Kapittel 2 går over sentrale valg som er blitt tatt, kapittel 3 fortsetter med beskrivelse av gjennomføringen av prosjektet, kapittel 4 tar for seg funn gjennom eksperimentering og til slutt tar kapittel 5 opp våre tanker og refleksjoner rundt valg og arbeid gjennom semesteret.

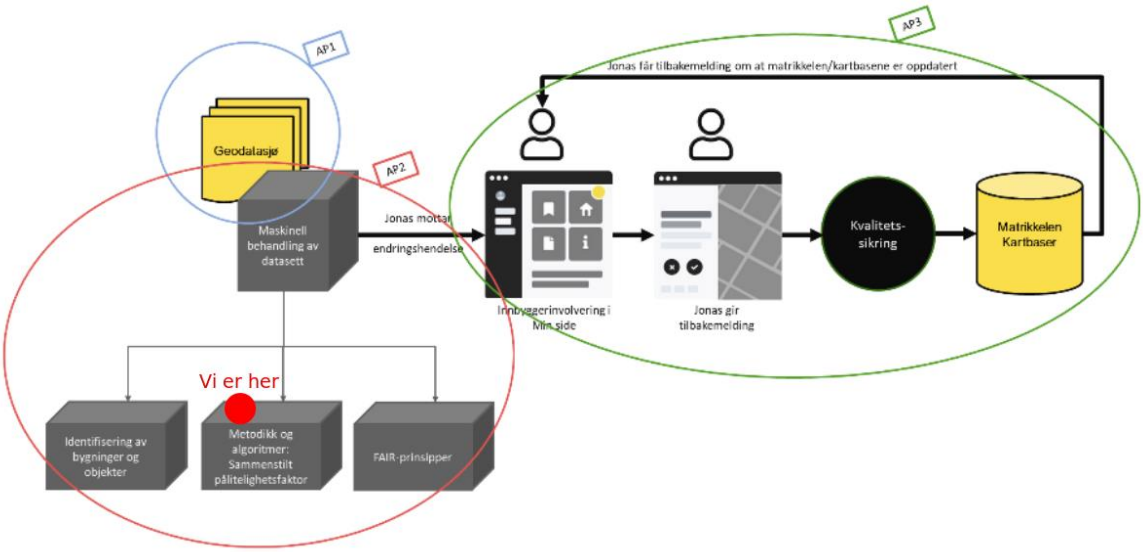
1.1 Om Norkart

Norkart er en norsk IT-bedrift som leverer flere forskjellige geografiske informasjonssystemer (GIS) til et bredt kundespekter. Bedriften har flere privatkunder, men er likevel kanskje mest anerkjent for sine mye brukte kommunale tjenester hvor blant annet Kristiansand Kommune er en av brukerne. Norkart har røtter helt tilbake til 1961 og har i dag 190 medarbeidere fordelt på fem kontorer i Trondheim, Sandvika, Lillehammer, Bergen og Kristiansand (Norkart, u.å.).

1.2 KartAi-prosjektet

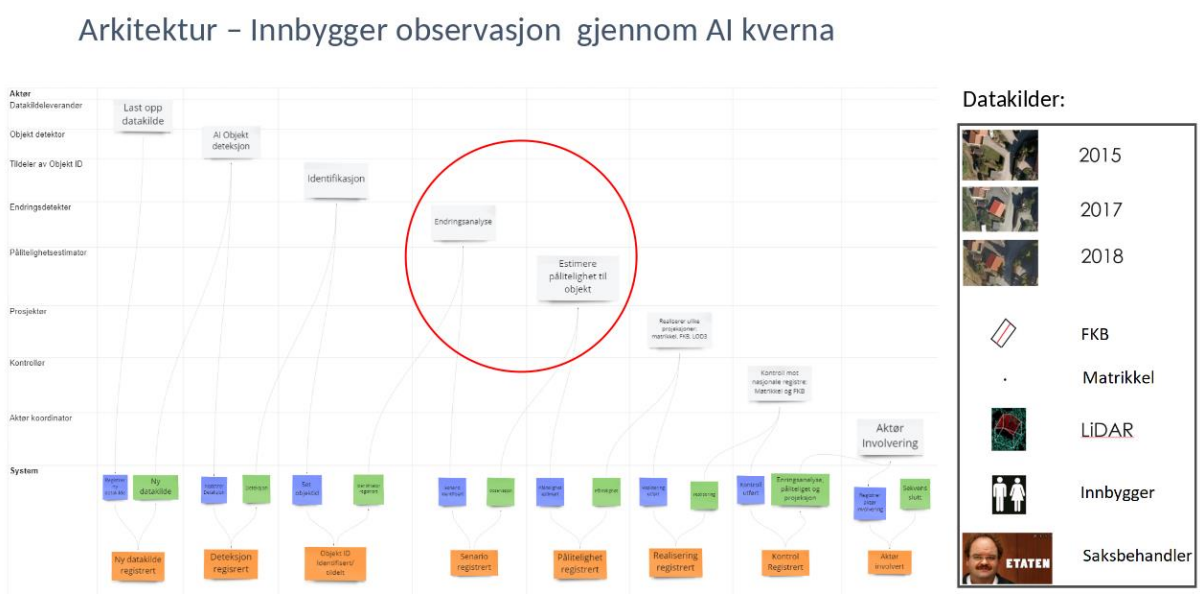
KartAi er et forskningsprosjekt som har som hovedmål å effektivisere saksbehandlingen i byggesaker samt heve kvaliteten på det kommunale eiendomsregisteret (matrikkelen) og Sentral felles kartdatabase (FKB) i Kristiansand, ved bruk av kunstig intelligens. Prosjektet hadde oppstart i 2021 og er estimert til å vare frem til 2023 og er et samarbeid mellom

Norkart, Kartverket og Universitet i Agder med Kristiansand Kommune som produkteier. For å nå disse målene utvikles en AI som baserer seg på en geodatasjø som består av data fra flere forskjellige datakilder for å identifisere bygg som avviker fra informasjonen i matrikkelen. Deretter kontaktes den aktuelle innbyggeren som eier bygget for å gi en bekreftelse på avviket som AI-en har identifisert. Denne bekreftelsen blir da en del av geodatasjøen og brukes igjen for å forsterke påliteligheten på AI-ens beslutning.



Figur 1-1 - Bearbeidet fra «Oversikt over arbeidspakkene i KartAi», av Peterson et al., 2021.

For å strukturere arbeidet, er KartAi delt inn i tre hoveddeler, eller såkalte arbeidspakker (AP). Disse er arbeidspakke 1 (AP1) som jobber med utformingen av dataen i geodatasjøen, arbeidspakke 2 (AP2) som jobber med den kunstige intelligensen (AI) og arbeidspakke 3 som tar seg av innbyggerinvolvering basert på AI-ens resultater. Dette er vist i Figur 1.1. Vårt prosjekt finner sted i AP2 og tar for seg delen etter at AI-en har produsert resultatene.



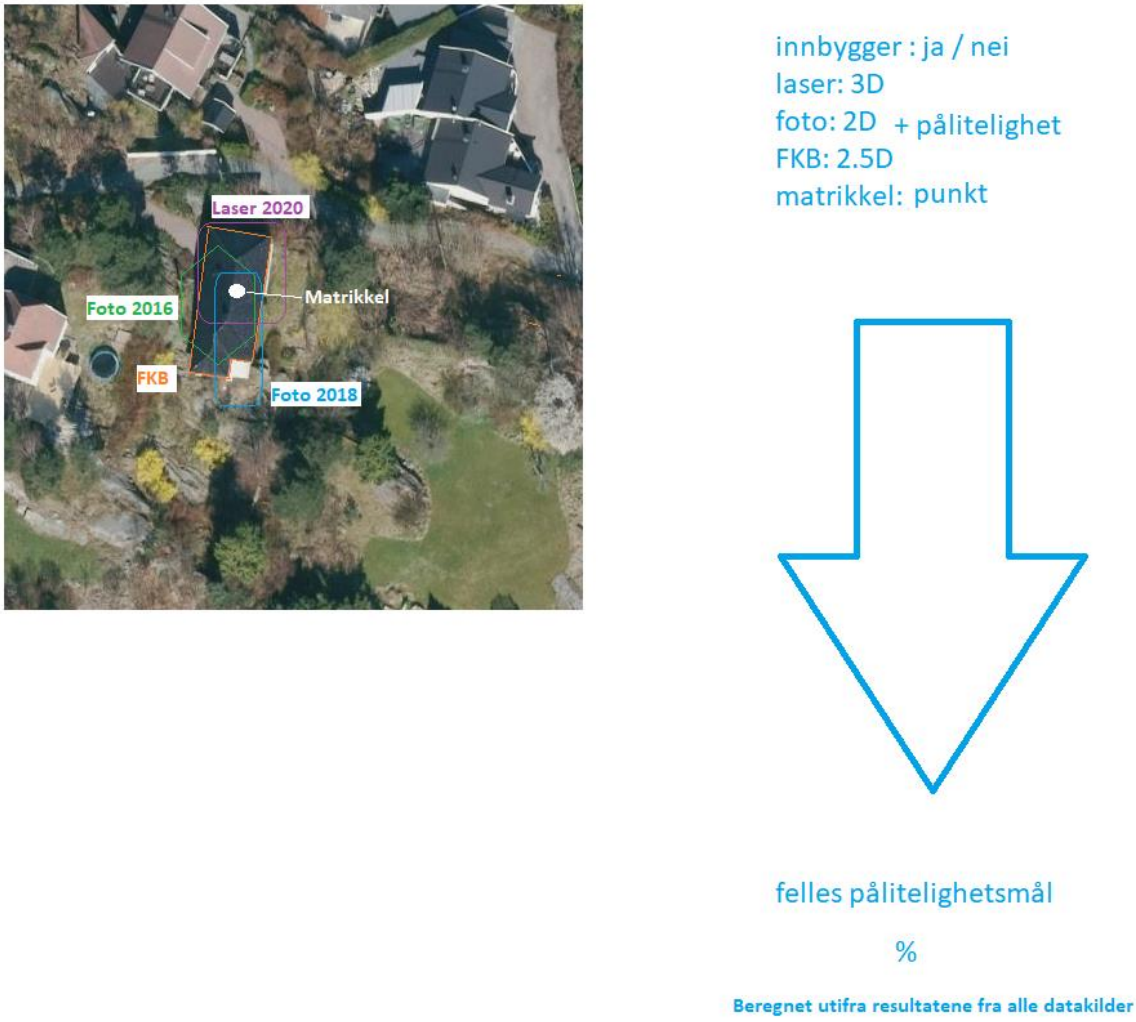
Figur 1-2 - Overordnet plassering i arbeidspakken. Av Oveland, I. (2022). Privat kommunikasjon.

Figur 1.2 viser hvor vi har arbeidet i forhold til de andre delene som er planlagt i AP2 i en mer detaljert modell. Disse modellene er vårt utgangspunkt for prosjektet i tillegg til oppgavebeskrivelsen fra Norkart som er formulert slik:

> Utvikle algoritme(r) og metodikk for pålitelighetsmål på resultater.

1.2.1 *Visjon*

Med en slik åpen beskrivelse er det mange retninger å gå for bachelorprosjektet og resultatet varierer sterkt i forhold til hvordan gruppen velger å løse oppgaven. Gjennom grundigere samtaler med våre samarbeidspartnere i Norkart og Kartverket ble det klart at dette er et forskningsprosjekt hvor målet er å «prøve og feile» med flere løsninger for et slikt pålitelighetsmål. I Figur 1.3 er det en oversikt over hvordan Norkart presenterte hvordan sammenligningen av de ulike datasettene tenkes å være. Om vi ønsker å utvikle faktiske fungerende algoritmer eller om vi hovedsakelig vil lage teoretiske modeller og drive forskning på området blir veldig opp til gruppen selv. Selve produktet er altså ikke klart definert, men heller det faglige området hvor det ønskes resultater.



Figur 1-3 - «Endringsanalyse scenario». Av Oveland, I. (2022). Privat kommunikasjon.

1.3 Vårt prosjekt

Etter videre samtaler med produkteier og analyse av den tildelte oppgaven, ble to problemstillinger utformet for denne rapporten. Den første problemstillingen handler om kjernen i oppgavebeskrivelsen nemlig hvordan finne et pålitelighetsmål og er derfor veldig åpenbar.

1. **Hvordan kan vi få pålitelighet til observasjoner fra KartAi-algoritmer basert på ulike datasett og ulike observasjonsalgoritmer?**

2 Sentrale vedtak

I dette kapittelet vil vi ta for oss sentrale valg og begrunnelser. I starten tar kapittelet for seg valg av metodikk og hvordan vi har valgt å gjennomføre prosjektet, videre går det inn på teknologier og til slutt ender det på hvilke ytterlige beslutninger vi har tatt for å gjennomføre prosjektet og sikre kvalitet. Målet for kapittelet er å gå gjennom alle sentrale valg som har vært med på å kvalitetssikre prosjektet og prosjektgjennomføringen. I tillegg har vi formulert begrep og ord som gjentar seg gjennom rapporten for å avklare hva det omtales om.

2.1 Krav fra produkteier

I det første møte med produkteier gikk vi gjennom og formulerte oppgaven vi skulle utføre gjennom semesteret. Her ble det satt forventninger og noen krav. Norkart forventer at ved slutten av bacheloroppgaven så vil de få en ferdig rapport og tilgang til alle kodebaser (Git repository) som er blitt brukt i prosjektet. Det er avklart at den ferdige rapporten kan være en sammenslåing med rapporten som skal leveres til UiA. Utover dette har Norkart ingen krav til teknologi som skal brukes under eksperimentering, og ser på det som en oppgave for oss prosjektutførere å finne det som passer best. Valg av teknologier beskrives i delkapittel 2.3.

2.2 Metodikk og prosjektstyringsvalg

Dette kapitlet vil ta for seg hvilke metoder som er blitt brukt for å gjennomføre dette prosjektet. Grunnlag for valg av metode vil bli presisert samt hvordan denne metodebruken har påvirket de forskjellige prosessene gjennom prosjektet.

2.2.1 *Den brukte metodikken*

Metodikken brukt for alt arbeidet i dette prosjektet er en blanding av de to populære agile rammeverkene Scrum og Kanban. Fra Scrum er sprint-strukturen brukt for å sørge for en jevnlig levering samt gi en praktisk oversikt over fullført og planlagt arbeid. I tillegg har teamet hatt daglige «standup-møter» klokken 12 gjennom hele prosjektet hvor aktiviteten «Daily Scrum» holdes. Her går hvert medlem gjennom hva de har gjort siden sist møte og hva som er planlagt fullført til det neste. Fra starten av prosjektet har det også blitt satt opp jevnlige statusmøter med vår representant fra Norkart. På disse møtene deltok av og til en

representant fra samarbeidsorganet Kartverket, for å sørge for grundigere tilbakemeldinger fra flere fronter. I tillegg til dette har det blitt holdt styringsgruppemøter arrangert av studentene hvor alle førnevnte er til stede i tillegg til veileder fra UiA. Alle disse møtene er blitt grundig dokumentert slik at prosessen fra start til slutt i hele prosjektet er tilgjengelig og kan sees i Appendix C og D. Gjennom studieløpet har UiA arrangert møter for å utveksle erfaringer med de andre bachelorgruppene på vårt kull og foreleseren i emnet. Disse møtene har gitt teamet gode erfaringer og innspill og har hjulpet arbeidsprosessen framover. Utveksling av erfaringer med andre i samme situasjon har vært til stor hjelp for refleksjon rundt teamets arbeid i dette prosjektet. Teamet har også vært nøye med å holde Sprint Planning og Sprint Retrospektiv for hver av de ni sprintene som har blitt gjennomført. Utover dette er de fleste andre elementer av Scrum blitt utelatt til fordel for de mer fleksible Kanban-metodene, da spesielt den klassiske «Kanban-veggen» (Agile Alliance, u.å.).

Verktøyene som er brukt for den agile prosjektgjennomføringen har vært de samme gjennom hele prosjektet. For å holde oversikt over oppgaver og ha versjonskontroll er Azure DevOps blitt brukt. Dette verktøyet ble valgt nettopp fordi det er så fleksibelt i hva det kan tilby. Både versjonskontroll og oppgaveoversikt gjennom en Kanban-vegg gjør at vi som gruppe har alt på ett sted. I tillegg lar Azure oss koble pushing av kode direkte til enkelte oppgaver, noe som øker gjennomsiktigheten og oversikten over arbeidet. Timeføring var noe upraktisk å gjøre gjennom dette verktøyet så teamet har brukt et regneark til å oppfylle dette formålet (Figur 2.1). Dette arket tar for seg hver sprint og regner automatisk ut timebruk per medlem per sprint. Til slutt for skriving og lagring av dokumentasjon, rapportskriving og generelle notater er Google Docs og Google Drive blitt brukt.

SPRINT 6 (24.mar - 6.mar)										
Thu 24. Mar	1	møte, planning	2.50	Arbeid med algoritme (mest oppsett)	1	Sprint møte og planning	2.5	algoritme		
Fri 25. Mar	3	Arbeid kartverket	3.00	Hos kartverket (-1 time for annet arbei	3	Hos kartverket, påbegynt algoritme	3	kartverket		
Sat 26. Mar		helg		Helg		helg		helg		
Sun 27. Mar		helg		Helg		helg		helg		
Mon 28. Mar	1	Mølte, statusarbeid	1.00	Daglig møte + statusmøtevideo	3	Arbeid med algoritme	1	daily + video		
Tue 29. Mar	3	Møte, rapportskriving	4.50	Daglige møte, algoritme arbeid, videre	3	Arbeid med algoritme + møte	3	daily + algoritme		
Wed 30. Mar	3	Møte, rapportskriving	0.50	Daglig møte	0.5	Fokus på andre ting. Daglig møte				
Thu 31. Mar		syk	3.00	Daglig møte + algoritme + dokumenter	1.5	Dokumentasjon på algoritme + Daglig	3	daily + algoritme + rapport		
Fri 1. Apr		syk	3.50	Møte med veileder + algoritme-arbeid	2	Doumentasjon på algoritme + Leting e	3.50	Møte med veileder + algoritme-arbeid		
Sat 2. Apr		syk		Helg		helg				
Sun 3. Apr		syk		Helg		helg				
Mon 4. Apr	2	syk (møte, rapport)	0.25	Møte og algoritme	2.5	Skrevet på utgangspunkt pålitelighetsf	0.25	daily		
Tue 5. Apr	1	møte	2.00	Møte og algoritme	2	Strukturering av rapport	2	møte og algoritme		
Wed 6. Apr	2.5	Møte, tilbakemelding fra lucia	3.00	Møte og algoritme	2	Møte og strukturering av rapport.	2.5	møte og algoritme		
Sprint 6:		**16**		**23.25**		**20.5**		**20.75**	**Team totalt:**	**80.50 h**

Figur 2-1 - Utdrag fra regneark over timeføring

2.2.2 *Grunnlag for valg av metodikk*

Valg av metode for dette prosjektet har vist seg å være en mer krevende oppgave enn forventet. Dette er hovedsakelig en konsekvens av at prosjektoppgaven har større fokus på forskning og undersøkelse, hvor et mer ordinært utviklingsprosjekt typisk ville rettet søkelyset mot iterative leveranser av et klart definert produkt. Et slikt produkt er hensiktsmessig ikke blitt definert fra oppdragsgiver ettersom meningen med prosjektet er å eksperimentere med og undersøke forskjellige muligheter for løsning på et problem. Med et slikt prosjekt ble det klart at en Scrum-prosess «rett fra boka» ikke ville være ideell. Mye fokus på eksperimentering betyr at teamet har behov for å fleksibelt kunne modifisere sine oppgaver basert på resultatene fra disse eksperimentene. Metodikken ble endret utover i prosjektet etter hvert som teamets erfaring med den mer tradisjonelle Scrum-prosessen vi er vant med passet dårlig inn i prosjektet vårt. Behovet for fleksibilitet gjorde det utfordrende å følge Scrum-metodikkens strenge regler for fast definerte sprinter. Gjennom de ulike sprintene ble vi nødt til å legge til nye oppgaver og følge andre tråder etter hvert som eksperimenteringen gikk fremover. Teamet hadde av den grunn et møte hvor det ble besluttet å forlate disse prinsippene i Scrum og legge til rette for mer dynamisk oppgavegenerering.

Starten av prosjektet var preget av denne endringen i metodikk og gjorde arbeidet mye mindre oversiktlig. I starten hadde vært enkelt medlem en egen oppgave kalt «eksperimentering» hvor alt individuelt arbeid skulle legges inn. Vår erfaring her ble at de eksperimenterings-oppgavene ble for generelle når de måtte defineres i Sprint Planning, og ble stående uendret gjennom sprinten. Resultatet ble at arbeidet ble gjort, men den generelle oversikten over hva slags arbeid hvert enkelt teammedlem holdt på med ble uklart. Ettersom teamet består av kun fire stykker som har daglige stand-up-møter gikk det fint å holde oversikten mens arbeidet foregikk, men dokumentasjonen av arbeidet ble redusert som et resultat av disse generelle oppgavene. Overgangen var naturlig for teamet og har hatt en tydelig positiv effekt på mengden og kvaliteten av dokumentasjonen

2.2.3 *Backlog og estimering*

Backloggen er som nevnt blitt lagt inn i Azure DevOps, hvor teamet har kunnet opprette relevante oppgaver og holdt oversikt over hvilke som er aktive og ikke. Estimeringen av arbeid har vært en utfordring gjennom hele prosjektet, ettersom en aldri vet akkurat hvor lang

tid et eksperiment vil ta. Dette var et problem som bedret seg mye gjennom prosjektet etter hvert som teamet ble mer komfortabel med måten å arbeide på. Mot midten av prosjektet med selve utviklingen av algoritmene opplevde teamet at estimeringen av arbeidet gikk bedre og oppgaver ble fullført i de sprintene de var planlagt til. Dette var ikke alltid tilfellet i begynnelsen og teamet ble nødt til å videreføre enkelte oppgaver til neste sprint på grunn av underestimering av tid.

Et annet punkt som var utfordrende, var å estimere tid brukt på kompetansebygging hos medlemmene. I starten hadde teamet mye fokus på å tillære seg teori rundt AI, geografiske programmer og koding i Python som gjorde estimering vanskelig. En slik prosess er både tidkrevende og vanskelig å holde oversikten over ettersom framgangen var ganske individuell. Men mot slutten ble oppgavene mer spesifikke etter hvert som prosjektet kom lengre og tidsestimering ble mer nøyaktig.

2.2.4 *Roller*

Gruppemedlem	**Rolle**
Bendix	Kommunikasjonsansvarlig
Johannes	Scrum Master
Eirik	Utvikler
Thomas	Utvikler

Tabell 2-1 - Roller i prosjektet

Kompetansen i vårt team har stor variasjon og forskjellige roller har vært svært naturlig å etablere som kan sees i Tabell 2.1. Den mest åpenbare rollen er «Scrum Master» som Johannes har fylt gjennom hele prosjektet. Denne rollen har gått ut på å holde Daily Scrum-møtene gående, dokumentere dem og holde oversikten over backloggen. Ettersom timeføring er såpass naturlig knyttet til backloggen falt ansvaret for det også raskt på ham. I tillegg har han også holdt Sprint Planning og Sprint Retrospektivene og dokumentert dem. Ettersom Bendix var i praksis hos Norkart semesteret før og allerede kjente godt til produkteier, har han vært kommunikasjonsansvarlig. Gjennom praksisen fikk Bendix tilgang og erfaring med Teams-området til KartAi samt flere mailadresser til viktige deltakere i prosjektet, så dette var en rolle som falt veldig naturlig. Ansvarsoppgavene ble å holde styr på kalenderen, booke

møter med relevante parter, svare på e-poster som omhandler prosjektet og holde flere av møtene. Thomas og Eirik ble mot midten av prosjektet satt til de mer tekniske oppgavene med algoritmeutviklingen. Deres ansvar ble å utvikle Python-script basert på algoritmemodellene som teamet laget i fellesskap. Disse rollene har dog ikke vært satt i stein og alle medlemmene har hatt innspill og samarbeidet med alle oppgavene.

2.3 Teknologier

Ettersom prosjektet ikke har dreiet seg om å utvikle et system, har vi stått ganske fritt til å velge teknologi. Vår endelige kodebase er et biprodukt ved siden av den endelige rapporten for Norkart, og den vil heller ikke være veldig stor. Valgene vi har tatt fra starten av har ledet til at vi eksperimenterte og jobbet med geometrier. Flere av teknologiene som er brukt gjennom prosjektet er funnet gjennom eksperimentering, og noen av disse har blitt satt som standarder underveis og blir derfor inkludert i dette kapittelet.

2.3.1 *Programmeringsspråk - Python*

Både fra produkteiers anbefalinger og våre egne tanker endte vi opp med å velge Python som programmeringsspråk. Python er et kjent programmeringsspråk som kan brukes til det aller meste. Med enkel syntaks og god dokumentasjon er det enkelt å lære og jobbe med. Maskinlæringsmodellene som blir brukt i KartAi utvikles også i Python og det er derfor veldig passende og praktisk å velge Python (Coursera, 2022) (JetBrains, u.å.). I tillegg er det en rekke biblioteker for Python som har hjulpet i eksperimenteringsarbeidet. Noen gode eksempler på dette er Shapely-biblioteket som brukes for å jobbe med geometrier og koordinater (Shapley, u.å.), GeoPandas som brukes for å jobbe med geografisk data (GeoPandas, u.å.) og Mathplotlib som er et visualiseringsverktøy (Mathplotlib, u.å.).

2.3.2 *Database*

Fra starten av var det ikke satt et valg for database-server. Vi var ikke engang sikre på om det var behov for det i det hele tatt, ettersom vi i begynnelsen lagret mockdata i json-filer. Men etter hvert som vi eksperimenterte og så etter mulige verktøy, fant vi PostGIS. PostGIS er en utvidelse for databaseserveren PostgreSQL som gjør det mulig å lagre geometrier i flere dimensjoner. Men mer interessant, har et stort utvalg av funksjoner for å jobbe med

geometrier. PostgreSQL ble deretter brukt primært som en plass for å lagre data, men også for å utnytte funksjonene til PostGIS (PostGIS, 2022).

2.3.3 *GIS*

Geografiske informasjonssystem (GIS) er en programvare for innhenting, lagring, analyse og presentasjon av geografisk data (Ørstavik & Mæhlum, 2020). Gjennom prosjektet har vi jobbet med QGIS som er en «Open Source» GIS-programvare. Grunnen for dette er at QGIS har vært det mest tilgjengelige og fungerer på tvers av plattformer. QGIS har ikke vært i bruk i en vesentlig stor del av prosjektperioden, men viste seg etter hvert å være veldig nyttig til å både visualisere data og til å lage mockdata.

2.4 Kvalitet og kvalitetssikring

Basert på sentrale avgjørelser er målet å skape et rammeverk rundt prosjektet hvor det er satte regler og prosesser for arbeidsmetodikk, kode og kommunikasjon. Sentrale avgjørelser er tatt for at hele teamet skal ha en felles forståelse for hvordan prosjektet gjennomføres. Dette vil være med på å sikre at både prosjektgjennomføring og produktet er av høyest mulig kvalitet. De øvre kapitlene har tatt for seg emnene arbeidsmetodikk og teknologier, som er svært sentrale i gjennomføringen av et IT-prosjekt. Vi vil nå videre komme inn på og forklare ytterligere valg som har vært med på å sikre kvalitet.

2.4.1 *Kommunikasjon og samarbeid*

Kommunikasjon er i ytterste grad kritisk for å gjennomføre et suksessfullt IT-prosjekt. Dette gjelder innad i gruppen, med produkteier og med veileder. Dårlig kommunikasjon innad i gruppen kan lede til misforståelser, unødvendige diskusjoner, dårlig stemning og generelt kaos som vil påvirke sluttresultatet. Uten god kommunikasjon mellom gruppe og produkteier vil en ofte kunne ende opp med at produktet devierer fra hva som er produkteiers forventninger. Et kjempeflott produkt kan fortsatt være ubrukelig for bedriften om det ikke oppfyller forventningene som er satt (Microsoft, u.å.).

Discord

Gruppens kommunikasjonskanal har vært gjennom plattformen Discord. I Discord kan det opprettes flere kanaler basert på hva som diskuteres i dem, og på den måten er det enkelt for gruppen å skille mellom diskusjoner om fag og fritid. For bachelorprosjektet har vi hatt en egen tekst-kanal og tale-kanal. Tekst-kanalen er primært blitt brukt til planlegging, spørsmål og opplysning. Tale-kanalen har blitt brukt gjennom semesteret for de aller fleste standup-møter, men også for parprogrammering og annet prosjektarbeid som er blitt gjort i fellesskap. Her er det også mulig å dele skjerm, som har hjulpet ved sprint planning og andre møter. Alle gruppens medlemmer har vært medlem i samme discord-server i flere semestre og er derfor godt kjent med verktøyet.

Microsoft Teams

Vår primære kommunikasjonskanal med produkteier har vært over Teams. I tillegg til dette er mail blitt mye brukt til planlegging av Teams-møter. Teams er den foretrukne kommunikasjonskanalen i KartAi-prosjektet, og vi ble derfor invitert i starten av prosjektet til KartAi sin kanal. Vi har generelt brukt Teams til sprint reviewer, intervju, styringsgruppemøter og spørsmål. Teams har mange av funksjonene Discord har, men ettersom gruppen allerede er godt vant med Discord, forble Teams bare en måte å kommunisere med produkteier.

2.4.2 *Kodestandarder*

For å forsikre lesbarhet i kodebasen, er det essensielt å følge «best practices» for å holde strukturen i koden fin og enkel å lese. Et tiltak vi tidlig tok her var å gjør det til et å krav å installere programtillegget SonarLint på programvaren (IDE-en) vi brukte for å skrive koden. SonarLint er en plugin for flere slike IDE-er som hjelper med å fikse dårlig kode, bugs, sikkerhetsproblemer og lignende. Underveis mens en skriver kode, gir SonarLint tilbakemeldinger på ting som kan endres. Tilbakemeldingene er basert på noen forhåndsbestemte regler avhengig av hvilket språk det programmeres i. På nettsiden rules.sonarsource.com kan en se alle reglene SonarLint bruker for å rette Python kode (SonarLint, u.å.).

Den endelige kodebasen i dette prosjektet er liten i forhold til rene programmeringsprosjekter, og den består av Python-scripts som stort sett er isolert fra andre scripts og kjører for seg selv.

Derfor har vi valgt å ikke kjøre et fullt ut objektorientert system, men vi har tatt et par prinsipper fra objektorientert programmering ved å lage interne funksjoner og klasser. Slik fjernet vi mye kodeduplikasjon for at scriptene skulle bli lettere å lese. Database-tilkoblingen ble også bygd som en klasse for å abstrahere bort dette laget og gjøre scriptene mindre komplekse.

2.4.3 *Versjonskontroll*

Under eksperimentering i et team med flere medlemmer er det viktig å ha kontroll over script som utvikles. Det vil også være viktig at kodebasen til enhver tid er oppdatert for hvert medlem. Derfor er det avgjørende å bruke versjonskontroll-verktøy. Vi valgte å benytte oss av Git i Azure Repos. Ved bruk av versjonskontroll i samme miljø som for prosjektstyring blir det enklere å koble sammen utvikling med gjøremål i backloggen. For å vedlikeholde kodebasen er det satt et krav om at for hver oppgave skal det opprettes en ny branch. Dette er gjort for å skape en god historikk slik at det er tydelig hva som er blitt gjort. Det vil også forhindre git-konflikter til en viss grad, ved at det bevisst arbeides på forskjellige oppgaver og forskjellige deler av kodebasen (Atlassian, u.å.).

2.4.4 *Risikoanalyse og evaluering*

Risiko defineres som en kombinasjon av mulige konsekvenser og den tilhørende usikkerheten som hører til (Aladon, 2018). Ved gjennomføringen av alle prosjekter vil det alltid høre med en grad av usikkerhet - og dermed risiko - rundt at noe uforutsett kan skje. For å ruste prosjektdeltakerne best opp rundt slike risikoer er det blitt foretatt en risikoanalyse i forkant av arbeidet. En risikoanalyse har som hensikt å identifisere så mange potensielle risikoer som mulig for så å vurdere dem. Grunnen til at dette gjøres før selve arbeidet begynner er at de involverte partene blir bevisste på risikoene og deres alvorlighetsgrad slik at de kan minimeres så mye som mulig. Figur 2.2 er en visualisering av resultatene fra denne prosessen hvor teamet tok i bruk en risikomatrise som måler sannsynlighet og konsekvens med en verdi fra 1-5. Totalsummen er en multiplikasjon av de to verdiene som brukes til å illustrere graden av alvor for hver identifiserte risiko.

Interne risikoer	Sannsynlighet	Konsekvens	Total
Kommunikasjonsproblemer innad i teamet	2	3	**6**
Utkastelse av ett eller flere medlemmer	1	5	**5**
Sosial loffing	3	5	**15**
Dårlig eller treg fremgang med eksperimentell prosjektgjennomføring	5	3	**15**
Manglende dokumentasjon av den agile arbeidsprosessen	1	3	**3**
Langvarig sykdom blant teamet eller viktige kontaktpersoner	2	4	**8**
Eksterne risikoer			
Manglende samarbeid med bedriften (økes om vi ikke får kontorplass)	3	4	**12**
Kravsspesifikasjon passer ikke det bedriften egentlig ønsker	1	3	**3**
Manglende samarbeid med universitetet	1	3	**3**
Sen eller ingen tilgang på viktig data og eller verktøy	2	4	**8**
Uventet personalendring i bedrit og eller KartAi-prosjektet	1	3	**3**
Faglige risikoer			
Feilberegning av tid	4	2	**8**
Overvurdering av oppgavers faglige krav	2	1	**2**
Manglende dokumentasjon av prosessen	1	5	**5**
Misforståelse av oppgaven	2	4	**8**
Nytteløse resultater på algoritmeutvikling	3	3	**9**
Manglende/feilfokusert tillæring av viktig ny kompetanse	3	3	**9**

Figur 2-2 - Risikomatrise

Risikomatrisen har en fargeskala for å markere de mest markante risikoene hvor da grønn er lav, gul er medium og rød er høy grad av risiko. En av de tre mest alvorlige risikoene som da ble merket i rødt var sosial loffing. Sannsynligheten for dette var ikke satt helt på topp ettersom teamet har jobbet sammen mye før bacheloroppgaven og er komfortable med hverandres arbeidsmetodikk. Den viktigste faktoren her er det faktum at prosjektet er i en mye større skala enn vi har vært borti før i tillegg til at prosjektets eksperimentelle natur også var nytt for teamet. Hovedgrunnen til at sosial loffing skjer er en mangel på klarhet blant deltakerne og med et slikt stort og nytt prosjekt med en veldig åpen oppgaveformulering var det tydelig at både sannsynligheten og konsekvensen av sosial loffing var veldig til stede (Martins, 2021).

En annen høy risiko var ganske enkelt «Dårlig eller treg fremgang med eksperimentell prosjektgjennomføring». Denne risikoen henger tett sammen med sosial loffing, men det er andre elementer i prosjektet som også er knyttet til denne. Manglende kunnskap, problemer med å finne den kunnskapen samt kommunikasjonsproblemer er også ting som kan spille inn på en tregere framgang med prosjektet. Risikoen var satt som veldig sannsynlig ettersom teamet forventet en treg start med et slikt annerledes prosjekt.

Den tredje røde risikoen var «Manglende samarbeid med bedriften (økes om vi ikke får kontorplass)». I ethvert prosjekt er kommunikasjon med produkteier essensielt for å levere et

forventet produkt. Vår kontaktperson i Norkart var veldig hjelpsom og kompetent, men som leder av hele KartAi-prosjektet forventet vi at det kunne bli en utfordring å ha jevnlig kontakt gjennom prosjektets løp. Sannsynligheten var ikke satt på maks ettersom det ble fra starten av satt opp statusmøter med vedkommende hver andre uke fram mot levering. Bekymringen kom fra at mye tok tid i starten, da spesielt å anskaffe kontorplass på Kartverket, men også det å få tilgang til de interne databasene vi trengte for å gjennomføre prosjektet. Det var konsekvensen av manglende kommunikasjon som dro denne risikoen opp mest.

2.5 Begreper

I dette kapittelet vil vi definere noen av begrepene som blir brukt gjennom rapporten.

Datakilder

Gjennom teksten vil det bli nevnt flere datakilder. En datakilde her er en kilde hvor informasjon om en eiendom kommer fra, delt inn etter en eiendoms-id (kalt GeoID).

Matrikkel

Matrikkelen er kanskje det mest sentrale aspektet i denne rapporten og er Norges offisielle eiendomsregister. Som datasett består matrikkelen kun av punktdata hvor hver eiendom får et ID-nummer og andre relevante opplysninger (Kartverket, 2021).

FKB

Felles Kartdatabase tar utgangspunkt i alle eiendommer som er ført inn i matrikkelen og fører inn de faktiske målene på alle bygg. Denne datakilden er da ren 2D og består av flere polygoner som brukes sammen med matrikkeldataen for å fremvise kommunens totale kartdata (Kartverket, 2021). Per i dag så føres denne dataen inn manuelt basert på flyfotoene.

Flyfoto

Foto er den hyppigst oppdaterte datakilden og blir generert ved jevnlige flygninger over landet hvor det tas flyfoto. Under denne prosessen blir hvert område flydd over flere ganger før resultatene kan brukes (Pihl & Solberg, 2021).

Ortofoto

Ortofoto er en sammensetning av alle flyfotoene som er tatt i én omgang av et spesifikt område. Ved å sette disse sammen blir også høydedata bestemt for hver piksel i bildet og gir dermed en «2.5D-effekt». Etterpå er det kjørt objektgjenkjenning ved bruk av AI, og fotoet blir gitt en GeoID for bruk i KartAi-prosjektet. (Mæhlum, 2021)

Laser / LiDAR

Lasermålinger er den mest omfattende datakilden ettersom den gir resultater i full 3D og ikke bare 2.5D. Ettersom den er såpass grundig er det også mye mindre hyppighet av disse observasjonene og det er bare én måling cirka hvert femte år.

Event Sourcing

Event Sourcing dreier seg om at i stedet for at data for et objekt er lagret i en celle og denne cellen blir oppdatert, så blir ny data lagret som nye rader. Dette gjør det mulig å se på historikken av endringer (Fowler, 2005).

3 Gjennomføring av prosjektet

Dette kapittelet tar for seg planlegging og gjennomføring av prosjektet. Først går det gjennom hvordan vi lagde en plan for hele prosjektet, så hvordan vi har valgt å opprette oppgaver og fordele dem og deretter fortelles det om de agile aktivitetene vi har utført gjennom prosjektet. Som forklart tidligere har vi valgt å beholde en del elementer fra Scrum fordi det er greie retningslinjer for å holde struktur i prosjektet. Til slutt oppsummeres hva som er blitt gjort i alle sprintene.

3.1 Planlegging

Fra starten av var det tenkt å gjennomføre prosjektet med Scrum-metodikken, men over tid viste det seg at Scrum ikke passet like godt for vårt prosjekt. Likevel har flere Scrum-elementer vært i bruk. Med relativt korte sprinter på to uker så fikk vi hyppige tilbakemeldinger som kunne brukes videre i sprint-planninger. Å ha et mer eksperimentelt prosjekt har åpnet opp muligheten til å være mer fleksible med tid. Ofte på grunn av at flere gjøremål var å forske og eksperimentere, så gled mye over i andre sprinter. Basert på

resultatene, ble det opprettet nye oppgaver som gjorde det vanskelig å lage en fullstendig plan fra starten av. Det ble laget et Gantt-chart som plan for hele semesteret vist i Tabell 3.1. I ettertid ser vi at vi har feilberegnet, og har ikke hatt tid til å ta i bruk maskinlæring for å måle pålitelighet.

TASK	PROGRESS	START	END
1. Analyse og design	100%		
1.1 Utforming av oppgave	100%	13/01/22	20/01/22
1.2 Oppgavefordeling og planlegging	100%	20/01/22	26/01/22
1.3 Flowchart av pålitelighetsmålsprosessen	100%		
1.3.1 Utformer ideer for prosessen individuelt	100%	27/01/22	01/02/22
1.3.2 Sette sammen ideer til et felles flowchart	100%	01/02/22	01/02/22
2. Manuell måling av påliteligheten til en endring	25%		
2.1 Utforme mock-data i JSON	100%	31/01/22	03/02/22
2.2 Utforme algoritmer	0%		
2.2.1 Teste og utforme algoritmene	0%	03/02/22	24/02/22
2.2.3 Føre statistikk på målingene	0%	05/02/22	24/02/22
3. Måle pålitelighet ved hjelp fra Maskinlæring	0%		
3.1 Utforske mulige løsninger for vårt problem	0%		
3.1.1 Implementere forskjellige maskinlæring-modeller	0%	26/02/22	10/03/22
3.1.2 Konkludere med hvilke modeller som funker og resultater	0%	10/03/22	17/03/22
4. Introdusere flere byggninger /dataset /observasjoner	0%		
4.1 Utvide arbeidsområdet	0%		
4.1.1 Teste algoritmer på ny data	0%	17/03/22	21/04/22
4.1.2 Føre statistikk på målingene	0%	17/03/22	21/04/22
5. Rapportskriving		18/02/22	30/05/22

Tabell 3-1 - Utdrag fra Gantt-chart

Som tidligere nevnt er Azure DevOps brukt for prosjektstyring. For hver Sprint-planning har teamet sittet sammen og utformet nye oppgaver basert på tilbakemeldinger fra Sprint-reviewet dagen før. Disse oppgavene, eller taskene, blir opprettet i Azure DevOps. Etter hvert som oppgaver blir tatt av et teammedlem så settes det et estimat på hvor lang tid oppgaven tar. Hvem som får hvilken oppgave, har basert seg mye på hva hvert medlem interesserer seg for. Noen har vært mer interessert i databaser, og derfor tatt mer initiativ i å undersøke database-relaterte teknologier og oppgaver. Flere oppgaver oppstår fra resultater av et medlems forskning og det er da naturlig for dem å jobbe videre med det. Eksempel på en slik oppgave er Figur 3.1.

Figur 3-1 - Eksempel på oppgave i Azure DevOps

3.2 Agile aktiviteter og Sprint-oversikt

Ettersom en agil metode er blitt brukt i prosjektet er det hensiktsmessig med en gjennomgang av de forskjellige aktivitetene som er blitt gjennomført og hvorfor de er brukt. Dette blir deretter fulgt av en gjennomgang av den agile prosessen.

3.2.1 *Sprint planning*

Sprint Planning aktiviteten er ment for å sparke i gang hver sprint og er essensiell for gode sprinter. Den offisielle Scrum-metodikken krever å ha produkteier til stede på hvert Sprint Planning-møte, og har blitt fulgt til teamets største evne (West, u.å.). Sprintlengdene på to uker ble designet slik at statusmøtene med produkteier alltid sammentraff med Sprint Planning-møtene. Denne aktiviteten er såpass viktig for tilbakemelding, planlegging og generell kommunikasjon at det naturlig ble besluttet å ha med i den agile prosessen.

3.2.2 *Daily Standup*

Daily Standup eller Daily Scrum er et kort møte mellom teammedlemmene hvor framgang og planer for neste dag blir diskutert. Dette møtet ble holdt av flere grunner. For det første så gir det hvert medlem en oversikt over alt arbeidet som gjøres, som igjen gir alle en økt helhetsforståelse av vårt eget arbeid og prosess. For det andre så gir det teamet god mulighet for grundig dokumentasjon av arbeidsprosessen som igjen gir gjennomsiktighet i arbeidsprosessen og kan brukes for videre læring. For det tredje hjelper det med kvalitetssikring ved å begrense sjansen for sosial loffing blant teammedlemmene (Chapman, 2018).

3.2.3 *Sprint Review*

Sprint Review er et viktig tilbakeblikk på hva som er blitt levert i den aktuelle sprinten og er blitt implementert i dette prosjektet (Scrum.org, u.å.). Å gå gjennom de fullførte oppgavene sammen med produkteier sikrer kvalitet og god kommunikasjon. Grunnet hvordan statusmøtene ble strukturert så ble det besluttet at Sprint Review og Sprint Planning tas samtidig på disse møtene. Opprettelse av oppgaver ble gjort i ettertid, ettersom det ble veldig tidkrevende for produkteier å være med på hele prosessen. Disse taskene ble diskutert under møtet.

3.2.4 *Sprint Retrospective*

I motsetning til Sprint Reviewets fokus på leverte produkter, fokuserer Sprint Retrospective på medlemmene i teamet og gangen i arbeidet (Cohn, u.å.). Scrum Master er tradisjonelt den som holder møtet og det er også det teamet har holdt seg til. Disse møtene blir gjort for å

forbedre de interne prosessene hos teamet selv ettersom det kun er teammedlemmene som er til stede.

3.2.5 *Sprintene*

For å gi et fullt innblikk i hele prosessen vår, følger et referat av alt som er blitt gjort av teamet. All denne informasjonen er basert på vår egen dokumentasjon da i all hovedsak fra notater fra Daily Scrum og Retrospektiv.

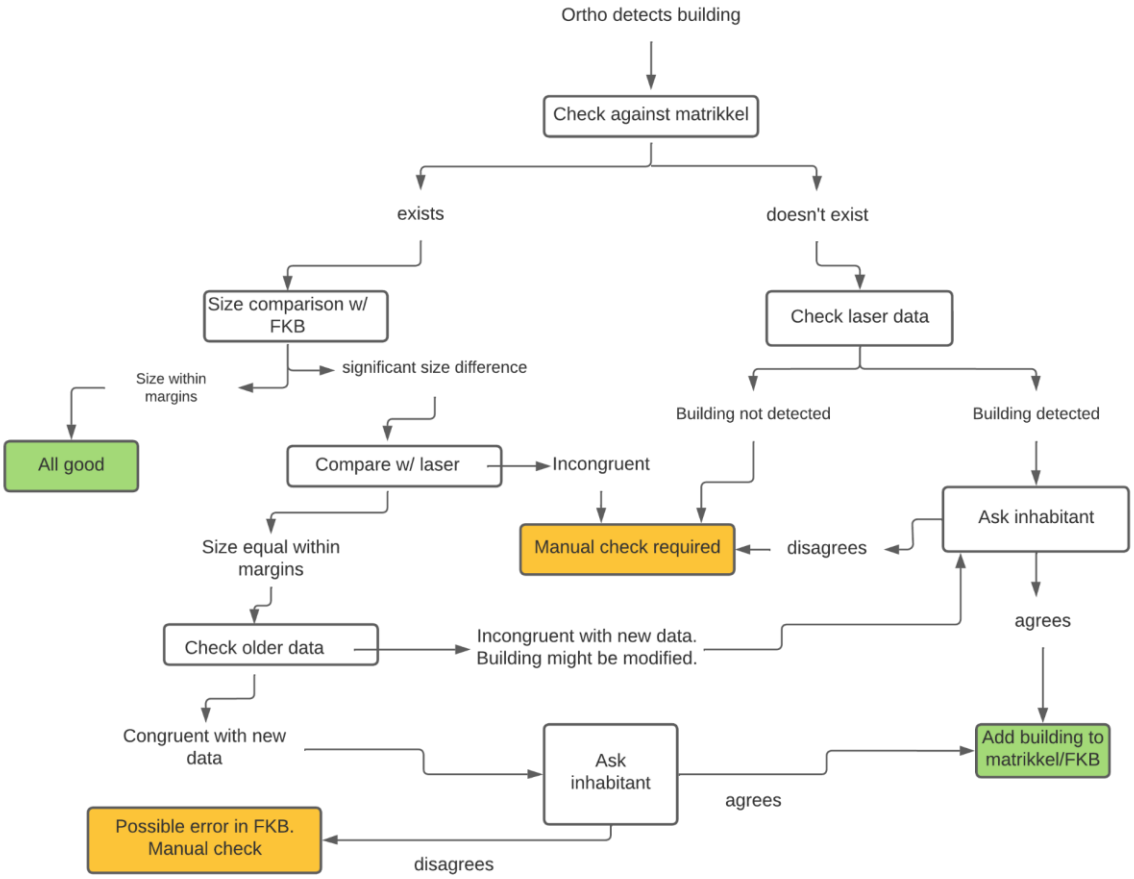
Pre-sprint og Sprint 1

Pre-sprinten og den første sprinten i prosjektet var preget av kartlegging av store mengder informasjon som ble presentert på én gang. Teamet hadde flere møter med produkteier fra Norkart og en samarbeidspartner fra Kartverket som introduserte oppgaven. Under pre-sprinten ble målene satt til å sette opp oppgavestyringen og versjonskontrollen i Azure DevOps, samt forberede oss for møter og intervjuer med samarbeidspartnere. Da disse målene ble nådd uten mye problemer planla vi den første sprinten. Oppgavene her var kompetanseheving for alle teammedlemmer som innebar lesing av KartAis rapporter som vi hadde fått tilgang til, samt innføring i AI-prinsipper og viktige verktøy som Python, QGIS og PostGIS. Lære å bruke verktøy og lese teori var ikke like tydelige produkter å vise frem på slutten av sprinten, men teamet opplevde en totalt sett virkningsfull effekt av målene.

Sprint 2

Etter å ha hevet kompetansen rundt fagområdet ble teamet klare for å sette mer håndfaste mål. Modellering av løsninger ble igangsatt for alle medlemmene etter sprint planning for Sprint 2 i tillegg til en fremdriftsplan. Johannes startet å utvikle mockdata på bygningsobservasjoner, Bendix utformet en risikoanalyse for prosjektet og Eirik og Thomas tok hovedansvaret for modelleringen. Thomas' modell er vist som Figur 3.2. Disse produktene ble levert før slutten

av sprinten og en felles modell ble utviklet basert på alles bidrag.



Figur 3-2 - Thomas' modell for systemet

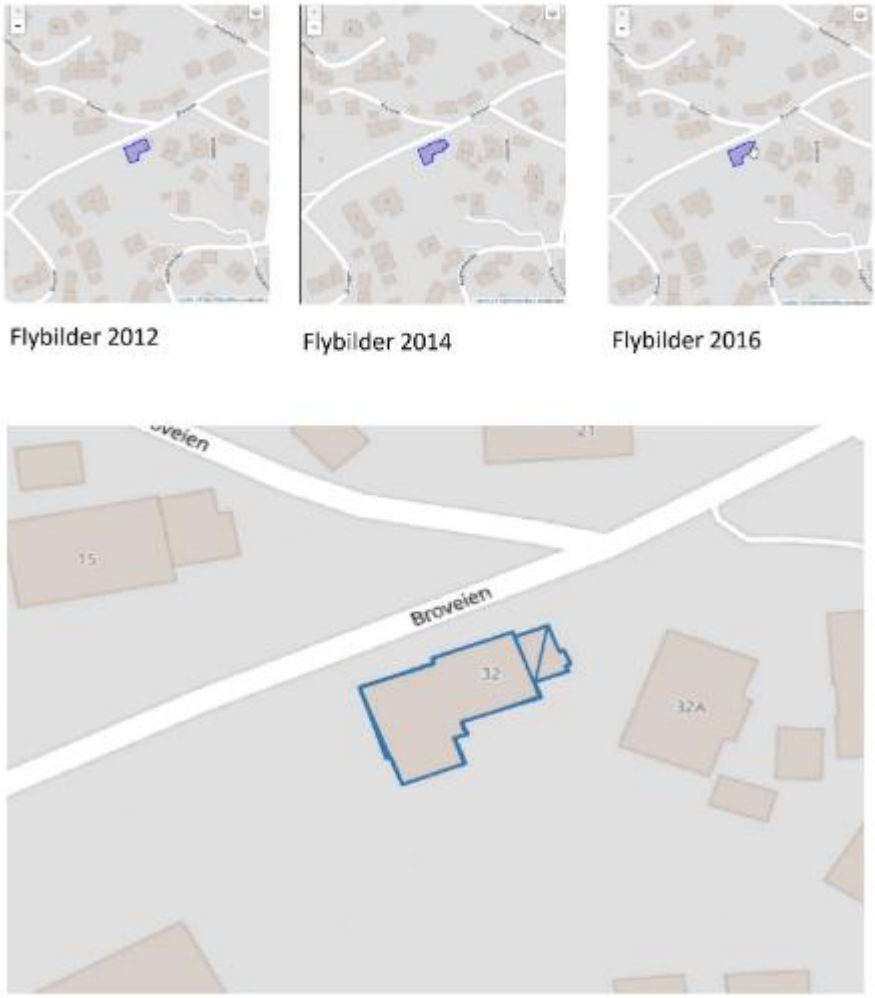
Sprint 3

Målene for denne sprinten var satt ganske lavt ettersom teamet hadde mye å gjøre utenfor selve bachelorrapporten som måtte prioriteres. Sideemnet IS-305 hadde flere obligatoriske aktiviteter i tillegg til at Norkart arrangerte et seminar for hele KartAi-prosjektet hvor teamet holdt en omfattende presentasjon over arbeidet som var blitt, og som skulle bli, gjennomført. Johannes fikk laget et Python-program for sammenligning av areal, mens de andre medlemmene foretok mer forskning for å bygge opp det videre arbeidet i prosjektet.

Sprint 4

Målene for denne sprinten var arbeid med databasetilkobling og mye fokus rundt 3D-data med Thomas i spissen. Teamet ble anbefalt av produkteier å ha videre intervjuer med to masterstudenter og en tidligere doktorgradsstudent for å få god relevant informasjon. Denne digresjonen gjorde at teamet ble nødt til å endre målsettingen i sprinten og 3D-data-målet ble

noe forsinket, men databasetilkoblingen ble satt opp som planlagt. Figur 3.3 viser hvordan mockdata for et bygg ble seende ut.



Figur 3-3 - Mockdata i PostGIS

Sprint 5

Stafettpinnen ble plukket opp igjen når Johannes også begynte å jobbe med 3D-dataen. Resten av teamet jobbet videre med nye pålitelighetsskisser som gikk mer i detalj enn de forrige. På samme vis som sist gang lagde alle én hver og diskuterte deretter seg fram til en samlet modell i fellesskap som var klar til slutten av sprinten. Teamet fant mye om 3D-data og sammenligning av den med 2D-data.

Sprint 6

Resultatene for modelleringen ble brukt til å sette ordentlig i gang med utviklingen og Thomas og Eirik startet å parprogrammere fram algoritmen. Samtidig fikk teamet innspill fra

```
BIoU for  1 mot 1 : 1.0
BIoU for  2 mot 1 : 0.9551849918790408
BIoU for  2 mot 2 : 1.0
BIoU for  3 mot 1 : 0.8533852135890375
BIoU for  3 mot 2 : 0.8512249193042466
BIoU for  3 mot 3 : 1.0
Sammenlignes med:  9
```

Figur 3-4 - Arbeid med BIoU-sammenligning i datasett.

produkteier og veileder om å ha større fokus på rapporten, og Bendix begynte på det. Johannes brukte det han og Thomas hadde funnet ut til å utvikle en algoritme for sammenligning av endringer mellom 2D og 3D-datasett. Eirik og Thomas' algoritme trengte mer utviklingstid på slutten av sprinten, men Johannes fikk levert etter planen. Bendix fikk også fylt ut mye av rapporten. Figur 3.4 viser sammenligning mellom polygoner.

Sprint 7

Framgang i denne sprinten ble sterkt påvirket av ferieavvikling i forhold til påsken og lite kontakt med produkteier av samme grunn. I tillegg var denne perioden preget av flere frister i andre fag som tok mye av tiden vekk fra det teamet ønsket å få gjennomført. Tross dette kom mye nyttig tilbakemelding fra samarbeidspartner i Kartverket som anbefalte å sette spissen på bruk av sikkerhetsmål fra AI-en og standardavvik i algoritmen. Dette ble det laget nye modeller på for videre utviklingsarbeid som måtte utsettes til neste sprint. En plan om å implementere algoritmen på ekte data produsert av noen samarbeidspartnere i KartAi ble lagt, men den dataen som teamet fikk var ikke fullstendig nok til å brukes og mer data ble etterspurt. Teamet holdt også et styringsgruppemøte nær slutten av påske som produkteier dessverre ikke kunne delta på, men samarbeidspartneren fra Kartverket og veileder var til stede. Med skrivingen av rapporten, fullførte Johannes en restrukturering av alle overskriftene og innholdet etter tilbakemelding fra veileder.

Sprint 8

Gjennom sprint nummer åtte hadde teamet to hovedpunkter å arbeide med: gjenværende rapportskriving og implementering av tilbakemeldingene på algoritmen som teamet fikk i sprint 7 fra Kartverket. Denne sprinten gikk svært effektivt med god framgang på rapporten og en ferdigstilt algoritme til slutten av sprinten.

Sprint 9

Den siste sprinten var preget av hyppige innleveringsfrister og mye finpussing og etterarbeid. IS-305 hadde en innlevering veldig nærme fristen for bacheloroppgaven så mye tid gikk vekk fra denne sprinten til det faget. Heldigvis hadde ikke Thomas dette faget så han fikk gjort mye mens de andre medlemmene var opptatte med IS-305. Etter at resten av teamet var ledige igjen, ble fullt fokus lagt på rapportskriving, og rapporten ble ferdigstilt.
Denne sprinten håpet vi å kunne ha brukt algoritmen på ekte data produsert internt i KartAi, men samarbeidspartnerne fikk dessverre ikke tid til dette, så denne planen ble vraket tidlig i Sprint 9.

4 Eksperimentering

I dette kapittelet tar vi opp alt av forskning og eksperimentering vi har vært gjennom. Fra starten hvor vi tar opp modelleringen av pålitelighetsmålet, til eksperimentering med forskjellige teknologier, og til slutt en gjennomgang av algoritmer som er blitt utviklet. Dette kapittelet fungerer som en ferdig rapport av vårt arbeid for Norkart og som det endelige produktet.

4.1 Analyse og modellering

Dette kapittelet går over analysefasen som omhandlet mest innhenting av informasjon og utvikling av kompetanse. Her beskrives også modellene som ble laget samt en beskrivelse av prosessen av å utforme dem.

4.1.1 *Analyse*

Hele den første sprinten i prosjektet ble brukt til analyse og design av oppgaven ettersom problemstillingen som ble presentert krevde en del forarbeid. Analysefasen var derfor en sentral del av prosjektet. Fokuset lå på grunnleggende kunnskap om AI-teknologi, digitale verktøy samt tilvenning til Python – et språk teamet ikke hadde vært nevneverdig borti før. Den første delen av denne fasen bestod av kompetanseheving om de relevante fagområdene som krevdes for å kunne gjennomføre prosjektet. Det mest åpenbare området å sette seg inn i var de tidligere rapportene fra AP2 som lå ute på KartAi sine hjemmesider (KartAi, u.å.). Da

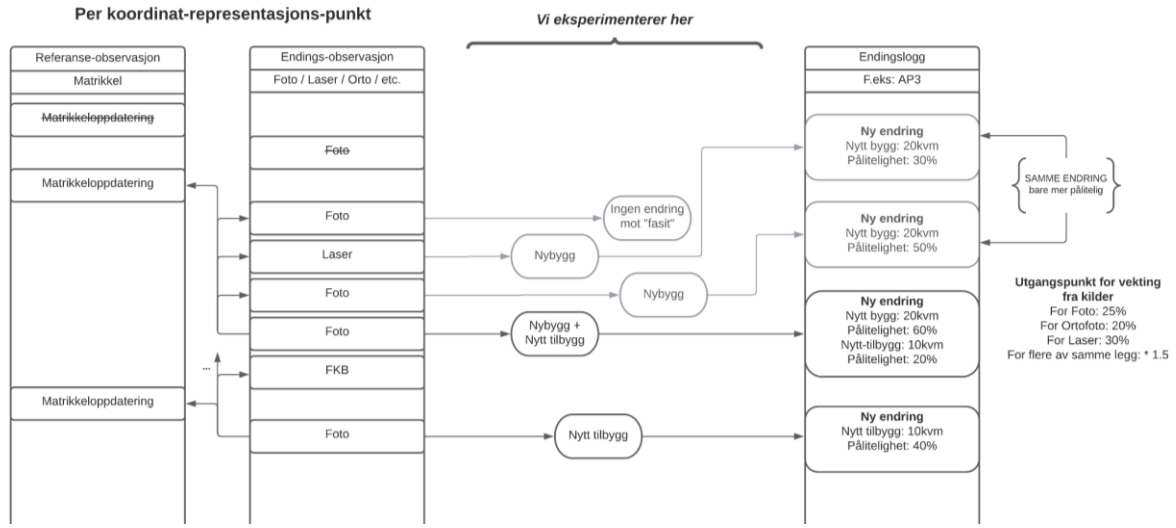
spesielt lå søkelyset på byggidentifikasjon-rapporten fra ansatte i prosjektet og en masteroppgave om en AI-modell som var tenkt til å brukes. Forskjellen mellom laserdataen (3D) og bildedataen (2D/2.5D) gjorde at forskningslitteratur om sammenligning mellom 3D og 2D også ble viktig for analysen. Teamet konkluderte med flere rapporter med mye relevant informasjon innen temaet. En av dem var en forskningsartikkel publisert i International Journal of Geo-Information (2019) som så på hvordan generere ortorektifiserte 2.5D-kart fra massive 3D modeller av byer kunne øke kvaliteten på kartinformasjonen (Liang et al., 2016). For å finne mer nøyaktighet var også en rapport fra Sensors (2019) som undersøkte et rammeverk for å oppdage 3D-objekter i en point cloud veldig nyttig (Xu et al., 2019).

Siste del av analysefasen gikk ut på å identifisere de teknologiene og programmene som var nødvendige for å gjennomføre prosjektet. Deretter begynte en fase med tilnærming og opplæring i bruk av disse programmene, da spesielt Python og QGIS.

4.1.2 *Modellering*

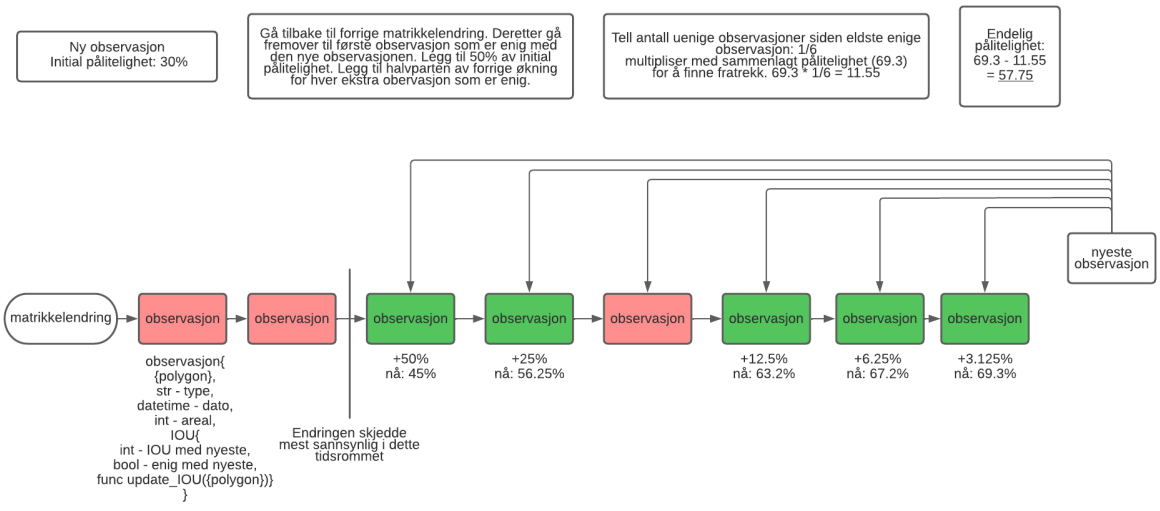
Etter at hvert gruppemedlem fikk utformet sine tanker gjennom bruk av wireframes, ble det utformet en modell som viser hvordan systemet som designes kan passe mot flyten av AI-resultater og oppdateringer. Modellering ble gjort ved bruk av flowcharts i programmet Lucidchart av alle teammedlemmene. Det ble laget flere utkast til løsninger slik at flest mulig ideer ble prøvd ut. Alle disse kan finnes i Appendix E. Her ble det sett på flere modeller for hvordan et endelig system kunne se ut hvor noen så mest på flyten av prosesser mens andre modeller tok for seg mer av detaljene i algoritmen som prosjektet krever. Den endelige modellen som ble tatt med videre var en sammensetning av flere aspekter fra de første modellene og kan sees på Figur 4.1. Modellen deler det opp i referanseobservasjoner og endringsobservasjoner som sammenlignes med hverandre. Matrikkelen er i referanseobservasjonen og blir hele tiden sammenlignet med endringsobservasjonene når det kommer en ny kilde fra AI-en. Samtidig lagres alle datakilder på samme bygg og når en ny observasjon kommer inn, går systemet gjennom alle tidligere lagrede observasjoner i endringsobservasjonen før den sammenligner med referanseobservasjonen. Hver gang dette skjer, sendes en endring til endringsloggen hvor pålitelighetsmålet kalkuleres. På grunn av

dette blir det flere endringer som kalkuleres mot hverandre for å beregne den endelige, totale påliteligheten på det ene bygget.



Figur 4-1 - Oversikt over oppgavens posisjon

Denne modellen fikk i gang tenkemåten på hvordan løsningen kan se ut og var til stor hjelp med videre arbeid. Men videre mot den faktiske utviklingen hadde teamet behov for en mer detaljert modell som tok for seg de faktiske tallene og utregningene som algoritmen må gjøre for å fungere. En ny modelleringsprosess ble startet hvor fokuset lå på en mer praktisk tilnærming som skulle kunne brukes til å produsere den spesifikke algoritmen. Prosessen var den samme i at hvert medlem lagde én modell individuelt før de ble sammenlignet i plenum. Alle disse kan finnes i Appendix F. Mange av de samme prinsippene som var tenkt fram i den overordnede modellen kom med i det endelige resultatet vist i Figur 4.2.



Figur 4-2 - Modell for utregning av pålitelighet på observasjoner

Modellens første mål er å identifisere hvor den faktiske endringen fant sted ved å sammenligne alle nye observasjoner med de tidligere og dermed «sette en strek» der modellen antar endringen skjedde. Modellens endelige formål er å bruke observasjonene etter streken til å regne ut påliteligheten på den endringen som modellen nettopp fant. Markert i grønt er observasjoner som er enige med den nyeste observasjonen som da høyner påliteligheten (som starter på 30% i dette eksemplet). Markert i rødt er observasjoner som er uenige med den nyeste observasjonen. Den første enige observasjonen legger til 50% av den daværende sannsynligheten, så vi får 45%. For alle grønne observasjoner etter dette halveres den 50%-økningen for å sørge for at økningen er skalerbar og at det resulterende pålitelighetstallet ikke kan bli astronomisk høyt. Etter de grønne observasjonene er lagt sammen til et pålitelighetstall skal de røde observasjonene brukes til å redusere tallet. De røde observasjonene trekker ned scoren i samsvar med forholdet av røde til grønne observasjoner etter streken. I dette eksempelet var én av seks observasjoner uenige, så det trekkes fra ⅙ av tallet vi kom fram til etter å ha lagt sammen de grønne observasjonene. Denne modellen dannet grunnlaget for hovedprinsippene for algoritmene teamet utviklet som utdypes i mer detalj senere i kapittel 4.

4.1.3 *Event sourcing*

Event sourcing er et konsept vi ble introdusert til i starten av prosjektet og er beskrevet under kapittel 2.5. Så for vårt prosjekt hvor vi har forskjellige datakilder hvor det er nye observasjoner hvert år, vil dette være en aktuell metode å bruke. Hvert bygg vil da ha en liste hvor alle observasjonene til det aktuelle bygget lagres. Utfra denne listen kan vi hente ut observasjoner innenfor et valgt tidsrom og se etter endringer som vi kan bruke til å lage et pålitelighetsmål ved å sammenligne observasjoner (Fowler, 2005).

Å modellere en database basert på Event Sourcing er utenfor vårt scope. Vårt prosjekt er en del av modulen «Endringsanalyse» og eksperimenter vi utformer vil baseres på en etterlignet versjon av en slik database.

4.2 Sammenligning av datasett

For å skape et pålitelighetsmål trenger en data å basere det på. Disse dataene produseres ved å sammenligne geometriske former mellom observasjoner fra forskjellige datasett. Disse

observasjonene vil være lagret i en database-tabell tilknyttet bygget de tilhører. De kan være uten geometri eller av typer som punkt, linje, flate eller volum-objekt. Dette kapittelet vil ta for seg forslag på hva som skal sammenlignes, hvordan det kan sammenlignes, teknologier, biblioteker og andre problemstillinger.

4.2.1 *Flater (Foto, ortofoto, FKB)*

FKB og AI-resultater fra foto og ortofoto har en 2D flate (polygon) som geometritype. I denne seksjonen går vi inn på hva som kan sammenlignes og hvilke funksjoner og verktøy som er blitt brukt til å gjennomføre sammenligningen.

Python

Det første vi har eksperimentert med er å se på forskjeller i areal hos flater. Dette er blitt gjort på mockdata i Geojson av observasjoner på et bygg (http://geojson.io/#map=21/58.52313/6.51088) med Python-biblioteker. I starten ble det brukt biblioteket Shapely for å lage polygoner ut fra koordinatene til mockdataen. Men å beregne arealet ut fra disse polygonene resulterte i feil resultat, ettersom programmet ikke forsto at polygonene er basert på den ekte verden. Derfor fant vi funksjonen «area» fra biblioteket «area» som tar 'geometry'-egenskapen fra et Geojson-objekt for å så returnere areal som kvadratmeter. (https://github.com/scisco/area). Figur 4.3 viser et resultat fra eksperimentering med å finne forskjeller mellom polygoner i Python.

```
2020-04-02 - Ortofoto - Polygon - 158.5306m²
|    Difference in area: 2020-04-02 - Foto: 0.0002m²
|    2020-02-02 - Matrikkel exists in between this observation's coordinates: True
|    Difference in area: 2020-02-02 - FKB: 0.0001m²

2021-02-02 - Foto - Polygon - 188.6582m²
|    Difference in area: 2020-04-02 - Ortofoto: 30.1276m²
|    Difference in area: 2020-04-02 - Foto: 30.1278m²
|    2020-02-02 - Matrikkel exists in between this observation's coordinates: True
|    Difference in area: 2020-02-02 - FKB: 30.1277m²
```

Figur 4-3 - Resultat fra algoritme

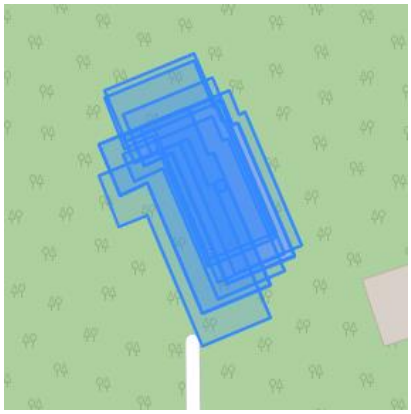
PostGIS

Etter en del eksperimentering og leting etter løsninger i Python ble vi oppmerksomme på PostGIS. PostGIS er utvidelse for PostgreSQL database-serveren som gjør det mulig å lagre geometridata. I tillegg har PostGIS en rekke funksjoner for alle typer geometriobjekter som inkluderer både 2D og 3D. Å bruke disse funksjonene for å få tak i data til pålitelighetsmålet passer fint, ettersom det er tenkt at noe av KartAis data skal ligge i PostGIS-databaser (Norkart, 2021, 22:40). Det er verdt å nevne at det ikke er 100% sikkert at all data vil ligge i PostGIS-databaser. PostGIS' innebygde funksjoner kan fortsatt brukes i spørringer med data fra andre systemer, og er derfor ikke avhengig av at data lagret i PostgreSQL. PostGIS er et verktøy vi har sett på som relevant ettersom PostGIS er blitt nevnt i tidligere rapporter og videoer, og det ser ut til å være den mest populære kombinasjonen av gis og database-system (PostGIS + PostgreSQL). Derfor har vi valgt å primært bruke PostGIS i eksperimentering.

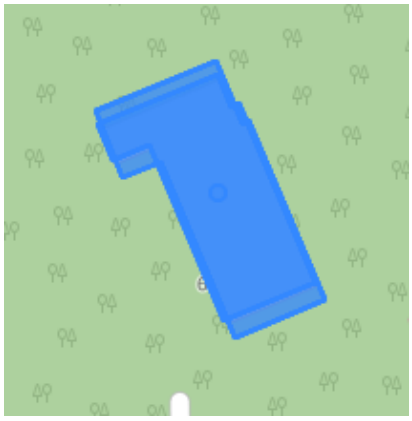
ST_Intersection er en funksjon som returnerer en flate som dekker fellesområde mellom to polygoner. For 3D-objekter kan funksjonen ST_3DIntersection brukes. ST_Difference returnerer det motsatte, altså en flate av forskjellen mellom to polygoner. Vi har funnet at dette er en funksjon som kan være veldig nyttig for å finne hvor det er gjort et påbygg.

ST_Union returnerer overlappen mellom to polygoner og kan brukes sammen med ST_Intersection og ST_Area (areal av geometri) for å finne Intersection over Union (IoU). For å se om et punkt eksisterer innad i en flate kan ST_Contains brukes.

For å flytte flater kan en bruke funksjonen ST_Translate ved å for eksempel flytte alle observasjoner til et punkt. Som du kan se på Figur 4.4 er det forskjell i hvor flatene ligger. Denne funksjonen vil være aktuell ettersom flybilder ikke blir tatt med i helt lik vinkel hver gang. Derfor vil det resultere i noe lignende av hva vi har mocket. Figur 4.5 viser resultatet av å bruke ST_Translate.



Figur 4-4 - Polygoner som er forskjøvet i forhold til hverandre



Figur 4-5 - Korrigerte polygoner

I tillegg til funksjonene over har PostGIS to andre funksjoner som gir et mål på hvor like to flater er: ST_HausdorffDistance og ST_FrechetDistance. Dette er to funksjoner som sammenligner to sett med koordinater mot hverandre. Hausdorff-distansen ser på avstanden mellom hvert nærliggende koordinat i hvert sett, mens Frechet-distansen regner ut likhet ved å se på kurvene og rekkefølgen av koordinater i polygonene.

Siste funksjon er ST_Rotate som er en funksjon for å rotere geometrier. Om det har seg at flatene fra den ekte dataen differerer i retning, kan en med kombinasjon av ST_Rotate og en utregning av IoU, rotere et polygon til en bedre retning. Tanken er at en utfører et IoU-mål for hver grad som roteres og konkluderer med at resultatet med høyest IoU kan brukes.

4.2.2 *Generalized intersection over union*

«Generalized intersection over union» (eller *GIoU*) er en forbedring på den standardiserte IoU-målingen. Standard IoU blir også kalt for Jaccard index, og har mange sterke sider som gjør den nyttig for sammenligning av data. Den er velegnet som en matematisk metric og den bryr seg ikke om skalaen på objektene som blir sammenlignet. En svakhet med Jaccard index er at den ikke tar for seg hvor lang distanse de ikke-overlappende områdene bommer med. Det vil si at hvis prediksjon-boksene og «ground truth»-boksene, altså de som faktisk er der, er rett ved siden av hverandre, men ikke overlapper, vil de få samme IoU-måling som om prediksjonen var på andre siden av jordkloden.

Med vanlig IoU er det binært om et område av prediksjonen overlapper eller ikke med «ground truth». Med GIoU er det en gradient som beskriver hvor langt det ikke-overlappende området bommer med, og dette blir brukt til å justere scoren.

Dette skjer fordi GIoU finner den minst mulige boksen som dekker både prediksjon og «ground truth», for så å finne arealet av den boksen minus prediksjon og «ground truth»-boksene, og scoren reduseres tilsvarende som beskrevet i Figur 4.6.

Algorithm 1: Generalized Intersection over Union

input : Two arbitrary convex shapes: $A, B \subseteq \mathbb{S} \in \mathbb{R}^n$

output: $GIoU$

1 For A and B, find the smallest enclosing convex object C, where $C \subseteq \mathbb{S} \in \mathbb{R}^n$

2 $$IoU = \frac{|A \cap B|}{|A \cup B|}$$

3 $$GIoU = IoU - \frac{|C \backslash (A \cup B)|}{|C|}$$

Figur 4-6 - Generalized Intersection over Union fra Rezatofighi et al, 2019, s. 3

IoU gir et tall mellom 0 og 1, mens GIoU gir et tall mellom -1 og 1. Der to bokser ikke overlapper i det hele tatt får de 0 uansett fra IoU, men de kommer nærmere -1 jo lengre de er i fra hverandre på en GIoU måling.

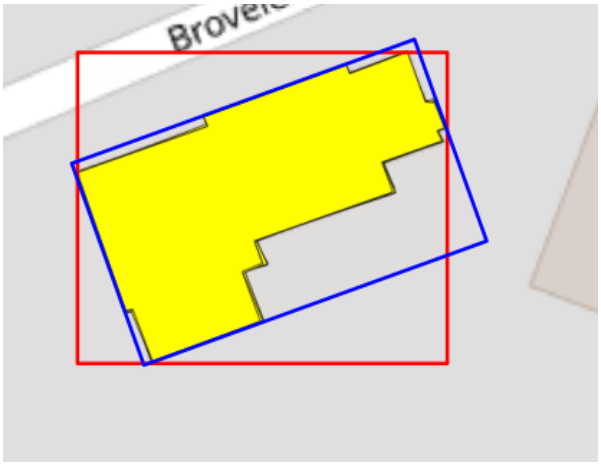
Vi testet ut GIoU med mockdata fra databasen for å se om dette kunne passe i prosjektet vårt. Mockdataen skapte vi ved å manuelt tegne polygoner gjentatte ganger over et bygg ved hjelp av QGIS. Ettersom disse ble tegnet for hånd hver gang uten å se på de forrige tegningene, oppnådde vi små variasjoner som er realistiske med tanke på AI-resultater. Etter dette tegnet vi nye versjoner der en del av bygget ikke er med. Det var tenkt at dette skulle simulere at det har skjedd en modifisering av bygget. Fra Stanford-rapporten (Rezatofighi et al., 2019) fant vi formelen for å regne ut GIoU (Figur 4.6). Dette lar seg gjøre med PostGIS-funksjoner, så vi utviklet en SQL-spørring (Figur 4.7) som sammenligner GIoU rett i databasen.

```sql
select
    iou,                                            -- final IoU
    iou - (( area_c - add_area) / area_c) as giou   -- calculated GIoU
from (
    select
        intersect_area / (sum_area - intersect_area)  as iou,
        ( st_xmax(union_poly) - st_xmin(union_poly) ) * ( st_ymax(union_poly) - st_ymin(union_poly) ) as area_c,
        sum_area - intersect_area  as add_area
    from (
        select
            st_area( st_intersection(geo1, geo2) )  as intersect_area,
            st_union(geo1, geo2)                    as union_poly,
            st_area(geo1) + st_area(geo2)           as sum_area
        from
            (
                select (select geom from observasjon ob where id = 10) as geo1, (select geom from observasjon ob where id = 11) as geo2
            ) as query_1
        ) as query_2
    ) as query_3
;
```

Figur 4-7 - SQL spørring i PostGIS som regner ut GIoU for to observasjoner

Det viste seg derimot at GIoU ikke egner seg så godt på ikke-rektangulære figurer. Vanligvis så gir du punktet til hvert hjørne av de to rektanglene du sammenligner til funksjonen for å regne ut GIoU. Bygg er ikke alltid rektangulære så vi var noe usikre på om dette kom til å gi sterke resultater for oss, men valgte å eksperimentere. Vi ga funksjonen de høyeste og laveste punktene på X- og Y-aksen (illustrert i Figur 4.8). Vi eksperimenterte også med hvordan vi skulle orientere disse boksene som er grunnlaget for GIoU, se Figur 4.8 og Figur 4.9, men vi så at tilbygg innenfor en av firkantene ikke ville gitt stort nok utslag på resultatet til å behandle det videre.

GIoU tar fire punkter som input for hvert polygon som skal sammenlignes. Dette fungerer utmerket når du sammenligner «bounding boxes» fra en objektdeteksjons-AI, men med komplekse, mangekantede bygg fungerer dette dårlig. Om det for eksempel skjer en utbygging nede til høyre på bygget i figuren, så vil boksen forbli den samme, men den forandrer seg mye om utbygget hadde skjedd på venstre siden av bygget. Siden det er bounding boksen som blir sammenlignet i GIoU-funksjonen er dette et stort problem.

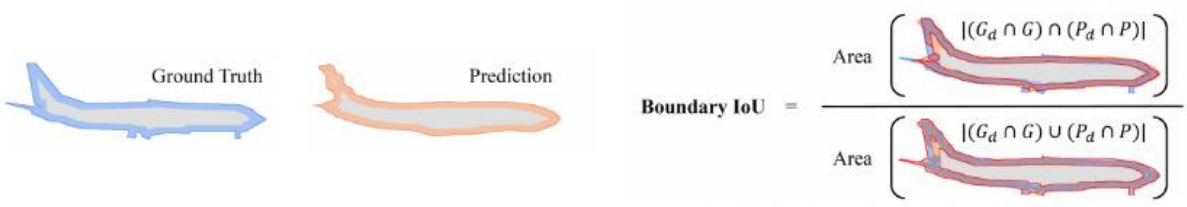


Figur 4-8 - Problematikk med GIoU

Figur 4-9 - Problematikk med GIoU #2

4.2.3 *Boundary Intersection over Union*

Da GIoU viste seg å bli vanskelig å benytte, begynte vi å se på «Boundary Intersection over Union», eller *BIoU*. Dette er en veldig ny måling som ser på objekt-sammenligning på en smart måte. Figur 4.10 viser hvordan BIoU sammenligner to objekter. Man lager en «inwards stroke» fra konturen til objektet for å få de fargede områdene. Det grå området blir tatt ut av betraktning. Deretter blir disse områdene sammenlignet med en standard IoU-måling. Dette oppnår et større fokus på formlikhet, da vi har både en indre og ytre kant som straffer scoren om de ikke samsvarer.



Figur 4-10 - Forklaring av boundary intersection over union fra Cheng et al, 2021, s. 5

4.3 3D objekter

Dette kapittelet vil ta for seg mulighetene vi har funnet for å kunne sammenligne 3D-objekter med hverandre og med 2D-flater. Laser-skanninger er i dag eksportert som dxf-filer. Hvordan disse skal transformeres til en gyldig type som kan lagres i en PostGIS-database eller noe annet, er utenfor prosjektets scope. Vi vil primært skildre ideer om hvordan 3D-objekter kan være med på å styrke et pålitelighetsmål.

Slik det er nå, produseres nye laserdata hvert femte år, mens flyfoto produseres hvert år og kan observere et bygg opp til seks ganger. Ortofoto produseres hvert andre år. Dermed oppstår det langt flere observasjoner fra fly- og ortofoto enn LiDAR, og foreløpig bruker ikke AI-en sistnevnte datasett. AI-modellene har hatt gode resultater, og blir mer nøyaktig ved bruk av ortofoto hvor høyden også hjelper til. Fortsatt er det utfordringer med 2D/2.5D-data, som for eksempel med ulike høyder i terrenget rundt bygningen og hus med torvtak, og det er derfor ønskelig å få inn 3D-data som en ekstra datakilde. Det tenkes at et 3D-objekt med flere og bedre detaljer, vil kunne være med på å tilnærmet bekrefte om en endring mellom noen 2D-observasjoner er korrekt.

4.3.1 *3D objekter i PostGIS*

PostGIS, som beskrevet i tidligere kapitler, kan også lagre 3D-geometrier og innebærer at en ekstra dimensjoner legges til for hver koordinat. I motsetning til PostGIS's support for 2D-geometrier, er dokumentasjonen for 3D-geometrier svak og det er få eksempler på sammenligning av 3D-objekter.

Dokumentasjonen nevner POLYHEDRALSURFACE Z som en 3D-type som består av polygoner. Og TIN som er en samling trekanter som eksisterer i 3D-rom. Men i tillegg til dette har en og muligheten for å lage punkter og linjer i 3D-rom og få 3D MultiPoints eller MultiLineStrings.

PostGIS har tre funksjoner som kan være reelle for å sammenligne 3D-objekter. Disse er ST_Volume, ST_3DDifference og ST_3DLenght. ST_Volume returnerer volumet til et 3D-objekt. En differanse i volum vil ofte samsvare med en differanse i areal hos et polygon, men ikke alltid. Et tilbygg som blir bygget under et tak vil ikke kunne fanges opp i et flybilde. Funksjonen ST_3DDifference sammenligner to 3D-geometrier og returnerer en 3D-geometri som representerer forskjellen mellom dem. Om ST_3DDifference kan gi noe nytte under pålitelighetsmålet er ikke sikkert, men videre i endringsanalysen hvor forskjeller skal identifiseres kan den være nyttig.

De tidligere forklarte funksjonene støtter alle typen POLYHEDRALSURFACE Z. Funksjonen ST_3DLength fungerer bare med MultiLineString og denne vil returnere lengden av alle linjer kombinert.

4.3.2 *3D mot 2D*

En av datakildene med betydelig omfang vi har tilgang til er LiDAR-skanninger. Etter noe bearbeiding kan disse produsere detaljerte point cloud 3D-objekter av hele det skannede området ved hjelp av lasermålinger. Vi vil sammenligne dette med dataen fra flyfoto, men flyfoto produserer ikke 3D-data og er ikke direkte sammenlignbart med dataen fra lasermålinger. Fordi det alltid blir tatt flere flyfoto av samme område, er det innhentet nok informasjon til å bruke visuelle målinger for å bestemme høyden til de avbildede byggene. Denne prosessen skaper ikke de samme resultatene som ordentlige 3D-målinger, men heller noe som kalles for 2.5D-objekter. For å kunne sammenligne disse med dataen fra lasermåling

vil vi nedskalere 3D-objektene til 2D, men ta vare på høydevariabelen. Da kan vi først gjennomføre en BIoU-måling mellom 2D-flatene, for så å sammenligne høydene.

2.5D-objekter kan ha overraskende nøyaktige høydemålinger på bygg, med tanke på at målingen er gjort visuelt i ettertid. Forskning publisert i International Journal of Geo-Information (Liang et al., 2016) viser at feilmarginen for høydeforskjell mellom todimensjonale visuelle målinger (2.5D) og faktiske 3D-målinger som regel er godt under en meter, selv for høye boligblokker (Tabell 4.1). Dette viser at høydemålingene er gode nok til å sammenlignes med dataene vi får fra LiDAR.

Table 4. Comparison of the building heights measured on a 2.5D map and in a 3D GIS.

Building ID	3D Measurement (m)	2D Measurement (m)	Difference (m)
1	12.507777	13.125066	0.617289
2	13.713325	13.732419	0.019094
3	14.129196	14.155275	0.026079
4	16.866035	16.933496	0.067461
5	17.225675	17.833031	0.607356
6	18.953028	18.25633	–0.696698
7	18.758588	18.653542	–0.105046
8	19.039177	19.47366	0.434483
9	20.484453	19.157041	–1.327412
10	20.829912	20.002697	–0.827215
11	20.874955	21.008558	0.133603
12	20.955293	20.958611	0.003318
13	22.353933	22.861286	0.507353
14	22.754253	22.423482	–0.330771
15	22.520097	22.066294	–0.453803
16	23.235158	24.293483	1.058325
17	33.576494	31.089237	–2.487257
18	37.405119	36.844787	–0.560332
19	55.114582	54.690003	–0.424579
20	56.291389	56.197712	–0.093677

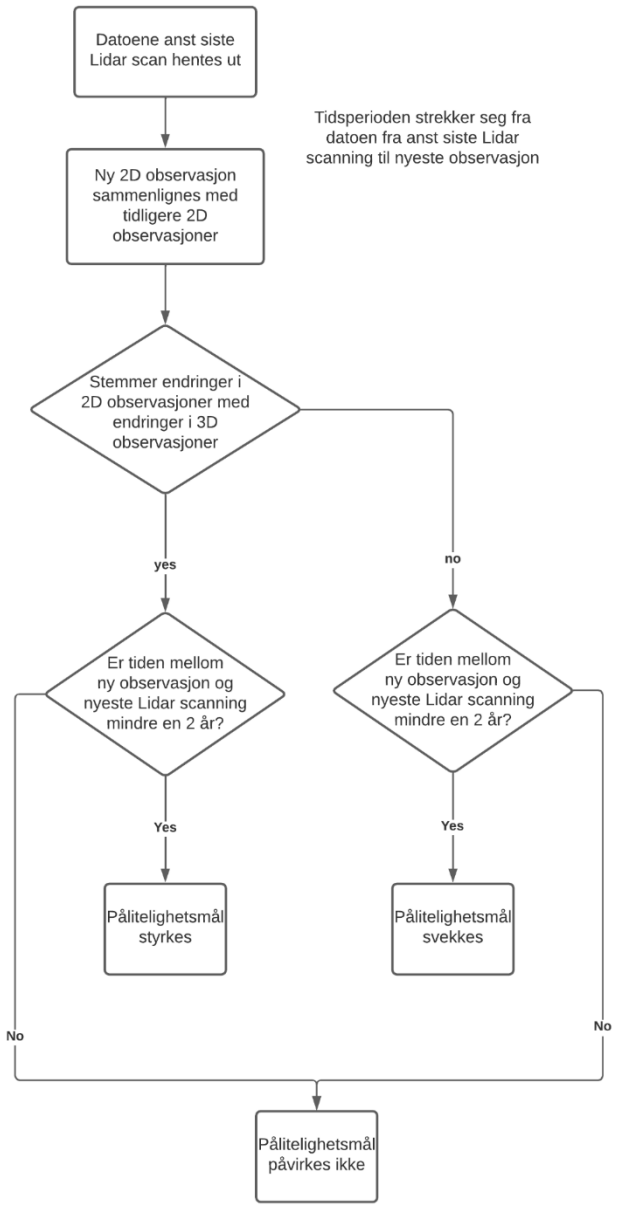
Tabell 4-1 - Sammenligninger av bygningshøyder detektert av lasermåling mot ortorektifisert data. av Liang et al, 2016, s. 3

4.3.3 *Endringer mot endringer*

En idé som kan løse noen utfordringer med å sammenligne 3D-objekter med 2D-objekter er å sammenligne endringer med endringer. Da tenkes det at 2D-objekter sammenlignes med 2D-objekter og 3D-objekter sammenlignes med 3D-objekter, og resultatene av disse sammenligningene vil da igjen sammenlignes. Dette vil resultere i at vi sammenligner forskjeller og likheter mellom datasett. På denne måten slipper vi å finne noe felles mellom et 2D- og 3D-objekt som kan sammenlignes. Figur 4.11 illustrerer vi ideen.

Ettersom LiDAR-skanninger i dag tas med et stort mellomrom på fem år og er mer detaljerte enn andre datakilder, så brukes tidsrommet mellom den nye observasjonen og den nest siste LiDAR skanning. Dette innebærer da at en kan få opp observasjoner fra seks – ca. femten år tilbake i tid. Underveis sammenlignes blant annet arealet hos 2D-observasjoner og volum hos 3D-observasjoner.

Forskjellene mellom 2D-objekter sammenlignes med de for 3D-objekter. Vi ser også på når sist LiDAR-skanning var som da kan ha oppstått mellom 1-5 år siden. Derfor vil det også være en faktor i sammenligningen å se på hvor nærme datoen er til den nye observasjonens dato. Her er den satt til to år. Dette er ganske enkelt bare en vekt vi har satt. Om det i fremtiden tas LiDAR-skanninger oftere, vil en kunne slippe å se langt tilbake i tid.



Figur 4-11 - Modell for analyse av LiDAR-observasjoner

4.4 Pålitelighetsfaktorer

Pålitelighetsfaktorer vil være verdier brukt i algoritmene som kan påvirke pålitelighetsmålet i en større eller mindre grad. Utfordringen her er hva disse pålitelighetsfaktorene skal baseres på. Det er tenkt at de kan baseres på flere kilder. Dette kan være avhengig av lokasjon og statistikk fra AI-modellene, samt kilder som SSB. Her produseres det ofte resultater fra en datakilde, hvilke datakilder som sammenlignes, hvilket tidsrom og mer. Vi vil i dette kapittelet komme med forslag til pålitelighetsfaktorer og hvordan de kan brukes.

4.4.1 *Utgangspunkt*

Før utregningen av pålitelighetsmålet begynner vil vi ha et utgangspunkt, eller en pålitelighetsscore som observasjoner starter med (også kalt initial pålitelighet). Dette utgangspunktet blir styrket eller svekket ettersom den blir manipulert av andre faktorer og utregninger under algoritmens prosess. Det er naturlig å først se på selve datakilden hvor det er flere forskjeller i detaljer. 3D-objekter inneholder detaljer fra flere vinkler, og ortofoto inneholder høydemål som blant annet kan hjelpe AI-modellene å se bort fra ting som badebasseng og lignende. I tillegg kan statistikk være med på å velge et utgangspunkt. Ved å ta alle nye bygg som er blitt registrert i for eksempel 2018 og dele dem på det totale antall eiendommer vil en komme frem til et tall som sier noe om hvor sannsynlig det er for at det oppstår et nytt bygg på en eiendom. Dette kommer litt an på dataen en har. SSB har en grei kalkulator for å se etter antall bygninger hvert år og filtrere etter bruksareal.

	Boliger (bebodde og ubebodde)						
	2015	2016	2017	2018	2019	2020	2021
K-4204 Kristiansand							
Under 30 kvm	1 666	1 862	1 891	1 941	1 925	1 962	2 023
30-39 kvm	811	874	884	908	907	945	989
40-49 kvm	1 678	1 742	1 808	1 823	1 857	1 885	1 949

Tabell 4-2 - Skjermdump av resultat fra statistikkbanken SSB.no. 2022. (https://www.ssb.no/statbank/table/06513/)

I Tabell 4.2 kan en se at det er 169 nye bygg som har 49 m² eller mindre bruksareal mellom 2020 og 2021. Deler en dette på antall eiendommer i Kristiansand får vi et veldig lite tall som kan være med å skape et mer realistisk utgangspunkt. Bruksarealet kan være fordelt på flere etasjer, noe AI-en ikke ser. Så her kan det muligens være greit å inkludere boliger med bruksareal opp til 100m².

Hver kommune har et forskjellig antall eiendommer og et forskjellig antall nye bygg. Derfor må en kunne ta inn informasjon om kommunen basert på hvor AI-resultatet kommer fra. Denne faktoren vil derfor bli dynamisk.

4.4.2 *AI-modellens mål*

Sammen med resultatene fra AI-modellen, følger et mål på hvor sikker AI-en er på en prediksjon. Semantisk segmentering er prosessen hvor hver piksel i et bilde blir tildelt en

klasse. For eksempel er AI-en trent opp til å kjenne igjen bygg, trær og biler. AI-modellen gjetter så hvilken klasse pikselen hører til ved å gi en score basert på hver klasse. Høyeste score avgjør hvilken klasse pikselen hører til. For å finne hvor sikker AI-en er på klassifiseringen av hele bildet, så summeres scoren fra alle pikslene i klassen som er blitt predikert og deretter deles det tallet på antall piksler (Kalderon, 2021). Dette er et mål som er forskjellig fra observasjon til observasjon og kan også brukes i utgangspunktet. Dette målet er ikke 100% troverdig i seg selv, men er avhengig av en gjennomsnittlig IoU-måling av AI-resultatene og andre faktorer som falske positiver og ikke-oppdagede-bygninger. Dette er fordi en kan ha en dårlig modell som er 100% sikker på at noe helt feil er et bygg.

4.4.3 *Vektlegging av pålitelighet per kilde*

Vektleggingen av pålitelighet er det tenkt å gjøre individuelt per kilde. Etter hvert vil verdiene bli dynamisk oppdatert, men det trengs et utgangspunkt. I Tabell 4.3 er det foreslått pålitelighetsfaktor per kilde. Disse er basert på samtaler med vår tekniske veileder, egne antagelser på hvordan resultatene kan være og hvor definerte kildene er. Matrikkel, saksbehandler og FKB er verdsatt ganske høyt, da de i de fleste tilfeller vil være kontrollert manuelt. Ortofoto får lavest utgangspunkt da det kommer fra en sammensatt mosaikk og har ganske få detaljer.

Kilde	**Pålitelighetsprosent**	**Pålitelighetsfaktor**
Matrikkel - (referansemål)	99%	0.99
Saksbehandler	95%	0.95
FKB	90%	0.9
Innbygger	90%	0.9
Laser	30%	0.3
Foto	25%	0.25
Ortofoto	20%	0.2

Tabell 4-3 - Kilders pålitelighet i utgangspunktet

Med utgangspunkt i de faktorene nevnt i Tabell 4.3, ønskes det å utvikle et system for å dynamisk kunne oppdatere disse verdiene. Hvordan dette kan gjøres vil diskuteres mer i «Kapittel 4.6 Veien videre».

4.5 Våre utprøvde algoritmer

I dette kapittelet vil vi oppsummere og fortelle om de algoritmene og scriptene vi har laget gjennom prosjektet som vi tenker er verdt å formidle om og utvikle videre. Vi har skapt mer avanserte algoritmer basert på BIoU og standardavvik, samt et mindre script med forslag til mindre prosesser som kan være med på å påvirke pålitelighetsmålet. All kildekode vil du finne i prosjektets repository: https://github.com/4bit-UiA/Bachelor-2022-KartAi.

4.5.1 *Endringer mot endringer*

Tidligere i kapittel 4.3.3 beskrev vi en idé hvor endringer sammenlignes mot endringer. I etterkant er det utviklet et script hvor vi har prøvd å få til en slik algoritme. Scriptet tar i bruk en liste over «observasjoner». Dette scriptet kan finnes i repositoriet under ‘src/endringer_mot_endringer.py’. Denne listen er en abstraksjon av observasjoner fra en database og holder på felt som «geometri». «datasett». «areal» og «volum». For at utregningen av pålitelighetsmål skal kjøre så må det være to LiDAR-skanninger her. Dette er fordi «Endringer mot endringer»-ideen går ut på at endringer i et dimensjonsformat (3D) sammenlignes med endringer i et annet format (2D). Tiden mellom nyeste observasjon og siste LiDAR-skanning tas også med i betraktning, ettersom det er en mulig blindsone her. I denne blindsonen kan det være flere observasjoner som ikke tas til betraktning i dette scriptet siden det er bare er disse to datasettene som tas med i prosessen. Gjennom scriptet er det satt noen faktorer som pålitelighetsmålet multipliseres med. Disse er vekter som er satt relativt vilkårlig for å ha noe å jobbe med. Dette er noe som tenkes kan settes bedre ved bruk av maskinlæring eller mer analyse av statistikk. I Figur 4.12 kan du se et utdrag fra scriptet hvor differansen i tid mellom nyeste observasjon og siste LiDAR-skanning påvirker hvor mye

pålitelighetsmålet skal forbedres.

```
# Om begge differ samsvarer, det vil si om begge har økt eller om begge har synket
if (diff_ny_observasjon_nr1_observasjon_2D > -0.2 and diff_lidar1_lidar2_3D > -0.5) or (
        diff_ny_observasjon_nr1_observasjon_2D < -0.2 and diff_lidar1_lidar2_3D < -0.5):
    # Hvor mye denne samsvaringen teller avgjøres av år mellom ny observasjon og sist 3D observasjon
    # 4 og over ignoreres
    match ny_observasjon['dato'].year - observasjoner_3D[0]['dato'].year:
        case 0:
            paalitelighetsmaal *= 1.3
        case 1:
            paalitelighetsmaal *= 1.2
        case 2:
            paalitelighetsmaal *= 1.1
        case 3:
            paalitelighetsmaal *= 1.05
```

Figur 4-12 - Kodeutsnitt av script for pålitelighetsmål

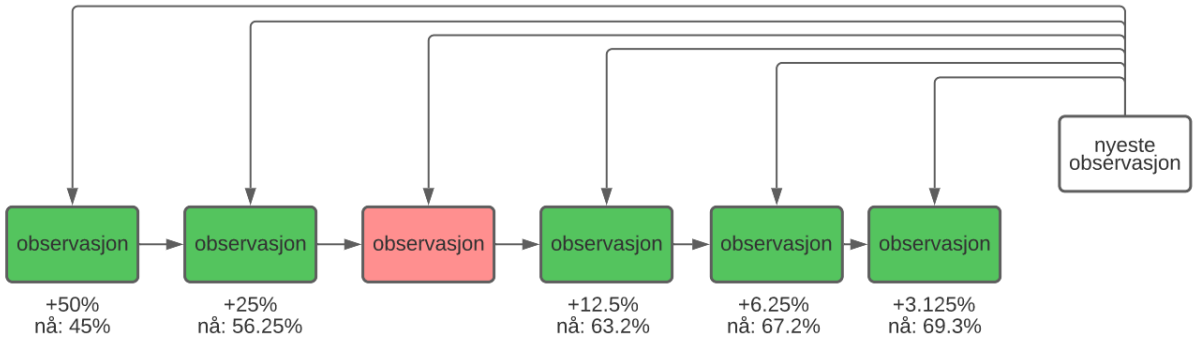
Figur 4.12 sammenligner forskjeller. Det vil si at det ses på sammenhengen mellom endringer i areal hos 2D objekt med endringer i volum hos 3D objekt. Så om begge endringer har enten økt eller sunket så blir pålitelighetscoren styrket. Om forskjellen i tid er over tre år, så blir sammenligningen ignorert.

Dette er ikke en veldig avansert utregning. Det tenkes at dette ikke kan brukes alene for å finne påliteligheten, men som en del av en større prosess. Denne modellen funker ikke i situasjoner hvor det bare eksisterer en observasjon i én dimensjon. Derfor tenkes det at dette kan være nyttig når et bygg allerede har eksistert i endel år. Under utvikling av denne ideen og scriptet var tanken å sammenligne endringer i 2D med endringer i 3D-objekter. Men i etterkant ser vi at det også kan benyttes mellom alle datakilder. At en bekreftelse fra en innbygger om at et bygg er endret, kan i et visst tidsrom også være med på å bekrefte en endring fra en annen datakilde til en viss grad.

4.5.2 *Endring mot tidligere observasjoner*

Vi har utviklet en modell som regner ut pålitelighet, gitt en rekke med «gyldige» observasjoner. Senere i kapittelet kommer vi tilbake til hvordan vi bestemmer hva som er

gyldige og ugyldige observasjoner. Modellen vises i Figur 4.13.



Figur 4-13 - Sammenligning av observasjoner

Når det kommer en ny observasjon brukes BIoU (nevnt i kapittel 4.2.3) for å sammenligne den med de gyldige tidligere observasjonene for det bygget. Innenfor dette har vi eksperimentert med flere forskjellige modelleringer av hvordan det kan regnes ut. Den mest gunstige har vært å øke pålitelighetsmålet med 50% av initial pålitelighet (initial er 30% i Figur 4.13, så da blir 30% + 50% = 45%) for den første observasjonen i rekka som er enig med nyeste observasjon. Så legges det til halvparten av forrige økning for hver ekstra observasjon som er enig. Når alle enige observasjoner i rekken er gått igjennom teller vi antall uenige observasjoner. I Figur 4.13 var det én av seks. Dette gjør vi til en brøk som skal trekkes i fra pålitelighetsmålet. Da vi var ferdige å telle antall enige observasjoner i figuren ble påliteligheten 69.3%, og en sjettedel av det er 11.55. Da blir regnestykket 69.3 - 11.55 = 57.75% endelig pålitelighet.

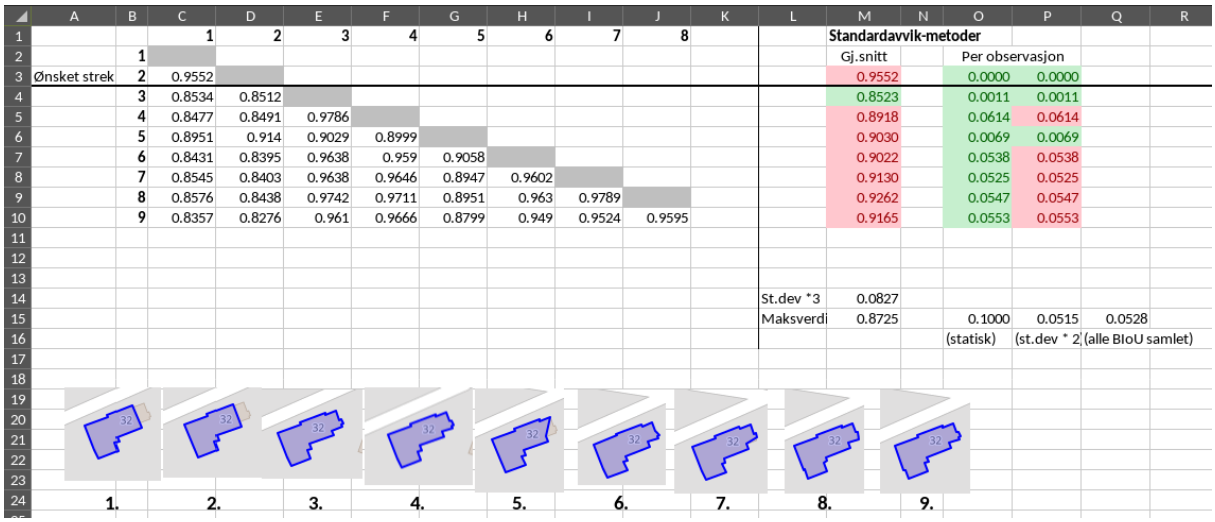
Pålitelighet per eldste enige observasjon

Den første modellen som ble utforsket her var å sammenligne BIoU for polygonene til den nyeste observasjonen med den eldste uverifiserte observasjonen, og jobbe seg fremover til den første som er enig. Hva som gjør at to observasjoner er enige må eksperimenteres med på mer data. På mockdataen som denne modellen analyserte fant vi at de er enige nok ved BIoU på 0.90. Denne verdien kan konfigureres. På Figur 4.13 er de enige observasjonene farget grønt.

BIoU - Standardavvik på gjennomsnitt

Videre ble det foreslått av vår tekniske veileder å undersøke om standardavvik kunne brukes for å definere gyldige observasjoner, som beskrevet tidligere. Dette ble modellert i regneark

basert på BIoU-målinger for mockdataens observasjoner. Her ble det brukt standardavvik på de gjennomsnittlige BIoU-målingene. Hvordan det gikk vises i kolonne M i Figur 4.14 og Figur 4.15. Overgangen fra rød til grønn celle viser hvor vi anser at endringen av polygonet har skjedd, som senere vil brukes til å kalkulere en pålitelighetsfaktor. På regnearket ser du også hvordan polygonene for den aktuelle «eiendommen» ser ut.

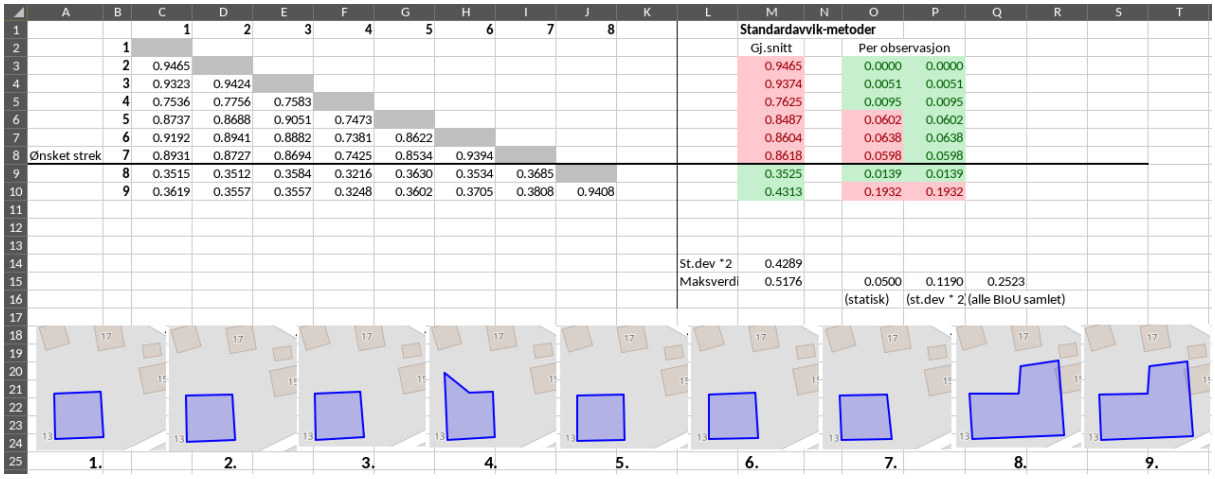


Figur 4-14 - Modell for sammenligning av observasjoner med standardavvik i regneark

BIoU - Standardavvik per observasjon

Ved presentering av den forrige modellen ble enda en ny fremgangsmåte foreslått, da bruk av gjennomsnitt før standardavvik tar vekk litt av poenget med standardavvik. Dette ble også modellert i regnearket i Figur 4.14. Først ved bruk av en forhåndssatt grense, hvor standardavvik må være over den grensa for å være en betydelig nok endring (Kolonne O). Det viste seg å ikke alltid fungere, så da ble det forsøkt å bruke standardavvik for å finne den grensa (Kolonne P). Vi har sett på standardavvik, og ser at det må brukes på minst tre verdier for at resultatet skal gi mening, så mockdataen vist i den tidligere nevnte figuren ble her litt mangelfull. Derfor ble det utarbeidet et annet sett med mockdata i Figur 4.15

Regnearket i sin helhet er lagt inn i modellerings-mappen i repository.

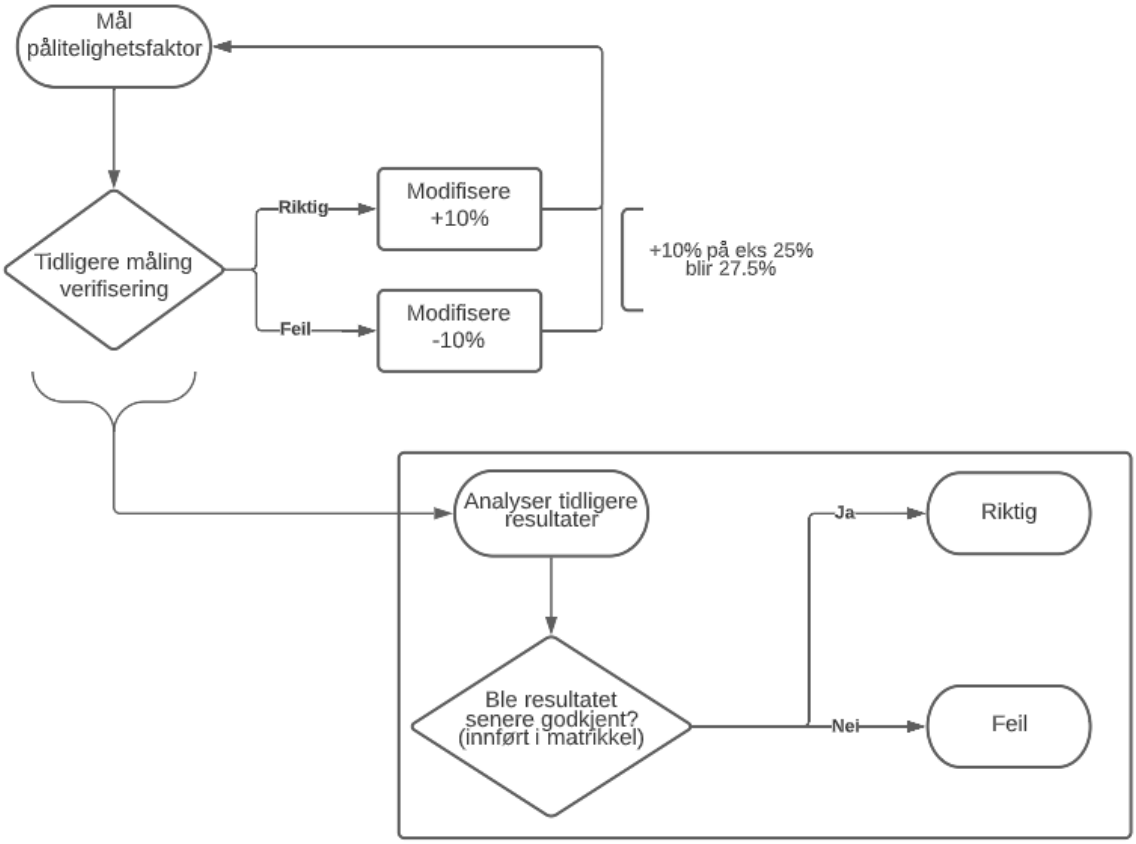


Figur 4-15 - Modell for sammenligning av observasjoner hvor datasettet har en feil i en observasjon

4.6 Veien videre

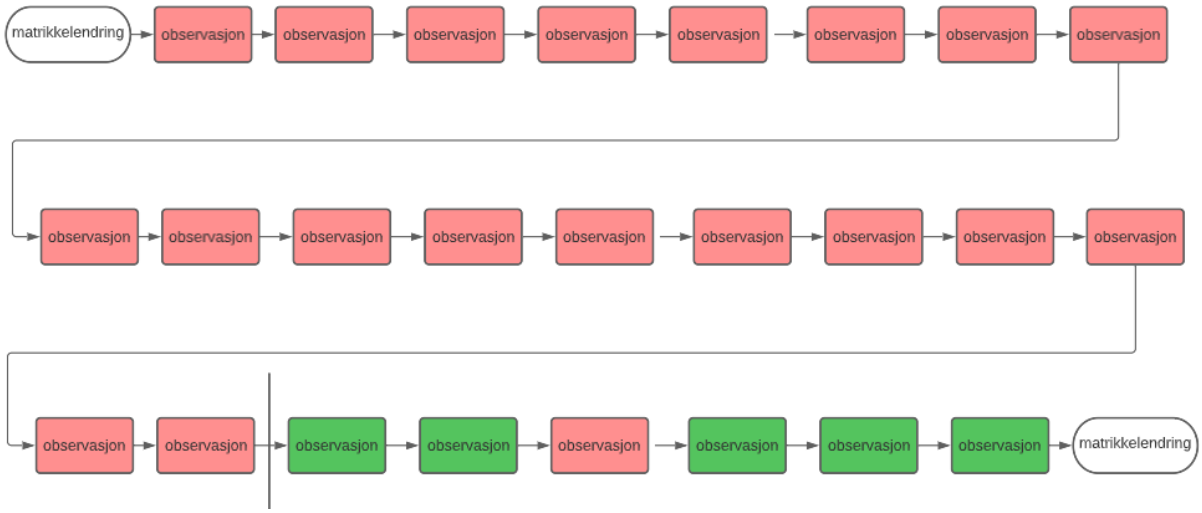
Pålitelighetsfaktoren ønskes å oppdateres i henhold til hvor sikker systemet kan være på at endringen stemmer. Si det viser seg at for eksempel alle resultatene fra AI på foto er feil i forhold til flere andre observasjoner. Da bør pålitelighetsfaktoren til flyfoto synke drastisk i forhold til resten av datakildene.

Dette kan gjøres ved å se på historikken til datakildens prediksjoner. For hver gang det skjer en matrikkelendring på grunnlag av AI-ens resultater kan alle observasjonene tilbake til forrige matrikkelendring bli sjekket for om de var riktige eller ikke. Denne dataen kan brukes til å finjustere den opprinnelige påliteligheten for hver datakilde. Figur 4-16 er vår foreslåtte modell til dette.



Figur 4-16 - Enkel modell for hvordan pålitelighet kan bli justert basert på tidligere observasjoner

Et problem vi imidlertid kan møte her er urettferdig justering av pålitelighet i tilfeller der det er lenge siden forrige matrikkelendring. I Figur 4-17 er det illustrert en kjede med observasjoner for et bygg, fra forrige matrikkelendring til nyeste matrikkelendring. Rød boks signaliserer at observasjonen ikke stemmer med den nye matrikkelendringen, grønn signaliserer det motsatte. Skal datakilden i dette tilfellet bli straffet like hardt for alle de røde observasjonene før «delestreken» på nederste rad, som for den røde observasjonen etter streken?



Figur 4-17 - Fiktiv obervasjonskjede til et bygg

For å løse dette problemet er en mulighet å si at jo nærmere observasjonen er matrikkelendringen, jo mer skal den påvirke påliteligheten til kilden om den er rett eller feil. En tanke er å kun ta observasjonene etter streken inn i betraktning. Dette er da gitt at streken settes ved eldste observasjon som er enig med den nye matrikkelendringen.

En annen faktor som kan være med på å bestemme hvor hardt observasjonen skal belønne eller straffe kildens pålitelighetsmål er å se på hvor sikker AI-en var på resultatet sitt. Om AI-en var veldig selvsikker på resultatet, og det viser seg å være feil, kan det tenkes at scoren burde reduseres med langt mer enn om AI-en var usikker.

Hvis vi sammenligner høydene fra et stort datasett i 2.5D og 3D kan vi finne standardavviket. Videre kan vi bruke dette til å vektlegge BIoU-målinger. Eksempelvis: om høydene samsvarer innenfor ett standardavvik så kan vi øke BIoU-scoren før vi regner pålitelighet. Om høydene er utenfor to standardavviksforskjell burde scoren fra BIoU bli redusert før den brukes til pålitelighetsmål. Dette gir en ny dimensjon for sammenligning av observasjoner på tvers av datakilder.

4.7 Konklusjon

For å konkludere rapporten går vi tilbake til problemstillingen som ble stilt ved begynnelsen.

> **Hvordan kan vi få pålitelighet til observasjoner fra KartAi-algoritmer basert på ulike datasett og ulike observasjonsalgoritmer?**

For å få en sikker pålitelighet trengs det en algoritme som er fleksibel og skalerbar nok til å ta imot og behandle store mengder historisk og fremtidig innkommende data fra varierende typer datakilder. Å sammenligne hvordan endringer samsvarer på tvers av datakilder er et effektivt verktøy, men som i seg selv er litt for enkelt å basere en hel algoritme rundt. Videre kan denne metodikken med fordel supplementere et mer utviklet system. I tillegg undersøkte vi sammenligning av 2.5D-og 3D-data som kan brukes til å vektlegge en GIoU-måling. Dette kan være en utrolig nyttig støttende faktor for å bestemme pålitelighet.

Vårt endelige forslag er en algoritme som ser på enighet blant observasjoner, utfører en endringsanalyse for å bestemme hvor endringen fant sted og gir en pålitelighetsscore basert på observasjonene i det relevante tidsrommet. Resultatene fra denne algoritmen gir veldig positive svar i våre teoretiske undersøkelser, men ettersom det dessverre ikke ble mulig å teste den på ekte data, er det enda litt arbeid igjen å gjøre her.

5 Refleksjon

I dette kapittelet reflekterer vi rundt arbeidet vi har gjort det siste halvåret. Her reflekteres det rundt valgene vi har tatt, ting vi ville ha forsøkt å gjøre annerledes, det ferdige «produktet» og utfordringer. Dette prosjektet har bydd på flere utfordringer og nye måter å gjøre ting på vi ikke har vært kjent med fra før. Generelt sett har denne opplevelsen vært svært lærerikt, til tider frustrerende og utfordrende.

5.1 Prosjektstyring

Som nevnt tidligere under kapittelet «Sentrale vedtak» så har vi hatt utfordringer med arbeidsmetodikken. Vi satset på Scrum ettersom det var det vi var mest kjent med, men møtte på utfordringer etter hvert som det ikke passet sammen med oppgavens gjøremål. Vi tilpasset så arbeidsmetodikken vår slik den ble mer fleksibel, men vi har følt på en viss usikkerhet gjennom prosjektet likevel. Dette er mye mulig på grunn av prosjektstyringsverktøyet «Azure DevOps» vi valgte for å holde styr på backlog og repository. I Azure DevOps opprettes det brukerhistorier som blir oppdelt i oppgaver, noe som ikke passet helt til de mer eksperimentelle oppgavene vi har holdt på med. Vi har ikke hatt noen brukerhistorier.

I etterkant tenker vi at det hadde passet bedre å bruke GitHubs funksjon «Project Boards». Project Boards tilbyr et enkelt brukergrensesnitt med Kanban-vegg hvor oppgaver kan kobles til branches. Azure DevOps er et fantastisk verktøy, men har vært litt for overdrevet for vårt prosjekt. Vi har også tenkt i etterkant at det hadde vært lurt å ha hørt med andre som jobber i KartAi om hvordan de styrer prosjekter. Primært gruppen som jobber med å utvikle og teste nye AI modeller. For å finne den beste modellen, så må ai-gruppen teste ut flere modeller. Dette kan være ferdige modeller, videreutviklet modeller og hyllevare. Mye lignende hva vi har gjort gjennom prosjektet, eksperimentering og forskning på forskjellige teknikker. Tanken om å bruke GitHubs funksjon Project Board kom også etter å ha sett gjennom GitHub repositoriet til AI teamet. Å få fysisk kontorplass har også vært en utfordring for arbeidet og noe vi etterspurte tidlig i prosjektet. COVID-restriksjonene ble opphevet i starten av prosjektet og bedriften hadde egne utfordringen med å selv komme tilbake til kontoret så kontorplass for oss tok forståelig nok lang tid. Å ha en fast, fysisk møteplass er et viktig element for produktiviteten og prosjektstyringen.

Vi konkluderer her med at en tilpasset agil arbeidsmetodikk har funket gjennom prosjektet, men at vi kunne spart oss tid og krefter om vi hadde hørt rundt om hva som hadde passet bedre til et slikt prosjekt som det vi har gjennomført.

5.2 Kvalitetssikring

Et av de viktigste delene av å kvalitetssikre et prosjekt er å sørge for å ha en konsis liste med spesifikke krav fra produkteier (Startup Development House, 2020). For dette prosjektet har det som nevnt vært utfordrende ettersom kravene nettopp er såpass åpne. Med slike åpne krav la teamet stor vekt på jevnlig kommunikasjon med bedriften og andre aktuelle samarbeidspartnere innad i KartAi-prosjektet. Jevnlige statusmøter med produkteier var satt opp fra starten av hver andre uke og teamet la stor vekt på å kunne få fysisk kontorplass. Dette tok dessverre lengre tid enn ønsket, men i tiden før kontorplassen ble ordnet holdt teamet jevnlig kontakt via epost og andre digitale kommunikasjonsmidler. Etter å ha fått kontorplassen ble den mye brukt for å få tilbakemelding på flere deler av prosjektet. Spesielt på de mer tekniske og matematiske delene av algoritmen som var rimelig nytt for teamet. Å kunne få jevnlig tilbakemelding fra vår fantastiske samarbeidspartner i Kartverket har gjort at flere uønskede aspekter og mangler i algoritmen ble identifisert veldig fort. Kommunikasjon med veileder har også vært et viktig fokus for teamet og jevnlige møter med veileder ble også

satt opp tidlig i prosjektet. Spesielt i forhold til rapportstruktur og viktige temaer å få med har veileder vært til stor hjelp i å sikre kvaliteten på selve rapporten.

Gjennom den agile metodikken som er brukt i prosjektet kommer det naturlig inn flere kvalitetssikringselementer. Daily Scrums er en aktivitet som er blitt tatt veldig seriøst blant medlemmene for å sikre at fokuset på bacheloren blir holdt stødig. Dette hjalp veldig på moralen i starten av prosjektet når teamet slet litt med hva som faktisk skulle gjøres. Daglige møter om bachelorrapporten har sørget for at fokuset ikke forsvinner i en hektisk studiehverdag hvor tiden fort går vekk til andre fag. I tillegg lar det alle de andre medlemmene se hva alle jobber med og gjør at feil eller mangler raskt kan bli plukket opp. Sprint planning og sprint-strukturen som kommer med agil prosjektstyring har hjulpet å holde oversikten og som en påminner for teamet om å alltid planlegge framover for å sikre framgang.

5.3 Resultatene

Vårt prosjekt har siden starten av vært et forskningsprosjekt. Forventningene fra produkteier var at vi skulle dokumentere funn av teknikker og teknologier, hva som kan fungere videre og hva som ikke har funket så bra. Det var derimot ikke forventet av produkteier at prosjektet ville resultere i et ferdig produkt. Og det er hva prosjektet vårt har resultert med, en ferdig rapport hvor vi har dokumentert ned om forskjellige teknologier, interessante teknikker og om koden som er blitt utviklet som følge av forskningen vi har vært gjennom. Resultatene våre vil ligge som grunnlag for videre utvikling av pålitelighetsmålet, og vi håper det er av god nytte. Flere ting har muligens påvirket hvordan resultatene våre har endt opp. Som for eksempel prosjektstyringen som er nevnt i starten av kapittelet. At dette kan ha hindret mye fremgang tidlig i prosjektet, kombinert med at vi var usikre på hvordan vi skulle gå frem med et slikt prosjekt. Vi er og en gruppe med mindre mattekunnskaper, og tenker at riktig kunnskap innenfor statistikk og geometri kunne ha bidratt til enda bedre resultater. Men vi føler vi har bidratt godt og fått til noen interessante resultater.

5.3.1 *Algoritmenes funksjon*

Det vil ikke komme som noen overraskelse at algoritmene er uferdige. Det vi har fått til med dem er å vise mulige fremgangsmåter for å kunne gi en pålitelighet for observasjoner. Den

algoritmen vi hadde mest suksess med var å sammenligne den nyeste observasjonen med den nyeste bekreftede observasjonen og jobbe seg fremover i tid til en observasjonene var enige om at det var gjort en endring. Påliteligheten blir i den modellen regnet ut på alle observasjonene imellom de to observasjonene.

Dessverre har vi ikke fått muligheten til å teste disse algoritmene og modellene på data fra KartAi, så det fungerer bare på et teoretisk nivå. Vi får et resultat på hvor pålitelige vi kan tenke at observasjoner er. Hadde vi fått testet mot reell data så kunne vi fått tilpasset funksjon og diverse grenser til å passe bedre.

6 Selvevaluering

I dette kapittelet vil hvert gruppemedlem si noe om hvordan de deltok i prosjektet.

6.1 Bendix

Min rolle i dette prosjektet har vært mest på den teoretiske siden med rapportskriving, kommunikasjon og litteraturanalyse. Jeg har skrevet en del om det meste i hele rapporten og har fungert litt som altmuligmann. I våre møter og beslutningsprosesser har jeg deltatt aktivt og jeg har hatt ansvaret for alt av kommunikasjon mellom vårt team og bedrift, universitet og andre samarbeidspartnere. Ettersom jeg var ute i praksis på KartAi-prosjektet for Norkart på høsten 2021, var det veldig greit for meg å ta dette ansvaret. Produkteier var til og med den samme så jeg hadde alt av kontaktinformasjon fra det øyeblikket vi begynte prosjektet i januar. Jeg synes mitt bidrag har vært til god nytte, selv om jeg nok er den med minst teknisk kompetanse på vårt team. Det har absolutt ikke vært mangel på ting å gjøre, men når jeg ser tilbake skulle jeg ønske jeg hadde hatt en litt mer teknisk del på noen deler av prosjektet. En slik erfaring tenker jeg er verdifull å ha med, men prosjektets veldig høy-tekniske natur gjorde det vanskelig og jeg endte opp med en mer overordnet rolle. Samarbeidet innad i teamet har vært helt strålende, men ettersom alle i teamet kjente godt til hverandre fra før var ikke det noen overraskelse.

6.2 Eirik

Dette har vært en veldig interessant oppgave å jobbe med. I starten av prosjektet var oppgaven veldig abstrakt og vanskelig å sette seg godt inn i. Oppgavene da var typisk å finne ut av hvordan vi kunne arbeide med oppgaven. I denne perioden erfarte jeg hvor viktig en god plan for gjennomføring kan være. Hadde det ikke vært for en god prosjektstyring så ville gjennomføringen gått mye tregere. Da vi kom litt lengre i prosjektet, og mer konkrete oppgaver manifesterte seg, spisset jeg fokuset mitt på tekniske oppgaver som arbeid med algoritmene, oppsett av miljø, strukturering av rapport osv. Det å par-programmere erfarte jeg som veldig nyttig, spesielt når det handler om å eksperimentere. Hovedsakelig par-programmerte jeg med Thomas, men vi gikk gjennom kode (code-review) med hele gruppen innimellom.

Før prosjektet var jeg relativt ny med å programmere i Python, men da jeg har endel erfaring fra tidligere kode-prosjekter ser det ut som jeg plukket det opp fort. Da vi begynte å lage algoritmer for å regne ut GIoU og BIoU var det veldig praktisk med tidligere erfaring. Disse to konseptene fant vi ingen biblioteker som kunne hjelpe oss med, så da måtte disse lages manuelt. Denne fasen av prosjektet fant jeg spesielt lærerik og motiverende, da det viste hvor ukjent terrenget vi jobbet i var.

Gruppearbeidet syns jeg har vært strålende, og vi har alle fått gitt av det vi kan. Og selvfølgelig lært mye nytt!

6.3 Johannes

I dette prosjektet har jeg jobbet mye med forskning og eksperimentering samtidig som jeg har vært «Scrum» - master. Jeg har brukt mye av tiden min til å se på PostGIS og hva som er mulig her med forskjellige geometrier. Jeg utviklet ideen om Endringer mot endringer og har ellers vært med på å dokumentere funn, skrive i rapporten og ført presentasjoner ved statusoppdateringer hos bedriften og UiA. I tillegg til dette har jeg som resten av teamet satt meg inn i Python, noe jeg tenker kan være nyttig i jobblivet. Som Scrum-master har jeg holdt Daily Standups fra starten, satt opp Azure DevOps og sørget for at estimering, timeføring og andre elementer er fulgt og gjort av resten av teamet. Jeg konkluderer med at jeg føler at jeg har bidratt godt og mye for å gjennomføre dette prosjektet med kvalitet. Om det er noe jeg syntes jeg kunne har gjort bedre så hadde det vært å ta mer initiativ til å arrangere sosiale

sammenkomster med gruppen. Noe jeg tenker kunne ha styrket teamspiriten. Men jeg er likevel glad for at gruppen har forblitt venner gjennom hele prosjektet!

6.4 Thomas

I starten av prosjektet arbeidet jeg mye med research. Da lærte jeg om temaer som maskinlæring, Python og geografiske informasjonssystemer. I denne prosessen oppdaget jeg sammenligningsmetodene Generalized Intersection over Union og Boundary Intersection over Union. Dette videreførte vi i algoritmene som ble skapt. Jeg lagde modellen for å opprette pålitelighet basert på endringer mot tidligere observasjoner ved hjelp av eldste enige observasjon. Dette utviklet Eirik og meg algoritmen for. Da dette var gjort fikk vi tips fra teknisk veileder i Kartverket om å ta i bruk standardavvik i koden vår. Ut fra dette utviklet Eirik og meg nye algoritmer. Utover dette har jeg skrevet på rapporten og dokumentert arbeidet. Dette prosjektet har jeg lært mye av, og jeg fikk særlig stort utbytte av parprogrammering. Mine bidrag førte til store deler av utviklingen, og jeg er godt fornøyd med utfallet.

7 Referanser

Agile Alliance. (u.å.). *What is Kanban?* Hentet 6. april 2022 fra https://www.agilealliance.org/glossary/kanban/

Aladon. (2018, 25. oktober). *Knowing the Difference Between Criticality, Consequence and Risk Can Lead to Better Business Outcomes and Avert Disaster.* https://www.aladon.com/criticality-consequence-and-risk/

Atlassian. (u.å.). *Why Git: Atlassian Git Tutorial*. Hentet 11. april 2022 fra https://www.atlassian.com/git/tutorials/why-git

Chapman, Z. (2018, 09. august). *Social Loafing in Scrum.* ClearlyAgile. https://www.clearlyagile.com/agile-blog/social-loafing-in-scrum

Cheng, B., Girshick, R. B., Dollár, P., Berg, A. & Kirillov, A. (2021, 30. mars). Boundary IoU: Improving Object-Centric Image Segmentation Evaluation. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).* pp. 15329-15337. https://doi.org/10.1109/CVPR46437.2021.01508

Cohn, M. (u.å.). *Sprint Retrospective.* Mountain Goat Software. Hentet 08. april 2022 fra https://www.mountaingoatsoftware.com/agile/scrum/meetings/sprint-retrospective

Coursera. (2022, 21. mars). *What is python used for? A beginner's guide*. https://www.coursera.org/articles/what-is-python-used-for-a-beginners-guide-to-using-python

Fowler, M. (2005, 12. desember). *Event sourcing*. martinFowler.com. https://martinfowler.com/eaaDev/EventSourcing.html

GeoPandas. (u.å.). About GeoPandas. Hentet 15. mai 2022 fra https://geopandas.org/en/stable/about.html

JetBrains. (u.å.). *Python programming - the state of developer ecosystem in 2021 infographic*. Hentet 11. april 2022 fra https://www.jetbrains.com/lp/devecosystem-2021/python/

Kalderon, S. (2021, 21. juli). *A simple guide to semantic segmentation*. BeyondMinds. https://beyondminds.ai/blog/a-simple-guide-to-semantic-segmentation/

KartAi. (u.å.). *Rapporter og resultater*. KartAi. Hentet 04. mai 2022 fra https://kartai.no/rapporter-og-resultater/

Kartverket. (2021, 11. september). *Ajourføring av byggtema.* https://www.kartverket.no/geodataarbeid/forvaltning-drift-og-vedlikehold/byggtema

Liang, J., Gong, J., Liu, J., Zou, Y., Zhang, J., Sun, J. & Chen, S. (2016, 11. november). Generating Orthorectified Multi-Perspective 2.5D Maps to Facilitate Web GIS-Based Visualization and Exploitation of Massive 3D City Models. *ISPRS International Journal of Geo-Information*. https://doi.org/10.3390/ijgi5110212

Mæhlum, L. (2021, 6. juli). *ortofoto.* I Store norske leksikon. https://snl.no/ortofoto

Martins, J. (2021. 16. april). *Why social loafing is more about clarity than productivity.* asana. https://asana.com/resources/social-loafing

Mathplotlib (u.å.) *Matplotlib: Visualization with Python* Mathplotlib Hentet 15. mai 2022 fra https://matplotlib.org/

Microsoft. (u.å.). *Anbefalte Fremgangs måter for prosjekt kommunikasjon i Project online*. Microsoft Support. Hentet 15. mai 2022, fra https://support.microsoft.com/nb-no/office/anbefalte-fremgangs-m%C3%A5ter-for-prosjekt-kommunikasjon-i-project-online-cbac57bb-3420-4108-875f-8ba8ddbfed5e

Norkart. (2021, 11. juni). *KartAi - prosjektseminar* [Video]. Vimeo. https://vimeo.com/norkart/review/561731941/877b458124

Norkart. (u.å.). *Om oss.* Hentet 8. mai 2022 fra https://www.norkart.no/om-oss/

Ørstavik, E. & Mæhlum, L. (2020, 23. november). *Geografisk informasjonssystem – GIS*. Store norske leksikon. https://snl.no/geografisk_informasjonssystem_-_GIS

Peterson, M., Ørstavik, M., Aasgaard, R., Pacivic, S., Mjaaland, S., Dalsgård, A. S., Rostad, S., Groesz, F., Overland, I., Malmquist, C., Tachon, M., Høksaas, E., Palkhanov, I., Økland, S., Berge, A. E. & Nenseth, M. (2021, 15. desember). Rapport – KartAI, arbeidspakke 2. Bygningsidentifikasjon. *KartAi*. https://kartai.no/wp-content/uploads/2022/02/AP2-Rapport-Byggidentifikasjon-KartAI-17.12.2021-1.pdf

Pihl, R., & Solberg, B. (2021, 5. mars). *Flyfoto*. I Store norske leksikon. https://snl.no/flyfoto

PostGIS. (u.å). *About PostGIS*. Hentet 12. april 2022 fra https://PostGIS.net/

Rezatofighi, S. H., Tsoi, N., Gwak, J., Sadeghian, A., Reid I. & Savarese, S. (2019, 25. februar). Generalized Intersection Over Union: A Metric and a Loss for Bounding Box Regression. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. pp. 658-666. https://doi.org/10.1109/CVPR.2019.00075

Scrum.org. (u.å.). *What is a Sprint Review?* Hentet 8. april 2022 fra https://www.scrum.org/resources/what-is-a-sprint-review

Shapely. (u.å.). *Shapely.* Hentet 14. mai 2022 fra https://shapely.readthedocs.io/en/maint-1.8/

SonarLint. (u.å.). *Free and open source code quality & security ide extension*. Hentet 25. april 2022 fra https://www.sonarlint.org/

Startup Development House. (2020, 13. januar). *What is Quality Assurance in IT?* https://start-up.house/en/blog/articles/quality-assurance-it

West, D. (u.å.). *Sprint Planning.* Atlassian. Hentet 8. april 2022 fra https://www.atlassian.com/agile/scrum/sprint-planning

Xu, J., Ma, Y., He, S., & Zhu, J. (2019). *3D-GIoU: 3D Generalized Intersection over Union for Object Detection in Point Cloud.* Sensors (Basel, Switzerland), 19(19), 4093. https://doi.org/10.3390/s19194093

Appendix

A - Uttalelse fra produkteier

B - Gruppekontrakt

C - Referat fra Styregruppemøte 1

D - Referat fra Styregruppemøte 2

E - Modeller første runde

F - Modeller andre runde

A - Uttalelse fra produkteier

Gruppa har hatt et eksemplarisk samarbeid med KartAi-prosjektet og samarbeidspartnerne. De har vist at de tilegner seg kunnskap raskt og har en spesielt god evne til hurtig problemløsning gjennom praktisk programmering og prototyping. Ideen som kommer opp i møter er dere i stand til å sette ut i live umiddelbart. Dette har gjort at ideen og konsepter raskt kan testes ut. En av årsakene til dette er et dere har laget et robust rammeverk i bunnen som enkelt lar seg endre. Det vitner om gode evner i å tenke software-design og god arkitektur.

Problemstillingene i KartAi-prosjektet er komplekse og utfordrer på et høyt forskningsmessig nivå. Det er derfor gledelig at gruppa raskt forstår problemstillingene i deres oppgave og utvikler konseptuelle prinsipper som er direkte gode resultater inn i fremtidens nasjonale kartleggingsprinsipper.

Gruppa er en sosialt hyggelig gruppe, Kartverket, Norkart og KartAi-prosjektet er strålende fornøyd med både den faglige og sosiale aspektet i samarbeidet.

Vennlig hilsen
Alexander Salveson Nossum
Innovasjon- og digitaliseringsansvarlig
Norkart Datavarehus
alexander.nossum@norkart.no
+47 41293632

B - Gruppekontrakt

Gruppekontrakt IS-304: Bacheloroppgave i informasjonssystemer

Gruppepnavn: 4Bit

Gruppens deltakere:

- Johannes Birkeland
  - Tlf: 906 29 261
  - Email: johannesbi@uia.no
- Thomas Låg Hjelleseth
  - Tlf: 954 64 663
  - Email: thomas.laag@gmail.com
- Eirik Thilesen Svagård
  - Tlf: 480 68 822
  - Email: eirik@svagard.no
- Bendix Melkeraaen Vagle
  - Tlf: 413 94 526
  - Email: bendixmv@uia.no

Skal jobbe med følgende prosjekt

IS-304 Bacheloroppgave i informasjonssystemer
Bachelorprosjekt hos bedriften Norkart. Prosjektets varighet er hele semesteret, samt forberedelser for dette.

Mål

Gruppen ønsker å oppnå en høy karakter i dette emnet, fortrinnsvis en A, samt tilegne seg teoretisk og praktisk kunnskap og kompetanse innenfor arbeid med digitalisering og teknologi i et reelt arbeidsmiljø hos Norkart.

Rollefordeling

Kontaktperson: Bendix Vagle
Scrum master: Johannes Birkeland

Samarbeidsverktøy

Discord, Microsoft Teams, Versjonskontroll-Systemer, og eventuelle andre verktøy som blir introdusert etterhvert.

Arbeidsplan

Gruppen er enig om å ha et Daily Scrum-møte klokken 12:00 alle ukedager mandag-fredag hvorav helgene i utgangspunktet er fritid med mindre en innlevering må prioriteres.

Regler

1. Alle på gruppen har plikt til å holde seg oppdatert på valgte kommunikasjonskanaler.
2. Hvert medlem er pliktig til å møte opp på alle fastsatte gruppemøter. Om noe skulle inntreffe må medlemmet si ifra så snart som mulig og ikke senere enn to dager før om dette lar seg gjøre.
3. Tid brukt på prosjektet skal føres i excelark for timeføring og i Azure Devops hver dag.
4. Brytes reglene vil det aktuelle medlemmet få en strike. Ved syv strikes må medlemmet forlate gruppen.
5. Brytes plikten ved å holde seg oppdatert, føre timer eller møte ved faste møter vil medlemmet få en varsel første gang. Ved neste gang får den en strike.

Sted, dato: Kristiansand, 08.03.2022

Underskrifter:

Birkeland, Johannes

Hjelleseth, Thomas Låg

Svagård, Eirik Thilesen

Vagle, Bendix Melkeraaen

C - Referat fra Styregruppemøte 1

Møte - Styregruppemøte 1

Møtested: *Digitalt på Teams*
Møtetidspunkt: *23. februar 2022 - 10:00*

Tilstede:

- Birkeland, Johannes (Bachelorgruppe)
- Hjelleseth, Thomas L. (Bachelorgruppe)
- Svagård, Eirik T. (Bachelorgruppe, referent)
- Vagle, Bendix M. (Bachelorgruppe)
- Herrera, Lucia C. (UIA Veileder)
- Nossum, Alexander S. (Norkart)

1. Velkomst og introduksjon

2. Gruppen presenterer status

- Introduksjon av vårt arbeid til nå

Bendix viser fram backloggen. Der er det ekspermient-oppgaver for alle sammen, hvor forskjellige oppgaver er lagt inn med subtasks. Vi har eksperimentert litt hver for oss. Ivar har etterspurt mer rapportskriving, så vi har lagt inn punkter for det også.

Det vises også frem timeføring-dokumentet.
Dette er en oversikt over mer overordnet tidsbruk. Her er det ikke spesifikt på oppgaver.

Viser rapport-disposisjonen. Foreløpig er strukturen slik:

---

1. Innledning
2. Begreper
3. Analyse og modellering
   3.1 Analyse
   3.2 Modellering
4. Sammenligning av datasett
   4.1 3D - 2D
   4.2 Flater (Foto, ortofoto, FKB)
   4.3 Innbygger - målinger
   4.4 Hvor sikker er observasjonene?
   4.5 Hvor langt bak i tid skal en ny observasjon sammenlignes med eldre?
   4.6 Koding (?)

5. Pålitelighetsfaktorer
   5.1 Vektlegging av pålitelighet per kilde
   5.2 Fremgangsmåte
   5.3 ...
6. Våre utprøvde algoritmer
   6.x ...
7. Konklusjon
8. Referanser

Vi har kommet greit igang med kapittel 3 og 4, og noe på kapittel 5 og 2

Risikoanalysen vises. De høyeste risikoene er

- sosial loffing. Dette vil eventuelt oppdages fort, gitt timeføring.
- Manglende samarbeid med bedriften. Til nå har vi hatt veldig godt samarbeid.

- Demonstrasjon av program for sammenligning av dataset

Johannes viser.
Det er laget et lite program i python. Programmet sammenligner punkter og polygoner med hverandre.
Observasjoner er laget basert på ét bygg, og det er lagt inn litt forskyvninger og endringer i flatene.
Programmet løper gjennom hver observasjon og sammenligner flatene eller punktene og gir resultat på forskjell i kvadratmeter.
Etter dette programmet er det sett på funksjoner som også sammenligner 3D. Det skal vi se videre på, samt å eksperimentere i PostGIS.

- Plan fremover

Thomas forteller.
Fokuset videre skal være på rapportskriving. Vi lager en rapport til KartAI, og kan gjenbruke mye av den rapporten til UiAs rapport.
I neste sprint skal vi jobbe videre med det.
Vi har også hatt møte med master-studentene Jørgen og Sander, som skal sende oss hva de har av data. Der kan vi se hvordan filer og observasjoner er satt opp, som vil gi oss en bedre forståelse for oppbyggingen av vårt prosjekt.

3. Diskusjon og tilbakemelding

[Alexander]
Dere får bra resultater her. Det dere gjør er å utvikle algoritmen som gir et pålitelighetsmål. Den består av mange ulike deler. Det å beskrive algoritmen i psuedo-

kode vil være en styrke i rapporten, og for de som ikke forstår python (eller GIS). Beskriv det som utvikles med psuedokode.

For diff-ingen mellom polygoner ("Hvor forskjellige er byggene"-algoritmer), der kom jeg på at her bør dere ta en rask sparring med Atle. Han har skrevet en doktorgrad hvor en stor del av den var diffing mellom geometri. Her kan dere hente inspirasjon og se om det er noe å bygge på, evt. ta det med inn. Han jobbet også med PostGIS og Python.
PostGIS - jeg er veldig på det verktøyet. Utforsk gjerne de mulighetene som er der. Se også GeoPandas og Shapely (kanskje allerede i GeoPandas?).
Det finnes også QGis software. Det finnes også som et Python-bibliotek. Kanskje det er funskjoner der dere kan bruke inn mot algoritmen deres?

Det er gøy å se at dere kommer videre her!

[Bendix]
Sender du rapporten hans, eller bør vi snakke med han?

[Alexander]
Dere bør nok helst snakke med han. Sender noe info:

- Diff mellom polygoner: Sparre med Atle Frenvik Sveen

- https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2723138

[Lucia]
Veldig imponert over at dere har gjort så mye på kort tid. Veldig glad for at dere har begynt å skrive.
Skal dere skrive to rapporter? En til KartAI og en til UiA?
[Bendix]
De vil bli ganske like, men den ene vil bli modifisert litt for å passe bedre.
[Luica]
Hvis de blir mer ulike kan den ene rapporten legges inn som Annex i Bachelorrapporten.

[Alexander]
Kan hende Ivar har sagt noe annet, men ikke bruk masse tid på å skrive to rapporter. Hvis den kan gjenbrukes for en bacheloroppgave er det helt fint for kartAI. Har Ivar sagt noe annet?
[Bendix]
Nei, ikke egentlig.
[Alexander]
Forslag: Skriv bacheloroppgavens deres, (dere vil få A, vel?). Det som er KartAI spesifikt, eller tekst-informasjon som er aktuelt for KartAI, bare legg det i Annex på bachelorrapporten. Det er helt fint det.
Det har aldri vært et studentprosjekt som har litt god nok tid til å skrive flere rapporter.

Plutselig kommer det masse greier, og så sitter man og har dårlig tid. Greit å minske volum av arbeid.

[Johannes]
Vet du (Alexander) om alle dagens observasjoner blir lagret i PostGIS?
[Alexander]
Det skal de, ja. De som jobber med dette skriver master-oppgave og jobber deltid i prosjektet samtidig. De jobber med å generere data og setter dette inn i postgis (tror jeg). Det å få det inn i PostGIS skal være en del av jobben dems.
Ellers må det være kjapt å sette inn i Postgis. For å importere ting kjapt kan man bruke ogr-ogr(?) verktøy. Det finnes ganske mange fine verktøy på det. Legger ved noen linker.

- https://gdal.org/

[Bendix]
Tilgang til FKB-databasen? Er det noe vi har muligheten til å få?
[Alexander] Jaaaa, det vil
Jeg skal spore opp litt tilganger. Det er en Azure-base. Det ligger der men jeg er litt usikker på hvem som kan ha tilgangen til det.
Det er et viktig læringsmål mot arbeidslivet; tilganger krever gjerne 20-30%. Det tar mye tid.

4. Avslutning

[Alexander]
Hvordan føler dere det går?
[Bendix]
Vi er greit igang nå, og det går greit. Klare for neste sprint.

[Lucia]
Når dere har noe å vise, så er det bare å si ifra. Så kan jeg kommentere direkte i rapporten.
[Bendix]
Det er kanskje litt tidlig å sende noe over nå..

[Alexander]
Bare send ting fortløpende til veileder (og gjerne til Ivar og meg også). Tenk litt på tids-bruk; spør enkle spørsmål "ser disposisjonen grei ut?". Send konkrete spørsmål, bruk gjerne utkast fra tekst. Da vil alle spare tid
Greit å slippe å måtte gi tilbakemeldinger på 50 sider på en gang. Bruk veileder, meg og Ivar best mulig til dette.

D - Referat fra Styregruppemøte 2

Møte - Styregruppemøte 2

Møtested: *Digitalt på Teams*
Møtetidspunkt: *20. april 2022 - 10:00*

Tilstede:

- Birkeland, Johannes (Bachelorgruppe)
- Hjelleseth, Thomas L. (Bachelorgruppe)
- Svagård, Eirik T. (Bachelorgruppe, referent)
- Vagle, Bendix M. (Bachelorgruppe)
- Herrera, Lucia C. (UIA Veileder)
- Oveland, Ivar (Kartverket)

1. Velkomst og introduksjon

2. Gruppen presenterer status

Rapporten

Vi har noe nytt i rapporten. Vi har sett endel på sammenligning av datasett.
Sammenligning av 2D objekter. Sener gikk vi over på funksjoner i PostGIS - mye ferdiglagde funksjoner.
Det er dokumentert så godt vi kan.
Det er jobbet med algoritmer som blant annet benytter GIoU og BIoU.
GIoU - Generalised Intersect over Union
BIoU - Boundary Intersect over Union

Lagt inn noe om genereringen av Mockdata.

Hvordan gå videre til 3D-sammenligning?

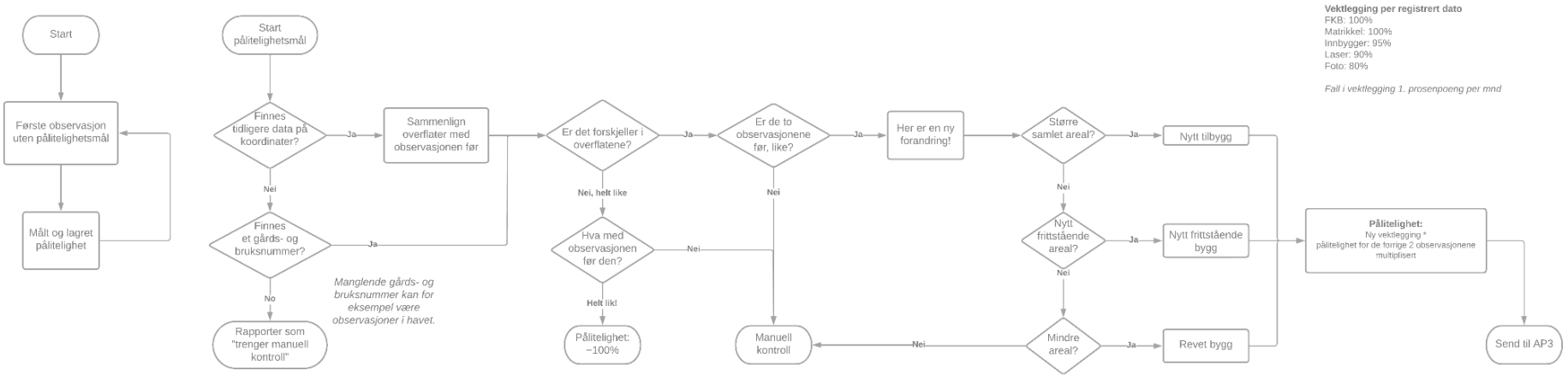
Vi har tatt utgangspunkt at den ferdige dataen vil ligge i PostGIS.

Det kommer noe om Sammenligning av 3D - 2D.
Sammenligning av endringer i volum.

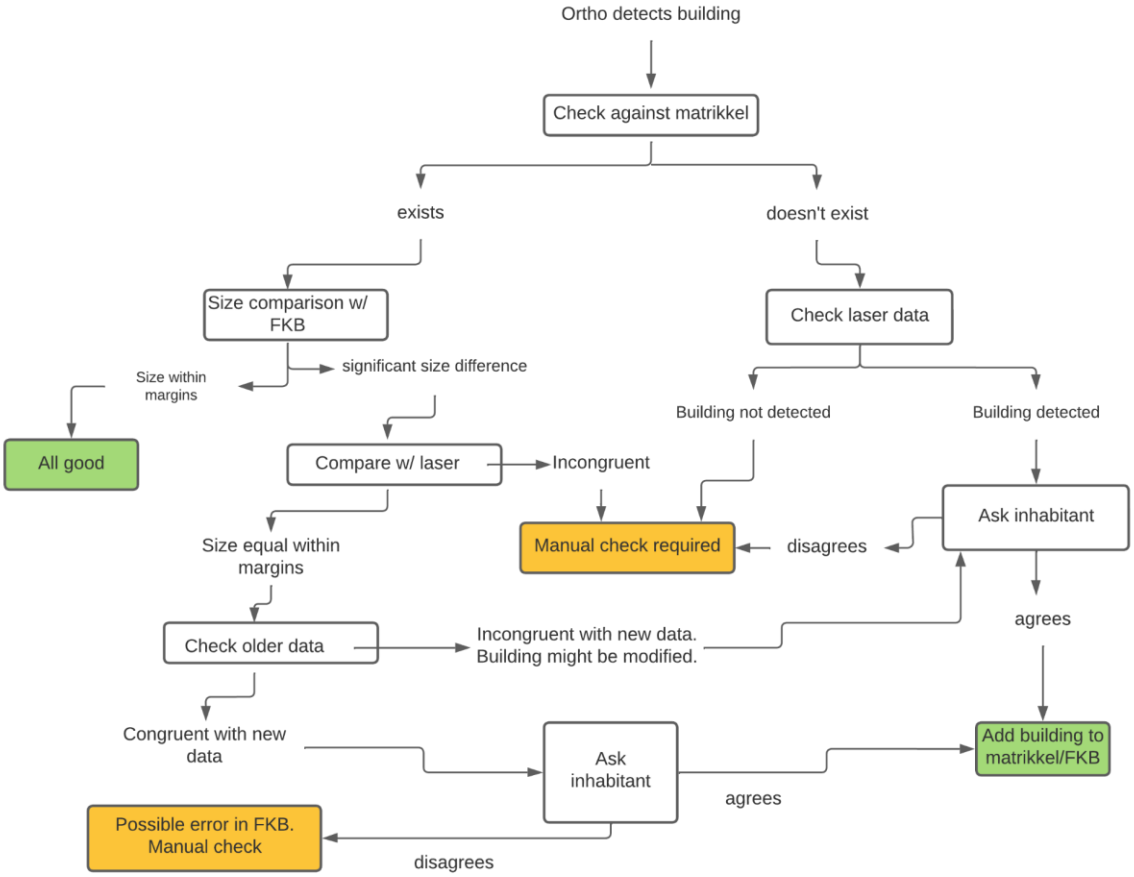
Videre:
Pålitelighetsfaktorer → hvilken score skal en observasjon ha til å begynne med?

- Statistikk
- Målet fra AI-modellen (bruk av AIs "confidence-score")
- Viktig å få med info om AI-ens generelle pålitelighet

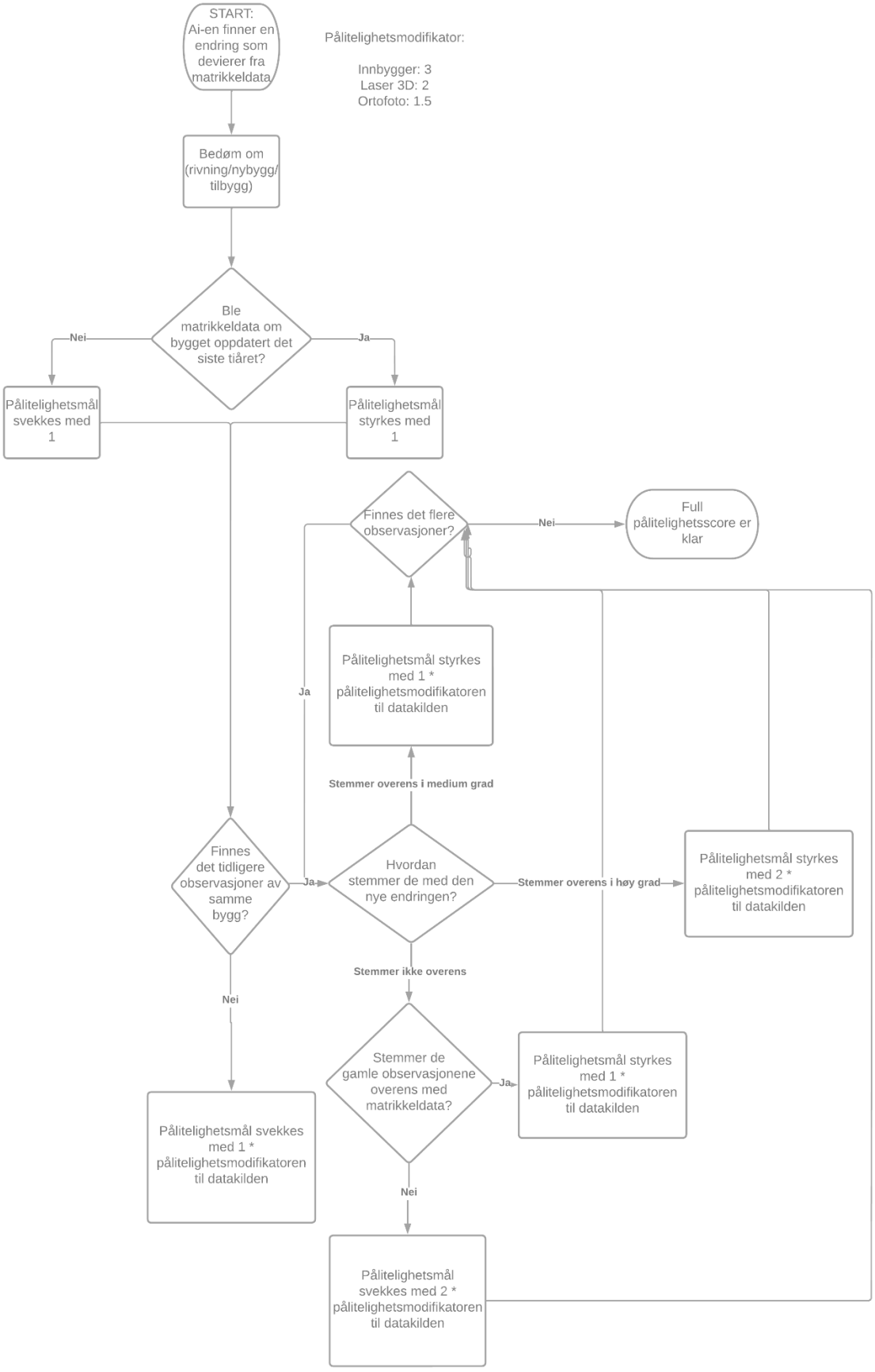
E - Modeller første runde



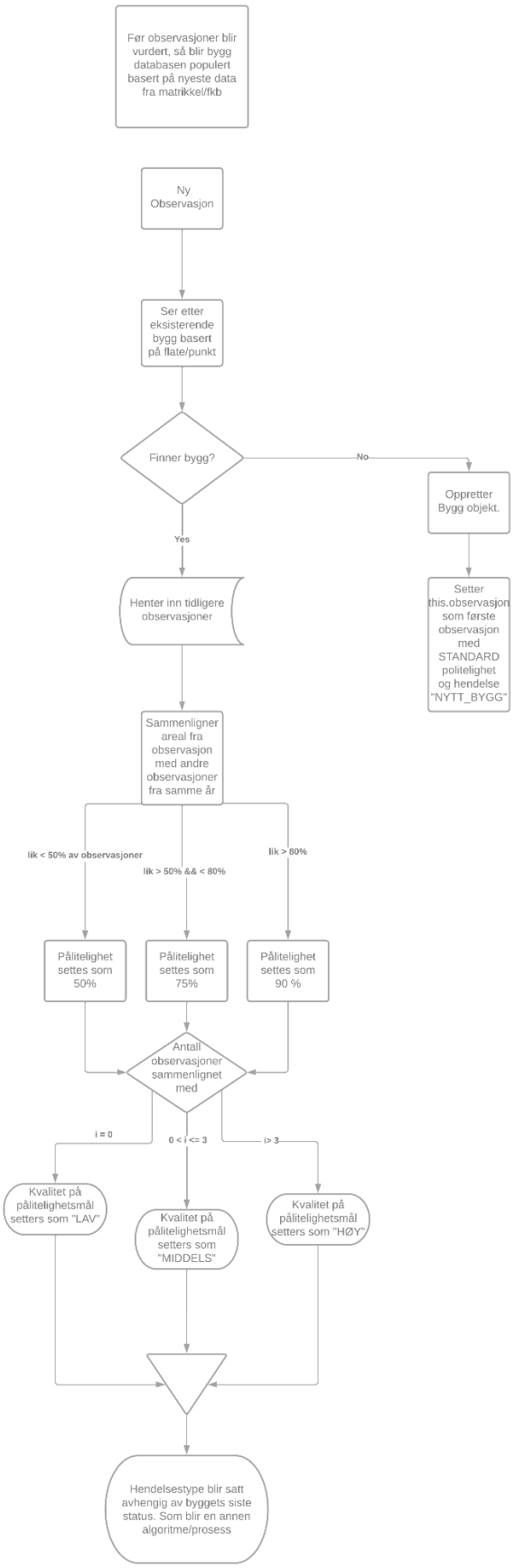
Eiriks modell. Denne tar for seg hele systemets posisjon



Thomas' første modell. Denne tar utgangspunkt i en mer decision-tree.

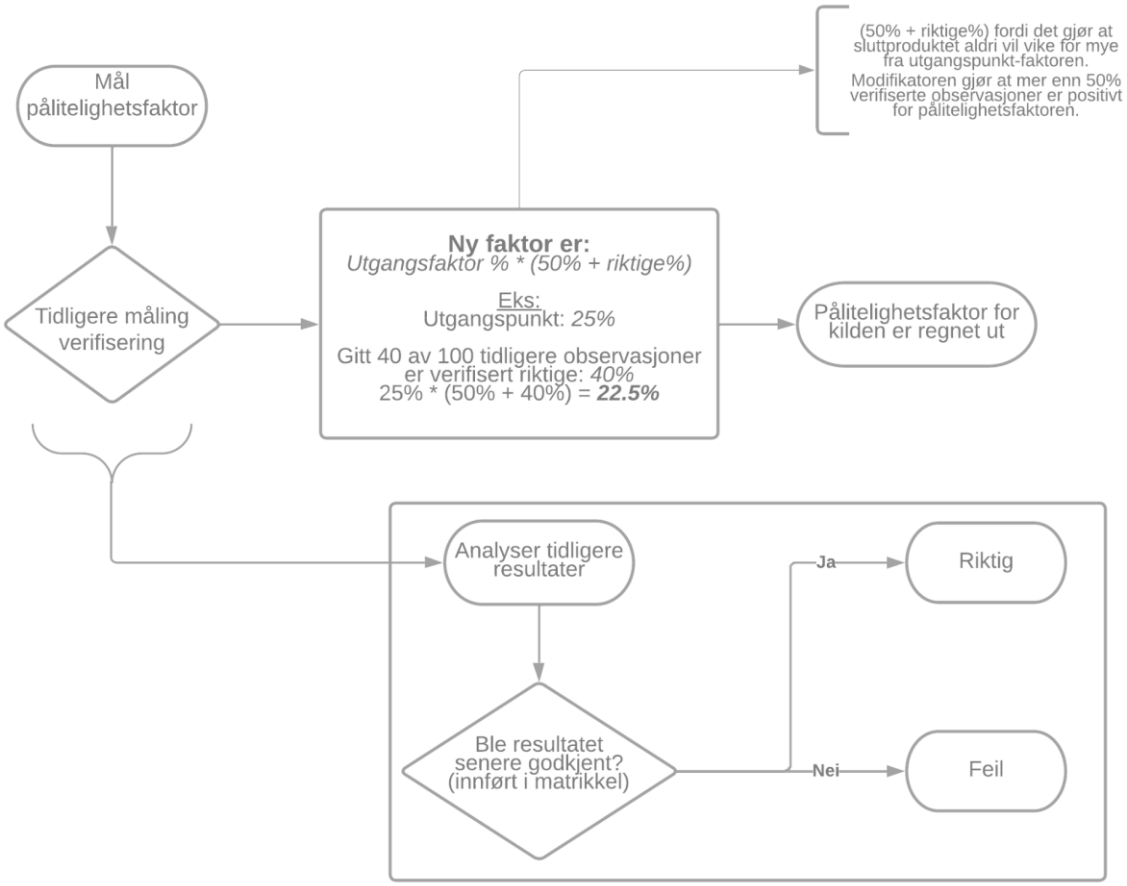


Bendix' første modell.

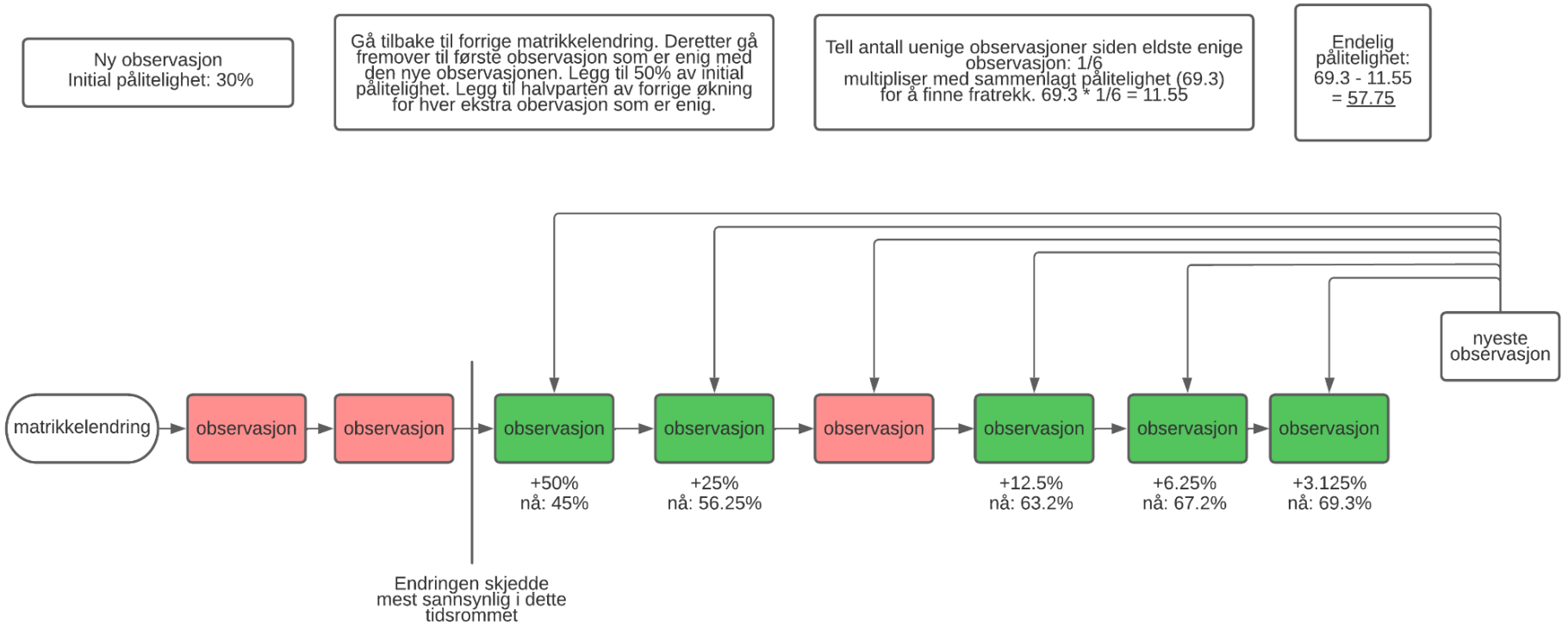


Johannes' første modell.

F - Modeller andre runde



Eiriks andre modell. Denne viser litt hvordan utgangspunkts-pålitelighet kan kalkuleres.

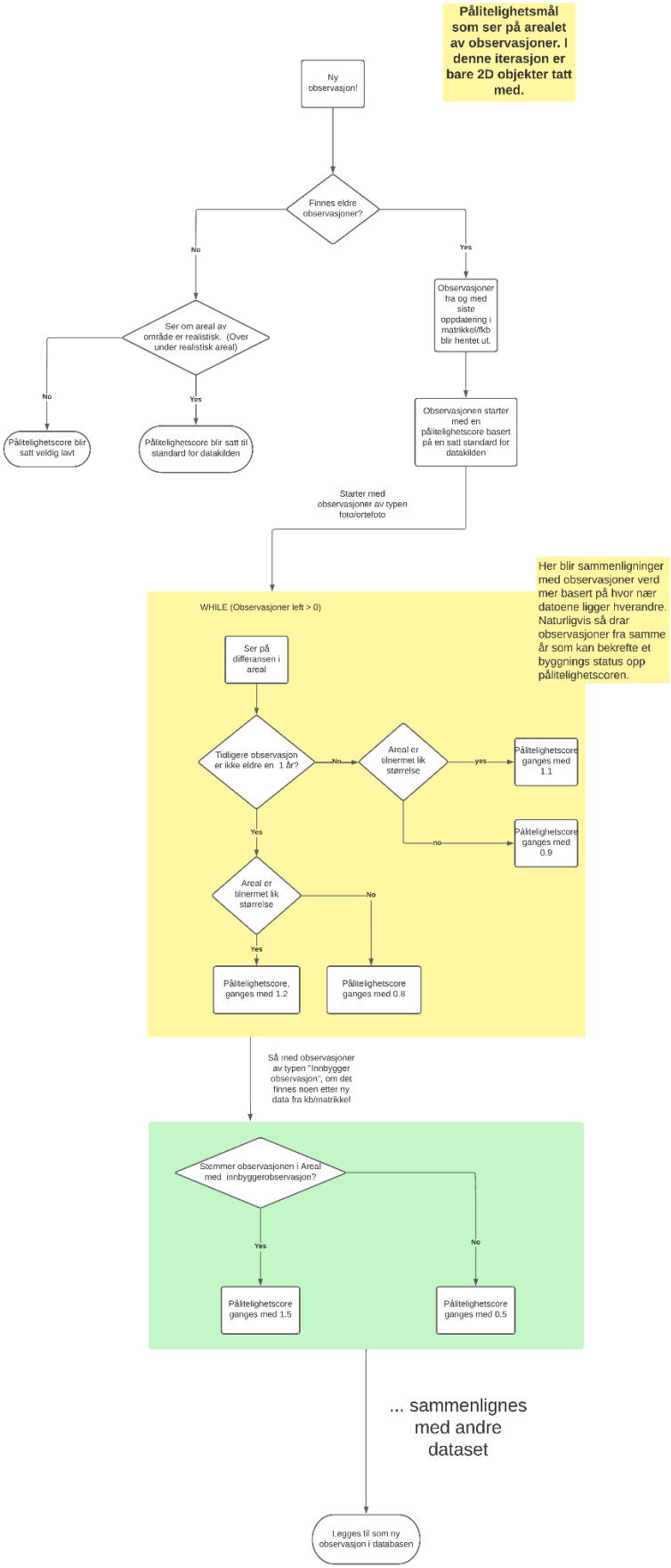


Thomas' andre modell. Endret etter at den ble laget første gang.

Observasjon 1	Faktor	Pålitelighet		Observasjon 2	Faktor	Pålitelighet		Observasjon 3	Faktor	Pålitelighet
Datakilde	Ortofoto	0.5		Datakilde	Laser	0.7		Datakilde	FKB	0.9
År	2008	0.14		År	2018	0.04		År	2022	0
Totalscore		0.36		Totalscore		0.66		Totalscore		0.9

Sammenligning	Gjennomsnitt	Vekt Gjennomsnitt	Likhet (mock-tall)	Vekt Likhet	Pålitelighet
Obs 1-2	0.84	50	0.9	90	0.88
Obs 2-3	1.11	50	0.35	90	0.62
Obs1-3	0.81	50	0.4	90	0.55
Pålitelighet uten FKB					0.88
Total Pålitelighet					0.68

Bendix' andre modell. Denne ble laget i et regneark



Johannes' andre modell.