Universitetet i Sørøst-Norge

Fakultet for teknologi, naturvitenskap og maritime fag

Prosjekt IMPRO

Skrevet av:

Senay Araya

Jehad Mohammed Babawat

Lars Erik Gyving Hoppestad

Ole Markus Lie

Kristoffer Paulsen

Samuel Said

Bacheloroppgave vår 2023

Forord

Dette forordet markerer slutten på vårt gruppearbeid og presentasjonen av denne rapporten. Rapporten er utarbeidet som en del av kravene for å oppnå bachelorgraden ved Universitet i Sørøst-Norge.

Formålet med dette gruppearbeidet har vært å undersøke og analysere kalibrering av treffpunkter på en skyteskive. Gjennom anvendelser og forskning på dette temaet, har vi fått en bredere forståelse og bidratt til å øke kunnskapen om dette temaet.

Vi vil gjerne rekke en spesiell takk til vår interne veileder Joakim Bjørk og vår eksterne veileder Kristoffer Gjone får deres kunnskap som har bidratt til at vi har kommet hit vi er i dag.

Vi har også lyst til å uttrykke takknemlighet ovenfor hverandre som gruppe. Vårt gode samarbeid har vært avgjørende for gjennomførelsen av dette prosjektet.

Sammendrag

Hensikten med denne rapporten er å undersøke hvorvidt bildebehandling kan brukes i kalibreringsprosessen av en målskive. For å utforske dette har den blitt utviklet et kalibreringssystem der hovedsakelig programvare brukes i kombinasjon med mekaniske elementer til å kalkulere posisjonen til vilkårlige treffpunkter på et bilde.

Faktorer som oppløsning, RGB-verdier, lysforhold og ulike kameratyper har blitt undersøkt for å se deres påvirkning på hvordan programvaren klarer å finne og plassere treffpunkter i et koordinatsystem. Gjennom manuell måling av treffpunkter og sammenligning av disse målte posisjonene med posisjonene funnet ved bildebehandling har man kommet frem til et resultat for avviket mellom de to. Dette avviket er innenfor 2 mm ved en avstand på 57,5 mm fra midten av målskiven. Utenfor denne avstanden bidrar tidligere nevnte faktorer til at avviket kan øke ytterligere enn 2 mm. Innenfor de nevnte dimensjonene kan pikselsøk-metoden brukt i programvaren anvendes til å finne treffpunkter presist.

Man har også funnet at avstanden bildet er tatt fra målskiven har stor innvirkning på hvordan RGB-verdiene i et bilde dannes. Lysforhold ved bruk av systemet bør være uniforme. Det vil si at varierende lysforhold i stor grad påvirker presisjonen på systemet. Metoder for å minimere tidbruk og øke presisjonen på systemet har blitt undersøkt, og bidratt til å forbedre systemet. Et brukergrensesnitt har blitt delvis utviklet for å knytte en bruker til kalibreringsprosessen. Dersom man vil minimere avviket ytterligere, og få en høyere presisjon på hele målskiven må man ta høyde for at avstanden kameraet har til ethvert punkt på målskiven varierer ut fra midten av målskiven. En matematisk modell må utvikles for å få til dette.

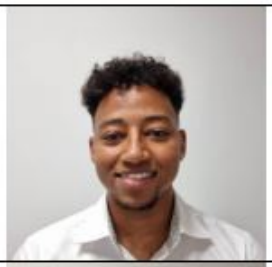
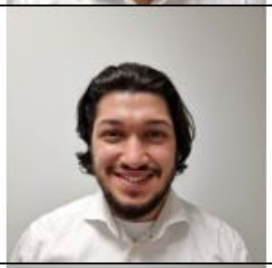
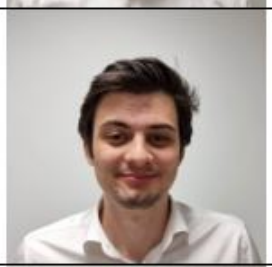
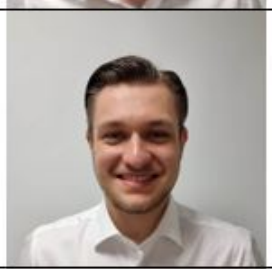
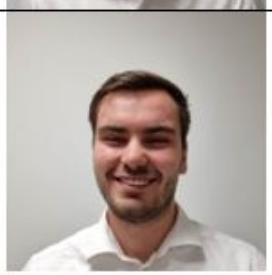
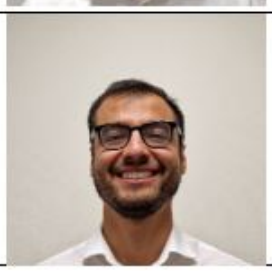
Innhold

1 Ordliste

Ord	**Betydning**
Bildebehandling	Identifikasjon av vilkårlige objekter i et bilde.
I/S	Forkortelse for "Ikke svart". Status er ukjent
Godkjent	Kravet er tilstrekkelig innfridd
Ikke godkjent	Kravet er ikke innfridd tilstrekkelig
API	API står for Application Programming Interface, og er en kode som brukes for å utveksle data mellom to forskjellige systemer eller apper. [1]

2 Prosjekt IMPRO

2.1 Medlemmer

	**Senay Araya** ❖ Dataingeniør ❖ Krav- og økonomiansvarlig
	**Jehad Babawat** ❖ Dataingeniør ❖ Prosessansvarlig
	**Lars Erik Gyving Hoppestad** ❖ Maskiningeniør ❖ Designansvarlig
	**Ole Markus Lie** ❖ Maskiningeniør ❖ Dokumentasjonsansvarlig
	**Kristoffer Paulsen** ❖ Dataingeniør ❖ Gruppeleder ❖ Risikoansvarlig
	**Samuel Said** ❖ Dataingeniør ❖ Testansvarlig

Figur 1: Gruppemedlemmer i Bachelorgruppe 7, Target A".

2.2 Kontrakt

På starten av prosjektet så lagde gruppen en kontrakt som alle gruppemedlemmen skriver under på. I denne kontrakten så er punkter som omhandler hvordan gruppen skal jobbe sammen, verdier vi ønsker at hver gruppemedlem skal ha, arbeidstider og hva som er forventet når gruppen har møter med veiledere.

2.3 Kongsberg Target Systems

KTS ble opprettet i 1994 og har installasjoner på 1900 skytebaner spredt over 30 land. Bedriften jobber med å utvikle, produsere og selge elektroniske målskiver til både sivil og militær bruk på skytebaner. Ved bruk av akustiske sensorer og triangulering, kan systemet detektere treffpunkter på en målskive for en kule skutt fra et skytevåpen. KTS har lokasjon og produksjon på Heistadmoen Industripark.

2.4 Oppgavebeskrivelse

Oppgavebeskrivelsen skal gi en forståelse og oversikt for målskivesystemet til KTS, og problemstillingen som er aktuell i bacheloroppgaven.

2.4.1 Produktbeskrivelse av målskiven

Systemet under utvikling (SUU) bruker målskiven til KTS som utgangspunkt. Derfor er det viktig å ta for seg egenskapene og funksjonene til denne målskiven. Skyteskiven vil være den flaten prosjektilet treffer først, og vil også være synlig for skytteren. Skyteskiven illustreres i figur 6. Det er plastplaten synlig for skytteren som SUU må forholde seg til. Målskivens prinsipper, materialer og utforming er alle faktorer som må undersøkes for å kunne lage et system tilpasset målskiven.

Figur 2: Sammensatt visning av målskiven til KTS

Prinsipp

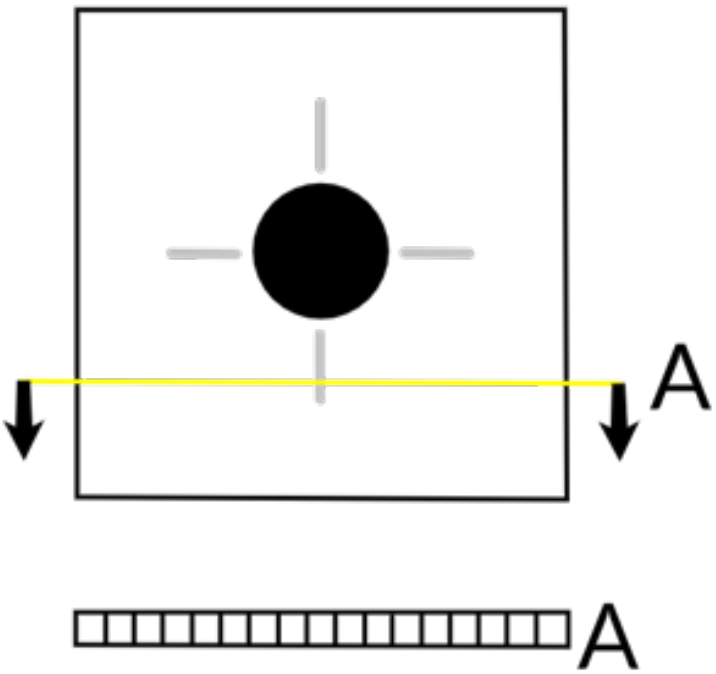
Skyteskiven har et isolert lydkammer med 4 akustiske sensorer (mikrofoner) lokalisert i hvert hjørne av målskiven. Hver sensor registrerer lydbølger som følge av treff, og tidspunkt ved registreringen. Den første mikrofonen som registrerer treff, starter en tidtaker. De 3 andre mikrofonene registrerer trefftidspunktet relativt til den første mikrofonen. Man bruker deretter lydens hastighet v og tiden t som lydbølgen *bruker* til å nå hver sensor, til å beregne strekningen s til treffpunktet. Dette er enkelt beskrevet ved (1):

$$s = v * t$$

Lydens hastighet v varierer noe med temperaturen, siden temperatur påvirker luftens tetthet. Temperatursensorer måler temperaturen slik at systemet kan ta høyde for temperaturendringer. Lydkammeret er også isolert slik at temperaturen i hele lydkammeret er jevn.

Materialer og utforming

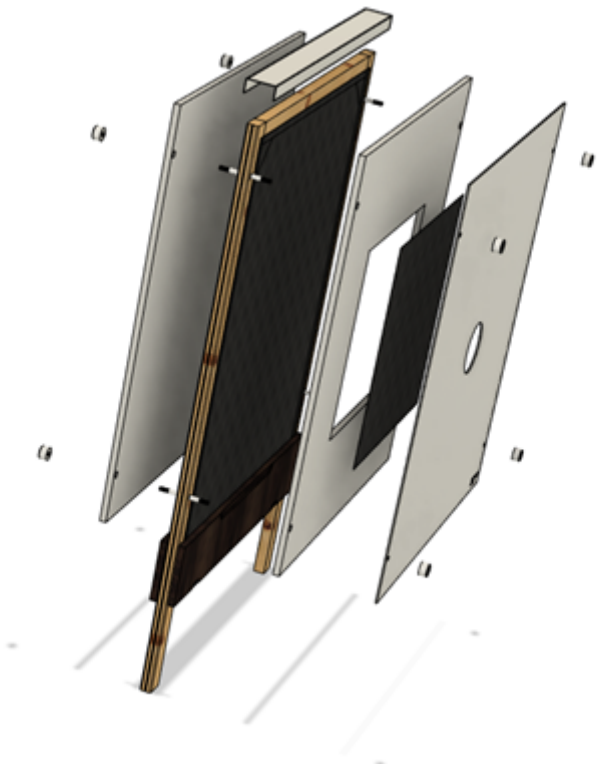
Skyteskiven er laget av en polymer (plast). Den indre strukturen består av mange vegger og er illustrert i forenklet visning ved tverrsnittet A i figuren under. Blinken er den figuren skytteren sikter på, og kan variere i fasong. I vår oppgave skal det brukes en sirkulær blink. Figuren er skåret ut fra skyteskiven og er synliggjort ved en svart bakgrunn laget av en polymer. En bruker kan finne midtpunktet av målskiven ved å følge en horisontal linje og en vertikal linje og se hvor disse krysser hverandre. Linjene som bestemmer midtpunktet kalles for et aksekors. Aksekorset er kappet i samme prosess som skyteskiven og er derfor sentrert absolutt i forhold til senteret av målskiven. Aksekorset er i praksis fire glipper skåret av en kniv, og er derfor vanskelig å se med det blotte øyet dersom man beveger seg lenger enn én meter fra målskiven. Naturligvis er aksekorset usynlig for skytteren, men synlig for den som opererer kalibreringsprosessen rett ved målskiven.



Figur 3: Målskiven til KTS

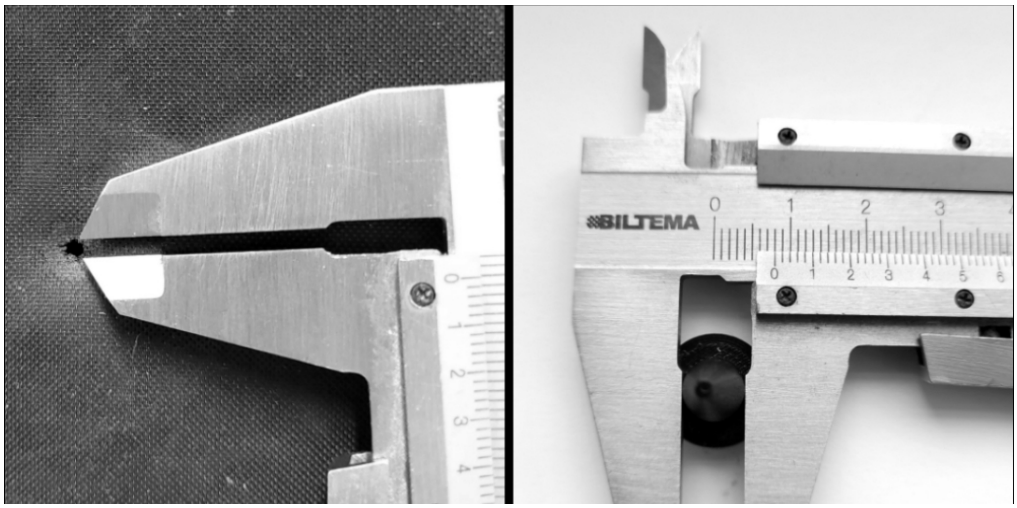
Målskiven består av flere lag med plater med ulike egenskaper for å sikre gode måleforhold. Systemet har 2 konfigurasjoner med enten 4 eller 3 akustiske sensorer, der 4 sensorer gir høyest nøyaktighet på målingene. Figur 4 og figur 6 illustrerer systemets komponenter i en utvidet og sammensatt visning.

Sett fra høyre vil man se den delen av skyteskiven som er synlig for skytteren. Denne delen består av 2 plater, en svart og en hvit. Den svarte delen består av 2 lag med polymer (plast). Det ytre laget som kulen treffer først er en polymer som er vevd som en tekstil. Denne tekstilen er i noe grad elastisk. Det betyr at etter en kule har penetrert dette laget og utvidet stoffet til kulens diameter, har stoffet en evne til å sammentrekke seg rundt treffpunktet. Ved observasjon er det åpenbart at hullet som dannes av kulen er mindre enn diameteren, altså kaliberen, til kulen. Se figur 5. Tykkelsen til tekstillaget er omtrent 1 mm. Det sekundære laget består av en skumgummiplate med lignende elastiske egenskaper til det



Figur 4: Målskive til KTS i utvidet visning

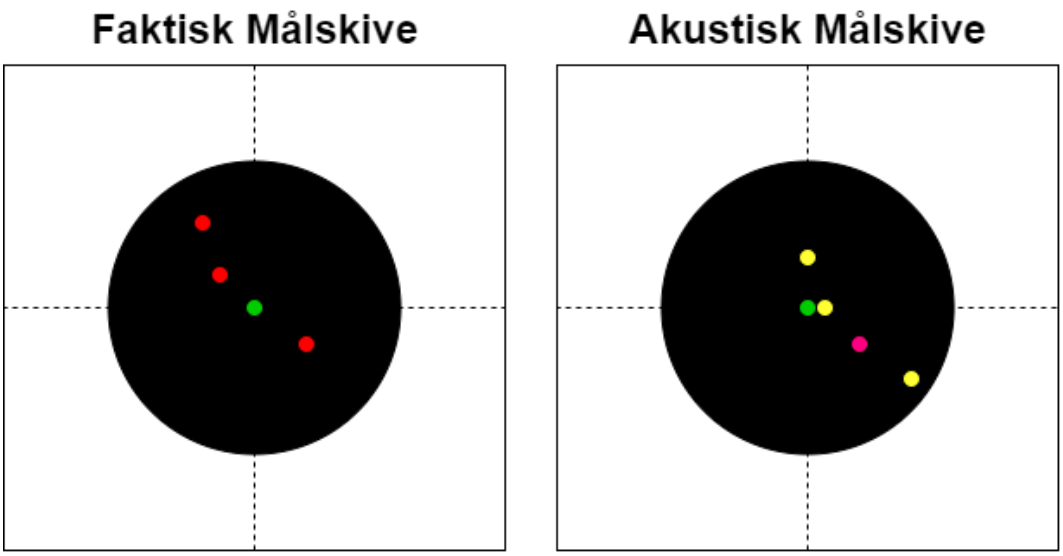
primære tekstillaget. Tykkelsen er omtrent 30 mm.



Figur 5: Forskjell i kaliber og hullstørrelse viser elastisiteten til det første polymerlaget av målskiven.

2.5 Kalibrering av målskiven

Sikteblinken er motivet skytteren forholder seg til når personen skal sikte. Det er essensielt at midten av sikteblinken samsvarer med midten på den elektroniske skyteskiven for å sikre høy nøyaktighet. Kalibreringen av disse to punktene foregår ved å skyte 2-3 skudd på den fysiske målskiven og måle x- og y-koordinater til treffpunktene manuelt. Man finner da avviket mellom de manuelt målte verdiene og den elektronisk skiven. Dette avviket brukes til å korrigere midtpunktet til den elektroniske målskiven. Da vil de fysiske treffpunktene og de elektroniske treffpunktene være like, og systemet er kalibrert. Se figur 6 og figur 7 for en illustrasjon av kalibreringen. Manuell måling av treffpunkter er tidkrevende og antall feilkilder knyttet til menneskelige feil blir betraktelig økt.



Figur 6: Figuren viser treffpunktene (Rød) på den faktiske målskiven og den akustiske målskiven. Med utgangspunkt i at det grønne referansepunktet er i senter av målskiven, vil det være en forskjell mellom de to målskivene. Denne forskjellen er overdrevet slik at den blir synlig.

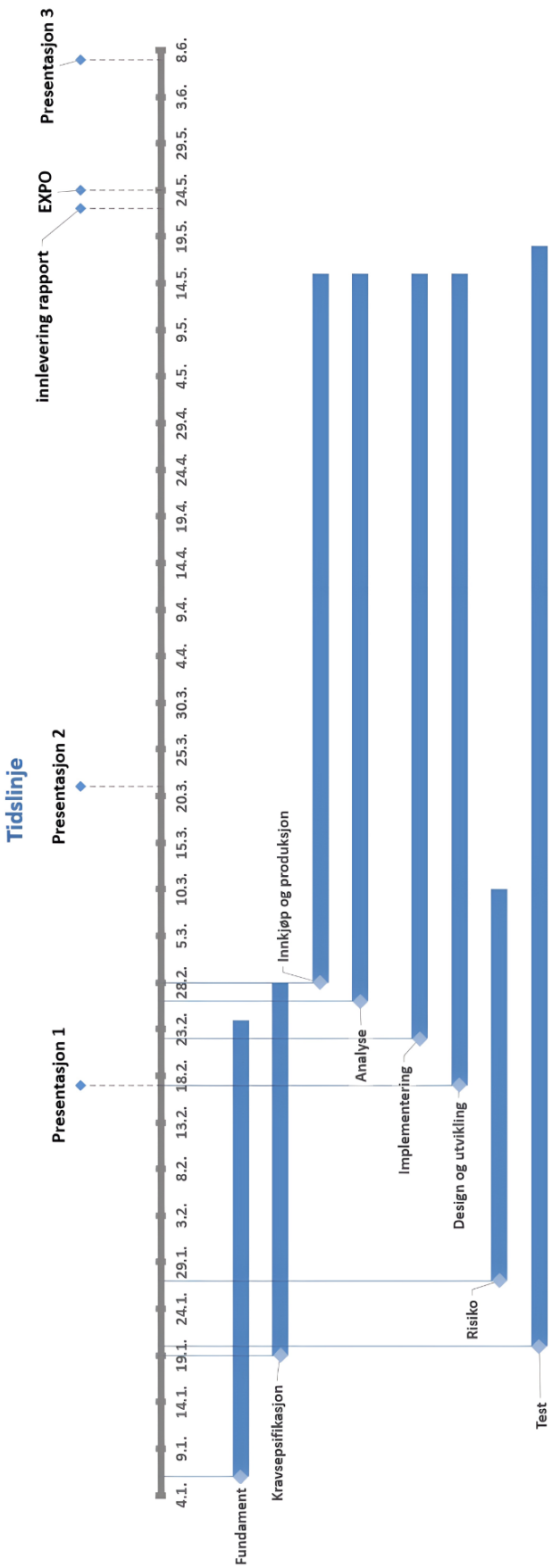
Figur 7: Figuren viser plasseringen av treffpunkter på både den faktiske og den akustiske målskiven. Man ser en tydelig forskyvning mellom de røde og gule punktene. Det er avviket i x- og y-koordinater markert av den blå linjen som brukes til å korrigere det lilla referansepunktet i figur 6 slik at denne samsvarer med det faktiske referansepunktet (grønn).

2.6 Oppgave

KTS ønsker å både forenkle og effektivisere brukerens opplevelse av kalibreringsprosessen. Dersom man enkelt kunne tatt et bilde av målskiven, og brukt denne til å finne treffpunkter og kalibrere målskiven automatisk, ville dette vært i brukerens interesse. Derfor skal det undersøkes hvorvidt bildehandling kan brukes til å kalibrere systemet automatisk. KTS krever at systemet detekterer de fysiske treffpunktene med tilstrekkelig nøyaktighet, altså høyere nøyaktighet enn ved manuell måling. Det skal også undersøkes hvilke krav som stilles til bildeoppløsningen for å oppnå ønsket nøyaktighet. Denne analysen skal dokumenteres.

3 Prosjektplan

Tidslinjen er utarbeidet fra en ferdiglagd Template på Excel med noen modifiseringer. Tidslinjen viser en overordnet oversikt over milepæler og oppgaver som skal gjennomføres. Tidslinjen tar utgangspunkt i framdriftsplanen fra jira på en mer komprimert og oversiktlig måte. Dette gjør det lettere å se hovedlinjene på hva og når ulike ting skal gjennomføres underveis i prosjektet. Som man ser på tidslinjen, står milepælene på oversiden av datoene og oppgaver på undersiden. De blå tykke strekene skal illustrere hvor lenge de ulike oppgavene skal vare.



Figur 8: Tidslinjen til prosjektet

4 Prosjektmodell

En prosjektmodell er en modell som definerer faser og kontrollporter for gjennomføring av prosjekter [2].

Sitatet ovenfor er en bred definisjon sammenlignet med andre definisjoner på hva en prosjektmodell er, men formålet med bruken av en prosjektmodell er kanskje like viktig som modellen selv.

4.1 Formålet med en prosjektmodell

I bruken av en prosjektmodell ønsker man å finne en strukturert, effektiv og konsis metode å utvikle et produkt på. Modellen man velger vil legge rammer og føringer for arbeidet, slik at dette blir utført på den mest hensiktsmessige måten. Det finnes flere arbeidsmetodikker, der alle har både positive og negative sider. I valget av vår prosjektmodell er det viktig for oss at vi faktisk kan bruke den korrekt og hensiktsmessig, og at den er forenlig med de verdiene vi har bestemt som gruppe. Vi lagde derfor en liste med verdier, og med diskusjon kom vi frem til hvordan vi synes det er fornuftig å jobbe sammen. Listen av verdier er følgende:

- Tillit
- Ærlighet
- Frihet
- Mot
- Ydmykhet
- Tydelighet

Med listen ovenfor som utgangspunkt, i tillegg til en overenstemmelse om hvordan vi synes det var fornuftig å jobbe sammen, kunne vi finne en arbeidsmetodikk som kan være tilpasset vår gruppe.

4.2 Hva er arbeidsmetodikk?

"Arbeidsmetodikk baserer seg på et system av teknikker, prosesser og regler brukt av de som jobber disiplinert" [3].

Med utgangspunkt i definisjonen ovenfor finnes det mange ulike arbeidsmetodikker som passer inn under denne. Ved undersøkelse av flere prosjektmodeller kom vi frem til 3 mulige arbeidsmetodikker som kunne være egnet for vår gruppe: Agile, Scrum og Kanban.

4.2.1 Agile metodikk

Agile metodikk ble dannet med hensyn til mål, iterasjoner og konstant feedback som utviklet seg fra "Waterfall"metodikken. "Waterfall"metoden innebærer en stegvis designprosess og var en metodikk som ofte ble brukt i programvareproduksjon. Denne progresjonsmetoden fungerte utilstrekkelig i møte med uventede hendelser som stilte nye krav til produktet. Dermed oppsto behovet for en iterativ modell, med økende inkrementer i hver iterasjon for å klare å holde tritt i teknologimarkedet.

Metoden er best for prosjekter som krever fleksibilitet og en viss grad av kompleksitet eller uforventede resultater. Dette passer best for et team som ikke har produsert et lignende produkt før.

4.2.2 Scrum metodikk

Scrum metodikk stiller krav og forventninger til hvert medlem av gruppen til å ha bestemte verdier som tilrettelegger for et godt arbeidsmiljø. Disse verdiene er

fokus, mot, åpenhet, forpliktelse og respekt. Med disse verdiene som grunnlag har Scrum en rekke verktøy som Scrum Roles, Scrum Events og Scrum Artifacts. Metodikken er basert på spesifikke begrensninger og regler for hvordan man skal utføre planlegging av prosjektet for å skape gode sluttresultater.

Metoden er best egnet for et lite team på under 7 personer med iterativ, fleksibel og revisjonert diskusjon.

4.2.3 Kanban metodikk

Kanban baserer seg på mye av det samme som Scrum inneholder, men den essensielle forskjellen mellom de to er at Scrum bruker sprints. Sprint vil si å ha et visst avgrenset tidspunkt der gruppen må prøve å oppnå et mål. I Kanban derimot brukes noe som kalles Kanban Board. Kanban Board blir brukt til å visualisere prosessen av prosjektet og skaper en forenklet forståelse for prosjektets status.

Metoden er best egnet for mindre grupper eller personlig bruk.

4.3 Utvalget og hvorfor?

Som gruppe har vi valgt å bruke Scrum som prosjektmodell. Grunnen til at vi valgte Scrum skyldes forhåndsplanleggingen av hvordan gruppen ønsket å jobbe sammen. Gruppen ble enige om følgende:

- Å ha en revisjon ved arbeidsukens slutt som inneholder oppsummering av hvordan arbeidet har gått, og planlegging til neste uke.
- Å ha daglige møter med oppdatering fra hvert medlem om arbeidet de driver med.
- Å ha bestemte verdier som hvert eneste medlem skal leve etter i gruppearbeidet.

Ved å undersøke Scrum litt nærmere kan man enkelt se at gruppen har satt opp noen store faktorer som er svært like arbeidsmetodikken. Likheten til Scrum er tydelig ved disse parallellene: Sprint retrospective"og Sprint reviewpasser med revisjon og planlegging. Daily scrumpasser med medlemmenes arbeidsoppdatering. Verdiene "Courage", Focus", "Commitment", Respect"og "Opennessfølger omtrent de verdiene vi ønsker å forholde oss til i gruppearbeidet [3]. Med hensyn til disse faktorene var Scrum en god prosjektmodell for gruppen.

4.3.1 Forventninger til gruppemedlemmer

Som et gruppemedlem er det viktig å forholde seg til og følge retningslinjene bestemt av Scrum. Disse verdiene og rollene er beskrevet i punktlistene under:

- Tillit: En vanlig utfordring som kan oppstå i arbeidsmiljøet er flaskehalssituasjonen. Det vil si en situasjon der en arbeidsoppgave tar mer tid å utføre enn forventet, samtidig som andre oppgaver er avhengige av den arbeidsoppgaven for å kunne utføres. I prosjekter er det viktig å ha så stor flyt som mulig og prøve å unngå problemer som flaskehalser. Ved å skape tillit bygger man troverdighet mellom gruppemedlemmer, slik at man kan kommunisere åpent og avdekke flaskehalssituasjoner før de oppstår. Dermed vil man skape en god progresjonsflyt.

- Mot: Hvert medlem forventes å ha nok mot til å utfordre ideene til andre gruppemedlemmer, slik at arbeidet blir ført riktig.

- Fokus: Hvert medlem skal holde fokus på det forhåndsplanlagte målet som skal oppnås innen en gitt tid.

- Dedikasjon: Hvert medlem tar en personlig beslutning om å ha dedikasjon til å prøve å oppnå målene som er satt for gruppen.

- Respekt: Hvert medlem viser respekt ovenfor andre medlemmer ved å anerkjenne at de er i stand til å utføre sine oppgaver og være selvstendige

medlemmer i gruppen.

- Åpenhet: Hvert medlem av gruppen har plikt til å være åpen og ærlig om arbeidet som blir utført og de utfordringer som oppstår.

4.3.2 Scrum roller og ansvar

- Scrum Master: Vedkommende som har mest kontroll og kompetanse innenfor Scrum og kan utnytte sin kunnskap til å veilede gruppen til å arbeide mer effektivt.
- Produkt eier: Vedkommende som har ansvaret for å sikre at gruppen produserer et produkt som kunden ønsker.
- Utviklere: Vedkommende som jobber sammen med andre utviklere for å innovere, produsere og løse problemstillinger.

4.3.3 Scrum Events

- Sprint: En periode på en måned eller mindre der et team bruker Scrum Events for å prøve å oppnå målene som er satt for sprinten. Umiddelbart etter at sprinten er ferdig, starter en ny.
- Sprint Planning: Perioden mellom slutten av forrige sprint og starten av den kommende sprinten, der målene for den kommende sprinten og hvilket arbeid som må gjennomføres, blir satt opp.
- Daily Scrum: Et møte i begynnelsen av dagen der hvert medlem diskuterer hva de skal forsøke å oppnå i løpet av dagen.
- Sprint Review: Et arrangement som finner sted i slutten av sprinten der teamet samles for å diskutere hva som er oppnådd og hva som har endret seg.

- Sprint Retrospective: Utføres umiddelbart etter Sprint Review, der gruppen diskuterer hvordan forrige sprint gikk og hvilke endringer som er nødvendige for å oppnå mer effektivitet.

4.3.4 Scrum Artifacts

- Product Backlog: En liste over ting som må legges til, oppdateres eller endres med produktet. Typisk illustrert som oppgaver, feil som må rettes eller historier.
- Sprint Backlog: En visuell liste over oppgaver som kan modifiseres (legge til, endre eller slette) som utviklerne har planlagt å gjennomføre for å oppnå brukerhistorie eller krav.
- Increment: Små deler av arbeidet som bygger et grunnlag mot bevegelsen mot hovedmålet.

4.4 Bruk av Scrum

Gruppen vil bruke Scrum-metodikken ved å bruke et utvalgt programvareverktøy som allerede har en Scrum-mal og funksjonaliteter som gruppen kan hoppe rett inn i. Dette programvareverktøyet heter Jira. Gruppen har fått benytte seg av Jira med , som har visse begrensninger, og begrenser oss til en Roadmap, Backlog og Scrum Board. Innenfor et Scrum Team så er det ingen som har en sjefstilling ovenfor noen andre i gruppen. Derimot har noen ulike roller med litt ulikt ansvar. Når en avgjørelse skal bli tatt så er hver medlems mening på lik linje som en annen.

4.4.1 Roadmap

En Roadmap vil enkelt vise frem en tidslinje og en felt man kan lage Epic (Som er et annet ord for User-stories og kan sammenlignes med produkt krav) og oppgaver som står under Epic. Man har mulighet til å sette opp flere Epic og deretter kategorisere oppgaver iforhold til Epic for å deretter sette dem i et Sprint som blir visualisert på en tidslinje der gruppen forventer å utføre det.

4.4.2 Backlog

Backlog i Jira vil være funksjonaliteten som tillater oss å lage så mange "issues" (synonym for oppgaver i sammenheng med Jira) vil og kan starte en sprint med hensyn til de oppgavene som passer til å oppfylle en Epic/User-Stories/Krav.

4.4.3 Scrum board

Når gruppen har kommet frem til en rekke oppgavene som kan plasseres i en sprint, kan Scrum Board bli sett på som en Sprint Backlog. Scrum Boardet gir oss en visuell oversikt over progresjonen i forhold til hva som ble planlagt i sprinten. Oppgaver i Scrum Boardet er delt opp i fire statuser.

- Gjøremålstatus. Oppgaven har blitt satt opp men ikke påbegynt eller utført.
- Under arbeid-status. Oppgaven er påbegynt og under arbeid.
- Testingstatus. En oppgave som faller innenfor testing og utvikling.
- Ferdigstatus. Oppgaven er fullført i tilhørende sprint.

4.4.4 Sprint

Sprintperioden blir vanligvis satt opp i en periode på én eller to uker. Det forventes å gjennomføre en Daily scrum hver arbeidsdag innenfor en Sprint. I slutten av en sprint samles hvert medlem for å gjennomføre Sprint review, Sprint retrospective og til slutt Increment som legges til produktet helt til målet er nådd. Disse tre Artifacts dokumenteres.E.1 Se vedlegget for sprintrevisjon. Deretter begynner gruppen på neste steg som er Sprint planning og forbereder oss til neste sprint.

Se seksjon E.1 for mer informasjon

4.5 Verktøy

Under dette prosjektet har vi brukt flere verktøy som har hjulpet oss med å gjøre prosjektet best mulig. Dette er forskjellige verktøy som tegneverktøy, kommunikasjonsverktøy og kunstig intelligens. Under er det listet opp de verktøyene vi har brukt:

- **Teams:** Vi har brukt teams til å holde styr på og dele dokumenter med hverandre. Vi har også brukt teams til å ha møter med veiledere.
- **Messenger:** Vi har brukt messenger for formell mens også uformell kommunikasjon mellom medlemmene i gruppen.
- **Visual Studio Code:** Vi har brukt Visual studio code til å utvikle frontend og backend av systemet vårt.
- **Solidworks:** Vi har brukt solidworks for 3D modellering og 2D-tegning
- **ChatGPT:** ChatGPT er en chatbot som har hjulpet oss med små og store problemer under prosjektet. Dette kan være alt fra kode deler til rettskriving.
- **Jira:** Vi bruker jira til å dokumentere hva de forskjellige medlemmene gjør av arbeidsoppgaver.
- **Excel:** Vi har brukt excel til å føre resutlater av tester, lage diagrammer og til å føre økonomi
- **Latex:** Til hele dokumentasjonen så har vi brukt Latex. Dette har vi brukt fordi vi er flere som jobber på samme dokumentasjon, så vi holder en god oversikt.
- **Draw.io:** Draw.io har blitt brukt til å tegne og illustrere de forskjellige figurene vi har.

- **Google foto** Vi har brukt Google foto til å dele bilder med hverandre som er tatt med de forskjellig telefonene til testing. Dette er for å sikre at kvaliteten holder seg.
- **Expo go:** Expo go er en app på telefonen som man kan bruke som emulator for å teste appen man utvikler.

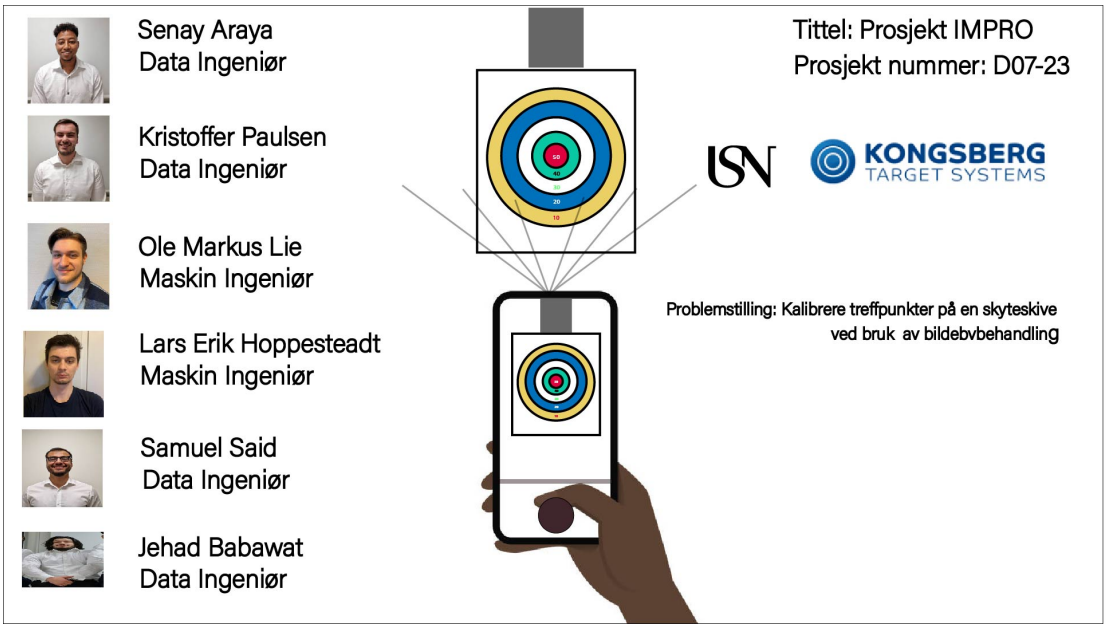
5 Grafisk design

5.1 Nettside

Et av arbeidskravene vi har er å lage en nettside. På denne nettsiden så star det informasjon om gruppen, en oppgavebeskrivelse og hvor man kan følge reisen vår igjennom dette vårsemesteret.
Dere kan følge med på vår reise på www.targeta.no

5.2 Plakat

Plakaten er et arbeidskrav som vi må gjøre. Denne plakaten skal vises på EXPO som er etter gruppen har levert bacheloroppgaven.



Figur 9: Plakat for EXPO

6 Risikoanalyse

I dette prosjektet er det flere risikoer som kan oppstå underveis. Vi har flere risikoer knyttet til prosjektet vårt. Disse kan være relatert til oppgaven i seg selv, at oppdragsgiver går konkurs, eller at et medlem av gruppen skulle forsvinne underveis. Det er derfor viktig at vi på forhånd har gjort oss noen tanker rundt dette og laget en plan på hvordan vi kan løse disse risikoene skulle de oppstå. Vi har laget en risiko matrise som baserer seg på en verdi som er på mellom 1-16. Denne verdien blir bestemt av sannsynligheten for at en hendelse vil inntreffe og ganger det med alvorlighetsgraden av hendelsen. Hvis en risiko har verdien 16, vil det si at det er veldig høy sannsynlighet for at hendelsen vil inntreffe, men også at det er katastrofalt at det skjer. Vi har valgt denne metoden for da kan vi sette en tallfestet verdi på en risiko slik at det blir mer konkret hvor stor en risiko faktisk er. I vår risikomatrise så har vi brukt fire forskjellige nivåer av risiko: Lav, medium, høy og veldig høy. Disse er representert i sine egne farger: Nedenfor er det laget noen beskrivelser av de forskjellige delene i risikoanalysen vår.

6.1 Beskrivelser av betydninger

Sannsynlighet

Alvorlighetsgrad	Høy sannsynlighet 4	Sannsynlig 3	Moderat 2	Liten sannsynlighet 1
Katastrofalt 4	Veldig høy - 16	Veldig høy - 12	Høy - 8	Medium - 4
Alvorlig 3	Veldig høy - 12	Høy - 9	Høy - 6	Medium - 4
Betydelig 2	Høy - 8	Høy - 6	Medium - 4	Lav - 2
Ubetydelig 1	Medium - 4	Medium - 3	Lav - 2	Lav - 1

Figur 10: Risikomatrise

Alvorlighetsgrad	Beskrivelse
Liten sannsynlig	Dette vil ikke påvirke gruppen
Moderat	Dette kan påvirke gruppen
Sannsynlig	Dette vil påvirke gruppen
Høy sannsynlighet	Dette vil påvirke gruppen i stor grad

Figur 11: Beskrivelse av alvorlighetsgradene

Sannsynlighetsgrad	Beskrivelse
Liten sannsynlig	Dette vil ikke oppstå
Moderat	Dette kan oppstå
Sannsynlig	Dette vil mest sannsynlig oppstå
Høy sannsynlighet	Dette vil oppstå

Figur 12: Beskrivelse av sannsynlighetene

Risiko	Betydning
Lav	Tiltak er ikke nødvendig
Medium	Tiltak bør vurderes
Høy	Tiltak må vurderes
Veldig høy	Tiltak må gjøres

Figur 13: Beskrivelse av risikoene

7 Kravspesifikasjon

Alle krav utarbeides fra User stories i henhold til arbeidsprosessen Scrum. Dannelsen av krav legger grunnlaget for videre produktutvikling. Se vedlegget i kapittel C for hele kravspesifikasjonen.

7.1 User stories

En "user storyer en beskrivelse av funksjonen til et system sett fra brukerens perspektiv. Disse "user storiesbrukes for å utforme kravene. På hver "user storyfølger identifikasjonen av kravene utledet fra brukerhistorien. Ett krav kan også være tilknyttet flere brukerhistorier.

7.2 User story 1

Som bruker ønsker jeg at plasseringen av treffpunktene på både den fysiske blinken og den elektroniske blinken skal være identiske, for at resultatene i videre skyting skal bli riktige. Derfor må jeg kalibrere målskiven før bruk.

Tilhørende krav: FK-A2, FK-A4, FK-A6, FK-A7, FK-A8, FK-A10, FK-A11, FK-A12, FK-B2, FK-B3, FK-B5, FK-C5

7.3 User story 2

Som bruker ønsker jeg å raskt og enkelt kunne ta et bilde av målskiven, og kalibrere målskiven automatisk for å unngå tidkrevende, unøyaktige og manuelle mål eller prosesser.

Tilhørende krav: FK-A1, FK-A3, FK-A8, FK-A9, FK- A10, FK-A11, FK-B3, FK-B4, FK-B5 FK-C1, FK-C2, FK-C3, FK-C6

7.4 User story 3

Som bruker ønsker jeg å undersøke resultater og data om kalibreringen til enhver tid for å verifisere at kalibreringen er riktig.

Tilhørende krav: FK-A3, FK-A4, FK-A9, FK-B1, FK- B4, FK-C4, FK- C6

7.5 User story 4

Som kunde ønsker jeg å tilegne kunnskap om parametere knyttet til kalibrering med bildebehandling for videre undersøkelse eller utvikling.

Tilhørende krav: FK-A5

[**?**]

7.6 Funksjonelle krav

Et funksjonelt krav beskriver hva som må gjøres for at et system skal fungere. Et funksjonelt krav er per definisjon, et verb. Denne typen krav bidrar til å tydeliggjøre og definere adferden systemet bør ha. I tillegg skal et funksjonelt krav være direkte tilknyttet brukerens opplevelse av systemet. Et eksempel på et funksjonelt krav er FK-A1: Programvare skal ved hjelp av bildebehandling detektere treffpunkter på en skyteskive.

7.7 Mal til krav

Malen beskriver aktuelle elementer relevant til kravets struktur.

Status	Hvilken status har kravet?
Krav identifikasjon	FunksjonelleKrav-AF1
Kravbeskrivelse	Hva er kravet
Prioritet	A/B/C

8 Testplan

Testplanen omhandler prosessen man må følge for å utføre testing av systemet. Dette er en viktig del av prosjektet for at man skal kunne vurdere om kravet er godkjent. I denne testplanen har det blitt valgt å følge IEEE-standard (IEEE 829). Ettersom IEEE-standarder er ment for større prosjekter så har det blitt tatt med kun punkter som er aktuelle for dette prosjektet. Disse punktene kommer fram videre i dokumentet.

Testene kjøres hovedsakelig sekvensielt, altså skal testene kjøres underveis for hvert krav som blir ferdig. Det skal også regresjonstestes dersom testen feiler i første omgang for å sjekke at tidligere feil er rettet opp. Videre så testes noen krav sammen på en test, ettersom de kravene naturligvis må fungere sammen. Ettersom det ikke er sikkert at alle kravene innenfor en test blir utført så er det markert hvilke deler av test beskrivelsen som tilhører til hvilket krav med kode farge som rød, grønn og blå. Så det vil si at dersom man rekker å utføre testen for et av kravene, men ikke alle kravene i testen så kan deler av testen fortsatt godkjennes. Dette er fordi en ikke kan forutse hvor langt man kommer i kravtestingen.

8.1 Prioritet

A-kravene er det som skal prioriteres først. Når A-prioritets tester blir godkjent som et system, så skal det kjøres samme prosess for å teste B-krav. Videre så skal C-kravene ha samme prosess dersom det er tid til overs. Siden kravene allerede er validert, trengs det ikke validering for hvert eneste krav underveis, men heller at det kjøres en systemtest mot slutten hvor kunden validerer at systemet er faktisk det han var ute etter.

8.2 Testmetoder

Metode	Beskrivelse
Funksjonell test	Funksjonellest er en test for å verifisere om arbeidet oppfyller funksjonelle krav.
Ikke-funksjonell test	Ikke-funksjonell test er en test for å verifisere om arbeidet oppfyller ikke-funksjonelle krav.
Brukertest	Tredjepart testing er en test hvor noen som ikke har vært med på arbeidet med prosjektet tester systemet.
Systemtest	Tester om systemet fungerer sammen inkluderer interaksjonen mellom komponenter i systemet.
Kompatibilitetstest	Tester systemet eller deler av komponentene av mot eksisterende system.
Enhetstest	Tester individuelle komponenter av systemet.
Automatisert test	Er en test hvor programverktøy brukes for å sende inn mange forskjellige inputs automatisk for å sjekke om man får forventede resultater.

8.3 Verifikasjonsmetode

Metode	Beskrivelse
Observasjon	Verifiserer testen ut ifra observering som for eksempel at man observerer at man får ut en x og y koodinat.
Måling	Verifiserer testen utifra målinger som for eksempel måling av avstanden mellom to punkter.
Utregning	Verifiserer testen utifra regning av dataer som for eksempel gjennomsnitt av brukerundersøkelse.

8.4 Testbeskrivelse

Status	
Test ID: FunksjonelleKrav-AT1	FK-A1
Krav ID	Funksjonelle Krav-Prioritet Nr.
Testmetode	Hvilken testmetode brukes?
Beskrivelse	Hvordan utføres testen?
Testmiljø	Lys og værforhold testen utføres i?
Verifikasjonsmetode	Hvordan verifiserer man testen?
Utstyr	Hvilket utstyr blir brukt for å utføre testen?
Resultat	Hva fikk man ut av testen?
Akseptansekriterium	Hva er det som må til for at testen skal være godkjent?
Utført av	Hvem utførte testen?

For å se alle testspesifikasjoner se på vedlegg.

9 System design

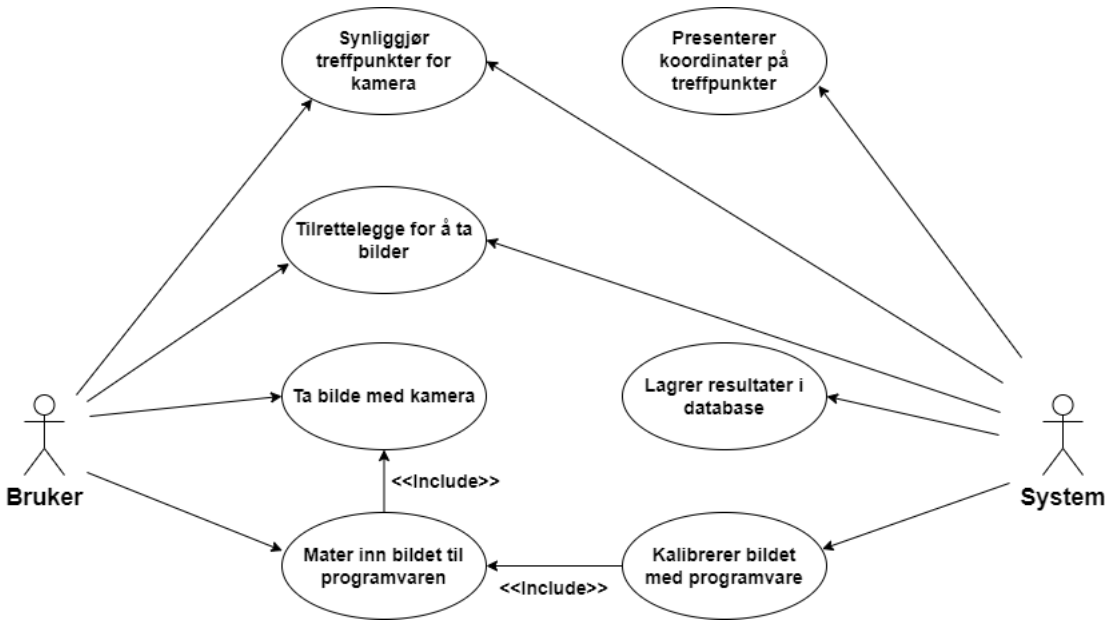
Formålet med systemdesign er å knytte brukerens krav til systemets adferd, og fysiske arkitektur. User stories legger grunnlag for systemkravene. I systemkravene finner man hovedfunksjoner som kan kartlegges med et use case diagram. Deretter kan rekkefølgen på de hovedfunksjonene beskrives med et flytdiagram. Et N2 diagram utfyller funksjonene i flytdiagrammet med relevante input- og output-data. Deretter kan man bryte ned hovedfunksjonene til subfunksjoner inntil man finner ett element som kan utføre én eller flere subfunksjoner. På den måten vil man skape en fysisk arkitektur fra den funksjonelle arkitekturen. Både N2 diagrammer og sekvensdiagrammer brukes for å beskrive funksjoner som foregår i samme tidsrom.

9.1 Use case diagram

Use case-diagrammet kartlegger:

- Funksjonene knyttet til brukeren og systemet.
- Interaksjonene mellom funksjonene, brukeren, og systemet.

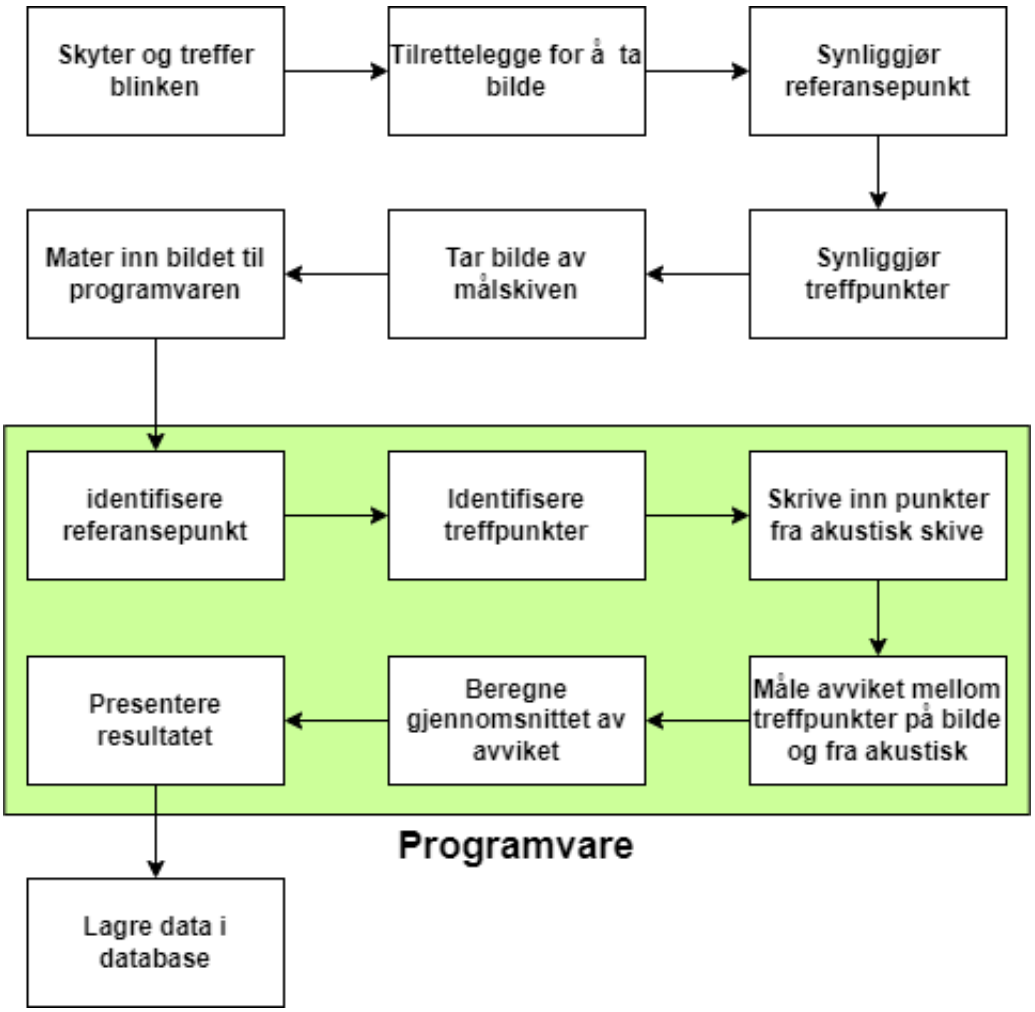
Systemet vil bestå av elementer som utfører funksjoner, men alle interaksjoner/grensesnitt både innad i systemet, men særlig utad, må også kartlegges for at systemet skal fungere helhetlig.



Figur 14: Use case diagrammet beskriver interaksjonen mellom brukeren og systemet

Med use-case diagrammet ser man hvilke funksjoner brukeren og systemet må gjøre, og hvordan disse er knyttet til hverandre. Med utgangspunkt i disse hovedfunksjonene er systemets adferd illustrert i flytdiagrammet nedenfor.

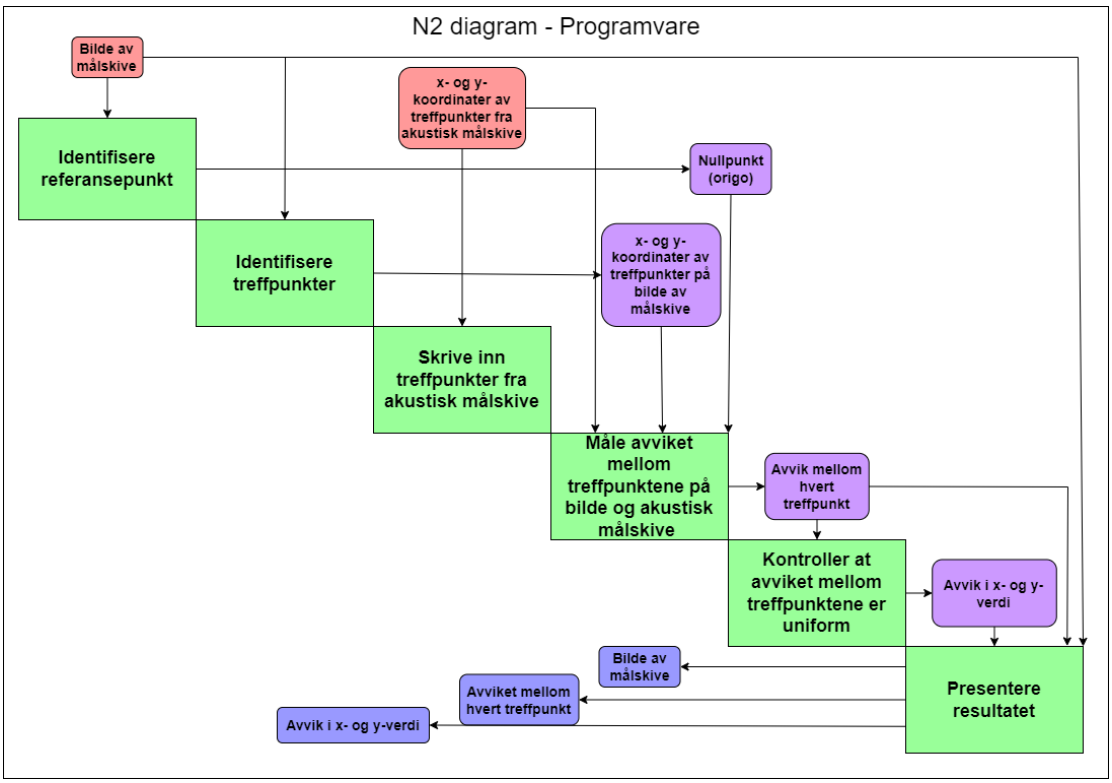
9.2 Flytdiagram



Figur 15: Funksjonelt flytdiagram. Diagrammet beskriver rekkefølgen på alle hovedfunksjoner fra brukerens første interaksjon med systemet til systemets formål er oppnådd

Flytdiagrammet ovenfor brukes for å se rekkefølgen funksjoner opererer i og hvordan disse er strukturert. Her ser man at seks av funksjonene kan grupperes inn i én programvare. Man ser også hvilken rekkefølge disse bør struktureres i, og dette gir en pekepin på hvilke funksjoner man må prioritere i utviklingen. For eksempel må et referansepunkt identifiseres som nullpunkt, før man kan identifisere et treffpunkt med x- og y-koordinater.

9.3 N2 diagram



Figur 16: N2 diagrammet illustrerer hvilke inputs og outputs som er relevant til hver funksjon. Outputs er plassert horisontalt og inputs vertikalt, sett fra hver funksjon.

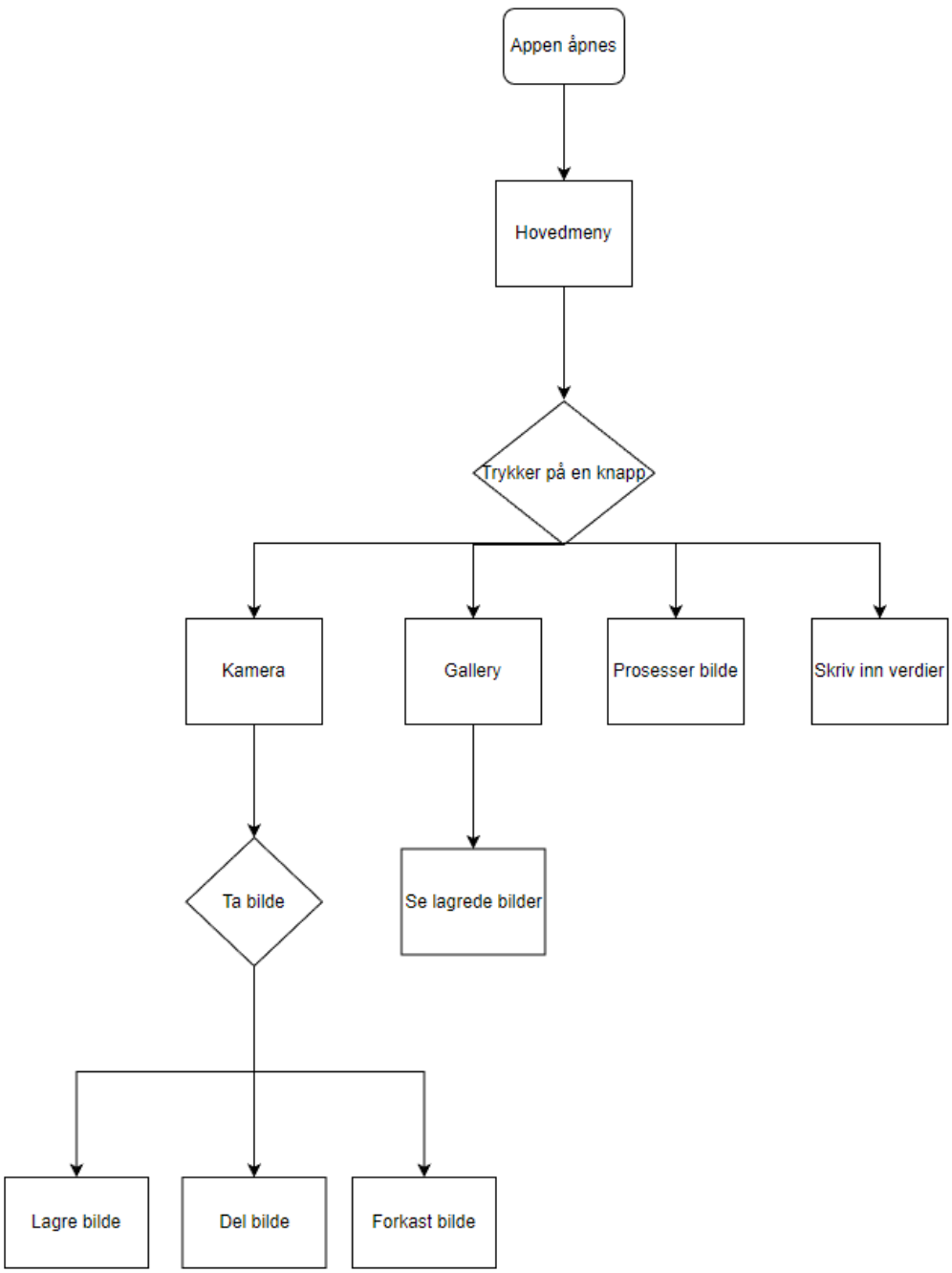
N2 diagrammet komplementerer det funkjsonellle flytdiagrmmet med informasjon om inputs og outputs. Den understreker dermed hvilke data som må samles for å utføre funksjonene. Funksjonene er plassert diagonalt i firkanten. Input-parametere er plassert vertikalt i relasjon til funksjonen og output-parameterene er plassert horisontalt i forhold til funksjonen. Som vist i diagrammet vil outputs fra én funksjon også kunne være inputs i andre funksjoner. Disse er interne inputs/outputs og er markert med fiolett farge. Eksterne inputs er markert med rød farge og eksterne outputs er blåfarget.

9.4 Pugh-matrise

Pugh-matriser blir brukt som verktøy i prosjektarbeidet. En Pugh-matrise illustrer fordeler og ulemper ved hvert konsept sammenlignet med hverandre. Hensikten med pugh-matrisen er å kartlegge hvilke kriterier som er aktuelle og hvilken prioritet disse har i relasjon til hverandre. En pugh-matrise vil aldri være absolutt objektiv. Det er derfor viktig at man ikke misbruker pugh-matrisen ved å støtte opp under forhåndsbestemte valg. Den skal ikke velge konsepter for oss, men veilede gruppen i hvilke kriterier som er viktigst å prioritere, og hvilket konsept som samsvarer med disse prioriteringene.

9.5 UX flowchart

Her er et brukerflytdiagram som viser trinnene i prosessen for hvordan appen fungerer. Dette er nyttig for å få en oversikt over appens funksjonalitet. Det hjelper også til å danne et bilde av hvordan appen fungerer fra brukerens perspektiv, og eventuelt identifisere områder for forbedring. [4]



Figur 17: Flowchart av app

9.6 Designkriterier

Hensikten med den konseptuelle designfasen er å undersøke alle mulige løsninger, og avgjøre hvilke(n) av disse som er mest gunstig å videreutvikle. Det konseptuelle designet skal utføres i samsvar med stilte krav og ønsker fra kunden. Det er hensiktsmessig at systemet under utvikling (SUU) er enkel, kjapp og presis å bruke. Presisjon, enkelhet og effektivitet er de viktigste parameterene fra brukerens perspektiv. Konsepter blir vurdert med pugh-matriser med tanke på designkriteriene listet under. Designkriteriene er en forenkling av kravspesifikasjonen som sikrer at det mest gunstige konseptet for design videreføres til neste steg i utviklingen. Følgende kriterier og begrunnelse for disse står opplistet under.

9.6.1 Effektivitet

De funksjonelle kravene legger føringer for hvor lang tid systemet kan bruke på å utføre oppgaver. Det er viktig for brukeren at man bruker fem minutter eller mindre på å kalibrere målskiven. Dermed vil mulige løsninger som bruker kortere tid enn andre til å utføre de samme oppgavene være mer hensiktsmessige å velge. Samtidig må det vurderes hvorvidt risikoen for andre parametere øker, dersom man vil prioritere en effektiv tidsbruk.

9.6.2 Presisjon

De funksjonelle kravene avgjør hvor nøyaktig systemet skal være. For eksempel beskriver krav FK-A8 at systemet skal detektere treffpunkter med en nøyaktighet på 2 mm. Dersom systemet skal være aktuelt for KTS er det viktig å prioritere presisjon fremfor andre kriterier. Den totale presisjonen skal være innenfor 2 mm, og må beregnes ved å legge sammen den totale unøyaktigheten som kan oppstå i hvert ledd av systemet.

9.6.3 Automasjon

For å ivareta høy presisjon og effektivitet vil man unngå manuelle prosesser så stor grad det lar seg gjøre. Dermed er automasjon et påfølgende kriterium å vurdere i hvert konsept.

9.6.4 Brukervennlighet

Brukervennlighet handler om interaksjonen mellom brukeren og systemet, og bør være smidig. Systemet skal ikke være ubeleilig for skytteren å bruke. Det er også viktig at systemet kan opereres av én vilkårlig person med normal funksjonsevne. Vekt er derfor et kriterium som må tas i betraktning. For å sikre god brukervennlighet bør systemet kunne brukes intuitivt av brukeren.

9.6.5 Kostnader

KTS ønsker at systemet skal ha lav produksjonskostnad. Det må derfor tas hensyn til masseproduksjon i design og produksjon av systemet. Systemet må være designet på en slik måte at kostnadene knyttet til produksjon minimeres. Dersom systemet har smarte, færre eller mindre fysiske bestander, vil kostnader knyttet til produksjon være lave.

9.6.6 Gjennomførbarhet

I forbindelse med varigheten på bachelorprosjektet, arbeidsmengden som kreves, og teknologiske begrensninger, må det vurderes hvorvidt et konsept er aktuelt å gjennomføre. Gjennomførbarhet som kriterium tar interne og eksterne begrensninger i betraktning. Den generelle gjennomførbarheten bør derfor vektlegges sterkt i vurderingen av et konsept.

10 Design av programvare

Designet av

10.1 Konsepter for bildebehandlingsløsninger

10.1.1 Ide 1

Denne ideen går ut på at man tar bilde av skyteskiven og algoritmen gir verdi til hver eneste piksel. Videre så kan man ta bilde etter skyte runden og de pikslene som har fått en forandring vil da bli markert som treffpunkter. Dermed kan algoritmen gjenkjenne alle piksler sammen med deres verdier men forandringen som har skjedd på noen av pikslene vil da ikke ha de verdiene og dermed kan de koordinatene markeres som skytepunkter.

Fordeler og ulemper

- Det er svært vanskelig å tilrettelegge for at bildet er likt mellom hvert bildet som er tatt . Det kan føre til at forskjellen mellom de to bildene fører til mange feilmarkerte punkter.
- Dersom bildets orientering er fullstendig likt før og etter skuddtreffene, vil metoden presist kunne detektere punkter.

10.1.2 Ide 2

Denne ideen går ut på å bruke «Deep learning» til å gjenkjenne treffpunkter. Det vil si at man mater algoritmen med forskjellige bilder av skyteskiven også markerer man selv hvor treffpunktene er. Etter en stor mengde med data så begynner maskinen å lære hvordan den skal se etter treffpunkter.

Fordeler og ulemper

- Deteksjon av treffpunktene vil være ganske nøyaktige når maskinen først har lært hva den skal se etter ut ifra hva den er lært opp av mennesker.
- Kan ta veldig langt tid å utvikle ettersom mennesker må markere en stor variasjon av treffpunkter for å lære opp maskinen.
- All dataen kan ta mye plass
- Det kan oppstå lysforhold som ikke er tatt høyde for i utviklingsprosessen.

10.1.3 Ide 3

Konseptet er definert ved at man markerer treffpunkter med markeringsobjekter som har en klar farge. Denne fargen skal algoritmen lete etter i hver pixel i bildet. Når den har funnet en pixel med en farge tilsvarende fargen på treffpunktet, vil man ha identifisert og posisjonert et treffpunkt. Det er kameraet som må kunne registrere fargen på lyset. Dvs. at treffpunktet ikke nødvendigvis behøver å være synlig for det blotte øyet. Det skyldes at et kamera kan registrere andre bølgelenger enn de som er vanlig å se med det blotte øyet. Konseptet er testet med algoritme i bildet under. Her ser man at tuppen av metallmarkøren er fargen som har blitt detektert av algoritmen og denne har blitt markert med grønn sirkel.

Fordeler og ulemper

- Farge oppfattes ulikt i forskjellig lysforhold og det er ikke sikkert fargen blir oppdaget i alle lysforhold.
- Algoritmen trenger ikke mye data for å finne treffpunkter.

Figur 18: Deteksjon av metall topp. Den grønne sirkelen markerer spissen, altså midten på punktet.

Figur 19: Deteksjon av treffpunkter på skyteskive

10.2 Vurdering av konsept

Dersom man skal ta en god avgjørelse på hvilken ide som er fornuftig å gå videre med er det viktig å vurdere hvert konsept opp mot hverandre, i forhold til de satte kriteriene og ressursene vi har tilgjengelig.

Ide 1: baserer seg på at man tar et bilde to ganger; før og etter skyting. Samtidig er man avhengig av at bildet er likt hver gang. Ved testing kom vi frem til at bildet ikke blir likt hver gang, selv med et stabilt oppsett. Dette kan skyldes mikroendringer i lysforhold eller posisjon til kamera. Da ville programvaren funnet treffpunkter der det faktisk ikke var tilfellet. Dessuten er det ikke åpenbart hvordan man skulle klare å skille falske treffpunkter med virkelige treffpunkter basert på denne ideen. Dette blir praktisk svært vanskelig å gjennomføre sammenlignet med de andre ideene som kun trenger et bildeinput etter skyting er gjennomført.

Ide 2: bygger på maskinlæring. Dette vil kreve store mengder data og en stor tidsbruk involvert i å trene opp maskinen. Presisjonen vil påvirkes av hvor stort utvalg av data man har. Siden målskiven skal brukes under svært varierende forhold, vil treffpunkter kunne variere mye i størrelse, fasong og generelt utseende. Derfor vil det være svært utfordrende å samle inn nok variert data. Sammenlignet med de andre 2 ideene, og tidsressursene vi har tilgjengelig, vil ikke ide 2 være et gunstig valg å gå videre med.

Ide 3: baserer seg på pikseltelling, noe som er ofte brukt i algoritmer for bildebehandling. Den er avhengig av at en bruker plasserer fysiske markører på en presis måte, og brukeren vil være avhengig av hjelpemidler for dette. Denne ideen skiller seg fra de andre ved at den er praktisk enkel for brukeren, og baserer seg på allerede eksisterende metoder. Samlet sett, vil ide 3 kunne innfri på de viktigste kriteriene, som presisjon, effektivitet og enkelhet. Samtidig er ideen noe som er praktisk gjennomførtbart ila. perioden arbeidet skal utføres i.

Med bakgrunn i testing og utforskning av de forskjellige ideene, er det bestemt at ide 3 leder oppgaven vår i riktig retning, for både kunden, KTS, brukeren, og bachelorgruppen vår. Derfor vil ide 3 - pikseltelling, være designkonseptet som programvaren skal baseres på. Med dette bestemt vil det videre designet av systemet være bygget på prinsippet i ide 3 - pikseltelling.

11 Aktuelle faktorer ved punktdeteksjon

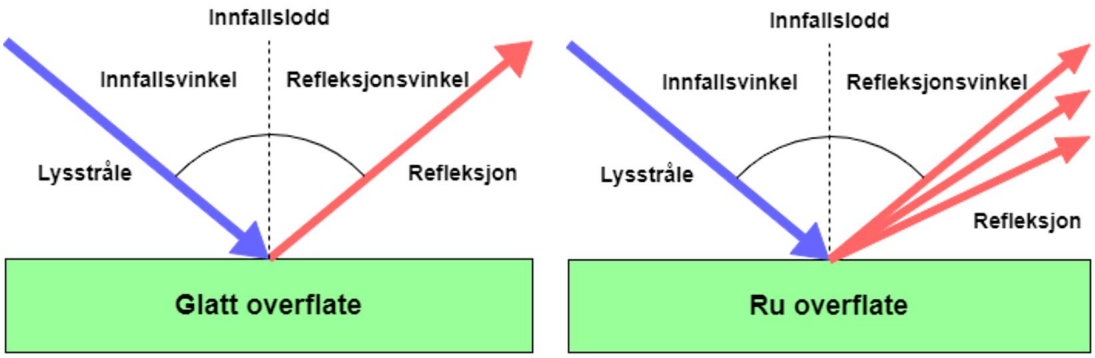
Et punkt skal kunne detekteres presist. Det er konkludert gjennom undersøkelse at skuddhull ikke kan identifiseres uten at disse er markert. Skuddhullene er svært vanskelige å detektere siden både sikteblinken og skuddhullene er svart. En større kontrast er nødvendig for å detektere treffpunktene. Derfor må enhver fysisk markering ha en utforming som muliggjør deteksjon av treffpunkter. Fasongen og overflaten av en gjenstand bestemmer hvordan lys reflekerer fra gjenstanden. I tillegg vil andelen lys i omgivelsene påvirke lysmengden i kameraet, og hvordan denne oppfatter farger. For å kunne detektere en markør presist, er det viktig at det er nok lys som reflekteres fra markøren. Optikk, altså læren om lys, vil kunne belyse hvilke betingelser som må stilles til en fysisk markør. I tillegg vil fysiske tester utføres for å undersøke hvordan fasongen av en overflate påvirker lysrefleksjonen.

11.1 Optikk

Lys reflekteres slik som beskrevet i figur 11.1. Lysstrålen treffer overflaten og reflekteres ut i én eller flere lysstråler avhengig av ruheten og fasongen til objektet. Derfor er refleksjonen av farger ustabil. En farge vil oppfattes som en fargegradient avhengig av overflaten den treffer. Lysmengden avgjør også hvor synlig fargen er. Dette kan være problematisk dersom en algoritme skal filtrere ut en farge som man ønsker å finne. Dersom overflaten er både glatt og kuleformet, vil det kunne dannes et gjenskinn, og stor variasjon i fargegradienten.

Dette kan illustreres med figur 11.1 hvor refleksjonen av en kule blir reflektert i mange retninger det punktet på kula som er vinkelrett i forhold til menneskets øye er det punktet som vil skinne lyst farge. De andre punktene på kula vil da

synes mørkere av samme fargen som kula har. Det er også verdt å merke seg refleksjon av farge kan oppfattes noe svakere dersom man ikke ser punktet med fra utfallsvinkelen som skal være lik innfallsvinkelen i forhold til normalen. Dette kan illustreres i figur Refleksjonsvinkel".

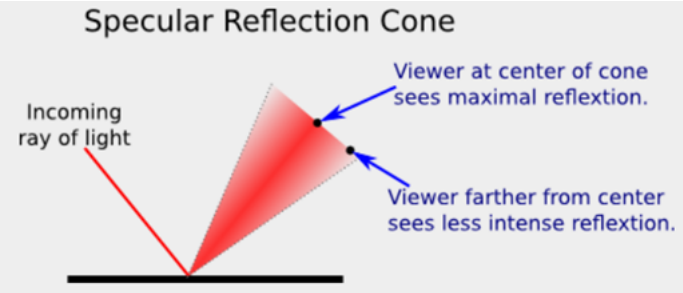


Figur 20: Lysrefleksjon på glatt versus ru overflate.

Figur 21: Lys refleksjon av en kule

11.2 Overflate

Et annet faktor som kan påvirke hvordan fargen på hver piksel oppfattes er om overflaten lyset treffer på er rett eller hakket. Dette kan gjøre at punktet som

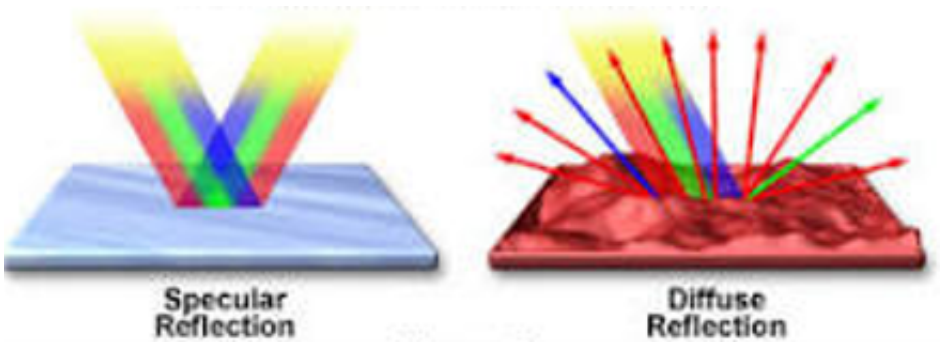


Figur 22: Langtidssykdom
[5]

reflekteres kan ha flere forskjellige farger fra ulike perspektiver. Videre så er det viktig å påpeke at dersom overflaten er glatt og lyset som treffer punktet har flere bølgelengder innenfor rød, grønn, blå bølgelengde så kan fargen oppfattes som hvit. [6] Dette kan man se på illustrasjonen til venstre i figur 23. Refleksjon av farger kan også påvirkes dersom punktet lyset treffer er hakkete da får man Diffus refleksjonsom igjen kan reflektere ulike bølgelengder som kan oppfattes som ulikt farge avhengig av vinkelen man ser det fra(Se Diffuse Reflection"ved figur 23).Dette kan skape et problem når man skal sjekke gjennom hver piksel og matche piksel fargen med en vis farge. Derfor må både materialet som blir brukt være godt gjennomtenkt og hvordan refleksjonen av markørene oppfattes i forhold til kamera.

11.3 Intervaller

For å kunne gjenkjenne farger man vil finne er det flere mulige løsninger både med programmering og ved å tilpasse markørene når de produseres. Dersom man baserer seg på programerings løsningen så bør man bruke intervaller for fargen som søkes etter i hver piksel. Så dersom man ser etter fargen hvitt så bør RGB verdien dekke majoriteten av hvite RGB verdier. Som for eksempel fra og med (225,225,225) til (255,255,255) ettersom hvitt farge er ca lik blanding av disse fargene.



Figur 23: **Specular Reflection** viser hvordan lys med flere bølgelengder reflekterer fra en glatt overflate. **Diffuse Reflection** viser hvordan lysstrålene reflekterer fra en ru overflate. Lysstrålene går i mange varierende retninger avhengig av topografien til overflaten.

[5]

11.4 Deteksjon av RGB farger

RGB-verdier representerer fargene rød, grønn og blå, og brukes ofte til å beskrive piksler som kvadratiske eller rektangulære områder i et bilde. Disse verdiene er angitt gjennom tre kolonner [Rød, Grønn, Blå]. Ved å bruke RGB-verdier kan vi identifisere markører på en skyteskive. Imidlertid er det viktig å merke seg at farger i virkeligheten ikke alltid gjengis nøyaktig på bilder. Fargene har intervaller som varierer basert på flere faktorer, inkludert fargeintensitet, lysforhold, avstand til objektet, oppløsning og algoritmer brukt av forskjellige telefon kameraer.

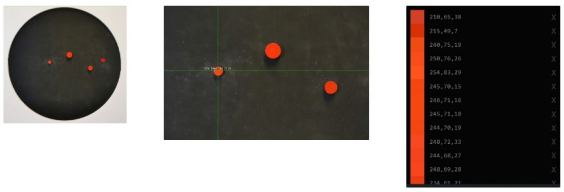
For å undersøke disse effektene, vil vi utføre tester med ulike kameraer, oppløsninger, farger, avstander og lysstyrker. Vi vil dokumentere de oppnådde dataene og lage tabeller som viser intervallene. Disse intervallene vil gi innsikt i størrelsen på fargeintervallene og terskelverdiene som oppdages. Det er viktig å være oppmerksom på to kritiske punkter i denne sammenhengen: større intervaller og inkonsistente intervaller kan føre til feilaktig deteksjon av piksler og skape utfordringer når man bruker ulike mobilkameraer.

For å utforske denne effekten vil vi ta et bilde og skrive det ut ved hjelp av matplotlib-biblioteket. Deretter vil vi manuelt undersøke RGB-verdiene og identifisere et intervall. Vi vil se etter forskjellen mellom den laveste og høyeste verdien for hver av fargene R, G og B, og sammenligne dem for å få innsikt i fargestyrken.

11.4.1 Farge og RGB, Realitet og Digital

For å kunne identifisere RGB-fargene i et bilde, kreves det en forståelse av hvordan RGB-farger fungerer i både den virkelige og digitale verden. Selv om en farge kan virke som en ren rød farge basert på menneskelig observasjon, er dette ikke nødvendigvis realiteten. Tidligere forskning har vist at farger er basert på kombinasjoner av rød, blå og grønn.

For å utvikle en dypere forståelse av dette fenomenet, har vi tatt et bilde av en skyteskive med røde markører og analysert det i et program. Dette gjøres med hensikt å undersøke og forstå hvordan observasjon av virkeligheten og det digitale påvirkes av farger.



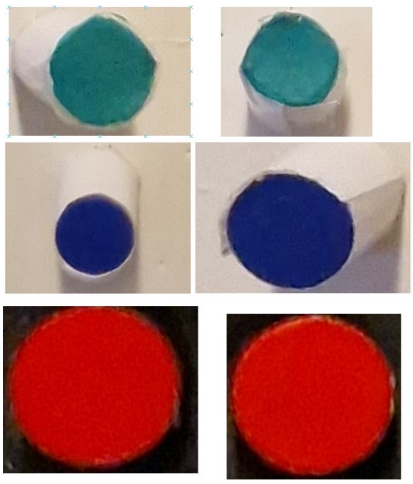
Vi har benyttet oss av programvaren PixSpy (https://pixspy.com/) for å analysere RGB-verdiene til ulike piksler i et bilde. Ved hjelp av dette verktøyet har vi undersøkt forskjellene mellom pikslene i den venstre, røde markøren. Resultatene, som illustreres i bilde til høyre, viser at pikslene som er observerbart røde, faktisk har forskjellige RGB-verdier. Dette fenomenet bekrefter at farger i den virkelige verden oppfattes som ulike RGB-verdier, til tross for at de kan observeres som røde.

11.5 Effekten av fargestyrke

RGB-farger vurderer styrken av en farge basert på to kriterier: lysstyrken eller mørkheten til fargen. Innenfor det RGB-fargemodellen har hver farge en minimums- og maksimumsverdi som spenner fra 0 til 255. Jo høyere verdien er, desto lysere blir fargen. I denne sammenhengen representeres fullstendig hvit som [255, 255, 255], mens svart representeres som [0, 0, 0]. Vi utnytter denne egenskapen for å oppnå en forståelse av hvor lys eller mørk en farge blir observert.

Ved å utnytte denne observasjonen kan vi velge markørfarger som kan detekteres uten at andre farger blir detektert. Dette er viktig for å skape en tydelig skillelinje mellom markørene og omgivende farger. Gjennom å utnytte

de maksimale og minimale verdiene innenfor RGB-fargemodellen kan vi tilpasse fargevalget slik at markørene skiller seg tydelig ut og kan detekteres med presisjon.

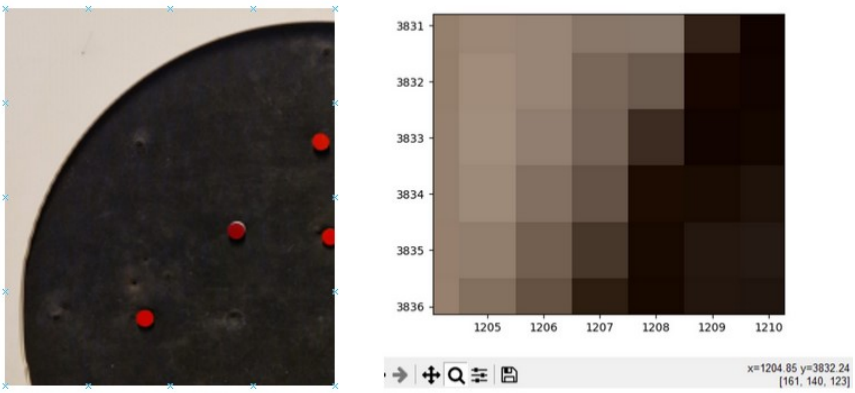


Farger	Markør 1	Markør 2	Total interval
Mørke Blå	Min = [13, 18, 83] Max = [36, 56, 115]	Min = [12, 26, 83] Max = [53, 60, 109]	sammen slått = [12-53, 18-60, 83-115]
Grønn	Min = [28, 94, 101] Max = [51, 139, 140]	Min = [13, 94, 82] Max = [61, 135, 120]	sammen slått = [13-61, 94-135, 82-140]
Lyse rød	Min = [133, 0, 0] Max = [206, 33, 22]	Min = [118, 1, 16] Max = [214, 36, 22]	sammen slått = [130 - 214, 1-36, 0-22]
Mørke Rød	Min = [96, 1, 0] Max = [151, 30, 30]	Min = [85, 9, 5] Max = [155, 28, 28]	sammen slått = [85-155, 1-30, 0-30]

To essensielle farger som forekommer i bilder brukt til bildebehandling av skyteskiver er hvit og svart. Basert på forskjellige lysstyrker og bildebehandlings algoritmer på ulike mobile enheter, kan gjennomsnittlige RGB-verdier for hvit farge variere i området [170-255, 170-255, 170-255], mens svart farge kan variere i området [0-60, 0-60, 0-60]. Disse verdiene er et resultat av omfattende testing med store datasett av bilder.

Det er fornuftig å bestemme RGB-verdiene for markørene, da de skal skille seg ut fra andre farger i bildet. De viktigste fargene som markørene må skille seg fra er hvit og svart. Selv om forskjellen i RGB-verdiene mellom svart og hvit er

betydelig, oppstår det en spesifikk utfordring. Problemet oppstår når det skjer en overgang mellom hvitt og svart, eller omvendt, innenfor bildet. Dette blir spesielt tydelig i periferien av den svarte sirkelen. Et eksempel på dette problemet er illustrert i bildet nedenfor



Disse bildene illustrerer en overgang som skaper et problem, da det resulterer i et intervall i RGB-verdiene som ligger i nærheten av fargeintervallene. Dette kan føre til utfordringer med å nøyaktig detektere og skille mellom markører og omkringliggende farger. Tabellen under viser intervallet som oppstår.

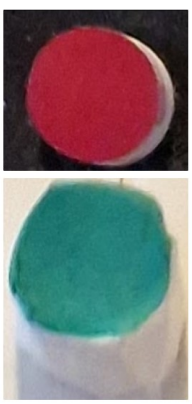
Farge	Piksel interval	Total Interval
Svart&Hvit	Min = [18, 4, 0] Max = [161, 140, 123]	sammen slått = [18-161, 4-140, 0-123]

Målet er å oppdage piksler som tilhører markørene og ingen andre områder. Ved å kombinere tabellene for fargede markører og svart/hvitt, observerer vi at bruken av visse farger kan resultere i at algoritmen ikke bare detekterer de fargede markørene isolert sett. Forskjellen i intervaller mellom farger og overgangen mellom hvitt og svart er at RGB-verdiene jevnt endres i pikslene mellom hvitt og svart. Dette innebærer at en piksel i overgangen mellom rødt, grønt og blått har en gjennomsnittlig terskelverdi på 0 til 40, og dermed er overgangen veldig jevn. Basert på denne observasjonen kan vi vurdere hvilke farger som er mest hensiktsmessige å bruke basert på fargestyrken.

Mørk blå og grønn er farger som kan føre til deteksjonsfeil mellom forskjellige farger og svart/hvitt. Basert på fargetabellen er terskelverdien mellom RGB-verdiene for mørk blå og grønn nær 40. I motsetning til dette har røde farger en høy rødverdi og lave blå og grønne verdier. Sannsynligheten for å detektere noe annet enn røde farger er minimal basert på de viste fargene.

11.5.1 Effekten av Oppløsning

Bilder med høyere oppløsning inneholder en større mengde pikseldata. Dette resulterer i mer detaljerte og tydelige bilder. Imidlertid kan det oppstå problemer når man reduserer detaljnivået, da piksler slås sammen og skaper større fargeintervaller. Bildene nedenfor viser bilder av forskjellige markører sammenlignet med det opprinnelige bildet tatt med en Samsung S20 i ulike oppløsninger. De brukte oppløsningene er 50 MP, 12 MP og 9 MP.



Samsung s20 - 50MP - 0.65m

Samsung s20 - 12MP - 0.65m

Samsung s20 - 9MP - 0.65m

Ved å analysere dataen i tabellen kan vi konkludere med at ulike oppløsninger fører til endringer i fargeintervallene. Den grønne markøren viser en rødfarge med en terskelverdi som varierer fra 80 til 51 ved overgang fra 50 MP til 9 MP oppløsning. For den mørkerøde markøren endrer terskelverdien seg fra 44 til 9 i den grønne kolonnen når oppløsningen går fra 12 MP til 9 MP.

Oppløsning	Farge	RGB Intervaller	Total interval	Terskelverdi for R, G, B
50MP	Mørke rød	Min = [137 ,20, 45] Max = [177, 40, 74]	slått sammen = [137-177, 20-40, 45-74]	[40, 20, 29]
12MP	Mørke rød	Min = [129, 25, 68] Max = [179, 69, 79]	slått sammen = [129-179, 25-69, 68-79]	[50, 44, 11]
9MP	Mørke rød	Min = [124, 35, 38] Max = [165, 44, 63]	slått sammen = [124-165, 35-44, 48-64]	[41, 9, 16]

Oppløsning	Farge	RGB Intervaller	Total interval	Terskelverdi for R, G, B
50MP	Grønn	Min = [26, 118, 103] Max = [106, 170.160]	slått sammen = [26-106, 118-170, 103-160]	[80, 52, 57]
12MP	Grønn	Min = [17, 106, 116] Max = [80, 160, 160]	slått sammen = [17-80, 106-160, 116-160]	[63, 54, 44]
9MP	Grønn	Min = [24, 118, 94] Max = [75, 164, 151]	slått sammen = [24-75, 118-164, 94-151]	[51, 46, 57]

11.5.2 Forskjellige mobilkameraer, forskjellige resultater

Basert på de forskjellige bildene vi har tatt ved hjelp av ulike mobilkameraer, har vi observert at det eksisterer ulike algoritmer som tolker RGB-verdiene på forskjellige måter. For å teste dette, utførte vi eksperimenter ved å bruke ulike kameraer plassert 0.65 meter fra objektet, med en oppløsning på 12 MP og under samme lysforhold.

Bildene og tabellen illustrerer nettopp dette fenomenet:

Iphone 11

Samsung S20

Samsung S20 FE

Telefon	Farge	RGB Intervaller	Total interval	Terskelverdi for R, G, B
Iphone 11	Lyse rød	Min = [185, 29, 8] Max = [215, 50, 41]	slått sammen = [185-215, 29-50, 8-41]	[30, 21, 33]
Samsung S20	Lyse rød	Min = [195, 14, 5] Max = [223, 50, 51]	slått sammen = [195-223, 14-50, 5-51]	[28, 36, 46]
Samsung S20 FE	Lyse rød	Min = [192 , 60, 50] Max = [232, 85, 104]	slått sammen = [192- 232, 60-85, 50-104]	[40, 25, 54]
Iphone 11	Mørke blå	Min = [17, 26, 145] Max = [47, 57, 184]	slått sammen = [17-47, 26-57, 145-184]	[30, 31, 39]
Samsung S20	Mørke blå	Min = [26, 37, 121] Max = [58, 67, 163]	slått sammen = [26-58, 37-67, 121-163]	[32, 30, 44]
Samsung S20 FE	Mørke blå	Min = [65, 74, 183] Max = [89, 97, 211]	slått sammen = [65-89, 74-97, 183-211]	[24, 23, 28]

Ved å analysere dataene i tabellen kan vi observere en minimal forskjell i totale intervaller mellom iPhone 11 og Samsung S20. Derimot viser Samsung S20 FE en betydelig større forskjell sammenlignet med de to andre telefonene. Når det gjelder lyse røde markører, faller RGB-verdiene i den blå kolonnen mellom 5 og 51 for både iPhone 11 og Samsung S20. For Samsung S20 FE er intervallet derimot mellom 50 og 104.

Vi ser tilsvarende resultater for mørkeblå farger i både den røde og grønne

kolonnen. iPhone 11 og Samsung S20 viser en minimal forskjell, mens Samsung S20 FE har en betydelig større avstand i verdier sammenlignet med de to andre telefonene.

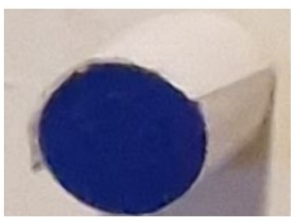
11.5.3 Hvordan lys påvirker farger

For å undersøke hvordan lys påvirker farger, benytter vi en skyteskive med markører i ulike farger. Formålet er å utvikle en algoritme som kan detektere markører på en skytebane under varierte naturlige lysforhold. Siden sollyset er ustabilt og stadig i endring, kan vi forvente at RGB-verdiene vil bli observert på forskjellige måter avhengig av mengden lys som belyser markørene. Bildene nedenfor viser en skyteskive med markører plassert på samme måte, men under ulike lysforhold, ved hjelp av en Samsung S20 og en avstand på 0,65 meter fra objektet.

inne i rom i-203

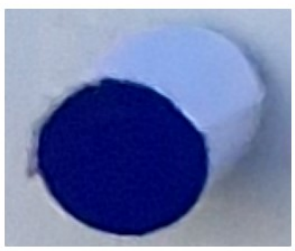
Ute i sollys

Ute, men dekket av skygge



inne i rom i-203

Ute i sollys



Ute, men dekket av skygge

Lys	Farge	RGB Intervaller	Total interval	Terskelverdi for R, G, B
Lys fra rom i-203	Lyse rød	Min = [175, 1, 0] Max = [210, 23, 15]	slått sammen = [175-210, 1-23, 0-15]	[35, 22, 15]
Ute i sollys	Lyse rød	Min = [142, 0, 0] Max = [210, 14, 16]	slått sammen = [142-210, 0-14, 0-16]	[68, 14, 16]
Ute i skygge	Lyse rød	Min = [160, 0, 1] Max = [183, 33, 31]	slått sammen = [160-183, 0-33, 1-31]	[23, 33, 30]
Lys fra rom i-203	Mørke blå	Min = [17, 22, 70] Max = [36, 53, 110]	slått sammen = [17-36, 22-53, 70-110]	[19, 31, 40]
Ute i sollys	Mørke blå	Min = [0, 0, 73] Max = [32, 23,117]	slått sammen = [0-32. 0-23, 73-117]	[32, 23, 44]
Ute i skygge	Mørke blå	Min = [5, 4, 83] Max = [12, 23, 110]	slått sammen = [5-12, 4-23, 83-110]	[7, 19, 27]

Lys påvirker hvordan farger oppfattes, og det er av betydning i denne forskningen å undersøke stabiliteten til RGB-verdiene under ulike lysforhold. Ved å analysere dataen vi har samlet inn for mørkeblå og lysrøde markører under forskjellige lysforhold, kan vi trekke konklusjoner om lysets effekt. Effekten av lys på RGB-farger viser seg å være moderat, med små forskjeller i både totalintervall og terskelverdier for samme markører.

Likevel observerer vi fortsatt forskjeller. Den største forskjellen oppstår i lysrøde markører under sollys sammenlignet med andre lysforhold. Her ser vi at RGB-verdiene viser en betydelig forskjell med en terskelverdi som varierer fra 23 til 68. Generelt sett holder mørkeblå markører en stabil intervall, men også her ser vi en liten forskjell. Under sollys har de en terskelverdi på 32, mens den reduseres til 7 i skyggen.

11.5.4 Effekt av avstand på RGB-verdier

Når et bilde blir tatt med et kamera, bestemmes antallet piksler på bildet av oppløsningen. Imidlertid er antallet piksler på et objekt i bildet også avhengig av avstanden. Et objekt med mange detaljer vil kreve at mobilkameraet er plassert nært. Avstanden er derfor en faktor som påvirker tilstedeværelsen eller fraværet av detaljer på objektet, og dette er noe vi ønsker å utforske nærmere. Ved å bruke dataen fra en Samsung S20 med en oppløsning på 50MP og ulike

avstander, vil vi opprette en resultatstabell. Tabellen vil inneholde intervaller for RGB-farger og vise hvordan intervallene endres basert på avstanden. Bildene nedenfor illustrerer markørene vi har brukt for å samle inn data.

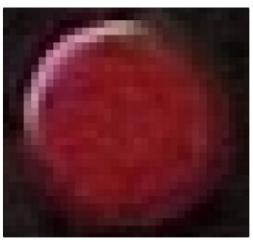
0.65m avstand fra objektet

1.5m avstand fra objektet

2.5m avstand fra objektet

0.65m avstand fra objektet

1.5m avstand fra objektet



2.5m avstand fra objektet

Avstand	Farge	RGB Intervaller	Total interval	Terskelverdi for R, G, B
0.65m	Mørke rød	Min = [150, 11, 13] Max = [178, 32, 41]	slått sammen = [150-178, 11-32, 13-41]	[28, 21, 28]
1.5m	Mørke rød	Min = [82, 0, 19] Max = [150, 35, 56]	slått sammen = [82-150, 0-35, 19-56]	[68, 45, 37]
2.5m	Mørke rød	Min = [72, 14, 21] Max = [194, 85, 161]	slått sammen = [72-194, 14-85, 21-161]	[126, 71, 140]
0.65m	Grønn	Min = [0, 99, 105] Max = [50, 115, 147]	slått sammen = [0-50, 99-115, 105-147]	[50, 16, 42]
1.5m	Grønn	Min = [0, 28, 40] Max = [42, 93, 99]	slått sammen = [0-42, 28-93, 40-99]	[42, 65, 59]
2.5m	Grønn	Min = [3, 52, 54] Max = [66, 82, 92]	slått sammen = [3-66, 52-82, 54-92]	[63, 30, 44]

Basert på tabellen ovenfor kan vi observere at avstanden kan ha en innvirkning på RGB-fargene og det nødvendige intervallet for å detektere disse pikslene. Vi kan se at ved en avstand på 0.65m er fargene stabile med et standard intervall og liten terskelstørrelse. Imidlertid, når bildet er tatt fra en avstand på 1.5m og 2.5m, ser vi drastiske endringer i verdiene. Terskelverdiene i den røde

kolonnen går fra 28 til 68 til 128. Det er også store endringer i den grønne og blå kolonnen. Den grønne markøren viser også forskjeller i intervallet, fra 16 til 65 til 30. Dette konkluderer med at avstanden har en betydelig effekt på fargene.

11.5.5 Konklusjon

Ved å samle inn ulike datasett og utføre undersøkelser om hva som påvirker RGB-farger, har vi observert store intervaller og inkonsekvente verdier som oppstår. Hver enkelt undersøkelse har enten hatt marginale eller betydelige effekter på RGB-intervallene. Imidlertid har vi ikke undersøkt størrelsen på intervallene når vi kombinerer alle disse inkonsekvente effektene. Lys, avstand, farge, oppløsning og RGB-algoritmer for hvert enkelt kamera er faktorer som påvirker bildetakingen. Dermed kan vi hevde at det å detektere RGB-farger for alle kameraer er en kompleks oppgave som ikke kan utføres enkelt. I denne oppgaven vil vi begrense oss til lys og avstand for å utvikle en algoritme som fungerer ved avvik.

12 Tekniske utfordringer

12.1 Forenklet illustrasjon av oppgaven

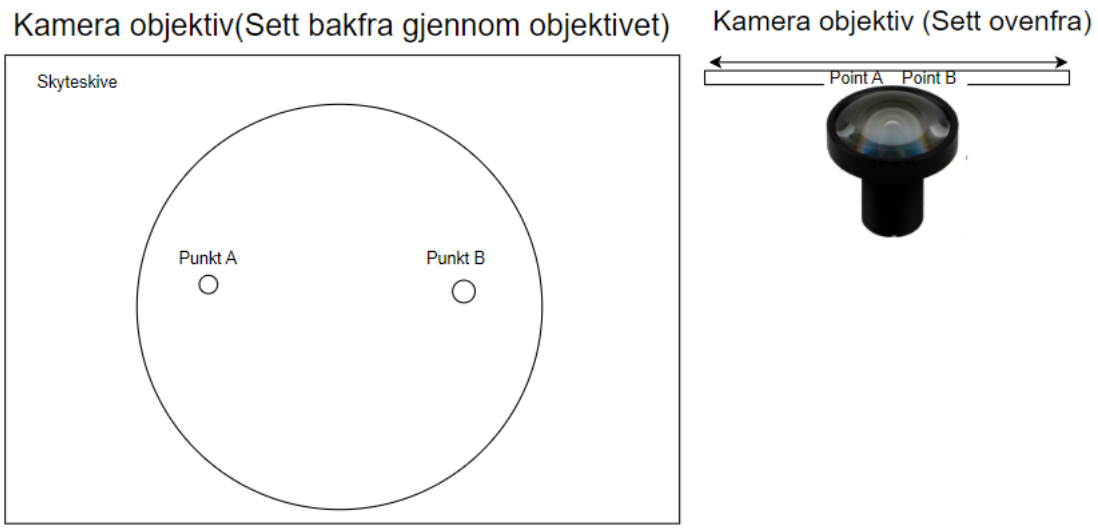
For å forenkle oppgaven kan man kan se for seg at kameraet er helt inntil skyteskiven. Se på figur(Linse inntil)24. Punktet A og B er punkter som man har markert med fargede markører slik at man kan oppdage dem i den digitale verden. Disse punktene kan brukes til å beregne avstanden mellom dem i piksler. Videre kan man også beregne avstanden mellom disse fysisk. Hvis man antar at avstanden mellom disse punktene er 10mm i virkeligheten og 500 piksler i den digitale verden, så kan man bruke dette til å beregne en faktor δy for hvor mye en piksel dekker vertikalt.

$$\delta_y = \frac{10\,\text{mm}}{500\,\text{piksel}} = 0.02\,\text{mm/piksel}$$

$$\delta_x = \frac{10\,\text{mm}}{500\,\text{piksel}} = 0.02\,\text{mm/piksel}$$

$$\sum_{i=0}^{\text{pxHorizontal}} \delta_{xi}$$

$$\sum_{i=0}^{\text{pxVertical}} \delta_{yi}$$



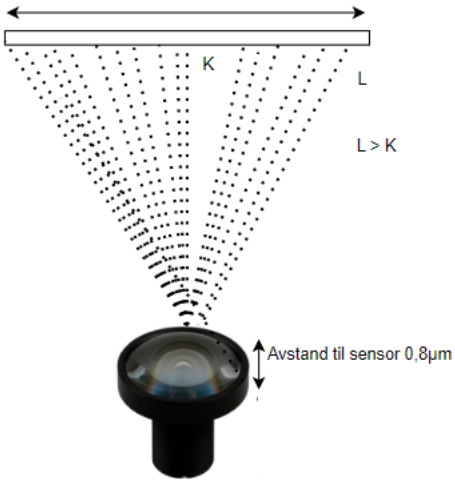
Figur 24: Linse inntil

Videre kan samme metode brukes til å beregne faktoren δx ved å ha to punkter C og D i horisontal retning. For enkelhetens skyld så kan man anta at det er samme avstand mellom punktet C og D både fysisk og i den digitale verden. Dette vil stemme overens med virkeligheten ettersom kameraet er helt inntil skyteskiven og kameraet er kvadratisk akkurat som skyteskiven. Det er fordi bilde som er blitt tatt er i 2D.

12.2 Areal dekkning av piksler

Den forenklede metoden kommer ikke til å stemme med virkeligheten dersom man øker avstanden mellom skyteskiven og kameraet. Se figur(Linse med en avstand i forhold til skyteskiven)25Dette kommer av at avstanden mellom hvert punkt i avbilda område og kameraet er ulik. Dette fører til at en piksel vil dekke større areal jo lenger punktet er fra skyteskiven. Ergo vil hverken δx eller δy være konstante verdier. Dette vil føre til at hver eneste piksel vil dekke forskjellige horisontale distanser og forskjellige vertikale distanser i virkeligheten.

Videre så vil også ytterste pikslene i det avbilda området dekke størst areal i

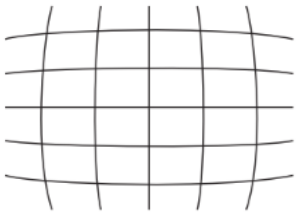


Figur 25: Linse med avstnd i forhold til skyteskiven

virkeligheten. For å konkludere denne utfordringen så er distansen mellom kamera og fotografiet en faktor når en skal beregne virkelighets avstand i den digitale verden.

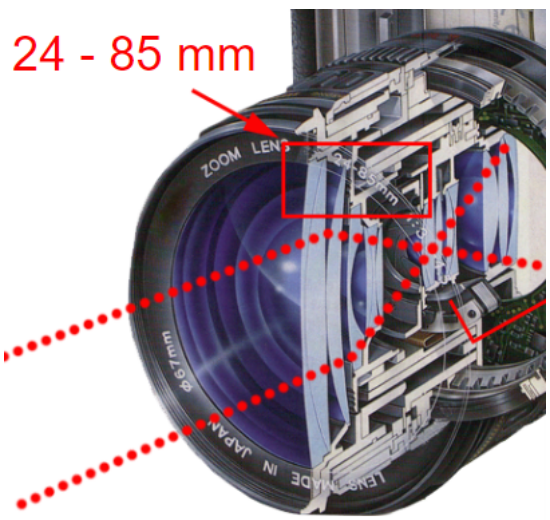
12.3 Barrel distortation

En annen utfordring som oppstår når det er ulik distanse mellom kamera og foto område er at det oppstår en vinkel mellom kamera og hver eneste piksel som ikke står normalt i forhold til kamera (se figur barrel distortation). Dette blir kalt for «Lens distortation». Dette fenomenet(linseforvrenning) er et avvik som oppstår av fra den ideelle projeksjonen [2] som er vurdert i hulkameramodellen[1]. Hvor mye avvik som oppstår avhenger mest av distansen til kamera i forhold til illustrasjons område, men brennvidden i kameralinsen er også en faktor på hvor mye linseforvrenning som oppstår. Denne formen for optiskaberrasjon fører til at illustrasjonen ikke blir korrekt, og linjene i bilde blir bøyd utover (Se figur2)[2]. For å gå dypere i denne problemstillingen er det viktig å vite hva brennvidde er, brennvidde er avstanden mellom det punktet i objektivet hvor lyset samles ifølge NDLA(Se



Figur 26: Barrel distortation

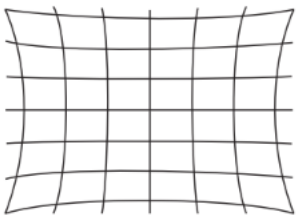
figur kamera objektiv). Jo mindre brennvidden er på en linse, jo mindre grad av forstørring, som det også kommer fram i NDLA[3]. Det er også viktig å legge merke til at jo nærmere objektet er kamera, jo mer «barrel distortation» oppstår det(se figur 5).



Figur 27: Kamera objektiv

Videre så er det også viktig å påpeke at dersom avstanden mellom kamera og bildeobjektivet øker og brennvidden er høy så kan et annet fenomenet «pincushion distortion opppstå»[2,side 32] dette kan man se på figur 3. Dette fører også til at område en piksel representerer blir forskjellige for hver piksel. Dette gjør at avstand mellom kamera og foto område er viktig og avstanden som representerer overgangen fra «barrel distortation» til «pincushion distortation» representerer det mest reelle bilde av virkeligheten. På en annen side så er det også viktig å påpeke at nøyaktigheten kan fortsatt holde mål

innenfor et intervall mellom overgangen av «barrel distoration» og «pincushion distortation».



Figur 28: Pincushion Distortation

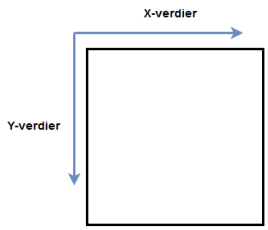
12.4 Mobil posisjon x-akse, y-akse og rotasjon om z-aksen

En annen faktor som påvirker hvor mye en piksel dekker i virkeligheten er mobil-kamera posisjonen i 3d-planet. Dette kan igjen påvirke avstanden mellom hvert punkt i foto område i forhold til kamera avhengig av hvilken akse mobilen roteres om.

12.5 Adskille markører ved deteksjons algoritme

Når algoritmen oppdager piksler på markørene, må vi ha en metode for å skille disse pikseldatasettene fra hverandre. Dette gjøres for å vite hvilket pikseldatasett som tilhører hvilken markør. Algoritmen løser dette ved å sammenligne alle oppdagede piksler med hverandre. Hvis avstanden mellom pikslene er større enn en manuelt angitt verdi i x- og y-retning, tilhører de ikke samme markør. Hvis avstanden er mindre enn den manuelle verdien, tilhører de samme markør.

Denne metoden for å skille pikseldatasett er relevant så lenge de ikke har felles y-verdier. Dette skyldes at metoden ser på forskjellen mellom pikslene. Algoritmen sammenligner forskjellen mellom det første pikselet og det neste som oppdages. Siden algoritmen leser gjennom x-verdiene før den hopper til neste y-verdi, vil den sammenligne to piksler i to forskjellige markører.



Figur 29: Hvordan algoritmen leser pikselene. Leser en hel rad av X-verdier, deretter tar den et nytt stygg i Y-verdi før den leser en helt ny rad av x-verdier.

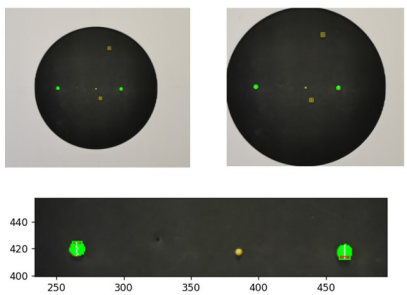
Hvis algoritmen sammenligner en piksel på kanten av en markør, vil den oppdage at pikselen tilhører en helt annen markør med en annen x-verdi. Imidlertid vil y-verdiene være identiske. Dermed vil algoritmen konkludere med at pikslene ikke tilhører samme markør. Dette problemet oppstår imidlertid ikke som en reell utfordring.

Figur 30: Hvordan algoritmen adskillte piksler fra hverandre.

Problemet oppstår når algoritmen oppdager at neste piksel ikke er i samme treffpunkt, og dermed antar at alle piksler den har funnet før i x-retning tilhører et markør. Dette resulterer i dannelse av separate grupper for hver markør. Problemet oppstår så lenge markørene har samme y-verdi. En tilsvarende situasjon ville oppstått for x-verdien hvis vi hadde lest gjennom alle y-verdiene i en kolonne først og deretter hoppet til neste kolonne hver gang y-verdien var ferdig behandlet. Bildene nedenfor illustrerer hvordan dette problemet ser ut i

bildebehandling:

Figur 31: Hvordan algoritmen grupperte piksler sammen før. I dette bilde illustreres problemet som oppstå hvis markører hadde piksler i samme Y-verdi.



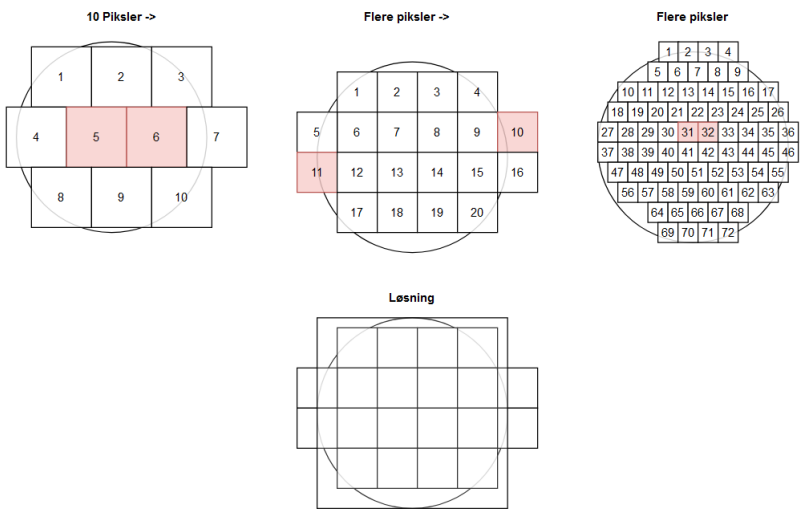
Figur 32: Hvordan algoritmen grupperte piksler sammen før. I dette bilde illustreres problemet som oppstå hvis markører hadde piksler i samme Y-verdi.

12.6 Metoder for å finne sentrum av markører

Algoritmen søker gjennom bildet med det formål å detektere piksler basert på et intervall av RGB-verdier. Når den har klart å detektere disse pikslene, lagrer vi posisjonsverdiene i en matrise (array). Deretter grupperer vi pikslene som tilhører samme markør og beregner medianen av antall piksler som er lagret i gruppen. Når medianen er funnet, kan algoritmen identifisere senterpikselen for markøren.

Selv om medianen av detekterte piksler kan beregnes, er det ikke nødvendigvis midtpunktet av markøren. Dette kan observeres i bildet nedenfor. Årsaken til dette er at ulike antall piksler kan resultere i ulike posisjoner for senterpikselen i forhold til markøren. Bildet nedenfor illustrerer hvordan dette kan oppstå når alle pikslene i en markør detekteres, der de rosa fargene representerer mulige medianverdier.

For å løse dette kan man øke presisjonen ved å ha en definert mengde piksler i et skytehull, tegne et rektangel rundt hele figuren og markere punktet som er midtpunktet. I eksempelet med 20 piksler kan utvalget være pikslene [7, 8, 13, 14].



12.7 Metoden for å ta hensyn til de tekniske utfordringene

Metoden som har blitt benyttet for å takle de tekniske utfordringene er å øke avstanden mellom kameraet og skyteskiven slik at kameraet dekker et større område. Dette resulterer i at punkter på skyteskiven ikke vil ha skarpe vinkler i forhold til kameraet. Som et resultat blir avstanden mellom pikslene i de ytre delene av skyteskiven redusert, slik at pikslene som dekker skyteskiven omtrent dekker samme areal. Dette muliggjør beregning av en koeffisient for

pikslene som indikerer hvor mye de dekker i henholdsvis x- og y-retning. Dette er basert på kjent fysisk avstand mellom aksekorsene og antall piksler mellom dem, som nevnt i forenkling av oppgaven". Videre fører denne tilnærmingen til en reduksjon i barrel distortation i bildene. Det er også viktig å merke seg at avstandene 2,45 meter og 2 meter som er valgt, representerer overgangen fra barrel distortation til pincushion distortion.

13 Kriterier for kamera

Med utgangspunkt i rammene rundt oppgaven og algoritmen så er det satt opp 6 kriterier for valg av kamera. Disse hovedpunktene er ISO sensorer, oppløsning, pris, tilgjengelighet og praktisk bruk, lukkerhastighet og blenderåpning.

13.1 RGB intervaller og ISO

RGB intervaller er en viktig del av oppgaven ettersom deteksjons algoritmen(Se bildebehandling) baserer seg på å finne treffpunktene ved å finne punkter med et gitt RBG intervall. Dette betyr at jo skarpere og tydeligere fargene er i de projeksjerte bildene, jo lettere er det å detektere treffpunktene. Dette gjør at ISO sensitiviteten til kamera en faktor. ISO er kameras sensitivitet til lys i følge adobe [7]. Videre er det viktig å påpeke at ISO sensitivitet er en del av eksponerings triangle"hvilke betyr ISO er en viktig faktor for bildekvalitet. For å konkludere, jo lavere ISO sensitivitet jo mindre lysstøy blir det på bildene ergo vil farger være mer klare.

13.2 Oppløsning

Oppløsning er en viktig kriterie med tanke på hvor mye en piksel dekker i virkeligheten(se punktdeteksjon). På bakgrunn av tester(se avviksanalyser) som er gjort så er 12MP og 108MP foretrukket. Det er fordi avvikene har vært innenfor A kravet når disse oppløsningene er brukt.

13.3 Kamerablenderåpning

Kamerablenderåpning er også en faktor for hvor klart bildekvaliteten blir som det også kommer fram i Nashville film institute" [8]. Kamerablenderåpning er

også en faktor i eksponerings trianglet" [8]. Dette kommer av at jo fortere kamerablenderen lukker seg å tar bilde jo mindre bevegelser blir det. Dette er en viktig faktor siden det er et menneske som tar bilde og da er stor sjanse for at det blir bevegelser før kamerablenderen lukker seg og bildet blir tatt.

13.4 Tilgjengelighet og Praktisk bruk

Det er to typer tilgjengeligheter som blir tatt med i betraktning for vurdering av kamera. Den ene er tilgjengelighet iform av marked. Den andre er tilgjengelighet ved bruk, og hvor lett det er å bruke kameraet ved bildetagning.

13.5 Pris

Pris er et viktig slutt kritereie for valg av kamera. Prisen påvirker kvalitetene av funksjoner tilgjengelig på kameraet. Funksjoner som har påvirkning på nøyaktigheten ved deteksjon av treffpunkter som nevnt ovenfor. Det er derfor viktig å vurdere pris på kameraet opp med hva hva slags funksjoner som er nødvendige for å oppnå nøyaktighetskravene gitt fra oppdragsgiver.

13.6 Valg av kamera

ved utvalg av kamera begrunnet av å finne treffpunkter ved bruk av bildebehandling er en enkel avgjørelse. Selv om begge har sine ulemper og fordeler der system kamera er et enkelt utvalg når det kommer til kvalitet av bilde på alle avstander, så oppnår telefon kamera det vi ønsker basert på komfor, praktisk bruk og muligheten til å finne treffpunkter på nær avstand. Prisene mellom begge varierer i henhold til kvalitet av bilder, men hovedmålet vårt er å tilby et produkt til kunde med minst mulig økonomisk bruk som kan fortsatt få jobben gjort. Dermed å spesifisere hvilket kamera som er det beste valget basert på nøyaktighet er noe vi må undersøke.

På bakgrunn av de økonomiske kostandene det vil medføre å kjøpe systemkamera ser vi det ikke hensiktsmessig å bruke det i bildebehandlingen. Selv om man kan få kameraer som kan detektere treffpunktene uten å bruke noen form for markører blir kostnaden på disse kameraene unødvendig høy. En annen faktor som spiller inn er hvordan algoritmen skal detektere treffpunkter ved bestemt RGB-verdi. Ut fra tester med gruppens mobiltelefoner er det viktig med en sterk og kontrastfull farge slik at algoritmen ikke mistolker farger. Dette betyr at det stilles krav til en form for fysisk markering for å synliggjøre treffpunktene. Videre ut fra testene ble det funnet ut at telefonkamera ikke kunne detektere treffpunktene. Dermed ble konklusjonen at vi må bruke fysisk hjelpemidler for å synliggjøre treffpunktene ved bildetagning. Bruk av mobiltelefon ved bildetagning er billigere og rask nok i henhold til kravene, og dermed det beste valget. (neste undersøke forholdet mellom nøyaktighet og oppløsning)

13.7 Maskinteknisk Design av treffpunktsystem

Treffpunktsystemet skal sørge for at treffpunkter blir identifisert med en x- og y-koordinat i forhold til et referansepunkt bestemt av referansepunktsystemet. Det er konkludert gjennom undersøkelse at skuddhull ikke kan identifiseres uten at disse er markert. Skuddhullene er svært vanskelige å detektere siden både sikteblinken og skuddhullene er svart. En større kontrast er nødvendig for å detektere treffpunktene. Derfor må enhver fysisk markering ha en utforming som legger til rette deteksjonen av treffpunkter.

13.7.1 Konsepter for treffpunktmarkører

Under er det utarbeidet ulike konsepter for treffpunktmarkører. Variabler som RGB-verdier, oppløsning, klarhet, osv. bestemt av kamera og lysforhold stiller krav til hvordan markøren utformes. Dessuten må man ta i betraktning hvordan brukeren skal interagere med systemet. Derfor er det viktig å ta hensyn til ulike kaliber (størrelse på kulens diameter) for å kunne plassere markørene i skuddhullene på skyteskiven. Materialet i sikteblinken (den svarte sirkelen) er elastisk. Dermed er det gunstig at diameteren av markøren er større enn skuddhullet, slik at markøren er låst i posisjon. Ved å skape friksjon og trykk mot veggene av skuddhullet vil markøren sitte godt og presist i sikteblinken. Dette er for å slippe å ha flere typer markører til ulike kaliber.

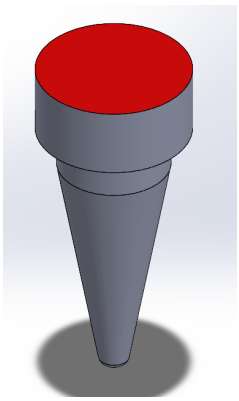
13.7.2 Farge

For at algoritmen ikke skal mistolke farger er det viktig at markørene har en sterk farge, som skiller seg fra de opprinnelige fargene på skyteskiven. Derfor er det mest hensiktsmessig å velge en av RGB-fargene, som skaper en størst mulig kontrast fra svart uten at den blir hvit. En lysere rød farge ble dermed valgt. Ut fra testing måtte man ha en stor RGB-intervall i algoritmen for å kunne detektere hele området, som er malt med rød på overflaten til

markøren. Dette gjorde at algoritmen detekterte rødfarge, som ikke var et treffpunkt. Grunnen til denne store RGB-verdi intervallet skyldes lysforhold rundt markøren og flere faktorer ved kamerainnstillingene diskutert i kapittel 13. Siden svart absorberer all lys, og er den minst reflekterende fargen. [9] Ved testing i mørkere lysforhold ble fargene vanskelige å detektere. Dersom lysforholdene ikke er gode nok, vil ikke farge være en aktuell faktor.

13.7.3 Konsept 1

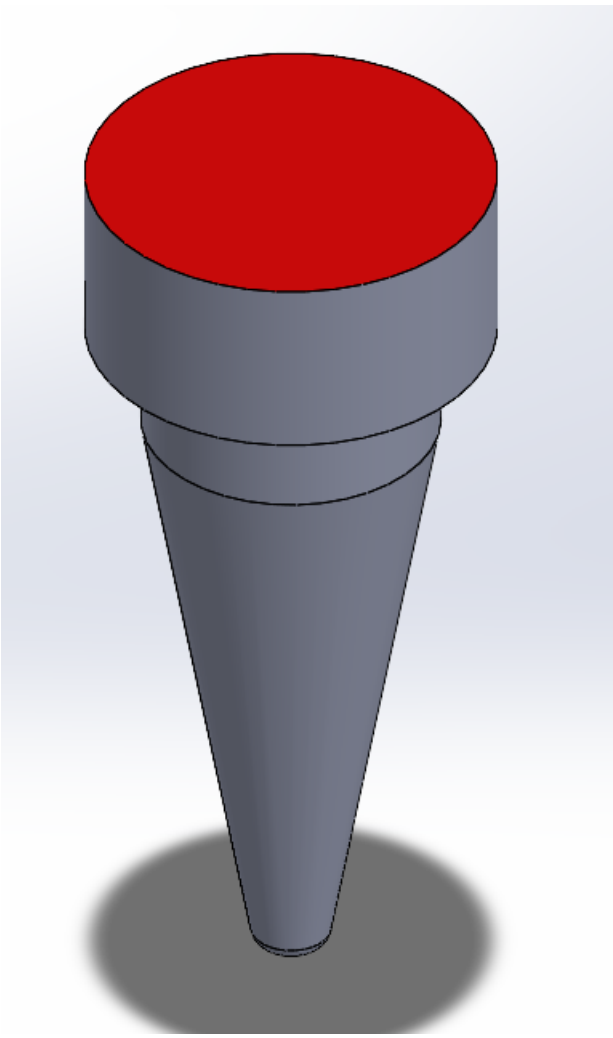
Konsept konsept 1 har en sirkulær overflate. Konseptet vil bli testet med ulike diametere. De ulike diametere er: Ø11, Ø15 og Ø19, slik at vi finner en best egnet dimensjon for markørene.



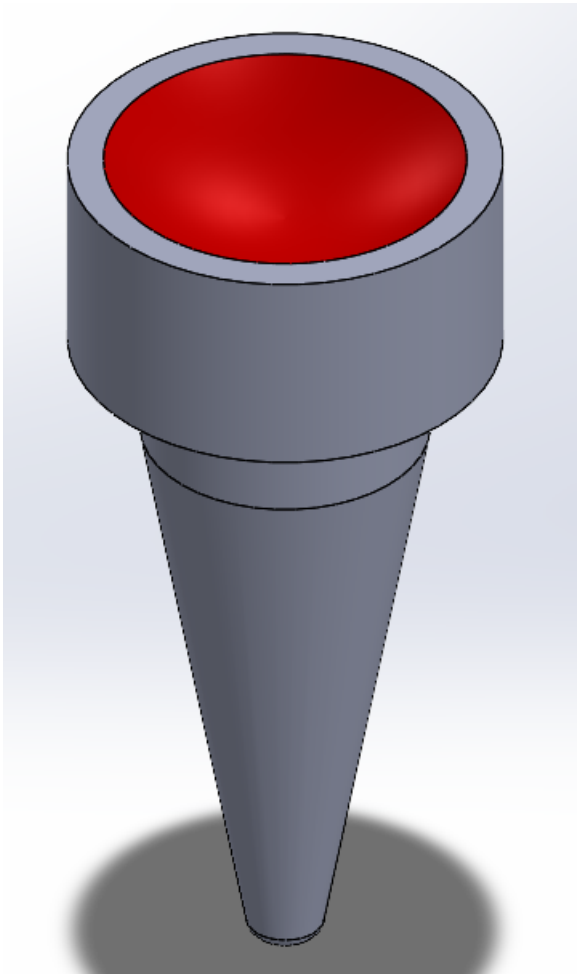
Figur 33: Konsept 1

13.7.4 konsept 2

Konsept 2 har en parabolsk/ sfærisk inn kutt på overflaten. Dette er for å sentrere lyset som kommer fra den røde overflaten mot kameralinsen, slik at fargen skal bli mer ensformig. Dette er med hensyn til algoritmen skal kunne bruke en lavere intervall av RGB farger som skal føre til bedre presisjon og mindre feilverdier ved deteksjon av treffpunktene. [10]



Figur 34: Konsept 1



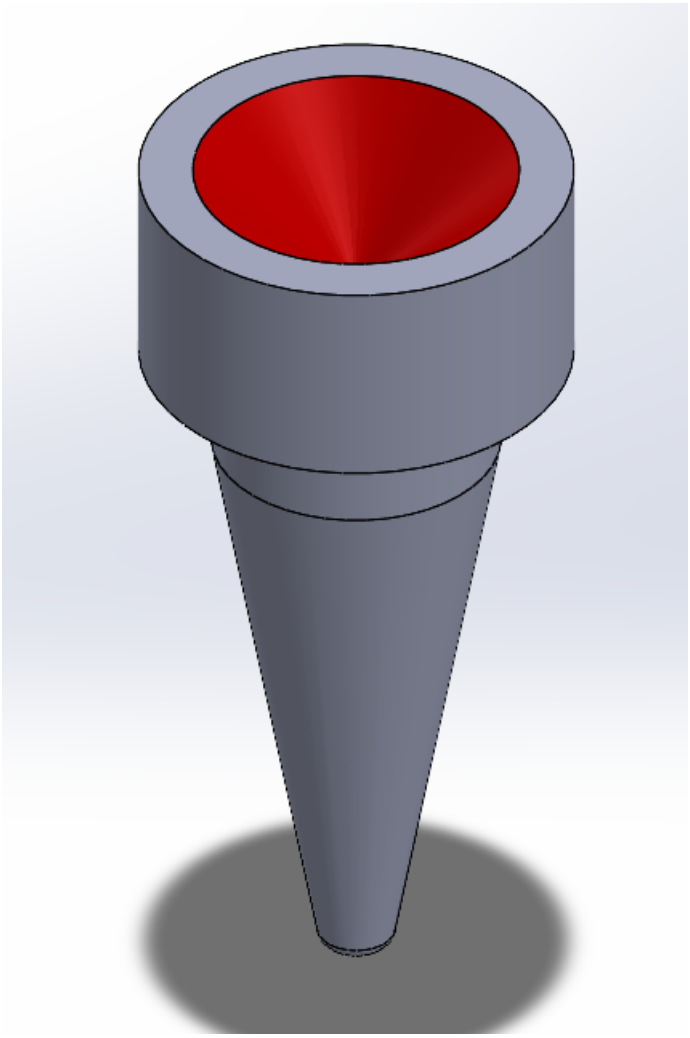
Figur 35: Konsept 2

13.7.5 Konsept 3

konsept 3 bygger på samme tankegang som konsept 2, men her er inn kuttet konisk. Den koniskformen forenkler produksjonsmetoden mot konsept 2, fordi den har en lavere vinkel fra overflaten og til senter av inn kutt. [10]

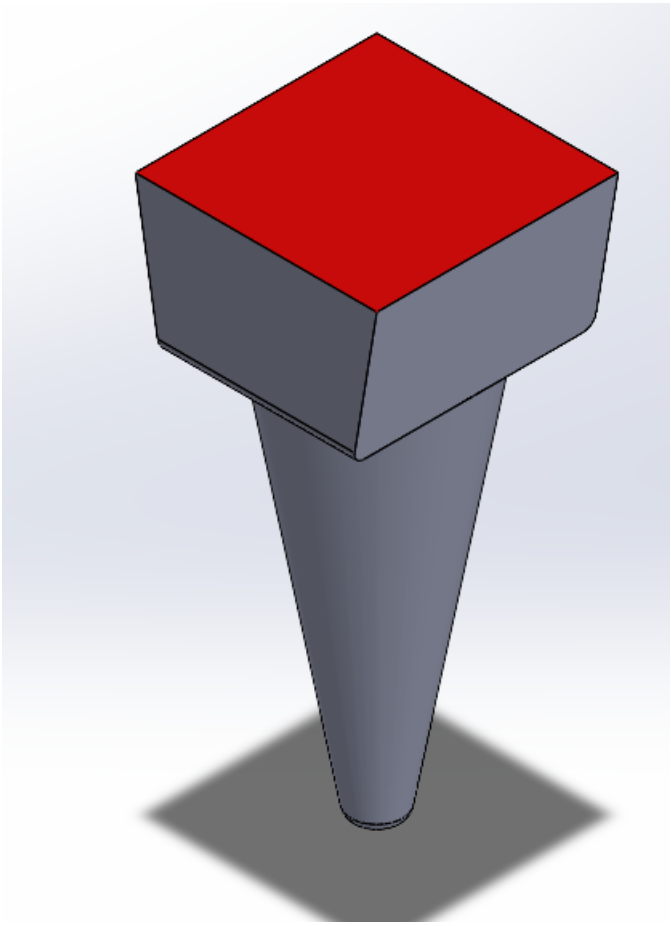
13.7.6 Konsept 4

Dette konseptet er bygget på bakgrunn av formen på piksler. Siden formen på pikslene er kvadratisk, kan det være hensiktsmessig å ha en kvadratisk form på overflate av markøren. Dette er for at pikslene skal kunne fylle overflaten



Figur 36: Konsept 3

bedre enn ved en sirkulær overflate.



Figur 37: Konsept 4

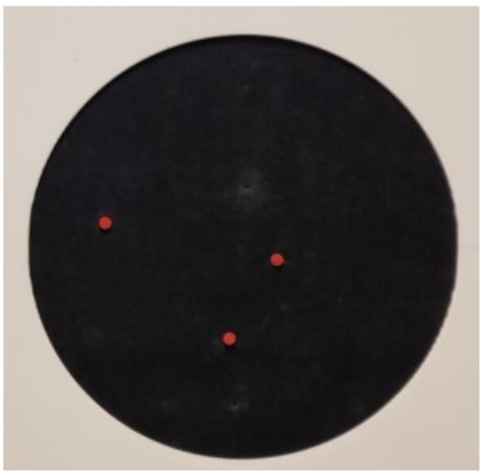
13.8 Programvare design av treffpunkt system

Formålet med oppgaven er å utføre kalibrering av treffpunkter ved hjelp av bildebehandling. Basert på våre utvalg har vi valgt å løse problemstillingen ved bruk av RGB-farger. For å kunne gjennomføre kalibreringen er det nødvendig å kunne detektere ulike markører og skille dem fra hverandre. Presisjon spiller også en viktig rolle i kalibreringen, derfor er det avgjørende å kunne finne senteret av hver markør så nøyaktig som mulig. For å utvikle en programvare av høy kvalitet er hastighet også en avgjørende faktor. Disse tre faktorene utgjør viktige områder som algoritmen skal fokusere på for å kunne skape en effektiv kalibrerings programvare.

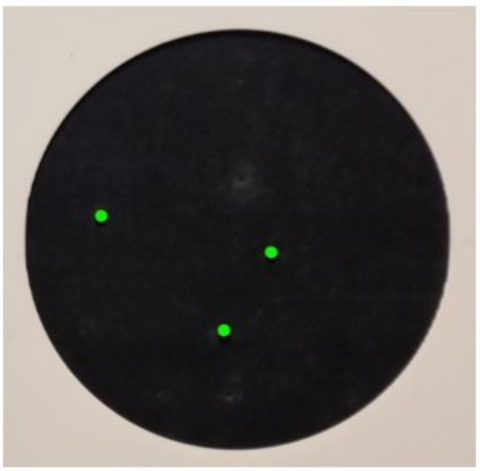
13.8.1 Deteksjon av RGB-verdier

For å detektere RGB-farger på et bilde benytter vi Matplotlib og OpenCV for å visualisere bildet og lese av RGB-fargene. Denne prosessen innebærer å iterere gjennom x (bredde) og y (høyde) verdiene i bildet. Vi søker deretter etter piksler innenfor et gitt fargeintervall som representerer markørfargene. Gjennom studier av fargestyrke har det blitt konkludert med at lyse rødfarger skiller seg mest ut fra andre farger på en skyteskive. Etter å ha identifisert et RGB-intervall for hver kolonne [R, G, B] = [kolonne, kolonne, kolonne], kan vi detektere piksler som tilhører markørene. I illustrasjonen nedenfor vises markørene før og etter deteksjon. De grønne pikslene representerer dataene som algoritmen har klart å detektere.

Etter å ha samlet inn data fra pikslene på markørene, er det nødvendig å kunne skille disse dataene fra hverandre. Dette betyr at algoritmen for øyeblikket har en betydelig mengde data bestående av x- og y-verdier fra alle markørene, uten å vite hvilken piksel som tilhører hvilken markør. Derfor vil neste trinn være å separere pikseldataene fra hverandre.

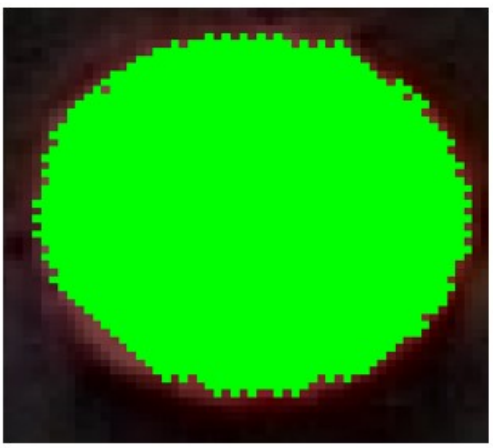


Bilde av skyteskiven før deteksjon



Bilde av skyteskiven etter deteksjon

Bilde av skyteskiven før deteksjon



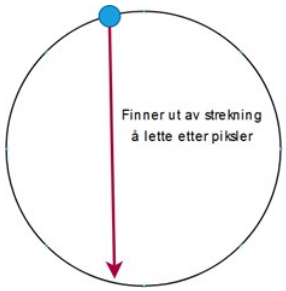
Bilde av skyteskiven etter deteksjon

13.8.2 Isolering av markørdata

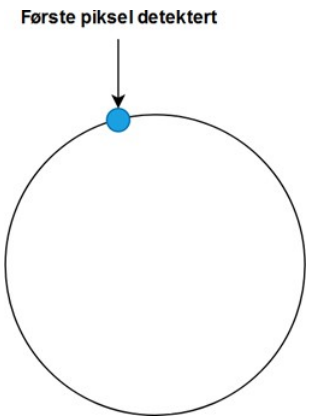
Adskillelse av markørdata utføres ved å benytte den første pikselen som blir detektert i hver markør. Ved hjelp av denne pikselen definerer vi en avstand for å søke etter nabopiksler ved å bruke den vertikale lengden fra første pikselen som er detektert til bunnen av markøren. Som illustrert i steg 2 på bildene under, observeres det at pikslene ikke treffer nøyaktig bunnen av markøren. Derfor legges det til en manuell verdi til denne avstanden for å kompensere for eventuelle avvik og sikre at hele markøren blir dekket.

Deretter utnytter vi denne avstanden til å opprette et søkeområde for å identifisere alle nabopikslene og separere dem fra andre markører. Når piksel dataene er adskilt fra hverandre, gjenstår det å utvikle en metode for å finne pikselen som ligger mest sentralt i markøren.

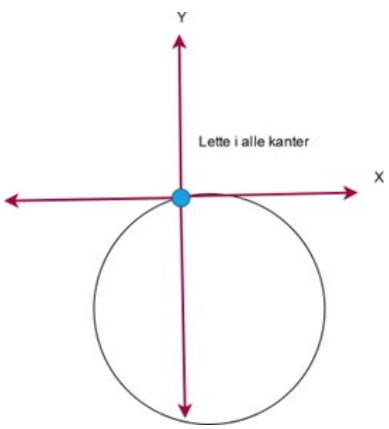
Bildene nedenfor illustrerer de fire stegene i adskillelsen av pikseldata. De viser også et bilde som illustrerer hvordan algoritmen utfører prosessen.



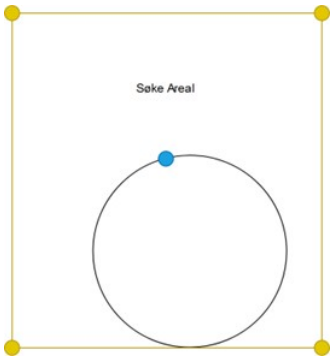
Figur 38: Steg 1: Bruke første piksel som er detektert



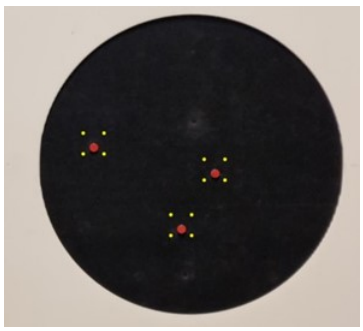
Figur 39: steg 2: Finne avstanden i å søke et nabo piksler



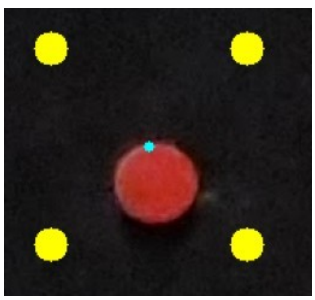
Figur 40: steg 3: lage et søke areal i alle kanter



Figur 41: steg 4: Start å søke etter nabo piksler



Figur 42: Hvordan det ser ut ved bildebehandling



Figur 43: Et nærmere bilde av markøren ved adskille piksel data fra hverandre

For å finne sentrum av en markør benytter vi en forhåndsdefinert funksjon i OpenCV kalt BoundingRect. Denne funksjonen tar inn en array av verdier og søker etter de ytterste punktene i begge aksene i 2D-planet. Den returnerer deretter x- og y-verdier, samtlig bredde og høyde for en rektangulær eller kvadratisk figur rundt objektet. Algoritmen bruker piksel dataene fra hver

markør som input til BoundingRect for å generere en rektangulær eller kvadratisk figur rundt markørene.

For å finne pikselen som ligger i sentrum bruker vi returneringsverdiene fra BoundingRect. X- og y-verdiene representerer pikselen øverst til venstre i den geometriske figuren. Bredde- og høydeverdiene angir hvor langt x- og y-verdiene skal strekke seg. Ved å betrakte bredde og høyde som et punkt (bredde, høyde), finner vi pikselen som ligger nederst til høyre i den geometriske figuren.

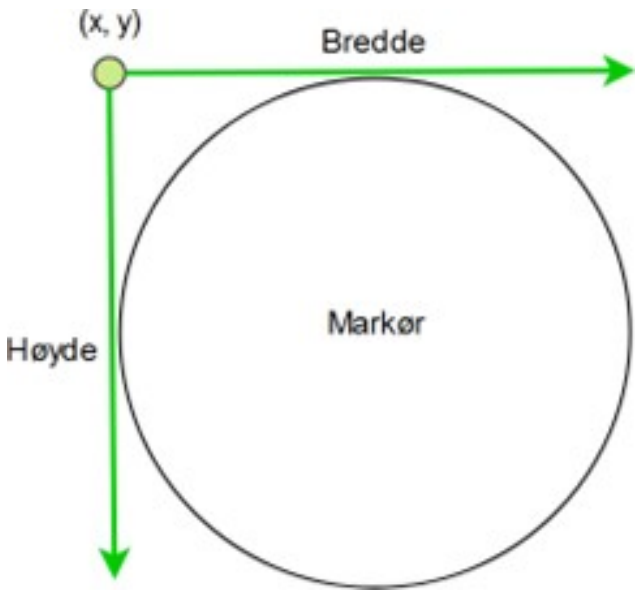
Sentrumspiksel kan beregnes ved hjelp av følgende formel:

$$\text{center\_x} = x + \frac{w}{2}$$
$$\text{center\_y} = y + \frac{h}{2}$$

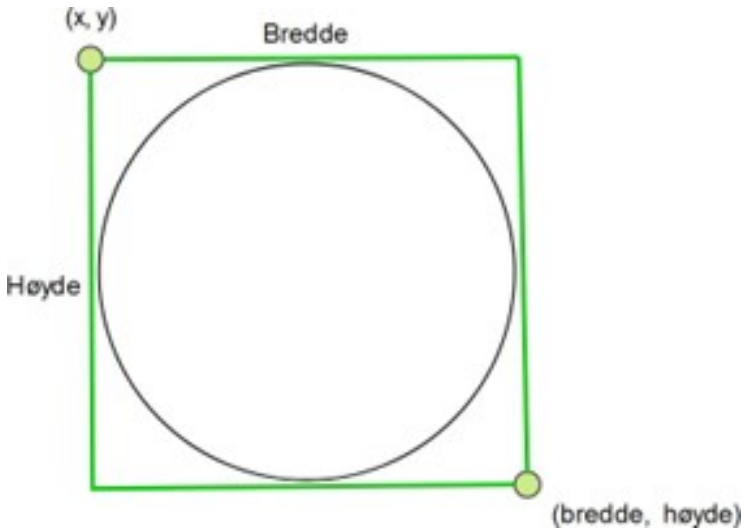
Formelen for sentrumspiksel forklarer at vi beveger oss i både x- og y-retninger basert på oppløsningen til bildet. Deretter legger vi til halvparten av verdiene for bredde og høyde for henholdsvis x- og y-koordinatene for å finne (center$_x$, $center_y$).

Bildene nedenfor illustrerer hvordan denne prosessen utføres:

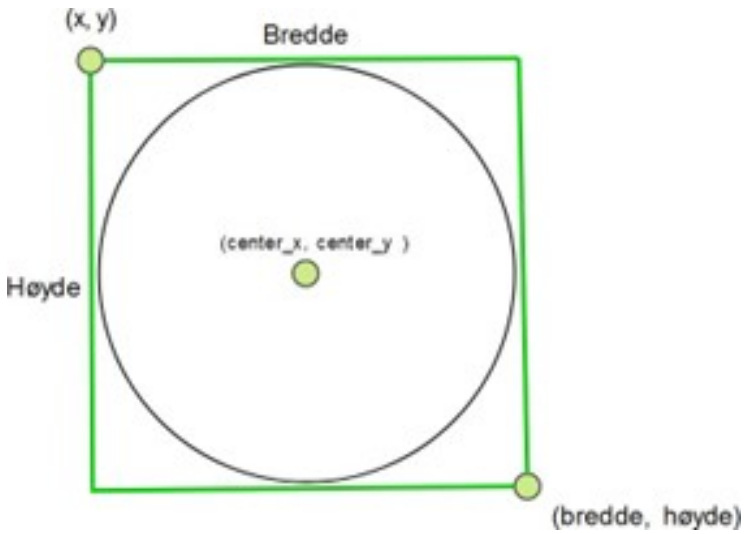
Figur 44: Pikselen som definerer hvor geometriske figuren starter



Figur 45: Bredde og høyde som beskriver arealet geoemtriske figuren dekker.



Figur 46: illustrasjon av hvis bredde og høyde er utnytte som et punkt.



Figur 47: illustasjon av hvordan formulen bruker halve verdien av bredde og høyde for å finne senter.

Ved å bruke denne metoden for å detektere senterpikselen til markørene, har vi oppnådd en presisjon på en til to piksler avstand fra sentrum. Dette bekreftes ved å observere avstanden mellom sentrum og hvitpikselens posisjon. Vi har benyttet telefon kameraene Samsung S21 Ultra og Samsung S20, og utført testing i avstander fra 1m til 2m. For å representere skuddhullene i bildene har vi brukt røde markører.

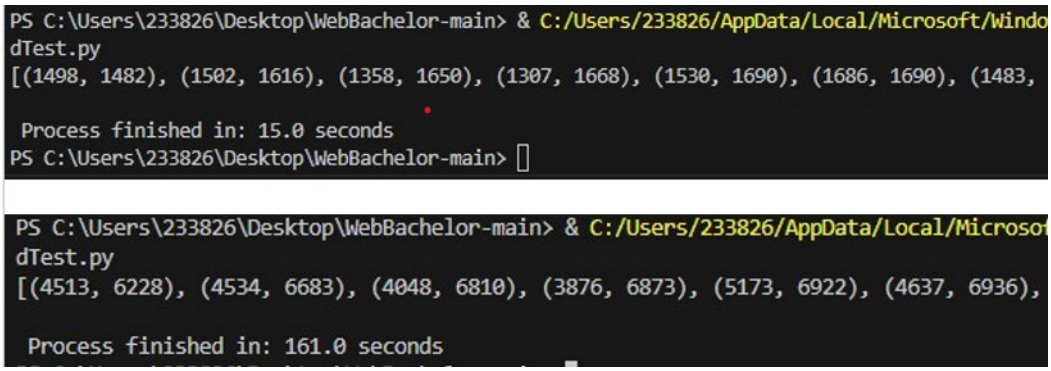
13.9 Hastighet for Treffpunkt system

Hastigheten ved å søke gjennom et bilde for å finne sentrene av markørene kan variere betydelig avhengig av to hovedfaktorer: bildeoppløsning og nøyaktighet.

Bildeoppløsning definerer størrelsen på det observerte området og mengden av verdier som må leses. Å søke gjennom et bilde med lavere oppløsning kan øke hastigheten betydelig. For eksempel utgjør forskjellen på 99 millioner piksler mellom et bilde med 9 megapiksler (9MP) og et bilde med 108 megapiksler (108MP) en betydelig forskjell i mengden data som må behandles.

Nøyaktighet oppnås ved å detektere flest mulig piksler i markørene for å bestemme markørenes sentre. Derfor er økt nøyaktighet direkte knyttet til å lese så mange piksler som mulig. For å øke hastigheten kan man utføre raskere deteksjon ved å hoppe over større horisontale og vertikale avstander i stedet for å lese hver piksel individuelt. For å utvikle en effektiv algoritme som balanserer både nøyaktighet og hastighet, kreves det en avveining mellom disse to faktorene.

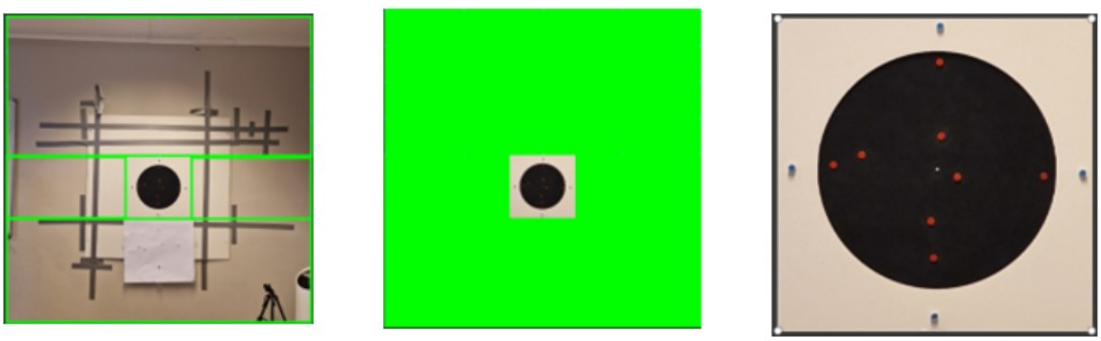
Koden og resultatene som er illustrert nedenfor, viser hastigheten ved å finne piksler på markørene ved å lese hver piksel i et bilde med 108MP sammenlignet med et bilde med 9MP.



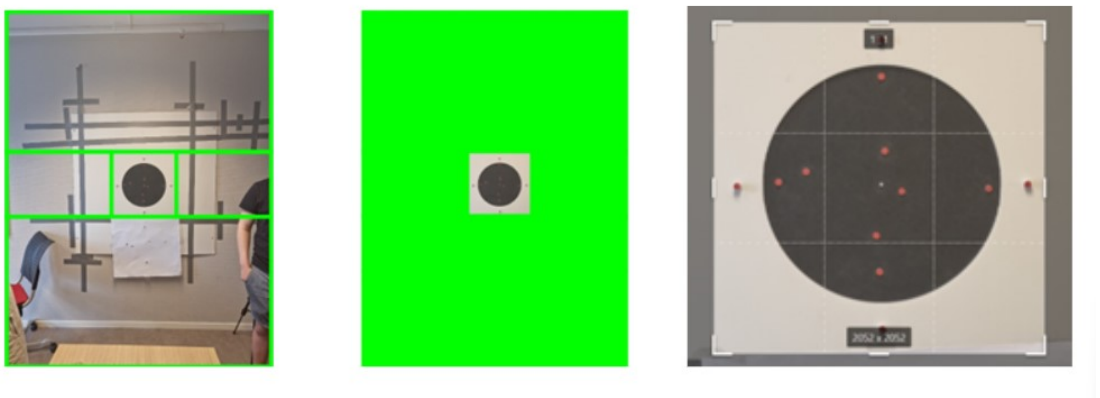
Figur 48: Hastigheten på å finne treffpunkt markører på et 9MP og 108MP piksel bilde ved full oppløsning

13.9.1 Teoretisk utregning

Siden vi ønsker å opprettholde så høy nøyaktighet som mulig, holder vi oss til å lese hver piksel. Imidlertid er løsningen for å oppnå høyere hastighet å prøve å begrense søkeområdet. Et bilde av en skyteskive kan inneholde store mengder unødvendig informasjon. Bildene nedenfor illustrerer den såkalte unødvendige og nødvendige informasjonen for å detektere markørene:



Figur 49: Unødvendig, nødvendig og hvordan nødvendig data ser ut for 9MP



Figur 50: Unødvendig, nødvendig og hvordan nødvendig data ser ut for 108MP

Basert på de tidligere nevnte bildene, er det teoretisk mulig å redusere søkeområdet på et 108MP-bilde fra 9000x12000 piksler til 2052x2052 piksler (4MP). Dette tilsvarer en reduksjon på 104 millioner piksler når bildet er tatt fra en avstand på 2,45 meter. På samme måte kan et bilde tatt fra en avstand på 2,45 meter med en oppløsning på 9MP redusere søkeområdet fra 3000x3000 piksler til 583x582 piksler, som tilsvarer 0,339MP. Dette utgjør en forskjell på 8,66 millioner piksler. Basert på disse to teoretiske beregningene kan vi forvente en betydelig økning i hastigheten ved å redusere søkeområdet.

13.9.2 effektivisering av Treffpunktsystem

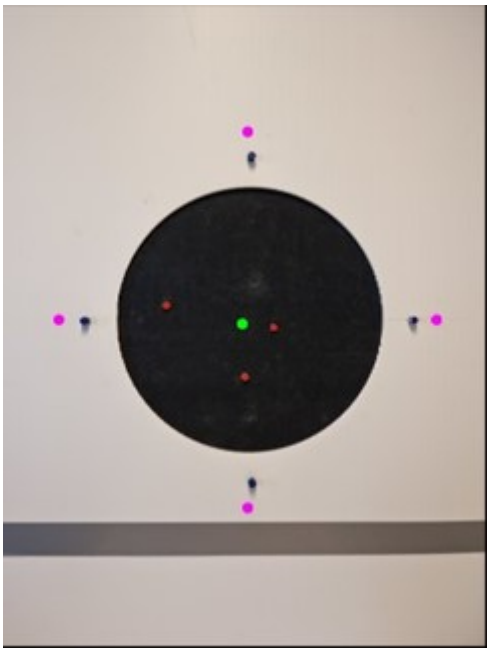
For å utvikle en algoritme som finner et søkeområde med nødvendig informasjon, er det viktig å analysere hvilken type data vi har å jobbe med. Som illustrert i de tidligere nevnte bildene, er den nødvendige informasjonen for oppgaven knyttet til et område som dekker alle markørene. Derfor er det beste valget å starte fra sentrum av den svarte skyteskiven og søke utover mot de ytre markørene. For å oppnå dette forventes brukeren å plassere den svarte skyteskiven i midten av bildet. Deretter kan vi hente oppløsningsdata og dele den på 2. Dette gir oss en sentral pikselverdi basert på bildets oppløsning. Sentrum for ulike oppløsninger er som følger:

- 108MP = (9000, 12000). Sentrum av denne oppløsningen blir (9000, 12000)/2 = (4500, 6000).
- 54MP = (6120, 8160). Sentrum av denne oppløsningen blir (6120, 8160)/2 = (3060, 4080).
- 12MP = (3000, 4000). Sentrum av denne oppløsningen blir (3000, 4000)/2 = (1500, 2000).
- 9MP = (3000 × 3000). Sentrum av denne oppløsningen blir (3000, 3000)/2 = (1500, 1500).

Siden senterpikselen er plassert i midten av bildet, og bildet er tatt med tanke på at den svarte sirkelen er i midten, øker sannsynligheten for at senterpikselen ligger innenfor den svarte sirkelen. Vi tar utgangspunkt i senterpikselen som et referansepunkt og søker i alle retninger for å finne begrensningene for søkeområdet.

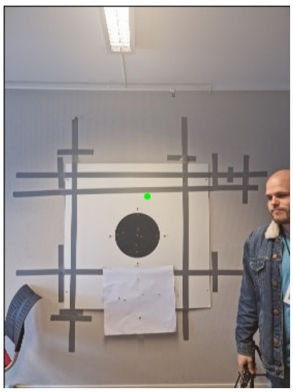
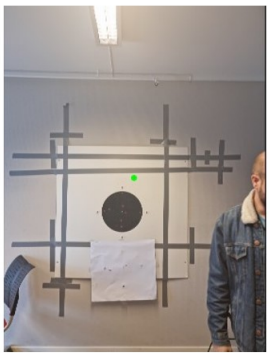
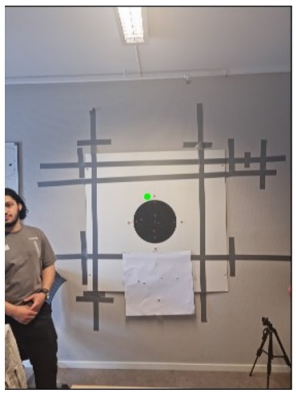
For å finne disse begrensningene utnytter vi RGB-farger og ser etter data som kan bekrefte at alle markørene er dekket. Ved å se på bildet, kan vi observere at avstanden mellom akseskorsmarkørene og den svarte sirkelen er relativt liten. Derfor kan vi konkludere med at vi kan løse dette problemet ved å undersøke om pikselen ved siden av senterpikselen og i ulike retninger er hvit. Hvis pikselen viser seg å være hvit, betyr det at vi har nådd et punkt utenfor periferien til den svarte sirkelen. Deretter er det sannsynlig at den hvite pikselen befinner seg under akseskorsmarkørene, så vi må ta et ekstra hopp for å sikre at akseskorsmarkørene er inkludert i søkeområdet.

Bildet nedenfor illustrerer hvordan denne prosessen utføres. Den grønne sirkelen representerer senterpikselen, og de rosa sirklene representerer begrensningene for søkeområdet.



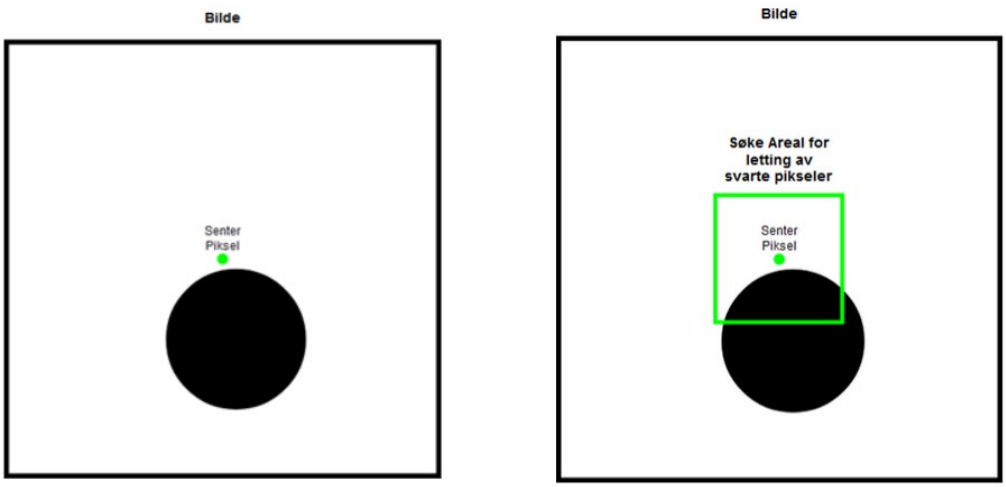
Hvis brukeren får beskjed om å plassere den svarte sirkelen i midten av bildet, øker sannsynligheten for at senterpikselen ligger innenfor den svarte sirkelen. Selv om sannsynligheten øker, er det ikke nødvendigvis en garanti. Ved større avstander blir den svarte sirkelen forminsket, noe som kan gjøre det vanskeligere for brukeren å sentrere den nøyaktig. Vi kom frem til denne observasjonen ved å la tre forskjellige og tilfeldige individer ta bilder av skyteskiven.

Instruksjonene de fikk var å prøve å holde telefonen i en 90 graders vinkel og plassere den svarte sirkelen så nøyaktig som mulig i midten når de tok bilder fra en avstand på 2.45 meter. Bildene nedenfor viser hvor senterpikselen befinner seg basert på hvordan bildene ble tatt. Den grønne pikselen markerer punktet (4500, 6000), siden alle bildene ble tatt med en oppløsning på 108MP.

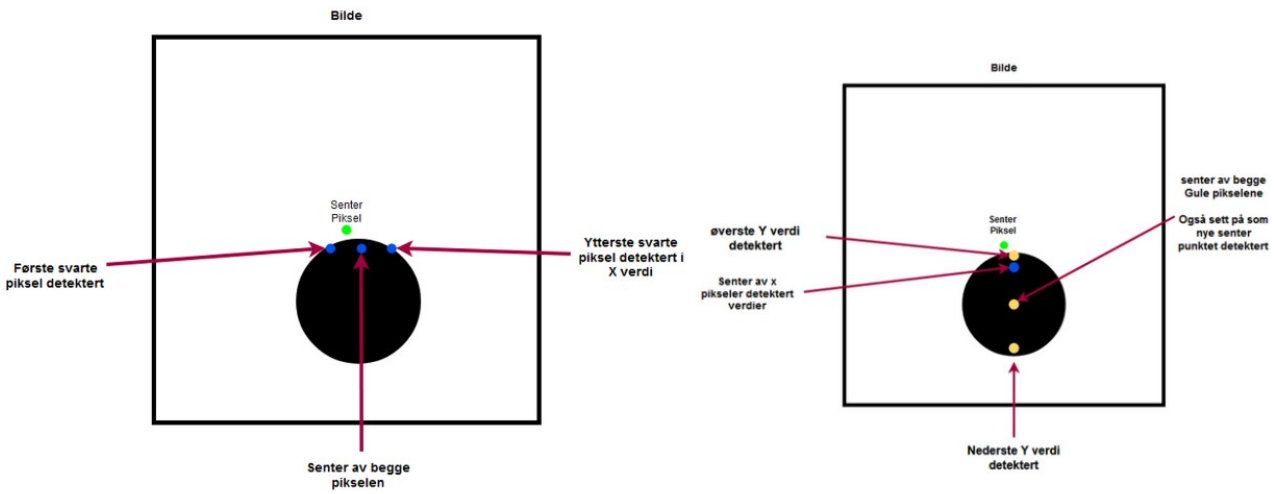




Hvis senterpikselen ikke ligger på den svarte flaten, vil algoritmen umiddelbart anta at den har overskredet periferien til den svarte sirkelen. Dette fører til at algoritmen oppretter en begrensning i form av et kvadratisk område på bildet som ikke dekker alle markørene. For å unngå dette må vi sikre at selv om brukeren ikke sentrerer den svarte sirkelen nøyaktig i midten av bildet, kan algoritmen ta hensyn til avviket i posisjoneringen.

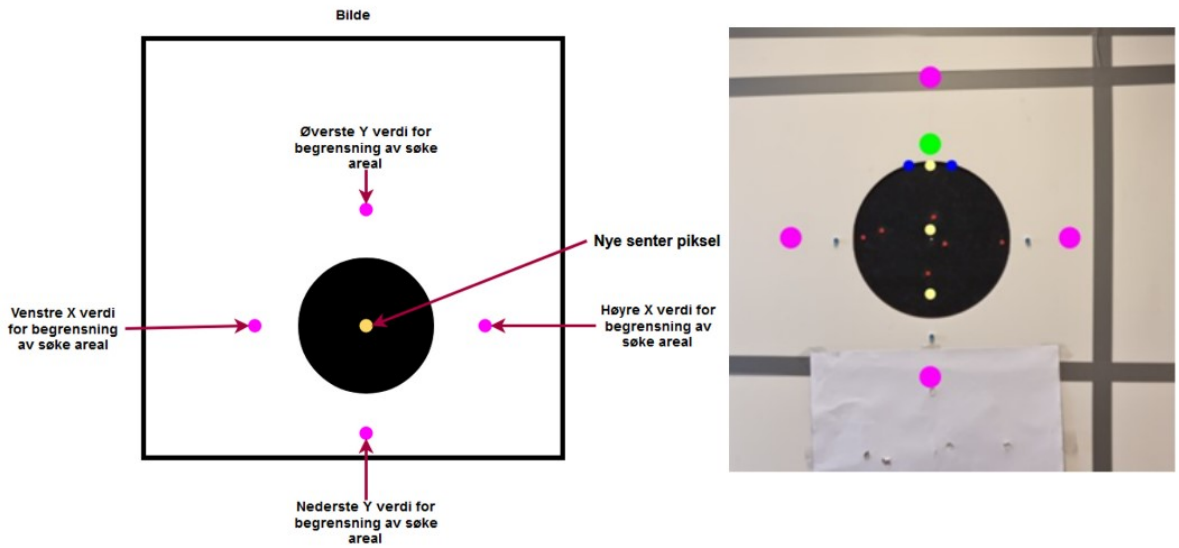
Dette oppnår vi ved å utnytte posisjonen til senterpikselen og opprette et kvadratisk søkeområde for å lete etter svarte piksler. Når en svart piksel blir funnet, blir verdien lagret. Deretter leter vi etter de ytterste svarte pikslene i x-retningen og finner sentrum av disse. Samme prosess gjentas for y-aksen ved å bruke dataene fra pikselen som ble funnet mellom de to ytterste x-aksene. Til slutt får vi en verdi som er sentrert i midten av den svarte sirkelen. Bildene nedenfor illustrerer denne prosessen.



Figur 51: steg 1 - steg 2

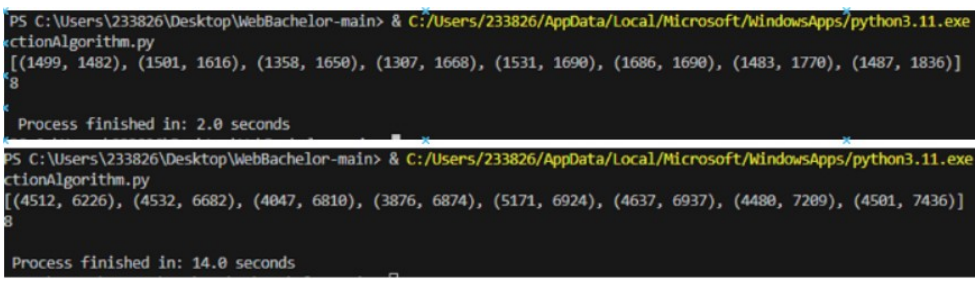


Figur 52: steg 3 - steg 4



Figur 53: steg 5 og hvordan siste

Ved å benytte denne nye metoden for å definere et søkeområde, kan vi ta hensyn til all unødvendig bakgrunnsdata og fokusere på den relevante informasjonen. Etter å ha testet metoden, har vi observert en betydelig endring i hastigheten. Bildene og tabellen nedenfor viser endringen i hastighet.



Figur 54: Ny hastighet ved bruk av søke areal. 9MP og 108MP

Oppløsning	**Hastighet uten søke areal**	**Hastighet med søke areal**
108MP	161s	14s
9MP	15s	2s

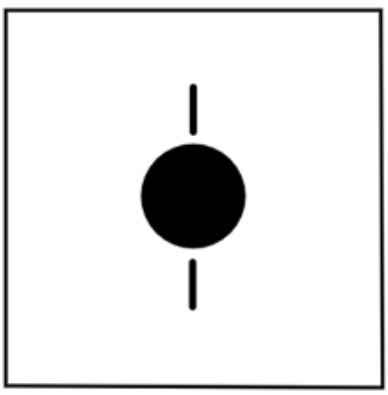
14 Test og utvikling av treffmarkører

Ut fra tester gjort var det konsept 1. En sirkulær overflate som var det mest hensiktsmessig valget. Sammenlignet med 4 andre konsepter. Konsept 2 og 3 er basert på samme prinsipp. Problemet ved disse konseptene var at de fikk en ujevn belysning. Dette førte til at en halvdel av overflaten til markøren ble skyggelagt og den andre ble belyst. Hadde belysningen vært kontrollert og tilpasset formen, vil disse konseptene fungert bedre. Konsept 4 med en kvadratisk overfalte baserer seg på formen til pikslene, men denne formen vil føre til at plasseringen av markørene ikke blir uniforme. Dette vil føre til problemer ved deteksjon av midtpunktet til markøren. Ut fra dette ble konsept 1 valgt.

Konsept 1 har hatt kun 1 iterasjon. Dette handler om at konseptet har en veldig enkel utforming. Tanken bak konseptet er at den skal være spiss slik at den enklere sklir inn i skuddhullet som har en mindre diameter enn markøren. Dermed blei et prosjektillignende form valgt.

14.1 Maskinteknisk design av treffpunktsystem

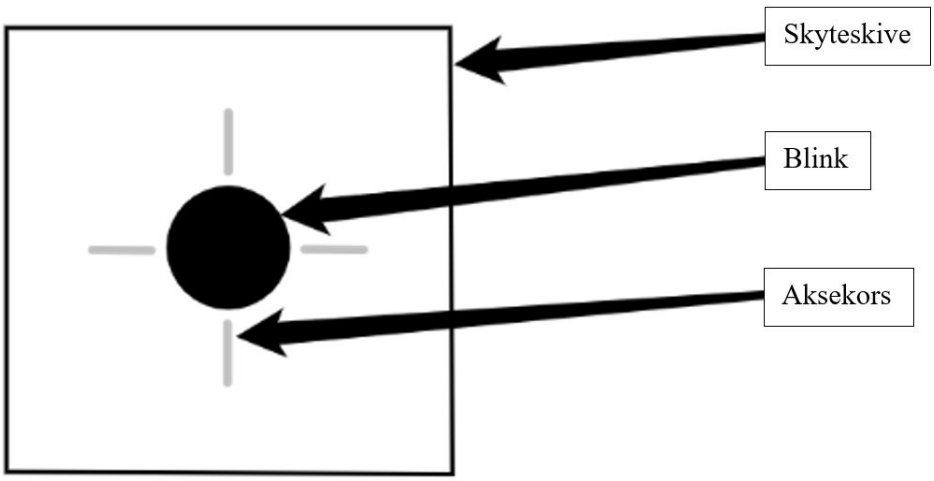
Produksjonsmetoden av treffmarkører skjer ved 3D-printing. Markørene blir bygget opp fra den røde overflaten til spissen av markøren. Ved 3D-printing rett på glassoverflaten ble diameteren til markøren litt større enn på tegninger. Det var derfor hensiktsmessig å printe først et tyntlag, og dermed printe markørene opp fra det tynne plast laget. Dette skaper en god nøyaktighet på printen i henhold til tegningene. Senere produksjonsmetode vil det mest hensiktsmessige være å støpe markørene. Dette gjør produksjonen raskere, og enda mer nøyaktig.



Figur 55: Konsept 1: sirkulær overfalte

15 Design av referansepunktsystem

Med i utgangspunkt i funksjonene systemet må ha i systemdesign vil det i ethvert konsept være avgjørende å identifisere et referansepunkt tilsvarende origo i et 2D koordinatsystem. Med referansepunktet kan man målsette treffpunkter i form av x- og y-koordinater i 2D-planet.



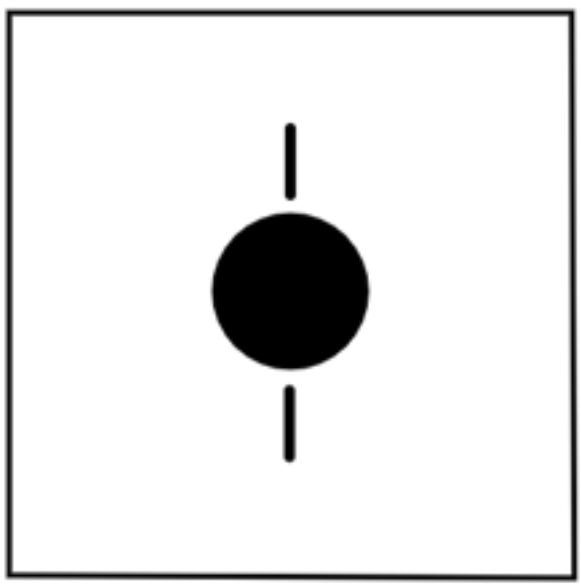
Figur 56: Figuren over beskriver hvilke elementer som kan brukes i bildebehandling til å detektere referansepunktet.

15.1 Konsepter for referansepunktsystemet

Metoden for å bestemme referansepunktet skal være ved hjelp av en bildebehandlingsprogramvare, og fysiske markører kan også bidra til å hjelpe programvaren. Vi er begrenset til å jobbe med den eksisterende målskiven til KTS. Med flere ulike konsepter kan man kartlegge hvilke markører og metoder som kan være mer hensiktsmessige enn andre å bruke. Konseptene skrevet under sammenlignes i en Pugh-matrise. Se figur 62.

15.1.1 Konsept 1

Dette konseptet går ut på å manuelt sette på fysiske markører som plasseres i de vertikale aksene i aksekorset for å bestemme referansepunktet i senter av skyteskiva. Figuren under illustrerer dette. To fysiske markører kan settes inn ved hvert sitt vertikale aksekors. Programvare kan med disse to punktene konstruere en vertikal linje som går igjennom midtpunktet på skyteskiven. Deretter kan avstanden mellom de to vertikale linjene brukes til å finne et midtpunkt. Med den vertikale linjen som f.eks. y-aksen, og midtpunktet, kan man konstruere en x-akse som er ortogonal på y-aksen og går gjennom senterpunktet. Dermed er det dannet et 2D koordinatsystem med ett referansepunkt.



Figur 57: Konsept 1

Fordeler og ulemper

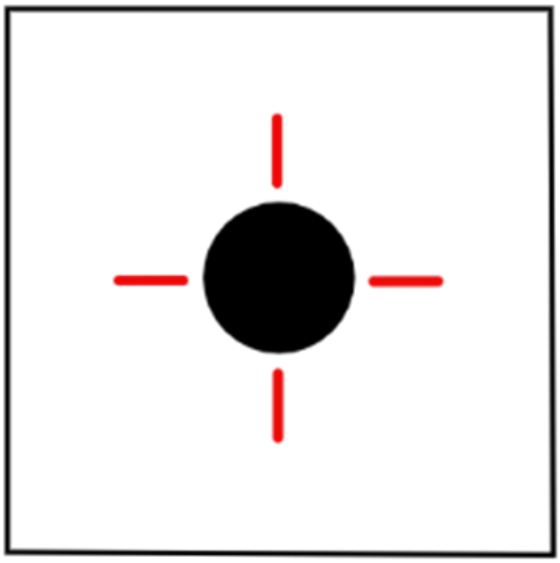
De midlertidige fysiske markørene er enkle å manuelt ta av og på, slik at det ikke blir forstyrrelser for skytteren. Likevel må markørene settes på manuelt, og det vil være noe rom for unøyaktighet dersom man ikke får en god metode og veiledning. Noe tid vil også gå til dette. Det er også noe økt kompleksitet ved å konstruere linjer og bruke disse i et koordinatsystem. Denne løsningen vil også kunne fungere over tid og varierende værforhold.

15.1.2 Konsept 2

Referansepunktet kan bestemmes ved å tegne på en kontrastskapende (f.eks rød) markering på aksekorset dannes en vertikal og horisontal linje. Programvare vil da kunne ekstrapolere de vertikale og horisontale linjene slik at disse danner et senterpunkt. Midtpunktet brukes som referansepunktet i kalibreringsprosessen. For å unngå at skytteren blir forstyrret, må aksekorsene dekkes over med teip e.l. etter kalibreringen.

Fordeler og ulemper

Å tegne på linjer på selve aksekorset er en manuell prosess, og kan føre til unøyaktighet, selv med eventuelle verktøy som kan tilrettelegge for dette. Krav må stilles til brukeren og dermed brukerinstruksen for å ivareta presisjon. Prosessen vil også være tidkrevende. Maskeringsteip e.l. må brukes i etterkant av kalibreringen til å dekke over aksekorsene. Ved slitasje pga. bruk og varierende værforhold vil det kunne være utfordrende å sikre at materialet dekker over aksekorsene og forblir uforstyrrende for skytteren. Fordelene med denne ideen er at markeringen blir svært tydelig, og kompleksitetsgrad mtp. programvaren blir lavere. []



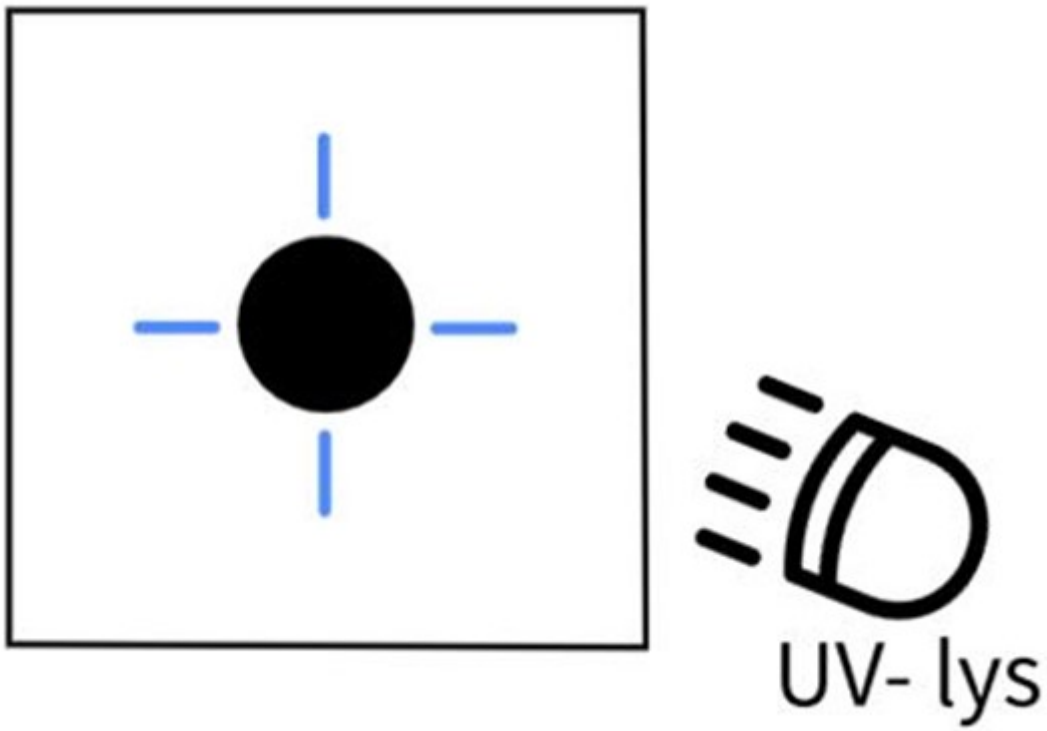
Figur 58: Konsept 2

15.1.3 Konsept 3

Konsept 3 er basert på den samme metoden som konsept 2 med noen endringer. Fargen i konsept 2 er synlig for kameraet og det blotte øyet ved det synlige fargespekteret. Ved å markere aksekorset med fluoriserende middel vil denne være synlig dersom denne er utsatt for ultrafiolett (UV) stråling. Samtidig vil den ikke være synlig for det blotte øyet, og dermed ikke være til hinder for skytteren.

Fordeler og ulemper

Fordelen med konsept 3 er at man slipper å skjule markeringen av aksekorsene med maskeringsteip e.l. fordi markeringen ikke er synlig for skytteren. Det er noe økt kompleksitet ifm. bruk ultrafiolett lys. Brukervennligheten man oppnår ved å slippe å se aksekorsene kan utlignes av lavere effektivitet eller automasjon, siden man må i større grad forberede målskiven.

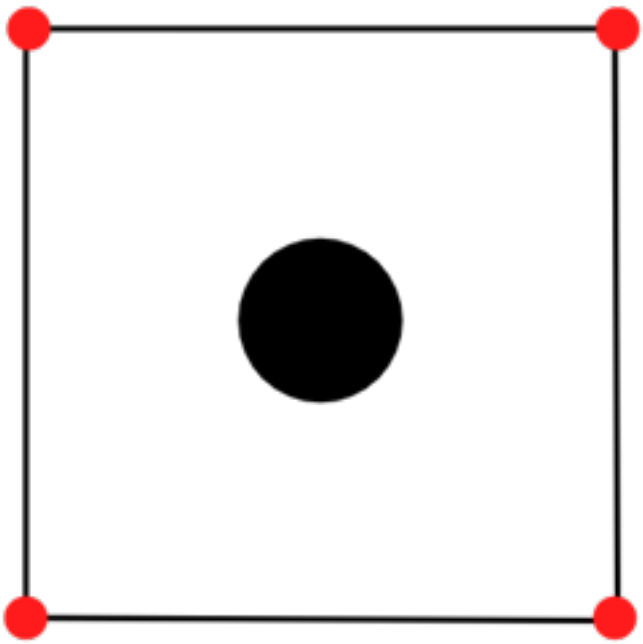


Figur 59: Konsept 3

15.1.4 Konsept 4

Dette konseptet går utpå å markere hver av hjørnene til skyteskiven slik at man kan bruke hjørnene til å definere et referansepunkt. Et slik type konsept vil føre til at en ikke trenger å dekke over disse markeringene i hjørnene, fordi de er såpass langt unna blinken på skyteskiven, vil markeringene sannsynligvis ikke være forstyrrende for skytter.

Fordeler og ulemper Med dette konseptet vil man unngå forstyrrelse for skytteren, og siden en markør må plasseres på hvert hjørne er den manuelle påvirkningen moderat. En ulempe kan være at presisjonen vil være avhengig av kanten til selve målskiven, og den manuelle plasseringen av markører. Man vet ikke om hjørnene har en like pålitelige nøyaktighet ved produksjon som aksekorsene.



Figur 60: Konsept 4

15.2 Konsept 5

Programvare kan brukes til å finne referansepunktet i et bilde, for eksempel ved hjelp av en bildebehandlingsalgoritme. Ved å mate inn et bilde av skyteskiven i algoritmen, kan algoritmen å markere en figur, og bruke denne til å finne referansepunktet på skytefiguren. Algoritmen er programmert i Python ved hjelp av bildebehandlingsbiblioteket OpenCV.

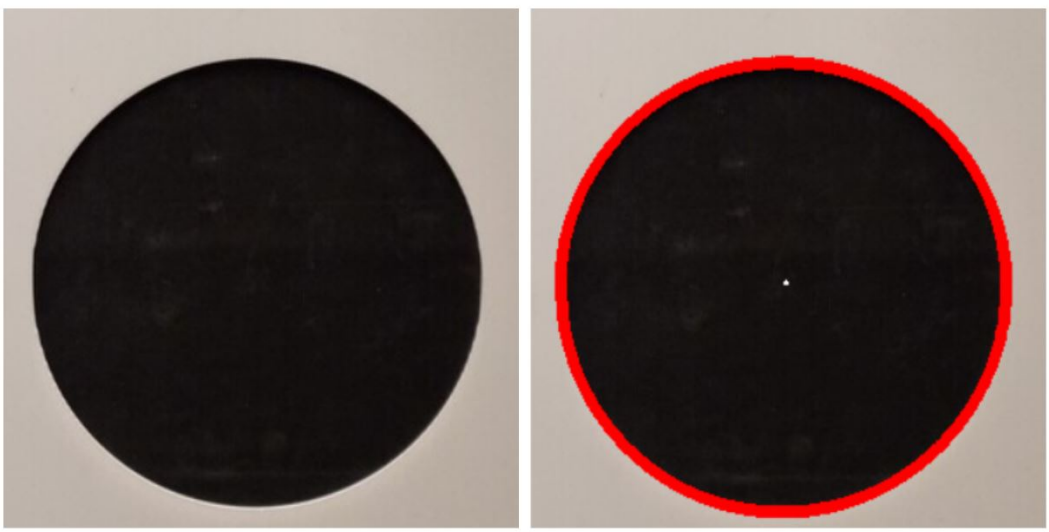
OpenCV

OpenCV (Open Source Computer Vision Library) er en av de mest populære bibliotekene for bildebehandling som er tilgjengelig i dag. Biblioteket er åpen kildekode og brukes til datamaskinsyn, maskinlæring og bildebehandling. Det inneholder funksjoner som gjør det mulig å behandle bilder for å identifisere

ulike objekter. OpenCV har støtte for flere programmeringsspråk, inkludert C++, Python, Java og MATLAB.

Algoritme

Algoritmen starter med å lese inn et originalbilde av skyteskiven. Deretter konverteres bildet til gråskala, ettersom det er nødvendig for at algoritmen skal fungere. Videre anvendes terskler på bildet, og konturene hentes ut fra terskelbildet. Segmentering av bildet skjer når tersklene er satt, og dette er viktig for å kunne skille mellom pikslene i bildet. Deretter finner algoritmen fasongen på skyteskiven og tegner konturen rundt den. Algoritmen finner til slutt senterpunkt av skyteskiven ved hjelp av konturen og markerer dette punktet. Se Figur 61



Figur 61: Konsept 5, Bilde av skyteblinken før og etter bildebehandling. Programvaren markerer omrisset av figuren med en rød linje. Deretter bestemmer den midtpunktet til figuren.

Fordeler og ulemper Den største fordelen med bruk av algoritmer til å finne senter av figurer er at man slipper manuelle påvirkninger ved fysiske markører. Prosessen rundt kalibreringen blir dermed enklere sammenlignet med andre konsepter. Ulempen er at omrisset algoritmen lager av figuren ikke er så presis, dvs. at det er 1-2 mm avvik. Dette er for upresist når man skal ta

andre unøyaktigheter med i den totale beregningen for presisjon.

15.3 Vurdering av konsepter til referansesystem

I vurderingen av de ulike konseptene for referansepunkt kan vi se hvilke kriterier som bør prioriteres fremfor andre. Pugh-matrisen (Figur 62) nedenfor gir en indikasjon og veiledning om hvilke elementer som kan være mer gunstige å ha fremfor andre.

Konsepter/ kriterier	Presisjon	Lav Kostnad	Effektivitet	Automasjon	Brukervennlighet	Gjennomførbarhet	Totalsum
Konsept 1	3	4	4	4	4	3	22
Konsept 2	4	5	3	3	4	5	24
Konsept 3	4	3	2	2	3	3	17
Konsept 4	3	4	3	3	3	5	21
Konsept 5	2	5	5	5	5	5	27

Figur 62: Pugh-matrise av konsepter for referansepunkt. Pugh-matrisen kartlegger og tydeliggjør hvilke kriterier som bør prioriteres foran andre

I Pugh-matrisen ovenfor har konsept 5 høyest poengsum, 27 av 30. Ved å ha en fullstendig programvarebasert løsning vil man kunne unngå komplikasjoner som finnes ved mange av kriteriene. Et koordinatsystem må ha både et midtpunkt, og 2 akser. Konseptet kan brukes for å finne midtpunktet presist, men samtidig kan den ikke oppfatte hvordan et koordinatsystem skal være orientert i henhold til en x- og y-akse. Uten en fast orientering vil man ikke kunne bestemme et koordinatsystem. Dersom presisjonskravet ikke er mulig å oppfylle, kan ikke konseptet brukes videre. Samtidig ser man mange fordeler ved å bruke programvare.

Konsept 2 har en poengsum på 24 av 30. Den har en middels effektivitet og automatikk, men en høy grunnleggende presisjon. Derfor vil man spare arbeid på å få presise resultater, men mer arbeid må legges inn i utføre kalibreringen effektivt og automatisk.

Konsept 1 har en vurdering på 22 av 30 poeng. Den skårer middels på presisjon, men bruken av fysiske markører sparer tid sammenlignet med konsept 2, 3 og 4.

Oppsummering

Ved å stille konseptene opp mot hverandre ser man at fordeler ved noen konsepter kan integreres inn i andre konsepter for å forbedre disse. Dersom man kan ha fysiske markører til å markere begge aksene i aksekorset, vil man få god presisjon, som er det høyest verdsatte kriteriet. Samtidig vil graden av automasjon gå ned. Ved å bruke programvare til de gjenstående funksjonene som må utføres, sørger man for god effektivitet. Kostnad vil være noe høyere på grunn av de fysiske markørene. Dersom man ønsker å få god nok presisjon må man øke kostnaden noe. Med bakgrunn i testing og diskusjon av konseptene har vi kommet frem til en løsning for deteksjon og bestemmelse av referansepunkt.

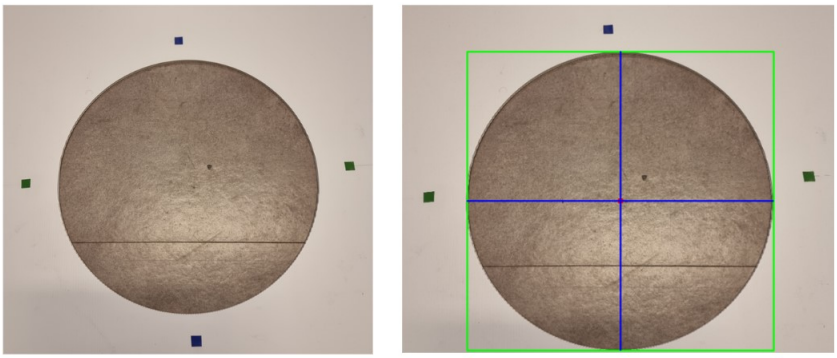
15.4 Programvareutvikling av referansepunktsystem

15.4.1 Referansepunkt ved kontur deteksjon

For å finne referansepunktet så vi på forskjellige måter og løse problemene på. Et av løsningene kom fra å spørre Chatgpt. Spørsmålet som ble spurt var:

"Can you create a code which is commented, where i can detect and mark the central point of a shooting target and get its coordination's by using python. Count also the amount of pixel it is from the center to the Top, Bottom, left and right most pixel. " Det som koden gjør, er hovedsakelig å benytte to biblioteker, OpenCV og NumPy. Disse bibliotekene brukes til å laste inn bilder av en skyteskive og finne sentrumspunktet av skyteblinken.

Først lastes et bilde av skyteskiven inn, og størrelsen på bildet endres for å sikre at resultatene kan vises tydelig på skjermen. Ettersom bildet ofte har en høy oppløsning, blir det skalert ned for å gjøre det mer oversiktlig. Deretter gjennomgår bildet en modifikasjon for å fjerne unødvendige detaljer, slik at thresholdfunksjonaliteten kan brukes til å skille skyteblinken fra bakgrunnen. Deretter identifiseres de fargene som skiller seg fra de hvite pikslene på skyteblinken. Når disse fargene er detektert, tegnes det opp en rektangulær ramme rundt skyteblinken på bildet. Når rektangelet er tegnet opp, kan vi beregne sentrumspunktet samt avstanden fra sentrum til pikslene som befinner seg øverst, nederst, til venstre og til høyre.



```
Picture size:  (4000, 3000, 3)
Picture resized:  (1000, 750, 3)


Center of target: (378, 529)
Distance from center to left pixel: 264 pixels
Distance from center to right pixel: 263 pixels
Distance from center to top pixel: 257 pixels
Distance from center to bottom pixel: 257 pixels
```

Algoritmen har høy presisjon basert på en pikselavvik-verdi som vises i outputen på terminalen. Horisontale linjer gir oss verdier fra venstre til midten på 264 piksler og fra midten til høyre på 263 piksler. Selv om presisjonen er god, er ikke algoritmen nødvendigvis tilpasset alle løsninger. Hvis bildet ikke er tatt i en 90 graders vinkel fra skyteskiven, kan den ene siden av skyteblinken ha flere piksler enn den andre, og dermed vil ikke referansepunktet være midten av skyteblinken. I tilfeller hvor skyteblinken ikke er en sirkel, men for eksempel en elg, kan dette skape problemer ved at referansepunktet ikke er midten av elgen, men heller plassert på bakdelen av elgen. Dermed vil ikke referansepunktet bli plassert riktig basert på denne algoritmen.

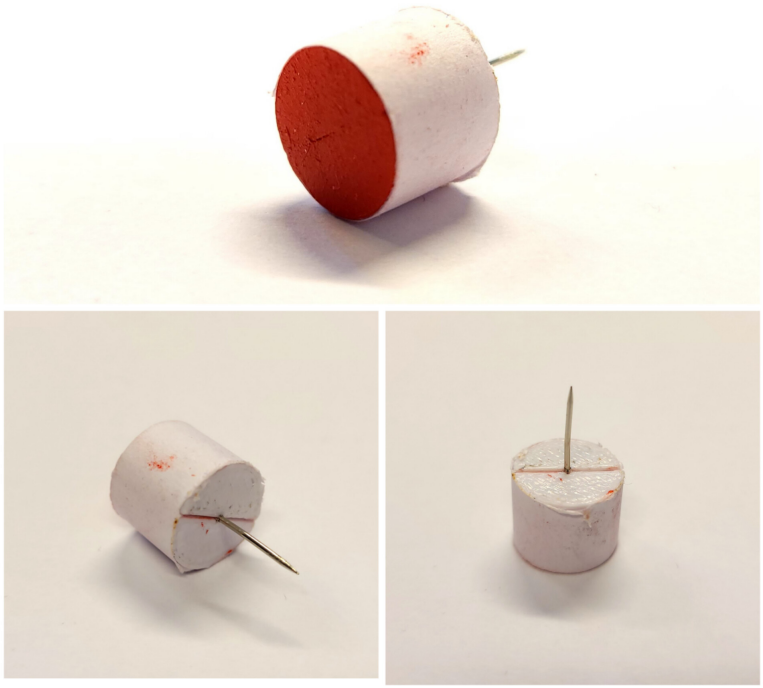
15.5 Maskinteknisk design av referansepunktmarkør

I den konseptuelle designfasen ble det vurdert at fysiske avtakbare markeringer i kombinasjon med programvare er mest forenlig med de kriteriene som er hensiktsmessig å prioritere. Derfor vil flere prototyper med disse konseptuelle elementene utarbeides. Designet av referansemarkøren har foregått over flere iterasjoner. Farge og utforming av referansemarkørene er bestemt av programvaren gjennom testing av flere prototyper. Disse prototypene er beskrevet i avsnittet under. Utviklingen av programvaren og de fysiske markørene har foregått samtidig. Brukervennlighet og HMS er også faktorer som har blitt vurdert under utviklingen av Referansemarkøren.

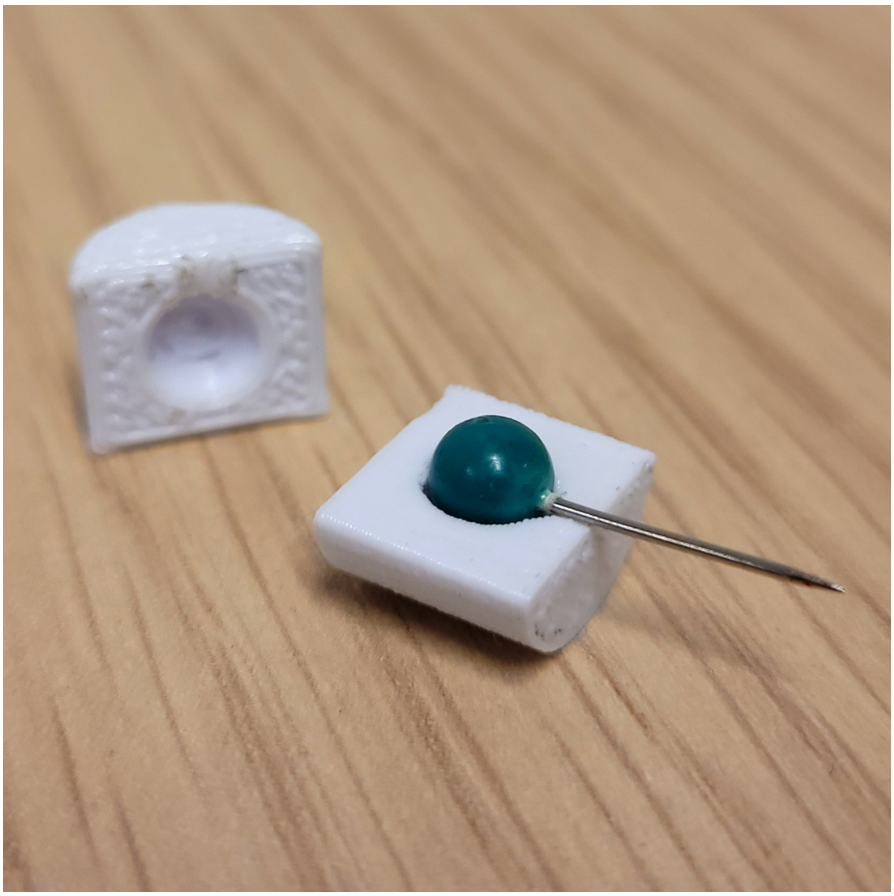
15.5.1 Prototype 1

I begynnelsen ble det forsøkt å bruke knappenåler som referansemarkører siden disse er billige og tilgjengelige. Samtidig var utformingen til knappenålhodet upålitelig, siden denne i stor grad varierte mellom 4,7 og 5,2 mm i diameter. Dessuten var knappenålhodet ikke sentrert i forhold til nålen. Selve knappenålhodet er sfærisk i sin utforming, og reflekerer lys utover slik at denne blir uklar. Dermed hadde programvaren problemer med å finne midtpunktet av knappenålhodet.

På bakgrunn av ulempene med en ordinær knappenål ble prototype 1 utformet. Prototypen er beskrevet i figur 63 og figur 64. To halvsylindriske elementer holder en knappenåle på plass. De 2 halvsirklene skaper sammen en hel flat sirkeloverflate som kan detekteres.



Figur 63: Prototype 1 av referanse markør.



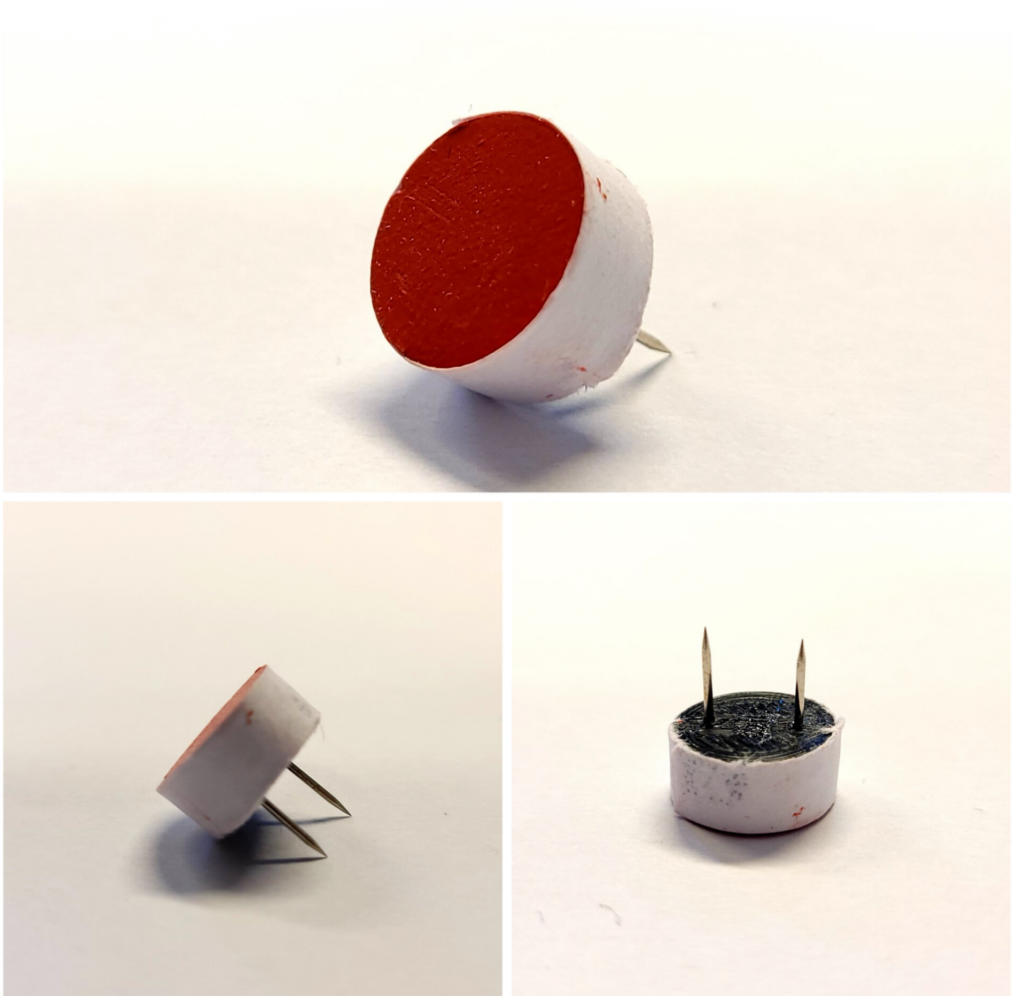
Figur 64: Prototype 1 i åpnet visning.

Ved prototype 1 var det særlig én ulempe som ble oppdaget gjennom testing av programvaren. Tykkelsen på prototype 1 utgjorde en overflate som førte til at lys ble reflektert fra sidene av markøren. Den store tykkelsen på 10 mm førte også til større skygger. Disse faktorene påvirket hvordan punktet ble detektert.

En annen ulempe ved prototype 1 var at nålen som stikker inn i målskiven kunne være litt skjev, noe som går utover presisjon. I tillegg hadde ikke nålen nok friksjon mot hullet den skapte i målskiven til at den satt godt fast. Disse to faktorene medførte at en ny iterasjon av design måtte utføres. Det er viktig å bemerke at et referanseverktøy trengs for å kunne plassere markøren riktig.

15.5.2 Prototype 2

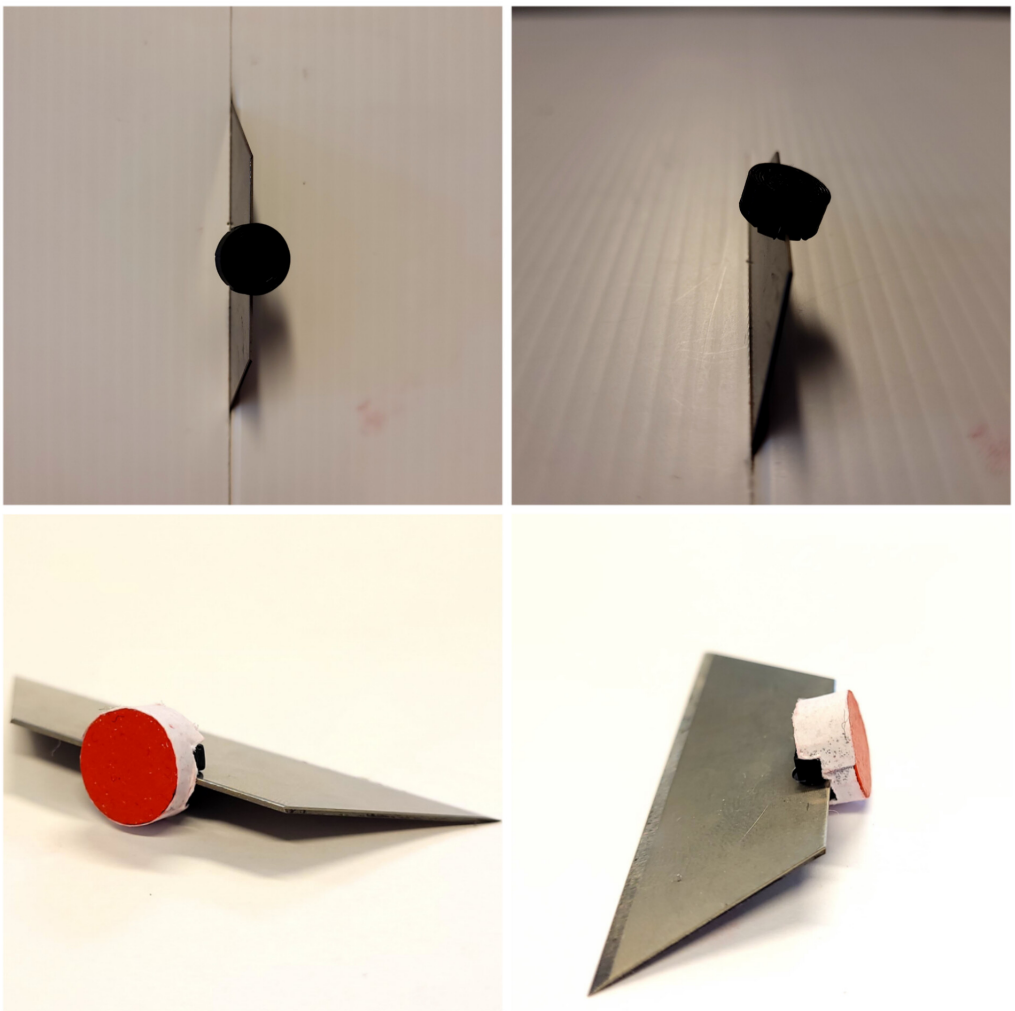
I designet av prototype 2 ble det forbedret i områder som tidligere var utilstrekkelige utformet i prototype 1. Tykkelsen ble redusert til 5 mm, for å minimere mengden lys som treffer sidene av målskiven, og skape mindre skygger. Ved å ha 2 nåler fremfor én, ble markøren mer stabil å plassere ved aksekorset. Se figur 65. Med 2 nåler sitter markøren godt fast i målskiven. Prototype 2 er avhengig av et referanseverktøy for å kunne plasseres riktig.



Figur 65: Prototype 2 av referansemarkør.

15.5.3 Prototype 3

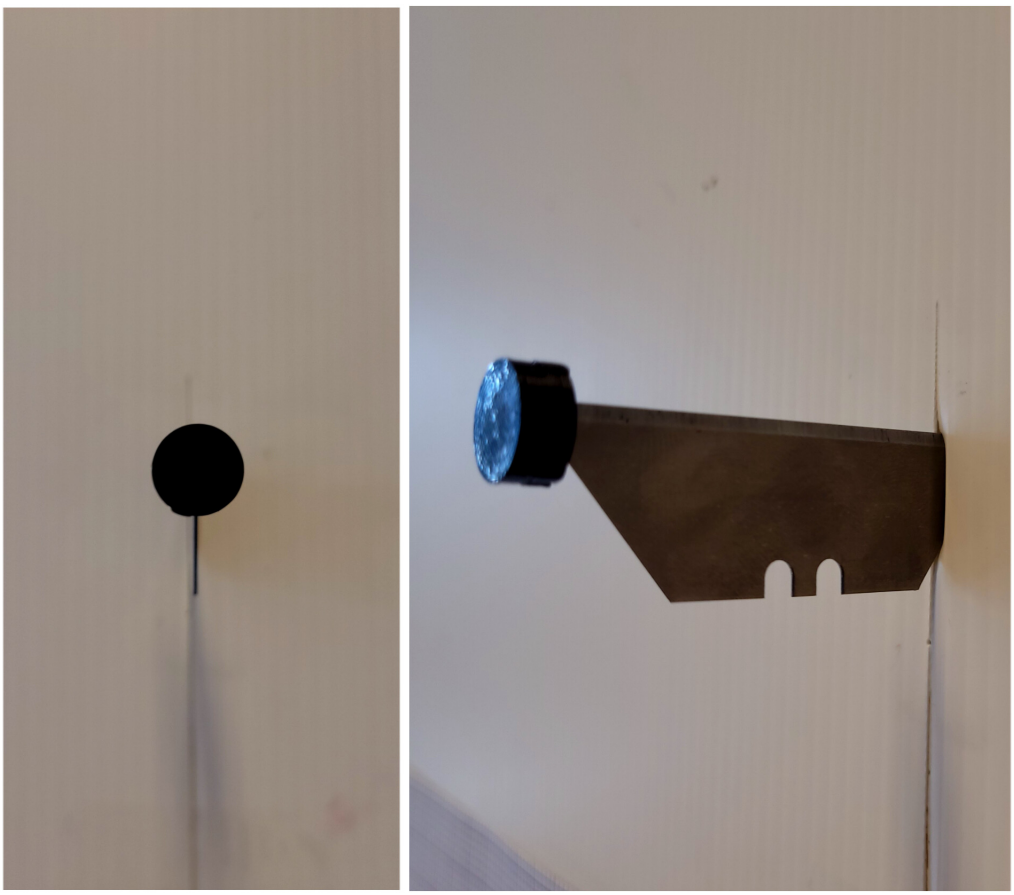
I designet av prototype 3 ville man undersøke hvorvidt man kunne bruke et tynt blad for å plassere referansemarkøren i selve aksekorset. Da ville man unngått å bruke et referanseverktøy. Se figur 66 og figur 67 for å se en mulig utforming av denne. Siden aksekorset er skåret av en kniv, vil tykkelsen til bladet føre til at en av kantene på innkuttet deformeres, og bladet blir skjevt. Det samme problemet oppstår ved den horisontale typen, men denne er i det minste vinkelrett i forhold til målskiven. For å kunne bruke et slikt blad for referansepunktmarkør må utformingen på selve målskiven endres. I stedet for å bruke en kniv til å skjære et snitt for aksekorset, må det brukes en sag, der tykkelsen til sagen er avgjørende. I tillegg bør innkuttet for aksekorset gå fullstendig gjennom målskiven. En sag vil fjerne materiale fra målskiven slik at tykkelsen til bladet får plass i aksekorsgløttet. Da vil man kunne sette inn et blad med litt større tykkelse enn sagbladet, og få en solid referansemarkering.



Figur 66: Prototype 3 med vertikalt blad av referansemarkør.

15.6 Maskinteknisk design av referanseverktøy

Referanseverktøyet ble utformet for å hjelpe brukeren plassere referansemarkørene presist. Utviklingen av referanseverktøyet har foregått over 4 iterasjoner, og 4 prototyper, der den endelig designet er prototype 4. Utformingen og størrelsen til verktøyet er i stor grad bestemt av hva som er passelig for brukeren å manipulere med hånden. Med et referanseverktøy kan man også plassere referansemarkøren slik at denne har en fast avstand til sirkelperiferien av siktеblinken.



Figur 67: Prototype 3 med horisontalt blad av referansemarkør. Deformasjonen ved bladets penetrering inn i målskiven fører til skjevhet.

15.6.1 Prototype 1

Prototype 1 fungerer ved å plassere referansepunktet inn i ett hull, samtidig som et gløtte i verktøyet veileder brukeren til å plassere markøren på aksekorslinja. Prototype en er tilpasset en 400 mm diameter på sikteblinken. Se figur 68.

Figur 68: Prototype 1 av referanseverktøy.

15.6.2 Prototype 2

Ved prototype 2 ble designet av referanseverktøyet forbedret ved å lage en brakett som veileder referansemarkøren. Da vil man kunne sikre at midtpunktet på markøren faktisk treffer linjen på aksekorset. I tillegg er det laget 2 halvsirkler som tangerer sirkelperiferien, slik at verktøyet kan brukes på enhver sirkulær sikteblink. Se figur 69.

Figur 69: Prototype 2 av referanseverktøy.

15.6.3 Prototype 3

Ved prototype 3 ble verktøyet inkrementert ved å fjerne en av sidene på verktøyet slik at det ble enklere for brukeren å se aksekorslinjen, når man skal sette referanseverktøyet inntil aksekorslinjen. I tillegg ble referansehullet fra tidligere iterasjoner gjort om til et spor slik at man kan lettere fjerne verktøyet uten å påvirke referansemarkøren. Se figur 70.

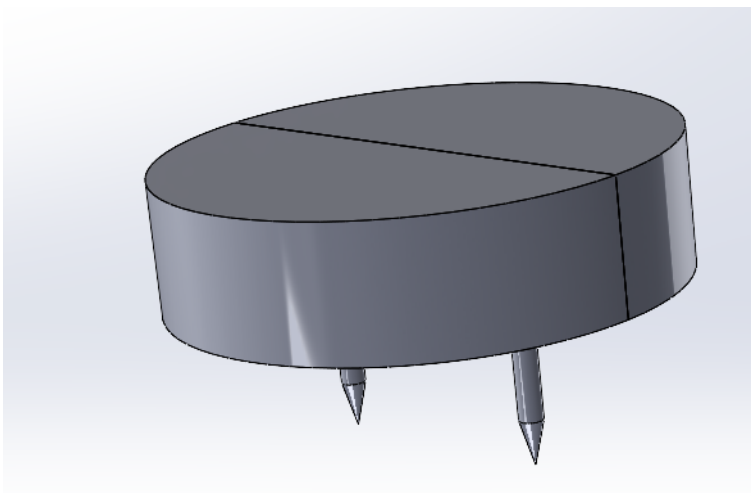
Figur 70: Prototype 3 av referanseverktøy.

15.7 Maskinteknisk design av referansepunktsystem

Det maskintekniske designet utarbeidet gjennom testing og utvikling resulterte i et referanseverktøy og en referansepunktmarkør. Det endelige designet av disse, samt mulige forbedringer diskuteres under dette avsnittet.

15.7.1 Referansemarkør

Referansemarkøren ble designet for å synlig gjøre aksekorset for å finne midtpunktet til målskiven. Det har blitt gjennomført 4 iterasjoner ved design av referansemarkører. Den 4 iterasjonen baserte seg på å gjøre diameteren til markøren større. Dette ble gjort på grunnlag av at markøren skal bli bedre å holde i for bruker. Det ble også gjort endringer på avstanden hullene nålene skal plasseres i, som følge av diameter endringene til referansemarkøren. Se figur 15.7.1.



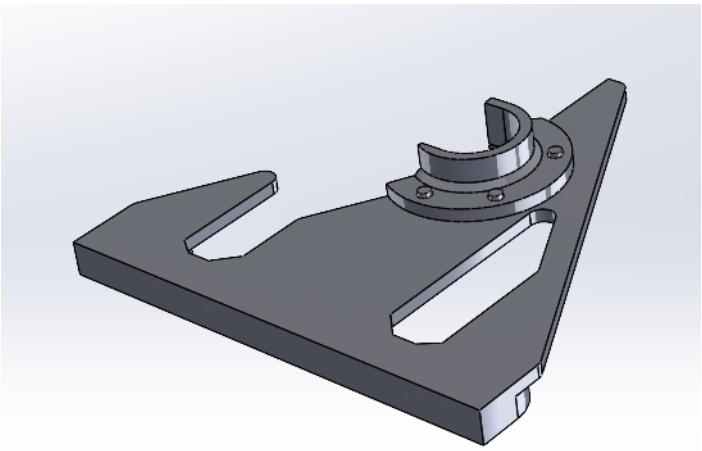
Figur 71: Endelig design av referansemarkør.

15.7.2 Referanseverktøy

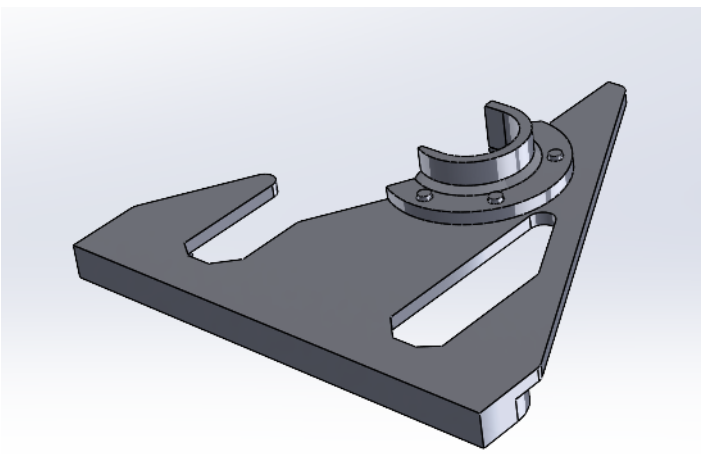
Referanseverktøyet ble utviklet for å sikre at brukeren posisjonerer referansemarkøren korrekt og presist. Det endelige designet på

referanseverkøyet er illustrert nedenfor.

Referansemarkøren ble designet for å synlig gjøre aksekorsen for å finne midtpunktet til målskiven. Det har blitt gjennomført 4 iterasjoner ved design av refernasemakrører. Den 4 iterasjonen baserte seg på å gjøre diameteren til markøren større. Dette ble gjort på grunnlag av at markøren skal bli bedre å holde i for bruker. Det ble også gjort endringer på avstanden hullene nålene skal plasseres i, som følge av endringen av størrelsen til referansemarkøren. Se figur 15.7.2 og figur 73.



Figur 72: Endelig design av referanseverktøy sett ovenfra.



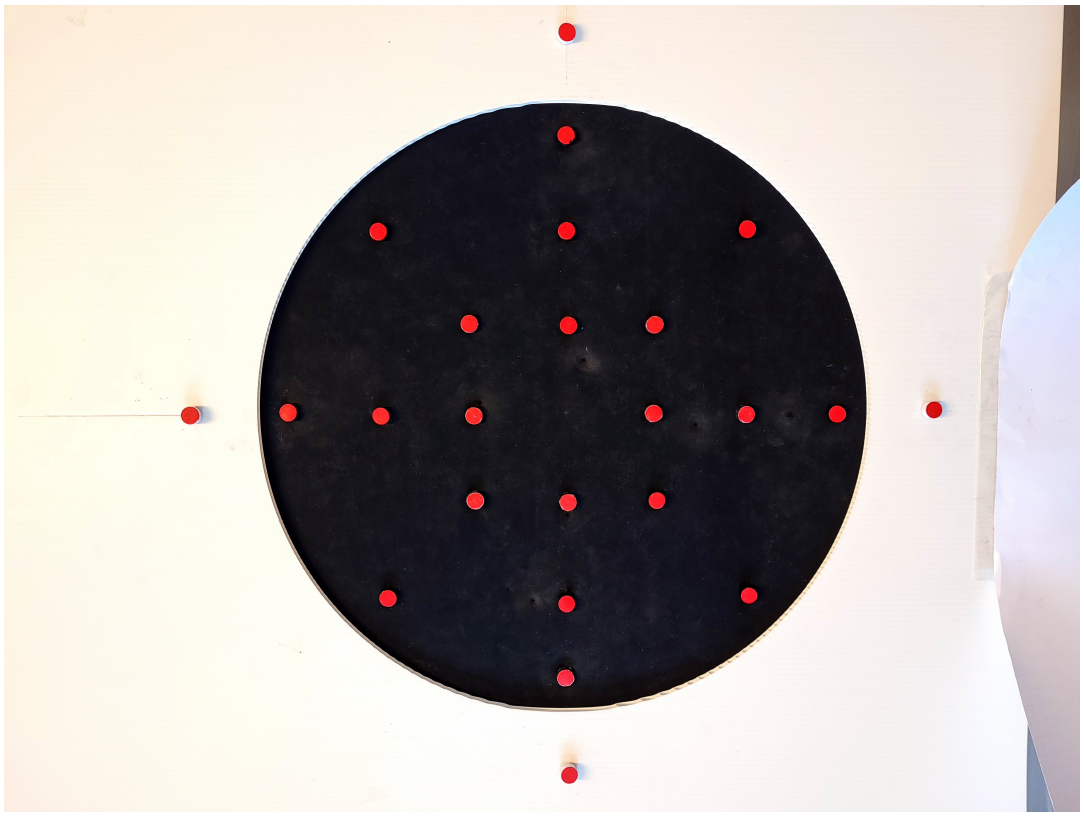
Figur 73: Endelig design av referanseverktøy sett underfra.

16 Avviksanalyser

Avviksanalyse på programvaren er en viktig prosess for å evaluere nøyaktigheten og kvaliteten på programvaren, samt undersøke hvilke begrensninger programvaren har med ulike inputdata. Med ulike inputdata menes ulike oppløsninger på bildet og ulike mobilkameraer. Ved denne analysen undersøkes avviket vårt system har i forhold til de faktiske treffpunktene. Ifølge krav FK-A8 så er makimalt tillat avvik lik 2mm. Det vil si at det er utført en manuell kontrollmåling av posisjonen til alle treffpunkter i x- og y-koordinater. Deretter sammenlignes kontrollverdiene på treffpunktene opp mot verdiene programvaren beregner ved bildebehandling. I denne analysen har man undersøkt dette avviket med varierende mobiltelefoner, og varierende oppløsning. Avstanden fra målskiven er mellom 2m til 2.5m.

16.1 Testoppsett

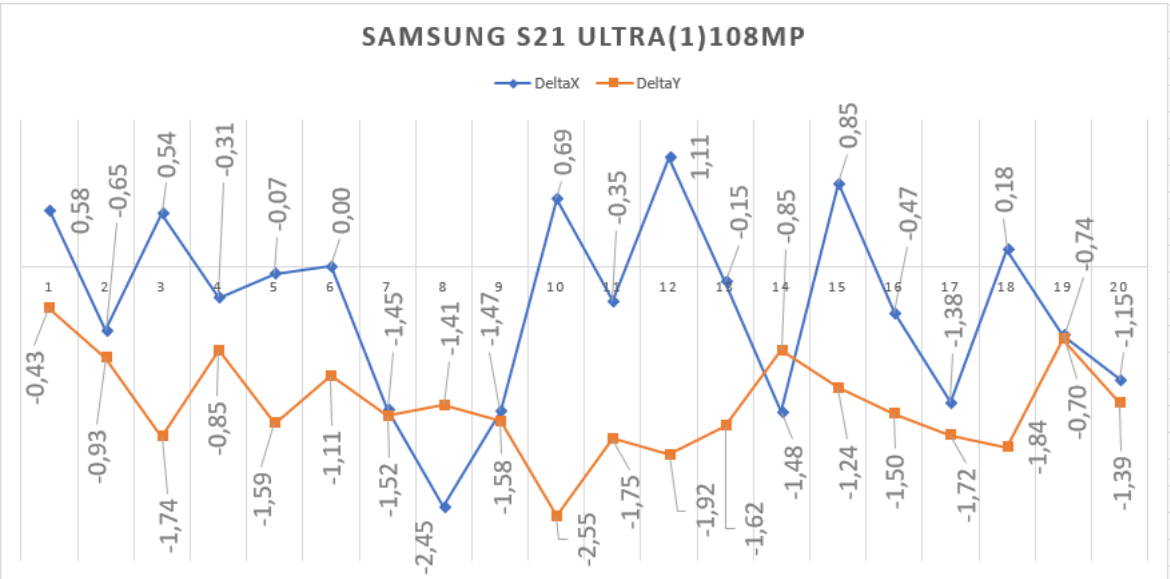
Se figur 16.1 som viser testoppsettet. Avstanden kameraet har til målskiven er mellom 2m til 2.5m. Dette er spesifisert under hver graf i resultatene. For å kontrollmåle posisjonen til hvert treffpunkt er det brukt verktøy som skyvelære og sytråd med lodd. Et papir med aksekorslinjer og rutenett er satt opp på målskiven. Aksekorslinjene på papiret er deretter posisjonert slik at de passer sammen med det faktiske aksekorset. Da vil senteret av papiret være likt senteret av målskiven. Deretter måles posisjonen (x, y) i millimeter til alle treffpunktene ut fra senter til midten av treffpunktet.



Figur 74: Målskive med treffpunktmarkører til testing.

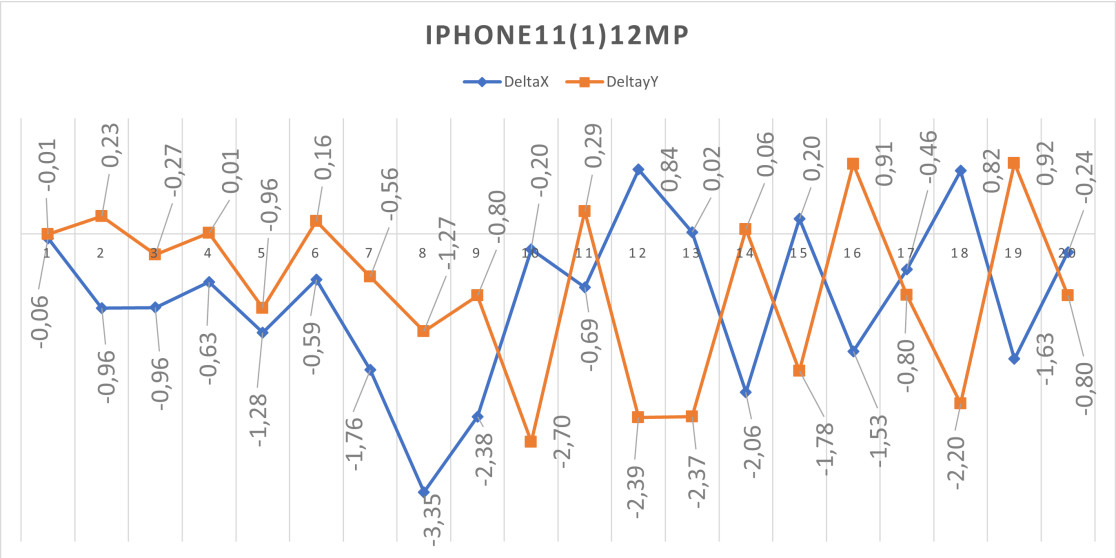
16.2 Resultater

Basert på tester som er gjort så er avvikene over kravet når treffmarkørene befinner seg utenfor senter kvadratet. Treffmarkører som markerer dette område er representert som treffmarkør nummer 5,6,7,11,15,16,17 og 18. Når man observerer disse treffmarkørene på grafen test 1 graf"og avvikdata A. Så ser man en tendens til at markører rundt dette senter kvadratet som regel er innenfor grensen på 2mm.

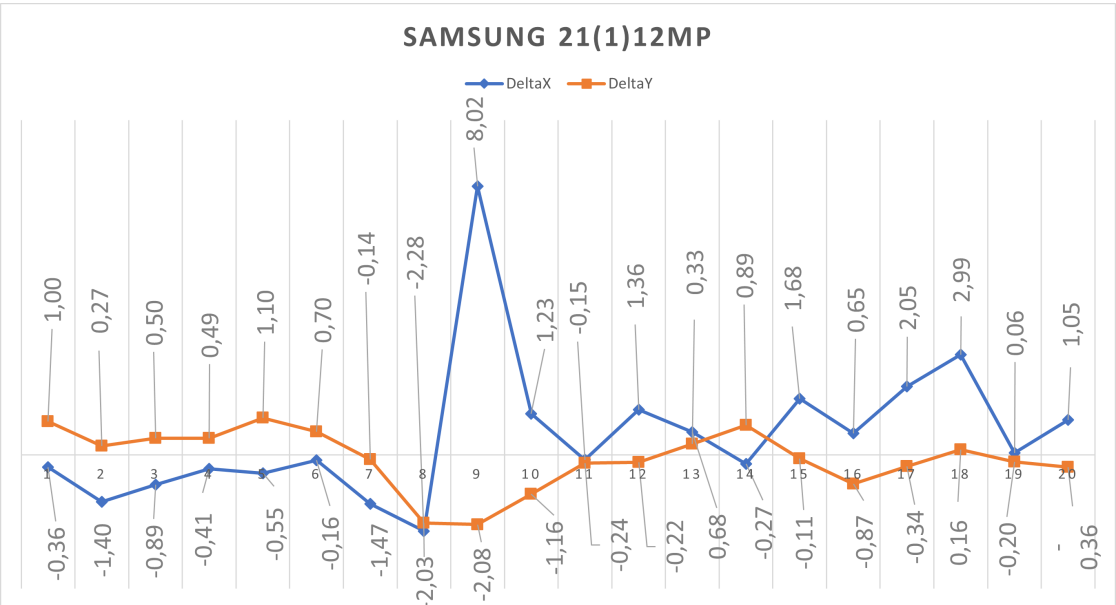


Figur 75: Test 1,Samsung s21 ultra graf

Denne tilnærmingen er basert på flere tester både for flere oppløsninger og mobiler. Det er også viktig å påpeke at testene er utført innenfor en gitt distanse intervall i forhold til skyteskiven. For oppløsningen 108MP så er distansen $2,45\,\text{m} \pm 0,10\,\text{m}$. Videre så er distansen $2\,\text{m} \pm 0,1\,\text{m}$ brukt for 12MP både på iphone11 og samsung s21. Under kan en igjen observere at treffmarkørene innen senter kvadratet oppfyller FK-A8 kravet.



Figur 76: Test, Iphone 11 graf



Figur 77: Test, samsung s21 graf

16.3 Svakheter med algoritmen

Basert på tester som er gjort så er det observert 3 svakheter hvor alle 3 svakhetene henger sammen. Den ene er distansen i forhold til skyteskiven,

svakheten i dette tilfellet er at intervallet er ikke stort nok. Dette gjør det vanskelig for kunder å kalibrere systemet på en effektiv måte. Videre så påvirker også distansen RGB farge deteksjonen i algoritmen siden fargene ikke blir like klare. En annen svakhet med systemet er at algoritmen ikke tar hensyn til grader som er over 5 grader. Basert på tester som er gjort så er det funnet ut at det kan være vanskelig å holde kamera i cirka 90 grader i forhold til skyteskiven.

For å konkludere avviksanalyse og sikre at skytingen er i overensstemmelse med de fastsatte kravene, så er det valgt regne ut forventningsverdi for alle deltaX og delta.(se slutten av kapittel A. Denne tilnærmingen bidrar til å opprettholde en balanse mellom akseptabel avvikstoleranse og oppnådd treffsikkerhet, og sikrer dermed pålitelige og konsistente resultater innenfor FK-A8 kravet.

17 Brukerinstruks - ImPro kalibreringssystem

I følge krav FK-A6 i kravspesifikasjonen skal det utarbeides en brukerinstruks for systemet. Brukerinstruksen skal bidra til a brukeren får en tydelig veiledning i hvordan systemet skal brukes. Veiledningen gjelder kun ImPro systemet, og tar ikke høyde for eventuelle operasjoner som må utføres av brukeren på KTS sin målskive. Se vedlegget i kapittel G.

18 App

Applikasjonen vi lager skal ha en interaksjon mellom brukeren og systemet vi utvikler. Programvaren skal være en app som brukeren kan bruke til å . Valget falt på å lage en app vi allerede bruker telefonen til å ta bilde. Det var også uttrykt et ønske fra oppdragsgiver at det beste hadde vært om man både kunne ta bilde med telefon og at brukergrensesnittet også er på telefon. I og med at vi oppdaget at det var mulig å ta bilde med telefon var det naturlig at også grensesnittet var på telefon.
Til å utvikle appen så bruker vi Javascript med et bibliotek som heter React Native. Vi startet å utvikle den i Android studio med java som programmeringsspråk, men etter en del utfordringer og noe ny informasjon valgte vi å gå bort fra dette og over til Javascript.

18.1 Funksjoner

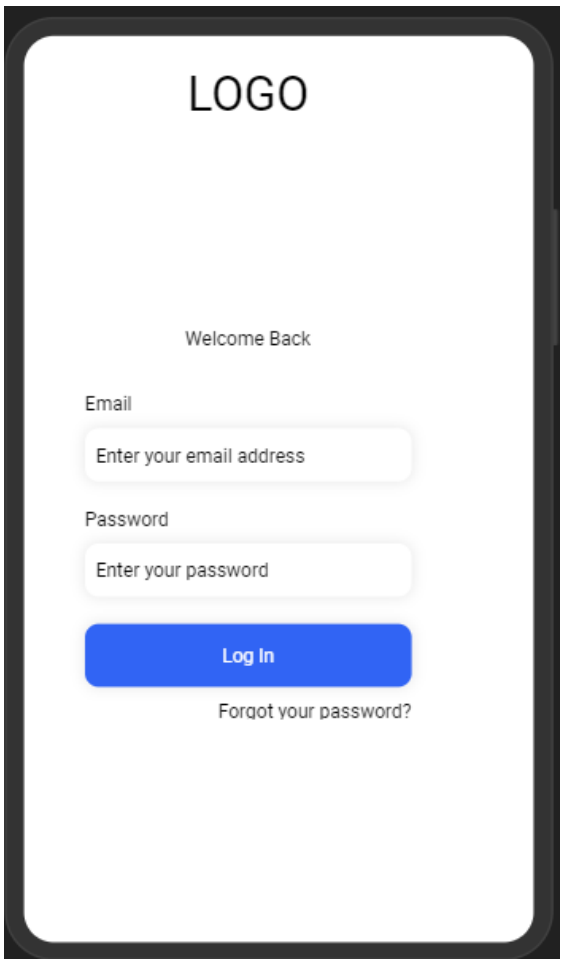
Inne i appen vår er det noen funksjoner som vi ser på som essensielle for at appen skal fungere som vi ønsker. Noen av disse funksjonen er:
Ta inn verdier manuelt: - Dette er verdier som kommer fra oppdragsgiver sitt eksisterende system som vi trenger for å sammenligne med verdier som vi gjør i vårt system
Presentere verdier: - Vi ønsker at verdiene som vi får ut ifra bildet skal kunne presenteres for brukeren inne i programvaren.
Presentere resultater: - Det skal også presentere resultater som blir gjort i form av bildebehandlingen.
Ta bilde: - Bruker skal kunne ta bilde inne i appen slik at det blir behandlet, og disse bildene skal kunne lagres i en mappe som brukeren har adgang til.
Lagre bilde inne i appen/database: - Det er ønskelig at bildene som blir behandlet skal lagres slik at man kan gå tilbake og kontroll sjekke i ettertid hvis det er nødvendig eller ønskelig.

18.2 App design

Før vi starter å lage appen, så er det viktig å lage et design slik at man har en plan på hvordan det skal se ut. Med et design så kan man tidlig oppdage utfordringer og problemer som man da kan fikse opp i tidlig. Vi lagde designet tidlig utifra hva vi mente at bruker ville trenge for at systemet skulle funke. Under designet og utviklingen av appen har vi fokusert på brukervennlighet, effektivitet og minimalistisk. Appen er ikke det viktigste av systemet vårt, så vi ønsker ikke å legge så alt for mye tid i å gjøre appen for avansert og ha unødvendige funksjoner.
Det første utkastet av appen ble laget i wireframe. Wireframe er et veldig enkelt format for å designe.
Nedenfor vises de forskjellige skjermene på hvordan vi ønsker at det skal se ut. I resultater vil det være hvordan det ender opp med å se ut og forklaringer på hva de forskjellige skjermene gjør og generell forklaring av appen.

18.2.1 Innlogging

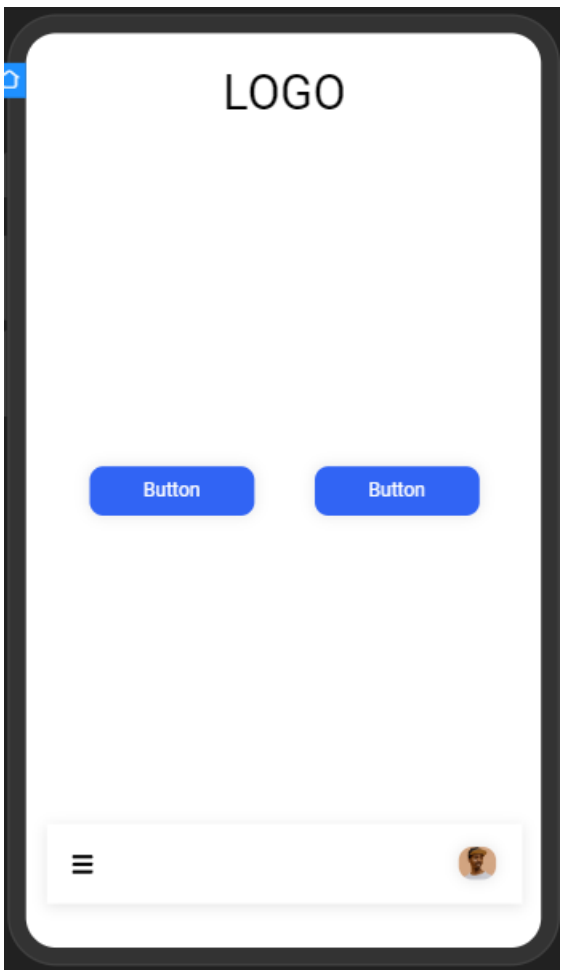
En innloggingsside hvor brukeren blir nødt til å logge inn eller lage bruker.



Figur 78: Innlogging

18.2.2 Hjem

Dette skal være den første skjermen bruker møter etter å ha logget inn. Herifra skal bruker kunne navigere seg videre.



Figur 79: Hjem skjerm

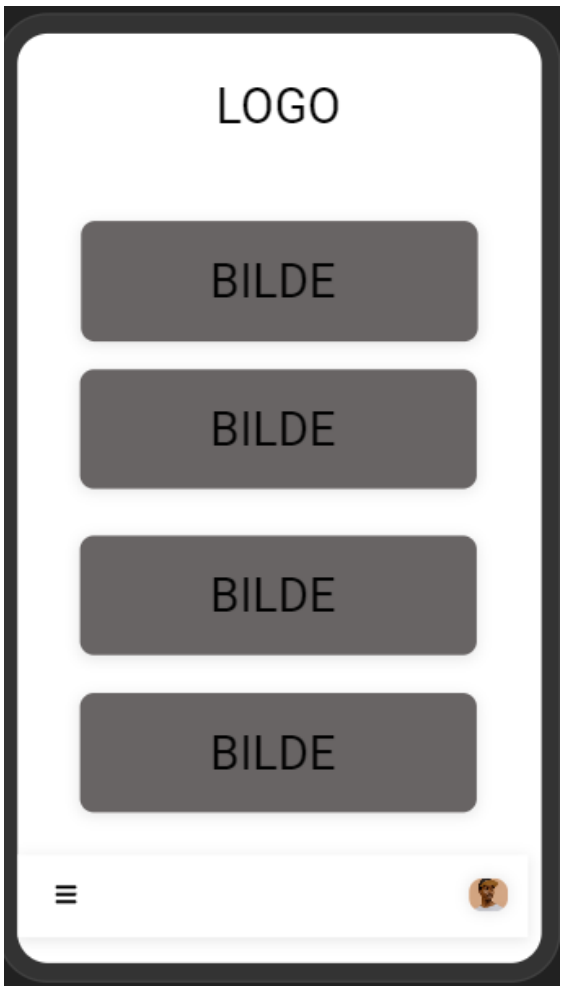
18.2.3 Kamera

Her skal kamera komme opp slik at brukeren kan ta bilde.

Figur 80: Kamera

18.2.4 Galleri

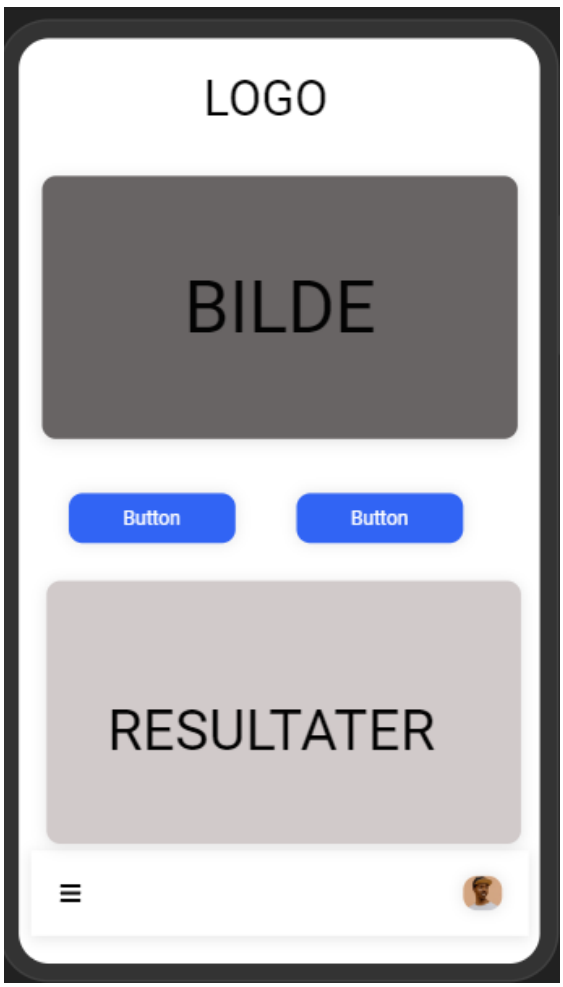
Et galleri hvor bilder som er tatt blir lagret.



Figur 81: Galleri

18.2.5 Resultat

En resultatside som brukeren skal kunne se hvor treffpunktene har blitt detektert og samtidig få ut verdiene til treffpunktene.

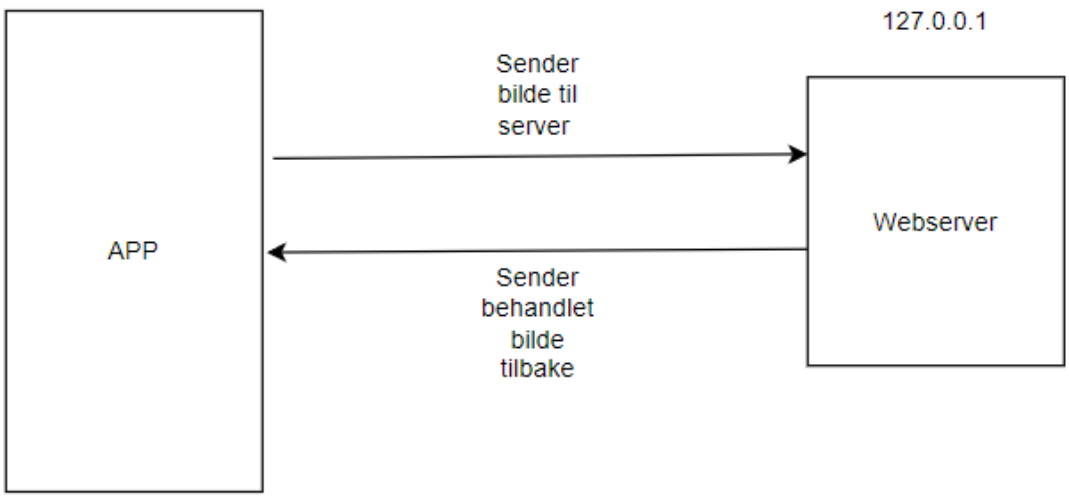


Figur 82: Resultatskjerm

18.3 Integrasjon

For at hele systemet skal fungere sammen så må vi integrere appen sammen med vår backend. Det som foregår i backend blir gjort i Python og appen er laget i Javascript. For at vi skal få disse til å jobbe sammen må vi bruke API som sender bilde til en webserver for så å kunne det bilde i Python programmet.

Under ligger det en liten illustrasjon på hvordan det planlegges:



Figur 83: Illustrajon av integrering

Her ser man at bildet som blir tatt inne i appen blir sendt til webserver som kjører Python. Når det sendes så kan Python programmet motta bilde og gjøre det som skal gjøres med bilde. Når bilde er ferdig behandlet og vi har fått treffpunktene og det vi trenger, sendes bilde tilbake til appen.

18.4 Resultat av integrasjon

Resultatet av integrasjonen gikk ikke helt som ønsket og forventet. Målet med integrasjonen var at med det så skulle vi få et helhetlig system som gjør at

frontend og backend jobber sammen. Dette var noe vi ikke lyktes til å få gjort.

Det er flere grunner til at vi ikke ble ferdig med integrasjonen. Vi endte med å gå frem og tilbake på hvordan vi ønsket at hele systemet skulle fungere. Vi startet med den tanken på at vi skal ha en kobling mellom frontend og backend, men etter et ønske fra oppdragsgiver i midten av april, gikk vi bort fra det. Det som kom fra ekstern veileder var at han ønsket at alt av systemet skal skje direkte inn i appen. Det vil si at han ønsket at også bildebehandlingen også skulle skje inne i appen. Argumentet for dette var at skytebanene rundt om i verden nødvendigvis ikke har internettilgang.

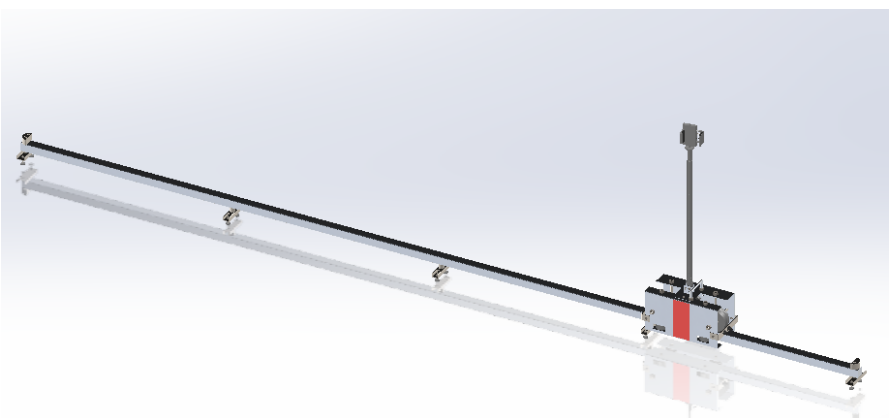
Etter dette tok vi en vurdering på om vi skulle prøve å innfri dette ønske, eller fortsette med vår originale plan, som var å bruke API for å koble delene av systemet sammen. Vi valgt å prøve å innfri dette fordi vi så på det som et godt argument det som veileder kom med at vi ville forsøke.

Etter å ha sett litt på det så vi at det ikke ville funke å endre all kode som blir gjort i Python over til Javascript. Javascript ville ikke kunne håndtere det vi hadde gjort i Python. På grunn av all den fram og tilbake gjøringen, gjorde at vi brukte mye tid som vi ikke burde har gjort.

19 Mekansik system

For at et mekanisk system skal forenkle kalibreringsprosessen ved deteksjon av treffpunktene. Er det mulig å unngå prosessen som baserer seg på å gå bort til skyteskive etter at skiven er påskutt, for å plassere fysiske markører inni treffpunktene, og deretter ta et bilde. Dette vil ta tid hvis man står og skyter fra en lengre distanse eller hvis man skal kalibrere flere skyteskiver samtidig. Dette kan løses ved at man tildekker skyteskiven med en form for maling o.l. på forhånd som er usynlig for skytter, men synlig for kamera ved biledtagning. En annen mulighet er å bruke et høyoppløsningskamera, men dette er veldig dyrt, så dette ser man på som lite realistisk å bruke. Et annet aspekt som må legges til grunn er monteringstiden av det mekaniske systemet, som kameraet skal festes på. Hvis det tar like lang tid å montere, som det gjør å gå bort til skyteskiven er en slik mekanisk system ubrukelig. Ut fra kriteriene ovenfor har en skinnegang blitt vurdert der kameraet vil bevege seg sideveis på skinnene. Dette vil føre til at man må gå bort til skyteskiven for å montere skinnegangen innenfor en gitt avstand. Dette ser vi på som et lite problem siden vi tar utgangspunkt i at skyteskiven er uten skuddhull når vi skal kalibrer skyteskiven. Derfor må man gå bort til skyteskiven for å plassere en ny blink uavhengig av løsning. Det negativ med dette er at et system som går på skinner vil være ganske tungt å frakte ut til skyteskiven og det må i tillegg monteres når man kommer fram til skyteskiven. Et annet alternativ er å bruke en drone. Denne dronen skal fly bort til skyteskiven å ta et bilde, slik at skytter slipper å gå bort til skyteskiven for å sette inn markørene i skuddhullene. Fordelen med dette kontra skinnegang er at en drone er lett å ta med seg og tar liten plass, og den slipper å bli montert for hver gang den skal brukes. Minuset her vil antakelig være prisen, selv om man i dag kan få en drone til noen tusen kroner. Prisforskjellen mellom skinnegangen og dronen vil bli sett på å være ganske liten, derfor så vil design av drone være mest hensiktsmessig å bruke.

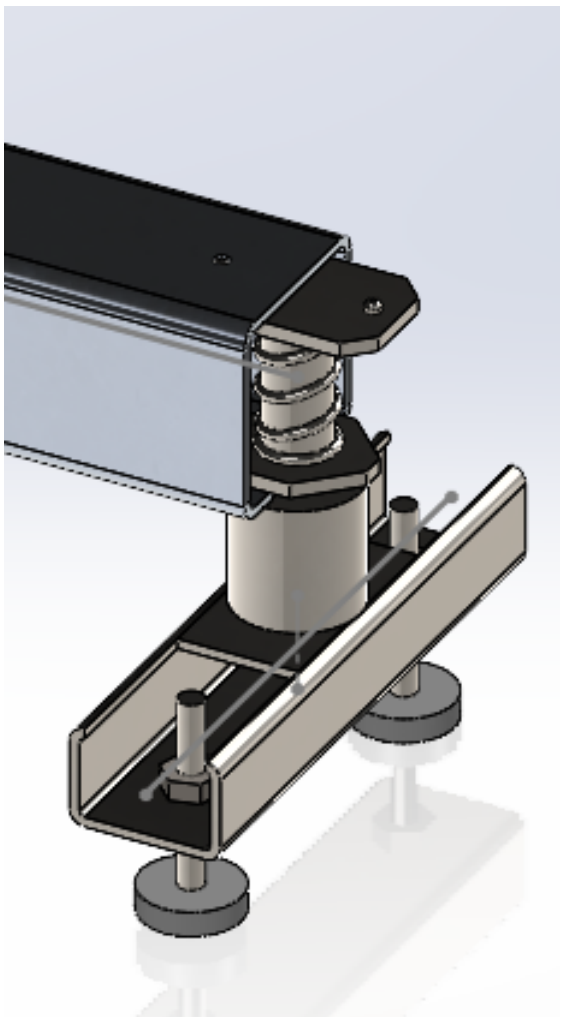
Siden droner er lett tilgjengelig er det ikke hensiktsmessig å designe et slikt hjelpemiddel, også med hensyn til tid av bacheloroppgaven. Ved å designe en skinnegang kan det også bli brukt til andre ting enn å feste en telefon. Et eksempel er at man kan ha en annen form for festeanordning istedenfor et kamerastativ, slik at man kan bruke systemet også som en bevegelig blink. Derfor blir design av skinnegang valgt som en mekanisk konsept. Når et slikt fysisk system skal designes er det viktig at systemet holder seg innenfor både vekt kravet på 15 kg og kravet om at systemet skal monteres innen 5 min for at et slikt system skal være aktuelt å designe. Ut fra dette kan man designe et tog system som skal være med på å forenkle kalibreringsprosessen. For å kunne oppfylle kravet angående vekt er material bruk et viktig aspekt. Det er derfor viktig å bruke små dimensjoner og lette materialer, og samtidig som materialet er værbestandig med hensyn til korrosjon. Siden aluminium er et material som har bedre korrosjonsbestandighet enn f.eks. stål, og i tillegg har lavere massetetthet gjør at aluminium er et hensiktsmessig valg av material bruk for de største delene til skinnegangen. Materialer brukt i sammenstillingen er hentet fra bibilioteket i SolidWorks. Material egenskapene av Tough PLA er hentet fra et datablad. **[?]**



Figur 84: Mekanisk togsystem

Siden skinnene er 1 m per enhet, så blir det plassert et bein for hver meter for å koble og støtte oppunder skinnene. Beinet er designet for å hurtig koble

sammen skinnene uten bruk av verktøy. Dette gjøres enkelt ved å dytte fjæren oppover, slik at den sklir inn i sporet til firkantrøret/ skinnen. Ved produksjon er stål ASIS 1020 mest hensiktsmessig å bruke med tanke på gode sveiseegenskaper. Sveisingen fører til mindre material bruk under produksjon av beinet. [**?**]



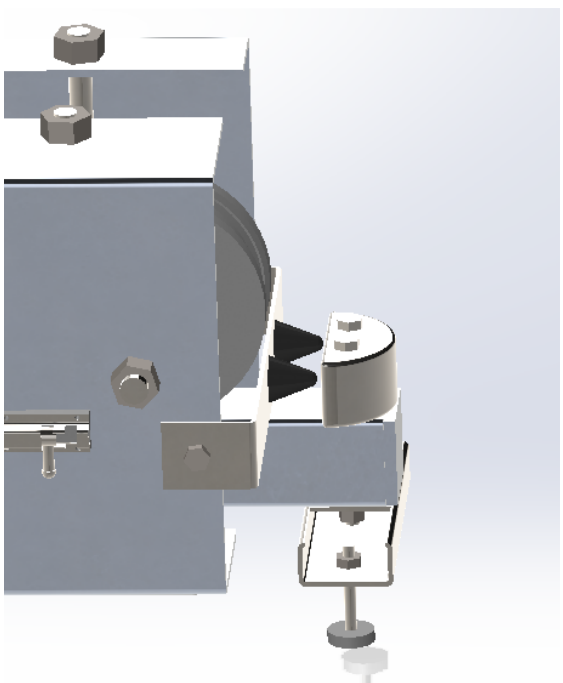
Figur 85: Bein

Selve skallet til toget er laget av 3mm tynnplater 1060 aluminium. Her var det mye vekt å spare ifølge en funkjson i solidworks kalt evaluation". Ved bruk av AISI 1020 stål økte vekten fra 768 g til 1,8 kg. Siden aluminiumstynnplater blir brukt er det lett å bøye karosseriet etter ønsket form. Dette gjør produksjonen

billigere sammenlignet ved maskinering pga materialtap og prisen man betaler for tjenesten. [**?**]

Akslinger er laget av AISI 1020 stål for å kunne ha en god nok stivhet og styrke ved rotasjon. Selv om aluminium er lettere er ikke vekt besparelsen stor nok for at dette skal bli valgt ovenfor AISI 1020 stål. En annen faktor som spiller inn, er at AISI 1020 er et mye brukt materialet til maskinering av nettopp akslinger. [**?**]

For at ikke toget skal kunne kjøre av skinnene er hver ende blokkert med en slags sikkerhetsbrakett. Denne braketten har som funksjon å stoppe toget fra å spore av skinnene. På toget er det en bøyle med to tapper av gummi for å dempe kollisjon ved enden av skinnene. Her er det viktig at materialet til braketten tåler et evt samenstøtt med toget. Derfor blir braketten produsert i AISI 1020 stål.



Figur 86: Sikkerhetsbrakett for toget

Stativet for kamerat er designet i tough PLA, som er en type 3D-print plast. Dette handler om at stativet ikke blir utsatt for noen krefter. En fordel ved bruk av plast er at det ikke korroderer, og det er veldig lett sammenlignet med

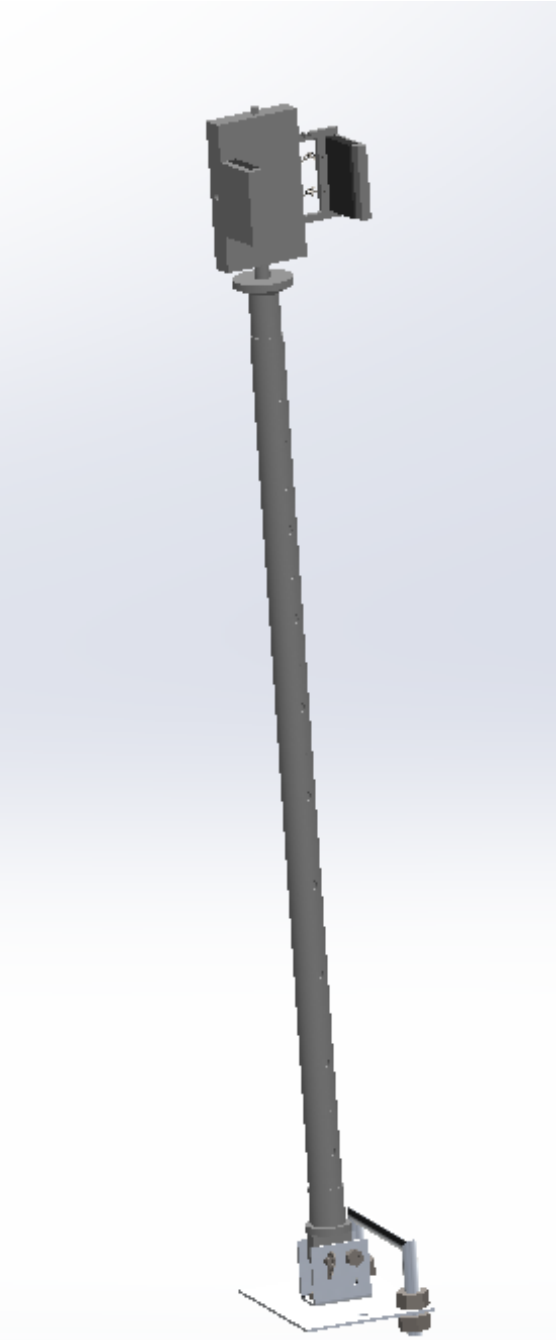
metalliske materialer. Dette stativet er festet til toget med to bolter, en bolt med mutter og en med splint. Hensikten med bolten og splinten er at man kan dra den raskt ut slik at man kan rotere stativet 90 grader for at det blir sammenleggbart. Bolten blir plassert i et annet hull som er parallelt med bolten med mutter i. Dette gjøre det lettere å frakte toget. Kamerastativet er også justerbar i høyde fra 600mm til 1100mm.

På toget er det 3 hjul horisontalt på skinnene, en på midten som er drivhjul og to hjul på hver side av drivhjulet. Disse hjulene er tiltenkt å bli 3D-printet laget av tough PLA. Drivhjulet har en svart gummi rundt for å få nok friksjon på skinnene for å kunne få framdrift. De to hjulene på hver side av drivhjulet er litt mindre. De har som hensikt å stabilisere toget slik at det ikke får en vaggete gange og/eller velter. På siden av skinnene er det to hjul på hver side. Disse er for å stabilisere toget for sideveis bevegelse, og for å sentrere hjulene oppe på skinnene. Toget plasseres enkelt på skinnene etter at skinnene er koblet sammen.

Motoren som skal drive toget er en 9 NM motor som veier 1.6 kg. [**?**] Motoren har fire festepunkter i hvert hjørne. Motoren festet på midten av karosseriet med 4*Ø6 bolter. Ut fra denne er det to koblinger som skal overføre kraften til et hjul for framdrift.

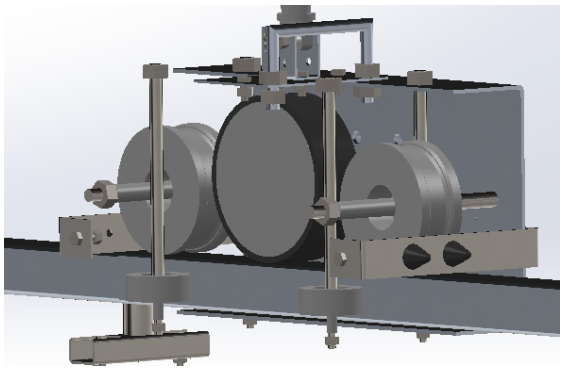
Batteriet som skal tilføre elektrisitet til motoren burde være enkel å ta av og på for å kunne lade. Dette gjøres ved to skyvelåser som enkelt fester batteriet til karosseriet til toget. Batteriet må ikke være for stort da vekten til hele systemet bør være lettest mulig (maksimalt 2 kg), men det må samtidig være stort nok for å unngå ladeprosessen i størst mulig grad. På siden av karosseriet vendt mot skytter er det markert en rød strekk vertikalt som skal gjøre det lettere for bruker å plassere midten av toget til midten av skyteskiven fra lengre avstander før bildetagning.

Vekten av hele systemet med 4 meters skinner (uten batteri og motor), som vist på bilde er det mekaniske systemet innefor vektkravet.

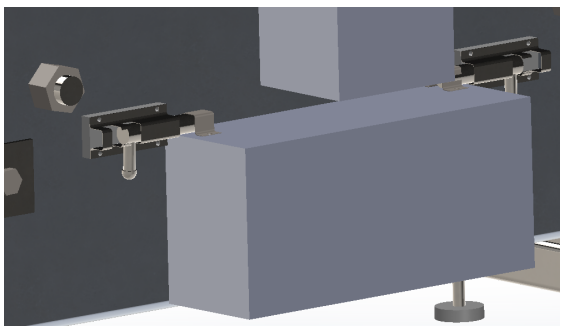


Figur 87: fig:image*label*

Ut fra funksjonen evaluation"i SolidWorks blir vekten ca. 5,2 kg. Dette er uten motor og batteri. Selv om vekten vil øke noe med batteri og motor. Vil den trolig ikke overstige 8,5 kg ved implementasjon av motoren og batteriet.



Figur 88: innsiden av toget



Figur 89: Innfesting av batteriet

Mass = 10507.80 grams

Volume = 4121836.79 cubic millimeters

Surface area = 2450914.00 square millimeters

Figur 90: Vekten av hele systemet med 4 meters skinner (uten batteri og motor

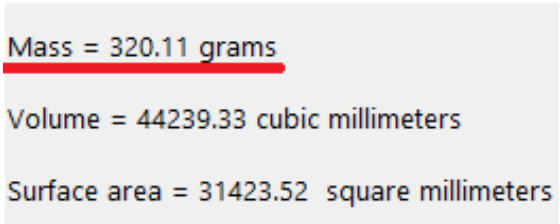
The center of mass and the moments of inertia are output in the coordinate syste
Mass = 5237.18 grams

Volume = 2652268.29 cubic millimeters

Surface area = 1091931.71 square millimeters

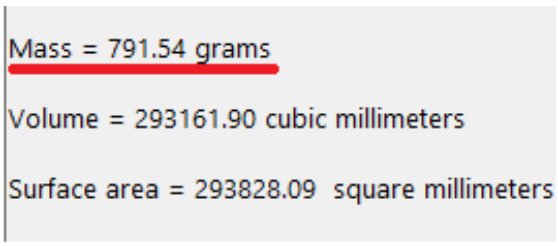
Figur 91: Vekten av toget

Vekten av beinet til skinnegangen



Figur 92: Vekten til beinet

Vekten av en skinne.



Figur 93: Vekt til firkantprofil

19.1 Resultat av app

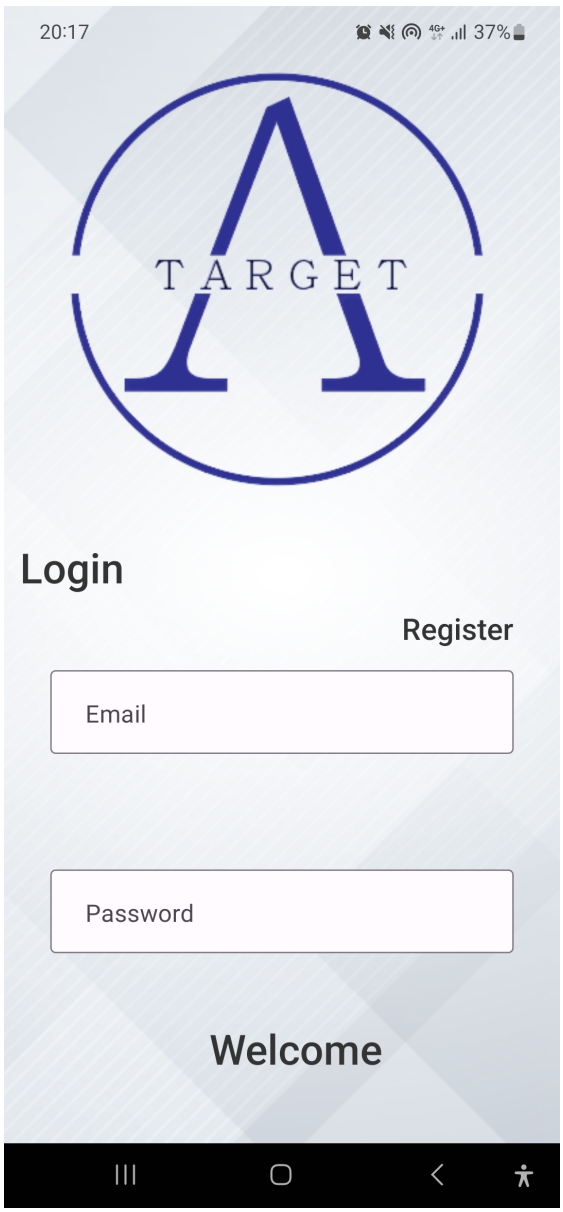
Appen har flere skjermer som brukeren kan navigere mellom for å bruke de ulike funksjonalitetene nevnt ovenfor. Den første skjermen er velkomst siden hvor innloggingsfeltet er og en knapp «Welcome» som bruker kan trykke på for å få tilgang til en annen skjerm hvor de ulike funksjonaliteter appen har ligger.

19.1.1 App.js

Dette er hovedfilen i React Native, og her ligger grunnkoden som muliggjør navigering i appen. For å navigere mellom de forskjellige skjermene i appen, benyttes pakken @react-navigation/native fra React Navigation-biblioteket. Den fungerer godt som en navigasjonsløsning. I tillegg til denne pakken, brukes også @react-navigation/native-stack. Ved hjelp av denne pakken kan en stack navigator implementeres, noe som gjør det mulig å håndtere alle skjermene som blir brukt i appen på en strukturert måte.

19.1.2 Login - skjerm

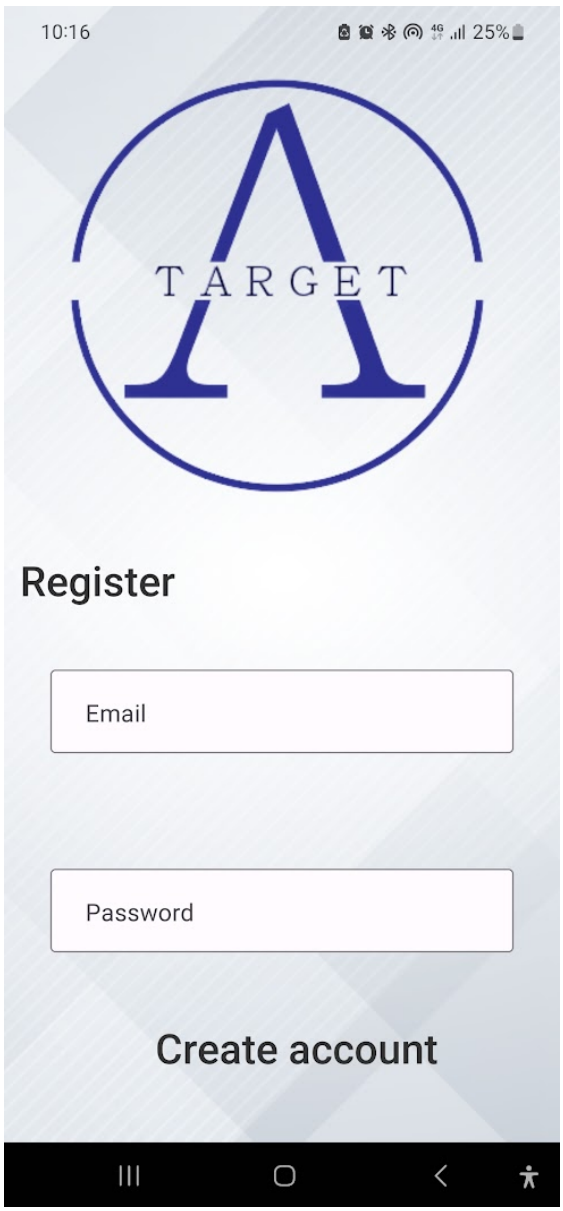
Logg inn - skjermen er velkomstsiden i appen, hvor brukeren må autentisere seg for å få tilgang til appens funksjonaliteter. Autentiseringsmetoden er satt opp med e-post og passord, men på grunn av begrenset tid og fokus på andre mer kritiske funksjonaliteter, ble ikke denne implementasjonen fullført. Tanken var å bruke Google sin Firebase-database for å lagre data knyttet til brukerne, slik som informasjon om skuddhistorikk.



Figur 94: Logg inn skjerm

19.1.3 Register

Registrer-skjermen skulle være stedet der brukerne kan opprette en brukerkonto hvis de ikke har en fra før. Men akkurat som Logg inn - skjermen, ble ikke denne funksjonaliteten fullført. Her var også planen å bruke Firebase.



Figur 95: Registerskjerm

19.1.4 Hjem - Skjerm

Hjem-skjermen er stedet hvor alle funksjonene i appen er tilgjengelige. Av den grunn har målet vært å prøve å holde den så enkel som mulig. Kamera- og gallerifunksjonene, som er funksjonalitene tilgjengelige i appen, er derfor framstilt i form av ikoner. Brukeren kan enkelt navigere til ønsket skjerm ved å trykke på de intuitive ikonene.

Figur 96: Hjem skjerm

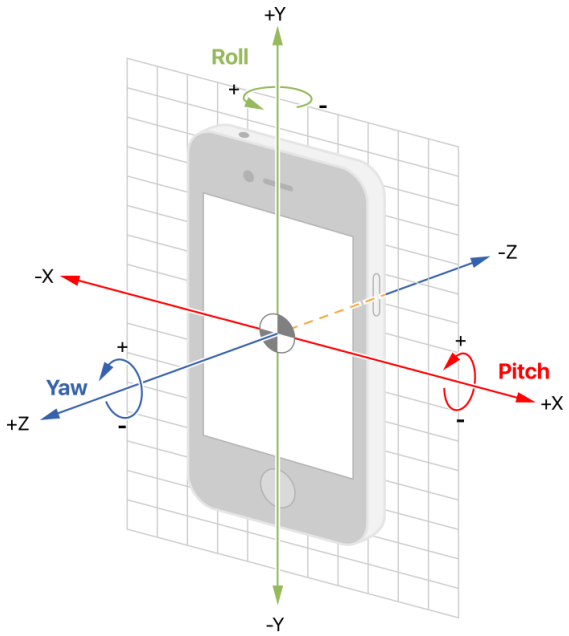
19.1.5 Kamera - Skjerm

En av funksjonalitetene som er tilgjengelig for brukeren, er kamerafunksjonen. Brukeren kan ta bilde, dele bildet, lagre bildet eller forkaste bildet. Kameraet har to overlay-komponenter: en sirkel, et aksekors med horisontale og vertikale linjer. Disse er ment som hjelpeverktøy for brukeren under bildeopptaket. Sirkelen er der for å visualisere største avstand brukeren kan ha fra skyteskiven når de tar bilde. Aksekorset skal hjelpe brukeren med orienteringen i forhold til skytefiguren, slik at de står riktig i forhold til flaten på skyteskiven.

For at systemet skal fungere som planlagt, er det nødvendig å ta hensyn til måten bildene av skyteskiven blir tatt på. Metoden for å løse dette problemet involverer bruk av telefonens akselerometersensor. Biblioteket expo-sensors"i React Native kan brukes til å få tilgang til denne sensoren. Akselerometer er en enhet som måler endring i fart eller endring i retning. Akselerometer i smarttelefoner måler den lineære akselerasjonen, altså endring i fart i en kjent retning. [11] Det er retningsaksene x, y og z det refereres til når det er snakk om retning.

I figuren ovenfor er det mulig å se de tre retningsaksene, sammen med tilhørende rotasjonsvinkler: Roll, pitch og yaw, som også er kjent som Euler-vinkler. Euler-vinkler beskriver hvordan et objekt orienterer seg. Euler-vinkler består opprinnelig av tre forskjellige vinkler: roll, pitch og yaw. [13] Hver av disse representerer rotasjon rundt en egen akse. Men på grunn av tidsbegrensninger og kompleksitet, ble kun pitch og yaw inkludert som en del av løsningen. Som illustrert i Figur 4, representerer pitch bevegelse fremover og tilbake, mens yaw representerer tilting av telefonen.

Akselerometersensoren fungerer på den måten at den gir akselerasjonen telefonen har i hver retningsakse. For å finne hvilken pitch- og yaw-vinkel dette representerer, er det nødvendig å bruke trigonometri med x-, y- og



Figur 97: Euler vinkler [12]

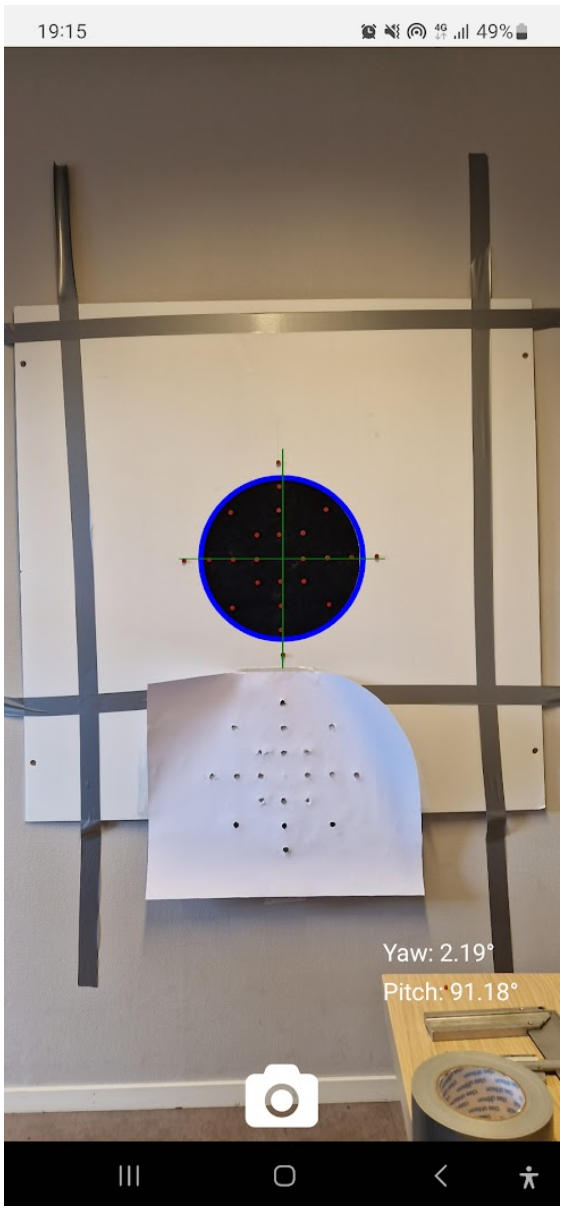
z-verdiene. Deretter genereres det kontinuerlige pitch- og yaw-vinkelavlesninger, som brukeren kan bruke til å orientere telefonen.

Denne løsningen fungerer og er god nok for bruk, men forbedringer kan gjøres. Det ville være mer hensiktsmessig om flere sensorer, som gyroskop og magnetometer, ble brukt sammen med akselerometeret for å få mest mulig presise målinger. Dette fenomenet er kjent som komplementærfilter, hvor målinger fra flere sensorer blandes for å generere et gjennomsnitt og dermed oppnå mer nøyaktige tallverdier. [14] Denne metoden vil mest sannsynlig være bedre, men den er også mer kompleks ettersom det er flere faktorer som må tas hensyn til.

Komplementærfilteret er ikke benyttet. I stedet er en annen metode anvendt for å gjøre dataene mer nøyaktige. Denne metoden involverer bruk av gjennomsnittet. Måten det fungerer på er at et sett med akselerometeravlesninger blir lagt inn i et datasett som deretter blir brukt til å jevne ut dataen ved at gjennomsnittet, som nevnt, tas. Dermed vil den siste vinkelverdien være en verdi som baserer seg på flere avlesninger. Med den

måten som denne metoden fungerer på, vil en økning av datasettet gi mer presise resultater, men tidsbruken vil også naturligvis øke. [15]

Fargen på aksekorset endres basert på den gitte vinkelen: den blir rød hvis vinkelen er utenfor grensen, og grønn hvis den er innenfor. På denne måten kan brukeren, basert på denne tilbakemeldingen, orientere telefonen for å justere posisjonen korrekt i forhold til målskiven. I tillegg er det to tekstbokser til stede på skjermen som viser live feed av vinkel dataen som et ekstra hjelpemiddel



Figur 98: Kamera skjerm

19.1.6 Galleri - Skjerm

Galleriskjermen er stedet der brukeren kan se bildene som lagres fra bildeopptaksprosessen. Ved å trykke på en knapp får brukeren tilgang til mappen med bildene. Her benyttes biblioteket expo-image-picker. Dette tillater tilgang til telefonens medielagring, men siden det kun jobbes med bilder, er det begrenset til at kun bilder vises.

20 Regnskap

Her er tabellen over alle utgiftene knyttet til arbeidet med bacheloroppgaven. Vi ble ikke tildelt noe budsjett, men kvitteringer for utgiftene ble sendt underveis til arbeidsgiveren KTS, etter enighet med dem. 'Bevertninger' vil si alle utgifter knyttet til mat og drikke i forbindelse med forberedelsesmøtene til både første og andre presentasjon. 'Andre' inkluderer alt annet som har blitt brukt til hjelp under arbeidet med bacheloroppgaven.

Produkt	Dato	Pris (Per stykk)	Antall	Pris (total)
Bevertning				
Kaffekanne - Stor	2/17/23	115kr	1	115
Druer røde 500g beger	2/17/23	29.90kr	1	30
6 Doughnuts	2/17/23	25kr	6	150
6 Donuts	3/21/23	25kr	6	150
2 Muffins sjokolade	3/21/23	42kr	2	84
2 Muffins Blåbær/Vanilje	3/21/23	42kr	2	84
Kaffekanne - Stor	3/21/23	115kr	1	115
Total pris				728
Andre				
Brevordner Esselte No1 A4 (Grønn perm)	1/7/23	69,90kr	1	70
Akrylspray	4/25/23	149,90kr	2	300
Akrylspray	5/5/23	149.90kr	1	150
Kopiering	5/5/23	195kr	1	195
Vevteip basic mattsvart 25	5/13/23	99.90kr	1	100
Kopiering	5/16/23	180kr	1	180
Total pris				995
Total pris alt				1723

21 Konklusjon

Gjennom denne bacheloroppgaven har vi undersøkt og analysert kalibrering av treffpunkter på en skyteskive ved hjelp av bildebehandling. Ved å forske og teste har vi fått en dypere forståelse av tematikken.

Vi har funnet ut at det er mulig å få den nøyaktigheten vi ønsker, men at den matematiske sammenhengen må utforskes dypere, for det er helt klart at det er en sammenheng med hensyn på vinkel.

Det er viktig å nevne at vi har utforsket dette innenfor visse begrensinser, sånn som tidsbegrensninger som en bacheloroppgave gir. Ved videre forskning tror vi at det er mulig å få en enda bedre nøyaktig enn det vi har kommet frem til i denne rapporten.

Referanser

[1] M. Aspelund, "Hva er api og api-integrasjon?" (Hentet 20.05.2023). [Online]. Available: https://www.visma.no/blogg/hva-er-api-sporsmal-og-svar/#Content1

[2] A. Rolstadås, "Prosjektmodell," *Store norske leksikon*, Hentet 23.03.2023. [Online]. Available: https://snl.no/prosjektmodell

[3] "What is scrum?" (Hentet 03.02.2023). [Online]. Available: https://www.scrum.org/learning-series/what-is-scrum

[4] A. Athuraliya, "How to make a user flow diagram," *cocreately*, (Hentet 26.04.2023). [Online]. Available: https://creately.com/guides/user-flow-diagram/

[5] D. J.Eck, "Prosjektmodell," *Introduction to Computer Graphics*, no. 3, 2021. [Online]. Available: https://math.hws.edu/graphicsbook/c4/s1.html

[6] T. Holtsmark, "Hvitt," *Store norske leksikon*, (Hentet 17.05.2023). [Online]. Available: https://snl.no/hvitt

[7] "What does low iso mean?" *Adobe*, (Hentet 17.03.2023). [Online]. Available: https://www.adobe.com/creativecloud/photography/hub/guides/when-to-use-low-iso-settings#:~:text=ISO20is20your20camera's20sensitivity,higher20ISO20means20more20sensitivity.

[8] "Exposure triangle – everything you need to know," *Store norske leksikon*, (Hentet 17.03.2023). [Online]. Available: https://www.nfi.edu/exposure-triangle/#:~:text=The20Exposure20Triangle20comprises20the,(film20or20digital20ISO).

[9] C. Deziel, "Which colors reflect more light?" *Sciencing.com*, (Hentet 12.04.2023). [Online]. Available: https://sciencing.com/colors-reflect-light-8398645.html

[10] T. Tag, "Reflectors by thomas tag," (Hentet 15.3.2023). [Online]. Available: https://uslhs.org/reflectors

[11] S. Sharma, "What is accelorometer? how to use accelerometer in mobile devices?" *Credencys Data Management Company*, (Hentet 04.05.2023). [Online]. Available: https://www.credencys.com/blog/accelerometer/

[12] D. Apple, "Understanding reference frames and device attitude," (Hentet 08.05.2023). [Online]. Available: https://developer.apple.com/documentation/coremotion/getting_processed_device-motion_data/understanding_reference_frames_and_device_attitude

[13] wikiwand, "Euler angles," (Hentet 16.05.2023). [Online]. Available: https://www.wikiwand.com/en/Euler_angles#/Rotation_matrix

[14] V. H. Adams, "Complementary filters," *vanhunteradams*, (Hentet 12.05.2023). [Online]. Available: https://vanhunteradams.com/Pico/ReactionWheel/Complementary_Filters.html

[15] G. J. Prinsloo, "How can i reduce noise from accelerometer and gyroscope values of the nao robot for classification?" *Research Gate*, (Hentet 12.05.2023). [Online]. Available: https://www.researchgate.net/post/How_can_I_reduce_noise_from_accelerometer_and_gyroscope_values_of_the_nao_robot_for_classification

A Avviksdata

Fysiske målinger

X	Y	
1.50	181.00	1
-2.50	121.00	2
116.70	120.50	3
-121.70	118.30	4
57.70	63.30	5
-57.30	60.70	6
-3.30	60.60	7
-179.60	0.90	8
-119.16	0.12	9
118.08	-1.16	10
-59.50	-0.98	11
179.04	-0.22	12
56.70	-1.16	15
-2.10	-56.50	16
57.70	-57.50	17
-58.70	-59.00	18
-2.00	-118.10	19
119.10	-117.60	20
-122.70	-120.20	21
-3.00	-178.50	22

Path 1

	1				
X	Y		DeltaX	DeltaY	
0.92	181.43	1	0.58	-0.43	
-1.85	121.93	2	-0.65	-0.93	
116.16	122.24	3	0.54	-1.74	
-121.39	119.15	4	-0.31	-0.85	
57.77	64.89	5	-0.07	-1.59	
-57.30	61.81	6	0.00	-1.11	
-1.85	62.12	7	-1.45	-1.52	
-177.15	2.31	8	-2.45	-1.41	
-117.69	1.70	9	-1.47	-1.58	
117.39	1.39	10	0.69	-2.55	
-59.15	0.77	11	-0.35	-1.75	
177.93	1.70	12	1.11	-1.92	
56.85	0.46	15	-0.15	-1.62	
-0.62	-55.65	177 16	-1.48	-0.85	
56.85	-56.26	17	0.85	-1.24	
-58.23	-57.50	18	-0.47	-1.50	
-0.62	-116.38	19	-1.38	-1.72	

Path 2

	2			
X	Y		DeltaX	DeltaY
1.39	181.86	1	0.11	-0.86
-1.39	122.16	2	-1.11	-1.16
116.54	121.86	3	0.16	-1.36
-120.69	119.38	4	-1.01	-1.08
57.73	64.64	5	-0.03	-1.34
-56.88	61.86	6	-0.42	-1.16
-1.70	61.86	7	-1.60	-1.26
-177.11	2.47	8	-2.49	-1.57
-117.61	1.86	9	-1.55	-1.74
117.46	0.93	10	0.62	-2.09
-59.04	0.93	11	-0.46	-1.91
177.81	1.24	12	1.23	-1.46
56.50	0.00	15	0.20	-1.16
-1.08	-55.91	16	-1.02	-0.59
56.81	-56.22	17	0.89	-1.28
-58.11	-57.45	18	-0.59	-1.55
-1.08	-116.45	19	-0.92	-1.65
118.69	-116.45	20	0.41	-1.15
-122.54	-119.23	21	-0.16	-0.97
-2.62	-177.30	22	-0.38	-1.20

Path 3

	3			
X	Y		DeltaX	DeltaY
0.00	182.09	1	1.50	-1.09
-2.56	122.25	2	0.06	-1.25
115.99	122.57	3	0.71	-2.07
-122.24	118.71	4	0.54	-0.41
57.19	64.98	5	0.51	-1.68
-58.24	61.45	6	0.94	-0.75
-2.88	61.77	7	-0.42	-1.17
-178.24	1.28	8	-1.36	-0.38
-118.72	0.64	9	-0.44	-0.52
116.95	1.28	10	1.13	-2.44
-60.16	0.32	11	0.66	-1.30
177.35	2.25	12	1.69	-2.47
55.91	0.00	15	0.79	-1.16
-1.60	-55.85	16	-0.50	-0.65
56.23	-56.17	17	1.47	-1.33
-59.20	-58.09	18	0.50	-0.91
-1.60	-116.17	19	-0.40	-1.93

-123.20	-119.68	21	0.50	-0.52
-3.20	-176.81	22	0.20	-1.69

Path4

	4			
X	Y		DeltaX	DeltaY
0.74	181.52	1	0.76	-0.52
-1.92	121.86	2	-0.58	-0.86
116.36	121.26	3	0.34	-0.76
-121.55	119.48	4	-0.15	-1.18
57.21	63.97	5	0.49	-0.67
-57.75	61.30	6	0.45	-0.60
-2.22	61.30	7	-1.08	-0.70
-178.27	1.93	8	-1.33	-1.03
-118.60	1.34	9	-0.56	-1.22
117.25	0.15	10	0.83	-1.31
-59.82	0.45	11	0.32	-1.43
177.58	0.45	12	1.46	-0.67
56.02	-0.44	15	0.68	-0.72
-1.63	-56.06	16	-0.47	-0.44
56.02	-56.65	17	1.68	-0.85
-59.23	-58.12	18	0.53	-0.88
-2.22	-116.39	19	0.22	-1.71
118.14	-116.39	20	0.96	-1.21
-123.32	-119.34	21	0.62	-0.86
-3.40	-177.02	22	0.40	-1.48

Path 5

	5			
X	Y		DeltaX	DeltaY
1.21	181.59	1	-0.29	-0.59
-1.81	121.67	2	-0.04	-0.67
116.36	121.06	3	-0.20	-0.56
-121.36	119.55	4	-0.03	-1.25
57.57	63.86	5	0.20	-0.56
-57.82	61.44	6	0.52	-0.74
-2.41	61.44	7	0.57	-0.84
-178.57	2.12	8	1.42	-1.22
-118.65	1.21	9	0.96	-1.09
116.97	-0.30	10	0.42	-0.86
-59.93	0.31	11	0.78	-1.29
177.27	0.00	12	0.65	-0.22
55.75	-0.89	15	1.10	-0.27
-2.11	-55.96	16	1.50	-0.54
55.75	-57.15	17	1.10	-0.35

-2.72	-116.71	19	2.10	-1.39
117.57	-116.71	20	1.35	-0.89
-123.77	-119.10	21	1.76	-1.10
-4.22	-176.86	22	2.37	-1.64

Iphone11

Path 12

6

X	Y		DeltaX	DeltayY
1.56	181.01	1	-0.06	-0.01
-1.54	120.77	2	-0.96	0.23
117.66	120.77	3	-0.96	-0.27
-121.07	118.29	4	-0.63	0.01
58.98	64.26	5	-1.28	-0.96
-56.71	60.54	6	-0.59	0.16
-1.54	61.16	7	-1.76	-0.56
-176.25	2.17	8	-3.35	-1.27
-116.78	0.92	9	-2.38	-0.80
118.28	1.54	10	-0.20	-2.70
-58.81	-1.27	11	-0.69	0.29
178.20	2.17	12	0.84	-2.39
56.68	1.21	15	0.02	-2.37
-0.04	-56.56	16	-2.06	0.06
57.50	-55.72	17	0.20	-1.78
-57.17	-59.91	18	-1.53	0.91
-1.54	-117.30	19	-0.46	-0.80
118.28	-115.40	20	0.82	-2.20
-121.07	-121.12	21	-1.63	0.92
-2.76	-177.70	22	-0.24	-0.80

Path 13

7

X	Y		DeltaX	DeltaY
-0.03	180.69	1	0.95	0.31
-2.49	120.76	2	0.64	0.24
116.25	122.00	3	-0.09	-1.50
-121.84	118.15	4	0.45	0.15
58.42	64.54	5	-0.65	-1.24
-57.11	60.21	6	-0.19	0.49
-1.85	61.54	7	0.00	-0.94
-176.79	-0.97	8	-0.36	1.87
-117.25	0.29	9	-0.44	-0.17
118.74	3.38	10	-1.35	-4.54
-58.34	0.29	11	-0.82	-1.27
179.06	3.99	12	-1.13	-4.21
57.18	0.90	15	-0.33	-2.06

57.80	-55.50	17	-0.95	-2.00
-57.11	-58.67	18	-1.12	-0.33
-0.03	-117.00	19	-0.58	-1.10
119.36	-114.46	20	-0.44	-3.14
-119.71	-121.44	21	-2.29	1.24
-0.65	-177.23	22	-1.20	-1.27

Path 14

8

X	Y		DeltaX	DeltaY
1.23	180.92	1	0.27	0.08
-1.85	120.61	2	-0.65	0.39
117.52	121.23	3	-0.82	-0.73
-121.84	118.15	4	0.14	0.15
58.76	63.38	5	-1.06	-0.08
-56.95	60.92	6	-0.35	-0.22
-1.85	61.54	7	-1.45	-0.94
-176.94	1.84	8	-2.66	-0.94
-117.56	1.23	9	-1.60	-1.11
118.14	1.84	10	-0.06	-3.00
-58.78	0.61	11	-0.72	-1.59
178.14	1.84	12	0.90	-2.06
56.28	0.61	15	0.42	-1.77
-1.23	-56.36	16	-0.87	-0.14
56.28	-56.36	17	1.42	-1.14
-58.17	-58.89	18	-0.53	-0.11
-1.23	-117.79	19	-0.77	-0.31
118.14	-115.89	20	0.96	-1.71
-121.23	-120.95	21	-1.47	0.75
-2.46	-177.94	22	-0.54	-0.56

Samsung s21

Path 17

9

X	Y		DeltaX	DeltaY
1.86	180.00	1	-0.36	1.00
-1.10	120.73	2	-1.40	0.27
117.59	120.00	3	-0.89	0.50
-121.29	117.81	4	-0.41	0.49
58.25	62.20	5	-0.55	1.10
-57.14	60.00	6	-0.16	0.70
-1.83	60.74	7	-1.47	-0.14
-177.32	2.93	8	-2.28	-2.03
-127.18	2.20	181 9	8.02	-2.08
116.85	0.00	10	1.23	-1.16
-59.35	-0.74	11	-0.15	-0.24

56.02	-1.49	15	0.68	0.33
-1.83	-57.39	16	-0.27	0.89
56.02	-57.39	17	1.68	-0.11
-59.35	-58.13	18	0.65	-0.87
-4.05	-117.76	19	2.05	-0.34
116.11	-117.76	20	2.99	0.16
-122.76	-120.00	21	0.06	-0.20
-4.05	-178.14	22	1.05	-0.36

Path 18

10

X	Y		DeltaX	DeltaY
-0.01	180.17	1	1.51	0.83
-2.13	120.35	2	-0.37	0.65
116.45	121.05	3	0.25	-0.55
-122.12	117.53	4	0.42	0.77
57.51	63.34	5	0.19	-0.04
-57.89	59.12	6	0.59	1.58
-2.13	60.52	7	-1.17	0.08
-177.18	0.00	8	-2.42	0.90
-127.06	0.00	9	7.90	0.12
117.16	0.70	10	0.92	-1.86
-59.30	-1.43	11	-0.20	0.45
177.51	1.40	12	1.53	-1.62
56.80	-0.72	15	-0.10	-0.44
-1.42	-56.98	16	-0.68	0.48
56.80	-56.98	17	0.90	-0.52
-57.89	-59.83	18	-0.81	0.83
-1.42	-118.22	19	-0.58	0.12
118.58	-116.80	20	0.52	-0.80
-122.83	-121.78	21	0.13	1.58
-2.13	-178.75	22	-0.87	0.25

PATH 19

11

X	Y		DeltaX	DeltaY
-0.73	180.00	1	2.23	1.00
-2.89	120.00	2	0.39	1.00
115.65	121.43	3	1.05	-0.93
-122.88	116.43	4	1.18	1.87
57.10	62.85	5	0.60	0.45
-58.93	60.71	6	1.63	-0.01
-2.89	60.00	7	-0.41	0.60
-177.49	-0.01	8	-2.11	0.91
-127.19	-0.01	9	8.03	0.13

-59.65	-0.73	11	0.15	-0.25
177.10	2.14	12	1.94	-2.36
56.37	-0.73	15	0.33	-0.43
-0.73	-57.29	16	-1.37	0.79
57.10	-56.56	17	0.60	-0.94
-58.22	-59.46	18	-0.48	0.46
-1.45	-118.19	19	-0.55	0.09
118.55	-116.02	20	0.55	-1.58
-121.45	-121.82	21	-1.25	1.62
-1.45	-179.10	22	-1.55	0.60

Ute sollys
PATH22

	11			
X	Y		DeltaX	DeltaY
1.45	182.89	1	-0.53	-1.46
-0.72	124.33	2	-1.13	-2.41
116.74	123.61	3	-0.58	-1.38
-119.28	121.44	4	-2.11	-2.29
57.30	65.70	5	0.47	-0.81
-56.05	62.89	6	-1.26	-1.08
-1.43	63.61	7	-0.41	-1.49
-176.05	3.61	8	-1.10	-1.30
-126.47	2.89	9	8.78	-1.19
117.46	3.61	10	-0.08	-2.22
-58.92	1.44	11	-0.23	-0.67
56.56	2.89	15	0.29	-2.42
-0.72	-54.62	16	0.10	-1.03
56.56	-53.90	17	0.29	-2.36
-57.48	-56.05	18	-0.75	-1.44
-1.43	-114.25	19	0.82	-2.12
118.19	-113.54	20	0.74	-2.22
-121.44	-119.28	21	-0.57	-0.18
-2.15	-175.33	22	0.31	-1.78

Ute sollys Iphone
PATH25

	11			
X	Y		DeltaX	DeltaY
-3.52	177.51	1	5.02	3.49
-4.22	119.29	2	1.72	1.71
113.99	121.42	3	2.71	-0.92
-122.88	116.43	4	1.18	1.87
56.65	62.48	5	1.05	0.82
-58.58	57.51	6	1.28	3.19

-177.88	-3.54	8	-1.72	4.44
-127.05	-2.83	9	7.89	2.95
117.10	1.42	10	0.98	-2.58
-59.65	-0.73	11	0.15	-0.25
176.99	1.41	12	2.05	-1.63
56.65	-1.42	15	0.05	0.26
0.01	-58.59	16	-2.11	2.09
57.35	-57.89	17	0.35	0.39
-58.22	-59.46	18	-0.48	0.46
0.01	-118.59	19	-2.01	0.49
120.36	-117.18	20	-1.26	-0.42
-121.45	-121.82	21	-1.25	1.62
-1.45	-179.10	22	-1.55	0.60

B Risikoanalyse

Risiko	Forklaring	S	A	Vurdering	Begrunnelse	Forebyggende	Tiltak
Feil i software	Dette er hvis software man lager har store feil slik at prosjektet blir forsinket	3	3	Høy 9	Feil i software kan by på store problemer og det kan forsinke progressen vi har. Siden vi er "uerfarne ingeniører" så ser vi på det som sannsynlig at det vi skje. Det er vanskelig å si noe om konsekvensen det vil ha, men store feil kan forsinke oss i en stor grad.	Gå over og teste kode oftere	Skulle det oppstå er det viktig å gå igjennom og ta en vurdering på hvor ille det er. Er det store feil må man vurdere om man skal fikse det eller gå videre.

Figur 99: Feil i software

Risiko	Forklaring	S	A	Vurdering	Begrunnelse	Forebyggende	Tiltak
Forsinkelser av deler	Dette er hvis noen deler man bestiller eller lager, men tar lenger tid enn forventet	2	3	Høy 6	Gruppen ser på det som moderat sannsynlighet at deler og slikt kan bli forsinket og alvorlig hvis det skulle skje.	Bestille deler så fort man har klart hva man trenger.	Vurdere om vi kan få tak i delene på en annen måte ved å gjøre det med et annet materiale for eksempel.

Figur 100: Forsinkelser av deler

Risiko	Forklaring	S	A	Vurdering	Begrunnelse	Forebyggende	Tiltak
Gruppemedlem blir borte	Risiko for at et gruppemedlem slutter underveis i prosjektet	1	4	Medium 4	Det er liten sannsynlig at noen i gruppen slutter fordi alle er like motiverte til å gjøre best mulig jobb, men skulle det skje får det store problemer for de gjenværende medlemmene	Pushe hverandre til å ville gjøre en god jobb og la alle føle seg som en like viktig del av gruppen. Slik hindrer vi at noen ønsker å slutte	Akkurat som ved sykdom, så kan gruppen minske konsekvensen av dette ved å ha en åpen dialog slik at alle vet hva de andre medlemmene jobber med.

Figur 101: Gruppemedlem blir borte

Risiko	Forklaring	S	A	Vurdering	Begrunnelse	Tiltak
Korttidssykdom	Dette er hvis noen blir syke en dag eller to, eller over en kort periode.	3	1	Medium 3	Dette er en medium risiko fordi sannsynligheten er høy for at medlemmer blir syke noen dager, i hvert fall i vinter månedene, men graden er ubetydelig fordi gruppen ser ikke på det som er stort problem at noen blir kortvarig syk.	Hvis noen skulle bli syk så vil det være mulighet for å jobbe hjemmefra hvis det er mulig. Er man så syk at det blir vanskelig så kan man eventuelt ta det igjen eller at andre medlemmer hjelper til.

Figur 102: Korttidssykdom

Risiko	Forklaring	S	A	Vurdering	Begrunnelse	Forebyggende	Tiltak
Langtidssykdom	Risikoen for at et gruppemedlem blir syk over en lengre periode	2	3	Høy 6	Denne er satt til risiko høy fordi konsekvensen av at handelsen inntreffer er alvorlig og sannsynligheten for at det skjer er moderat. Vi er en generell frisk gruppe, så langtidssykdommer ser vi ikke på noe risiko skal inntreffe.	At medlemmene er oppdaterte på hva de andre driver med til enhver tid. Da vil det bli lettere å overta oppgavene hvis noen skulle bli borte	Skulle langtidssykdom skje er det viktig at arbeidsoppgavene til den person blir fordelt rundt til de andre og man lager en plan på hvordan man skal fortsette arbeidet med en mindre.

Figur 103:

Risiko	Forklaring	S	A	Vurdering	Begrunnelse	Forebyggende
Ødeleggende	Risiko forbundet med om noe ødelegges eller at systemet er ødeleggende fordi det ikke er slik det skal	3	3	Høy 9	Risikoen er satt til høy fordi gruppen ser på det som sannsynlig og alvorlig hvis deler av systemet vårt ikke er riktig, slik at brukeren får ut feil verdier og dermed viser feil videre.	Være ekstra oppmerksom på ting som er kritiske for systemet. Å få ut feil verdier kan være veldig ødeleggende

Figur 104: Ødeleggende

Risiko	Forklaring	S	A	Vurdering	Forebyggende	Løsning
Oppdragsgiver går konkurs	Risikoen for at Kongsberg Target Systems går konkurs	1	3	Medium 3	Risikoen for at oppdragsgiver går konkurs er satt til medium fordi at gruppen ser på det som lite sannsynlig at det skal skje, men at konsekvensen er alvorlig skulle det skje.	Selv om oppdragsgiver går konkurs underveis kan man muligens fortsette med oppgaven fordi man vet allerede hva man skal gjøre.

Figur 105: Oppdragsgiver går konkurs

Risiko	Forklaring	S	A	Vurdering	Begrunnelse	Forebyggende	Tiltak
Problemer innad i gruppen	Problemer som måtte oppstå innad i gruppen. Dette kan være store uenigheter, større uenigheter eller at vi blir uvenner og ikke klarer å jobbe sammen.	2	3	Høy 6	Vi er 6 forskjellige individer som har sine meninger og som ikke er redd for å si dem. Dette kan bli et problem, men samtidig skal vi nok løse de problemene som oppstår på en ordentlig måte.	Ha ærlighet i gruppen slik at alle føler seg som en del av gruppa slik at alle tørr å si det de mener. Viktig at vi også tar opp ting med en gang man føler på noe slik at ikke irritasjon og problemer bygger seg opp over tid.	Hvis problemer oppstår er det viktig at vi tar det opp og finner den beste løsningen slik at arbeidet kan fortsette.

Figur 106: Problemer innad i gruppen

Risiko	Forklaring	S	A	Vurdering	Begrunnelse	Forebyggende	Tiltak
Skader	Dette er risikoer forbundet med om brukere kan skade seg når man bruker systemet eller at systemet i seg selv er skadende	1	1	Lav 1	Gruppen ser på det som lav risiko at noen kan bli skadet når man bruker systemet vårt fordi dette i hovedsak er et digitalt system som man bruker datamaskin til.	Sette seg inn i brukeren sitt perspektiv på hvordan systemet vil bli brukt og hva slags verktøy man eventuelt må bruke	Endre på det som er skadene. Dette kan være å gjøre store endringer eller mindre slik at sannsynligheten for at noen skader seg skal inntreffe.

Figur 107: Skader

C Krav

C.1 A-krav

Status	I/S
Krav identifikasjon	FK-A1
Kravbeskrivelse	Programvare skal ved hjelp av bildebehandling detektere treffpunkter på en skyteskive.
Prioritet	A

Status	I/S
Krav identifikasjon	FK-A2
Kravbeskrivelse	De elektroniske beregningene av treffpunkter fra KTS skal kunne skrives inn i programvaren manuelt.
Prioritet	A

Status	I/S
Krav identifikasjon	FK-A3
Kravbeskrivelse	Programvare skal sammenligne elektronisk beregnede treffpunkter med treffpunkter bestemt av bildebehandling.
Prioritet	A

Status	I/S
Krav identifikasjon	FK-A4
Kravbeskrivelse	Programvaren skal presentere resultatene av sammenligningen til brukeren.
Prioritet	A

Status	I/S
Krav identifikasjon	FK-A5
Kravbeskrivelse	Forholdet mellom oppløsning og den teoretiske nøyaktigheten på treffpunktene på skyteskiven skal undersøkes.
Prioritet	A

Status	I/S
Krav identifikasjon	FK-A6
Kravbeskrivelse	En brukerinstruks for systemet skal lages.
Prioritet	A

Status	I/S
Krav identifikasjon	FK-A7
Kravbeskrivelse	Systemet skal ikke være forstyrrende for skytter.
Prioritet	A

Status	I/S
Krav identifikasjon	FK-A8
Kravbeskrivelse	Treffpunkter skal detekteres med en nøyaktighet på 2 mm.
Prioritet	A

Status	I/S
Krav identifikasjon	FK-A9
Kravbeskrivelse	Programvaren skal presentere resultater av sammenligningen til brukeren innen 60 sekunder.
Prioritet	A

Status	I/S
Krav identifikasjon	FK-A10
Kravbeskrivelse	Detektere treffpunktene med nøyaktighet (FK-A8) ut ifra bilde som er tatt 90 +/- 5 grader i forhold til flaten på skyteskive.
Prioritet	A

Status	I/S
Krav identifikasjon	FK-A11
Kravbeskrivelse	Det fysiske systemet skal kunne monteres innen 5 minutter.
Prioritet	A

Status	I/S
Krav identifikasjon	FK-A12
Kravbeskrivelse	Systemet skal kunne brukes utendørs i varierende værforhold.
Prioritet	A

C.2 B-krav

Status	I/S
Krav identifikasjon	FK-B1
Kravbeskrivelse	Data fra kalibreringsprosessen skal lagres lokalt. Data defineres som all informasjonen knyttet til: • Treffpunkter hentet fra akustisk behandling. • Treffpunkter hentet fra bildebehandling. • Sammenligningen av akustisk beregnede treffpunkter og treffpunkter funnet med bildebehandling. • Lagre originalbilde
Prioritet	B

Status	I/S
Krav identifikasjon	FK-B2
Kravbeskrivelse	Treffpunkter skal detekteres med en nøyaktighet på under en 1 mm.
Prioritet	B

Status	I/S
Krav identifikasjon	FK-B3
Kravbeskrivelse	Maksimal vekt på systemet skal være 15 kg
Prioritet	B

Status	I/S
Krav identifikasjon	FK-B4
Kravbeskrivelse	Programvaren skal presentere resultater av sammenligningen til brukeren innen 10 sekunder.
Prioritet	B

Status	I/S
Krav identifikasjon	FK-B5
Kravbeskrivelse	Det fysiske systemet skal kunne monteres innen 2.5 minutter.
Prioritet	B

C.3 C-krav

Status	I/S
Krav identifikasjon	FK-C1
Kravbeskrivelse	Systemet skal automatisk ta hensyn til kameravinkelen ved deteksjon av treffpunkt.
Prioritet	C

Status	I/S
Krav identifikasjon	FK-C2
Kravbeskrivelse	I programvaren skal treffpunkter fra bildebehandling sammenlignes med treffpunkter fra akustisk behandling automatisk.
Prioritet	C

Status	I/S
Krav identifikasjon	FK-C3
Kravbeskrivelse	Programvaren skal automatisk skrive kalibreringsdata fra bildebehandlingen tilbake til den elektroniske skyteskiva.
Prioritet	C

Status	I/S
Krav identifikasjon	FK-C4
Kravbeskrivelse	Alle data fra kalibreringsprosessen skal lastes opp og synkronisere fra lokale databaser til en skytjeneste.
Prioritet	C

Status	I/S
Krav identifikasjon	FK-C5
Kravbeskrivelse	Treffpunkter skal detekteres med en nøyaktighet på under 0,5 mm.
Prioritet	C

Status	I/S
Krav identifikasjon	FK-C6
Kravbeskrivelse	Programvaren skal presentere resultater av sammenligningen til brukeren innen 5 sekunder.
Prioritet	C

Status	I/S
Krav identifikasjon	FK-C7
Kravbeskrivelse	Det fysiske systemet skal kunne monteres innen 1 minutt.
Prioritet	C

D Testplan

Status	
Test ID: FK-AT1	Dato 08.05.23
Krav ID	FK-A1
Testmetode	Funksjonstest og enhetstest
Beskrivelse	1. Sett opp kamera 2. Skann/ta bilde. 3. Mat bildet til pc-en 4. Trykk på «finn treffpunkter». 5. Sjekk at du har fått opp x og y verdi.
Testmiljø	1. Ute i sola 2. I regnet 3. I skyggen ute
Verifikasjonsmetode	Observasjon
Utstyr	Pc og kamera
Resultat	Godkjent av Ole-markus
Akseptansekriterium	X og y koordinater kommer ut.
Utført av	Samuel

Status	
Test ID: FK-AT2 FK-AT3 FK-AT4	Dato
Krav ID	FK-A2 FK-A3 FK-A4
Testmetode	Funksjonstest, enhetstest og automatisert test
Beskrivelse	1. Skriv inn x og y koordinater manuelt inn i program-met. 2. Sjekk at koordinatene blir akseptert. 3. Sjekk at koordinatene vises tilbake. 4. Sjekk at koordinatene brukes i algoritmen og at det blir sammenlignet 5. Se at resultatet til slutt blir vist tilbake til bruker
Testmiljø	Ikke relevant
Verifikasjonsmetode	Måling og Observasjon
Utstyr	Pc
Resultat	
Akseptansekriterium	Alle X og y koordinater skal kunne skrives inn, sammenlignes og presenteres
Utført av	

Status	
Test ID:FK-AT5	Dato:08.05.23
Krav ID	FK-A5
Testmetode	Funksjonell test
Beskrivelse	1. Undersøk med flere kilder om bilde oppløsning og teorier om nærmeste teoretisk nøyaktighet i bildehandling. 2. Sjekk om distansen mellom skyteskiven og billedtagnings enheten kan påvirke oppløsningen og i hvilken grad.
Testmiljø	Ikke relevant
Verifikasjonsmetode	Måling og Observasjon
Utstyr	Pc
Resultat	Godkjent av Kristoffer
Akseptansekriterium	Sjekket hvertfall 3 kilder for nøyaktighet av bilde behandling. -Sjekket hvor mye distanse kan påvirke bildeoppløsning. -Kommet fram til en beslutning basert på disse kildene.
Utført av	Samuel

Status	
Test ID:FK-AT6	Dato:09.05.23
Krav ID	FK-A6
Testmetode	Bruker test
Beskrivelse	En brukerinstruks for systemet skal lages. 1. La 3 personer sette opp systemet med brukerinstruksen. 2. Undersøk hvor bra de klarer det.
Testmiljø	Ute i sollys og ute i regnvær.
Verifikasjonsmetode	Observasjon
Utstyr	Fysiske elementer som skal settes opp.
Resultat	Godkjent av Jehad
Akseptansekriterium	3 forskjellige personer kan sette opp systemet uten å ha kjennskap til systemet -Kommet fram til en beslutning basert på disse kildene.
Utført av	Ole-Markus

Status	
Test ID: FK-A7	Dato
Krav ID	FK-A7
Testmetode	Bruker test og samsvarstest
Beskrivelse	1. Lag en spørreundersøkelse med 3 spørsmål for hvordan opplevelsen er for skytter. Hvor svaret er fra 1-10. 1 er ikke forstyrrende i det heletatt og 10 er veldig forstyrrende. 2. Sett opp skyteskiven 3. 3. La 3 personer se på skyteskiven og la dem bedømme om hvorvidt systemet er forstyrrende via spørreundersøkelsen.
Testmiljø	Ute i sollys, ute i skyggen og inne.
Verifikasjonsmetode	Observasjon og utregning
Utstyr	Pen og papir.
Resultat	Ikke godkjents
Akseptansekriterium	Gjennomsnitts svar på spørreundersøkelsen må være under 5
Utført av	

Status	
Test ID:FK-AT8 FK-AT9 FK-AT10	Dato
Krav ID	FK-A8, FK-A9, FK-A10
Testmetode	Funksjonell test og enhets test.
Beskrivelse	1. Sett opp kamera i 90 +/- 5 grader i forhold til flaten på skyteskive. 2. Sjekk grader med gradskive. 3. Skann/Ta bilde 4. Kjør programmet 5. Sjekk bildebehandlingsdata. 6. Mål manuelt med skyvelære 7. Start sammenligning, start stoppeklokke, stopp klokken når sammenligningen blir presentert. 8. Sjekk om forholdet mellom skuddene holder en nøyaktighet på 2 mm. 9. Stopp stoppeklokken når sammenligningen er ferdig og resultatet er presentert. 10. Gjør forsøket 10 ganger med forskjellige treffpunkter.(Dette punktet må også fullføres for rødt krav).
Testmiljø	Ute i sollys, ute i regnet, ute i skyggen og inne.
Verifikasjonsmetode	Observasjon og måling
Utstyr	Skyvelære, pc, stoppeklokke
Resultat	
Akseptansekriterium	Treffpunktene har nøyaktighet 2 mm og tiden den bruker på å presentere sammenligningen er under 60 sekunder.
Utført av	

Status	
Test ID:FK-AT11	Dato
Krav ID	FK-A11
Testmetode	Bruker test
Beskrivelse	1. 3 personer skal montere systemet 2. Ta tiden de bruker
Testmiljø	Ikke relevant
Verifikasjonsmetode	Måling,observasjon,utregning
Utstyr	Fysiske systemet, brukermanual og stoppeklokke
Resultat	Ikke godkjent
Akseptansekriterium	Gjennomsnittstiden alle 3 personer bruker på å montere systemet er under 5 minutter
Utført av	

Status	
Test ID:FK-BT1	Dato
Krav ID	FK-B1
Testmetode	Funksjonstest og enhetstest
Beskrivelse	1. Ta bilde/skann skyteskiven 2. Kjør algoritmen. 3. Sjekk lokale databasen om treffpunkter fra akustiske skyteskiven er lagret. 4. Deretter sjekk at data hentet fra bildebehandling er lagret 5. Så sjekk at data sammenligningen av akustiske skyteskiven og bildebehandlingen er lagret. 6. Så sjekk at originalt bilde er lagret.
Testmiljø	Ikke relevant
Verifikasjonsmetode	Observasjon
Utstyr	Måleskive, elektronisk enhet, kamera.
Resultat	Ikke utført
Akseptansekriterium	Data som treffpunkter fra den akustiske skyteskiven. treffpunkter fra bildebehandlingen, sammenligningen av treffpunkter og bilde av treffpunkter på skyteskiven skal lagres.
Utført av	

Status	
Test ID:FK-BT2	Dato
Krav ID	FK-B2
Testmetode	unksjonell test, automatisk test og enhetstest.
Beskrivelse	1. Skann/Ta bilde 2. Kjør programmet 3. Sjekk bildebehandlingsdata. 4. Mål manuelt med skyvelære 5. Sammenlign og sjekk om forholdet mellom skuddene holder en nøyaktighet på 1 mm. 6. Gjør forsøket 10 ganger med forskjellige treffpunkter som er 10 mm fra hverandre og 4 mm nærme hverandre.
Testmiljø	Ute i sollys, ute i regnet, ute i skyggen og inne.
Verifikasjonsmetode	Måling og observasjon
Utstyr	Skyvelære, pc og kamera.
Resultat	ikke godkjent
Akseptansekriterium	Deteksjonen holder seg innenfor 1 mm for hvert forsøk.
Utført av	

Status	
Test ID:FK-BT3	Dato:11.08.23
Krav ID	FK-B3
Testmetode	Brukertest og funksjonell test
Beskrivelse	1.Vei opp systemet med en badevekt 2.Finn 3 personer som kan prøve å bære systemet det er viktig at de prøver å bære systemet hver for seg selv.
Testmiljø	Ikke relevant
Verifikasjonsmetode	Observasjon
Utstyr	Badevekt
Resultat	Godkjent av Samuel
Akseptansekriterium	De 3 personene er i stand til å bære systemet og vekten er innenfor 15 kg.
Utført av	Lars-Erik

Status	
Test ID:FK-BT4	Dato
Krav ID	FK-B4
Testmetode	Funksjonell test, automatisk test og enhets test
Beskrivelse	1. Sett opp bruktskyteskive hvor skyteskiven er skutt på tidligere. 2. Ta bilde 3. Kjør kalibrerings algoritmen samtidig som du starter stoppeklokken. 4. Stopp stoppeklokken når algoritmen er ferdig med å presentere dataen til bruker.
Testmiljø	Ikke relevant
Verifikasjonsmetode	Måling og observasjon
Utstyr	Stoppeklokke, pc og kamera.
Resultat	Ikke godkjent
Akseptansekriterium	Programvaren bruker under 10 sekunder på å presentere resultatet av sammenligningen.
Utført av	Jehad

Status	
Test ID:FK-BT5	Dato:11.05.23
Krav ID	FK-B5
Testmetode	Bruker test og Funksjonell test
Beskrivelse	1. Tre personer skal montere systemet 2. Ta tiden de bruker
Testmiljø	Ikke relevant
Verifikasjonsmetode	Måling og observasjon
Utstyr	Fysiske systemet, brukermanual og stoppeklokke
Resultat	Godkjent av Samuel
Akseptansekriterium	Gjennomsnittstiden alle 3 personer bruker på å montere systemet er under 2.5 minutter.
Utført av	Ole-markus

Status	
Test ID:FK-CT1	Dato
Krav ID	FK-C1
Testmetode	Funksjonstest og enhetstest
Beskrivelse	1. Skyt skudd på skyteskiven 2. Ta bilder fra forskjellige vinkler 3. Sjekk at algoritmen klarer å detektere treffpunkter.
Testmiljø	Ikke relevant
Verifikasjonsmetode	Måling og observasjon
Utstyr	Måleskive, elektronisk enhet, kamera.
Resultat	Ikke godkjent
Akseptansekriterium	Uavhengig av vinkel får i tilbake nøyaktige x og y koordinater.
Utført av	Samuel

Status	
Test ID:FK-CT2	Dato
Krav ID	FK-C2
Testmetode	Funksjonstest, enhetstest og kompatibilitetstest
Beskrivelse	1. Sjekk at programvaren får opp bildebehandling dataen og akustiske dataen 2. Kjør sammenligningsprosessen 3. Se at resultatet kommer ut
Testmiljø	Ikke relevant
Verifikasjonsmetode	Observasjon
Utstyr	Elektronisk enhet, kamera.
Resultat	Ikke utført
Akseptansekriterium	Programvaren klarer å sammenligne de to dataene automatisk.
Utført av	

Status	
Test ID:FK-CT3	Dato
Krav ID	FK-C3
Testmetode	Funksjonstest, enhetstest og kompatibilitetstest
Beskrivelse	1. Se om programvaren får ut kalibreringsdata fra bildebehandlingen 2. Send dataen til den elektroniske skyteskiven 3. Bekreft at den elektroniske skiven mottar den kalibrerte dataen.
Testmiljø	Ikke relevant
Verifikasjonsmetode	Observasjon
Utstyr	Måleskive, elektronisk enhet, kamera.
Resultat	Ikke godkjent
Akseptansekriterium	Den elektroniske skyteskiven mottar kalibrert data.
Utført av	

Status	
Test ID:FK-CT4	Dato
Krav ID	FK-C4
Testmetode	Funksjonstest og enhetstest
Beskrivelse	1. Test skann/ta bilde 2. Trykk på lagre 3. Sjekk at data er lagret lokalt 4. Sjekk at data er synkronisert med skytjeneste.
Testmiljø	Ikke relevant
Verifikasjonsmetode	Observasjon
Utstyr	Måleskive, elektronisk enhet, kamera.
Resultat	ikke godkjent
Akseptansekriterium	Dataen fra kalibreringsprosessen lastes opp fra lokale databaser og kan synkroniseres til en skytjeneste.
Utført av	

Status	
Test ID:FK-CT5	Dato
Krav ID	FK-C5
Testmetode	Funksjonell test, automatisk test og enhets test
Beskrivelse	1. Skann/Ta bilde 2. Kjør programmet 3. Sjekk bildebehandlingsdata. 4. Mål manuelt med skyvelære 5. Sammenlign og sjekk om forholdet mellom skuddene holder en nøyaktighet på 0.5 mm. Gjør forsøket 10 ganger med forskjellige treffpunkter som er 10 mm fra hverandre og 4 mm nærme hverandre
Testmiljø	Ute i sollys, ute i regnet, ute i skyggen og inne.
Verifikasjonsmetode	Måling
Utstyr	Skyvelære, pc og kamera.
Resultat	Ikke godkjent
Akseptansekriterium	Deteksjonen holder seg innenfor 0.5 mm for hvert forsøk.
Utført av	

Status	
Test ID:FK-CT6	Dato
Krav ID	FK-C6
Testmetode	Funksjonell test
Beskrivelse	1. Sett opp bruktskyteskive hvor skyteskiven er skutt på tidligere. 2. Ta bilde 3. Kjør kalibrerings algoritmen samtidig som du starter stoppeklokken. 4. Stopp stoppeklokken når algoritmen er ferdig med å presentere dataen til bruker.
Testmiljø	Ikke relevant
Verifikasjonsmetode	Måling og observasjon
Utstyr	Stoppeklokke, pc og kamera.
Resultat	Ikke godkjent
Akseptansekriterium	Programvaren bruker under 5 sekunder på å presentere resultatet av sammenligningen.
Utført av	

Status	
Test ID:FK-CT7	Dato
Krav ID	FK-C7
Testmetode	Funksjonell test og bruker test
Beskrivelse	1. 3 personer skal montere systemet 2. Ta tiden de bruker
Testmiljø	Ikke relevant
Verifikasjonsmetode	Måling og observasjon
Utstyr	Fysiske systemet, brukermanual og stoppeklokke
Resultat	Ikke godkjent
Akseptansekriterium	Gjennomsnittstiden alle 3 personer bruker på å montere systemet er under 1 minutt.
Utført av	Ikke godkjent

E Sprintrevisjon

Sprintrevisjonen bidrar til den iterative etosen til «scrum»-prosessen. En sprintrevisjon skal inneholde en følgende oversikt for hver sprint:

1. En vurdering hvorvidt hvilke oppgaver fra «backloggen» skal videreføres eller fjernes til neste sprint.
2. En beskrivelse av problemer knyttet til arbeidsprosessen ved sprinten, samt tiltak som kan prøves til neste sprint.

E.1 Pre-sprint

Dette dokumentet er strukturert som en logg for å dokumentere endringer i arbeidsprosessen. De seks første sprintene i fremdriftsplanen er utført uten bruk av revisjonsdokumentet. Disse sprintene har inneholdt arbeid knyttet til rammene prosjektet skal utføres i. Arbeidsmetodikken vår ble først bestemt i sprint 2 og ble ferdig utredet i sprint 6. Dermed vil sprint 7 i Jira være den første sprinten i prosjektarbeidet ferdig utredet “scrum” arbeidsmetodikk.

E.2 Sprint 1

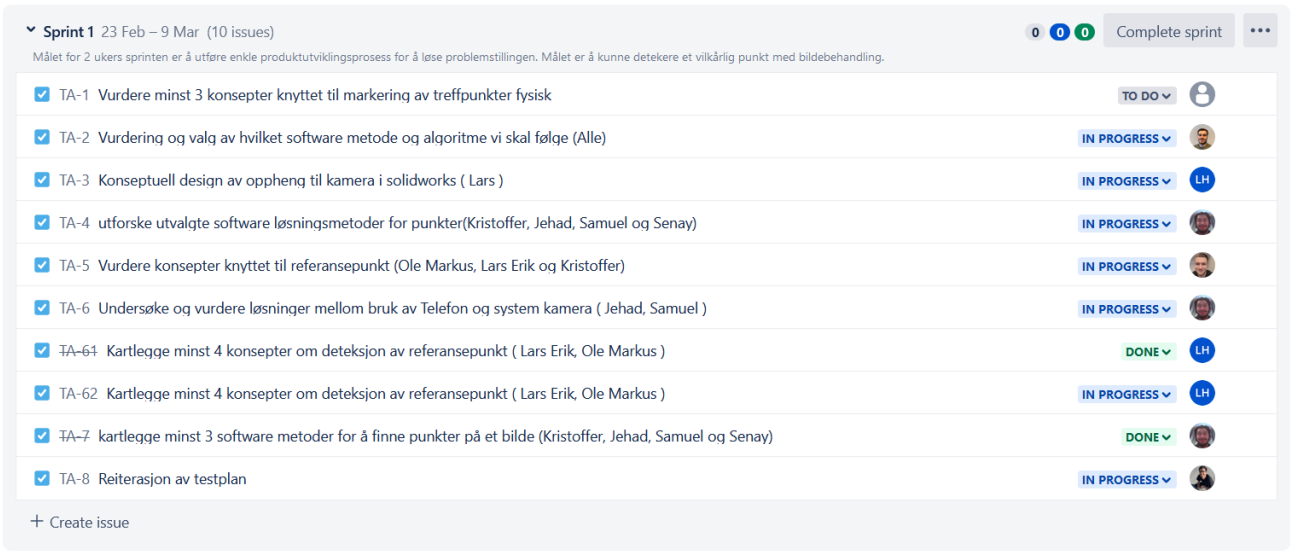
Sprint 1: Målet for 2 ukers sprinten er å utføre enkle produktutviklingensprosess for å løse problemstillingen. Målet er å kunne detektere et vilkårlig punkt med bildebehandling

Hva vi har klart å inkrementere på produktet:

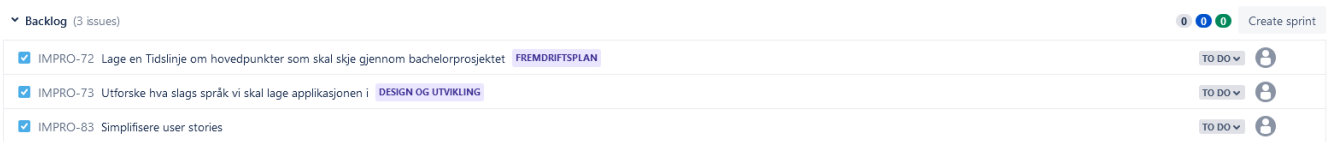
- Finne treffpunkter på et bilde av en skyteskive der treffpunkter først har blitt markert av en farge og vi kan dermed detektere den fargen.
- Vi har utforsket algoritimer på å finne referansepunktet som skal være på midten av skyteblinken. Kommet langt nok til å markere referansepunktet, men er ikke helt nøyaktig posisjonert.
- Designet 3d modell for oppheng av kamera på Solidworks.

Endringer vi må utføre:

- utdype oppgaver mer i Jira for å skape progresjon i Scrum board
- Dele oppgaver i en mer bredt spektrum så vi kan utnytte så mye info som mulig
- Sette opp oppgaver i Backlog for gjennomføring i senere tid
- Sett sprint målet i flere hovedpunkter
- Dokumentere underveis ''issues'' som blir gjennomført.
- Alle sprinter skal forholde seg til teknisk produkt



Figur 108: Sprint 1 board



Figur 109: Sprint 1 backlog

E.3 Sprint 2

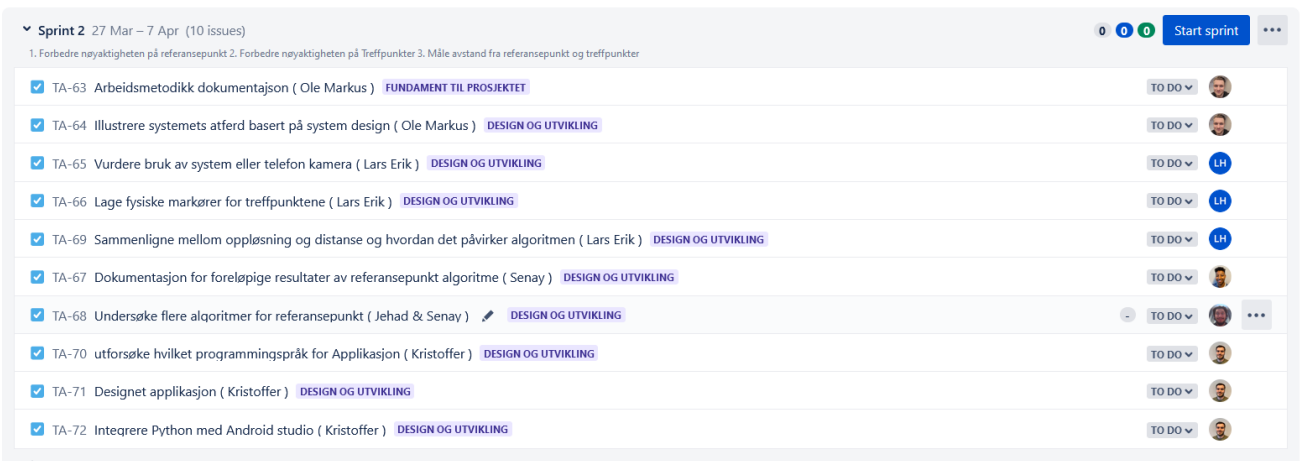
1. Forbedre nøyaktigheten på referansepunktet (ja) 2. Forbedre nøyaktigheten på treffpunkter (ja) 3. Måle avstand fra referansepunkt og treffpunkter (nei)

Hva vi har klart å inkrementere på produktet

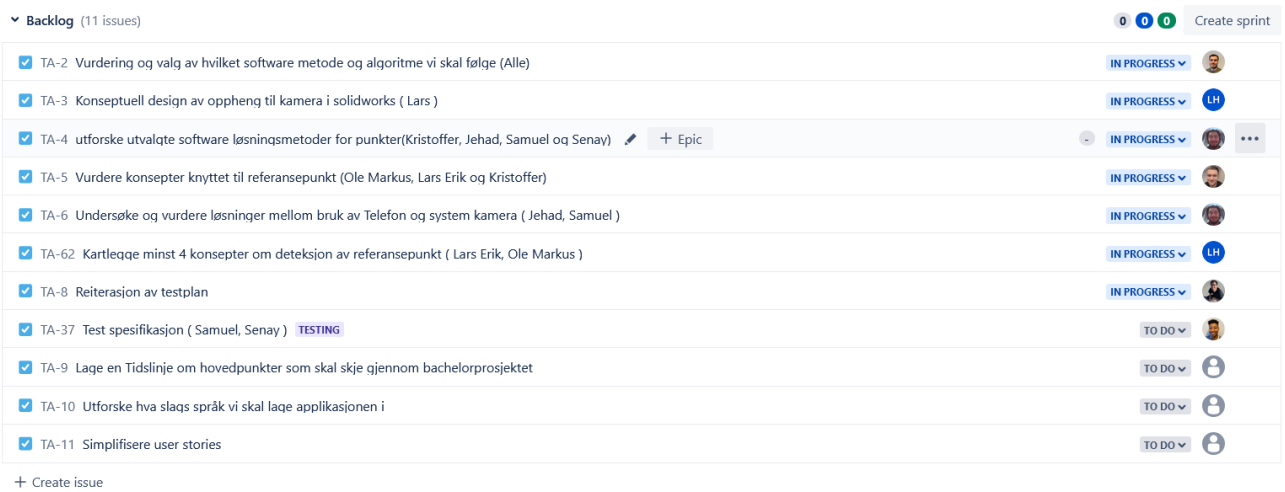
- Vi har klart å finne referansepunkt baser på ChatGPT fil, som ikke er bra nok når det kommer til forskjellige treff figurer
- Vi har klart å finne treffpunkter basert på velge et piksel som er nærme senterpunkt til et treffpunkt.

Endringer vi må utføre:

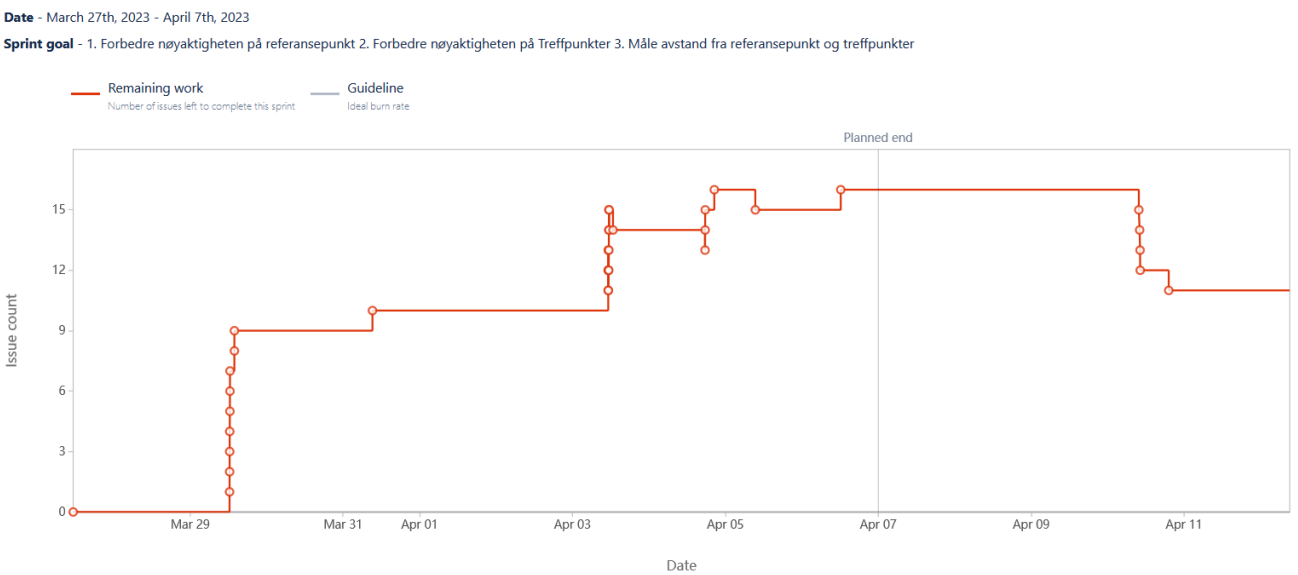
- Ikke alle må sitte sammen, men på starten av dagen så må alle sitte sammen for en liten daily scrum. På slutten av dagen, 1.5t før dagen er slutt så sitter alle sammen.



Figur 110: Sprint 2 board



Figur 111: Sprint 2 backlog



Figur 112: Sprint 2 burndownchart

Completed issues

View in issue navigator

Key	Summary	Issue type	Epic	Status	Assignee	Issue count
TA-63	Arbeidsmetodikk dokumentajson (Ole Markus)	Task	FUNDAMENT TIL PROSJE...	DONE		1
TA-64	Illustrere systemets atferd basert på system design (Ole Markus)	Task	DESIGN OG UTVIKLING	DONE		1
TA-67	Dokumentasjon for foreløpige resultater av referansepunkt algoritme (Se...	Task	DESIGN OG UTVIKLING	DONE		1
TA-68	Undersøke flere algoritmer for referansepunkt (Jehad & Senay)	Task	DESIGN OG UTVIKLING	DONE		1
TA-70	utforsøke hvilket programmingspråk for Applikasjon (Kristoffer)	Task	DESIGN OG UTVIKLING	DONE		1
TA-72	Integrere Python med Android studio (Kristoffer)	Task	DESIGN OG UTVIKLING	DONE		1
TA-1	Vurdere minst 3 konsepter knyttet til markering av treffpunkter fysisk	Task		DONE	LH	1
TA-73	Utvikle treffpunkt algoritmen til å utvelge en piksel som er nærmest sente...	Task	DESIGN OG UTVIKLING	DONE		1
TA-74	Finne piksel høyde og bredde i cm utifra X og Y akse korset	Task	DESIGN OG UTVIKLING	DONE		1

Figur 113: Sprint 2 completed issues

E.4 Sprint 3:

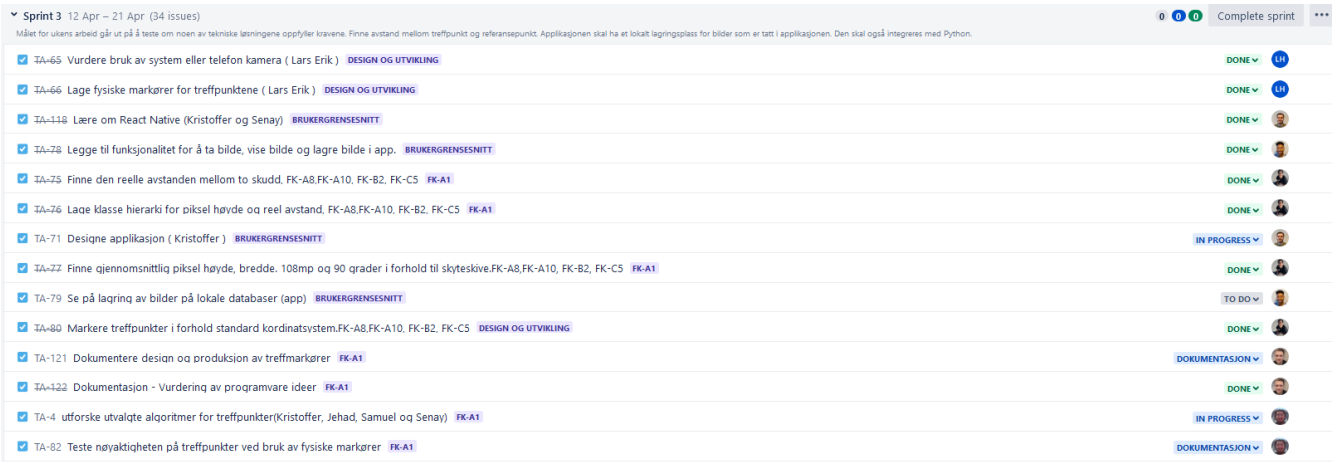
Målet for ukens arbeid går ut på å teste om noen av tekniske løsningene oppfyller kravene. Finne avstand mellom treffpunkt og referansepunkt. Applikasjonen skal ha et lokalt lagringsplass for bilder som er tatt i applikasjonen. Den skal også integreres med Python.

Hva har vi klart å inkrementere på produktet

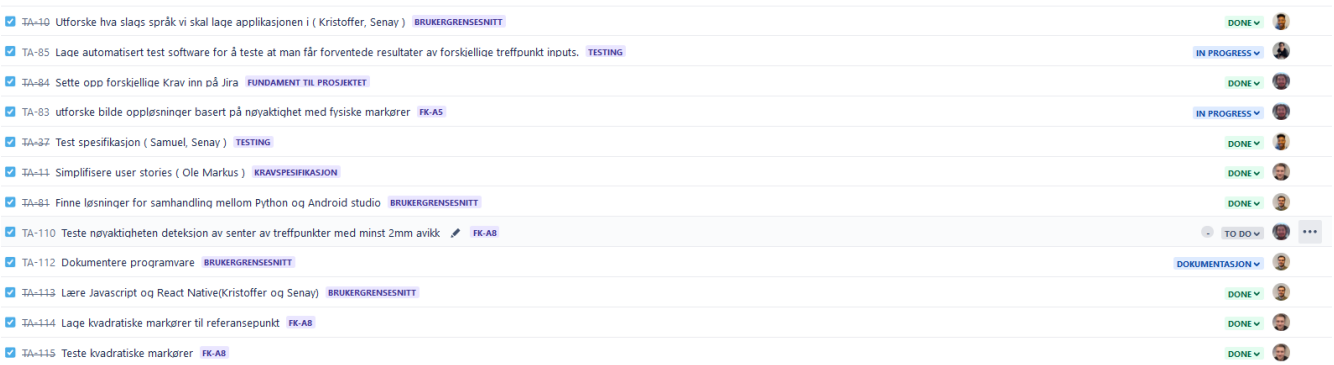
- Finne avstand mellom treffpunkt og referansepunkt (bilde har oppløsning 108MP)
- Applikasjon har et lokalt lagringsplass for bilder som er tatt i applikasjonen

Endringer vi må utføre

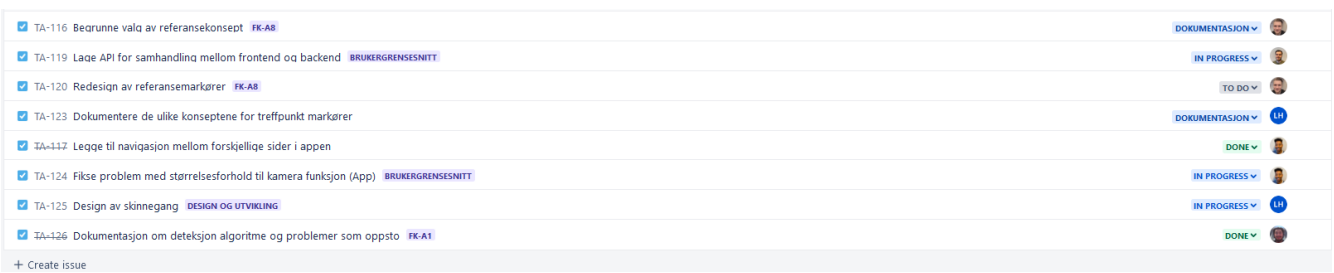
- Siden det er en måned igjen til innlevering av bachelorprosjektet så endrer vi til bare en ukes sprint
- Alle er pliktig til å selv legge inn dokumentasjon om hva de har gjort på overleaf på onsdager og fredager.



Figur 114: Sprint 3 board

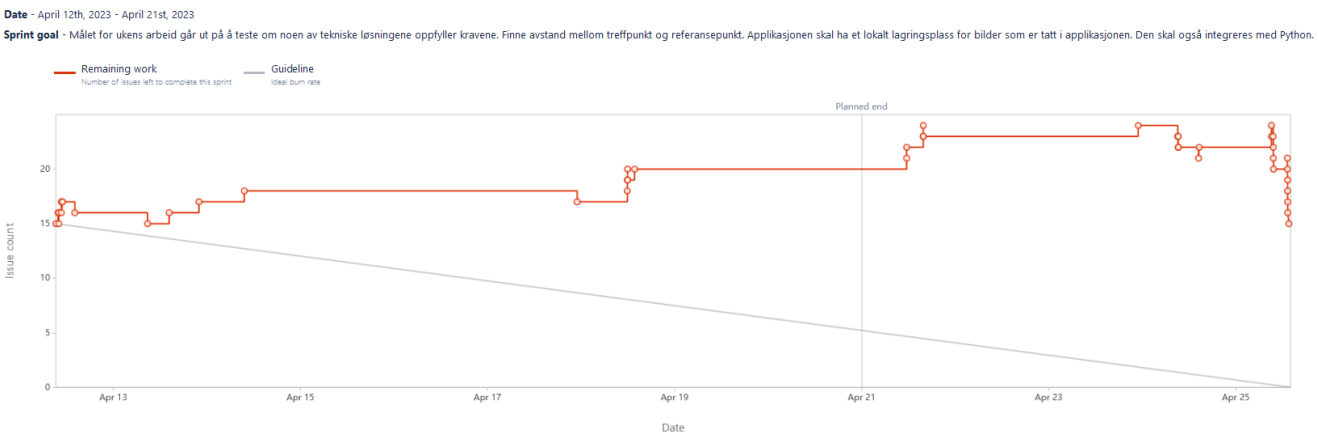


Figur 115: Sprint 3 board

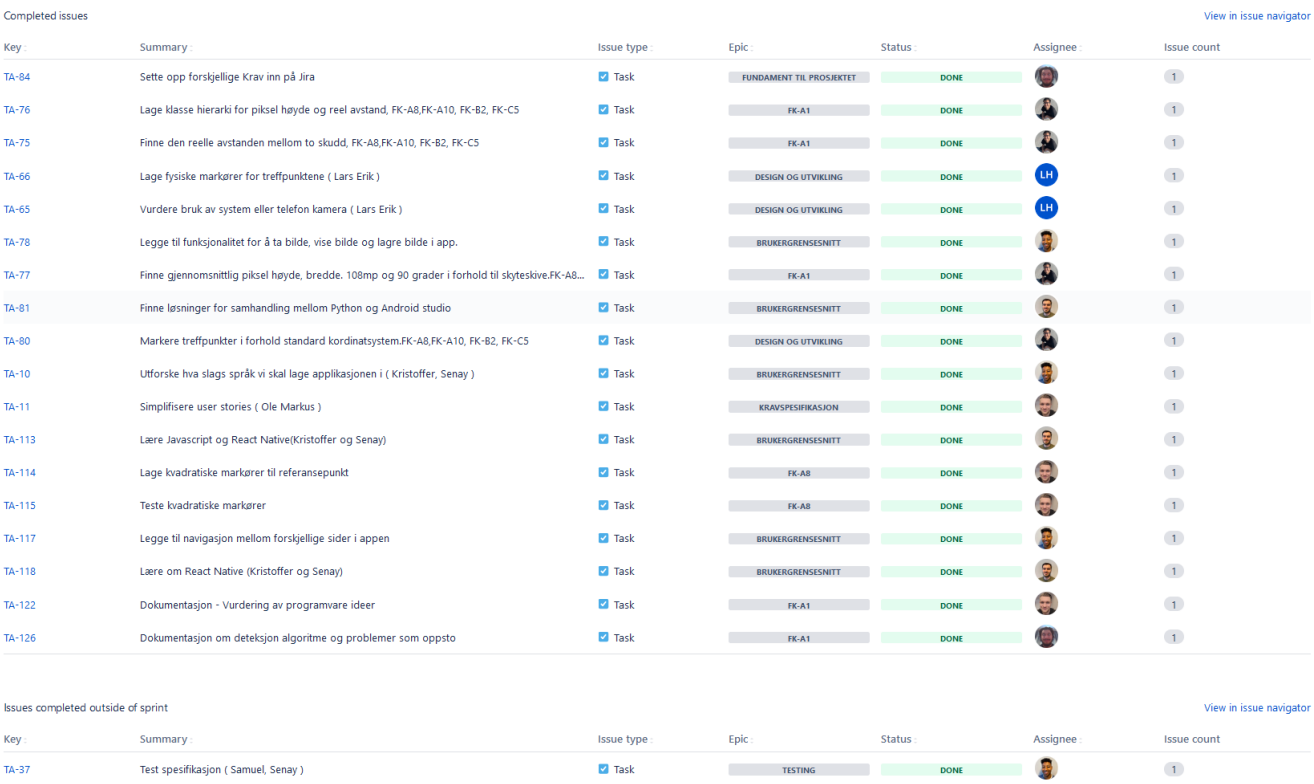


Figur 116: Sprint 3 board

Figur 117: Sprint 3 backlog



Figur 118: Sprint 3 burndownchart



Completed issues

View in issue navigator

Key	Summary	Issue type	Epic	Status	Assignee	Issue count
TA-84	Sette opp forskjellige Krav inn på Jira	Task	FUNDAMENT TIL PROSJEKTET	DONE		1
TA-76	Lage klasse hierarki for piksel høyde og reel avstand, FK-A8,FK-A10, FK-B2, FK-C5	Task	FK-A1	DONE		1
TA-75	Finne den reelle avstanden mellom to skudd, FK-A8,FK-A10, FK-B2, FK-C5	Task	FK-A1	DONE		1
TA-66	Lage fysiske markører for treffpunktene (Lars Erik)	Task	DESIGN OG UTVIKLING	DONE	LH	1
TA-65	Vurdere bruk av system eller telefon kamera (Lars Erik)	Task	DESIGN OG UTVIKLING	DONE	LH	1
TA-78	Legge til funksjonalitet for å ta bilde, vise bilde og lagre bilde i app.	Task	BRUKERGRENSESNITT	DONE		1
TA-77	Finne gjennomsnittlig piksel høyde, bredde. 108mp og 90 grader i forhold til skyteskive.FK-A8...	Task	FK-A1	DONE		1
TA-81	Finne løsninger for samhandling mellom Python og Android studio	Task	BRUKERGRENSESNITT	DONE		1
TA-80	Markere treffpunkter i forhold standard kordinatsystem.FK-A8,FK-A10, FK-B2, FK-C5	Task	DESIGN OG UTVIKLING	DONE		1
TA-10	Utforske hva slags språk vi skal lage applikasjonen i (Kristoffer, Senay)	Task	BRUKERGRENSESNITT	DONE		1
TA-11	Simplifisere user stories (Ole Markus)	Task	KRAVSPESIFIKASJON	DONE		1
TA-113	Lære Javascript og React Native(Kristoffer og Senay)	Task	BRUKERGRENSESNITT	DONE		1
TA-114	Lage kvadratiske markører til referansepunkt	Task	FK-A8	DONE		1
TA-115	Teste kvadratiske markører	Task	FK-A8	DONE		1
TA-117	Legge til navigasjon mellom forskjellige sider i appen	Task	BRUKERGRENSESNITT	DONE		1
TA-118	Lære om React Native (Kristoffer og Senay)	Task	BRUKERGRENSESNITT	DONE		1
TA-122	Dokumentasjon - Vurdering av programvare ideer	Task	FK-A1	DONE		1
TA-126	Dokumentasjon om deteksjon algoritme og problemer som oppsto	Task	FK-A1	DONE		1

Issues completed outside of sprint

View in issue navigator

Key	Summary	Issue type	Epic	Status	Assignee	Issue count
TA-37	Test spesifikasjon (Samuel, Senay)	Task	TESTING	DONE		1

Figur 119: Sprint 3 completed issues

E.5 Sprint 4

Målet for ukens arbeid går ut på å teste om noen av tekniske løsningene oppfyller kravene. system integrasjon av backend programvare. Applikasjon kan integreres med Python. Lage en fullstendig konseptuell design for Applikasjon.

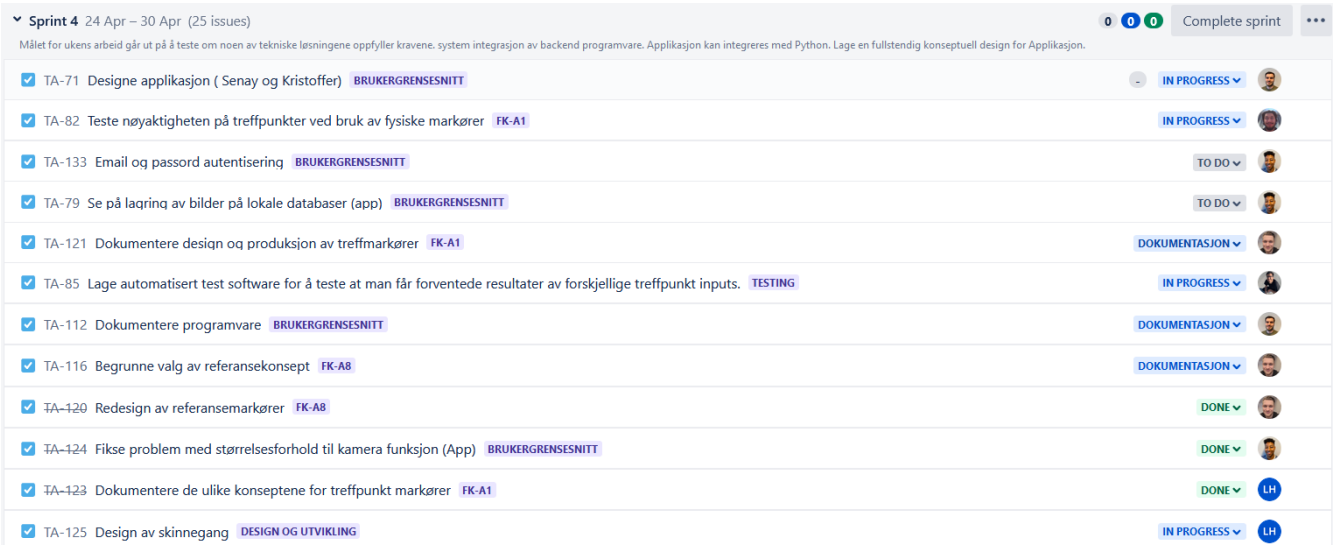
Basert på mål har vi ikke incrementet produktet, derimot vi har bygd opp flere funksjoner for å oppnå de målene.

Hva vi har klart å inkrementere på produktet:

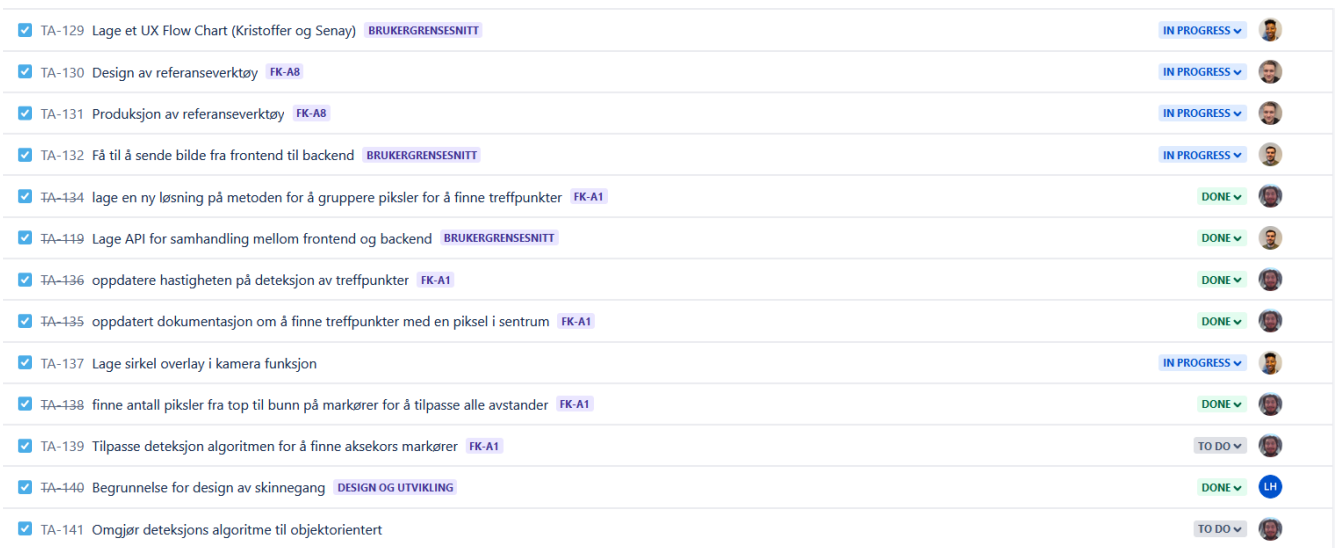
- Oppdatert konseptuell design for applikasjon
- Finne treffpunkter med hastighet på 10 sekunder når bildeoppløsning er 12K * 9K = 108MP
- Basert på ny metode for å finne treffpunkter på kan vi finne markører med mindre avstand for hverandre
- Finne senterpunktet av skytefiguren automatisk basert på aksekors markører.
- Lagde et fysisk verktøy for å plassere aksekors markører nøyaktig.

Endringer vi må utføre:

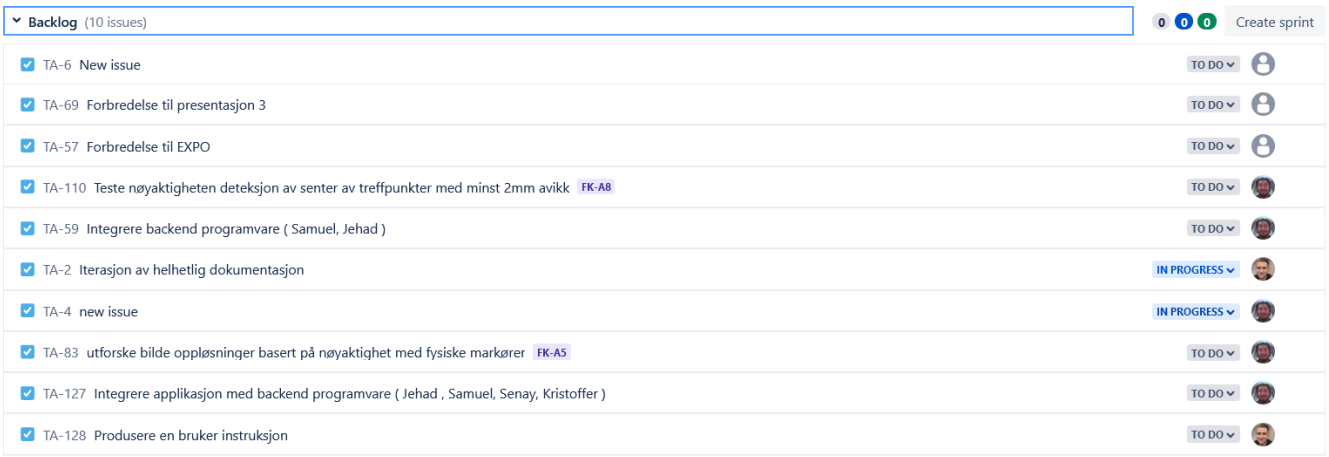
- Klargjøre kilder til argumentasjon på møter



Figur 120: Sprint 4 board



Figur 121: Sprint 4 board



Figur 122: Sprint 4 backlog

Figur 123: Sprint 4 burndownchart

Completed issues

View in issue navigator

Key	Summary	Issue type	Epic	Status	Assignee	Issue count
TA-123	Dokumentere de ulike konseptene for treffpunkt markører	Task	FK-A1	DONE	LH	1
TA-120	Redesign av referansemarkører	Task	FK-A8	DONE		1
TA-124	Fikse problem med størrelsesforhold til kamera funksjon (App)	Task	BRUKERGRENSESNITT	DONE		1
TA-119	Lage API for samhandling mellom frontend og backend	Task	BRUKERGRENSESNITT	DONE		1
TA-134	lage en ny løsning på metoden for å gruppere piksler for å finne treffpun...	Task	FK-A1	DONE		1
TA-135	oppdatert dokumentasjon om å finne treffpunkter med en piksel i sentrum	Task	FK-A1	DONE		1
TA-136	oppdatere hastigheten på deteksjon av treffpunkter	Task	FK-A1	DONE		1
TA-138	finne antall piksler fra top til bunn på markører for å tilpasse alle avstander	Task	FK-A1	DONE		1
TA-140	Begrunnelse for design av skinnegang	Task	DESIGN OG UTVIKLING	DONE	LH	1

Figur 124: Sprint 4 completed issues

E.6 Sprint 5

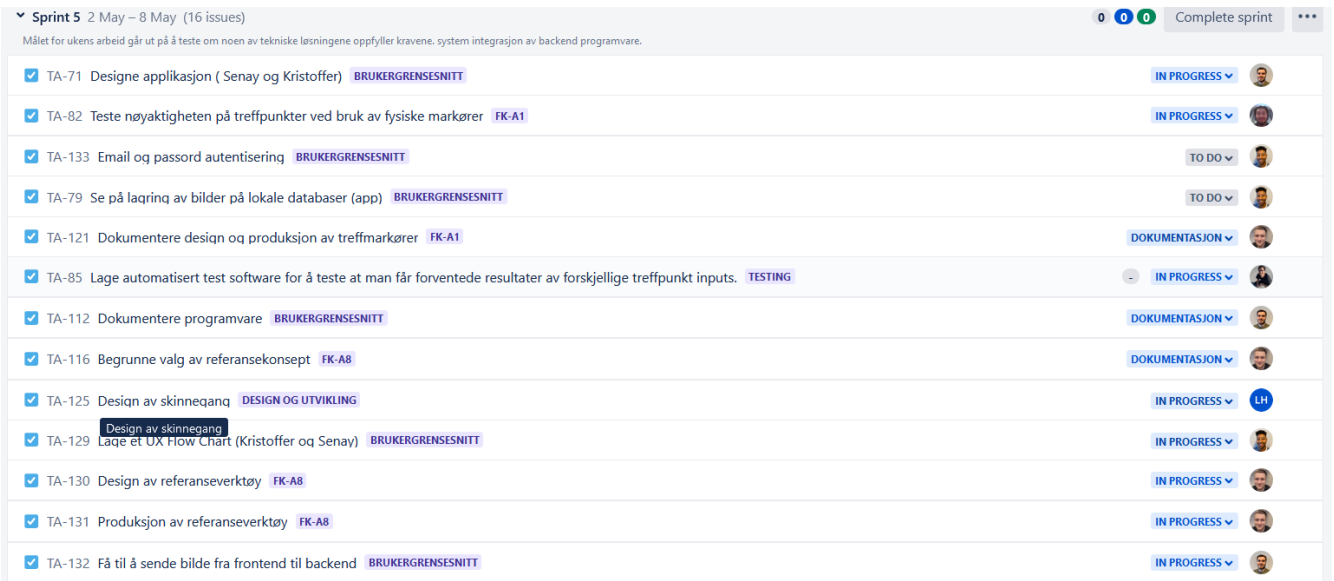
Målet for ukens arbeid går ut på å teste om noen av tekniske løsningene oppfyller kravene. system integrasjon av backend programvare.

Hva vi har klart å inkrementere på produktet

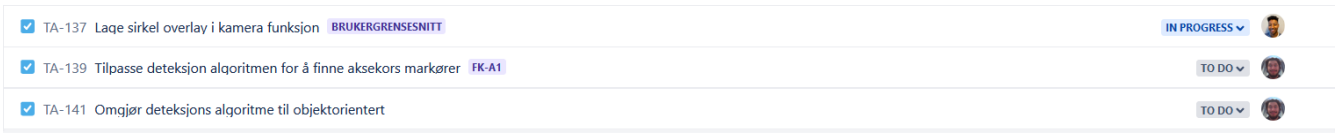
- Klart å kombinere backend programvare
- Lagt til et gyroscope og sirkler som overlay på applikasjone for å definere avstand fra skyteblinken
- Modifisering av aksekors verktøy for å plassere markører på aksekorasene riktig

Endringer vi må utføre

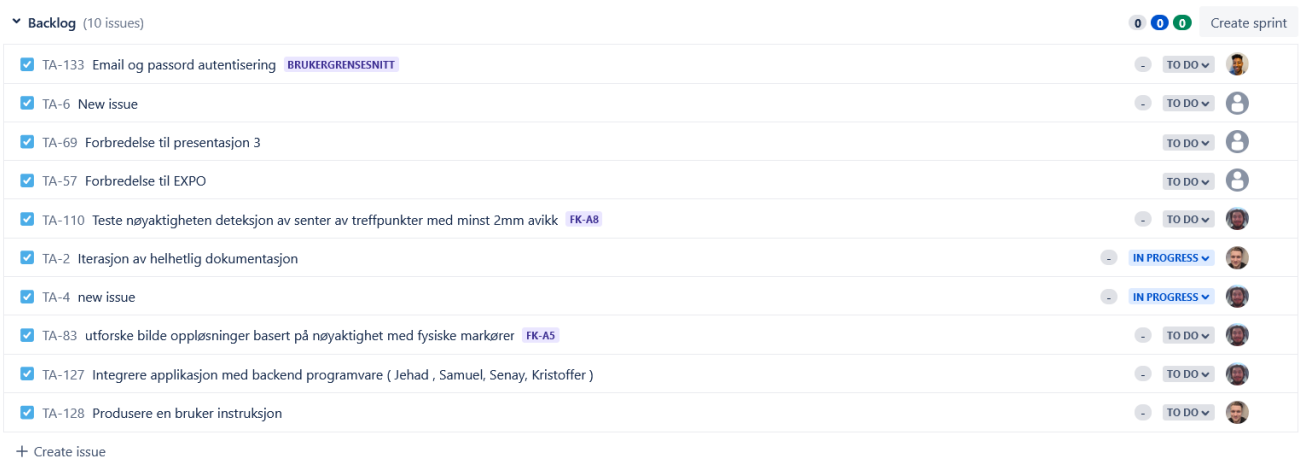
-



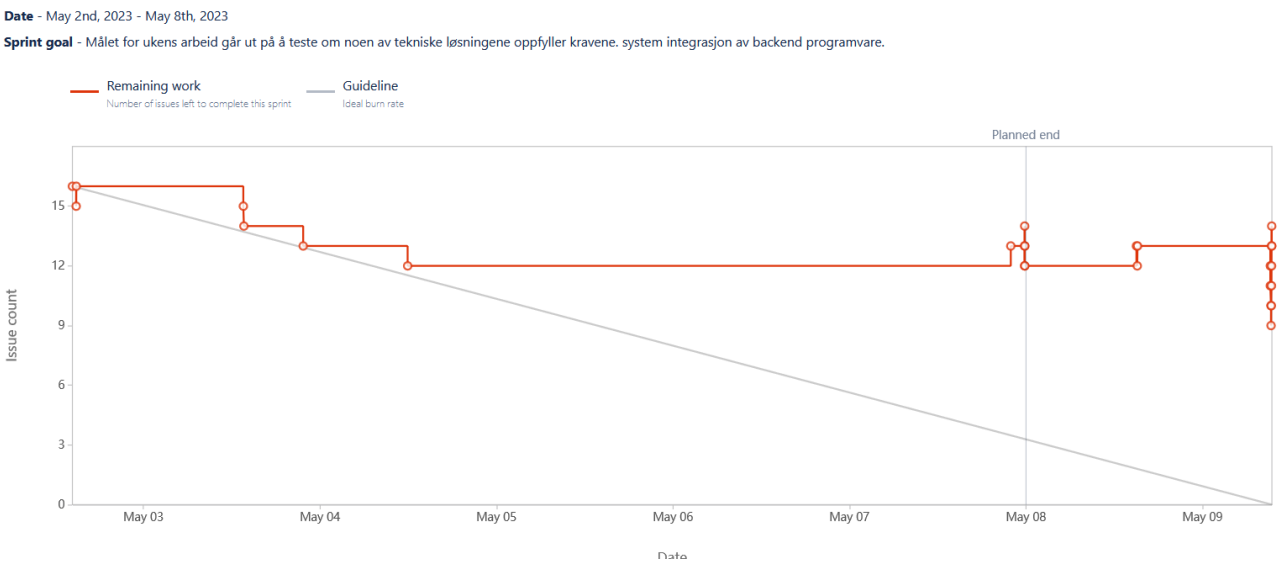
Figur 125: Sprint 5 board



Figur 126: Sprint 5 board



Figur 127: Sprint 5 backlog



Figur 128: Sprint 5 burndownchart

Completed issues

View in issue navigator

Key	Summary	Issue type	Epic	Status	Assignee	Issue count
TA-71	Designe applikasjon (Senay og Kristoffer)	Task	BRUKERGRENSESNITT	DONE		1
TA-141	Omgjør deteksjons algoritme til objektorientert	Task	DESIGN OG UTVIKLING	DONE		1
TA-129	Lage et UX Flow Chart (Kristoffer og Senay)	Task	BRUKERGRENSESNITT	DONE		1
TA-130	Design av referanseverktøy	Task	FK-A8	DONE		1
TA-131	Produksjon av referanseverktøy	Task	FK-A8	DONE		1
TA-132	Få til å sende bilde fra frontend til backend	Task	BRUKERGRENSESNITT	DONE		1
TA-137	Lage sirkel overlay i kamera funksjon	Task	BRUKERGRENSESNITT	DONE		1
TA-139	Tilpasse deteksjon algoritmen for å finne aksekors markører	Task	FK-A8	DONE		1
TA-142	Legge til gyroskop	Task	BRUKERGRENSESNITT	DONE		1
TA-144	teste forskjellige farge på markører mørkeblå, lyse blå, grønn	Task	DESIGN OG UTVIKLING	DONE		1
TA-59	Integrere backend programvare (Samuel, Jehad)	Task	DESIGN OG UTVIKLING	DONE		1
TA-146	Produksjon av treffmarkører	Task	FK-A8	DONE		1
TA-151	redesign av referanseverktøy	Task	DESIGN OG UTVIKLING	DONE	LH	1

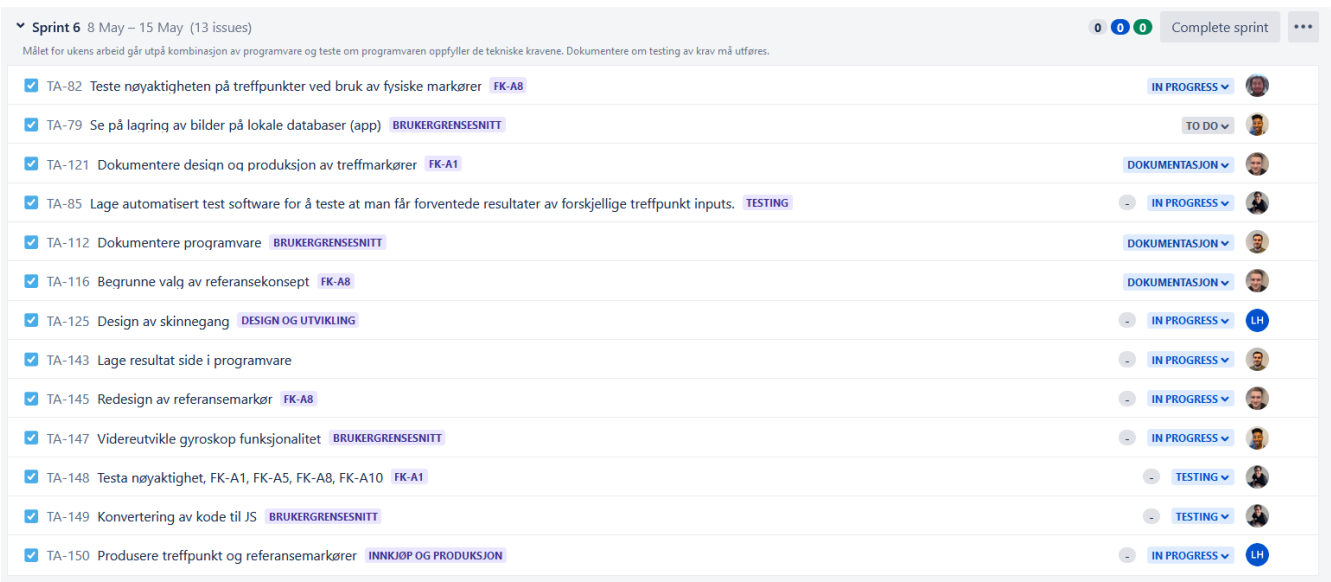
Figur 129: Sprint 5 completed issues

E.7 Sprint 6

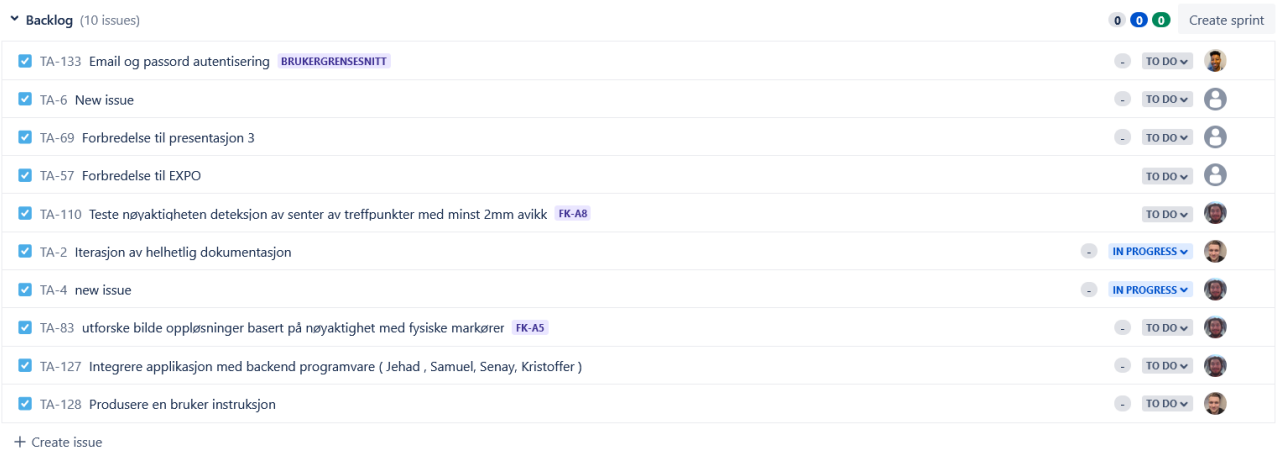
Målet for ukens arbeid går utpå kombinasjon av programvare og teste om programvaren oppfyller de tekniske kravene. Dokumentere om testing av krav må utføres.

Hva har vi klart å inkrementere på produktet

- Klarer å finne markører på forskjellige oppløsninger men ikke samme telefon i henhold til forskjellige RGB algoritmer i hvert telefon.
- økt nøyaktighet på deteksjon av senter av treffpunkter fra gjennomsnittlig nøyaktighet på 3-2mm til under 2mm.
- Gyroskop og avstand sirkel er plassert i senter av kamera når man tar et bilde på applikasjonen.
- referanseverktøy for å plassere aksekors markører mer nøyaktig
- Design av skinnegang for kamera til kalibrering av flere skyteskiver



Figur 130: Sprint 6 board



Figur 131: Sprint 6 backlog

Figur 132: Sprint 6 burndownchart

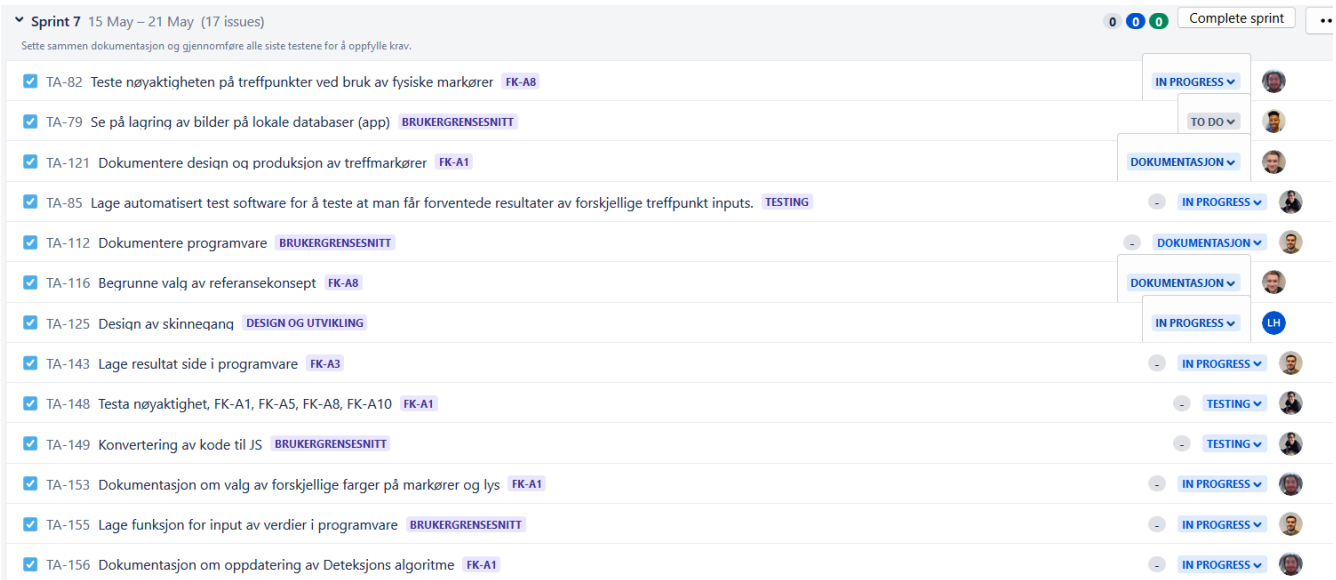
Key	Summary	Issue type	Epic	Status	Assignee	Issue count
TA-145	Redesign av referanseverktøy	Task	FK-A8	DONE		1
TA-147	Videreutvikle gyroskop funksjonalitet	Task	BRUKERGRENSESNITT	DONE		1
TA-150	Produsere treffpunkt og referansemarkører	Task	INNKJØP OG PRODUKSJ...	DONE	LH	1
TA-152	finpusse og kommentere deteksjons Algoritmen	Task	FK-A8	DONE		1
TA-154	Redesign av referansemarkør	Task	FK-A8	DONE		1
TA-157	Produksjon av referanseverktøy	Task	FK-A8	DONE		1
TA-158	Produksjon av referansemarkører	Task	FK-A8	DONE		1
TA-110	Teste nøyaktigheten deteksjon av senter av treffpunkter med minst 2mm ...	Task	FK-A8	DONE		1
TA-159	Sette sammen og kommentere applikajons algoritme	Task	BRUKERGRENSESNITT	DONE		1
TA-160	Lage metode for å velge oppløsning i appen	Task	BRUKERGRENSESNITT	DONE		1
TA-161	endre på deteksjons algoritmen for å finne nærme senter av skyteskiven f...	Task	FK-A8	DONE		1
TA-162	tilpasse deteksjonsalgoritmen til flere oppløsninger	Task	FK-A1	DONE		1
TA-165	Måle treffpunkter som kontroll for testing av programvare	Task		DONE		1
TA-166	Se på muligheten til å gjøre om kode slik at alt kan gjøres inne i appen	Task	BRUKERGRENSESNITT	DONE		1

Figur 133: Sprint 6 completed issues

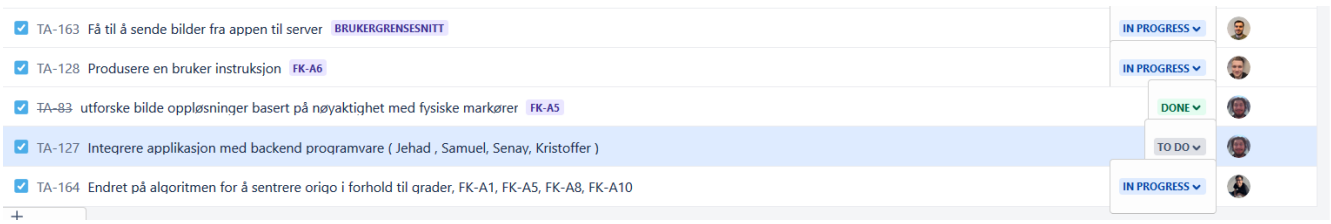
E.8 Sprint 7

sette sammen dokumentasjon og gjennomføre alle siste testene for å oppfylle krav.

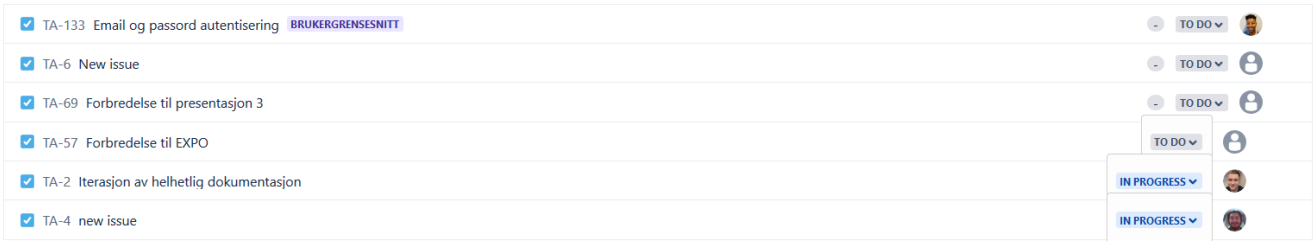
- Dokumentasjon og kravene er gjennomgått og gjennomført



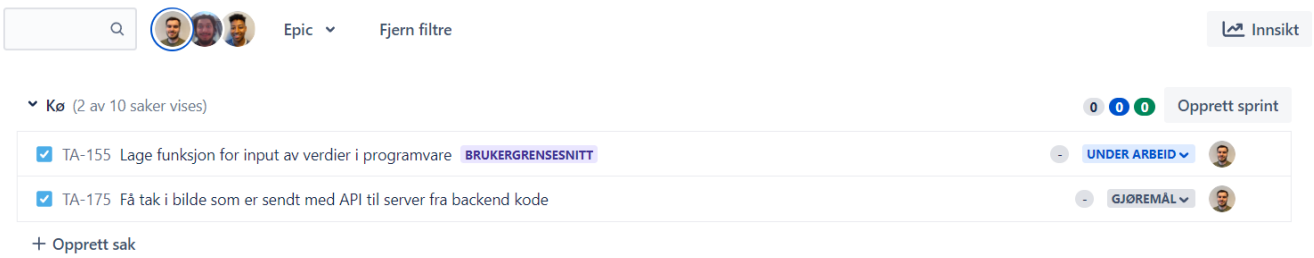
Figur 134: Sprint 7 board



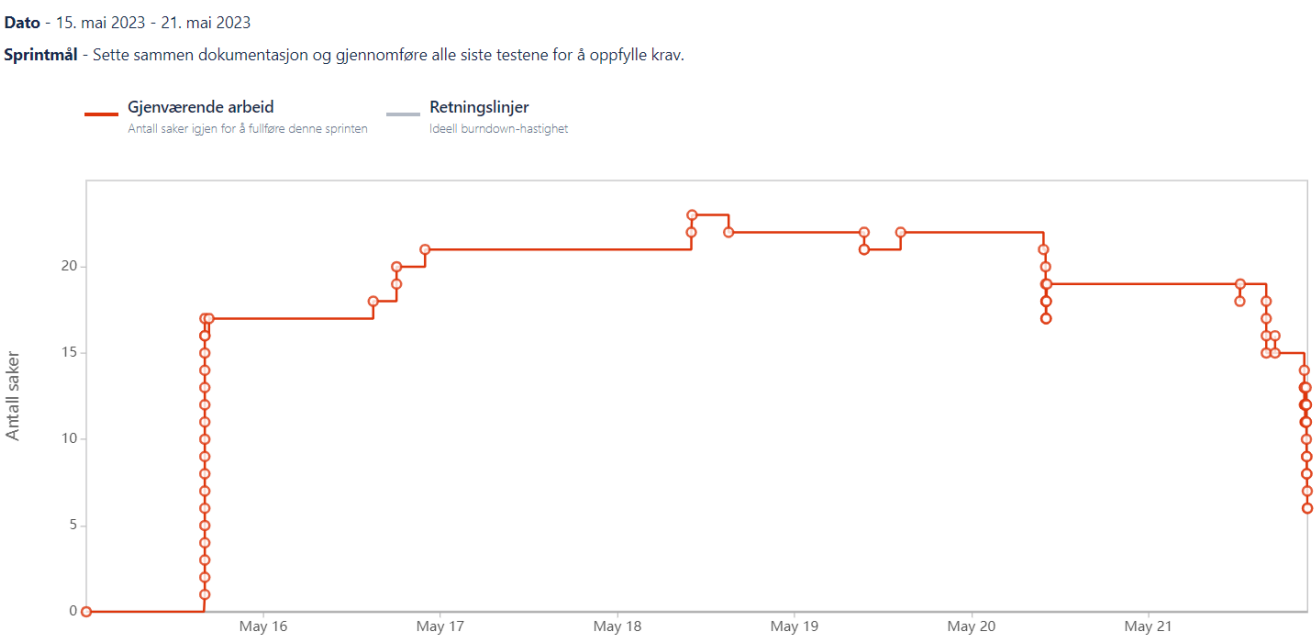
Figur 135: Sprint 7 board



Figur 136: Sprint 7 backlog før sprint



Figur 137: Sprint 7 backlog etter sprint



Figur 138: Sprint 7 burndownchart

Fullførte saker Vis i saksnavig

Nøkkel	Sammendrag	Sakstype	Epic	Status	Tilordnet person	Antall saker
TA-163	Få til å sende bilder fra appen til server	Oppgave	BRUKERGRENSESNI...	FERDIG		1
TA-164	Endret på algoritmen for å sentrere origo i forhold til grader, F...	Oppgave		FERDIG		1
TA-143	Lage resultat side i programvare	Oppgave	FK-A3	FERDIG		1
TA-121	Dokumentere design og produksjon av treffmarkører	Oppgave	FK-A1	FERDIG		1
TA-125	Design av skinnegang	Oppgave	DESIGN OG UTVIKL...	FERDIG	LH	1
TA-148	Testa nøyaktighet, FK-A1, FK-A5, FK-A8, FK-A10	Oppgave	FK-A1	FERDIG		1
TA-149	Konvertering av kode til JS	Oppgave	BRUKERGRENSESNI...	FERDIG		1
TA-128	Produsere en bruker instruksjon	Oppgave	FK-A6	FERDIG		1
TA-85	Automatisert test software for å teste at man får forventede re...	Oppgave	TESTING	FERDIG		1
TA-82	Teste nøyaktigheten på treffpunkter ved bruk av fysiske markø...	Oppgave	FK-A8	FERDIG		1
TA-153	Dokumentasjon om RGB verdier og forskning	Oppgave	FK-A1	FERDIG		1
TA-112	Dokumentere programvare	Oppgave	BRUKERGRENSESNI...	FERDIG		1

Figur 139: Sprint 7 backlog før sprint

TA-156	Dokumentasjon om hastighet og unødvendig data på deteksj...	Oppgave	FK-A1	FERDIG		1
TA-116	Begrunne valg av referansekonsept	Oppgave	FK-A8	FERDIG		1
TA-83	utforske bilde oppløsninger basert på nøyaktighet med fysiske...	Oppgave	FK-A5	FERDIG		1
TA-167	Produsere treffmarkører	Oppgave	FK-A8	FERDIG		1
TA-168	Testing av treffpunktnøyaktighet langs x- og y-akse	Oppgave	FK-A5	FERDIG		1
TA-2	Iterasjon av helhetlig dokumentasjon	Oppgave	PROSJEKT RAPPORT	FERDIG		1
TA-169	Dokumentere programvare sin virkemåte	Oppgave	BRUKERGRENSESNI...	FERDIG		1
TA-170	Dokumentere designet for skinnegang	Oppgave		FERDIG	LH	1
TA-171	Manuelle målinger av treffpunkter for å teste nøyaktigheten til...	Oppgave		FERDIG	LH	1
TA-172	Dokumentasjon av koden i appen	Oppgave	BRUKERGRENSESNI...	FERDIG		1
TA-173	Analysere treffpunkt data med statistikk, FK-A1, FK-A5, FK-A8, ...	Oppgave		FERDIG		1
TA-174	Tilpasse algoritmen til flere oppløsninger og telefoner Iphone...	Oppgave		FERDIG		1
TA-177	2D-tegninger	Oppgave		FERDIG	LH	1
TA-178	Extracta data fra excel til latex, FK-A1, FK-A5, FK-A8, FK-A10	Oppgave		FERDIG		1

Figur 140: Sprint 7 backlog før sprint

TA-179	Gjøre klart alt vi trenger å ha med i rapport og på minnepenn	Oppgave	FERDIG		1
TA-180	finpusse Dokumentasjon om bruk av Chatgpt kode for å finne ...	Oppgave	FERDIG		1
TA-181	Dokumentere om forskjellige måter å løse deteksjons algoritme	Oppgave	FERDIG		1
TA-182	Finpusse kode og kommentere ferdig på alle essenielle kodefil...	Oppgave	FERDIG		1
TA-183	Kommentere kode	Oppgave	FERDIG		1
TA-184	Brukerinstruks	Oppgave	FERDIG	LH	1

Figur 141: Sprint 7 backlog før sprint

F Oppfølgningsdokumenter

F.1 Oppfølgningsdokument uke 1-5

Dette er et felles oppfølgningsdokument for gruppen fra uke 1 til og med 5 fordi igjennom disse ukene så har vi som gruppe jobbet stort sett sammen, og det har vært de første ukene hvor vi ikke har jobbet så mye med oppgaven fordi oppstart i andre emner. Fra uke 6 begynner vi å føre mer individuelle oppfølgninsdokumenter som viser hva hvert medlem har jobbet med og hva man tenker å jobbe med uken etter.

Disse ukene har vi hatt oppstartsfase hvor vi har jobbet mye med å få på plass ting. Dette er både prosjekt relatert med roller og ansvarsområdet, også har vi vært på besøk hos oppdragsgiver. Disse ukene har vi ikke hatt noen spesielle individuelle arbeidsoppgaver, men disse starter fra uke 6. Oppgaver som krav og user stories er gjort i felleskap.

Timeliste

Timeliste	Uke 5	Totalt
Senay	23	75
Ole Markus	23	71
Lars Erik	23	75
Jehad	24	78,5
Samuel	23	83
Kristoffer	23	74
Sum	142 timer	456,5 timer

F.2 Oppfølgnignsdokument uke 6

Denne uken begynner vi med å jobbe litt mer individuelt enn vi har gjort til nå. Frem til nå så har vi jobbet med å utarbeide gruppen, kravene og arbeidsmetodikk.

Senay

- Jobbet med testplan

Jehad

- Jobbet med arbeidsmetodikk

Lars Erik

- Jobbet med Gantt-skjema

Ole Markus

- Jobbet med Gantt-skjema

Kristoffer

- Jobbet med risikoanalyse

Samuel

- Forberede testmetode og testplan forslag til gruppen
- Delt opp oppgavene mellom oss hvor jeg og Senay jobbed med testmetode og testplan

Timeliste

Timeliste	Uke 6	Totalt
Senay	28	103
Ole Markus	22	93
Lars Erik	22	97
Jehad	29	107,5
Samuel	30	113
Kristoffer	23	97
Sum	154 timer	610,5 timer

Neste uke: Neste uke er 1. presentasjon. Det er på fredagen så hele uken går til å klargjøre dokumentasjonen, lage presentasjon og øve slik at vi er klare.

F.3 Oppfølgningsdokument uke 7

Denne uken var det tid får vår 1. presentasjon som var på fredagen. Hele uken gikk dermed for alle å klargjøre dokumentasjonen til denne og å klargjøre seg for presentasjonen. Neste uke begynner vi for alvor å gå løs på oppgaven og begynne å jobbe mot å fullføre kravene våre.

Senay

- Ferdigstille dokumentasjonen til 1. presentasjon
- Øve til 1. presentasjon

Jehad

- Ferdigstille dokumentasjonen til 1. presentasjon
- Øve til 1. presentasjon

Lars Erik

- Ferdigstille dokumentasjonen til 1. presentasjon
- Øve til 1. presentasjon

Ole Markus

- Ferdigstille dokumentasjonen til 1. presentasjon
- Øve til 1. presentasjon

Kristoffer

- Ferdigstille dokumentasjonen til 1. presentasjon
- Øve til 1. presentasjon

Samuel

- Webutvikling
- Testplan
- Rapport
- testmetoder
- User stories
- Presentasjon

Timeliste

Timeliste	Uke 7	Totalt
Senay	36	139
Ole Markus	40	133
Lars Erik	35	132
Jehad	37,5	145
Samuel	48	161
Kristoffer	35	132
Sum	231,5 timer	842 timer

Neste uke Neste uke er planen at gruppen skal jobbe med tilbakemeldingene vi får på presentasjonen og fikse på det som må fikses. Samtidig skal vi for alvor begynne på det tekniske og starte å komme opp med konsepter og mulige løsninger som vi må utforske.

F.4 Oppfølgningsdokument uke 8

Vi har nå endelig kommet i gang med å jobbe med oppgaven og jobbe mot kravene som er satt. Data studentene har begynt å utforske bildebehandling og begynt å programmere litt for å se hva vi kan få til med dette. Maskin har begynt å tegne litt og tenke på hvordan dette fysiske systemet kan se ut.

Senay

- Jobbe med tilbakemeldingene
- Utforske hvilke programmeringsspråk som egner seg best til vår oppgave
- Utforske bildebehandling

Jehad

- Jobbe med tilbakemeldingene
- Utforske bildebehandling
- Utforske hva slags typer kameraer og bildeoppløsning som tilpasser oppgaven vår.

Lars Erik

- Jobbe med tilbakemeldingene
- Utforske konsepter til løsninger
- Prosjektplanlegging

Ole Markus

- Jobbe med tilbakemeldingene
- Utforske konsepter

- Jobbe med dokumentasjonen

Kristoffer

- Jobbe med tilbakemeldingene fra presentasjonen
- Lære om hva bildebehandling er
- Jobbe med dokumentasjonen

Samuel

- Utforsking av hvordan man lager executable files
- Utforsking av forskjellige løsninger av bildebehandling
- Reiterasjon av testplan etter tilbakemeldinger fra første presentasjon
- Utforsking av forskjellige kamerae oppløsninger
- Utforsking av forskjellige mobile kameraoppløsninger

Timeliste

Timeliste	Uke 8	Totalt
Senay	25	164
Ole Markus	26	159
Lars Erik	26	158
Jehad	12	157
Samuel	20	181
Kristoffer	21	153
Sum	130 timer	972 timer

Neste uke: Neste uke skal vi fortsette med full kraft å jobbe med der vi slapp fra forrige uke og forhåpentligvis ser vi fremgang mot noen av A kravene som vi har.

F.5 Oppfølgningsdokument uke 9

Vi har nå endelig kommet i gang med å jobbe med oppgaven og jobbe mot kravene som er satt. Data studentene har begynt å utforske bildebehandling og begynt å programmere litt for å se hva vi kan få til med dette. Maskin har begynt å tegne litt og tenke på hvordan dette fysiske systemet kan se ut.

Senay

- Få nettsiden i gang på skolen sin server
- Utforske hvilke programmeringsspråk som egner seg best til vår oppgave
- Teste ut ulike bildebehandlingsalgoritmer relevant til oppgaven

Jehad

- Mer grundig undersøking av forskjellen mellom system og telefon kamera
- Fortsettelse på å utforske bildebehandling algoritme

Lars Erik

- Konseptuell design av referansepunkt
- 3D-modellering av oppheng til kamera

Ole Markus

- Konseptuell design av referansepunkt.
- Lage UML diagrammer som beskriver den generelle adferden til systemet.

Kristoffer

- Gå dypere ned i hva bildebehandling er og hvordan vi kan bruke det til vår oppgave
- Utforsket openCv
- Utforsket PIL
- Prøvd meg frem med å programmere noen løsninger for å detektere punkter på et bilde
- Laget diagrammer for å visualisere hvordan systemet vårt ser ut

Samuel

- Funnet flere slags biblioteker vi kan bruke for å løse oppgaven
- Vist fram de bibliotekene så vi kan jobbe som gruppe
- Undersøkelse av biblioteker som kan hjelpe oss å løse oppgaven
- Fikset flere slags problemer med git for gruppa.
- Hjulpet gruppa med forskjellige git kommandoer og hva de må passe på når de bruker git
- Lagt inn flere tasks i jira og bekreftet tilværelsen av jira til nå
- Revisjon av åssen det har gått og kommet fram til at vi skal gi skriftlige beskjeder på saker vi har kommet til enighet om

Ole Markus

- Konseptuell design av referansepunkt.
- Lage UML diagrammer som beskriver den generelle adferden til systemet.

Timeliste

Timeliste	Uke 9	Totalt
Senay	28	192
Ole Markus	26	185
Lars Erik	25	183
Jehad	18	175
Samuel	30	211
Kristoffer	28	181
Sum	155 timer	1127 timer

Neste uke: Neste uke kommer vi til å jobbe en dag mindre enn det vi har gjort de siste ukene. Vi har eksamen bare noen dager rett etter vår 2. Presentasjon så vi må prioritere dette også. Vi kommer til å fortsette arbeide og programmere disse forskjellige verktøyene som blir brukt i bildebehandling

F.6 Oppfølgningsdokument uke 10

Vi har nå endelig kommet i gang med å jobbe med oppgaven og jobbe mot kravene som er satt. Data studentene har begynt å utforske bildebehandling og begynt å programmere litt for å se hva vi kan få til med dette. Maskin har begynt å tegne litt og tenke på hvordan dette fysiske systemet kan se ut.

Senay:

- Videre arbeid med bildebehandlings algoritme for deteksjon av flere figurer.
- Forbedre nøyaktigheten av senterpunktet bildebehandlings algoritmen finner på figur.

Jehad:

- Videre utforsking av å finne markinger på et bilde ved bruk av piksel farger
- Undersøke hva som passer vår løsning best når det kommer til telefon eller system kamera

Lars Erik:

- Jobbe med tilbakemeldingene
- Utforske konsepter til løsninger
- Prosjektplanlegging

Ole Markus:

- Jobbe med tilbakemeldingene
- Utforske konsepter

- Jobbe med dokumentasjonen

Kristoffer:

- Gjort klart alt rundt 2. presentasjon med å booke rom, avtale tid osv.
- Jobbet med openCv
- Prøvd meg frem med å programmere to forskjellige måter til å detektere punkter på et bilde.

Samuel:

- Utforsking av hvordan man lager executable files
- Utforsking av forskjellige løsninger av bildebehandling
- Reiterasjon av testplan etter tilbakemeldinger fra første presentasjon
- Utforsking av forskjellige kamerae oppløsninger
- Utforsking av forskjellige mobile kameraoppløsninger

Timeliste uke 10

Timeliste	Uke 10	Totalt
Senay	18	210
Ole Markus	21	206
Lars Erik	20	203
Jehad	20	195
Samuel	35	246
Kristoffer	20	201
Sum	134 timer	1261 timer

Neste uke: Neste uke kommer majoriteten av vår tid til å fikse dokumentasjonen og klargjøre den til den skal leveres fredag morgen. Vi kommer også til å begynne på presentasjonen slik at den er klar til tirsdagen uken etter.

F.7 Oppfølgningsdokument uke 11

Denne uken har vi jobbet med å klargjøre dokumentasjonen som skal leveres på morningen på fredag. Gruppen har ikke gjort noe annet i forbindelse med dette da det krever stort fokus å få den ferdigstilt.

Timeliste uke 11

Timeliste	Uke 11	Totalt
Senay	15	225
Ole Markus	20	226
Lars Erik	21	224
Jehad	24,5	219,5
Samuel	24	270
Kristoffer	22	223
Sum	126,5 timer	1387,5 timer

F.8 Oppfølgningsdokument uke 12

Denne uken har vi hatt presentasjon. Dette var på tirsdagen. Det er også eksamensuke så denne uken er det ikke gjort noe videre arbeid med prosjektet. Vi starter opp neste uke igjen. Da er det fullt fokus på bacheloren da vi er ferdig med alt annet.

Timeliste uke 12

Timeliste	Uke 12	Totalt
Senay	4	210
Ole Markus	8	206
Lars Erik	5	203
Jehad	8	195
Samuel	8	246
Kristoffer	8	201
Sum	41 timer	1428,5 timer

F.9 Oppfølgningsdokument uke 13

Denne uken er første uken etter eksamen, så nå er det fullt fokuse på oppgaven. Vi vil fremover fra dette punktet øke antall timer. Dette vil resultere at vi vil bevege oss fremover i et raskere tempo enn vi har gjort til nå. Denne uken kommer vi først og fremst til å jobbe med tilbakemeldinger som vi fikk på 2. presentsasjon. I tillegg til det skal vi fortsette med det tekniske der vi slapp sist gang.

Senay:

- Jobbed med dokumentasjon av algoritme for referansepunkt.
- Utforsket andre referansepunktalgoritmer
- Testet flere algoritmer for referansepunkt.

Jehad:

- Utforske løsningsmetoder for å finne referansepunkt.
- Finne referansepunkt ved bruk av ChatGPT og dokumentere
- Starte med å finne mer presisjon til å finne treffpunkter

Lars Erik:

- Designe konsepter for markører til treffpunkter
- Omformulere teksten om forholdet mellom oppløsning og nøyaktighet
- Undersøke forholdet mellom oppløsning og nøyaktighet

Ole Markus:

- Fullføre user stories

- Undersøke systemdesign for å finne hvordan systemet bør utledes fra kravene, og hvordan det kan illustreres.
- Dokumentere systemdesign - utlede systemarkitektur fra kravene.

Kristoffer:

- Fikset på hull i dokumentasjonen
- Fikset småting i dokumentasjonen som vi fikk tilbakemeldinger på. Dette er skrivefeil og ting som er plassert riktig.
- Startet med utkast av design til app
- Undersøkt språk og hvordan vi skal lage appen
- Startet med å lage appen

Samuel:

- Finne den reelle avstanden basert på piksler mellom akse korsene. Finne piksel høyde og bredde ut ifra X og Y aksekors. Forsøk på å finne den reelle avstanden mellom skudd med Pytagoras. age klasse hierarki for piksel egenskaper, også videre for andre moduler.

Timeliste	Uke 13	Totalt
Senay	29	262
Ole Markus	36	269
Lars Erik	32	261
Jehad	31,5	259
Samuel	56	334
Kristoffer	32	263
Sum	180,5 timer	1648 timer

Neste uke: Neste uke er det påske. Vi blir ikke å jobbe sammen på skolen fordi det er utfordringer med kollektivtransporten som medfører at enkelte medlemmer ikke kommer seg til skolen. Planen er at alle skal jobbe noe, men alle har noe planer så vi får se hvordan arbeidet blir. Alle i gruppen er klar over hva man skal gjøre i påsken. Det er planlagt å ha noen møter over teams noen av dagene i påsken hvor vi oppdaterer hverandre på hvordan det går.

F.10 Oppfølgningsdokument uke 14

Senay:

- Jobbing med doxygen dokumentasjon av kode
- Ferdigstilling av dokumentasjon med algoritme
- Lage ferdig funksjonalitet for å ta bilde og lagre bilde i app
- Få til tilgang til mappe med bilder i app

Jehad:

- Finne mer presisjon til treffpunkter ved bruk av median til detektert piksler i et treffpunkt. Dokumentert også.
- Finne mer presisjon til treffpunkter ved bruk av senter piksel i et tegnet rektangel rundt skytehullene.

Lars Erik:

- Designe og dokumentere 4 konsepter for markering av treffpunkt.

Ole Markus:

- Omskrive kapittelet om "Prosjektmodell" i dokumentasjonen.

Kristoffer:

- Jobbet med å finne ut hvordan man integrerer Python i Android studio. Dette førte ikke noe steder

Samuel:

- Klassekommunikasjons diagram fra a til å. Markere treffpunkter i forhold til KTS koordinatssystem. Pytagoras løsning i kombinasjon med vektorer for å finne den
- reelle avstanden fra senter av figuren og til hvert skudd.
- Algoritme ide for å få finne nøyaktig senter av markeringsobjekter.

Kristoffer:

- Jobbet med å finne ut hvordan man integrerer Python i Android studio. Dette førte ikke noe steder

Timeliste	Uke 14	Totalt
Senay	24	286
Ole Markus	16	285
Lars Erik	5,5	266,5
Jehad	19,5	278,5
Samuel	56	390
Kristoffer	19	282
Sum	140 timer	1788 timer

F.11 Oppfølgningsdokument uke 15

Senay:

- Android studio
- Funksjonaliteter i appen
- Behandling av bilde
- Filutforsker

Jehad:

- Teste forskjellige fysiske markører i henhold til å finne treffpunkt algoritme. Se hvordan det funker sammen. Kvadratiske og sirkulære i forskjellige størrelse.
- Se på andre metoder for å finne referansepunkt

Lars Erik:

- Dokumentere konseptene om treffpunkt markører
- Finpusse design av oppheng til kamera
- Vurdere bruk av system kamera eller telefon kamera
- Redesign av markører

Ole Markus:

- Rettskrive og omskrive dokument for prosjektmodell
- Lage referansemarkør til referansesystem.
- Begrunne valg av referansesystem i dokumentasjonen

Kristoffer:

- Jobbet med Android studio
- Utforsket andre verktøy for utvikling av app
- Sett på integrasjon av forskjellige deler
- Tatt det vanskelige valget om å gå bort fra Android studio
- Sett på andre løsninger for å utvikle appen.
- Sett på React Native og Xamarin

Samuel:

- Testet ut nøyaktigheten til algoritmen når bilde er tatt vinkelrett.
- Testet ut hvor nøyaktig avstanden mellom origo og et treffpunkt blir med 108 mp og når bilde er tatt vinkelrett.
- Sjekket ut informasjon om hvor nøyaktig mobil gyroskop er.

Timeliste	Uke 15	Totalt
Senay	33	319
Ole Markus	33	318
Lars Erik	29,5	296
Jehad	25	303,5
Samuel	40	430
Kristoffer	31	313
Sum	191,5 timer	1979,5 timer

F.12 Oppfølgningsdokument uke 16

Senay:

- Jobbet med react native
- Implementere funksjoner
- Kamera og behandling av bilder
- Design av app
- Lage flere sider

Jehad:

- Teste nøyaktighet til treffpunkt deteksjon basert på flere markører.
- Forskjellige posisjonert markører
- Forskjellige vinkler
- Forskjellige oppløsninger
- Oppdaget problem med deteksjon av treffpunkter, prøver å rett opp på det

Lars Erik:

- Justering av mål på oppheng av kamera
- Undersøke/ dokumentere teorien bak design av markører.
- Designe bein og skinnegang for kameraet
- Teste nøyaktigheten av treffpunkt
- Begrunnelse av mekanisk system

Ole Markus:

- Designe referansemarkører til testing.
- Produksjon av prototyper til referansesystem og treffpunktsystem.
- Testing av prototyper til punktmarkører.
- Systemdesign til referansepunktsystem.
- Strukturere dokumentasjonen slik at den er mer sammenhengende.
- Ordne referanser i dokumentasjonen.
- Dokumentere valg av programvaremetode – pikseltelling.

Kristoffer:

- Jobbet med å koble sammen frontend og backend.
- Lært om API
- Undersøkt forskjellige metoder for å løse API

Samuel:

- Testet ut nøyaktigheten til algoritmen når bilde er tatt vinkelrett med forskjellige kamera oppløsninger som 108 mp
- Testet nøyaktigheten til algoritmen med bilde tatt fra forskjellige vinkler og sett om avviket blir stort.
- Matematisk formel for å kunne ta hensyn til avvik.
- Testet ut nøyaktighet med håndholdt bildetagning.
- Ferdigstilt nettsiden

Timeliste	Uke 16	Totalt
Senay	29	348
Ole Markus	39	357
Lars Erik	37,7	333,5
Jehad	31	334,5
Samuel	56	486
Kristoffer	36	349
Sum	228,5 timer	2208 timer

F.13 Oppfølgningsdokument uke 17

Senay:

- Fikse størrelsesforhold problem i kamera funksjon til appen
- UX flow chart design av appen
- Videre jobb med design i appen
- Muligens brukerautentisering i appen

Jehad:

- Oppdatere deteksjon algoritme ved ny metode for å gruppere piksler på. Gruppering via avstand fra første detektert piksel i et treffpunkt. Gamle versjon var sammenligning av to piksler og avstanden mellom dem.
- Oppdatere dokumentasjon om deteksjon algoritme
- Oppdatere hastighet på deteksjon av markører fra 80 sekunder til 10 sekunder
- Lage en metode for å tilpasse å finne markører på i henhold til forskjellige avstander

Lars Erik:

- Begrunnelse for et mekanisk system
- Design av toget som skal bevege seg på skinnene
- Produksjon av konsepter til treffpunktmarkører
- Produksjon av referansemarkører
- Teste nøyaktigheten til treffpunkt fra ulike distanser

- Lære å legge inn dokumentasjon i overleaf

Ole Markus:

- Design av referanseverktøy
- Dokumentasjon av referansesystem
- Dokumentere oppgavebeskrivelsen tydeligere
- Dokumentere design av treffpunktsystem
- Teste nøyaktighet til treffpunkt fra ulike avstander

Kristoffer:

- Dokumentere om integrasjon
- Lage en kobling mellom frontend og backend
- Prøvd å sende bilde fra app til webserver

Samuel:

- Teste mange ulike distanser
- Teste ulike oppløsninger
- Teste flere mobile kameraer
- Automatisert test software for å teste at man får forventede resultater av forskjellige treffpunkt inputs.
- Finne ut hvor mye barrel distortation påvirker nøyaktigheten og hvordan det kan unngås

Timeliste	Uke 17	Totalt
Senay	37	385
Ole Markus	41	398
Lars Erik	37	370.5
Jehad	35	369.5
Samuel	40	526
Kristoffer	35	384
Sum	225 timer	2433 timer

F.14 Oppfølgningsdokument uke 18

Senay:

- Designe overlay komponent i appen
- Lage gyroskop funksjonalitet i appen

Jehad:

- Endre deteksjonsalgoritmen til objektorientert
- Deteksjons algoritme klarer å finne aksekors markører og gi oss en senter verdi av markørene.
- Teste forskjellige markører farger som mørkeblå, lyseblå og grønn
- Integrere fullstendig backend programvare

Lars Erik:

- Produksjon av referansemarkører
- Design av toget som skal bevege seg langs skinnene
- Innkjøp av mørkere farger av spraymaling
- Innkjøp av aksekorspapir til testing av nøyaktighet ved deteksjon av treffpunkter
- Måle manuelt x og y-verdier av treffpunkter
- Modifikasjoner av referanseverktøy slik at de passer alle typer størrelser for skyteblink.

Ole Markus:

- Var syk hele uken

Kristoffer:

- Jobbet med å få webserver til å motta bilde fra app
- Lage resultatside til appen
- Lagt til på app design

Samuel:

- Manuelle målinger
- Teste nøyaktigheten med mange skudd
- Teste ut
- Sjekket ut om det finnes noen sammenheng i avvikene
- Satt opp excel for resultat data

Timeliste	Uke 18	Totalt
Senay	35.5	420.5
Ole Markus	7	405
Lars Erik	38	408.5
Jehad	51	420.5
Samuel	41	567
Kristoffer	35	419
Sum	207.5 timer	2604.5 timer

F.15 Oppfølgningsdokument uke 19

Senay:

- Forbedre nøyaktighet av gyroskop målingene
- Implementere mer funksjonalitet til kamera skjermen
- Finjustere ting i appen(Overlays og design)

Jehad:

- Finpusse og kommentere deteksjonsalgoritmen
- Teste nøyaktighet på å finne markører og se om de er innen 2mm avvik
- Skape en større hastighet på å se gjennom piksler og tilpasse avstand
- Tilpasse deteksjonsalgoritmen til flere oppløsninger

Lars Erik:

- Design av skinnegang og tog
- Måle og teste nøyaktigheten til treffpunkter
- Produsere treffmarkører og referansemarkører
- Testing av deteksjon av treffpunkter i sollys

Ole Markus:

- Var syk hele uken

Kristoffer:

- Smått begynne å planlegge sluttrapport og hva vi skal ha med

- Jobbet videre med å få til koblingen mellom frontend og backend.
- Dokumentere om integrasjonen

Samuel:

- Dokumentere om integrasjonen
- Teste nøyaktighet
- Legge til funksjon i algoritmen for å kunne skille mellom hvilket aksekors man skal ut utgangs punkt fra
- Omgjøre algoritmen for at den skal ta utgangspunkt fra origo og ut til sidene. Slik at sirkelen blir delt i 4 deler.
- Utforskning av formler som kan ta hensyn til grader avvik i bildene
- Dokumentere «Barrel distortation, pincushion distortation

Timeliste	Uke 19	Totalt
Senay	56	476.5
Ole Markus	44	449
Lars Erik	45	453.5
Jehad	52	472.5
Samuel	41	608
Kristoffer	44	463
Sum	282 timer	2922.5 timer

F.16 Oppfølgningsdokument uke 20

Senay:

- Dokumentasjon av appen sin virkemåte
- Forberedelse av doxygen for kode dokumentasjon
- Justering av program og finjusting
- Dokumentasjon av regnskap
- Legge inn det som skal være i latex
- Legge inn gjenstående av det som skal inn i latex
- Finpusse dokumentasjon

Jehad:

- Dokumentasjon av RGB-verdier
- Dokumentasjon av deteksjonsalgoritme
- Dokumentasjon av flere løsningsmetoder for deteksjonsalgoritme. Kommentere og finpusse essensiel kode.

Lars Erik:

- Manuelle målinger for testing av nøyaktighet
- Design av skinnegang
- Dokumentasjon av skinnegang
- Dokumentasjon av treffpunktmarkører
- Dokumentasjon av oppheng til kamera

- 2D-tegninger av alle 3D-modulerte deler og assembelys
- Brukerinstruksjon
- Dokumentasjon av kriterier for system kamera eller mobilkamera

Ole Markus:

- Gjennomgang av dokumentasjon i hovedrapport
- Dokumentasjon av referansepunktsystem
- Utarbeide brukerinstruks
- Planlegge og sette opp dokumentstruktur i hovedrapport
- Produksjon av treffpunktmarkører
- Omformulering og rettskrivning av tekster i hovedrapport
- Utført kontrollmåling av treffpunkter til testing
- Dokumentert testmetode for avviksanalyser i hovedrapport

Kristoffer:

- Gjort klart og lagt over alt som skal på minnepennen
- Skrevet forord
- Gjort ferdig dokumentasjon på app del
- Lagt inn alt av oppfølgningsdokumenter.

Samuel:

- Mobil kamera vs digital kamera
- Referansepunktsystem(Programvare logikk)

- Avviksanalyse
- Algoritme data
- Extracta data
- Svakheter med algoritmen
- Resultater

Timeliste	Uke 20	Totalt
Senay	76	552.5
Ole Markus	78	527
Lars Erik	70	523.5
Jehad	73	545.5
Samuel	105	713
Kristoffer	68	531
Sum	470 timer	3392.5 timer

G Brukerinstruks - ImPro kalibreringssystem

G.1 Bruk

1. IMPRO kalibreringssystem skal brukes til å kalibrere en elektronisk målskive fra Kongsberg Target systems.
2. Systemet kan brukes på enhver målskive med sirkulær sikteblink.

G.2 Før kalibrering

1. Skyting av kalibreringsskudd skal foregå før kalibreringsprosessen.
2. Ved skyting skal inntil 3 skudd plasseres i området ved midten av skiven.
3. Skudd som skytes i blinken skal ikke være inntil eller berøre hverandre.
4. Posisjonen i x- og y-koordinater til de elektroniske treffpunktene noteres ned for senere bruk.

G.3 Aksekorsmarkering

1. Verktøyet illustrert i figuren skal benyttes ved plasseringen av aksekorsmarkører.
2. Aksekorsene skal markeres med aksekorsmarkørene. Se figuren for aksekorsmarkører.
3. Plasser verktøyet på kanten av den svarte sirkelen og det hvite planet slik som beskrevet i figuren.
4. Beveg verktøyet langs sirkelperiferien til en linje av aksekorset er på kanten til verktøyet.

G.4 Treffpunktmarkering

1. De fysiske treffmarkørene plasseres inn i hvert sitt skuddhull.
2. Sett spissen av markøren i skuddhullet og trykk markøren inn slik at hodet til markøren er inntil blinken. Ved mye motstand, skru markøren samtidig som den trykkes inn.
3. Treffmarkører skal ikke plasseres inntil hverandre. Dersom skuddhullene er for nærme hverandre må skuddene tas på nytt.

G.5 Bildetagning

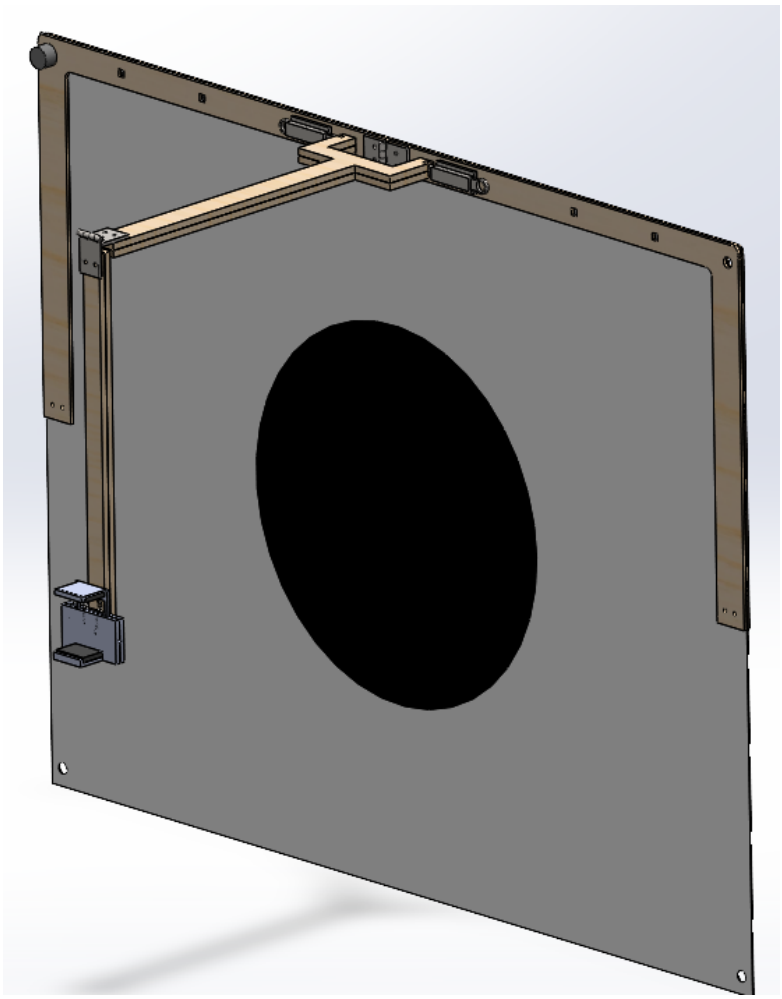
1. Logg inn med brukernavn og passord i kalibreringsappen.
2. Nederst i menyen trykk på figuren av kameraet for bildetagning.
3. Hold sirkelen på skyteskiven innenfor den blå sirkelen i kameraet.
4. Hold "Yaw"innenfor -5 og 5. Da vil kameraet være tilstrekkelig i vater.
5. Hold Pitch"innenfor 85 og 95. Da vil kameraet være stilt rett på målskiven.
6. Korset innenfor den blå sirkelen vil signalisere at bilde kan tas ved at fargen til korset skifter fra rødt til grønt.
7. Trykk på utløseren for å ta bilde av målskiven.
8. x- og y-koordinater til treffpunktene vil dukke opp etter kort tid.

H Oppheng av kamera

For at nøyaktigheten skal bli best mulig kreves det at billedtagningen blir tatt 90 grader på skyteskiven, og fra en spesifikk avstand fra skyteskiven. Derfor har vi 3D-modellert et opphengsystem til kameraet, som skal settes på skyteskiven ved billedtagning. Dette fører til at man luker bort evt menneskelige unøyaktighet som f.eks. vinkling, avstand og posisjonering av kamera i forhold til skyteskive. Selv om man har laget en app med gyroskop, og en sirkel som skal hjelpe deg med å ta hensyn til disse faktorene er det fortsatt vanskelig å holde kameraet 100

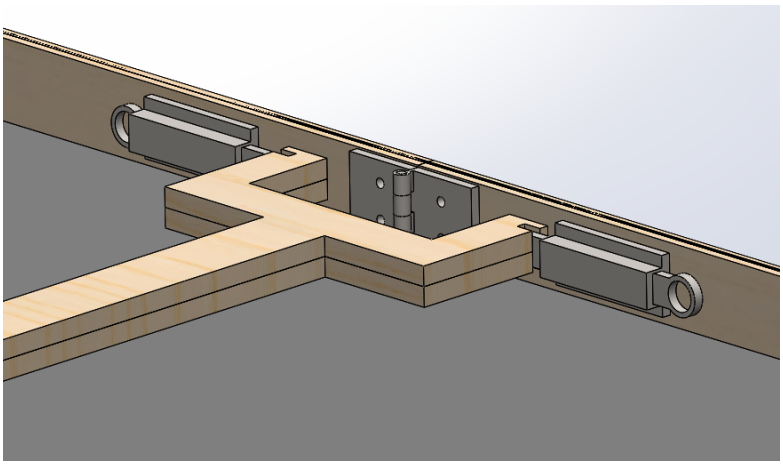
Materialer tiltenkt brukt til produksjon er kryssfiner og tough PLA. Produksjonsmetoden vil derfor være laserkutter og 3D-printing. Produkter som hengsle,låsemekanisme og fjær er ferdig innkjøpte deler av systemet.

På bilde ser man hele opphengsystemet, og hvordan den henges på i forhold til skyteskiven. På skyteskiven er det fire tapper i hvert hjørne som skyteskiven henges på. Disse tappene kan man også bruke til å henge opp opphengsystemet.



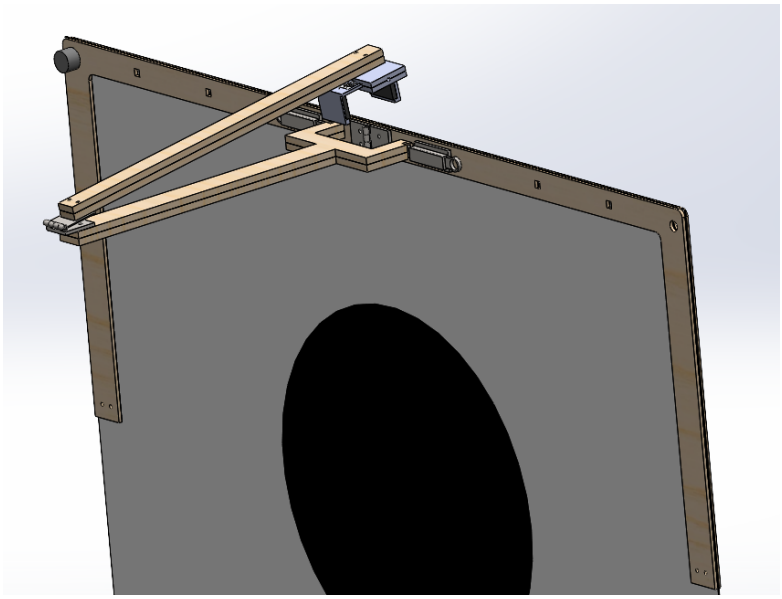
Figur 142: Annretningen festet på skiven

Her ser man hvordan armen som går utfra skyteskiven på bilde ovenfor er festet til rammen til opphengsystemet. Vi har to låsemekanismer på hver side som holder på plass armen. Disse låsene gjør det lett å montere og demontere systemet. I midten har vi ett hengsle, slik at rammen til opphengsystemet er sammenleggbar.



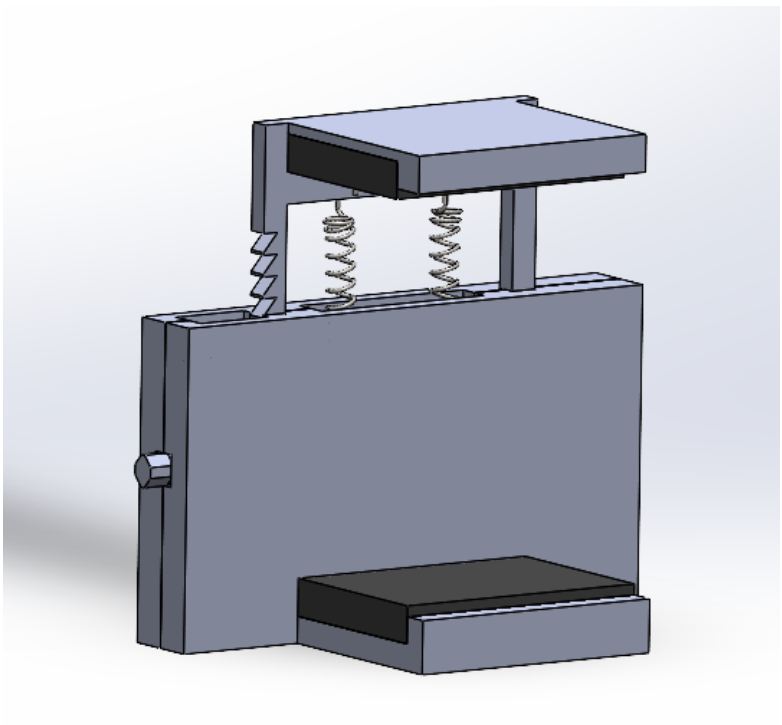
Figur 143: Rammen

På dette bilde ser man hvordan armen ut fra skyteskiven er to delt med et hengsle som fester de to delene fra en horisontal til en vertikal retning. Dette gjør at armen er lett å ta med seg og tar liten plass under forflytning.



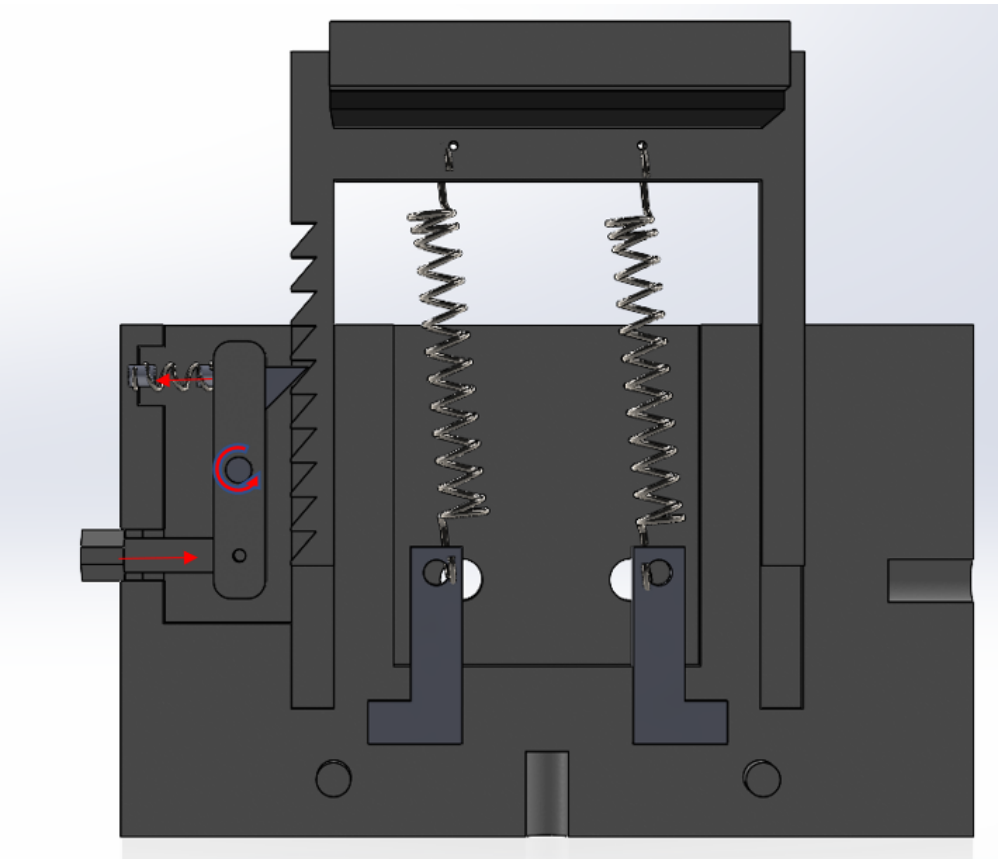
Figur 144: Sammenleggbaringen

På dette bilde ser man kameraholderen som plasseres i enden på armen på opphengsystemet, som er sentrert i midten av skyteskiven. Denne holderen er justerbar i vertikal retning slik at den skal kunne passe ulike størrelser på telefonen. Kameraholderen er designet for å ta bilde horisontal.



Figur 145: Kameraholderen

På innsiden av kameraholderen ser man hvordan man kan justere bredden på disse klemmene etter størrelsen på telefonen. Mekanismen funker ved at man presser inn en knapp på venstre side av bilde. Dette vil føre til at en aksling med en tann på begynner å rotere, slik at den presser inn en trykkfjær. Dette vil føre til at tannen går ut av hakket. Når kraften på knappen forsvinner vil kraften fra trykkfjæren danne en motkraft som presser tanna tilbake i hakket.



Figur 146: Innsiden av kameraholderen. De røde påilene skal illustrere kraftretning/ bevegelsesretningen ved påført trykk

I Kode

I.1 Kamera kode

```
import Home from './screens/Home';
import Camera from './screens/Cameraa';
import Gallery from './screens/Gallery';
import Login from './screens/login';
import Register from './screens/register';

import { NavigationContainer } from '@react-navigation/native';
import { createNativeStackNavigator } from '@react-navigation/native-stack';

const Stack = createNativeStackNavigator();
```

Figur 147: Kode for Kamera

```
export default function App() {
  return (
    <NavigationContainer>
      <Stack.Navigator>
        <Stack.Screen
          name="Login"
          options={{headerShown: false}}
          component={Login}
          />
          <Stack.Screen
          name="Register"
          options={{headerShown: false}}
          component={Register}
          />
        <Stack.Screen
          name="Home"
          options={{headerShown: false}}
          component={Home}
          />
        <Stack.Screen
          name="Camera"
          options={{headerShown: false}}
          component={Camera}
          />
        <Stack.Screen
          name="Gallery"
          options={{headerShown: false}}
          component={Gallery}
          />
      </Stack.Navigator>
    </NavigationContainer>
  );
}
```

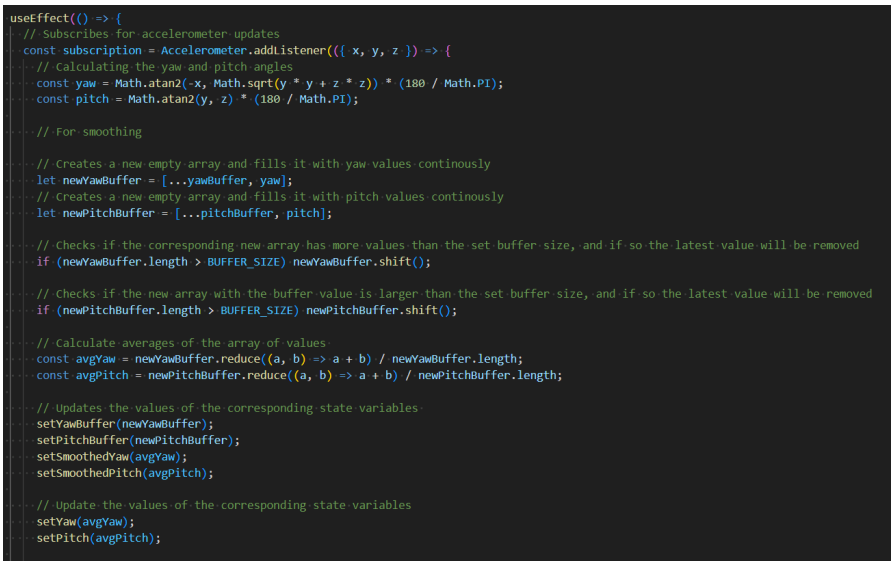
Figur 148: Kode for Kamera

```
import { useEffect, useRef, useState } from 'react';
import { StatusBar } from 'expo-status-bar';
import { StyleSheet, Text, View, Image, TouchableOpacity } from 'react-native';
import { Camera } from 'expo-camera';
import { shareAsync } from 'expo-sharing';
import * as MediaLibrary from 'expo-media-library';
import { SafeAreaView } from 'react-native-safe-area-context';
import { Svg, Circle } from 'react-native-svg';
import { Accelerometer } from 'expo-sensors';
import Icon from 'react-native-vector-icons/FontAwesome';

export default function Cameraa() {
  // State variables
  let cameraRef = useRef();
  const [hasCameraPermission, setHasCameraPermission] = useState();
  const [hasMediaLibraryPermission, setHasMediaLibraryPermission] = useState();
  const [photo, setPhoto] = useState();
  const [firstCircleRef, setFirstCircleRef] = useState(null);
  const [isAt90Degrees, setIsAt90Degrees] = useState(false);
  const [yaw, setYaw] = useState(0);
  const [pitch, setPitch] = useState(0);
  const [smootedYaw, setSmoothedYaw] = useState(0);
  const [smoothedPitch, setSmoothedPitch] = useState(0);
  const [yawBuffer, setYawBuffer] = useState([]);
  const [pitchBuffer, setPitchBuffer] = useState([]);

  const BUFFER_SIZE = 8; // Use last 8 readings for smoothing the angle value
```

Figur 149: Kode for Kamera

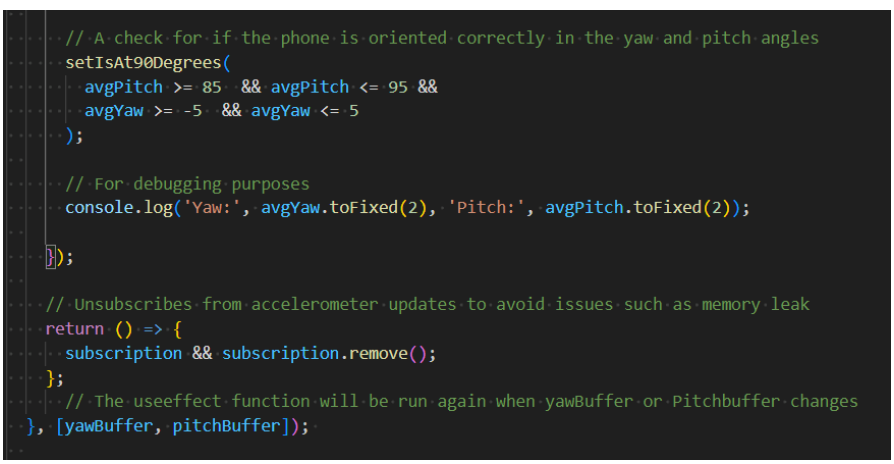
```
// Hook to ask user for different permissions needed and update the state variables belonging
useEffect(() => {
  (async () => {
    const cameraPermission = await Camera.requestCameraPermissionsAsync();
    const mediaLibraryPermission = await MediaLibrary.requestPermissionsAsync();
    setHasCameraPermission(cameraPermission.status === "granted");
    setHasMediaLibraryPermission(mediaLibraryPermission.status === "granted");
  })();
}, []);
```

Figur 150: Kode for Kamera



```
useEffect(() => {
  // Subscribes for accelerometer updates
  const subscription = Accelerometer.addListener(({ x, y, z }) => {
    // Calculating the yaw and pitch angles
    const yaw = Math.atan2(-x, Math.sqrt(y * y + z * z)) * (180 / Math.PI);
    const pitch = Math.atan2(y, z) * (180 / Math.PI);

    // For smoothing

    // Creates a new empty array and fills it with yaw values continously
    let newYawBuffer = [...yawBuffer, yaw];
    // Creates a new empty array and fills it with pitch values continously
    let newPitchBuffer = [...pitchBuffer, pitch];

    // Checks if the corresponding new array has more values than the set buffer size, and if so the latest value will be removed
    if (newYawBuffer.length > BUFFER_SIZE) newYawBuffer.shift();

    // Checks if the new array with the buffer value is larger than the set buffer size, and if so the latest value will be removed
    if (newPitchBuffer.length > BUFFER_SIZE) newPitchBuffer.shift();

    // Calculate averages of the array of values
    const avgYaw = newYawBuffer.reduce((a, b) => a + b) / newYawBuffer.length;
    const avgPitch = newPitchBuffer.reduce((a, b) => a + b) / newPitchBuffer.length;

    // Updates the values of the corresponding state variables
    setYawBuffer(newYawBuffer);
    setPitchBuffer(newPitchBuffer);
    setSmoothedYaw(avgYaw);
    setSmoothedPitch(avgPitch);

    // Update the values of the corresponding state variables
    setYaw(avgYaw);
    setPitch(avgPitch);
```

Figur 151: Kode for Kamera



```
    // A check for if the phone is oriented correctly in the yaw and pitch angles
    setIsAt90Degrees(
      avgPitch >= 85 && avgPitch <= 95 &&
      avgYaw >= -5 && avgYaw <= 5
    );

    // For debugging purposes
    console.log('Yaw:', avgYaw.toFixed(2), 'Pitch:', avgPitch.toFixed(2));

  });

  // Unsubscribes from accelerometer updates to avoid issues such as memory leak
  return () => {
    subscription && subscription.remove();
  };
    // The useeffect function will be run again when yawBuffer or Pitchbuffer changes
}, [yawBuffer, pitchBuffer]);
```

Figur 152: Kode for Kamera

```
// If the requirement for the state variable is satisfied then change the line colors to green, otherwise stay red
const lineStyles = {
  horizontalLine: {
    backgroundColor: isAt90Degrees ? 'green' : 'red',
  },
  verticalLine: {
    backgroundColor: isAt90Degrees ? 'green' : 'red',
  },
};

// Function with configurations of options for picture taken
let takePic = async () => {
  let options = {
    quality: 1,
    base64: true,
    exif: false,
  };

  // Picture gets captured and the state variable setPhoto is updated
  let newPhoto = await cameraRef.current.takePictureAsync(options);
  setPhoto(newPhoto);
};

// Share the picture taken
if (photo) {
  let sharePic = () => {
    shareAsync(photo.uri).then(() => {
      setPhoto(undefined);
    });
  };
```

Figur 153: Kode for Kamera

```
  // Saves the picture taken inside the local gallery in own folder, voluntarary folder
  let savePhoto = async () => {
    if (hasMediaLibraryPermission) {
      const asset = await MediaLibrary.createAssetAsync(photo.uri);
      await MediaLibrary.createAlbumAsync('IMG', asset, false);
      setPhoto(undefined);
    }
  };

  // Componenets to be rendered on the screen
  return (
    <SafeAreaView style={styles.container}>
      <Image style={styles.preview} source={{ uri: "" + photo.base64 }} />
      <View style={styles.btnContainer2}>
        <TouchableOpacity
          onPress={sharePic}>
          <Icon name="share" size={50} color="grey" />
        </TouchableOpacity>

        {hasMediaLibraryPermission &&
          <TouchableOpacity onPress={savePhoto}>
            <Icon name="download" size={50} color="gray" />
          </TouchableOpacity>
        }
        <TouchableOpacity onPress={() => setPhoto(undefined)}>
          <Icon name="trash" size={50} color={"grey"} />
        </TouchableOpacity>
      </View>
    </SafeAreaView>
  );
}
```

Figur 154: Kode for Kamera

```
// Componenets to be rendered on the screen
return (
  <SafeAreaView style={styles.container}>
    <Camera
      style={{ aspectRatio: '3/4' }}
      ref={cameraRef}
    >
      <View style={styles.circlContainer}>
        <Svg height="80%" width="80%">
          <Circle
            cx="50%"
            cy="50%"
            r="60"
            fill="none"
            stroke="blue"
            strokeWidth="4"
            ref={setFirstCircleRef}
          />
        </Svg>
        <View style={[styles.line, styles.horizontalLine, lineStyles.horizontalLine]} />
        <View style={[styles.line, styles.verticalLine, lineStyles.verticalLine]} />
      </View>
      <View style={styles.overlayContainer}>
        <View style={styles.overlayTextContainer}>
          <Text style={styles.overlayText}>
            Yaw: {yaw.toFixed(2)}°
          </Text>
          <Text style={styles.overlayText}>
            Pitch: {pitch.toFixed(2)}°
          </Text>
        </View>
      </View>
```

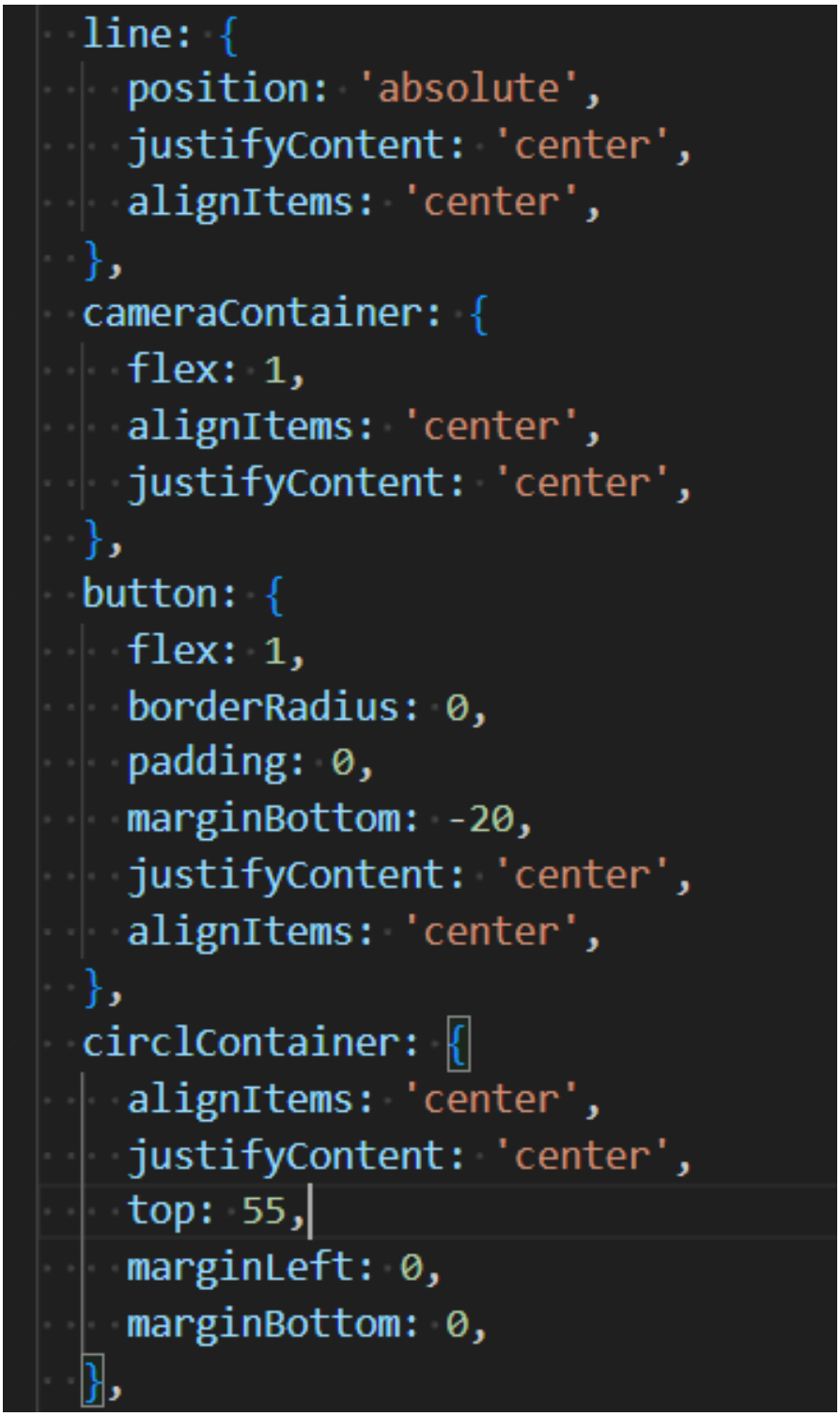
Figur 155: Kode for Kamera

```
        <View style={styles.btnContainer}>
          <TouchableOpacity
            style={styles.button}
            onPress={takePic}>
            <Icon name="camera" size={50} color="white" />
          </TouchableOpacity>
        </View>
        <StatusBar style="auto" />
      </Camera>
    </SafeAreaView>
  );
}
```

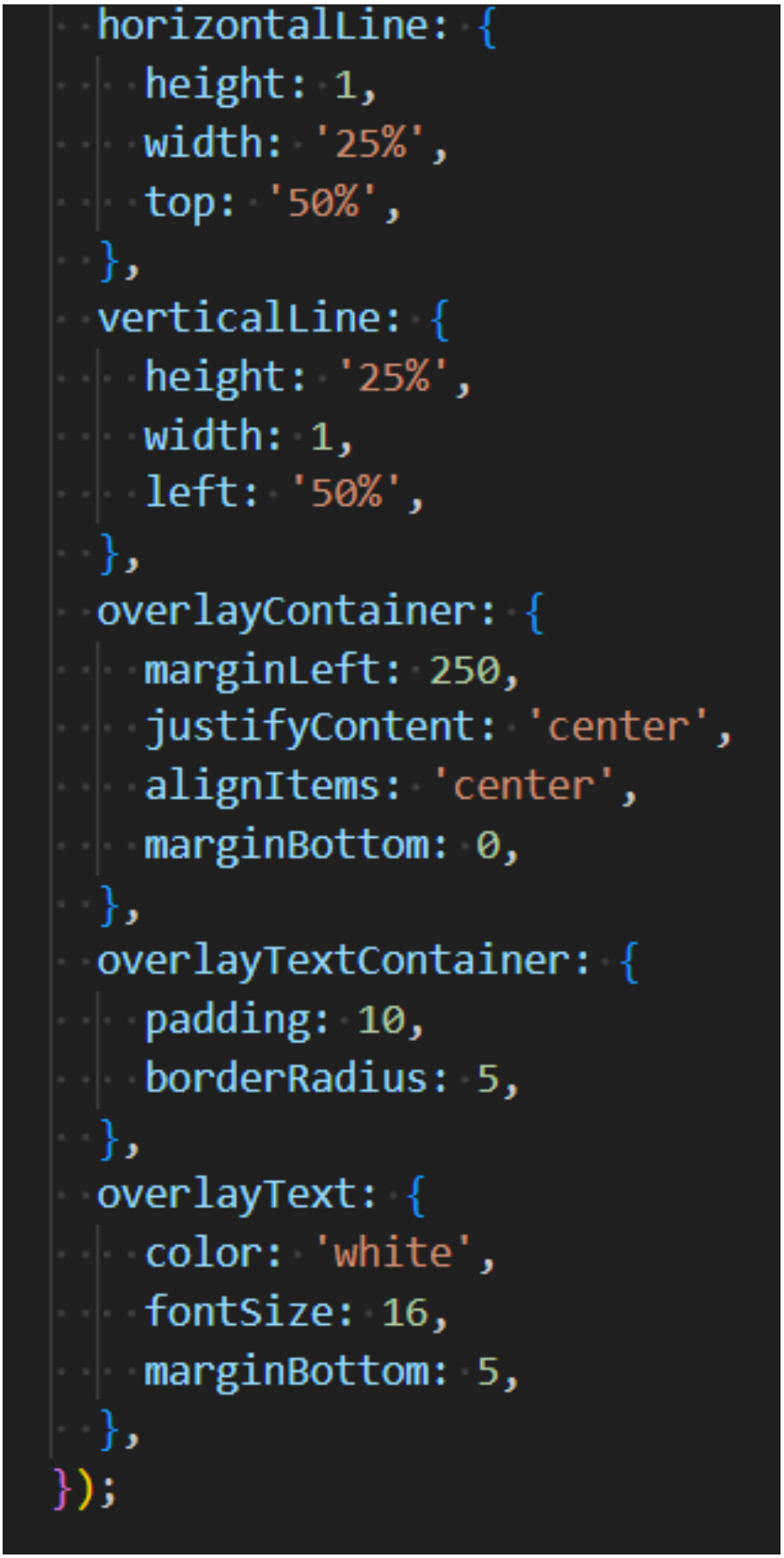
Figur 156: Kode for Kamera

```
// Styling of all the rendered components
const styles = StyleSheet.create({
  container: {
    flex: 1,
    alignItems: 'center',
    justifyContent: 'center',
  },
  buttonContainer: {
    justifyContent: 'center',
    alignItems: 'center',
    marginBottom: 200,
  },
  btnContainer: {
    flex: 1,
    alignItems: 'center',
    justifyContent: 'center',
    marginTop: 0,
  },
  btnContainer2: {
    borderRadius: 10,
    padding: 5,
    flexDirection: 'row'
  },
  preview: {
    alignSelf: 'stretch',
    flex: 1
  },
  camera: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
  },
```

Figur 157: Kode for Kamera



Figur 158: Kode for Kamera



```
  horizontalLine: {
    height: 1,
    width: '25%',
    top: '50%',
  },
  verticalLine: {
    height: '25%',
    width: 1,
    left: '50%',
  },
  overlayContainer: {
    marginLeft: 250,
    justifyContent: 'center',
    alignItems: 'center',
    marginBottom: 0,
  },
  overlayTextContainer: {
    padding: 10,
    borderRadius: 5,
  },
  overlayText: {
    color: 'white',
    fontSize: 16,
    marginBottom: 5,
  },
});
```


Figur 159: Kode for Kamera

```python
from flask import (Flask, jsonify, request)
# from flask_restful import Resource, Api
from flask_cors import CORS, cross_origin

#This is making the flask server
app = Flask(__name__)
CORS(app)

#This makes the server be able to handle GET request
@app.route("/api", methods=["GET"])
def get():
    return jsonify(hello='bouh !')

#This is starting the flask server on a decided host
if __name__ == '__main__':
    app.run(host="127.29.69.30", port=5000, debug=True)
```

Figur 160: Flask server

```
import React from 'react';
import { StyleSheet, Text, View, Image, TouchableOpacity} from 'react-native';
import * as ImagePicker from 'expo-image-picker';
import axios from 'axios';

const MyComponent = () => {
    // State variables
    const [selectedPhoto, setSelectedPhoto] = React.useState(null);
    const [displayedPhoto, setDisplayedPhoto] = React.useState(null);

    // Function to take a photo and upload it
    const takePhotoAndUpload = async () => {
        // Launch image picker for you to choose image
        let result = await ImagePicker.launchImageLibraryAsync({
            allowsEditing: false,
            aspect: [4, 3],
            quality: 1,
        });

        if (result.canceled) {
            return;
        }

        let localUri = result.uri;
        setDisplayedPhoto(localUri);
        let filename = localUri.split('/').pop();

        let match = /\.(\w+)$/.exec(filename);
        let type = match ? `image/${match[1]}` : `image`;
```

Figur 161: Sender bilde til server

```
    // Create a FormData object with the selected photo
    let formData = new FormData();
    formData.append('photo', { uri: localUri, name: filename, type });

    // Send the photo to the server using Axios
    await axios.get('http://127.29.69.30:5000/api', formData, {
        headers: { 'Content-Type': 'multipart/form-data' },
    }).then(res => {
        setSelectedPhoto(res.data.photo.photo);
    }).catch(err => {
        console.log(err.response);
    });
}

// Function to discard the currently displayed image
const discardImage = () => {
    setDisplayedPhoto(null);
}
```

Figur 162: Sender bilde til server

```
return (
    <View style={styles.mainBody}>
        <View style={styles.titleContainer}>
            <Text style={styles.title}>React Native Image Upload Axios</Text>
        </View>

        {/* Display the selected photo if available */}
        {displayedPhoto &&
            <View style={styles.imageContainer}>
                <Image
                    source={{ uri: displayedPhoto }}
                    style={{ width: '100%', height: 350 }}
                />
            </View>
        }
```

Figur 163: Sender bilde til server

```
            {/* Button to trigger the photo selection and upload */}
            <TouchableOpacity
                style={styles.buttonStyle}
                activeOpacity={0.5}
                onPress={takePhotoAndUpload}
            >
                <Text style={styles.buttonTextStyle}>Upload Image</Text>
            </TouchableOpacity>
            {/* Button to discard the displayed image */}
            <TouchableOpacity
                style={styles.buttonStyle}
                activeOpacity={0.5}
                onPress={discardImage}
            >
                <Text style={styles.buttonTextStyle}>Discard Image</Text>
            </TouchableOpacity>
        </View>
    );
}
```

Figur 164: Sender bilde til server

```
//Everything here is styling.
const styles = StyleSheet.create({
    mainBody: {
        flex: 1,
        justifyContent: 'center',
        padding: 20,
    },
    buttonStyle: {
        backgroundColor: '#307ecc',
        borderWidth: 0,
        color: '#FFFFFF',
        borderColor: '#307ecc',
        height: 40,
        alignItems: 'center',
        borderRadius: 30,
        marginLeft: 35,
        marginRight: 35,
        marginTop: 15,
    },
    buttonTextStyle: {
        color: '#FFFFFF',
        paddingVertical: 10,
        fontSize: 16,
    },
```

Figur 165: Sendebilde til server

Figur 166: Sender bilde til server

```
import React, { Component, useState } from 'react';
import { View, Button, Text, Image, StyleSheet, TouchableOpacity } from 'react-native';
import { ImageBackground } from 'react-native-web';
import Icon from 'react-native-vector-icons/FontAwesome'
import { TextInput } from 'react-native-paper';
import { SafeAreaView } from 'react-native-safe-area-context';
import { useRoute } from '@react-navigation/native';

export default function Result({navigation}, {route}){

    return (
    //This is the background image
    <ImageBackground
      style={styles.background}
      source={require('../assets/wh.jpg')}>
    <View style={styles.centerText}>
        <Image
            style={styles.ImagePlace}
            source={require('C:/Users/krist/OneDrive/Skrivebord/npx/Bachelor/assets/logo.png')}
        />
        <View style={styles.buttonContainer}>
        <TouchableOpacity
             style={styles.button}
             //This makes a button that navigates to another screen when pressed
             onPress={() => navigation.navigate("inputdata")}>
             <Icon name="camera" size={50} color="blue"/>
            </TouchableOpacity>
            <TouchableOpacity
             style={styles.button}
```

Figur 167: Kode for resultatside

```
             //This makes a button that navigates to another screen when pressed
             onPress={() => navigation.navigate("inputdata")}>
             <Icon name="star" size={50} color="blue"/>
            </TouchableOpacity>
            <View style={styles.centerText}>
            </View>

        </View>
    </View>
    </ImageBackground>
    );
  }
```

Figur 168: Kode for resultatside

```
//This is styling
const styles = StyleSheet.create({
    background: {
        flex: 1,
        resizeMode: "cover",
        justifyContent: "center"
    },
    centerText: {
        flex: 1,
        alignItems: 'center',
        justifyContent: 'center'
    },
    ImagePlace: {
        width: 200,
        height: 200,
        alignSelf: 'center',
        position: 'absolute',
        top: 0
    },
```

Figur 169: Kode for resultatside

Figur 170: Kode for resultatside

```
import React, { useState } from "react";
import { StyleSheet, View, Text, Button, TextInput } from 'react-native';
import { useRoute } from "@react-navigation/native";

export default function Inputdata({navigation}) {

    const [text,setText] = useState('');

    //Sends the input the user writes
    const sendData = () => {
        const data = 'Hello from inputdata!';
        //Navigates to result and takes the data with it
        navigation.navigate('Result', {data})
    }
```

Figur 171: Kode for brukerinput

```
    return (
        <View style={styles.maincontainer}>
            <Text style={styles.title}>Manuelle verdier</Text>
            <View style={styles.container}>

                <TextInput
                //This makes so that the user can write input
                    style={styles.input}
                    placeholder="Legg inn ønsket verdi"
                    onChangeText={(text) => setText(text)}
                    value={text}
                />

                <Button title="submit" onPress={() => navigation.navigate("Result") }
                />
            </View>
        </View>
    );
}
```

Figur 172: Kode for brukerinput

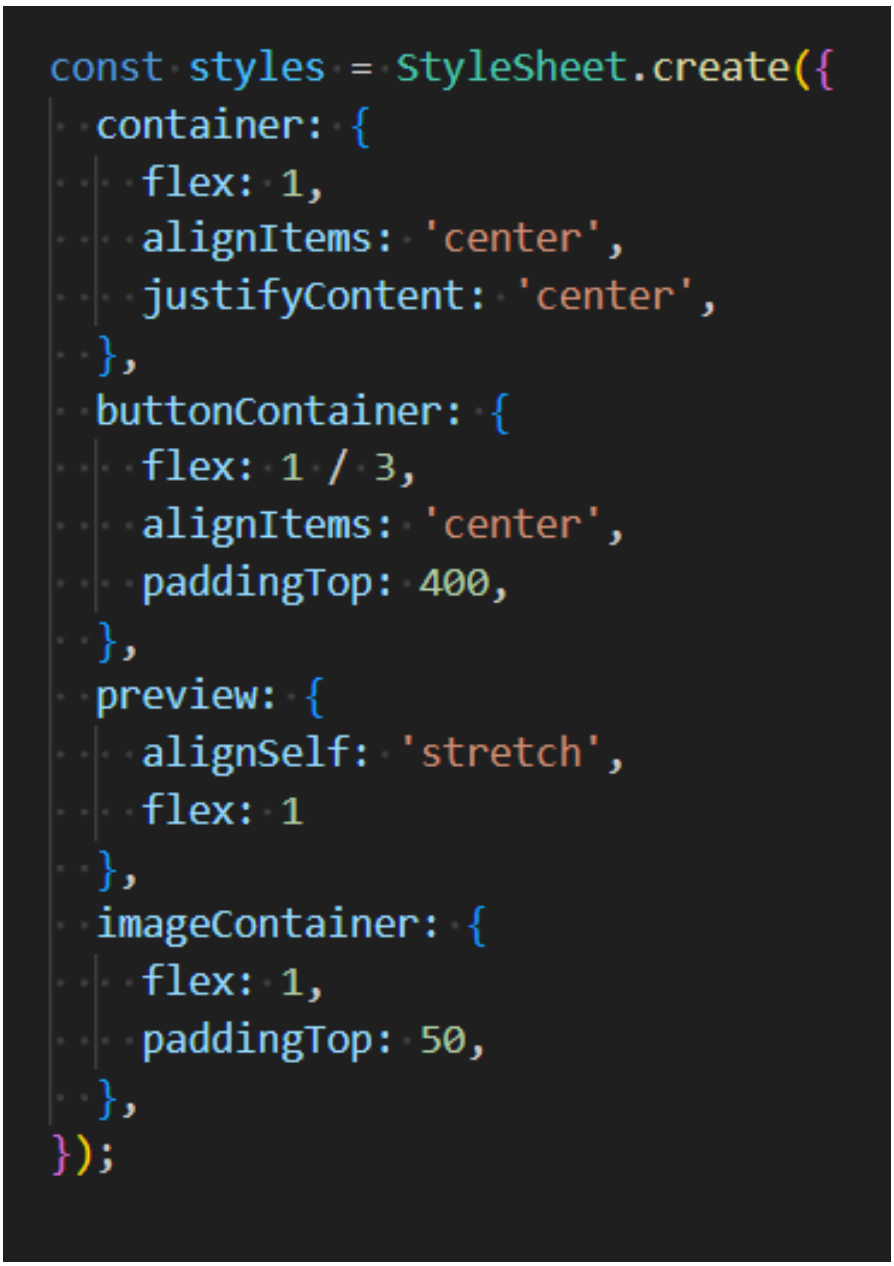
```
const styles = StyleSheet.create({
    maincontainer: {
        marginTop: 40,
    },
    input:{
        borderWidth:1,
        marginBottom:10,
        padding:10,
        width:'100%',
        borderRadius:10,
     },
    title: {
        backgroundColor: 'red',
        textAlign: 'center',
        padding: 10,
        fontSize: 20,
        color: '#FFFF',
        fontWeight:'bold',
    },
    container: {
        marginTop: 40,
        alignItems: 'center',
    },
});
```

Figur 173: Kode for brukerinput

I.2 Gallery

```jsx
import { StatusBar } from 'expo-status-bar';
import { StyleSheet, View, Button, Image } from 'react-native';
import {useState } from 'react';
import * as ImagePicker from 'expo-image-picker';

export default function Gallery() {
  const [image, setImage] = useState(null);

  // Uses the ImagePicker component from the corresponding library to access the image library in the phone
  const pickImage = async () => {
    let result = await ImagePicker.launchImageLibraryAsync({
      mediaTypes: ImagePicker.MediaTypeOptions.Images,
      allowsEditing: false,
      aspect: [4, 3],
      quality: 1,
      album: "IMG",
    });

    if (!result.canceled) {
      setImage(result.assets[0].uri);
    }
  };

  return (
      <View style={styles.buttonContainer} >
        <Button
        title="Pictures"
        onPress={pickImage}
        />
        {image && <Image source={{ uri: image }} style={{ width: 200, height: 200 }} />}
      <StatusBar style="auto" />
      </View>
  );

}
```

Figur 174: Kode for galleri



Figur 175: Kode for galleri

I.3 Home

```
import { StatusBar } from 'expo-status-bar';
import { StyleSheet, View, TouchableOpacity, Image, ImageBackground} from 'react-native';
import Icon from 'react-native-vector-icons/FontAwesome'


export default function Home({navigation}) {
  return (
    <ImageBackground
      style={styles.background}
      source={require('../assets/wh.jpg')}>
    <View style={styles.container}>
        <Image
            style={styles.homeLogo}
            source={require('../assets/logo.png')}
            />
        <View style={styles.buttonContainer}>
            <TouchableOpacity
            style={styles.button}
                onPress={() => navigation.navigate("Camera")}>
                <Icon name="camera" size={50} color="blue"/>
            </TouchableOpacity>

            <TouchableOpacity
                style={styles.button}
                onPress={() => navigation.navigate("Gallery")}>
                <Icon name="image" size={50} color="blue"/>
            </TouchableOpacity>
        </View>
      <StatusBar style="auto" />
    </View>
    </ImageBackground>
  );
}
```

Figur 176: Kode for hjemskjerm

```
const styles = StyleSheet.create({
  background: {
    flex: 1,
    resizeMode: "cover",
    justifyContent: "center"
  },
    container: {
        flex: 1,
        backgroundColor: 'transparent',
        alignItems: 'center',
      },
    homeLogo: {
        flex: 1,
        resizeMode: 'contain',
        paddingTop: 58
      },
    buttonContainer: {
        flexDirection: 'row',
        justifyContent: 'space-around',
        marginTop: 20,
    },
    button: {
      borderRadius: 10,
      padding: 10,
    },
});
```

Figur 177: Kode for hjemskjerm

I.4 Login

```
import { StyleSheet, Text, View, Button, Image, TouchableOpacity,SafeAreaView, ImageBackground} from 'react-native';
import React from 'react';
import { TextInput } from 'react-native-paper';
//import { getAuth, signInWithEmailAndPassword } from "firebase/auth";

export default function Login({navigation}){
  const [email, onChangeText] = React.useState('');
  const [password, onChangeNumber] = React.useState('');
  //const auth = getAuth();

  /* const handleSignIn = async () => {
  try {
    const userCredential = await signInWithEmailAndPassword(auth, email, password);
    console.log('Login successfull:', userCredential.user.uid);
    navigation.navigate('Home');
  } catch (error) {
    console.log(' Login failed:', error);
  }
  };     */
```

Figur 178: Kode for Login

```
return(
  <SafeAreaView style={{flex: 1, justifyContent: 'center' , alignItems: 'center'}}>
    <ImageBackground
      style={styles.background}
      source={require('../assets/grey.jpg')}>
    <Image
          style={styles.homeLogo}
          source={require('../assets/logo.png')}
          />
    <Text style={styles.text}> Login </Text>

    <TouchableOpacity
   style={styles.buttonReg}
   onPress={() => navigation.navigate("Register")}>
   <Text style={styles.textReg}> Register </Text>
   </TouchableOpacity>

  <TextInput
      label="Email"
      style={styles.input}
      onChangeText={onChangeText}
      value={email}
      mode="outlined"
    />
  <TextInput
      label="Password"
      style={styles.input}
      onChangeText={onChangeNumber}
      value={password}
      mode="outlined"
      secureTextEntry
    />
```

Figur 179: Kode for Login

I.5 Register

```
        <TouchableOpacity
      style={styles.button}
      onPress={() => navigation.navigate("Home")}>
      <Text style={styles.textLog}> Welcome </Text>
      </TouchableOpacity>
      </ImageBackground>
    </SafeAreaView>
    )
};
```

Figur 180: Kode for Login

```
const styles = StyleSheet.create({
  homeLogo: {
    flex: 1,
    resizeMode: 'contain',
  },
  text: {
    fontSize: 28,
    fontWeight: '500',
    color: '#333',
    marginBottom: 0,
    marginLeft: 10,
  },
  input: {
    height: 40,
    margin: 40,
    padding: 10,
  },
  button: {
    borderRadius: 10,
    padding: 10,
    marginBottom: 40,
  },
  buttonReg: {
    borderRadius: 10,
    padding: 10,
    marginBottom: -40,
  },
  textLog: {
    fontSize: 28,
    fontWeight: '500',
    color: '#333',
    marginBottom: 0,
    marginLeft: 140,
```

```
  textReg: {
    fontSize: 20,
    fontWeight: '500',
    color: '#333',
    marginBottom: 0,
    marginLeft: 285,
  },
});
```

Figur 182: Kode for Login

```
import { StyleSheet, Text, View, Button, Image, TouchableOpacity,SafeAreaView, ImageBackground} from 'react-native';
import React from 'react';
import { TextInput } from 'react-native-paper';
//import { getAuth, createUserWithEmailAndPassword } from "firebase/auth";

export default function Register({navigation}){
  const [email, onChangeText] = React.useState('');
  const [password, onChangeNumber] = React.useState('');
  //const auth = getAuth();

  /* const handleRegister = async () => {
    try {
      const userCredential = await createUserWithEmailAndPassword(auth, email, password);
      console.log('User registered successfully:', userCredential.user.uid);
      () => navigation.navigate("Login");
      // Navigate to the home screen or show a success message
    } catch (error) {
      console.log('Error registering user:', error);
      // Show an error message to the user
    }
  };     */
```

Figur 183: Kode for Login

```
return(
    <SafeAreaView style={{flex: 1, justifyContent: 'center' , alignItems: 'center'}}>
        <ImageBackground
            style={styles.background}
            source={require('../assets/grey.jpg')}>
        <Image
                style={styles.homeLogo}
                source={require('../assets/logo.png')}
                />
        <Text style={styles.text}> Register </Text>

    <TextInput
            label="Email"
            style={styles.input}
            onChangeText={onChangeText}
            value={email}
            mode="outlined"
        />
        <TextInput
            label="Password"
            style={styles.input}
            onChangeText={onChangeNumber}
            value={password}
            mode="outlined"
            secureTextEntry
        />
        <TouchableOpacity
            style={styles.button}
            onPress={() => ""}>
            <Text style={styles.textLog}> Create account </Text>
            </TouchableOpacity>
        </ImageBackground>
    </SafeAreaView>
    )
};
```

Figur 184: Kode for Login

```
const styles = StyleSheet.create({
  homeLogo: {
    flex: 1,
    resizeMode: 'contain',
  },
  text: {
    fontSize: 28,
    fontWeight: '500',
    color: '#333',
    marginBottom: 0,
    marginLeft: 10,
  },
  input: {
    height: 40,
    margin: 40,
    padding: 10,
  },
  button: {
    borderRadius: 10,
    padding: 10,
    marginBottom: 40,
  },
  textLog: {
    fontSize: 28,
    fontWeight: '500',
    color: '#333',
    marginBottom: 0,
    marginLeft: 100,
  },
  textReg: {
    fontSize: 15,
    fontWeight: '500',
    color: '#333',
    marginBottom: 0,
    marginLeft: 300,
  },
});
```

Figur 185: Kode for Login

```python
from flask import (Flask, jsonify, request)
# from flask_restful import Resource, Api
from flask_cors import CORS, cross_origin

#This is making the flask server
app = Flask(__name__)
CORS(app)

#This makes the server be able to handle GET request
@app.route("/api", methods=["GET"])
def get():
    return jsonify(hello='bouh !')

#This is starting the flask server on a decided host
if __name__ == '__main__':
    app.run(host="127.29.69.30", port=5000, debug=True)
```

Figur 186: Flask server

I.6 Kode for treffpunktmarkører

```python
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt

class markingCenters:

    # A list to store the coordinates of all pixels in all shooting holes that are detected
    shotCoords = []

    def __init__(self, img):
        self.img = img
        self.width = img.shape[1]
        self.height = img.shape[0]

        '''
        # Jump certain amount of pixel to find the limit of searching Area
        self.jumpPixel_limit = jumpPixel_limit = 500

        lowResolutionLimit = 5000
        mediumResolutionLimit = 9000

        resolutionFactor = 3

        # if/elif to check what the resoultion size is and modifiy how much to search and jump looking for pixel rgbs
        # check if resolution is lower than 5000x5000
        if (self.width < lowResolutionLimit and self.height < lowResolutionLimit):
            searchingRadius = searchingRadius // resolutionFactor
            jumpPixel_x = int(jumpPixel_x / resolutionFactor )
            jumpPixel_y = int(jumpPixel_y / resolutionFactor )
            jumpFactor = int(jumpFactor / resolutionFactor)

            jumpPixel_limit = jumpPixel_limit // resolutionFactor

            markedDiameter = markedDiameter // resolutionFactor
            pixelDistance_deviation = pixelDistance_deviation // resolutionFactor
```

Figur 187: Kode for treffpunktmarkører

```python
        # check if resolution is less then 9000x9000
        elif(self.width < mediumResolutionLimit and self.height < mediumResolutionLimit):
            searchingRadius = searchingRadius // (resolutionFactor - 1)
            jumpPixel_x = int(jumpPixel_x / (resolutionFactor - 1))
            jumpPixel_y = int(jumpPixel_y / (resolutionFactor - 1))
            jumpFactor = int(jumpFactor / (resolutionFactor - 1))

            jumpPixel_limit = jumpPixel_limit // (resolutionFactor - 1)

            markedDiameter = markedDiameter // (resolutionFactor - 1)
            pixelDistance_deviation = pixelDistance_deviation // (resolutionFactor - 1)
        '''
```

Figur 188: Kode for treffpunktmarkører

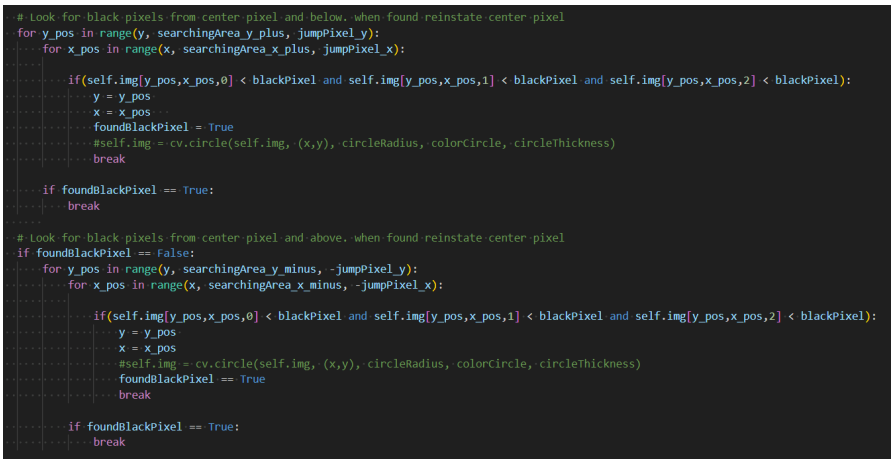
I.7 Kode for utregning av treffpunkt markører

```python
    # find a pixel close to the center of the shooting target
    def find_center(self, display = False):

        # center of image based on resolution
        x = self.width // 2
        y = self.height // 2

        # Array for black pixels detected, black pixels is supposed to represent the target
        blackPixelsArray = []

        # count how many times we detect white pixel
        count = 0

        # value that is max threshold for black pixel
        blackPixel = 30

        # value that is min threshold from 170 to 255
        whitePixel = 170

        # check amount of times for white pixelø
        checkLimit = 6
        lowResolutionLimit = 5000
        mediumResolutionLimit = 9000

        # Searching radius
        searchingRadius = 1000
        # how much to jump in each axis
        jumpPixel_x = 100
        jumpPixel_y = 100
        jumpFactor = 50
```

Figur 189: Kode for treffpunktmarkører

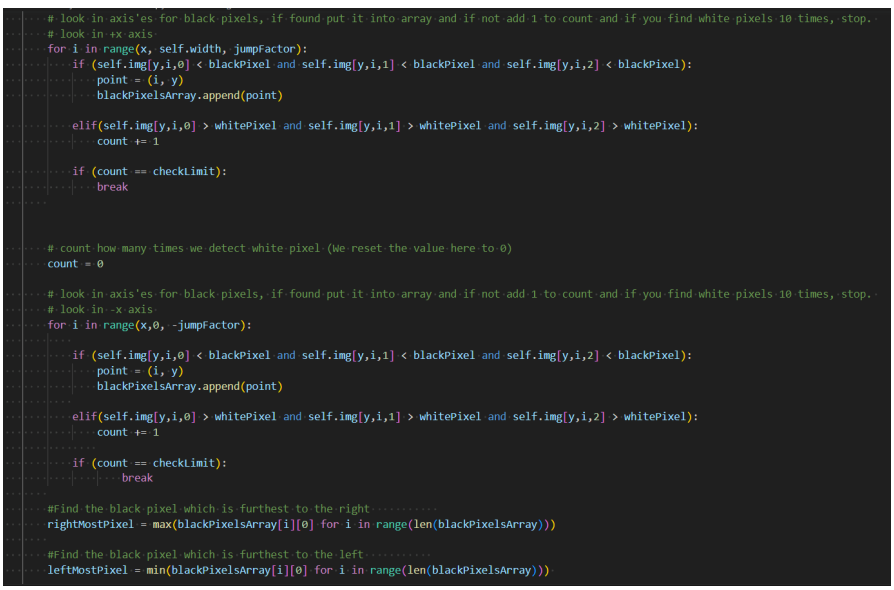
```python
    resolutionFactor = 2
    # if/elif to check what the resoultion size is and modifiy how much to search and jump looking for pixel rgbs
    # check if resolution is lower than 5000x5000
    if (self.width < lowResolutionLimit and self.height < lowResolutionLimit):
        searchingRadius = int(searchingRadius / resolutionFactor)
        jumpPixel_x = int(jumpPixel_x / resolutionFactor )
        jumpPixel_y = int(jumpPixel_y / resolutionFactor )
        jumpFactor = int(jumpFactor / resolutionFactor)

    # check if resolution is less then 9000x9000
    elif(self.width < mediumResolutionLimit and self.height < mediumResolutionLimit):
        searchingRadius = int(searchingRadius / (resolutionFactor - 0.5))
        jumpPixel_x = int(jumpPixel_x / (resolutionFactor - 0.5))
        jumpPixel_y = int(jumpPixel_y / (resolutionFactor - 0.5))
        jumpFactor = int(jumpFactor / (resolutionFactor - 0.5))

    # Searching limit for each axis
    searchingArea_x_plus = x + searchingRadius
    searchingArea_y_plus = y + searchingRadius
    searchingArea_x_minus = x - searchingRadius
    searchingArea_y_minus = y - searchingRadius

    # Used for checking if a black pixel is foubd
    foundBlackPixel = False

    # the size of circle
    circleRadius = 50

    # the thickness of a circlce ( if -1, mark the whole circle)
    circleThickness = -1

    # circle color
    colorCircle = (255, 255, 255)
```

Figur 190: Kode for treffpunktmarkører



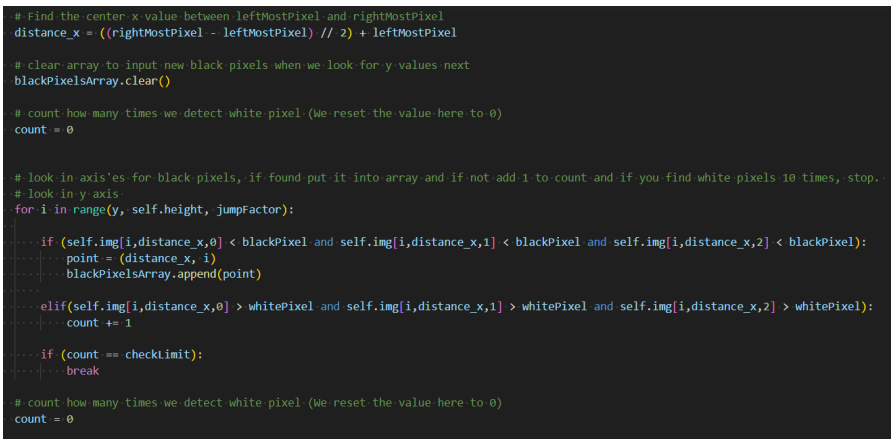
```
# Look for black pixels from center pixel and below. when found reinstate center pixel
for y_pos in range(y, searchingArea_y_plus, jumpPixel_y):
    for x_pos in range(x, searchingArea_x_plus, jumpPixel_x):

        if(self.img[y_pos,x_pos,0] < blackPixel and self.img[y_pos,x_pos,1] < blackPixel and self.img[y_pos,x_pos,2] < blackPixel):
            y = y_pos
            x = x_pos
            foundBlackPixel = True
            #self.img = cv.circle(self.img, (x,y), circleRadius, colorCircle, circleThickness)
            break

    if foundBlackPixel == True:
        break

# Look for black pixels from center pixel and above. when found reinstate center pixel
if foundBlackPixel == False:
    for y_pos in range(y, searchingArea_y_minus, -jumpPixel_y):
        for x_pos in range(x, searchingArea_x_minus, -jumpPixel_x):

            if(self.img[y_pos,x_pos,0] < blackPixel and self.img[y_pos,x_pos,1] < blackPixel and self.img[y_pos,x_pos,2] < blackPixel):
                y = y_pos
                x = x_pos
                #self.img = cv.circle(self.img, (x,y), circleRadius, colorCircle, circleThickness)
                foundBlackPixel == True
                break

        if foundBlackPixel == True:
            break
```

Figur 191: Kode for treffpunktmarkører



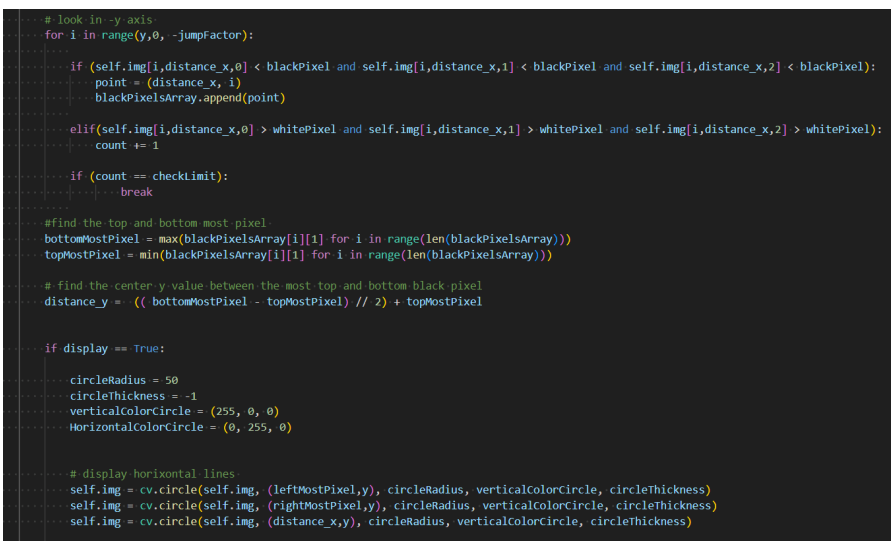
```
    # look in axis'es for black pixels, if found put it into array and if not add 1 to count and if you find white pixels 10 times, stop.
    # look in +x axis
    for i in range(x, self.width, jumpFactor):
        if (self.img[y,i,0] < blackPixel and self.img[y,i,1] < blackPixel and self.img[y,i,2] < blackPixel):
            point = (i, y)
            blackPixelsArray.append(point)

        elif(self.img[y,i,0] > whitePixel and self.img[y,i,1] > whitePixel and self.img[y,i,2] > whitePixel):
            count += 1

        if (count == checkLimit):
            break



    # count how many times we detect white pixel (We reset the value here to 0)
    count = 0

    # look in axis'es for black pixels, if found put it into array and if not add 1 to count and if you find white pixels 10 times, stop.
    # look in -x axis
    for i in range(x,0, -jumpFactor):

        if (self.img[y,i,0] < blackPixel and self.img[y,i,1] < blackPixel and self.img[y,i,2] < blackPixel):
            point = (i, y)
            blackPixelsArray.append(point)

        elif(self.img[y,i,0] > whitePixel and self.img[y,i,1] > whitePixel and self.img[y,i,2] > whitePixel):
            count += 1

        if (count == checkLimit):
            break

    #Find the black pixel which is furthest to the right
    rightMostPixel = max(blackPixelsArray[i][0] for i in range(len(blackPixelsArray)))

    #Find the black pixel which is furthest to the left
    leftMostPixel = min(blackPixelsArray[i][0] for i in range(len(blackPixelsArray)))
```

Figur 192: Kode for treffpunktmarkører



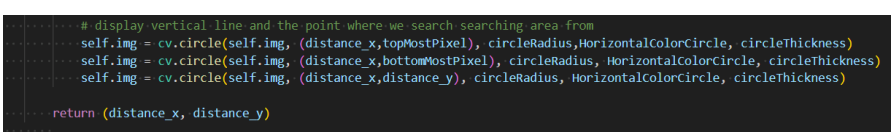
```
# Find the center x value between leftMostPixel and rightMostPixel
distance_x = ((rightMostPixel - leftMostPixel) // 2) + leftMostPixel

# clear array to input new black pixels when we look for y values next
blackPixelsArray.clear()

# count how many times we detect white pixel (We reset the value here to 0)
count = 0


# look in axis'es for black pixels, if found put it into array and if not add 1 to count and if you find white pixels 10 times, stop.
# look in y axis
for i in range(y, self.height, jumpFactor):

    if (self.img[i,distance_x,0] < blackPixel and self.img[i,distance_x,1] < blackPixel and self.img[i,distance_x,2] < blackPixel):
        point = (distance_x, i)
        blackPixelsArray.append(point)

    elif(self.img[i,distance_x,0] > whitePixel and self.img[i,distance_x,1] > whitePixel and self.img[i,distance_x,2] > whitePixel):
        count += 1

    if (count == checkLimit):
        break

# count how many times we detect white pixel (We reset the value here to 0)
count = 0
```

Figur 193: Kode for treffpunktmarkører



```
    # look in -y axis
    for i in range(y,0, -jumpFactor):

        if (self.img[i,distance_x,0] < blackPixel and self.img[i,distance_x,1] < blackPixel and self.img[i,distance_x,2] < blackPixel):
            point = (distance_x, i)
            blackPixelsArray.append(point)

        elif(self.img[i,distance_x,0] > whitePixel and self.img[i,distance_x,1] > whitePixel and self.img[i,distance_x,2] > whitePixel):
            count += 1

        if (count == checkLimit):
            break

    #find the top and bottom most pixel
    bottomMostPixel = max(blackPixelsArray[i][1] for i in range(len(blackPixelsArray)))
    topMostPixel = min(blackPixelsArray[i][1] for i in range(len(blackPixelsArray)))

    # find the center y value between the most top and bottom black pixel
    distance_y = (( bottomMostPixel - topMostPixel) // 2) + topMostPixel


    if display == True:

        circleRadius = 50
        circleThickness = -1
        verticalColorCircle = (255, 0, 0)
        HorizontalColorCircle = (0, 255, 0)


        # display horixontal lines
        self.img = cv.circle(self.img, (leftMostPixel,y), circleRadius, verticalColorCircle, circleThickness)
        self.img = cv.circle(self.img, (rightMostPixel,y), circleRadius, verticalColorCircle, circleThickness)
        self.img = cv.circle(self.img, (distance_x,y), circleRadius, verticalColorCircle, circleThickness)
```

Figur 194: Kode for treffpunktmarkører



```
        # display vertical line and the point where we search searching area from
        self.img = cv.circle(self.img, (distance_x,topMostPixel), circleRadius,HorizontalColorCircle, circleThickness)
        self.img = cv.circle(self.img, (distance_x,bottomMostPixel), circleRadius, HorizontalColorCircle, circleThickness)
        self.img = cv.circle(self.img, (distance_x,distance_y), circleRadius, HorizontalColorCircle, circleThickness)

    return (distance_x, distance_y)
```

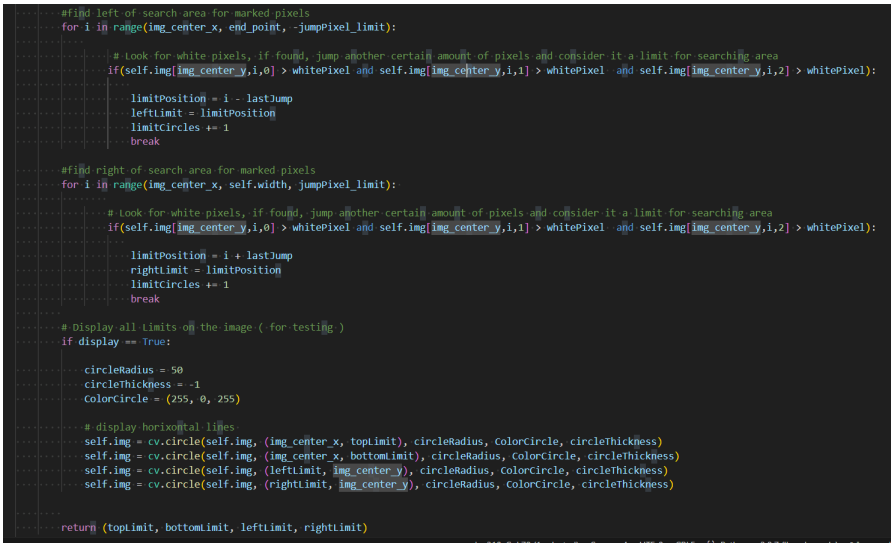
Figur 195: Kode for treffpunktmarkører

```python
# create a square for searching area
def find_searching_area_limits(self, display = False):

    # Starting point from where we utilize Jump factor to find searching Area
    centerPoint = self.find_center()
    img_center_x = centerPoint[0]
    img_center_y = centerPoint[1]

    # Jump certain amount of pixel to find the limit of searching Area
    jumpPixel_limit = 100

    lowResolutionLimit = 5000
    mediumResolutionLimit = 9000

    resolutionFactor = 2

    # if/elif to check what the resoultion size is and modifiy how much to search and jump looking for pixel rgbs
    # check if resolution is lower than 5000x5000
    if (self.width < lowResolutionLimit and self.height < lowResolutionLimit):
        jumpPixel_limit = int(jumpPixel_limit / resolutionFactor)

    # check if resolution is less then 9000x9000
    elif(self.width < mediumResolutionLimit and self.height < mediumResolutionLimit):
        jumpPixel_limit = int(jumpPixel_limit / (resolutionFactor - 0.5))



    # to calculate start of the image in y and x axis
    end_point = -1

    # min value for detection of white pixel
    whitePixel = 150

    # values for knowing the limit of searching area
    topLimit = bottomLimit = leftLimit = rightLimit = 0
```

Figur 196: Kode for treffpunktmarkører

```python
# count amount of circles created ( for testing )
limitCircles = 0

lastJump = 500

#find Toplimit of search area for marked pixels
for i in range(img_center_y, end_point, -jumpPixel_limit):

        # Look for white pixels, if found, jump another certain amount of pixels and consider it a limit for searching area
        if(self.img[i,img_center_x,0] > whitePixel and self.img[i,img_center_x,1] > whitePixel and self.img[i,img_center_x,2] > whitePixel):

            limitPosition = i -lastJump
            topLimit = limitPosition
            limitCircles += 1
            break


#find bottom of search area for marked pixels
for i in range(img_center_y, self.height, jumpPixel_limit):

        # Look for white pixels, if found, jump another certain amount of pixels and consider it a limit for searching area
        if(self.img[i,img_center_x,0] > whitePixel and self.img[i,img_center_x,1] > whitePixel and self.img[i,img_center_x,2] > whitePixel):

            limitPosition = i +lastJump
            bottomLimit = limitPosition
            limitCircles += 1
            break
```

Figur 197: Kode for treffpunktmarkører

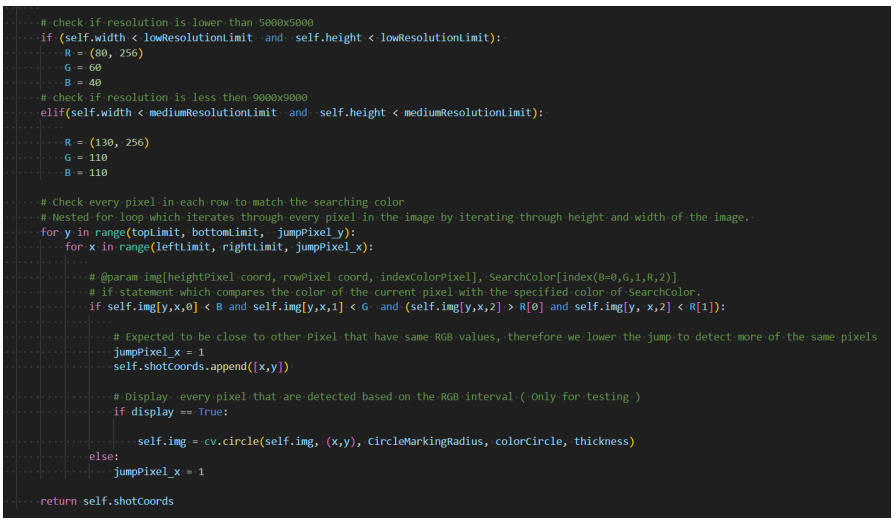


```
    #find left of search area for marked pixels
    for i in range(img_center_x, end_point, -jumpPixel_limit):

        # Look for white pixels, if found, jump another certain amount of pixels and consider it a limit for searching area
        if(self.img[img_center_y,i,0] > whitePixel and self.img[img_center_y,i,1] > whitePixel  and self.img[img_center_y,i,2] > whitePixel):

            limitPosition = i - lastJump
            leftLimit = limitPosition
            limitCircles += 1
            break

    #find right of search area for marked pixels
    for i in range(img_center_x, self.width, jumpPixel_limit):

        # Look for white pixels, if found, jump another certain amount of pixels and consider it a limit for searching area
        if(self.img[img_center_y,i,0] > whitePixel and self.img[img_center_y,i,1] > whitePixel  and self.img[img_center_y,i,2] > whitePixel):

            limitPosition = i + lastJump
            rightLimit = limitPosition
            limitCircles += 1
            break

    # Display all Limits on the image ( for testing )
    if display == True:

        circleRadius = 50
        circleThickness = -1
        ColorCircle = (255, 0, 255)

        # display horizontal lines
        self.img = cv.circle(self.img, (img_center_x, topLimit), circleRadius, ColorCircle, circleThickness)
        self.img = cv.circle(self.img, (img_center_x, bottomLimit), circleRadius, ColorCircle, circleThickness)
        self.img = cv.circle(self.img, (leftLimit, img_center_y), circleRadius, ColorCircle, circleThickness)
        self.img = cv.circle(self.img, (rightLimit, img_center_y), circleRadius, ColorCircle, circleThickness)


    return (topLimit, bottomLimit, leftLimit, rightLimit)
```

Figur 198: Kode for treffpunktmarkører

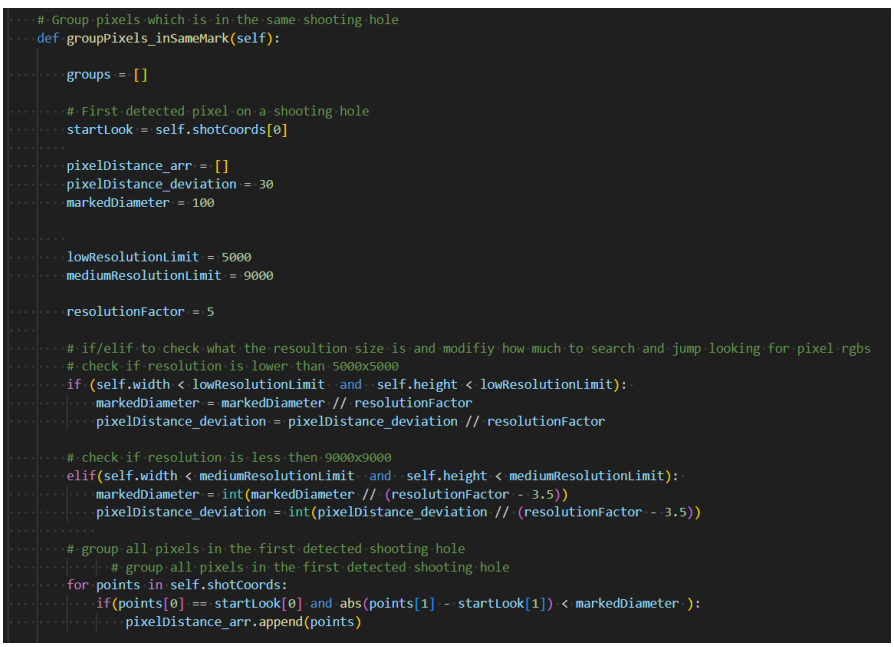
```
    # look for pixel with a same RGB interval and group them
    def groupPixelWithSameRGB(self, color, display = False):

        # find the searching area
        searchingLimits = self.find_searching_area_limits()


        # values of each limits for the searching area
        topLimit = searchingLimits[0]
        bottomLimit = searchingLimits[1]
        leftLimit = searchingLimits[2]
        rightLimit = searchingLimits[3]

        # amount of pixels to jump
        jumpPixel_x = 1
        jumpPixel_y = 1

        CircleMarkingRadius = 1
        colorCircle = (0, 255, 0)
        thickness = -1

        # RGB interval to find the pixels we are looking for
        if color == "red":

            lowResolutionLimit = 5000
            mediumResolutionLimit = 9000
```

Figur 199: Kode for treffpunktmarkører



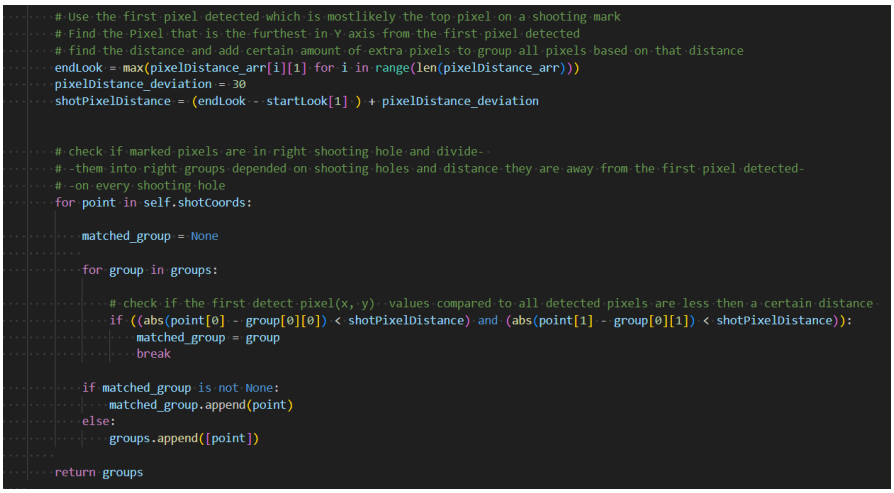
```
    # check if resolution is lower than 5000x5000
    if (self.width < lowResolutionLimit and self.height < lowResolutionLimit):
        R = (80, 256)
        G = 60
        B = 40
    # check if resolution is less then 9000x9000
    elif(self.width < mediumResolutionLimit and self.height < mediumResolutionLimit):

        R = (130, 256)
        G = 110
        B = 110

    # Check every pixel in each row to match the searching color
    # Nested for loop which iterates through every pixel in the image by iterating through height and width of the image.
    for y in range(topLimit, bottomLimit, jumpPixel_y):
        for x in range(leftLimit, rightLimit, jumpPixel_x):

            # @param img[heightPixel coord, rowPixel coord, indexColorPixel], SearchColor[index(B=0,G,1,R,2)]
            # if statement which compares the color of the current pixel with the specified color of SearchColor.
            if self.img[y,x,0] < B and self.img[y,x,1] < G and (self.img[y,x,2] > R[0] and self.img[y, x,2] < R[1]):

                # Expected to be close to other Pixel that have same RGB values, therefore we lower the jump to detect more of the same pixels
                jumpPixel_x = 1
                self.shotCoords.append([x,y])

                # Display every pixel that are detected based on the RGB interval ( Only for testing )
                if display == True:

                    self.img = cv.circle(self.img, (x,y), CircleMarkingRadius, colorCircle, thickness)
            else:
                jumpPixel_x = 1

    return self.shotCoords
```

Figur 200: Kode for treffpunktmarkører



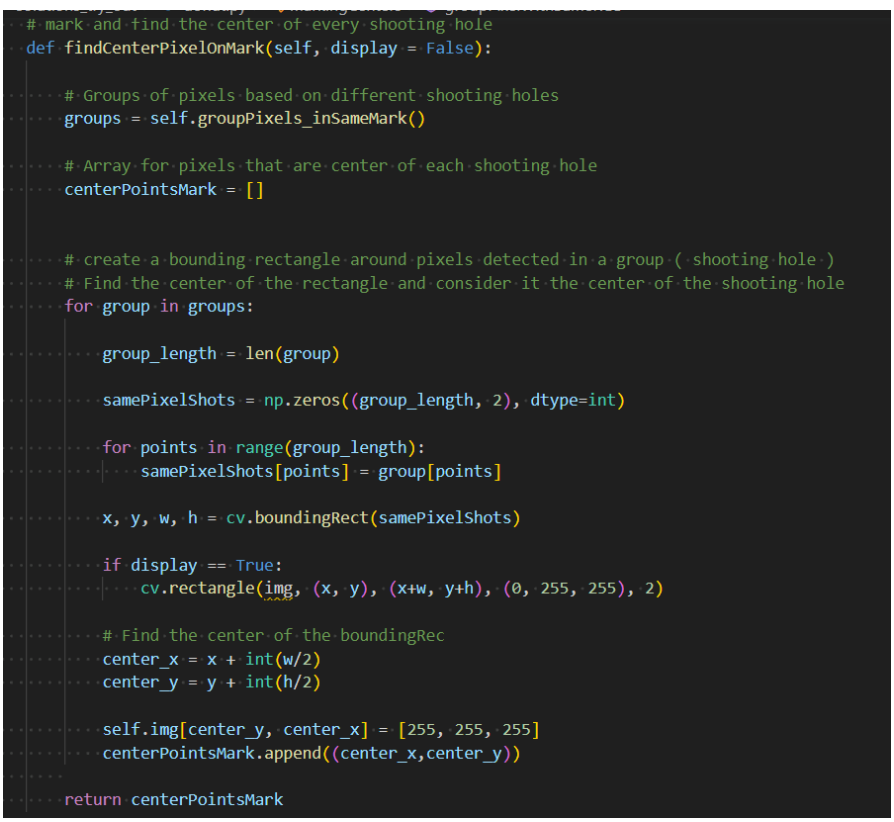
```
    # Group pixels which is in the same shooting hole
    def groupPixels_inSameMark(self):

        groups = []

        # First detected pixel on a shooting hole
        startLook = self.shotCoords[0]

        pixelDistance_arr = []
        pixelDistance_deviation = 30
        markedDiameter = 100


        lowResolutionLimit = 5000
        mediumResolutionLimit = 9000

        resolutionFactor = 5

        # if/elif to check what the resoultion size is and modifiy how much to search and jump looking for pixel rgbs
        # check if resolution is lower than 5000x5000
        if (self.width < lowResolutionLimit and self.height < lowResolutionLimit):
            markedDiameter = markedDiameter // resolutionFactor
            pixelDistance_deviation = pixelDistance_deviation // resolutionFactor

        # check if resolution is less then 9000x9000
        elif(self.width < mediumResolutionLimit and self.height < mediumResolutionLimit):
            markedDiameter = int(markedDiameter // (resolutionFactor - 3.5))
            pixelDistance_deviation = int(pixelDistance_deviation // (resolutionFactor - 3.5))

        # group all pixels in the first detected shooting hole
                # group all pixels in the first detected shooting hole
        for points in self.shotCoords:
            if(points[0] == startLook[0] and abs(points[1] - startLook[1]) < markedDiameter ):
                pixelDistance_arr.append(points)
```

Figur 201: Kode for treffpunktmarkører



```
    # Use the first pixel detected which is mostlikely the top pixel on a shooting mark
    # Find the Pixel that is the furthest in Y axis from the first pixel detected
    # find the distance and add certain amount of extra pixels to group all pixels based on that distance
    endLook = max(pixelDistance_arr[i][1] for i in range(len(pixelDistance_arr)))
    pixelDistance_deviation = 30
    shotPixelDistance = (endLook - startLook[1] ) + pixelDistance_deviation


    # check if marked pixels are in right shooting hole and divide
    # them into right groups depended on shooting holes and distance they are away from the first pixel detected
    # on every shooting hole
    for point in self.shotCoords:

        matched_group = None

        for group in groups:

            # check if the first detect pixel(x, y) values compared to all detected pixels are less then a certain distance
            if ((abs(point[0] - group[0][0]) < shotPixelDistance) and (abs(point[1] - group[0][1]) < shotPixelDistance)):
                matched_group = group
                break

        if matched_group is not None:
            matched_group.append(point)
        else:
            groups.append([point])

    return groups
```

Figur 202: Kode for treffpunktmarkører



```
# mark and find the center of every shooting hole
def findCenterPixelOnMark(self, display = False):

    # Groups of pixels based on different shooting holes
    groups = self.groupPixels_inSameMark()

    # Array for pixels that are center of each shooting hole
    centerPointsMark = []


    # create a bounding rectangle around pixels detected in a group ( shooting hole )
    # Find the center of the rectangle and consider it the center of the shooting hole
    for group in groups:

        group_length = len(group)

        samePixelShots = np.zeros((group_length, 2), dtype=int)

        for points in range(group_length):
            samePixelShots[points] = group[points]

        x, y, w, h = cv.boundingRect(samePixelShots)

        if display == True:
            cv.rectangle(img, (x, y), (x+w, y+h), (0, 255, 255), 2)

        # Find the center of the boundingRec
        center_x = x + int(w/2)
        center_y = y + int(h/2)

        self.img[center_y, center_x] = [255, 255, 255]
        centerPointsMark.append((center_x,center_y))

    return centerPointsMark
```

Figur 203: Kode for treffpunktmarkører

```
def get_centerValues(self, color, display):
    self.shotCoords.clear()
    self.groupPixelWithSameRGB(color)
    self.groupPixels_inSameMark()
    centerPoints = self.findCenterPixelOnMark()
    return centerPoints
```

Figur 204: Kode for treffpunktmarkører

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg      You, 1 second ago • Uncommitted changes
from markedAxisShoot2 import markingCenters
import math
import cv2 as cv


path1 = "C:/Users/Samue/Desktop/BachelorGit/WebBachelor/Program/samuel_filer/samuel_bilder/108mpSamsungS21ultraTest/4444.jpg"
path2 = "C:/Users/Samue/Desktop/BachelorGit/WebBachelor/Program/samuel_filer/samuel_bilder/108mpSamsungS21ultraTest/5555.jpg"
path3 = "C:/Users/Samue/Desktop/BachelorGit/WebBachelor/Program/samuel_filer/samuel_bilder/108mpSamsungS21ultraTest/7777.jpg"
path4 = "C:/Users/Samue/Desktop/BachelorGit/WebBachelor/Program/samuel_filer/samuel_bilder/108mpSamsungS21ultraTest/8888.jpg"
path5 = "C:/Users/Samue/Desktop/BachelorGit/WebBachelor/Program/samuel_filer/samuel_bilder/108mpSamsungS21ultraTest/9999.jpg"

path12 = "C:/Users/Samue/Desktop/BachelorGit/WebBachelor/Program/samuel_filer/samuel_bilder/iphone12mpTest/iphone2.jpg"
path13 = "C:/Users/Samue/Desktop/BachelorGit/WebBachelor/Program/samuel_filer/samuel_bilder/iphone12mpTest/iphone3.jpg"
path14 = "C:/Users/Samue/Desktop/BachelorGit/WebBachelor/Program/samuel_filer/samuel_bilder/iphone12mpTest/iphone4.jpg"

path17 = "C:/Users/Samue/Desktop/BachelorGit/WebBachelor/Program/samuel_filer/samuel_bilder/senayTest/s1.jpg"
path18 = "C:/Users/Samue/Desktop/BachelorGit/WebBachelor/Program/samuel_filer/samuel_bilder/senayTest/s2.jpg"
path19 = "C:/Users/Samue/Desktop/BachelorGit/WebBachelor/Program/samuel_filer/samuel_bilder/senayTest/s3.jpg"

```

Figur 205: Kode for utregning av origo og utregning av treffpunkter

```
path22 = "C:/Users/Samue/Desktop/BachelorGit/WebBachelor/Program/samuel_filer/samuel_bilder/sollysSamsungs21/1.jpg"
#*****************************************************************************
path25 = "C:/Users/Samue/Desktop/BachelorGit/WebBachelor/Program/samuel_filer/samuel_bilder/sollysIphone11/sol1.jpg"
#*****************************************************************************


img = cv.imread(path25)

heightPixels = (img.shape[0])   #Width in pic in pixles
widthPixels =(img.shape[1])     #Height in pic in pixles


redMarks = markingCenters(img).get_centerValues("red",False)



lowest_x = redMarks[0] # initialize lowest element x in list
lowest_y = redMarks[0] # initialize lowest element y in list
highest_x = redMarks[0]
highest_y = redMarks[0]
```

Figur 206: Kode for utregning av origo og utregning av treffpunkter

```python
for tuple in redMarks:
    if tuple[0] < lowest_x[0]:
        lowest_x = tuple
        tuple = lowest_x
    if tuple[1] < lowest_y[1]:
        lowest_y = tuple
    if tuple[0] > highest_x[0]:
        highest_x = tuple
    if tuple[1] > highest_y[0]:
        highest_y = tuple

# Define the starting and ending points of the first vector
hor_start = np.array([lowest_x[0],lowest_x[1]])
hor_end = np.array([highest_x[0], highest_x[1]])

# Define the starting and ending points of the second vector
ver_start = np.array([lowest_y[0],lowest_y[1]])
ver_end = np.array([highest_y[0],highest_y[1]])
intersection_point = []

# Finding direction vector of horizontal and vertical axis vecktor
horVecDirection = hor_end - hor_start
verVecDiretion = ver_end - ver_start
```

Figur 207: Kode for utregning av origo og utregning av treffpunkter

```python
# Calculate the determinant of the matrix formed by the direction vectors
det = np.linalg.det(np.vstack((horVecDirection, verVecDiretion)))

# Check if the vectors are parallel (determinant is 0)
if np.isclose(det, 0):
    # Check if the vectors are collinear
    print("The vectors are either parallel or collinear and do not intersect.")
else:
    # Calculate the parameter values for each vector using Cramer's rule
    v1_param = np.linalg.det(np.vstack((ver_start - hor_start, verVecDiretion))) / det
    v2_param = np.linalg.det(np.vstack((ver_start - hor_start, horVecDirection))) / det

    # Check if the intersection point is within the parameters of both vectors
    if (0 <= v1_param <= 1) and (0 <= v2_param <= 1):
        # Calculate the intersection point using the parameter values
        intersection_point = hor_start + v1_param * horVecDirection

        print("Intersection coord is:", intersection_point)
    else:
        print("Vectors intersect, but they are not within the parameters of both vectors")
print(intersection_point)
```

Figur 208: Kode for utregning av origo og utregning av treffpunkter

```python
#Calculate delta x1 for square 1,4 and delta x2 for 2 and 3
pixel_size_horizontal_1_4 = 240/abs(intersection_point[0]-hor_end[0])
pixel_size_horizontal_2_3 = 240/abs(intersection_point[0]-hor_start[0])


#Calculate delta y1 for square 1, 2  and 3,4
pixel_size_vertical_1_2 = 240/abs(intersection_point[1]-ver_start[1])
pixel_size_vertical_3_4 = 240/abs(intersection_point[1]-ver_end[1])


#Create check up variables in case intersection to axis are not 240 mm
diff_h_1_2  = diff_h_3_4=  diff_w_1_4 =  diff_w_2_3= 0

diff_xy_1_4 = diff_xy_2_3 = diff_yx_1_2 = diff_yx_3_4 = 0
```

Figur 209: Kode for utregning av origo og utregning av treffpunkter

```
#Check if distance between intersection point (Origo) to vertical axis are 240 and if not add and subtract the difference
if (highest_y[1]-intersection_point[1])*pixel_size_vertical_1_2 != 240 or (abs(lowest_y[1]-intersection_point[1])*pixel_size_vertical_1_2 != 240):
    #Positive means center is dragged to right axis with height diff
    diff_h_3_4 = 240-(highest_y[1]-intersection_point[1])*pixel_size_vertical_3_4
    diff_h_1_2 = 240+(lowest_y[1]-intersection_point[1])*pixel_size_vertical_1_2
    print(pixel_size_vertical_1_2*(-1*(redMarks[0][1]-intersection_point[1])),"1,2")
    print(pixel_size_vertical_3_4*(-1*(redMarks[0][1]-intersection_point[1])),"3,4")
    print(pixel_size_vertical_3_4*(-1*(redMarks[-1][1]-intersection_point[1])),"3,4Sisteskudd")
    print(diff_h_1_2,diff_h_3_4)


#Check if distance between intersection point (Origo) to horizontal axis are 240 and if not add and subtract the difference
if ((highest_x[0]-intersection_point[0])*pixel_size_horizontal_1_4 != 240 or abs(lowest_x[0]-intersection_point[0])*pixel_size_horizontal_2_3) !=240:
    #Positive means center is dragged down to bottom axis with width diff
    diff_w_1_4 = 240-(highest_x[0]-intersection_point[0])*pixel_size_horizontal_1_4

    diff_w_2_3 = 240+((lowest_x[0]-intersection_point[0]))*pixel_size_horizontal_2_3


fig, ax = plt.subplots()
tempX = tempY = 0
```

Figur 210: Kode for utregning av origo og utregning av treffpunkter

```
#Automatic test for all shoots
for x in range (0,len(redMarks),1):
    print("*****************************************")
    if ((redMarks[x][0]-intersection_point[0]) > 0):
        if(tempX == -240 and tempX > 0.5):
            print("Grader er over 5 og resultatene vil avvike")
        tempX = (pixel_size_horizontal_1_4*(redMarks[x][0]-intersection_point[0])) -abs(diff_w_1_4)
    else:
        tempX = (pixel_size_horizontal_2_3*(redMarks[x][0]-intersection_point[0])) +abs(diff_w_2_3)
    if ((redMarks[x][1]-intersection_point[1]) > 0):
        tempY = (pixel_size_vertical_3_4*(-1*(redMarks[x][1]-intersection_point[1]))) -abs(diff_h_1_2)
        if(tempY == -240 and tempX > 0.5):
            print("Grader er over 5 og resultatene vil avvike")

    else:
        tempY = (pixel_size_vertical_1_2*(-1*(redMarks[x][1]-intersection_point[1])))+abs(diff_h_3_4)
        if(tempY == 240 and tempX < -0.5):
            print("Grader er over 5 og resultatene vil avvike")

    print(x,tempX,"X",tempY, "Y")
    ax.plot([intersection_point[0],redMarks[x][0]], [intersection_point[1], redMarks[x][1]], color='g', linewidth=5)
```

Figur 211: Kode for utregning av origo og utregning av treffpunkter

```
#Define starting and ending points of axis. Start at left top corner
plt.xlim(0,widthPixels)
plt.ylim(heightPixels,0)
img = cv.cvtColor(img, cv.COLOR_BGR2RGB)
ax.imshow(img)

plt.show()
```

Figur 212: Kode for utregning av origo og utregning av treffpunkter