Høgskulen
på Vestlandet

BACHELOROPPGAVE

Bildekvalitetsdetektor for dekk på kjøretøy i bevegelse

Image Quality Detector for tires on vehicles in motion

Gruppe D14

Håkon Lervåg

Øyvind Holter

Bjørn Ellingsen

DAT191
Fakultetet for ingeniør- og naturvitenskap

Institutt for datateknologi, elektroteknologi og realfag
Harald Soleim
Innleveringsdato

TITTELSIDE FOR HOVEDPROSJEKT

Rapportens tittel: Detektor av bildekvalitet for kjøretøy i bevegelse	*Dato:* *13.05.2024*
Forfatter(e): Øyvind Holter, Bjørn Ellingsen, Håkon Lervåg	*Antall sider u/vedlegg: 79* *Antall sider vedlegg: 91*
Studieretning: Dataingeniør og informasjonsteknologi	*Antall disketter/CD-er:* 0
Kontaktperson ved studieretning: Harald Soleim	*Gradering:* Ingen
Merknader: Ingen	

Oppdragsgiver: Counting Hero	*Oppdragsgivers referanse:* EB-3
Oppdragsgivers kontaktperson: Ruben Patel	*Telefon:* 920 43 919

Sammendrag:

Prosjektet omhandler deteksjon av ulike feil i bilder av dekk på kjøretøy i bevegelse. Ettersom det er bevegende objekt i stillestående bilder kan det fort forekomme uskarpheter. Målet med prosjektet var derfor å utvikle detektorer for "motion blur", utilstrekkelig belysning og vanndråper, i bilder. I tillegg lagde vi en nettside som oversiktlig viser bildene hvor slike feil har blitt detektert. Dette var av stor interesse for oppdragsgiver da slike forhold kan redusere kvaliteten på dataene deres og påvirke deteksjonen av detaljer på dekk, gjort med kunstig intelligens.

The project deals with the detection of various errors in pictures of tires on vehicles in motion. As there are moving objects in still images, blurring can likely occur. The goal of this project was therefore to develop detectors for motion blur, insufficient lighting and water drops, in pictures. Additionally we made a web page which displays the pictures where these mistakes have been detected in a clear way. This is of big interest for our client as it can reduce the quality of their data and influence the detection of details on tires, done with artificial intelligence.

Stikkord:

Deteksjon	KI, ML, DL	Terskelverdier

Høgskulen på Vestlandet, Fakultet for teknologi, miljø- og samfunnsvitenskap
Postadresse: Postboks 7030, 5020 BERGEN Besøksadresse: Inndalsveien 28, Bergen
Tlf. 55 58 75 00 Fax 55 58 77 90 E-post: post@hvl.no Hjemmeside: http://www.hvl.no

FORORD

Prosjektgruppen ønsker med dette å rette en stor takk til både oppdragsgiver og veileder. Dere har begge vært til veldig stor hjelp, både med det teoretiske og det inspirerende. Takk til oppdragsgiver for å ha tatt seg tid til å hjelpe med vanskelige tema, tilbudt arbeidsplass og for friheten vi fikk i oppgaven vår. Takk til veileder for gode råd til rapportskriving, og ikke minst for å være en fantastisk motivasjonsfaktor gjennom hele prosjektet.

ORDLISTE

Ord	Forklaring
Event	En hendelse som er aktivert av enten maskinen eller brukere. Signaliserer at noe har skjedd. Eksempelvis, når en bruker taster en knapp på tastaturet.
VSCode	Utviklingsverktøyet Visual Studio Code.
Strobelys	Strobelys er lys som sender ut en lyspuls hver gang kameraet tar et bilde, som kan minne om “blitz” i et vanlig kamera.
Neuron	Tilsvarende en node i et kunstig nevralt nettverk, altså kan man se på det som "prosess stasjoner" i en kunstig hjerne.
Kunstig nevralt nettverk	En kunstig databasert hjerne. Et nettverk av prosesseringskraft og data som brukes til å prosessere informasjon og ta beslutninger.
Translasjon	Forskyving / Forflytting.
Kant / Edge	Grensesnitt mellom front-end og back-end.
Terskelverdi	En terskelverdi er en grenseverdi som definerer skillet mellom to resultater. I vårt tilfelle vil dette være skille mellom dårlige og gode bilder.
“Motion blur”	Uskarpheter i et bilde tatt av et kamera, eller bilde av et objekt, i translasjon (bevegelse uten rotasjon) eller rotasjon
“Motion blur”-korrigering / “deblurring”	I rapporten skriver vi de to av og til om hverandre, men det er altså to benevninger for samme konsept. Uttrykkene er brukt for å forklare fjerningen av “motion blur” fra bilder.
Histogram	Grafisk illustrasjon av numeriske data fordelt på rektangulære søyler. Høyden på disse er antallet eller frekvensen av data som tilhører kategorien søylen representerer
Støy	Tilfeldig variasjon i pikselverdiene

FIGURLISTE

Figur 2.1: Skjermdump av hvordan Counting Hero teller biler og personer med bruk av KI..... 4
Figur 2.2: Problemsammendrag..... 5
Figur 2.3: Brukstilfellediagram..... 6
Figur 2.4: HTML-prototype av nettsiden..... 7
Figur 2.5: Utforming av oppgaven - iterasjon 2..... 9
Figur 2.6: Endelig problemstilling etter avgrensninger..... 10
Figur 2.7: Kunstig intelligens (KI /AI), maskinlæring (ML) og deep learning (DL)..... 12
Figur 2.8: Konvolusjon for å gjøre et bilde skarpere..... 13
Figur 2.9: Generator som prøver å generere et skarpt bilde ut fra det uskarpe bildet..... 14
Figur 2.10: Et eksempel på oppsampling og nedsampling..... 14
Figur 2.11: Uthenting av egenskaper med “ResBlocks”, oppsampling og nedsampling..... 14
Figur 3.1: Kanban-tavle for progresjonen vår i uke 3 og 4..... 18
Figur 3.2: Hvordan våre 2-ukers sprinter såg ut..... 18
Figur 3.3: Gantt-diagram som viser hvordan vi planla utviklingen vår..... 19
Figur 4.1: Systemarkitektur..... 22
Figur 4.2: Kantklasser - Klasser som ligger på kanten og er selve kjernen i prosjektet..... 24
Figur 4.3: Webklasser - Klassene i web-delen av prosjektet..... 26
Figur 4.4: Klassen “MedieManipulator”..... 27
Figur 4.5: “Mean”-funksjonen fra OpenCV biblioteket..... 28
Figur 4.6: Histogram som viser lysnivået til alle bildene vi mottok fra Counting Hero..... 29
Figur 4.7: Et bilde med strobelys..... 30
Figur 4.8: Et bilde uten strobelyset..... 30
Figur 4.9: Hvordan et oppkuttet bilde med bare bildekket kan se ut..... 30
Figur 4.10: Histogram over fordelingen av lysnivå etter at bildene er beskåret til bare bildekket.... 31
Figur 4.11: Histogram for skille mellom lavt og ikke-lavt lys for forskjellige deler av dekket..... 31
Figur 4.12: Hvordan vi deler opp de ulike delene av dekket..... 31
Figur 4.13: Fordeling av lysnivå når vi bare bruker den delen av dekket som har høyest lysnivå.... 32
Figur 4.14: Bildet som viser forskjellen på “motion blur” på øvre del og nedre del av dekket..... 33
Figur 4.15: Rotasjon + translasjon = dobbel hastighet på toppen av dekket..... 33
Figur 4.16: Variansformel..... 34
Figur 4.17: Normalisert histogram for mappen “Originale_Bilder” med bilder..... 34
Figur 4.18: Normalisert histogram for mappen “Lagt_til_Motion_blur” med bilder..... 34
Figur 4.19: Histogram over differansen i varians for bilder som inneholder bildekket..... 35
Figur 4.20: Histogram for variansen til bildene som bare viser toppen av dekkene..... 36
Figur 4.21: Eksempel på hvordan bilder som bare viser toppen av dekket ser ut..... 36
Figur 4.22: Histogram som viser et skille mellom differansen på toppen og bunnen av dekket..... 37

Figur 4.23: Histogram som viser varianseforskjeller mellom bilder tilhørende ulike mapper..... 38
Figur 4.24: Histogram som viser forskjellene i lysnivået for de ulike delene av dekkene..... 38
Figur 4.25: Histogram som viser resultater for variansen på våte dekk under lysnivå 75..... 39
Figur 4.26: Histogram som viser resultater for variansen på våte dekk over lysnivå 75..... 39
Figur 4.27: Eksempel på klart bilde..... 40
Figur 4.28: Eksempel på dårlig og tåkete bilde..... 40
Figur 4.29: Bilde som viser godt eksempel på mye vann..... 40
Figur 4.30: Bilde som viser godt eksempel på mye vann..... 40
Figur 4.31: Histogram for vanndråpe-detektoren..... 41
Figur 4.32: Generative adversarial networks model..... 42
Figur 4.33: Relasjonen mellom generatoren og diskriminatoren i cGAN-rammeverket..... 43
Figur 4.34: Resultater fra *Nah et.al* vs DeblurGAN..... 44
Figur 4.35: Hvordan rammeverket Shift-Net fungerer..... 44
Figur 4.36: En sammenligning av resultater mellom Shift-Net og andre konkurrerende metoder.... 46
Figur 4.37: Tidligere versjon av Counting Hero sin nettside..... 47
Figur 4.38: Bilde av nettsiden vår..... 48
Figur 4.39: Logo og søkeknapp på nettsiden vår..... 48
Figur 4.40: Flere søkefelt viser når man trykker på “søk” knappen..... 49
Figur 4.41: Objekttabell..... 50
Figur 4.42: Eksempel på brukerdefinert søk på nettsiden vår..... 51
Figur 4.43: Resultat av brukerdefinert søk..... 51
Figur 4.44: Et bildeobjekt som er flagget for lav belysning..... 51
Figur 5.1: Ett av to bilder som ble identifisert som falskt-positiv..... 57
Figur 5.2: Ett av to bilder som ble identifisert som falskt-positiv..... 57
Figur 5.3: Bilde som ble falskt identifisert som dårlig på grunn av skygge..... 58
Figur 5.4: Bilde som ble falskt identifisert som dårlig med bare hint av feil..... 58
Figur 5.5: Bilde som ble falskt identifisert som dårlig uten noen form for feil..... 58

Ligning 5.1: Formelen for beregning av nøyaktighet..... 59
Ligning 5.2: Formelen for beregning av presisjon..... 59
Ligning 5.3: Formelen for beregning av sensitivitet..... 59
Ligning 5.4: Formelen for beregning av F1-Score..... 60

Tabell 1.1: Strukturen i rapporten..... 3
Tabell 3.1: Risikoanalyse..... 20
Tabell 5.1: Terskelverdier..... 56
Tabell 5.2: Forvirringsmatrise..... 57
Tabell 5.3: Resultatmatrise for detektorene..... 60

INNHOLDSFORTEGNELSE

FORORD ... 3
1 INNLEDNING ... 1
1.1 Kontekst ... 1
1.2 Motivasjon ... 1
1.3 Prosjekteier ... 2
1.4 Problembeskrivelse og mål ... 2
1.5 Oppbygging av rapporten ... 3
2 PROSJEKBESKRIVELSE ... 4
2.1 Praktisk bakgrunn ... 4
2.1.1 Tidligere arbeid ... 4
2.1.2 Initiell løsnings-idé ... 5
2.2 Avgrensninger ... 8
2.3 Ressurser ... 11
2.4 Litteratur om problemstillingen ... 11
2.4.1 Kunstig intelligens, maskinlæring og dyp læring ... 11
2.4.2 "Motion blur" ... 12
2.4.3 ResBlocks, oppsampling og nedsampling ... 13
3 DESIGN AV PROSJEKTET ... 15
3.1 Forslag til løsning ... 15
3.1.1 Alternativ løsning 1 ... 15
3.1.2 Alternativ løsning 2 ... 15
3.1.3 Diskusjon av alternativene ... 16
3.2 Valgt løsning ... 16
3.3 Valg av verktøy ... 16
3.4 Prosjektmetodikk ... 17
3.4.1 Utviklingsmetodikk ... 17
3.4.2 Prosjektplan ... 19
3.4.3 Risikovurdering ... 20
3.5 Evalueringsplan ... 21
4 DETALJERT LØSNING OG PROSJEKTGJENNOMFØRING ... 22
4.1 Oppbygging ... 22
4.1.1 Backend - Kant ... 23
4.1.1.1 Bildeobjekter og lokal lagring ... 24
4.1.2 Nettside ... 25
4.1.3 Analyse i "MediaManipulator" ... 26
4.2 Lysdetektor ... 27
4.2.1 Terskelverdi for lysnivået ... 28
4.3 "Motion blur" detektor og Vanndråpedetektor ... 33
4.3.1 "Motion blur" detektor ... 33
4.3.2 Vanndråpedetektor ... 40
4.4 Deblurring ... 42
4.4.1 Generative Adversarial Network ... 42
4.4.1.1 DeblurGAN ... 43
4.4.2 Grouped Spatial-Temporal Shift ... 44
4.4.2.1 Shift-Net ... 45

4.5 Nettside....46
4.5.1 Design....*47*
4.5.2 Logo og søkefelt....*48*
4.5.3 Objekttabell....*49*
4.5.4 Resultat av søk på nettsiden....*50*
5 RESULTATER....52
5.1 Evalueringsmetoder....52
5.2 Evalueringsresultat....53
5.3 Prosjektresultat....55
5.3.1 Prosjektresultat og evaluering av detektorene....*55*
5.3.2 Prosjektresultat - "motion blur"-korrigering....*60*
5.3.3 Prosjektresultat - nettside....*60*
6 DISKUSJON....61
6.1 Valg av oppgave, avgrensninger og avvik....61
6.1.1 Valg av oppgave....*61*
6.1.2 Avgrensninger....*61*
6.1.3 Avvik....*61*
6.2 Valg av verktøy og arbeidsprosess....62
6.2.1 Valg av verktøy....*62*
6.2.2 Arbeidsprosess....*62*
6.3 Styrker og svakheter i endelig løsning....63
6.4 Forbedringer og andre valg....64
7 KONKLUSJON OG VIDERE ARBEID....65
7.1 Konklusjon....65
7.2 Videre arbeid....66
8 REFERANSER....67
9 VEDLEGG....71

1 INNLEDNING

Prosjektet tar utgangspunkt i et samarbeid mellom Counting Hero og Statens vegvesen der hensikten er å øke tilgangen på statistikk relatert til dekkskader på norske kjøretøy. Innledningsvis vil vi presentere konteksten og motivasjonen til oppgaven, prosjekteieren, problembeskrivelsen og mål, i tillegg til strukturen i rapporten.

1.1 Kontekst

De siste årene har anvendelsen av kunstig intelligens (KI) og spesielt generative modeller, slik som ChatGPT, hatt en stor økning [1]. Veksten kan nok skyldes både økt tilgjengelighet hos forbrukere og nye teknologiske fremskritt. Det vi kan si sikkert er at denne teknologien brukes på mange ulike områder i dag, og har blitt svært relevant for mange bedrifter. En av disse bedriftene er Statens vegvesen som har satset stort på digitalisering, hvor de har jobbet med å lage automatiseringer og applikasjoner for å hjelpe sjåfører [2].

Videre er det meget interessant å se på hvordan en kan bruke KI, eller nærmere bestemt maskinlæring, til å øke datamengden som samles inn i trafikken, med mål om å forbedre veisikkerheten. Som et resultat av disse målsetningene har vi utviklet et prosjektet sammen med Counting Hero, som spesialiserer seg på telling av kjøretøy og personer, samt dekkskanning, ved hjelp av KI [3]. Counting Hero bruker KI-deteksjon på direktestrømmer fra sanntidskamera for å fange opp skarpe bilder av dekk i bevegelse. Prosjektet vårt benytter seg av disse bildene.

Prosjektet kommer som følge av denne satsingen mellom Statens vegvesen og Counting Hero, og vil bidra til deteksjon av feil i bilder.

1.2 Motivasjon

Tidlig i prosjektet ble det tydelig at vi hadde stor frihet til å forme oppgaven slik vi ønsket, dermed bestemte vi oss for å ta på oss oppdraget. Muligheten til å utforske nye kodespråk og teknikker var også en stor motivasjonsfaktor for hele prosjektgruppen.

Prosjektet har flere interessante aspekter som gjør det motiverende å jobbe med. Et stort motivasjonspunkt er muligheten til å få dypere innsikt i ulike deteksjonsmetoder for feil i bilder, samt hvordan KI brukes til å korrigere feil med bilder. Med korrigering av feil i bilder refererer vi til “motion blur”-korrigering og “deblurring” som vil bli brukt om hverandre i rapporten. Uttrykkene er brukt for å forklare fjerningen av “motion blur” fra bilder. Videre vil gruppen få muligheten til å utforske flere ulike KI-modeller, se på forskjeller, samt fordeler og ulemper, for å kunne gi en bedre oversikt over anvendbare modeller innen feilkorrigering i bilder. Det er også interessant å undersøke hvordan vi kan redusere tiden det tar å oppdage en slik feil, og deretter forbedre kvaliteten på bildene.

1.3 Prosjekteier

Counting Hero [4] er et selskap som leverer kritisk informasjon fra sanntidsvideoer og lydstrømmer for nøyaktig telling. Bedriften har videobaserte løsninger som sporer personer, kjøretøy, dekk, og personer i kjøretøy. Ved bruk av maskinlæringsmodeller utvikler Counting Hero sensorsystemer som kan detektere detaljer som er vanskelig å oppdage for mennesker. Counting Hero er oppdragsgiveren til dette bachelorprosjektet og tjenestene de leverer kan deles opp i tre følgende kategorier:

Personteller for ombordstigning av ferge:

- Etter at manuell billettservice ble fjernet i Norge, har det vært en utfordring å holde orden på nøyaktig antall passasjerer ombord på fergene til enhver tid. Dette utgjør en betydelig problemstilling, da det er kritisk å ha nøyaktig informasjon i tilfelle en nødsituasjon som krever evakuering. Counting Hero tilbyr en løsning som benytter kameraer, sensorer og maskinlæring for å gi mannskapet en omfattende oversikt i sanntid.

Teller for kjøretøy på veien:

- Telling av kjøretøy på veien er en annen bruk av sensorsystemene til Counting Hero. Bedriften kan klassifisere kjøretøyene i forskjellige kategorier og lengder, og dermed trekke ut informasjon slik som blant annet typen kjøretøy.

Skanning av dekk:

- Det kan være ganske farlig dersom man er ute for å få skader på, eller generelt har feil dekkene sine. Dette er høyst relevant i Norge, med et veldig varierende klima. Tjenesten som Counting Hero tilbyr til vegvesenet er skanning av dekk for verdifull informasjon, slik som dekktype og eventuelle skader.

1.4 Problembeskrivelse og mål

I dagens løsning av dekkskanning kan bilder bli uskarpe på grunn av faktorer som blant annet “motion blur", utilstrekkelig belysning og vanndråper. Dette kan resultere i tap av data, da oppdragsgiver mangler en løsning for slike scenarier. Dagens ordning innebærer at utviklerne hos Counting Hero må utføre manuelle sjekker for å sikre at bildene som strømmer inn til databasene er i orden og ikke inneholder feil slik som faktorene vi nevnte over.

Problemstillingen vi da stiller oss er:

Hvordan kan vi automatisk detektere og varsle om feil på en klar og pålitelig måte for å redusere tiden det tar å manuelt kvalitetssjekke bildene?

Målet med prosjektet var å tilby en løsning som viste detekterte feil i bilder på en oversiktlig måte. Dette ønsket vi å oppnå ved å eksperimentere med ulike metoder vi kunne prosessere bilder på. Ideelt sett ville dette gitt oss tydelige og pålitelige terskelverdier for detektorene gjennom en iterativ tilnærming. En terskelverdi er en grenseverdi som definerer skillet mellom to resultater. I vårt tilfelle var dette skille mellom dårlige og gode bilder. Videre skulle vi implementere en brukervennlig nettside som ga oss en tydelig oversikt over kvaliteten på bildene. Ved å ha et varslingssystem kan vi redusere tiden det tar å manuelt kvalitetssjekke bildene.

1.5 Oppbygging av rapporten

Tabell 1.1: Strukturen i rapporten

1	Innledning	Gir en introduksjon til prosjektet vårt. Kontekst, motivasjon, prosjekteier samt problembeskrivelse og mål.
2	Prosjektbeskrivelse	Et dykk inn i ressurser og rammer rundt prosjektet.
3	Design av prosjektet	Forklaring av eventuelle løsninger og valg som er gjort for design og spesifisering av prosjektet.
4	Detaljert løsning	Presentasjon av endelig løsning.
5	Resultater	Resultater og evaluering.
6	Diskusjon	Diskusjon om ulike valg, og konsekvenser.
7	Konklusjon og videre arbeid	Våre tanker om sluttproduktet og muligheter for videre arbeid.
8	Referanser	Referanseliste.
9	Vedlegg	Vedlagte støttedokumenter. Derav, prosjekthåndbok, visjonsdokument, kravdokumentasjon og systemdokumentasjon.

2 PROSJEKBESKRIVELSE

Dette kapittelet gir en beskrivelse av den teoretiske bakgrunnen og vår tidlige visjon for prosjektet. Begge to er avgjørende for å få en helhetlig forståelse av prosjektet. Innledningsvis presenterer vi tidligere og forberedende arbeid, i tillegg til både initielle løsnings-idé og krav. Deretter tar vi opp avgrensninger som har blitt gjort for å avgrense oppgaven fra den initielle løsnings-idéen til en bestemt problembeskrivelse og til den endelige problemstillingen i kapittel 1.4. Til slutt viser vi hvilke ressurser vi har brukt og presenterer litteraturen vi har funnet for problemstillingen.

2.1 Praktisk bakgrunn

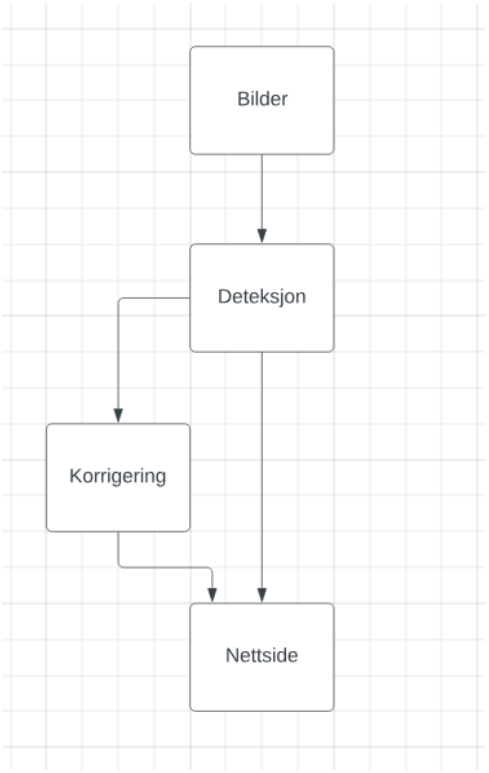
2.1.1 Tidligere arbeid

Gruppen arbeidet med et prosjekt som ikke hadde noen tidligere utvikling i Counting Hero. Systemet vi har utviklet tar utgangspunkt i de funksjonelle kravene som ble utarbeidet basert på behovene til oppdragsgiver. Disse kravene er beskrevet i visjonsdokumentet. Det var interessant for oss å prøve å gjøre prosjektet så tilpasningsdyktig som mulig med relatert arbeid i bedriften. Med relatert arbeid refererer vi til de eksisterende systemene for deteksjon av kjøretøy ved hjelp av sanntidskameraer. At vi ønsket å være tilpasningsdyktige påvirket også vårt valg av verktøy. Figur 2.1 viser hvordan Counting Hero detekterer og klassifiserer ulike kjøretøy og personer med et av deres kameraer utplassert på en fergekai. Videre var det også meget relevant for oss å undersøke ulike modeller fra GitHub relatert til "deblurring". Vi ønsker å utforske disse modellene for å bygge opp relevant informasjon som bedriften kan benytte i fremtiden.

Figur 2.1 Skjermdump av hvordan Counting Hero teller biler og personer med bruk av KI [4]

2.1.2 Initiell løsnings-idé

Vi planla i hovedsak å utvikle et system som fungerer som en "pipeline". Figur 2.2 illustrerer hvordan systemet skulle ta inn bilder og analysere de for kvalitetsbrudd slik som “motion blur”, dårlig lys og urent kamera. Disse bildene var av dekk på kjøretøy som kjørte forbi kameraene til Counting Hero. Ved deteksjon av feil med bildekvaliteten ønsket vi å gjennomføre "motion blur"-korrigering på bildet. Etter at deteksjon og eventuell korrigering av feil var gjennomført, ville både før- og etterbilder, sammen med relevant informasjon, bli presentert på nettsiden. Bilder med kvalitetsbrudd ville blitt tydelig markert for identifisering på nettsiden.



Figur 2.2: Problemsammendrag. Hentet fra visjonsdokument.

2.1.3 Initielle krav

For å tydeliggjøre det systemets opprinnelige krav på en mer visuell og forståelig måte, utarbeidet vi et brukstilfellediagram basert på de funksjonelle kravene som er beskrevet i kapittel 5 i visjonsdokumentet. Figur 2.3 viser en helhetlig representasjon av vårt opprinnelige system, både på web og “kant”. I dette prosjektet brukte vi ordet "kant" for å beskrive den delen av koden som tar inn bilder, prosesserer dem og lagrer dem lokalt.

Kanten til Counting Hero er den delen av prosessen som utføres ved kameraet. Denne delen prosesserer bildene direkte og sender dem til databasen for videre analyse. Koden for

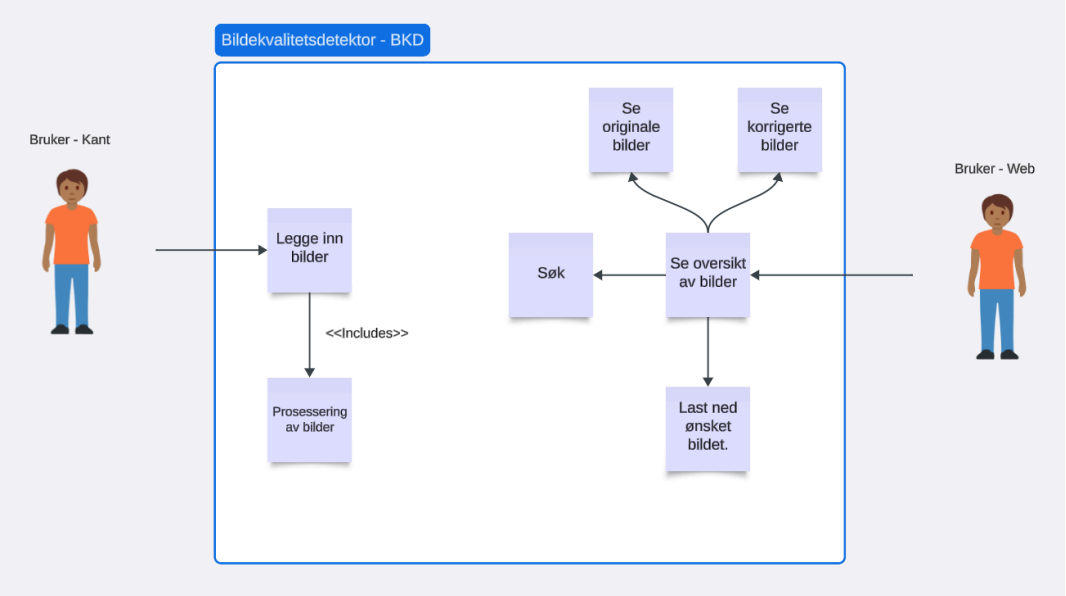
prosesseringen i dette prosjektet er tenkt å være plassert i kanten til Counting Hero. Dette var en beslutning som ble tatt tidlig i prosjektet. Om det er nødvendig å plassere seg i kanten eller ikke, kommer an på hvilken “deblurrings”-kode som blir brukt. Dette blir diskutert nærmere i kapittel 4.

Som figur 2.3 viser, har brukeren i kant mulighet til å "Legge inn bilder". Etter at bildene er lagt inn, blir de prosessert av vårt system. Prosesseringen av bildene har visse krav som den skal oppfylle. Den skal:

- Kunne ta imot bilder av bildekk for deteksjon.
- Kunne detektere om et bilde med “motion blur”, dårlig lys og urent kamera.
- Kunne korrigere bilder med “motion blur”
- Lagre bilder i lokal lagring.

Som figur 2.3 viser, har brukeren på web muligheten til å "Se oversikt av bilder" på nettsiden. Den interaktive delen av prosjektet er nettsiden som brukere kan benytte for å se bildene som er blitt analysert for dårlig lys, "motion blur" og urent kamera. Nettsiden har visse krav som den må oppfylle:

- Oversikt over alle instanser av kjøretøy med relevant informasjon.
- Brukerdefinert søking etter ID, dato, klokkeslett, sted eller flagg.
- Kolonne med oversikt over originale bilder
- Kolonne med oversikt over korrigerte bilder
- Knapp for nedlasting av bilder.

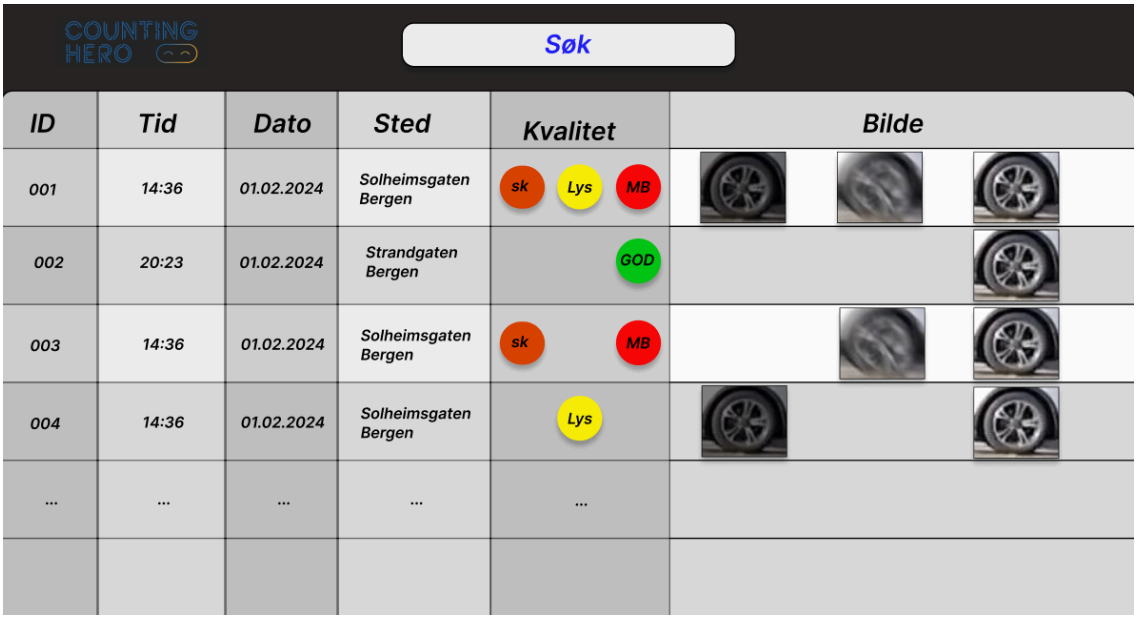


Figur 2.3: Brukstilfellediagram

Videre har vi laget en HTML-prototype av nettsiden vår basert på de initielle kravene. Figur 2.4 viser informasjonen som blir vist frem på nettsiden. Dette inkluderer identifikasjonsnummer (ID), tidspunkt, dato og sted for bildet som er tatt. Disse dataene kan søkes etter i søkefeltet for å gjøre det enklere for brukerne å navigere på nettsiden.

En sentral funksjon i denne prototypen er definisjonen av bildekvalitet. Disse indikatorene gir en visuell representasjon av det som er oppdaget i bildet, og markerer eventuelle feil eller mangler som kan påvirke bildekvaliteten. Indikatorene er "GOD" for god bildekvalitet, "SK" for skittent/urent kamera, "Lys" for dårlig belysning, og "MB" for "motion blur".

Videre viser prototypen både det originale bildet og det prosesserte bildet. Det prosesserte bildet inkluderer eventuelle korrigeringer som er gjort for å forbedre bildet.



Figur 2.4. HTML-prototype av nettsiden

Vår initielle løsningsidé hadde en del problemer som vi identifiserte utover prosjektet. Vi hadde dermed behov for å avgrense og gjøre endringer for å komme frem til den endelige problemstillingen.

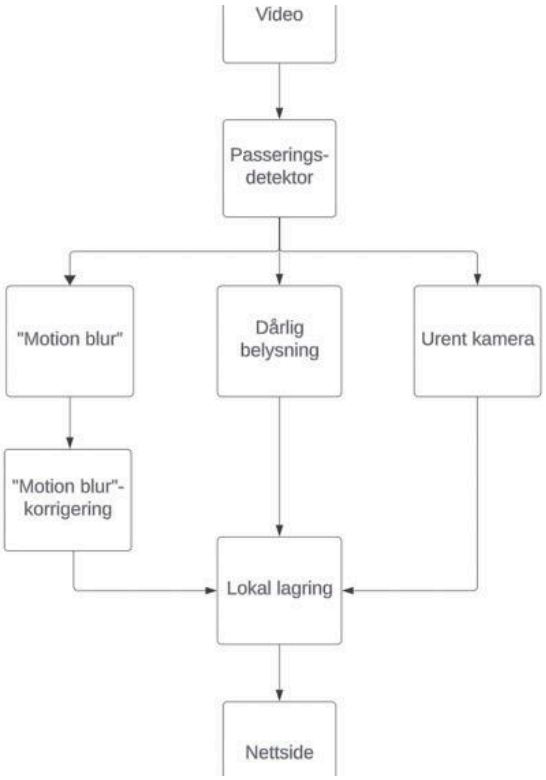
2.2 Avgrensninger

Counting Hero tilbød oss en åpen oppgave med stor grad av frihet når det gjelder gjennomføring. Opprinnelig var hovedfokuset i oppgaven rettet mot "motion blur"-korrigering. Overskriften fra prosjektkatalogen da gruppen valgte oppgaven var "Korrigering av «Motion Blur» på dekk for kjøretøy i bevegelse". Ønsket fra oppdragsgiver var å undersøke ulike modeller som vi fant på internett, hovedsakelig GitHub [25], og evaluere disse på Counting Hero sine bilder av dekk.

"Motion blur"-korrigering har allerede et bredt utvalg av tilgjengelige KI-modeller [5] på GitHub. Dette ville ført oss inn i et forsknings- og eksperimenteringsorientert spor, hvor målet var å evaluere og velge den mest effektive modellen for korrigeringen. Etter tidlige diskusjoner med oppdragsgiver og veileder, ble det klart at vi burde vurdere et større innslag av utviklingsarbeid. Vi valgte derfor å gjøre noen endringer i oppgaven og avvike fra den først tenkte problemstillingen.

Ved utformingen av systemet utarbeidet vi en grunnleggende plan for utviklingen. Vi trengte en bedre forståelse av problemet vårt og hva som krevdes for å løse det. Derfor opprettet vi spesifikke klasser for videosegmentering. På dette stadiet i utviklingen brukte vi ikke de bildene av dekk som vår endelige løsning benytter, da vi ennå ikke hadde fått tilgang til disse. Vi søkte på nett og fant en video [43] med passende lysforhold som et utgangspunkt for å skaffe bilder av kjøretøy å jobbe med. Formålet med å bruke en oppkuttet video samt utvikle disse klassene var å parallellisere arbeidet vårt for å øke effektiviteten i utviklingsprosessen. På dette tidspunktet hadde vi satt opp prosjektet vårt slik at vi kunne begynne å utvikle nettsiden med midlertidige bilder, samtidig som vi begynte å arbeide med deteksjonen på bildene som vi kuttet opp fra denne testvideoen.

Neste iterasjon av problemstillingen vår inkluderer videosegmenteringen som vi utviklet. Figur 2.5 viser en kant som segmenterer en video til individuelle bilder av kjøretøy og en “Passerings-detektor" som kunne avgjøre om et bilde inneholdt et kjøretøy eller ikke. Planen vår var at vi skulle behandle, detektere og korrigere bilder lokalt på datamaskinen. Med denne tilnærmingen ble oppgaven vår omformet til å skape et nytt system som ikke bare fokuserte på "motion blur"-korrigering, men også på deteksjon av feil og visning av resultatene på en nettside.

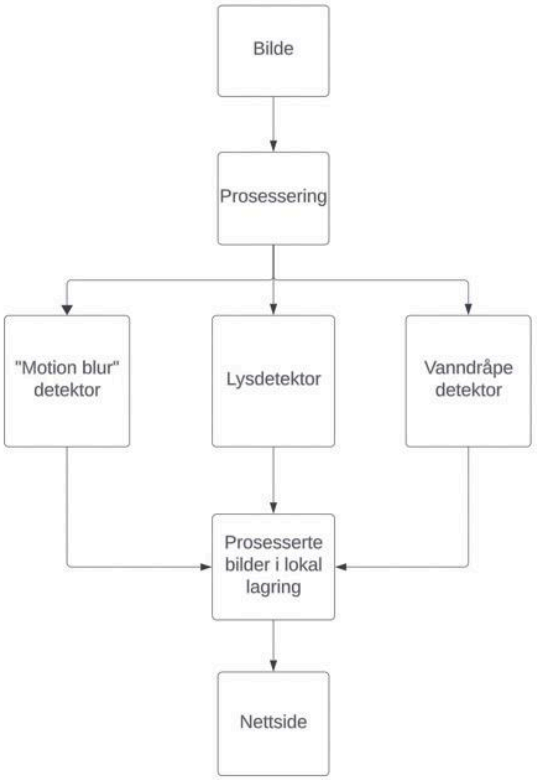


Figur 2.5: Utforming av oppgaven - iterasjon 2

Etter noen ukers arbeid så vi behovet for å avgrense oppgaven ytterligere og justere fokuset. Vi hadde nå fått tilgang til bildene av dekk som vår endelige løsning bruker. I samråd med oppdragsgiver bestemte vi oss for å legge større vekt på deteksjon ved å fastsette terskelverdier, ettersom dette området var av stor interesse for dem.

Videosegmenteringen ble utviklet for testing samt for å effektivisere utviklingen. Dette var ikke relevant for den endelige løsningen. Dette innebar å sette mesteparten av videosegmenteringen til side, da Counting Hero også allerede hadde systemer for å detektere kjøretøy samt ta bilder av dem. Utviklingen av videosegmenteringen var ikke bortkastet da gruppen kunne ta med seg videre flere av teknikkene som ble brukt til den endelige problemstillingen.

Figur 2.6 viser den endelige problemstillingen og de klargjorte avgrensningene. Alle parter ble enige om at vi nå hadde en mer omfattende oppgave som fortsatt oppfylte de opprinnelige kravene som var interessante for Counting Hero og inneholdt en god mengde utviklingsarbeid.



Figur 2.6: Endelig problemstilling etter avgrensninger

Som figur 2.6 viser, avgrenset vi oppgaven ved å droppe urent kameradetektor for vann, snø og skitt. Gruppen besluttet at det ville kreve trening av en egen KI-modell for å oppnå pålitelige resultater. Dette var fordi det var for mange variabler som kunne gjøre kameraet urent. En slik detektor ville altså kreve for mye data og lengre treningsperioder enn vi hadde tid til. I stedet for urent kamera avgrenset vi prosjektet til å omfatte to hovedfeil: "motion blur" og dårlig lys. Ettersom mange bilder fra oppdragsgiver inneholdt mye vann, ble en vanndråpedetektor også aktuell senere i prosjektet. Vanndråpene gjorde det vanskelig å skille bilder med og uten “motion blur” fra hverandre, og denne detektoren ble derfor relevant.

Videre, slik det er illustrert i figur 2.6, er implementering av "motion blur"-korrigering fjernet fra den endelige løsningen. Dette besluttet vi ettersom implementering av dette var noe som vi så på som urealistisk gitt tidsrammene for prosjektet. Vi bestemte oss i stedet for å avgrense oss til å undersøke potensielle KI-modeller for "motion blur"-korrigering. Disse undersøkelsene kan gi bedre innsikt til bedriften for eventuell fremtidig videreutvikling.

I den endelige løsningen lagrer vi de prosesserte bildene lokalt som vist i figur 2.6. Counting Hero benytter for tiden et databasesystem som vi valgte å ikke integrere i vår utvikling, ettersom dette ikke var relevant for å oppnå målet for oppgaven.

2.3 Ressurser

For å gjøre prosjektet vårt så vellykket som mulig, sørget Counting Hero kontinuerlig for å gi oss nyttige ressurser. For det første ga de oss data til testformål for å sikre at systemet vårt fungerte korrekt og ga ønskelige resultater.

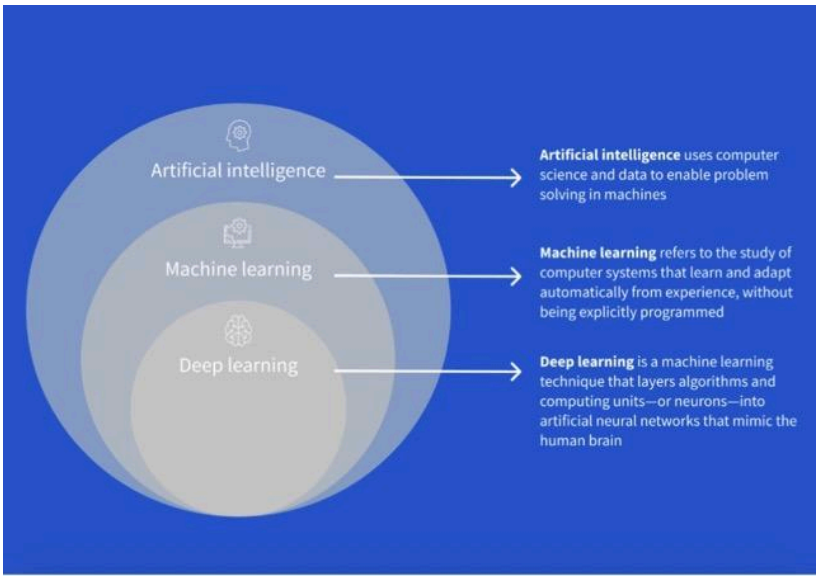
For det andre tilbød Counting Hero oss et sted å arbeide en gang i uken. Dette hjalp med å koble oss til arbeidsmiljøet og forkortet avstanden mellom oss og oppdragsgiver, i tilfelle vi hadde spørsmål om prosjektet. Bedriften var også i stand til å gi innsikt i noen av bibliotekene vi jobbet med og kodespråket vi benyttet som de kjente godt til. Dette innebar blant annet PyTorch [6] og kodespråket Python som begge er flittig brukt hos Counting Hero. Vi valgte derfor å benytte disse for å prøve å være så tilpasningsdyktige til bedriften som mulig.

2.4 Litteratur om problemstillingen

Dette kapittelet er dedikert til å forklare vanskelige begreper som kommer til å bli relevante i prosjektet. Mesteparten er forklaringer for begrep knyttet til “deblurring”, som blir relevant i kapittel 4.

2.4.1 Kunstig intelligens, maskinlæring og dyp læring

For å bedre forstå informasjonen rundt «deblurring» i oppgaven vår, er det nyttig å definere kunstig intelligens (KI), maskinlæring (ML) og dyp læring (DL) slik de er vist i figur 2.7. Kunstig intelligens er det overordnede begrepet som først oppsto og omfatter maskiner som etterligner menneskelig intelligens, for eksempel taleforståelse og selvkjørende biler [7]. Maskinlæring, som er en del av KI, automatiserer byggingen av analytiske modeller ved å lære av data, uten å bli eksplisitt programmert [7, 8]. Dyp læring, en videreutvikling av maskinlæring, er hvor modeller består av flere lag med algoritmer og dataenheter (neuroner) som danner kunstige nevrale nettverk, etterlignende en menneskelig hjerne [8]. Dette gir modellen større frihet og potensial i sine svar.

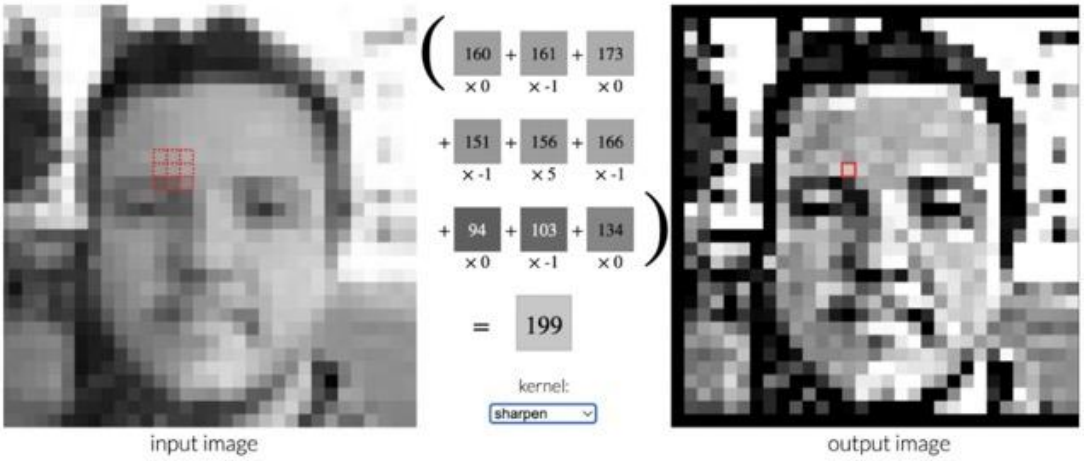


Figur 2.7: Kunstig intelligens (KI / AI), maskinlæring (ML) og deep learning (DL) [8]

2.4.2 “Motion blur”

“The blurring process is characterized by the relative rotation or translation between cameras and objects within camera lens exposure time. [9]”

"Motion blur" refererer til uskarpheten som oppstår i et bilde tatt av et kamera eller et objekt i bevegelse, enten det er translasjon (bevegelse uten rotasjon) eller rotasjon. Det har vært mange forsøk på å løse problemet med “motion blur”, og noen av disse vil være relevante i denne oppgaven. Vi skiller helt generelt mellom blind og ikke-blind “motion blur”-korrigering, hvor hovedforskjellen ligger i om “blur kernel”-en er kjent eller ikke. En “blur kernel” [10, 11] er en matrise med verdier som kan brukes i både “blurring” og “deblurring” av bilder. Verdiene i denne matrisen avhenger av hva man ønsker å oppnå, noe som vil si at dersom man har et uskarpt bilde, vil ofte disse “blur”-verdiene være ukjente, og vi må prøve å estimere de for å få et skarpt bilde. Prosessen kalles en konvolusjon [11] og er visualisert i figur 2.8. Konvolusjon kan forklares som en matematisk operasjon hvor man multipliserer en matrise av pikselverdier i et område, eksempelvis 3x3, i et bilde, med en “blur” matrise, og legger til resultatet i den første matrisen. Hver piksel blir multiplisert med alle sine naboer og en “blur” matrise som varierer etter hvilken effekt man ønsker.

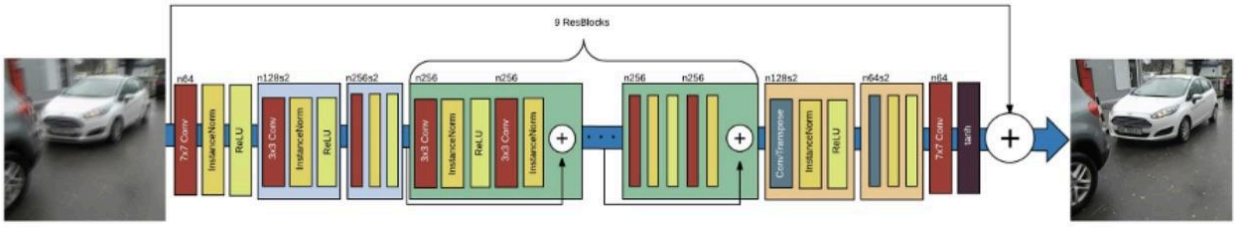


Figur 2.8: Konvolusjon for å gjøre et bilde skarpere. 3x3 område (matrise) ganget med blur kernel (matrise) for å få skarpere bilde [10]

Blind [9] og ikke-blind [12] "deblurring" er to teknikker innen bildeprosessering som har til hensikt å gjenopprette skarpe bilder. Blind "deblurring" håndterer tilfeller der "blur-kernel"-en er ukjent, mens i ikke-blinde tilfeller er den kjent. Førstnevnte er spesielt relevant i dagens kontekst, særlig innen fotografering for å forbedre bildekvaliteten i blant annet overvåkingskamera og smarttelefoner [9]. Dette er også metoden som er mest relevant for oss, da "blur kernel"-en ikke er kjent i bilder som blir tatt utenfor helt kontrollerte rammer, i motsetning til bilder som er bevisst gjort uskarpe. Med kontrollerte rammer refererer vi til bilder som blir tatt i situasjoner hvor alle variabler kan kontrolleres, og vi klarer å bestemme hvor mye "motion blur" som kommer på bildene.

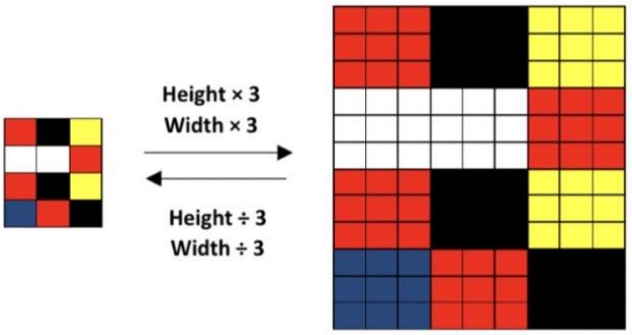
2.4.3 ResBlocks, oppsampling og nedsampling

Når vi omtaler oppbyggingen av dyp læring i modeller for korrigering av "motion blur", er det noen viktige begreper å være klar over. En av dem er restnettverk [13], vanligvis kjent som «ResBlocks». Dette er grunnleggende byggeblokker i arkitekturen for begge de to metodene vi skal presentere senere i kapittel 4.4. I figur 2.9 kan vi se to grønne rektangler som er eksempler på to "ResBlocks". Disse blokkene består av en mengde operasjoner for behandling av bilder. Hver rektangel i de grønne boksene er eksempler på slike operasjoner. Vi kan også observere at de to grønne "ResBlock"-ene har hver sin sorte pil. Dette er "shortcut connections", eller snarveier. En slik snarvei tillater informasjonen som passerer gjennom generatoren å hoppe over byggeblokken den ellers måtte passere gjennom. På slutten av byggeblokken blir informasjonen som hoppet over blokken, og informasjonen som passerte gjennom byggeblokken konkatenert med hverandre. Hensikten med en slik funksjon er å minimere tap av viktig informasjon ved å la det hoppe fremover til der det skal brukes.



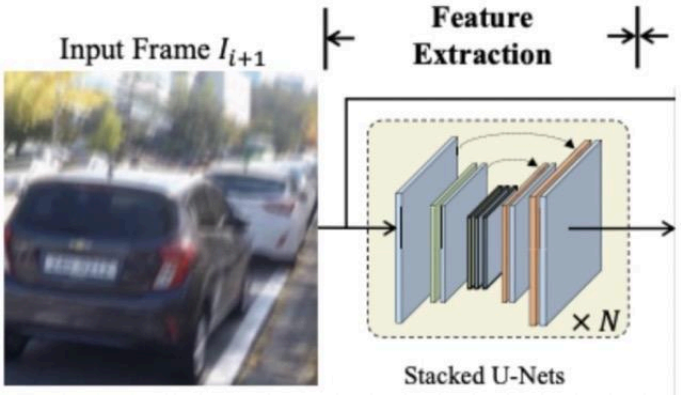
Figur 2.9: Eksempel på en generator som prøver å generere et skarpt bilde ut fra det uskarpe bildet med bruk av blant annet «ResBlocks» [14].

Oppsampling og nedsampling er to typer operasjoner som er avgjørende for å justere den romlige («spatial») oppløsningen av bildet. Ved en nedsampling betyr dette å redusere både informasjonen og antall piksler i bildet. Dette er gunstig for å gjøre bildene billigere og mer effektive å lagre, samt flytte på. En oppsampling er det motsatte, hvor vi vil øke pikslene for å kunne zoome inn på spesifikke plasser. Figur 2.10 viser hvordan dette kan se ut.



Figur 2.10: Et eksempel på oppsampling og nedsampling [15].

Sammen utgjør disse to komponentene, samt «ResBlocks», en viktig rolle i en del operasjoner relatert til "deblurring", slik som uthenting av informasjon i bilder. I figur 2.11 kan vi se et eksempel på dette. Til venstre er et uskarpt bilde som skal inn i en prosess til høgre, for uthenting av egenskaper. De blå blokkene til høgre er "ResBlocks", grønt er nedsampling, og oransje er oppsampling. Prosessen med opp- og nedsampling gjør det mulig å flytte og lagre bildene, samt forstørre og forminske områdene vi må se nærmere på. Informasjonen som blir hentet ut spiller en viktig rolle for restaureringen av bilder.



Figur 2.11: Uthenting av egenskaper med "ResBlocks", oppsampling og nedsampling [16].

3 DESIGN AV PROSJEKTET

Hvilke fremgangsmåter og metoder vi skulle benytte for å løse problemstillingen vår har vært en av de større utfordringene med prosjektet. Vi anså det som helt sentralt å avklare disse områdene for å jobbe godt og strukturert som en gruppe. En diskusjon rundt hvordan vi har valgt å gå frem, i tillegg til valg av verktøy, prosjektmetodikk og evalueringsplan, er hva vi vil presentere i dette kapittelet.

3.1 Forslag til løsning

I den opprinnelige oppgaven var fokuset på å undersøke, teste og evaluere ulike KI-modeller til korrigering av «motion blur». Det endelige prosjektet vi nå har utviklet har fortsatt en del undersøkelser, men med fokus på deteksjon av "motion blur" i stedet for korrigering. I tillegg vil vi detektere både dårlig belysning og vanndråper. Det er viktig å få til gode deteksjoner, og hvilke metoder vi bruker vil være avgjørende i å redusere tiden det tar å kvalitetssjekke bildene. Under er to løsninger vi diskuterte å anvende, i et forsøk på å lage gode og pålitelige detektorer.

3.1.1 Alternativ løsning 1

Det første alternativet vi vurderte var å utvikle en god og veldokumentert plan med kjente metoder som vi ville teste og som vi så for oss ville gi tilfredsstillende resultater. Vi ville først gjennomført et ressurssøk etter metoder som tidligere har blitt benyttet for å håndtere problemstillingene relatert til dårlige bilder, samt metoder som vi eventuelt hadde ansett som effektive. Dette ressurssøket ville omfatte litteratur fra nettet, kontakt med oppdragsgiver og en gjennomgang av dokumentasjon. Vi ville deretter evaluert disse metodene basert på deres potensial til å løse våre spesifikke utfordringer med «motion blur», dårlig belysning og vanndråper. Evalueringen ville inkludert vurdering av metodens relevans, pålitelighet, hvor komplekst det er å implementere samt tidligere suksessrater i lignende situasjoner. Basert på denne evalueringen ville vi deretter implementert og testet de mest lovende metodene i vårt eget program.

3.1.2 Alternativ løsning 2

Et annet interessant alternativ var å benytte en litt mer improvisert fremgangsmåte. I denne ville vi analysert resultater når vi fikk dem, samt aktivt prøvd å løse problemene når de oppstod. Med andre ord hadde vi ikke nødvendigvis en oversikt over metoder vi skulle teste og evaluere, men ville funnet dem underveis dersom de passet til problemet vi stod med. En slik løsning ville også kunne bety at vi utviklet hjelpemidler underveis, som ikke nødvendigvis var dokumenterte metoder. Dette kunne potensielt gitt forbedret resultat ettersom problemene vi møtte på og metodene vi utviklet var tilpasset de bildene som vi jobbet med. Løsningen var også ganske aktuell for oss ettersom vi innledningsvis stod med ganske begrensede kunnskaper innenfor temaene vi jobbet med i oppgaven.

3.1.3 Diskusjon av alternativene

Løsningsmetode 1 innebærer å utvikle en strukturert plan basert på godt dokumenterte metoder. Dette kunne være fordelaktig for å kunne ta utgangspunkt i en klar fremdriftsplan og dra nytte av etablerte metoder med dokumentert suksess. En slik tilnærming ville sørget for at vi hadde gode grunnlag og ressurser for valgene vi hadde gjort, og gruppen kunne gitt veldokumentert innsikt i metodene som hadde vist seg å fungere. Imidlertid kunne denne metodikken begrenset kreativiteten til teamet, og mangelen på fleksibilitet kunne begrense mulighetene for å utforske nye løsninger. Ettersom gruppen ville tilegnet seg mange nye kunnskaper gjennom prosjektet, kunne det å binde seg til en strukturert plan virket mot sin hensikt. Dette er fordi gruppen sannsynligvis ville sett mange nye løsninger underveis som ikke var en del av den opprinnelige oversikten av metoder som vi skulle teste.

Løsningsmetode 2 involverte derimot en mer improvisert tilnærming til problemet. Løsningen kunne bidratt til å opprettholde fokus på å oppnå resultater samt øke fleksibiliteten til gruppen når det gjelder å håndtere endringer og utfordringer underveis i prosjektet. Denne tilnærmingen kunne oppmuntret til kreativ tenking og eksperimentering, samt gitt muligheter for rask tilpasning til nye funn eller behov. Likevel kunne en improvisert tilnærming ført til ustrukturert arbeid og noe manglende dokumentasjon, noe som kunne påvirket kvaliteten på prosjektet.

3.2 Valgt løsning

På bakgrunn av evalueringen i forrige avsnitt, besluttet gruppen til slutt å ta i bruk løsning 2. Til tross for at vi kunne fått mindre dokumentasjon og referanser å vise til, vurderte vi fleksibiliteten til å kunne gjøre vurderinger og evalueringer underveis til å være av for stor betydning til å ikke gå for løsningen. Dette var også spesielt relevant, gitt vår begrensede kunnskap på de aktuelle områdene i begynnelsen av prosjektet. Selv om løsningen åpnet for improviserte metoder som anses som relevante for det daværende problemet, er det likevel viktig å poengtere at vi også har benyttet noen dokumenterte metoder.

3.3 Valg av verktøy

Utviklingsverktøy

Da vi valgte å kode i Python, stod vi mellom PyCharm[17] og VSCode [18] som våre utviklingsverktøy, da disse er henholdsvis den mest populære IDE-en (Integrated Developer Environment) og "Code Editor"-en for Python utvikling [19]. Forskjellen mellom en IDE og en "Code Editor" er tradisjonelt sett at en IDE vil ha flere funksjoner, slik som integrert feilsøking og automatisk kodeutfylling [20].

Kompatibilitet med bibliotekene PyTorch [6] og OpenCV [21], samt rammeverket Flask [22], er faktorer vi vurderte som avgjørende, noe både PyCharm og VSCode har. PyTorch og OpenCV var

viktige for prosjektet ettersom de inneholdt nødvendige funksjoner for prosessering av bilder, slik som “mean”, “imread” og “var”. Til tross for at begge er veldig gode utviklingsverktøy som passer prosjektet vårt, falt til slutt valget på VSCode ettersom alle i gruppen hadde tidligere erfaringer med den. VSCode er også et lettvektig utviklingsverktøy som ga oss stor fleksibilitet i arbeidet, i tillegg til å være mer ytelseseffektivt [19].

Samarbeidsverktøy

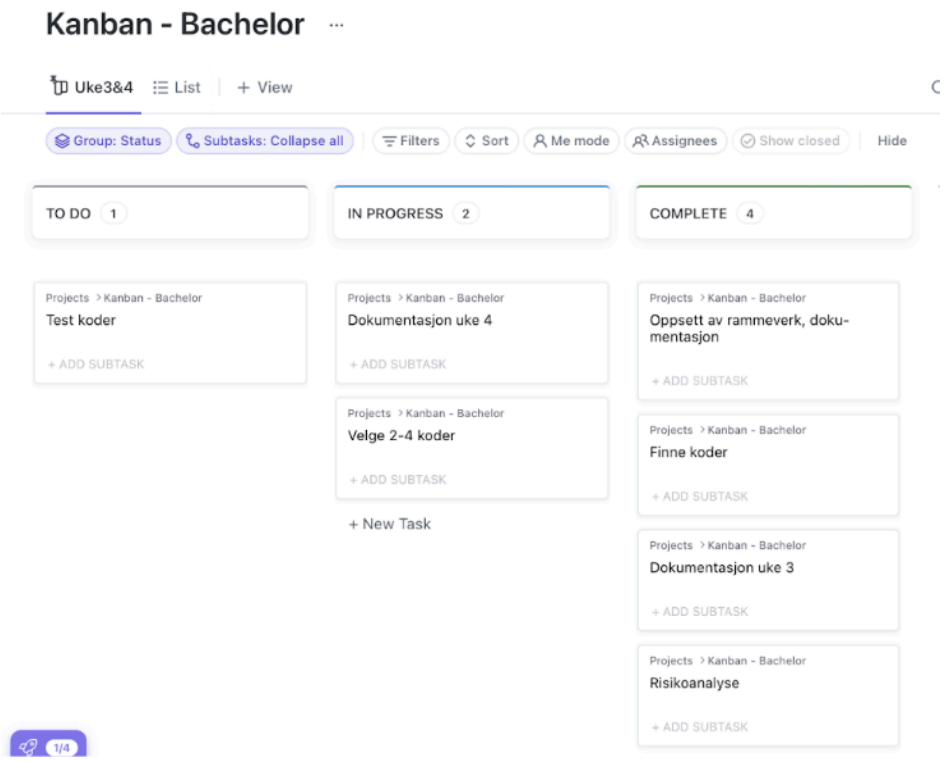
Gjennom prosjektet har vi jobbet synkront både på arbeidsplassen og campus, men også asynkront hjemmefra. Av den grunn har vi benyttet oss av en rekke verktøy for å gjøre prosjektarbeidet lettere. Vi har brukt Discord [23] og Messenger [24] for å avtale møter og generell kommunikasjon, mens vi har brukt både Github [25] og Github Desktop [26] for å samarbeide med koden. Når det gjelder dokumentasjon, har vi benyttet Google Disk [27] og andre Google-apper, slik som Google Dokumenter [28]. Avslutningsvis har vi brukt ClickUp [29] for å administrere oppgavene våre ved hjelp av en Kanban-tavle, og Lucidcharts [30] for skaping av visuelle modeller.

3.4 Prosjektmetodikk

3.4.1 Utviklingsmetodikk

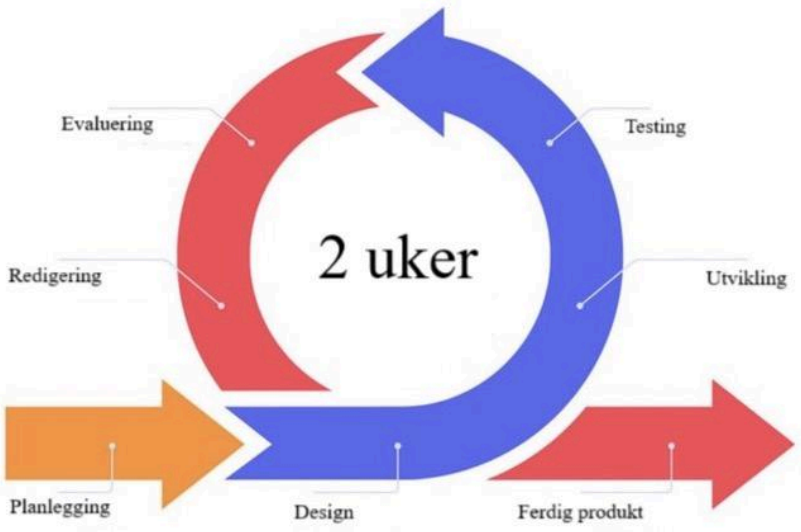
Oppskriften på en vellykket oppgave begynner ofte med en god plan, derfor valgte vi å benytte oss av agile metoder. Fra tidligere hadde gruppen god kjennskap til Scrum, og en del erfaring med bruk av Kanban-tavler. Begge disse agile metodene ble til dels brukt i prosjektutviklingen, da de tilbød ulike fordeler [31].

Vi valgte hovedsakelig å benytte oss av Kanban-tavler i prosjektet på grunn av den betydelige fleksibiliteten det tilbød. Dette var spesielt viktig for oss, da vi innledningsvis manglet en del kunnskap om prosjektet, som gjorde det utfordrende å forplikte oss til en strukturert plan med faste tidsrammer slik Scrum er mer egnet til. I tillegg var Kanban-tavler bra tilpasset oss ettersom prosjektet foregikk i en kontinuerlig flyt hvor vi satt opp tavler for toukers perioder av gangen slik som illustrert i figur 3.1. Denne segmenteringen ga prosjektet god gjennomførbarhet, og gruppen opplevde mestringsfølelse gjennom hele prosjektperioden.



Figur 3.1: Kanban-tavle for progresjonen vår i uke 3 og 4

Videre tok vi også noen læringspunkter fra Scrum-metoden ved å holde daglige møter for å vise fremgang og diskutere eventuelle utfordringer. Dette simulerer iterasjonene gjennom en sprint i Scrum, hvor en sprint er en toukers periode fra Kanban-tavlen vår. Vi innførte denne tilnærmingen ettersom vi i perioder måtte jobbe på forskjellige områder av prosjektet - en arbeidsform som Scrum er mer egnet for. Gruppen tok derimot bare lærdom fra metoden fordi vi ikke ønsket å miste fleksibiliteten i arbeidet.



Figur 3.2: Hvordan våre 2-ukers sprinter såg ut [32]

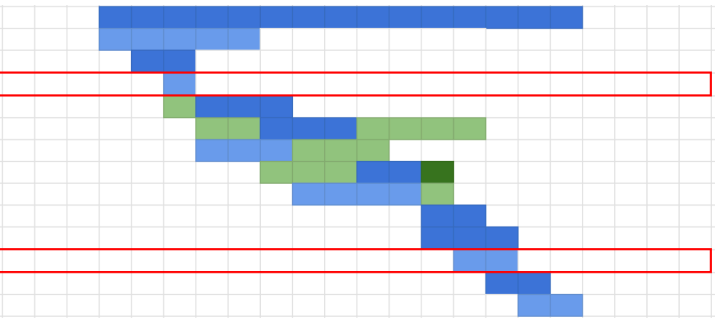
Arbeidet vårt kan illustreres med figur 3.2 som viser en toukers sprint som startet med planlegging, og gikk deretter over i design, utvikling og testing. Dette er normalt hva vi gjorde innad i gruppen på våre egne møter, mens evaluering og redigering ble gjort i lag med oppdragsgiver. Vi hadde kontinuerlig gjennom hele prosjektet en nær kontakt med oppdragsgiver, og vi hadde ukentlige møter i enkelte perioder, og møter annenhver uke ellers. Dette fungerte veldig bra for oss ettersom det har gitt oss anledning til å få jevnlig evaluering fra oppdragsgiver gjennom hele prosjektet og sikret at vi jobbet med relevante metoder.

3.4.2 Prosjektplan

En annen metode som ble brukt i prosjektet er et Gantt-diagram [33]. Ved å dele inn prosjektet i ulike aktiviteter slik vi ser i figur 3.3, ble denne metoden nyttig for å holde orden på tidsbruk og fremgang. Diagrammet ble tidlig utviklet, og fungerte som en overordnet plan med forbehold om endringer gjennom prosjektets løp.

Ettersom vi manglet en del kunnskap om prosjektet i oppstartsfasen og dermed måtte gjøre justeringer underveis, brukte vi farger for å holde orden på diagrammet. De lyseblå og mørkeblå feltene representerer det opprinnelige tidsrommet vi planla å utføre aktivitetene i, mens grønne felt indikerer oppdaterte tidsrom. Lysegrønne felt betyr at det opprinnelig ikke var planlagt å jobbe med den valgte aktiviteten i dette tidsrommet (hvitt felt), men det ble likevel gjort på grunn av frem- eller forskyvninger i tidsplanen. Mørkegrønne felt indikerer at det var planlagt å jobbe med den valgte aktiviteten i dette tidsrommet (blått felt), men det ble ikke gjort på grunn av for- eller fremskyvninger i tidsplanen. De mørke- og lyseblå radene har bare alternerende farger for å skille dem fra hverandre.

20	**Utvikling**	**Alle**	**105 dager**	**29.01.24**	**13.05.24**
21	Opplæring - Python	Alle	35 dager	29.01.24	04.03.24
22	Segmentere video (test)	Øyvind	14 dager	05.02.24	19.02.24
23	Deteksjon av kjøretøy (test)	Øyvind	7 dager	12.02.24	19.02.24
24	Lysdetektor	Håkon	28 dager	12.02.24	11.03.24
25	"Motion-blur" detektor	Bjørn	63 dager	19.02.24	22.04.24
26	Nettside	Bjørn	42 dager	19.02.24	01.04.24
27	Undersøke "Deblurring"-kode	Øyvind	35 dager	04.03.24	08.04.24
28	Forbedre deteksjonsterskel	Øyvind	35 dager	11.03.24	15.04.24
29	Vanndetektor	Øyvind	14 dager	08.04.24	22.04.24
30	Utvikling av Unit-tester	Håkon	21 dager	08.04.24	29.04.24
31	Evalueringsintervju	Bjørn	9 dager	14.04.24	23.04.24
32	Rydde koden	Håkon	14 dager	22.04.24	06.05.24
33	DocString(JavaDocs)	Bjørn	14 dager	29.04.24	13.05.24



Figur 3.3: Gantt-diagram som viser hvordan vi planla utviklingen vår

Fra figur 3.3 kan vi også se noen røde rammer rundt enkelte punkter. Dette er milepæler vi satt opp for oss selv i prosjektet for å signalisere viktige faser. Den første rammen i diagrammet viser hvor vi hadde fått en bedre forståelse av hva vi jobbet med, og hvor vi hadde utviklet kode som gjorde at vi kunne segmentere våre egne bilder fra videostrømmer. Dette markerte også tiden hvor vi startet å parallellisere arbeidet vårt, samt jobbe på ulike området av prosjektet. Den siste milepælen var en sluttevaluering i form av et intervju som vi hadde med oppdragsgiver. En slik

evaluering var til stor hjelp for gruppen, og milepælene er markert fordi de var ekstra viktige faser i fremdriftsplanen.

3.4.3 Risikovurdering

En omfattende risikoanalyse var viktig for å sikre at vi var klar over risikoene med prosjektet vårt, og var godt forberedt på å håndtere dem [34]. For å utarbeide en god risikoanalyse trengte vi å:

1. Identifisere problemet
2. Finne årsaken
3. Beregne sannsynlighet for at hendelsen oppstår
4. Beregne konsekvensen dersom hendelsen oppstår
5. Velge tiltak

Ettersom HVL har lagt frem en mal på risikovurdering som besvarer alle punktene over, har vi valgt å benytte denne. I vedlegg 5 ligger modellen vi har brukt for gradering av risikoprodukt. Modellen er benyttet i utformingen av risikoanalysen i tabell 3.1.

Tabell 3.1: Risikoanalyse

	Hendelse /Risiko	Årsak	Sannsynlighet	Konsekvens	Risiko-produkt	Tiltak
1	Finner ikke tilstrekkelig litteratur.	Dårlige søk.	Lav (2)	Høy (4)	8	Sette av nok tid i starten til å finne gode kilder og få et godt fundament for prosjektet.
2	Gruppen blir mindre.	Medlem drar til CERN.	Lav (2)	Svært høy (5)	10	Redusere omfanget av oppgaven.
3	Gruppen blir mindre.	Sykdom (lengre periode).	Lav (2)	Middels (3)	6	Jobbe hjemmefra over Discord/Messenger.
4	Feiltolking av oppgave.	Dårlig kommunikasjon med oppdragsgiver og veileder.	Lav (2)	Svært høy (5)	10	Ha gode og jevnlige møter med oppdragsgiver og veileder. Ikke være redd for å spørre.
5	Dårlig prioritering.	For lite planlegging.	Høy (4)	Høy (4)	16	Ha god struktur i fremdriften vår.
6	Dårlig samarbeid i gruppen.	Uenigheter blant medlemmene.	Svært lav (1)	Høy (4)	4	Være åpne for idéer og ha gode samtaler.
7	Avgrenser ikke problemstillingen nok.	Overvurderer / undervurderer egne ferdigheter.	Middels (3)	Høy (4)	12	Ettersom vi har en veldig åpen oppgave, må vi bruke mye tid på å finne ut hva vi vil oppnå med oppgaven. God definering av problemstilling.
8	Avviker fra problemstilling.	Ikke er strukturert nok og begrenser oss der det er nødvendig.	Middels (3)	Svært høy (5)	15	Bruke dokumentasjon og hverandre på en god måte.
9	Mangelfullt utstyr.	Mangler utstyr slik som NVIDIA GPU eller lignende. Nødvendig for testing av "deblurring"-metoder som var aktuelt tidligere i prosjektet.	Lav (2)	Middels (3)	6	Låne av høgskolen, eller dele oppgavene inn slik at medlemmer med nødvendig utstyr påtar seg dette.

3.5 Evalueringsplan

Prosjektgruppen har hele veien planlagt å gjennomføre fire typer evalueringer: testing ved bruk av enhetstester [35], en egendefinert testklasse kalt “MedieManipulator”, et avsluttende dybdeintervju med prosjekteieren og brukertesting med prosjekteier hvor vi fikk evaluert fremgangen og idéene i prosjektet.

Ettersom hvert medlem utviklet forskjellige deler av programmet, har vi også jobbet av og på med ulike aspekter av prosjektet. Gruppen så det derfor som mer hensiktsmessig å teste mindre deler av programmet om gangen, i stedet for en omfattende ende-til-ende-test [35]. Enhetstesting ble derfor en type evaluering som vi benyttet for å sikre at enkelte komponenter i programmet fungerte slik som forventet. Denne typen testing var både effektiv og rask, og dermed god for testing av mindre kodebiter. Hoveddelen av testingen i koden ble gjort av en egenutviklet klasse som heter “MedieManipulator”. Denne var tiltenkt testing av alle metoder og idéer vi prøvde ut, noe som utgjorde mesteparten av koden. Denne ble brukt til å undersøke mange ulike metoder, samt vise resultatene de ga oss.

Imidlertid var det ikke bare koden som måtte evalueres, men også brukernes meninger og tilbakemeldinger. Standardiserte evalueringsmetoder slik som SUS-skjema [36], var derimot ikke relevant for oss ettersom prosjektet ikke var rettet mot større brukergrupper. Hovedsakelig ønsket vi tilbakemeldinger fra oppdragsgiver, og derfor besluttet gruppen seg for at vi skulle gjennomføre et dybdeintervju med han. I intervjuet stilte vi spørsmål knyttet til forskjellige deler av programmet når det nærmet seg prosjektslutt.

Gruppen innså at dersom vi ventet med tilbakemeldinger fra oppdragsgiver til de siste ukene av prosjektet, kunne vi fort bomme på gjennomføringen vår. Derfor hadde vi kontinuerlige møter med oppdragsgiver. Dette ble vår form for brukertesting hvor vi viste fremgang og diskuterte problemstillinger vi støtte på, samt mottok tilbakemeldinger og råd.

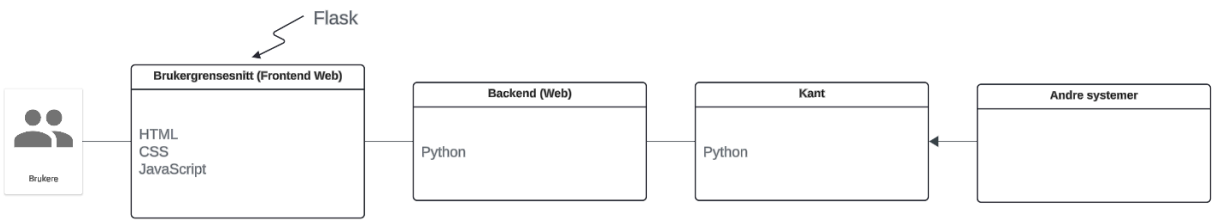
4 DETALJERT LØSNING OG PROSJEKTGJENNOMFØRING

Dette kapittelet omfatter den endelige løsningen samt de undersøkelsene vi har gjennomført for å oppnå resultatene våre. Slik vi konkluderte med tidligere består prosjektet av tre detektorer for “motion blur”, dårlig lys og vanndråper. Disse er utformet med mål om å kunne oppdage feil på en klar og pålitelig måte. For å prøve å oppnå dette har vi testet en rekke ulike metoder for å tilegne oss gradvis bedre terskelverdier. I tillegg har vi gjort undersøkelser knyttet til "deblurrings"-modeller og designet en nettsiden som har til hensikt å varsle samt presentere hva som er feil med bildekvaliteten.

4.1 Oppbygging

Systemet vårt er bygget opp som en pipeline. Figur 4.1 viser de viktigste komponentene i systemet som inkluderer brukere, brukergrensesnitt, backend (web), kant og andre systemer. I punktene nedenfor forklarer vi hvordan arkitekturen til figur 4.1 ser ut:

- Andre systemer: De systemene i Counting Hero som identifiserer det beste bildet av dekket på bilen og gir kanten vår et beskåret bilde som er klart for prosessering. Systemet er ikke integrert med Counting Hero sine systemer, så vi simulerer dette ved å beskjære bildene selv. Kapittel 4.2.1 forklarer nærmere hvorfor vi beskjærer bildene som vi bruker.
- Kant: Dette er stedet hvor vi utfører prosessering av bildene som vi mottar fra kameraene til Counting Hero.
- Backend (Web): En serviceklasse som henter objekter og bilder som skal vises på nettsiden.
- Brukergrensesnitt (Frontend Web): Nettsiden viser resultatene av deteksjonen på bildene.
- Brukere: Brukere sjekker om det oppdages feil i bildekvaliteten ved å utføre søk på nettsiden.



Figur 4.1: Systemarkitektur. Hentet fra arkitektur i systemdokumentasjon.

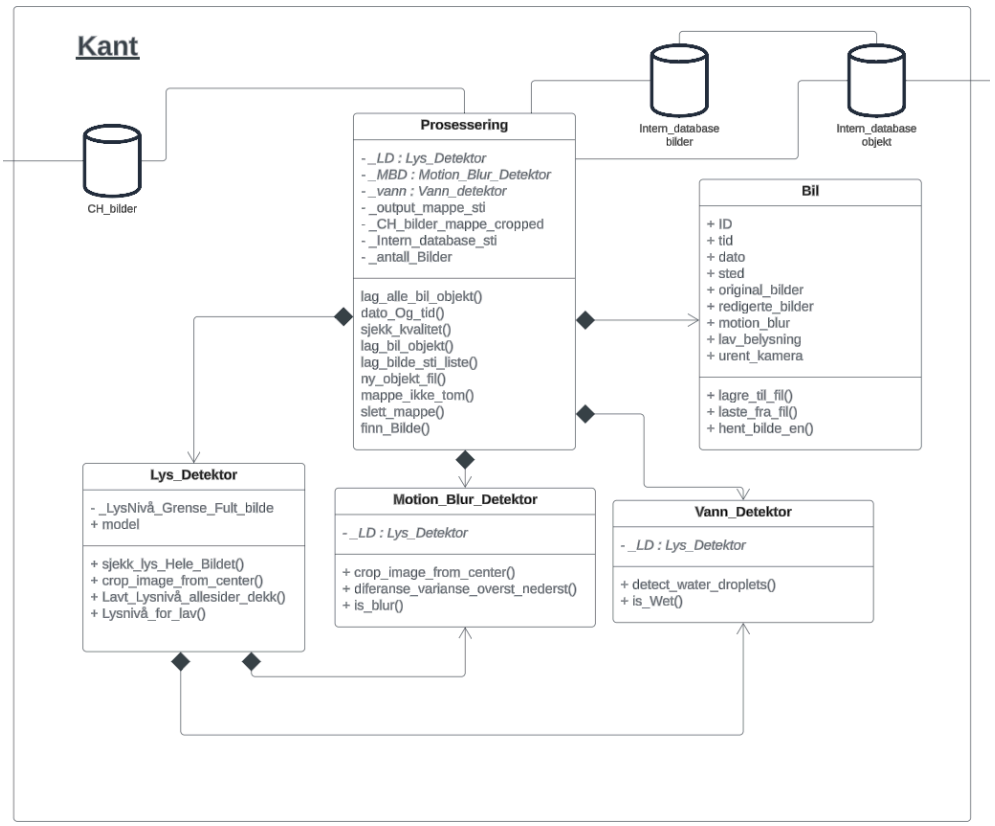
For å få en mer detaljert innsikt i systemet utarbeidet vi et klassediagram for systemet som beskriver klassene, attributtene, funksjonene og forholdet mellom dem. Dette presenteres i resten av kapittel 4.1.

4.1.1 Backend - Kant

Fra figur 4.2 ser vi at kanten vår hadde muligheten til å prosessere bilder fra en mappe som heter “CH_bilder”. Disse bildene ble gitt til oss av oppdragsgiveren en stund inn i prosjektperioden. Mappen "CH_bilder" inneholdt bildene som vi fikk fra Counting Hero sine kameraer, og er lagt inn i mapper markert med tid og dato. Et eksempel på et mappenavn kunne være "D20230324_T134257", hvor alt etter "D" indikerer dato og alt etter "T" indikerer tiden. Disse mappene inneholder ett bilde per dekk på kjøretøyet.

Ettersom vi håndterer en mengde data, er det viktig å opprettholde orden på de ulike typene bilder både før og etter behandling. Dette håndteres av klassen "Prosessering", som også fungerte som et bindepunkt for systemet vårt. "Prosessering" tar bildene den mottar og kjører detektorene på dem for å sjekke etter feil i bildekvaliteten. Klassen bruker detektorene til å sjekke bildene for lysnivå, vått dekk og "motion blur". Som vist i figur 4.2, er "Motion_Blur_Detektor" og "Vann_Detektor" avhengige av "Lys_Detektor" fordi de trenger lysnivået i bildet for å bestemme terskelverdien de skal bruke. Grunnen til at disse detektorene er avhengige av lysnivået blir forklart senere i kapittel 4.3.

"Prosessering" oppretter bilobjekter ved hjelp av klassen "Bil". Disse bilobjektene blir deretter lagret i "Intern_database_objekt". Disse objektene inneholder viktig informasjon som vises frem på nettsiden. "Prosessering" lagrer så de bildene som har blitt prosessert i en mappe ved navn "Intern_database_bilder". Bilobjektene som er lagret i "Intern_database_objekt" inneholder variabler med en referanse til hvor bildene er lagret i "Intern_database_bilder". Etter at bildene og objektene er lagret, er de klare til å vises frem på nettsiden. En nærmere beskrivelse av hvordan bilobjektene fungerer, samt hvorfor vi har to separate mapper, vil bli gitt i det påfølgende kapittelet.



Figur 4.2: Kantklasser - Klasser som ligger på kanten og er selve kjernen i prosjektet. Hentet fra klassediagram i systemdokumentasjon.

4.1.1.1 Bildeobjekter og lokal lagring

Som beskrevet i kap 2.2, benytter Counting Hero for tiden et databasesystem som vi valgte å ikke integrere i vår utvikling, ettersom dette ikke var relevant for å oppnå målet for oppgaven. Derfor valgte vi å lagre all informasjonen lokalt. Vi benyttet PKL-filer [38], også kjent som pickel-filer, for å lagre våre bilobjekter. Disse filene er basert på Python og tillater lagring av data som en strøm av bytes. Denne filtypen er designet spesielt for Python-objekter og samsvarer med Counting Hero's praksis for lagring av objekter.

Bilobjektene som blir lagret inneholder forskjellige variabler som inneholder relevant informasjon. Variablene inkluderer identifikasjonsnummer, tidspunkt, dato, sted, originale bilder, korrigerte bilder, "motion blur", lav belysning og våte dekk. Målet var å strukturere disse objektene slik at viktig informasjon kunne ekstraheres på nettsiden. Vi kunne nå sortere dataene etter behov og tildele bilobjektene boolske verdier, "True" eller "False", avhengig av om de inneholdt noen av feilene vi detekterer. I tillegg ønsket vi å gi brukerne muligheten til å se både det originale, uprosesserte bildet, samt det ferdigstilte bildet etter det var “deblurret”. Dermed kunne oppdragsgiver beholde begge versjonene for eventuelle framtidige behov. Dette påvirket

designet av nettsiden, men endte opp med å ikke bli like relevant for bilobjektene siden vi utelatte implementasjon av “motion blur”-korrigering.

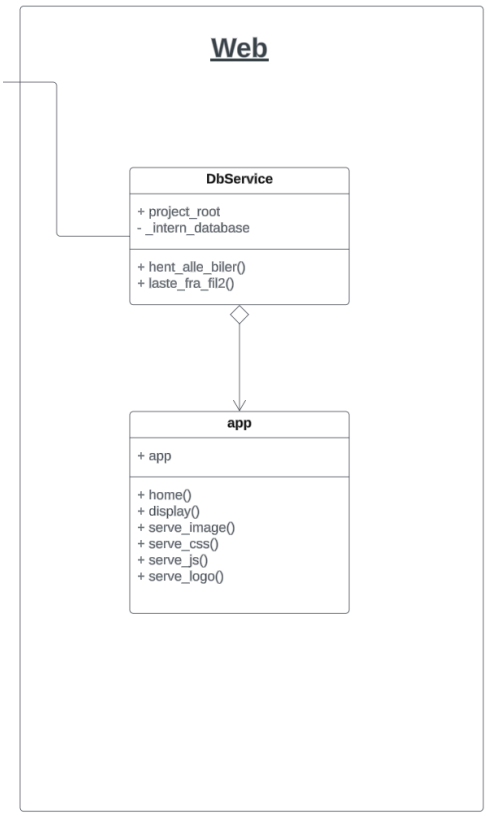
For å konstruere objektene med tilhørende informasjon, utformet vi en objektklasse. Denne ga vi navnet “Bil” i figur 4.2, og for hvert kjøretøy vil et slikt objekt bli laget, med følgende attributter:

- Identifikator: ID ble tildelt som et løpenummer for hvert kjøretøy som passerte gjennom systemet.
- Tid og dato: Disse attributtene ble hentet fra navnet på mappene bildene ligger i som forklart i forrige kapittel 4.1.1.
- Sted: Denne attributten er hardkodet til “Bergen” ettersom vi bare fikk noen håndplukkede bilder uten noen indikatorer på hvor bildene er fra. Variabelen er derimot tiltenkt å kunne vise hvor bildene kommer fra for å kunne si hvilke kamera som gir dårlige bilder.
- Kvalitet: Kvalitet er en samling av tre boolske attributter: “motion blur”, lys og vått dekk. Dersom alle tre indikerer falskt, er bildet i orden. Opprinnelig planla vi også å inkludere en detektor for urenheter i kameraet. Denne detektoren ble avgrenset som forklart i kapittel 2.2.

Vi opplevde en utfordring med at PKL-filen sin størrelse ble for stor når vi forsøkte å inkludere bildene og variablene i samme fil. Derfor bestemte vi oss for å lagre bildene og variablene i to separate mapper kalt “intern_database_bilder” og “intern_database_objekt”, slik vi viste i figur 4.2.

4.1.2 Nettside

For webutvikling har gruppen benyttet Flask [22]. Standard for applikasjoner som benytter dette rammeverket er å ha en klasse med navn “app” slik figur 4.3 viser. Denne klassen inneholder logikk som fungerer slik som en kontroller gjør i SpringBoot [39] ved å ha nøkkelord som brukes til å dirigere brukeren til forskjellige sider. "DbService" fungerer som backend for nettsiden og utfører funksjoner for å hente bilobjekter fra der de er lagret, slik at de kan vises på nettsiden.



Figur 4.3: Webklasser - Klassene i web-delen av prosjektet. Hentet fra klassediagram i systemdokumentasjon.

4.1.3 Analyse i “MediaManipulator”

Figur 4.4 inkluderer en klasse kalt "Media Manipulator". Denne klassen blir blant annet brukt til å generere histogrammer av dekkene fra bildene vi mottok fra Counting Hero. Et histogram er en grafisk illustrasjon av numeriske data fordelt på rektangulære søyler. Høyden på disse er antallet eller frekvensen av data som tilhører kategorien søylen representerer [37]. "MedieManipulator" spilte en viktig rolle i testingen av deteksjonen for å finne den passende terskelverdien. Histogrammene som genereres av denne klassen vil være relevante i kapittel 4.2 og 4.3 når vi analyserer våre resultater og evaluerer om vi har identifisert tilstrekkelige terskelverdier.

Figur 4.4: Klassen “MedieManipulator”. Hentet fra klassediagram i systemdokumentasjon.

"MedieManipulator"-en i figur 4.4 inneholder en stor del av alle løsningene og metodene vi har undersøkt. Noen av funksjonene i denne klassen har ikke gitt tilfredsstillende resultater og er derfor forkastet. Metodene er derfor ikke brukt i fastsettelsen av terskelverdiene, men er likevel metoder som vi har testet.

4.2 Lysdetektor

Lysdetektoren var meget relevant for Counting Hero, da utilstrekkelig lysnivå kunne påvirke deres KI-modell sin evne til å analysere informasjon fra bilder av bildekk. For å adressere dette problemet har Counting Hero strobelys på sine kamera, samt evnen til å justere lukkerhastigheten på kameraene for å forbedre resultatene. Strobelys er lys som sender ut en lyspuls hver gang kameraet tar et bilde, noe som kan minne om “blitz” i et vanlig kamera. Lukkerhastigheten er hvor lenge lukkerne på kameraet er åpen og sensorene er eksponert for lys [40]. Har man en kort lukkerhastighet blir kameraet eksponert for mindre lys og vi vil få mørkere bilder. En vil derimot

kunne fange raskere bevegelser med mindre risiko for “motion blur“. Dersom vi har lengre lukkerhastighet vil kamera få mer lys, men vi kan ikke ta bilder av like hurtige objekt uten “motion blur” til stede. At strobelysene til oppdragsgiver svikter, er veldig sjeldent, men det er likevel et problem å ta stilling til. Derfor var dette et aspekt de ønsket å kunne oppdage så fort som mulig.

For å kalkulere lysnivået til bildene må vi først hente ut bgr-verdiene deres (blue, green, red). Dette har vi gjort ved å benytte oss av funksjonen “imread” fra biblioteket OpenCV [21]. Denne funksjonen tar imot et bilde, leser av bgr-verdiene i bildet og lagrer verdiene i en “NumPy-array” [41]. Dette er effektive datastrukturer som organiserer bildeinformasjon i flerdimensjonale tabeller, og tillater komplekse matematiske operasjoner. I vært tilfellet er tabellen tre-dimensjonal, og kan se ut som en matrise. Disse tabellene muliggjorde en rekke operasjoner for oss, hvorav den viktigste er bruken av funksjonen “mean” fra OpenCV biblioteket, vist i figur 4.5. Funksjonen beregner gjennomsnittet av alle verdiene som er lagret av “imread”-funksjonen. Disse bgr-verdiene lagres i området fra (0, 0, 0) som representerer svart, til (255, 255, 255) som representerer hvitt. Med disse verdiene kan vi kalkulere den gjennomsnittlige bgr-verdien til hvert bilde slik vi forklarer under. Den gjennomsnittlige verdien er hva vi refererer til som lysnivået til hvert enkelt bilde. Desto høyere verdien vi får fra “mean”-funksjonen, desto høyere er lysnivået. For å bestemme en god terskelverdi for lys må vi finne det laveste lysnivået hvor bilder med og uten strobelys skilles.

I figur 4.4 finner vi formlene for *N* og «Mc». «Mc» representerer "mean" -funksjonen som beregner gjennomsnittet av bgr-verdiene. *N* er en metode som brukes for å normalisere funksjonen "Mc" ved å dele bgr-verdiene i bildet på antallet posisjoner i matrisen. For eksempel, i en 3x3-matrise, vil *N* være lik 9. I denne sammenhengen betyr "mtx(I)c" verdien ved indeks *I* i kolonne c i matrisen mtx. For eksempel, gitt en matrise (1, 2, 3), vil "mean"-funksjonen summere disse tre tallene og deretter dele summen på N = 3, siden det er tre indekser, og returnere 2 som gjennomsnittsverdien. I dette eksemplet vil verdien 2 være lysnivået vårt. Betingelsen "mask(I)≠0" angir at ingen indekser i matrisen kan være null.

$$N = \sum_{I:\ \mathtt{mask}(I) \neq 0} 1$$

$$M_c = \left(\sum_{I:\ \mathtt{mask}(I) \neq 0} \mathtt{mtx}(I)_c \right) / N$$

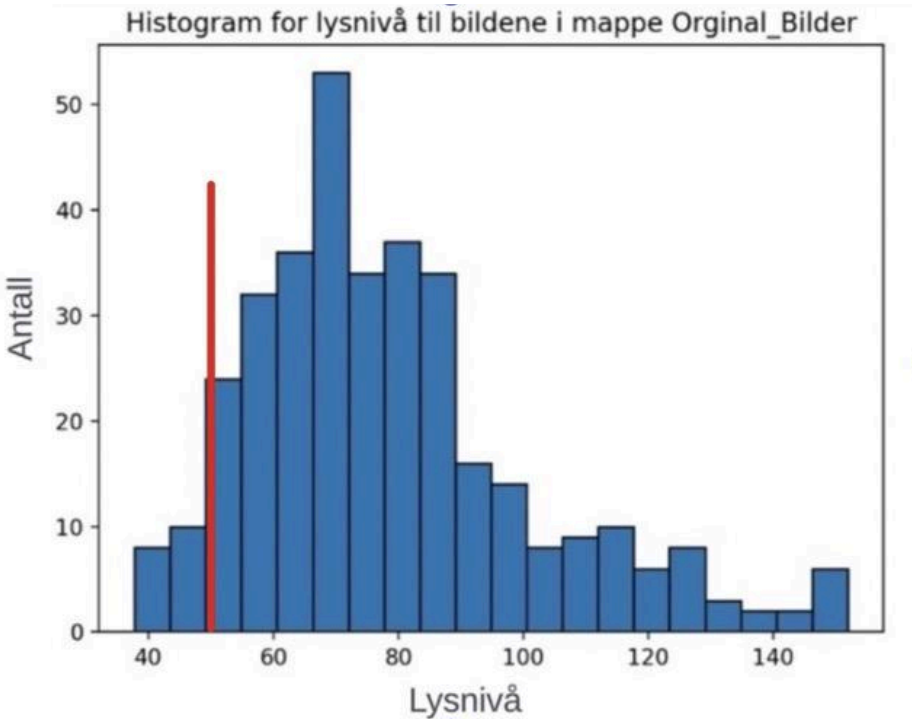
Figur 4.5: “Mean”-funksjonen (Mc) fra OpenCV biblioteket [42]

4.2.1 Terskelverdi for lysnivået

For å finne terskelverdien, begynte vi med å teste ulike verdier uten bilder fra Counting Hero. Som forklart i kapittel 2.2, søkte vi på nettet og fant en video [43] med passende lysforhold som et

utgangspunkt for å teste detektorene. Deretter produserte vi vår egen video med forskjellige lave lysnivåer. I tillegg tok vi bilder med "motion blur" under ulike lysforhold. Videoen som vi produserte ligger i koden for prosjektet og kan kjøres. Hvordan man kjører denne videoen er forklart i kapittel 8 av systemdokumentasjonen. Hensikten med disse tiltakene var opprinnelig rettet mot å danne en midlertidig forståelse av hva vi kunne forvente av koden. Ettersom bildene vi brukte viste hele kjøretøy, hadde disse resultatene en begrenset relevans for den endelige terskelverdien som skulle være basert på bilder av dekk. Vi utførte også tester med altfor lave lysnivåer, som viste seg å ikke gi meningsfulle resultater. Imidlertid ga denne prosessen et verdifullt innblikk i hva vi kunne forvente senere, og etablerte en midlertidig terskelverdi for koden inntil vi fikk tilgang til bilder fra Counting Hero.

Når vi mottok bildene fra Counting Hero, bestemte vi oss for å generere et histogram over lysnivåene til alle bildene.

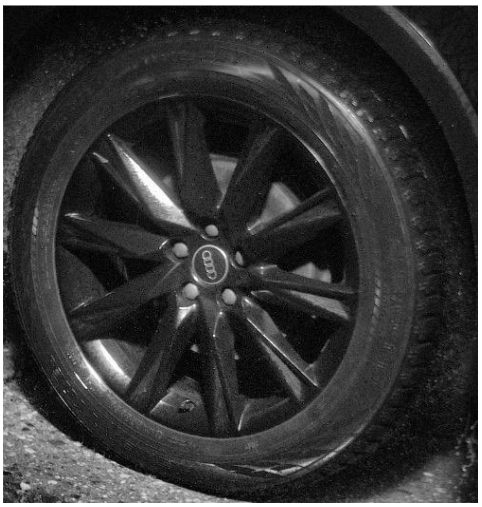


Figur 4.6: Histogram som viser lysnivået til alle bildene vi mottok fra Counting Hero. Bildene vi benyttet her viser hele kjøretøy.

På histogrammet i figur 4.6 fremgår fordelingen av antall bilder basert på deres lysnivå. Vårt mål var å identifisere en klar forskjell mellom bildene tatt med strobelys og de uten. Dessverre var ikke den observerte forskjellen så markant som forventet. Imidlertid var det en tydelig reduksjon i antallet bilder med et lysnivå under 50, illustrert med den røde linjen i figur 4.6. Som et resultat ble det utviklet en dedikert kode for å skille ut bilder med et lysnivå under 50. To eksempler, både med og uten strobelys, er gitt i de følgende figurene.

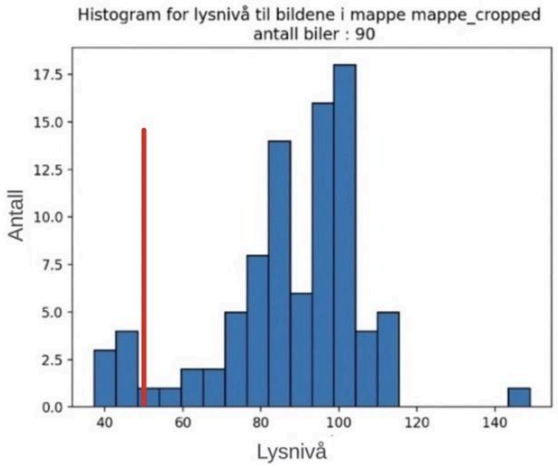

Figur 4.7 og 4.8: Et bilde med strobelyset og et bilde uten strobelys.

Når vi inspiserte bildene vist i figur 4.7 og 4.8 var det tydelig at begge var tatt om kvelden. Imidlertid hadde bare det ene bildet direkte belysning fra strobelyset på hjulet, mens det andre ikke hadde dette. Siden vår metode for beregning av lysnivået ble utført på hele bildet, ville den mørke og ubetydelige delen av bilen utgjort en for stor del av beregningen, og dermed senket gjennomsnittet. Vi diskuterte derfor å begrense bildene til å bare inkludere dekket, for å se om dette kunne hjelpe med problemstillingen vi hadde. Figur 4.9 viser hvordan et slikt beskåret bilde kunne se ut.



Figur 4.9: Hvordan et oppkuttet bilde med bare bildekket kan se ut.

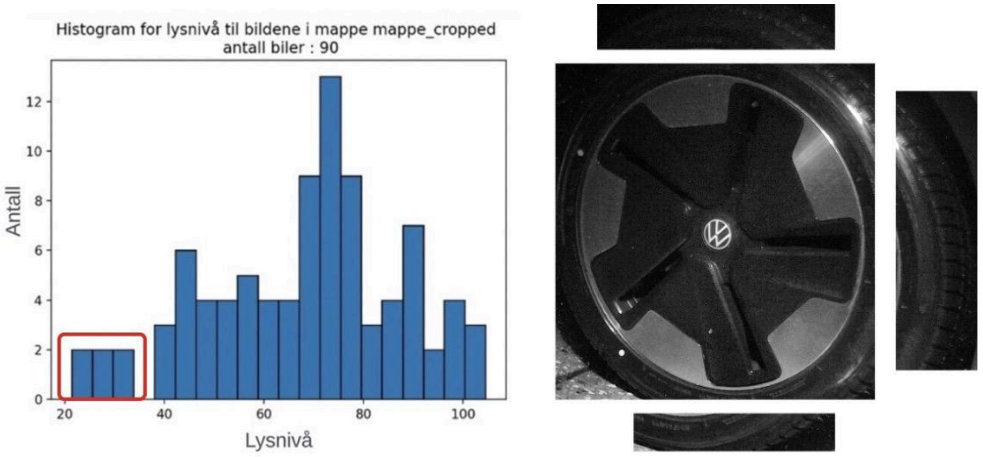
Counting Hero sitt nåværende system inkluderer en KI-modell for gjenkjenning av dekk. Denne modellen kan i teorien kunne brukes til å automatisk beskjære bildene. Integrering av denne modellen i vårt system ville imidlertid vært for tidkrevende. Etter diskusjoner med oppdragsgiver, besluttet vi å manuelt beskjære noen bilder til testformål. Med de nye bildene genererte vi et nytt histogram, som ga følgende resultater:



Figur 4.10: Histogram over fordelingen av lysnivå etter at bildene er beskåret til bare bildekket.

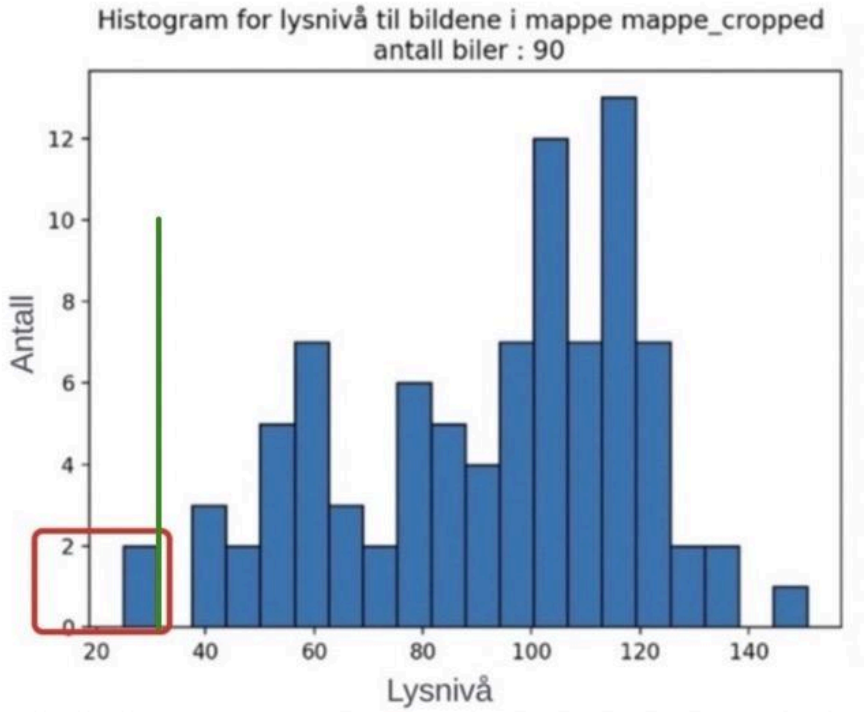
Resultatene vi ser i figur 4.10 viser en positiv fremgang. Dette er fordi vi kan se et markant skille mellom resultatene fra lysnivå 50 og ned. Fra over 300 bilder beskjærte vi 90 bilder som vi kunne bruke i figuren over. Vi hadde ønsket å kunne beskjære og bruke flere bilder, men dette var for tidkrevende. Til tross for forbedringen vi så i figur 4.10 håpet vi fortsatt på en mer distinkt forskjell mellom bildene som var akseptable og de som ikke var det. For å løse dette bestemte vi oss for å filtrere ut bildene med et lysnivå under 50.

Det som ble observert gjentatte ganger på noen av disse bildene, var at felgene var svarte. Dette resulterte i at noen få bilder ble merket med lav belysning selv når dette ikke var tilfellet. Imidlertid, selv om de fleste bildekk er sorte og refleksjonene vanligvis var tydelig synlige hvis blitsen var aktivert, valgte vi å segmentere bildet i tre deler: øvre, nedre og høyre del av dekket. Deretter beregnet vi gjennomsnittet av lysnivået for disse delene og genererte et nytt histogram basert på denne verdien:



Figur 4.11 og 4.12: Histogrammet viser et klart skille mellom lav og ikke-lav belysning når vi analyserer forskjellige deler av dekket slik figur 4.12 viser.

I dette forsøket oppnådde vi resultatet vi ønsket. Gruppen kunne nå se at det var en klar separasjon mellom bildene med utilstrekkelig lys og de med tilstrekkelig lys, representert med den røde boksen i figur 4.11. Til tross for dette hadde vi fortsatt noen uventede resultater. I figur 4.12 ser vi at dekket er veldig mørkt, men høyre side av dekket ligger i praksis på grensen til godt lys. Dette resulterer i at det kunne ha blitt godkjent av systemet til Counting Hero. Imidlertid, på grunn av skyggelegging på noen deler av dekket, ble gjennomsnittlig lysnivå lavere enn forventet. Vi bestemte oss derfor for å beregne hvilken av de tre delene av dekket vi segmenterte i figur 4.12, som hadde høyest lysnivå. Delen med det høyeste lysnivået ble valgt og tatt med i histogrammet i figur 4.13



Figur 4.13: Fordeling av lysnivå når vi bare bruker den delen av dekket som har høyest lysnivå

Den nye tilnærmingen illustrert i figur 4.13 ga et resultat hvor vi så at de to mørkeste bildene, i den røde boksen, ble separert fra resten av datasettet. Derimot var det noen mørke bilder med høy gjenskinn på enkelte deler av dekket som fortsatt kunne bli markert som gode. Gruppen så altså at det å beregne hvilken del som hadde høyest lysnivå, og basere histogrammet på dette, ikke ga noe bedre resultater enn figur 4.11. Vi besluttet derfor å bruke den tidligere metoden fra figur 4.11 som tok snittet av de tre forskjellige delene av dekket.

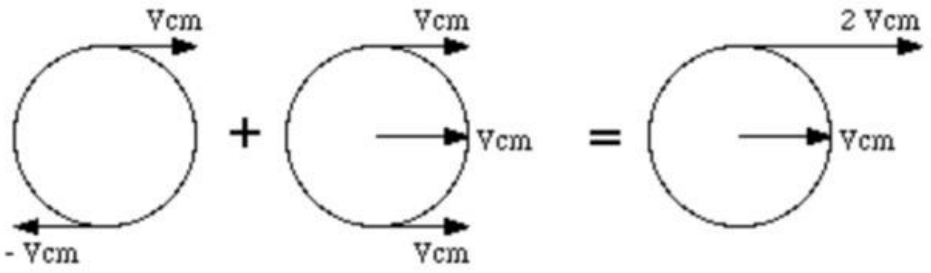
4.3 “Motion blur” detektor og Vanndråpedetektor

4.3.1 “Motion blur” detektor

Med terskelverdien for lys på plass, gjenstod det å utvikle en metode for å detektere "motion blur". Dette er en gjentagende utfordring Counting Hero har møtt på, som hindrer dem i å nøyaktig observere kvaliteten og hente informasjon fra bildekket. Ved å undersøke dekkene er det flere viktige aspekter som bør bemerkes.

Figur 4.14: På dette bildet kan vi se at gummien på øvre del av dekket har mye uskarpheter (“motion blur”), mens gummien på nedre del av dekket er ganske klar og tydelig.

Når vi inspiserer den nedre delen av bildekket i figur 4.14, er det tydelig at “motion blur” reduseres, mens det er mest markant på øvre del. Dette kommer av at toppen på dekket opplever både kjøretøyet sin translasjon (bevegelse), og dekkets rotasjon. Disse kombinert resulterer i dobbel hastighet på toppen av det rullende dekket, i forhold til midten av dekket [44], slik figur 4.15 illustrerer. På nedre del av dekket beveger rotasjonshastigheten seg imidlertid i motsatt retning av translasjonsbevegelsen og forblir relativt stillestående. Vi får dermed mer “motion blur” på toppen enn ellers på hjulet.



Figur 4.15: Rotasjon + Translasjon = Dobbel hastighet på toppen av dekket [45].

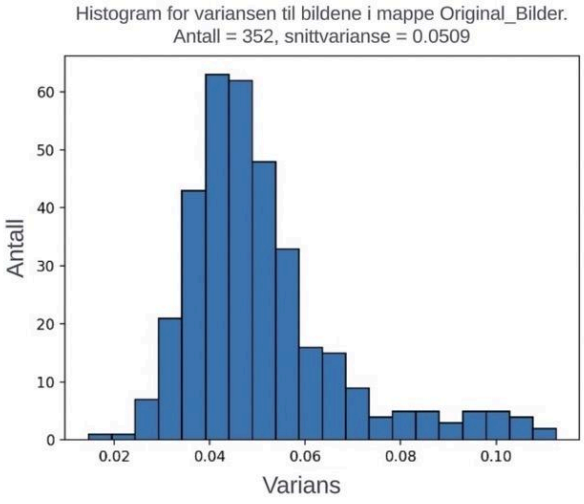
Vår antakelse er at pikslene i områder på dekket med høy grad av "motion blur" vil bli mer lik hverandre. Som et resultat av dette vil variasjonen mellom pikslene være lavere. Derfor ønsker vi å begynne med å teste denne hypotesen.

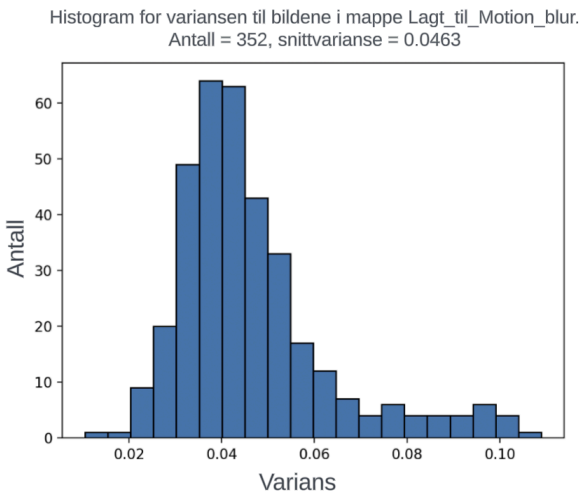
På samme måte som i lysdetektoren, brukte vi funksjonen “imread” fra OpenCV for å konvertere bildet til en “NumPy array” [41]. Deretter ønsket vi å beregne variasjonen mellom pikslene i bildet. Til dette formålet benyttet vi “var”-funksjonen fra PyTorch biblioteket i figur 4.16. I formelen under står x for utvalget av elementer, $\bar{X}$ er utvalgsgjennomsnittet, N er antall prøver, og δN er korreksjonen

$$\sigma^2 = \frac{1}{\max(0,\ N - \delta N)} \sum_{i=0}^{N-1} (x_i - \bar{x})^2$$

Figur 4.16: Variansformel [46]

Ettersom denne koden genererer verdier som overstiger tusen, valgte vi først å normalisere verdiene til bildet mellom 0 og 1 før vi beregnet variasjonen. Vi forventer at variasjonsverdier nærmere 0 inneholder bilder som har mer “motion blur”. Når vi hadde utført beregningen av variasjonen for et bilde, konstruerte vi deretter to histogram over verdiene for alle bildene i to mapper.

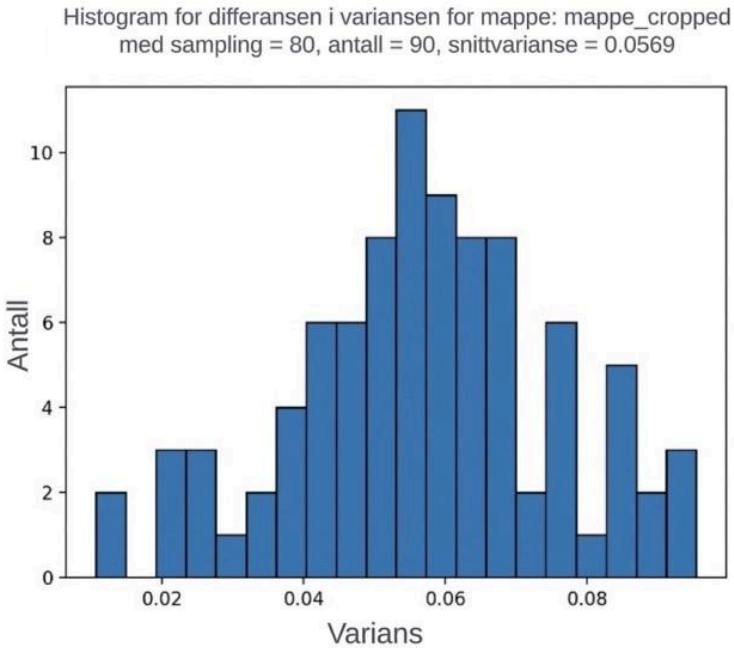




Figur 4.17 og 4.18: Normaliserte histogram for to ulike mapper med bilder - “Original_Bilder” og “Lagt_til_Motion_blur”.

Figur 4.17 og 4.18 presenterer to histogrammer: ett inneholder de originale bildene mottatt fra Counting Hero, mens det andre inneholder de samme bildene, men med nok tillagt "motion blur" at det er synlig. Det bemerkelsesverdige er nedgangen i gjennomsnittlig variasjon når "motion blur" er lagt til, noe som bekrefter hypotesen vi antok innledningsvis. Vi ønsket opprinnelig med denne testen å sette en terskelverdi som separerer bildene med "motion blur" og bildene uten. Vi ser derimot at når det er høy overlapping mellom histogrammene for bilder med og uten "motion blur", og vi klarer ikke å sette en terskelverdi som separerer dem med høy nøyaktighet. Dette ser vi ettersom de to histogrammene er ganske like, noe som er en ugunstig observasjon. I teorien bør alle bildene i mappen "Lagt_Til_Motion_blur" ha en lavere varisjonsverdi i forhold til bildene i mappen “Original_Bilder”. Vi kan derimot se ut fra figur 4.17 og 4.18 at dette ikke er tilfellet.

På samme måte som helt innledningsvis i lysdetektoren, hadde også bildene vi brukte her store deler av kjøretøyet i seg, når vi egentlig bare er interessert i dekket. Dette medførte at mange irrelevante piksler ble tatt med i beregningen, og påvirket variansen mellom pikslene i bildet. De irrelevante pikslene brakte en rekke variabler slik som unødvendig støy, farge og vann. Med begrepet støy mener vi tilfeldig variasjon i pikselverdiene. På grunn av disse variablene valgte vi å adoptere tilnærmingen vi utviklet for lysdetektoren vår og fokusere utelukkende på bilder av bildekkene. Figur 4.19 viser hvordan histogrammet så ut når vi implementerte disse endringene til bildet.



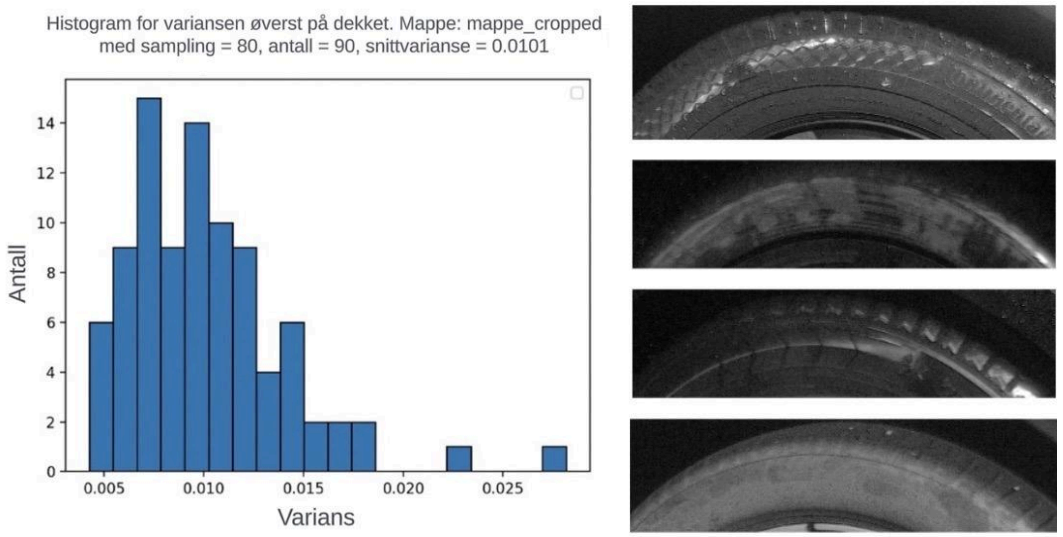
Figur 4.19: Et histogram over differansen i varians for bilder som er beskåret til å bare inneholde bildekket.

Ettersom ulike kjøretøy har forskjellig størrelsene på dekkene sine, vil pikseltettheten variere fra kjøretøy til kjøretøy når de beskjæres. Med variasjon i pikseltettheten mener vi i dette tilfellet hvor mange piksler et kjøretøy sitt dekk tar opp i motsetning til et annet mindre eller større kjøretøy. På

grunn av disse forskjellene ble bildene samplet slik at de fikk en konsistent pikseltetthet på 80, noe som ligger litt under den laveste verdien vi observerte i våre bilder. Disse resultatene var lovende, men vi var fortsatt ikke fornøyde.

Fra histogrammet i figur 4.19 så vi at antall bilder droppet av for variansverdier under 0.04. Derfor utførte vi en test hvor bildene med en varians under denne verdien ble separert. I denne testen observert vi at en del av disse bildene var de samme som også ble identifisert som lavt lysnivå i lysdetektoren. Vi observerte noen sammenhenger som for eksempel at de to bildene som skiller seg ut fra figur 4.19, er de to samme bildene som skiller seg ut fra figur 4.13. Vår kode ble strukturert slik at bildene med utilstrekkelig belysning ikke ble testet for "motion blur", noe som gjør de fleste av disse bildene irrelevante. Ved en gjennomgang av de resterende bildene i figuren, så vi ingen tydelig sammenheng mellom variasjonen og nivået av "motion blur" i bildene. Vi ble derfor nødt til å prøve å finne en ny tilnærming.

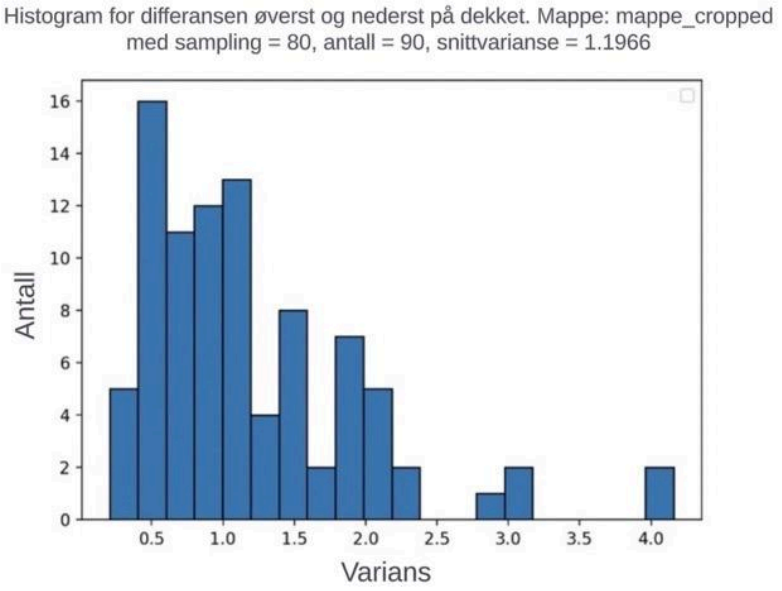
Så langt hadde vi beregnet variasjonen for hele bildet. Vi så derimot at forskjellige typer felger påvirket variasjonen i bilder forskjellig. Ettersom vi forventet mest “motion blur” i øvre del av dekket, valgte vi å utføre en test av variasjonen der.



Figur 4.20 og 4.21: Til venstre er histogrammet som viser variansen til bildene som er beskåret til å bare vise toppen av dekkene hvor uskarpheten er størst. Til høgre er eksempel på hvordan disse bildene ser ut.

Resultatene vi fikk fra den nye tilnærmingen vi brukte i figur 4.20 var ikke helt det vi håpet på. Selv etter at vi har eliminert felgene som betydelig påvirket variansen, ser vi fortsatt at det er veldig mange faktorer som spiller inn. Dette inkluderte fuktighetsgrad, tykkelse, mønstre på dekkets side, gjenskinn med mer. Ettersom det var veldig mye variasjon mellom de ulike dekkene, besluttet vi i stedet å sammenligne bildet med seg selv. Vi gjorde dette ettersom dekket forventes å

være relativt konstant med seg selv. Siden det ikke er noe “blur” på bunnen av dekket men veldig mye på toppen, kunne vi beregne forskjellen i variasjonen mellom toppen og bunnen av dekket.

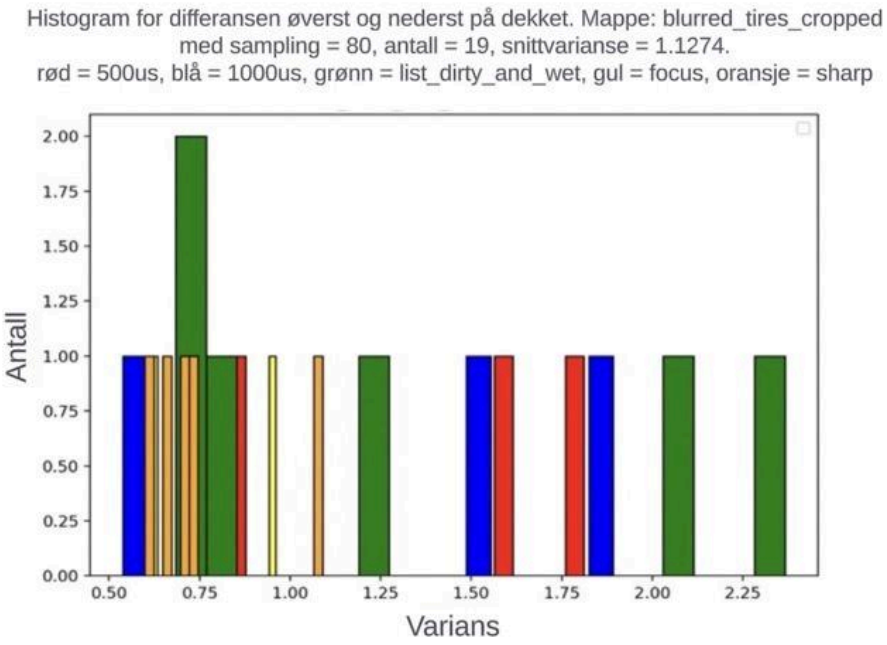


Figur 4.22: Dette histogrammet viser et klart skille mellom differansen på bilder av toppen av dekket og bunnen av dekket.

Fra resultatene i figur 4.22 håpet vi at bildene med de høyeste variasjonsverdiene ville være bildene med mest “motion blur”. Noen av disse bildene passet til våre forventninger, men vi observerte at enkelte gode bilder med betydelig forskjell i lysnivå mellom toppen og bunnen av dekket fikk en høyere verdi enn forventet. I tillegg ble det observert at noen av bildene med mest “motion blur” hadde relativt lav verdi. Disse bildene var hovedsakelig mørke og hadde våte dekk. Dette var noen ganske uforventede og forvirrende resultater, men på dette stadiet fikk vi nå flere bilder i organiserte mapper fra Counting Hero, slik at vi kunne teste ytterligere.

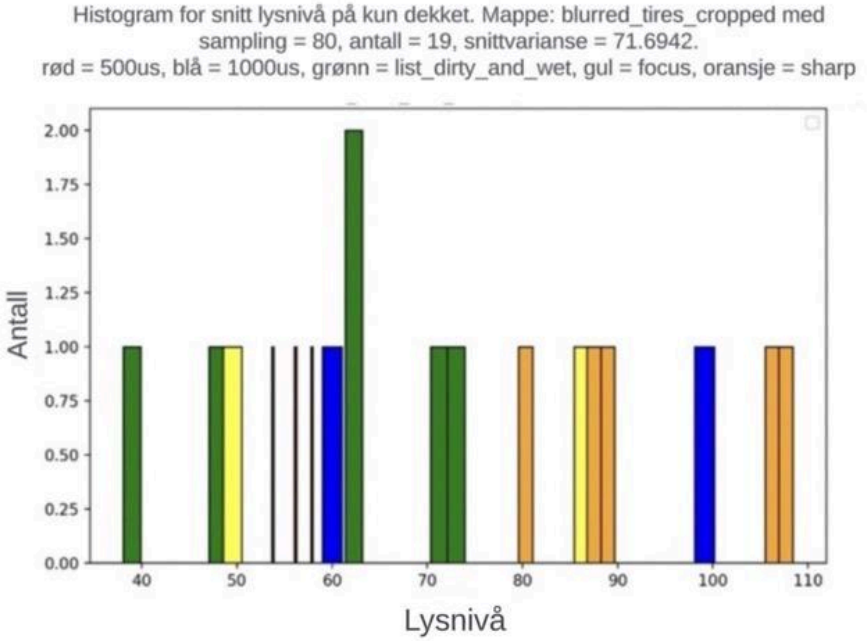
Mappene vi fikk ble klassifisert med ulike attributter: "dirty and wet", "focus", "sharp", og "motion blur", hvorav sistnevnte ble videre delt inn i to underkategorier basert på lukkertiden til kameraet, derav "500us" og "1000us". Disse navnene viser til lukkerhastigheten på kameraet [47]. Bilder tatt med “1000us” har en lukkerhastighet på 1/1000 dels sekund, og dermed vil ha mindre lys, men vil være mindre påvirket av bevegelse og dermed bedre egnet for bilder på dagtid. Derimot har bilder tatt med “500us”, en lukkerhastighet på 1/500 dels sekund, noe som vil si at det vil ha mer lys, men vil være mer påvirket av bevegelse, og dermed være bedre egnet for kveldsbilder.

Med de nye mappene besluttet vi å lage et histogram hvor vi separerte dem basert på mappenavnene. Innholdet i mappene varierte fra to til seks bilder, noe som vil si at utvalget fra hver enkelt mappe var begrenset. Mappen "sharp" inneholder gode bilder, mens de andre inneholder bilder med forskjellige bildekvalitetsfeil. Dette histogrammet ligger under som figur 4.23.



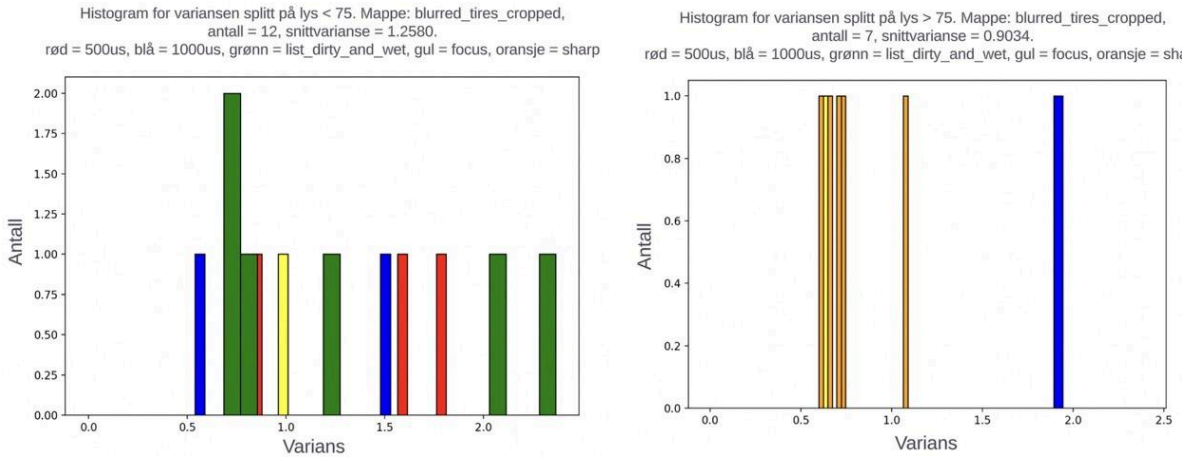
Figur 4.23: Histogram som viser varianseforskjeller mellom bilder tilhørende ulike mapper.

Resultatet vi fikk fra alle de ulike mappene fremstår som noe uordnet, men det som er merkbart er at bildene som anses som gode (representert med fargekoden oransje) holder seg relativt tett sammen, mens de dårlige bildene er mer spredt. En veldig god observasjon er at en god gruppe av dårlige bildene var sentrert rundt 0.9. Vi spekulerte om dette kunne ha en sammenheng med lysnivå, og utførte derfor en sammenligning med et histogram over det gjennomsnittlig lysnivået for de ulike delene av dekket.



Figur 4.24: Histogram som viser forskjellene i lysnivået for de ulike delene av dekkene.

Histogrammet i figur 4.24 viser at alle de grønne søylene som representerer bilder av våte dekk har et lysnivå under 75. Samtidig er de aller fleste bildene som anses som gode over dette lysnivået, derav de skarpe og fokuserte bildene, samt bildene med rask lukkerhastighet. Basert på denne observasjonen, bestemte vi oss for å konstruere et nytt histogram der vi beregner variasjonen, men splitter bildene i to grupper: de med lysnivå over 75 til høyre i figur 4.26 og de med lysnivå under 75 til venstre i figur 4.25.



Figur 4.25 og 4.26: Kalkulerer variansen for bilde av våte dekk. Vi splittet dekkene opp i de over og under et lysnivå på 75. Til venstre er histogrammet for bildene under 75, mens til høyre er de over.

Begrunnelsen for hvorfor vi utførte denne testen var for å utforske om vi så noen sammenheng mellom separasjonen basert på lysnivået, og basert på variansen. I figur 4.26 vises variansen for alle bildene med lysnivå på over 75. De oransje søylene er gode bilder mens den blå har “motion blur”. Disse resultatene viser at for et lysnivåover 75 kommer det stort sett bare gode bilder med, og det er i tillegg god separasjon ned til det dårlige bildet. Dette er veldig lovende da det gjør det lett å estimere en pålitelig terskelverdi for “motion blur” for bilder med lys over 75.

I figur 4.25 viser bildene med lysnivå under 75. Alle disse bildene er blitt klassifisert som dårlige av Counting Hero. Vi ser her at alle bildene er veldig spredt, noe som indikerer at vi vil måtte definere flere forskjellige terskelverdier basert på lysnivå på andre variabler. Det viste seg altså at å få en pålitelig måling av “motion blur” var utfordrende ettersom det var mange variabler som påvirket bildene. En vellykket løsning krever en rekke ulike tilnærminger for å analysere kvaliteten av bildet.

Figur 4.27 og 4.28: Bilder som viser forskjellen mellom klart bilde kontra dårlig og tåkete bilde

Vi observerte at metoden, som sammenlignet de øvre og nedre delene av bildet, fungerte godt på klare bilder, slik som vi kan se i figur 4.27. Metoden presterte derimot mindre tilfredsstillende på uskarpe bilder som vi kan se i figur 4.28. I kontrast fungerte den andre metoden, som baserte seg på gjennomsnittlig variasjon av de tre sidene av dekket, bedre på uskarpe bilder.

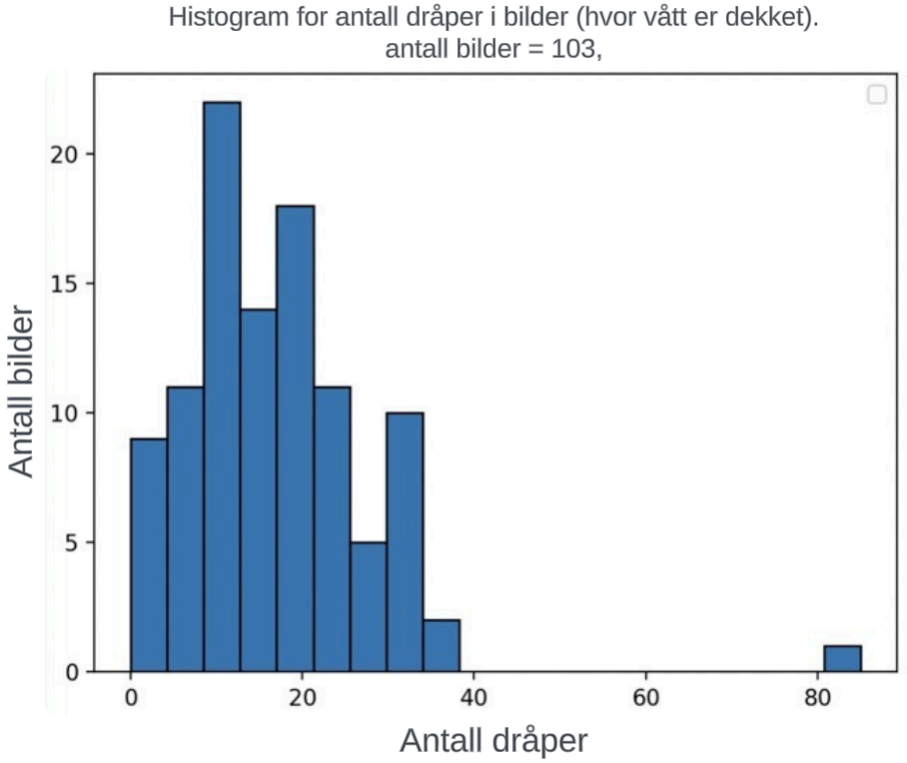
Lysnivået viste seg å ha betydelig innvirkning på variansen i bildene. Vi satt derfor en rekke ulike terskelverdier basert på to distinkte lysforhold. Dette tillot oss å identifisere et større antall bildeeksempler med "motion blur" samtidig som vi reduserte antall gode bilder som ble markert som "motion blur".

4.3.2 Vanndråpedetektor

En utfordring vi møtte på når vi prøvde å separere gode og dårlige bilder var bilder som inneholdt store mengder vann slik vi ser i figur 4.29 og 4.30.

Figur 4.29 og 4.30: To bilder som viser to god eksempeler på mye vann

Vanndråpene ga stort utslag på variasjonsmønstrene i bildene, noe som gjorde dem ekstra utfordrende å detektere. Disse vanndråpene identifiseres som lyse prikker på bildene og hadde ofte en forstyrrende effekt på fokuset og lysforholdene i bildet, noe som spesielt påvirket ytelsen til lysdetektoren. For å imøtekomme denne utfordringen besluttet vi å implementere en tilnærming basert på telling av vanndråpene i bildet, samt lage et histogram for resultatene



Figur 4.31: Histogram for vanndråpe-detektoren

Vi utarbeidet et histogram, figur 4.31, der antall detekterte vanndråper var representert horisontalt, mens mengden bilder som inneholdt ulike antall dråper var representert vertikalt. Ved manuell kontroll av bildene fant vi ni bilder som var problematiske på grunn av vanndråper, men histogrammet over viste at nær alle ble detektert med noen vanndråper. Siden vår metode søkte etter hvite prikker, kunne støy oppfattes som vanndråper. Dette utøvde noe usikkerhet i detektoren sin nøyaktighet, men vi så at bildene av de mest våte dekkene ble tydelig skilt fra resten. Basert på histogrammet bestemte vi oss derfor for å sette en terskelverdi på 30 dråper, ettersom det var i området hvor gode og dårlige bilder blandet seg. Gruppen anså bilder med et høyere antall dråper enn 30 som vannbelastet.

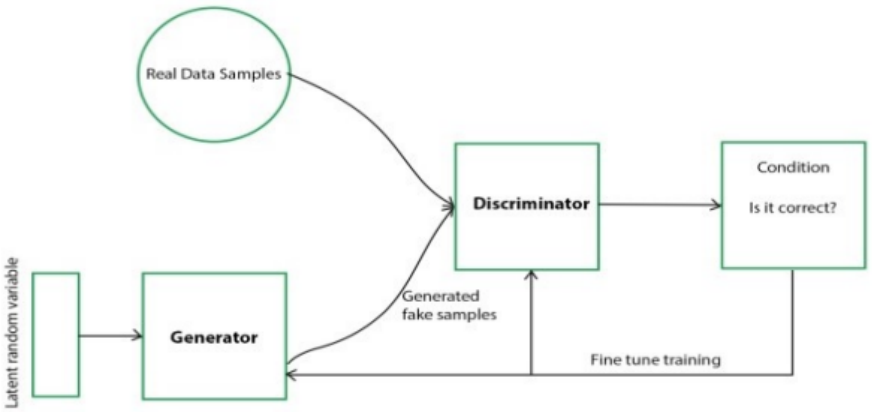
I den første testen klarte vi å registrere nesten alle bildene som vi tidligere hadde utfordringer med knyttet til vanndråper. Siden en del av de lyse bildene hadde en del støy i skyggene sine, implementerte vi to forskjellige terskelverdier for vanndråpedetektoren basert på lysforholdene i den endelige løsningen. Resultatet av hvor bra detektorene vi har laget er, vil bli evaluert i kapittel 5.4.

4.4 Deblurring

Et av målene vi satt for prosjektet var å gjøre en del undersøkelser rundt eventuelle modeller for "deblurring" som Counting Hero kunne vurdere ved fremtidig videreutvikling. Dette kapittelet omfatter derfor de modellene vi har sett på, samt litt fordeler og ulemper med dem. Målet er noe vi anser som viktig for temaene vi har jobbet med, og det har spilt en rolle i utviklingen av prosjektet, slik som på nettsiden sitt utseende.

4.4.1 Generative Adversarial Network

Generativt motstandernettverk (GAN) [14, 48] er en modelleringsmetode som har vist godt potensial innen generativ modellering i nyere år. Slik figur 4.32 viser, består GAN-rammeverket av to nettverk: en generator og en diskriminator. Det generative nettverket produserer et skarpt bilde som er ment å etterligne det ekte bildet så mye som mulig. Bildet er deretter sammenlignet med det ekte, skarpe bildet, i det diskriminerende nettverket som skal prøve å skille mellom ekte og falske prøver. De to nettverkene trenes samtidig på en konkurransepreget måte, der det generative nettverket forsøker å generere falsk data som er umulig å skille fra ekte data, mens det diskriminerende nettverket forsøker å korrekt klassifisere den genererte dataen som falsk. Når det ene eller det andre nettverket kommer ut seirende må motstanderen finjusteres og lære av sine feil. Denne prosessen med motstander (adversarial) trening fører til kontinuerlig forbedring av begge nettverkene, noe som resulterer i generering av stadig mer realistiske data.

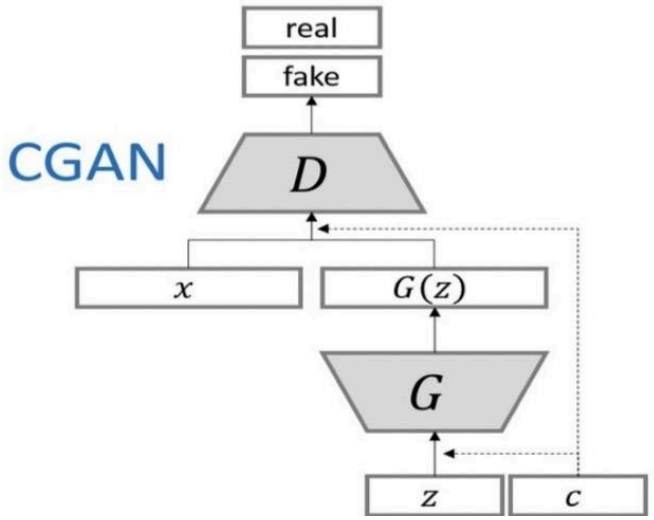


Figur 4.32: Generative adversarial networks model [48]

Dette rammeverket er en type "blind deblurring" hvor "blur-kernel"-en ikke er kjent. I stedet har vi en generator som ikke har tilgang til det ekte bildet, men som jobber ut fra en tilfeldig støyvektor. Dette er en vektor med verdier som typisk sett kommer fra en enkel distribusjon, slik som for eksempel en Gaussian distribusjon [49]. Verdiene vil sannsynligvis være ganske tilfeldige i starten, hvor det blir laget data som ikke er særlig troverdig. Denne feilmarginen kan deretter bli målt og vi kan kalkulere hvor stort tap vi har i bildet, og gjette bedre neste gang. Til slutt er tanken at bildene blir såpass gode at diskriminatoren ikke ser forskjell på ekte og falske bilder.

4.4.1.1 DeblurGAN

DeblurGAN [14] er en ferdiglaget modell tilgjengelig via GitHub [25], basert på "Conditional Generative Adversarial Network (cGAN)", en forbedret versjon av det tradisjonelle GAN-rammeverket. Begge disse modellene har en generator som genererer falske bilder ved hjelp av tilfeldig støy fra de uskarpe bildene vi gir den. Målet til generatoren er å gjøre et falskt bilde så lik det skarpe bildet som mulig slik at en diskriminator ikke lenger ser forskjellen på fiktive og ekte bilder. Forskjellen mellom rammeverkene ligger i at cGAN inkluderer ekstra informasjon til generatoren og diskriminatoren, illustrert som bokstaven “c” i figur 4.33, i tillegg til den tilfeldige bildestøyen. Informasjonen bidrar som et tillegg for å gi bedre kontekst til både generatoren og diskriminatoren, og dermed øke presisjonen deres. Eksempelvis kan det være at både diskriminatoren og generatoren har tilgang til det uskarpe bildet. Ved å inkludere denne tilleggsinformasjonen lærer generatoren å generere mer realistiske bilder ettersom den har ytterligere data å gå på, mens diskriminatoren forbedrer sin evne til å skille mellom ekte og falske bilder. Til slutt vil dette resultere i et forbedret "deblurred" bilde.



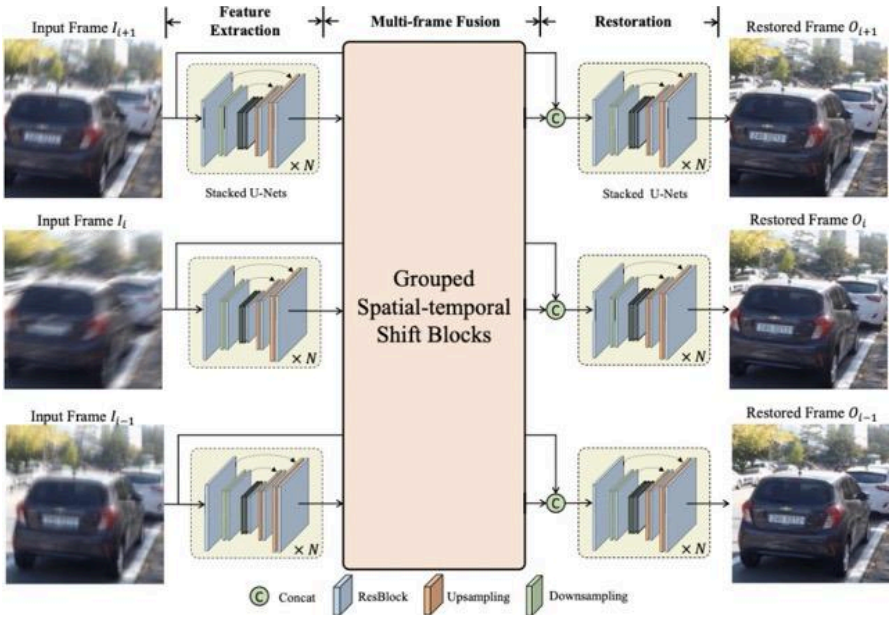
Figur 4.33: Visuelt eksempel på hvordan relasjonen mellom generatoren og diskriminatoren er i cGAN-rammeverket [48].

For å identifisere mulige KI-modeller som oppdragsgiver kan utvide prosjektet med, er det viktig å vurdere både fordelene og ulempene. DeblurGAN er en ganske tung modell som benytter noen ganske avanserte teknikker, og det kan derfor være ganske kostbart å ta denne i bruk. I tillegg er det en modell som begynner å bli litt gammel, med tanke på hvor stor utvikling KI har. Ettersom DeblurGAN er en generativ modell som lager falske bilder, er det også alltid en fare for at bildene kan få tap av innhold. På tross av dette viser DeblurGAN gode resultater sammenlignet med andre modeller slik som figur 4.34 viser. Dette kommer av at modellen blant annet baserer resultatene sine på “content-loss”, som er en måling på hvor mye innhold eller detaljer som går tapt. En siste viktig grunn til hvorfor vi vurderte denne modellen er at den baserer seg på korrigering av enkeltbilder. På grunn av dette står vi fritt til å kunne plassere modellen hvor vi vil i Counting Hero sitt system, og er ikke bundet til å plassere oss før oppdragsgiverens system velger det beste bildet.

Figur 4.34: Venstre - uskarpt bilde, midten - "deblur"-et bilde med modell av *Nah et.al* [50], høyre - "deblur"-et bilde med DeblurGAN [14]

4.4.2 Grouped Spatial-Temporal Shift

"Grouped Spatial-Temporal Shift" [16] er et rammeverk som har til hensikt å øke kvaliteten i videoer ved å se på og sammenligne mange ulike "frames" på samme tid (grouped) i både rom (spatial) og tid (temporal). "Frames" vil videre bli referert til som bilder, ettersom vi jobber med stillestående "frames". Metoden stiller ulike bilder ved siden av hverandre i rekkefølge for å sammenligne dem (shift), samt sette bildene der de hører hjemme. Denne metoden kan hjelpe med å gi kontekst og ytterligere informasjon til "deblurring"-en om bildene rundt det nåværende bildet og dermed gi en bedre tilnærming til et skarpt resultat. Figur 4.35 illustrerer denne metoden i tre etapper: Ekstraksjon av egenskaper, "multi-frame" fusjon og restaurering. Med andre ord består det av ekstraksjon av nøkkelinformasjon fra relevante bilder, deretter kombineres denne informasjonen og brukes til behovet sitt, før rekkefølgen og videoen blir gjenopprettet.



Figur 4.35: Hvordan rammeverket Shift-Net fungerer [16]

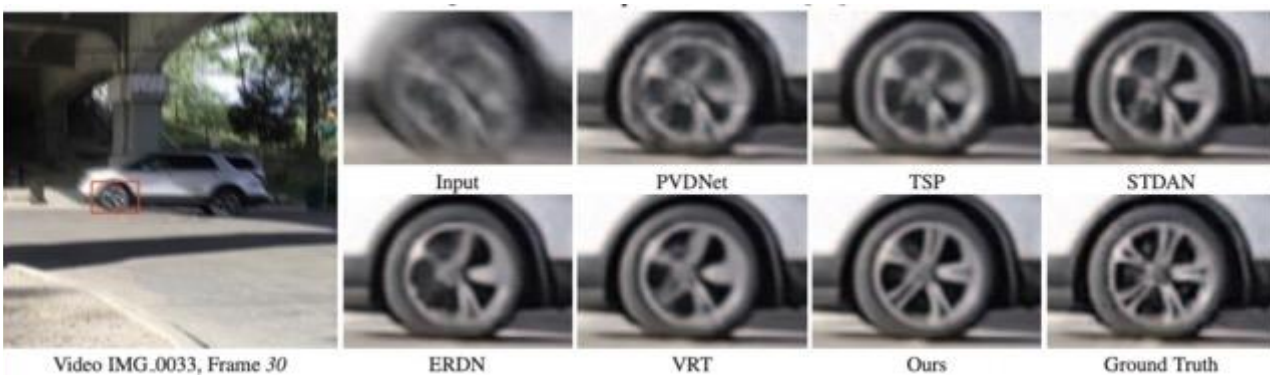
I fasen for ekstraksjon av egenskaper blir informasjon fra de originale bildene til venstre i figur 4.35 tatt ut og prosessert i ulike dimensjoner ved hjelp av nedsampling og oppsampling. De nye oppsamplede bildene blir så kombinert i en “multi-frame fusion” før den nye sammenhengen av "frames" blir konkatenert med de opprinnelige bildene. Når vi så er kommet til restaureringen av det som forhåpentligvis vil bli bedre bilder, har vi både de originale bildene, samt oppsamlede bilder av de originale bildene. Ved å benytte all informasjonen vi har ekstrahert fra de oppsamplede bildene og de originale bildene, vil man kunne restaurere tilbake et nytt og skarpere bilde.

4.4.2.1 Shift-Net

Shift-Net [16] er en ferdiglaget og tilgjengelig modell via GitHub [25], som baserer seg på det ovennevnte rammeverket, “Grouped Spatial-Temporal Shift”. Modellen deler opp videostrømmen i flere enkeltbilder eller "frames". Denne tilnærmingen gir en rikere kontekst for prosessen med “deblurring” ved å gi flere perspektiver og sammenhenger til hvert uskarpt bilde. Ved å dra nytte av denne segmenteringen, kan Shift-Net bidra til å forbedre nøyaktigheten og kvaliteten på resultatene.

På samme måte som DeblurGAN har også Shift-Net noen styrker og svakheter. En vesentlig ulempe er at i motsetning til den forrige modellen er Shift-Net basert på korrigering av bilder i videostrømmer. Dette gjør at dersom Counting Hero velger å anvende denne modellen, må den plasseres på kanten. Dette kommer av at oppdragsgiver sitt system velger ut det beste bildet blant flere for et enkelt kjøretøy, og tar dette med videre i prosessen sin. Modellen må derfor plasseres før dette systemet, noe som skjer på kanten.

En annen ulempe er at modellen har små “blur-kernels” og dermed også små mottakelige felt (“receptive fields”). Dette betyr at det er begrenset hvor store områder modellen kan dra informasjon fra. Det vil si at dersom vi har veldig store områder med “motion blur”, kan det hende at modellen ikke får nok informasjon fra omgivelsene til å utføre “deblurring” like bra som den ellers ville gjort. En annen eventuell svakhet er at Shift-Net benytter noe enklere teknikker enn konkurrentene sine for å kalkulere bevegelser. Dette gjør at for avanserte bevegelser kan det være situasjoner hvor bevegelsen ikke blir beregnet korrekt. Vi ser derimot ikke på dette som et problem ettersom bildene Counting Hero tar, er av biler på strake veier med enkle bevegelser, og metoden er derfor tatt med her. En annen fordel med modellen er at det er en lettvektig tilnærming til “deblurring” som likevel gir overbevisende resultater slik vi kan se i figur 4.36.



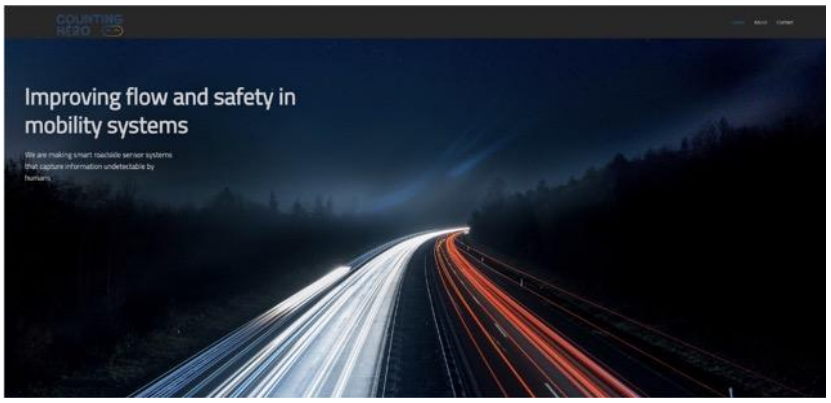
Figur 4.36: En sammenligning av resultater mellom Shift-Net (ours) og andre konkurrerende metoder [16]

4.5 Nettside

Hensikten med nettsiden er å tilby en enkel løsning, designet for intern bruk i Counting Hero. Formålet med nettsiden er å tilrettelegge for enkel evaluering av kvaliteten på bildene fra kameraene. Nettsiden er utformet som en loggføringsside som gir brukerne muligheten til å søke gjennom og få en oversiktlig fremstilling av eventuelle feil med bildekvaliteten. Målet er å visualisere resultatene av deteksjonen på en ryddig og oversiktlig måte.

Vi har som mål å realisere en nettside med bra "User experience design" (uXD) for Counting Hero. "uXD" representerer en prosess innen webutvikling som tar sikte på å øke brukertilfredsheten ved å forbedre brukervennlighet, tilgjengelighet og effektivitet i brukerens interaksjon med nettsiden [51].

Designet av nettsiden i dette prosjektet ble opprinnelig basert på Counting Hero sin daværende hjemmeside [4] og HTML-prototypen som ble designet i kapittel 2.1.2. Nettsiden til oppdragsgiver gjennomgikk derimot en betydelig oppdatering i prosjektperioden og fremstår i dag med modifiserte farger og stil [52]. I figur 4.37 illustreres den tidligere versjonen av Counting Heros nettside.



Figur 4.37: Tidligere versjon av Counting Hero sin nettside [4].

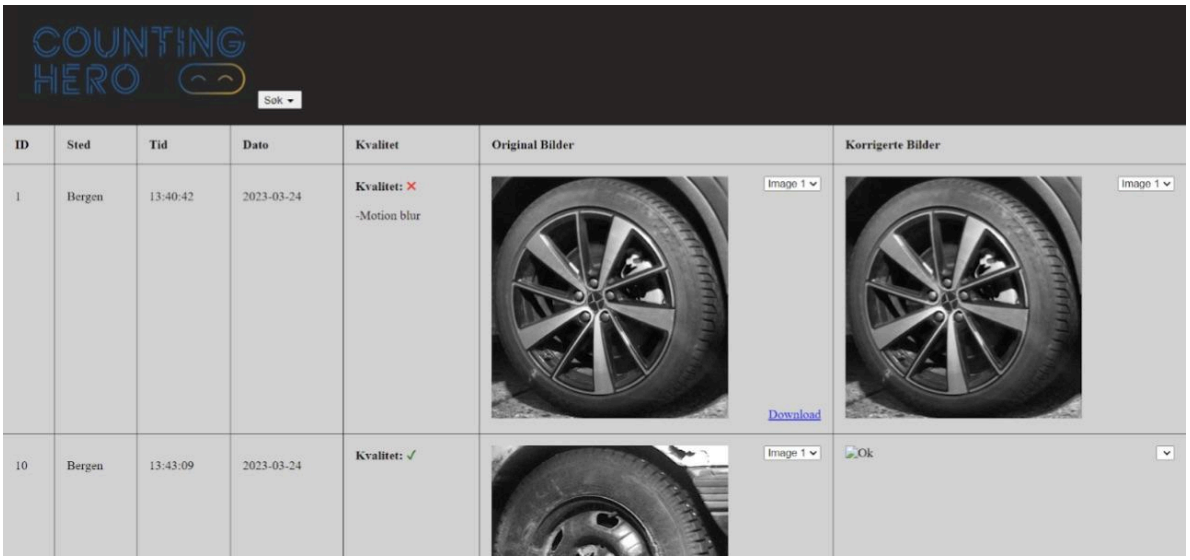
Nettsiden er ikke hovedfokuset i dette prosjektet, derfor er funksjonalitet og kompleksitet forholdsvis enkel. Designet er ikke oppdatert opp mot den nye versjonen av Counting Hero sin nettside [52] fordi dette ble nedprioritert og oppdaget sent i prosjekt.

Gruppen besluttet at vi ikke hadde behov for en avansert nettside ettersom hensikten bare var å utføre en spesifikk oppgave. Dette innebærte muligheten til å søke og identifisere eventuelle feil i bildekvaliteten, samt legge dette frem på en klar og oversiktlig måte. Målet var å effektivt sjekke om det var noe galt med bildene. En dypere diskusjon om styrkene og svakhetene til nettsiden vil bli utforsket nærmere i kapittel 6, da det er flere områder som vi ønsker å adressere.

4.5.1 Design

Vi benyttet Flask [22], et Python-bibliotek, for enkel oppsett av nettsiden. Nettsidens struktur ble utformet med HTML og pyntet med CSS, mens implementeringen av sortering- og søkefunksjonalitet ble gjort med JavaScript.

Etter at den relevante informasjonen har blitt prosessert, som beskrevet i kapittel 4.1.1.1, henter vi objektene og bildene, og presenterer dem på nettsiden. Denne informasjonen lagres lokalt, og bare filstien til mappen hvor bildene er lagret, benyttes for å vise dem på nettsiden. I figur 4.38 illustreres nettsiden ved oppstart.



Figur 4.38: Bilde av nettsiden vår

Gråfargen og stilen vi ser i figur 4.38 vil ikke være ukjent for brukerne av nettsiden, ettersom den ligner på det tidligere nettstedet til Counting Hero. Hovedfokuset vi hadde når vi laget nettsiden var at den skulle være lettleslig, behagelig og visuelt fin [51]. Ved å tilpasse oss noe av webdesignet til oppdragsgiver, kan det forbedre aksepten av vår nettside.

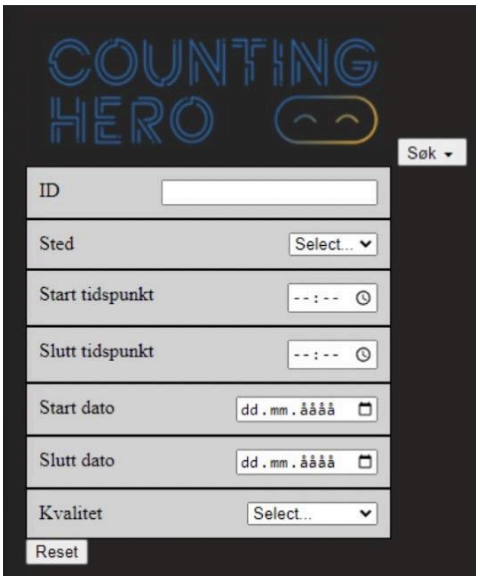
4.5.2 Logo og søkefelt

Når man åpner nettsiden er søketabellen skjult bak en søkeknapp, som illustrert i figur 4.39. Dette har vi gjort ettersom vi ønsker å ha en klar og ryddig nettside og skjuler derfor unødvendig informasjon som ikke må presenteres umiddelbart. Dette bidrar til en forbedret brukeropplevelse [51].

Figur 4.39: Logo og søkeknapp på nettsiden vår

Vi har utformet nettstedet med logoen og søkeknappen plassert øverst til venstre, ettersom dette er karakteristisk for Counting Heros tidligere hjemmeside [4], og fordi brukere av nettsiden typisk retter oppmerksomheten mot det øvre venstre hjørnet [51].

Vi har forsøkt å gjøre søkefunksjonaliteten på nettstedet så bra som mulig slik at Counting Hero kan oppnå en tydelig oversikt over feil, tidspunktene for feil og identifikasjonen til bildene. Nettstedet inkluderer en søketabell, som vist i figur 4.40, som muliggjør sortering av bilder etter ulike kategorier, inkludert ID, sted, dato, start- og slutttidspunkt, start- og sluttdato og kvalitet. Ved behov kan tabellen tilbakestilles ved å trykke enten på reset-knappen eller på logoen, slik at søket nullstilles. Alle bilder vil da igjen vises slik de gjør når nettsiden først åpnes.

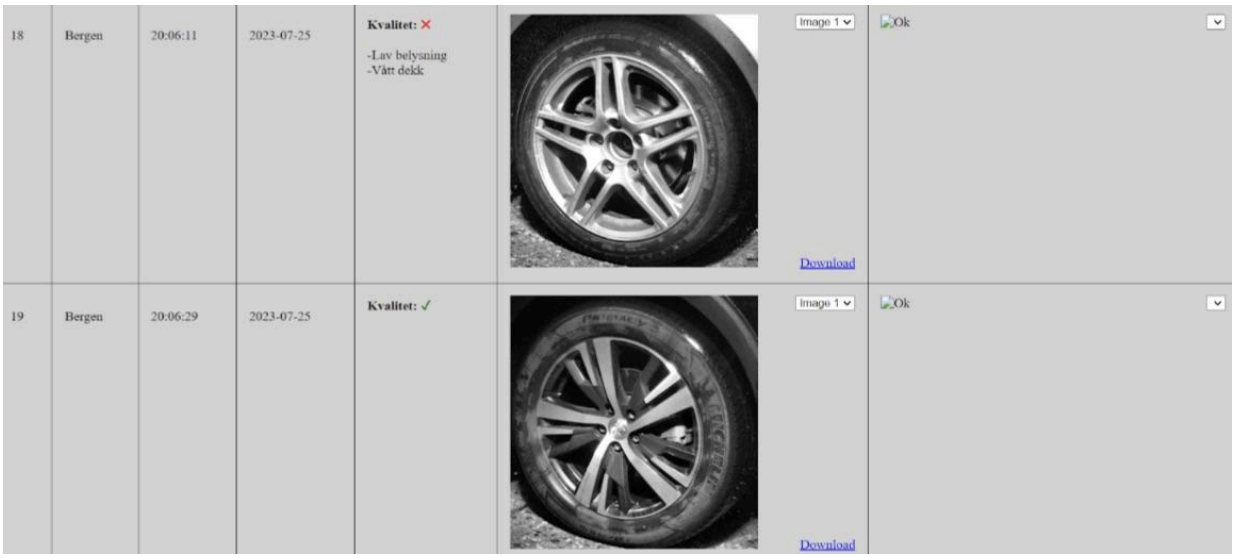


Figur 4.40: Flere søkefelt viser når man trykker på “søk” knappen.

Betegnelsene for de ulike variablene i søkefeltene blir forklart i kapittel 4.1.1.1, med noen unntak. Variablene starttidspunkt, sluttidspunkt, startdato og sluttdato brukes til å spesifisere søketidsrommet i tabellen. Dette gir brukeren mulighet til å identifisere kvalitetsproblemer over lengre tidsperioder.

4.5.3 Objekttabell

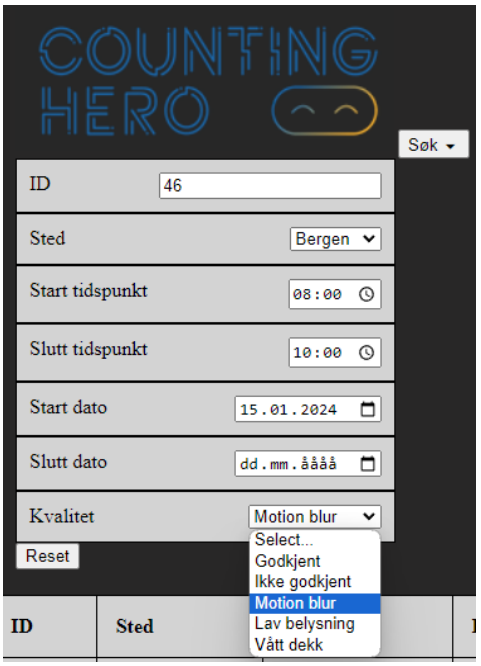
I figur 4.41 presenteres objekttabellen på nettsiden, som viser noen av bilobjektene sammen med tilhørende bilder. Kolonnene inneholder variabler som “ID”, “Sted”, “Tid”, “Dato”, “Kvalitet”, “Originale Bilder” og “Korrigerte Bilder”. For originale og korrigerte bilder er det gjort tilgjengelig en nedtrekksmeny som tillater valg av bilder, vanligvis to bilder per kjøretøy - ett for hvert dekk. Antallet bilder kan variere avhengig av kjøretøyets type. Hvert av de originale bildene kan også lastes ned ved å klikke på "download" nederst til høyre ved siden av bildene. Denne funksjonen for nedlasting kan benyttes til å laste ned en kopi av bildet lokalt i nettleseren, for eksempel for loggføring og innsamling av bilder som er feil.



Figur 4.41: Objekttabell.

4.5.4 Resultat av søk på nettsiden

I figur 4.42 og 4.43 presenteres et eksempel på et søk utført på nettsiden. Det er hensiktsmessig å kunne spesifisere ulike variabler i søkefeltet, da dette muliggjør en rask oversikt over hvilke tidsperioder og klokkeslett hvor det foreligger feil i bildekvaliteten. På nettsiden kan man også søke etter de forskjellige kvalitetene slik som for eksempel godkjente bilder, “dårlig lys”, “motion blur” med mer. I figur 4.41 vises også det korrigerte bildet, som i dette tilfellet er identisk med det opprinnelige bildet, da korrigeringen ikke er implementert i prosjektet. Imidlertid har vi forberedt for implementering av dette, i tråd med den opprinnelige planen. I figur 4.43 kan man se at deteksjonen har oppdaget “motion blur” og “lav belysning” på bildet av dekket.



Figur 4.42: Eksempel på brukerdefinert søk på nettsiden vår.

Figur 4.43: Resultat av brukerdefinert søk.

I figur 4.44 vises et eksempel på en hendelse der vått dekk er blitt oppdaget. Her observeres tydelig at det er mange vanndråper i bildet.

Figur 4.44: Et bilobjekt som er flagget for vått dekk.

5 RESULTATER

Resultatene av prosjektet ble evaluert i henhold til evalueringsplanen fremlagt i kapittel 3.5. Gjennom prosjektet har vi systematisk gjennomført kontinuerlige evalueringer, inkludert brukertesting, enhetstesting, testing i "MedieManipulator", og avsluttende dybdeintervju. For å evaluere resultatene rundt hvor godt deteksjonene fungerte, har vi også laget en forvirringsmatrise [53]. Disse evalueringsmetodene er benyttet for å vurdere sluttresultatet og innsatsen vår, samt demonstrere om løsningen er i samsvar med forutsetningene.

5.1 Evalueringsmetoder

Brukertesting:

I løpet av prosjektet hadde vi regelmessige møter med oppdragsgiveren, hvor vi mottok tilbakemeldinger angående prosjektets fremgang. Dette bidrog til at vi kunne få luftet idéer og fremgangsmåter, samt få profesjonell assistanse med teoriområder som kunne være ekstra vanskelige. Det var også en fin måte for oppdragsgiver å få holde seg oppdatert i prosjektet.

Enhetstesting:

I løpet av prosjektet har vi benyttet enhetstesting i VSCode ved hjelp av Pytest [54]. Et enkelt-å-bruke testrammeverk med et rikt økosystem av plugins, som gjør det til et fleksibelt verktøy for testing [55].

Testing i “MedieManipulator”:

En god del av funksjonene, spesielt i klassen “MedieManipulator”, som står beskrevet i kapittel 4.1.1, er laget for å undersøke og kunne evaluere resultatene fra deteksjonene, samt hvor aksepterte de er.

Dybdeintervju:

Mot slutten av prosjektet ble det gjennomført en sluttevaluering i form av et dybdeintervju med Counting Hero. For å gjennomføre et suksessfullt intervju konkluderte gruppen med at det var viktig å følge noen retningslinjer [56]. Blant disse retningslinjene valgte vi å ikke inkludere mer enn 15 hovedspørsmål for å fokusere på kvaliteten, mens vi fulgte opp med tilleggsspørsmål der vi så det som relevant. Slik mange intervju blir gjennomført, var det meget viktig at vi også stilte åpne spørsmål. Dette gjorde at personen vi intervjuet kunne gi personaliserte og genuine svar, uten at vi ledet han i noen retning.

I intervjuet ble det stilt spørsmål knyttet til de funksjonelle egenskapene som var beskrevet i visjonsdokumentet for nettsiden og kanten. Videre ble det også stilt spørsmål knyttet til problemstillingen og prosjektets mål. Spørsmålene ble kategorisert i tre hovedgrupper: nettside, deteksjon og generelle spørsmål. Svarene som ble gitt av oppdragsgiver under intervjuet, ble dokumentert mens oppdragsgiver var til stede. Disse spørsmålene og svarene presenteres i neste kapittel 5.2, "Evalueringsresultat".

5.2 Evalueringsresultat

Brukertesting:

Oppdragsgiveren hadde ikke direkte tilgang til prosjektet vårt utenom møtene, ettersom alt var lokalt lagret på vår maskin. Imidlertid ga oppdragsgiveren muntlige tilbakemeldinger om hva vi kunne gjøre for å forbedre løsningen vår gjennom hele prosjektet. Disse tilbakemeldingene var svært nyttige for å gjennomføre deteksjonen på en bra måte og fremheve viktige aspekter på nettsiden vår.

Enhetstesting:

Som beskrevet i kapittel 3.5 brukte vi enhetstesting for å sikre at enkelte komponenter i programmet fungerte som forventet. Resten av programmet gikk på testing av metoder for detektorene, noe som “MedieManipulator”-en hadde ansvar for å evaluere. Enhetstesting ble derfor bare brukt til å evaluere noen få gjenværende aspekt av koden. Vi opplevde enhetstesting som både effektiv og rask, og dermed passende for testing av kodebiter. Mer om hvordan man kjører enhetstestene står beskrevet i kapittel 10 i systemdokumentasjonen.

Testing i “MedieManipulator”:

Funksjonene i “MedieManipulator” har blant annet blitt brukt til å lage histogrammer. Disse ble aktivt brukt for å evaluere fremgangen vår i å nærme oss mer pålitelige og effektive terskelverdier over tid. Klassen har vært viktig for å komme frem til de resultatene vi fikk, samt visualisere dem.

Dybdeintervju - 25.04.24 med Ruben Patel fra Counting Hero:

Før dybdeintervjuet hadde vi et møte hvor vi gjennomgikk resultatene våre ved å vise frem nettsiden og hva som ble flagget som dårlig bildekvalitet. Notater fra samtalen før dybdeintervjuet står i prosjekthåndboken, referat fra 25.05.24.

Nettsiden:

- Hvordan oppleves nettsiden?

Informativ. Den viser det den skal vise.

- Hvor enkelt er det å navigere seg rundt og søke på nettsiden?

Veldig enkelt. Ikke komplisert.

- Dersom du skulle dratt frem noe positivt og noe som kunne vært bedre med nettsiden, hva ville det vært?

En positiv ting er at den er enkel og viser det den trenger. En ting som kunne vært bedre er at nettsiden kunne vært mer estetisk.

Detektorer:

- Hvor relevant mener du at vår problemstilling er for Counting Hero?

Veldig relevant. Vi trenger en slik detektor i vårt system. Vi vet ikke hvorfor bildene blir dårlige, men med deres problemstilling kan vi flagge problemene med bildene.

- Basert på diskusjonene underveis i prosjektet og den endelige løsningen, vil du si at vi har brukt gode fremgangsmetoder? Hvorfor / Hvorfor ikke?

Ja, dere har vært metodisk å sette det opp. Dere har funnet ut av mye selv. Bra fremgangsmetode.

- I hvor stor grad vil du si at resultatene fra lysdeteksjonen fungerer i forhold til ønskede resultat

Det ser bra ut. Overbevisende resultater.

- Vil du si at arbeidet vårt med "motion blur”-detektering har resultert i en dypere forståelse for hva som kreves for å lage en slik deteksjon?

Ja, dere har innovative og kreative metoder, som for eksempel å telle regndråper. Vi ser at i de ideelle tilfellene virker det greit, men når man får ekte data med mye variasjon, så er det mer utfordrende.

- Vil du si at problemstillingene vi har oppdaget, og beslutningene vi har tatt under utviklingen, er relevante for videreutvikling av systemet? Hvorfor / Hvorfor ikke?

Ja, vi ser at dette ikke er en triviell problemstilling. Mye variasjoner i bildene skaper problemer. Dere har løst dette på en god måte med de bildene dere har hatt tilgjengelig.

Generelt:

- Hvordan synes du at gruppen har tilpasset seg løpende evalueringer i form av tilbakemeldinger og anbefalinger?

Bra. Vi har hatt flere samtaler i løpet av prosjektet, og dere har fulgt opp på tilbakemeldinger.

- Dersom gruppen skulle arbeide videre med prosjektet, hva ville du likt å endre?

Ville likt å se eksempler og resultater av eksisterende systemer for detektorene. Videre arbeid ville datasettet blitt mye større.

- Vil du si at en slik løsning kan bidra til å redusere tiden det tar å kvalitetssjekke bildene i dag?

Vi gjør ingen direkte kvalitetssjekking av bildene i dag, men det ville vært et godt hjelpemiddel til å finne ut av hvorfor vi får unormale tellinger. Det vil redusere tiden det tar å finne feil.

- Er det noe mer du ønsker å legge til som vi ikke har tatt opp?

Det virker som dere har jobbet bra gjennom prosjektet og fordelt oppgavene fint.

5.3 Prosjektresultat

5.3.1 Prosjektresultat og evaluering av detektorene

For detektorene endte vi opp med en del forskjellige terskelverdier. Alle terskelverdiene er oppsummert i tabell 5.1.

Lysdetektoren bruker en løsning hvor bildet består av tre biter av dekket - nedre, øvre og høyre del. Vi tar så snittet av disse verdiene. Terskelverdien for lysdetektoren ble til slutt satt til 55, noe som betyr at dersom lysverdien er lavere enn 55, så anser vi bildet som for mørkt.

Vanndråpedetektoren går ut på å telle antall dråper i bildet. Siden dette var problemer hos veldig lyse bilder med mye støy i skyggene sine, valgte vi to forskjellige terskelverdier. Vi har en terskelverdi som sjekker om lysnivået på bildet er under 50. Dersom bildet oppfyller dette så har vi en terskelverdi som sier at bilder med mer enn 23 dråper er våte. Den andre terskelverdien sier at alle bilder med mer enn 30 dråper anses som våte. Her har vi ikke noe terskel for lysnivå siden noen av de våte bildene fikk høyt lysnivå grunnet gjenskinn fra vannet. Dette er den biten av koden som skaper flest tilfeller av gode bilder som ble markert som dårlige. Siden noen lyse bilder

fikk mye støy i skyggene sine. Det var bare noen få av disse bildene som ble registrert med mer enn 30 dråper. Vi kunne justert terskelen for færre feilmarkerte gode bilder, men vi bestemte oss for at det var bedre å registrere flest mulig faktisk våte bilder.

“Motion blur”-detektoren våres endte opp med fire forskjellige verdier. Dette er fordi vi satt opp to ulike metoder for beregning av variasjon i bilder som vi presenterte i kapittel 4. Disse metodene får to forskjellige terskelverdier basert på lysnivået.

- For klare bilder så vi på forskjellen i varians mellom øvre og nedre del av dekket. Hvis lysverdien er over 70 blir bilder med en forskjell på mer en 3.5 markert som “motion blur”
- For de uklare bildene endte vi med metoden som regner ut gjennomsnittet av variasjonene på den øvre, nedre og høyre delen av dekket. Hvis lysnivået er over 60 blir bilder med mindre enn 0.015 i variasjonen merket som “motion blur”. Bildene under et lysnivå på 60 blir bildene med mindre enn 0.02 i variasjon markert som “motion blur”

Tabell 5.1: Terskelverdier

Detektor	Lavt lysnivå	Våte dekk	Motion blur
Terskel	lys < 55	Dråper > 30, eller Lys < 50 og dråper > 23	Lys > 60 og gj.snitt < 0.015, eller Lys < 60 og gj.snitt < 0.02, eller Lys > 70 og var > 3.5, eller Lys < 70 og var > 1.5,

Disse terskelverdiene er basert på et relativt lite datasett på 103 bilder. Testing på et større datasett kan derfor være aktuelt ved behov for å gjøre endringer på terskelverdiene.

For å analysere resultatene vi fikk, samt finne ut hvor nøyaktig detektorene var, bestemte vi oss for å benytte en forvirringsmatrise. En slik matrise ville også hjelpe oss til å klassifisere bildene, noe som i vårt tilfelle var om et bilde var dårlig eller ikke. Gruppen lagde forvirringsmatrisen ved å gå gjennom alle bildene og tildele dem fire forskjellige kategorier. Kategoriene var som følger:

- "OK": Bilder som ble ansett som tilstrekkelig lesbare for Counting Hero-systemet, selv om de kunne være litt fuktige eller mørke.
- "Lys": Bilder som var for mørke.
- "MB": Bilder som viste tegn på "motion blur".
- "WET": Bilder av våte dekk.

Et bilde kunne bli kategorisert med hver enkelt av tre sistnevnte kategoriene, samt kombinasjoner av dem. Eksempelvis kunne et bilde ha både “motion blur” (MB) og ha dårlige lys (Lys). Dersom bildet var kategorisert med "OK", hadde det derimot utelukkende denne merkingen.

Deretter markerte vi filnavnene til en del bilder med merker etter å ha manuelt gått gjennom hva som var galt med dem. Disse bildene ble så brukt til testformål i en kode vi utviklet for å sammenligne resultatene fra detektorene med de manuelle merkene vi hadde gitt bildene. Denne koden genererte fire forskjellige verdier:

- Ekte-positiv for bilder som var merket som "OK" og ikke ble identifisert som dårlige av noen av detektorene.
- Falsk-positiv for bilder som hadde feil, men ikke ble merket som dårlige (dette var det minst ønskede resultatet, da det kunne forvirre Counting Hero-systemet).
- Ekte-negativ for bilder som ble identifisert med feil og hadde en eller flere problemer.
- Falsk-negativ for bilder som var "OK", men likevel ble merket med feil.

Etter å ha kjørt systemet flere ganger og justert verdiene så godt som mulig innenfor tidsrammene vi hadde, oppnådde vi følgende resultater.

Tabell 5.2: Forvirringsmatrise

		Faktisk Resultat	
		Ekte	Falsk
Forventet resultat	Positiv	72	2
	Negativ	21	8

I tabell 5.2 presenteres de fire diskuterte verdiene, sammen med antallet bilder som ble kategorisert. Vi har totalt et relativt lite datasett på 103 bilder, derav 80 gode bilder og 23 dårlige bilder. Det vil si at noen enkeltbilder kan ha stor påvirkning på resultatene. For å få en høyere pålitelighet for resultatene, kreves det et mye større datasett enn hva vi har hatt tilgang til. Vår analyse av disse dataene viser derimot at flertallet av bildene blir korrekt klassifisert som "gode" av våre detektorer. Imidlertid representerer kategoriene falsk-positiv og ekte-negativ samtlige av de dårlige bildene. Bare to av de totalt 24 dårlige bildene unngikk å bli identifisert som dårlige, og endte i falsk-positiv. Dette var kategorien vi ønsket desidert minst bilder i, noe terskelverdiene vi definerte sørger for. De to falsk-positive bildene er illustrert under.

Figur 5.1 og 5.2: De to bildene som ble identifisert som falsk-positive.

I figur 5.1 og 5.2 ser vi to bilder som skulle blitt klassifisert som dårlige, selv om utfordringene i bildene ikke nødvendigvis ville være problematiske. I tilfellet av figur 5.1 ser vi en tilfredsstillende lysstyrke, men den avgjørende faktoren er at figuren potensielt inneholder for mye "motion blur" til at vi eller en KI-modell kan lese av informasjonen på dekket. Dette var typen bilder vi håpet å kunne detektere ved å bruke en detektor som sammenlignet øvre og nedre deler av dekket, men på grunn av skyggelegging fikk vi ikke dette til. Dette kom av at skyggelegging på nedre del av dekket resulterte i en lavere variasjon i dette området, og dermed reduserte forskjellen mellom den øvre og nedre delen. Det neste vi så på var falsk-negative bilder.

Figur 5.3, 5.4 og 5.5: Bilder som ble falskt identifisert som dårlige. Figur 5.3 på grunn av en skygge, figur 5.4 som ligger i grenseland med hint av feil og figur 5.5 som er helt feil identifiserte dårlige bilder som er gode.

Av de 80 bildene som ble klassifisert som gode i tabell 5.2, ble kun 8 identifisert som dårlige. Flere ulike årsaker til dette ble observert. Eksempelvis, i tilfelle av figur 5.3, var det et spesialtilfelle hvor en stor skygge dekket den øvre delen av dekket, noe som reduserte variasjonen i pikslene og førte til markeringen av "motion blur".

En annen kategori var bilder som lå på grensen til å bli klassifisert som dårlige. Disse bildene viste kanskje litt "motion blur", lav belysning eller inneholdt noen vanndråper, som i tilfellet med figur 5.4. Til tross for at noen av disse bildene hadde et par av de karakteristikkene som vanligvis ville ført til at de ble klassifisert som dårlige, var teksten likevel lesbar på dem. Derfor ble de definert som “OK” av oss, selv om noen av dem kunne ha vært vurdert annerledes.

Problemet representert ved figur 5.5, involverte helt gode bilder av dekk som ble feilaktig klassifisert som dårlige. Fem av de 8 dårlige bildene falt inn i denne kategorien. Disse bildene ble alle detektert av vanndråpedetektoren og viste mørke skygger med mye støy. Til tross for dette var fortsatt teksten på dekket lesbar, og de skulle ha vært klassifisert som gode. Vi forsøkte å adressere problemene med disse bildene ved å først vurdere lysnivået på bildet, ettersom disse bildene alltid hadde god belysning. Dette gjorde det noe lettere å sette terskelverdiene, men det var utfordrende å sette gode terskler ettersom en for høy verdi kunne føre til at noen dårlige bilder ikke ble merket som dårlige. Basert på dette konkluderte vi med at det var bedre å ha fem bilder merket som falsk-negative enn å risikere å ha to ekstra bilder markert som falsk-positiv. På bakgrunn av dette

satt vi terskelverdiene i tabell 5.1, og fikk resultatene i tabell 5.2. Dilemmaet vi hadde her kunne muligens vært redusert gjennom ytterligere arbeid og testing med vanndråpedetektoren. Siden denne detektoren fikk minst oppmerksomhet, er det mulig at den kunne blitt mer nøyaktig med mer tid og testing.

For å evaluere detektorene benyttet vi oss av resultatmatrise, representert i tabell 5.3, for utregning av hvor nøyaktig, presis og sensitive detektorene var. Vi regnet også ut F1-Scoren til detektorene.

$$Nøyaktighet = \frac{(TP+TP)}{(TN+FP+FN+TP)}$$

Ligning 5.1: Formelen for beregning av nøyaktighet [53]

For å regne ut nøyaktigheten på detektorene, brukte vi formelen over. I ligning 5.1 er "TN" ekte-negativ, "TP" er ekte-positiv, "FN" er falsk-negativ og "FP" er falsk-positiv. Ved å bruke verdiene fra matrisen i figur 5.1, får vi resultatet 0.9, som vil si en 90% nøyaktighet. Disse verdiene er som tidligere nevnt basert på et relativt lite datasett, men er likevel et lovende resultat. Nøyaktigheten vår kunne blitt enda bedre, men siden vi vektla å detektere så mange mulige av de negative bildene ofret vi nøyaktigheten våres til å detektere gode bilder. Vi kan regne ut presisjonen for detektering av dårlige og gode bilder med formelen i ligning 5.2.

$$Presisjon = \frac{TP}{TP+FP}$$

Ligning 5.2: Formelen for beregning av presisjon [57]

Vi gjorde denne utregningen for både de positive og negative resultatene. De positive var lik 0.97, som tilsvarer 97% presisjon. Dette vil si at majoriteten av bildene som blir klassifisert som gode er faktisk gode. Dette vil si at nesten alle de dårlige bildene som kommer gjennom systemet vårt blir fanget opp som dårlige. De negative resultatene ga 0.72, som tilsvarer 72%. Det vil si at nesten én av tre bilder som blir markert som dårlige, er gode bilder. Hvis et bilde er bra, vil systemet til Counting Hero bruke bildet uavhengig av hva vår detektor sier. Det er derfor ikke et stort problem at gode bilder blir markert som dårlige. For å få en klarere oversikt av kvaliteten til detektorene kan vi regne ut sensitiviteten / gjennvinningsmetrikken med formelen i ligning 5.3.

$$Sensitivitet = \frac{TP}{TP+FN}$$

Ligning 5.3: Formelen for beregning av sensitivitet [57]

Vi brukte denne formelen for å regne ut sensitiviteten til både positive og negative resultater. Positive ga 0.9, noe som vil si at 90% av de gode bildene som ble detektert som positive. Negative ga resultatet 0.91, noe som vil si at 91% prosent av de dårlige bildene som ble registrert som negative. Dette er et godt resultat, ettersom det å finne ut hva som er galt når det er problemer med bildene, er hele poenget med detektorene. Her er det kun to bilder som ikke ble merket som dårlig, men dette utgjorde fortsatt ni prosent av hele det negative datasettet. På grunn av at vi har så få bilder vil en test med et mye større datasett kreves for å få et pålitelig resultat. Til slutt kan vi

regne ut F1Scoren som gir oss det beste forholdet mellom presisjon og sensitivitet. Den regner vi ut med formelen i ligning 5.4.

$$F1Score = \frac{2*precision*recall}{precision+recall}$$

Ligning 5.4: Formelen for beregning av F1-Score [57]

Vi regnet først ut F1Score for de positive og fikk et resultat på 0.93, tilsvarende en balanse på 93%. Dette vil si at vi har både en høy presisjon og sensitivitet for å identifisere de positive tilfellene. De negative ga derimot et resultat på 0.8, som gir en balanse på 80%. For de negative har vi en del lavere resultat ettersom vi hadde en lavere presisjon. Men som nevnt kommer dette fra valget vårt med å prioritere minst mulige falske-positive resultater. Resultatene for hvor gode detektorene var, er vist i matrise / tabell 5.3 under.

Tabell 5.3: Resultatmatrise for detektorene

	Nøyaktighet	Presisjon	Sensitivitet	F1-Score
Positiv	0.9	0.97	0.9	0.93
Negativ	0.9	0.72	0.91	0.8

5.3.2 Prosjektresultat - “motion blur”-korrigering

Vi hadde ingen spørsmål til oppdragsgiver angående “motion blur”-korrigering under intervjuet fordi vi ikke hadde noe implementasjon å vise til. Det vi har gjort er å undersøke mange ulike modeller, og kommet frem til to metoder som vi ville fremheve. Av disse er det en som korrigerer enkeltbilder, mens den andre korrigerer videostrømmer. Vi undersøkte DeblurGAN og Shift-Net som begge virket lovende for prosjektet vårt. Disse modellene kan Counting Hero vurdere dersom implementasjon av “motion blur”-korrigering blir aktuelt.

5.3.3 Prosjektresultat - nettside

Nettsiden er der for å vise frem resultatene fra deteksjonen. Her kan brukeren søke og se hvilke bilder som blir oppdaget som feil blant bildene som vi har hatt tilgjengelig gjennom prosjektet. Ettersom vi ikke implementerte “motion blur”-korrigering er det en tom kolonne på nettsiden som ikke viser noe resultat. Det er derimot lagt opp til at det kan legges inn dersom Counting Hero ønsker det. Hvis oppdragsgiver ønsker å teste flere bilder med nettsiden vår, må de beskjære bildene slik at de bare viser det området som inneholder dekket, og deretter legge dem inn i den riktige mappen før de kjører prosjektet. Nettsiden vil da vise de nye bildene.

6 DISKUSJON

I dette kapitlet vil vi diskutere de valgene som ble gjort i løpet av prosjektet. Vi ser nærmere på valg av oppgave og de definerte avgrensningene, samt deres påvirkning på resultatet. Videre ser vi på arbeidsprosessen og valget av verktøy. Vi identifiserer både styrker og svakheter i prosjektet. Til slutt reflekteres det over potensielle forbedringsmuligheter dersom prosjektet skulle gjentas.

6.1 Valg av oppgave, avgrensninger og avvik

6.1.1 Valg av oppgave

Som presentert i kapittel 3.1.1 og 3.1.2 angående valget av oppgave, valgte vi alternativ 2 som vår foretrukne løsning. Dette viste seg å være et vellykket valg med tanke på resultatene vi fikk. Gruppen opplevde økt fleksibilitet og kunne vise større kreativitet i utførelsen av prosjektet. Grunnet begrenset erfaring innledningsvis innen feltene vi jobbet med, ville alternativ 1 ha krevd betydelig mer forståelse fra gruppen, med behov for mer utdypende undersøkelser av tung teori og dokumentasjon rundt eksisterende løsninger.

6.1.2 Avgrensninger

En betydelig del av prosjektperioden i januar og februar ble brukt til å avgrense oppgaven skikkelig. Dette kommer frem da den endelige løsningen er avgrenset mye fra den opprinnelige oppgaven om evaluering av modeller for "motion blur"-korrigering. En tidligere definert problemstilling ville ha bidratt til at gruppen kunne hatt mer tid til detektorene. Da dette representerer et område hvor mer tid og undersøkelser antageligvis ville ha ført til bedre resultater, er dette noe vi tar med oss videre.

Til tross for begrensede kunnskaper innen feltet, samt avgrensninger og manglende implementering av "motion blur"-korrigering, mottok vi positive tilbakemeldinger under intervjuet med oppdragsgiver. I intervjuet, spesielt i diskusjonen om "Detektorer", ble det bemerket at resultatene fra lysdeteksjon og "motion blur"-deteksjon med telling av vanndråper ble løst på en god og kreativ måte, og ble ansett som relevante for videreutvikling.

6.1.3 Avvik

Manglende implementeringen av "motion blur"-korrigering resulterte i en ufullstendig nettside som manglet de korrigerte bildene. Som beskrevet i kapittel 2.2, betraktet vi implementeringen av "motion blur"-korrigering som urealistisk innenfor de gitte tidsrammene for prosjektet. Til tross for at dette ikke var hovedfokuset i den endelige løsningen, ble det likevel inkludert som en del av kapittel 4 og presentert i form av utseende til nettsiden.

"Motion blur"-korrigering var et område som ble undersøkt gjennom prosjektperioden, men viste seg å være mer utfordrende enn først antatt. Dette ledet til at vi ikke fikk gjennomført noen bildekorrigering, og har dermed ingen konkrete resultater å fremvise på dette området. Det tok blant annet lang tid å utføre kjøringen på grunn av en rekke uforutsette feil med “deblurrings”-kodene som vi hadde pekt oss ut. Vi opplevde vanskeligheter med å kjøre koden på flere av våre maskiner, hovedsakelig på grunn av omfattende avhengigheter som måtte installeres og problemer med maskinvare. Det største problemet var imidlertid knyttet til behovet for å trene modellene på egne data. Dette ville være nødvendig for å tilpasse modellene skikkelig til bruksområdene Counting Hero hadde trengt den til. For å få en pålitelig og akseptert modell ville dette krevd store mengder med data, og det ville i tillegg tatt mye tid å trene den opp.

Sammenlagt førte dette til at gruppen, sammen med oppdragsgiver, valgte å forkaste implementeringen av “motion blur” -korrigering. I stedet gjennomførte vi et større litteratursøk av modellene og fokuserte mer på å forbedre kvaliteten på deteksjonen av "motion blur".

6.2 Valg av verktøy og arbeidsprosess

6.2.1 Valg av verktøy

I tråd med beskrivelsen i kapittel 2.2, valgte vi å ikke integrere vår utvikling i Counting Hero sitt system, da dette ikke var relevant for å oppnå målet for oppgaven. Ved å kjøre alt lokalt på vår egen maskin, kunne vi fokusere på å oppnå best mulige resultater med deteksjonen, og unngikk unødvendig tid som ville blitt brukt på å sette oss inn i systemer som ikke var relevante for oppgaven. Som beskrevet i visjonsdokumentet i kapittel 3.3, ville for eksempel bruk av en database vært aktuelt i et tilfelle hvor vi hadde benyttet sanntidsvideo i motsetning til de bildene vi har brukt. Oppdragsgiver informerte oss i intervjuet under seksjonen "Generelt" at de ikke gjør noe direkte kvalitetssjekking av bildene i dag, altså at det ikke var noen eksisterende systemer å integrere prosjektet med. Dette ga oss betydelig frihet til å selv velge hvilke utviklingsverktøy, programmeringsspråk og rammeverk vi mente var best egnet for å løse oppgaven.

6.2.2 Arbeidsprosess

Oppdragsgiver mente i intervjuet under “Detektorer” at vi hadde hatt en god fremgangsmåte i arbeidet vårt. I spørsmål under "Generelt" bemerket oppdragsgiver at vi har fulgt opp på tilbakemeldinger, jobbet godt gjennom prosjektet og fordelt oppgavene fint. Valget av en agil arbeidsmetodikk, slik beskrevet i kapittel 3.4.1, ser ut til å ha hatt en positiv innvirkning på vår arbeidsprosess og resultater.

6.3 Styrker og svakheter i endelig løsning

Detektorer har vist både styrker og svakheter. Vanndråpedetektoren kunne blant annet blitt forbedret ved å minimere antall falsk-negative resultater. Vi hadde også noen dårlige bilder som ikke ble merket som dårlige. Den begrensede størrelsen på datasettet vårt antyder også en mulig begrensning i resultatenes pålitelighet. Analysen i kapittel 5.4 av prosjektresultatet avslører at detektorene effektivt oppdager feil i bildene som ble undersøkt. Mer spesifikt viste resultatene at 91% av de dårlige bildene ble klassifisert som dårlige. Denne sensitiviteten er av særlig betydning, da det er essensielt for Counting Hero å forstå årsakene bak dårlige bilder dersom de skulle forekomme.

En av problemene vi hadde med “motion blur”-detektoren våres var bilder med skygger nederst på dekker som ble diskutert i kapittel 5.3.1. En mulig løsning kunne ha vært å anvende en kant-detektor som ville sjekket om kanten på dekket hadde blitt jevnet ut av “motion blur”. Dette var noe vi testet, men ikke hadde nok tid til å få relevante resultater ut av. Ellers kunne vi kanskje jevne ut lysnivået på bildet. Disse alternativene ble imidlertid ikke utforsket grunnet tidsbegrensninger.

I figur 5.2 fra kapittel 5.3.1, ser vi et bilde av et dekk med våt overflate og høyt lysnivå. Antallet vanndråper er begrenset, og de få som er tilstede, er av betydelig størrelse. Dette resulterer i vanskeligheter for vår nåværende deteksjonsmetode for å se om dekket er vått. En eventuell løsning kunne ha vært å analysere gjenskinn i dekkene, ettersom dette er en vanlig forekomst på våte dekk. Alternativt kunne anvendelse av en KI-modell for deteksjon av vanndråper ha vært utforsket. Dette er for å oppnå bedre resultat i deteksjonen av ulike dråpestørrelser. Disse alternativene ble ikke utforsket grunnet tidsbegrensninger.

Counting Hero plasserer bilder av bakhjulet og forhjulet til et kjøretøy i de samme navngitte mappen. Disse mappene bruker vi til å definere objektene våre. En svakhet i løsningen vår er at når vi tester kvaliteten av bilder, tar vi det første godkjent bildet vi finner i mappen. Dette gjør vi fordi hvis det er problem med et bilde, går vi ut i fra at det er en god sjanse for at det er problemer med de andre i samme mappe også. Dette reduserer prosessorkraften som kreves for å kjøre systemet, men kan forårsake at noen dårlige bilder ikke blir markert. Dette var en beslutning tatt tidlig i prosjektet, men den var ikke nødvendigvis den beste beslutningen. Men med tiden vi hadde tilgjengelig anså vi ikke dette som en prioritering. Resultatene i kapittel 5.4 er ikke basert på denne måten å gå gjennom bilder på. Alle bildene som vi hadde tilgjengelig, er brukt for å lage statistikken om hvor nøyaktige detektorene våre er i kapittel 5.4.

I intervjuet sammen med oppdragsgiver, under seksjonen "Nettside", ga han uttrykk for at nettsiden er enkel og viser den informasjonen den trenger. Oppdragsgiver påpeker også at nettsiden kunne vært mer estetisk, noe vi er enige i, og ser på som en svakhet med løsningen. Som

nevnt i kapittel 6.1.3, ved mangel på "motion blur"-korrigering, er nettsiden ikke fullstendig, noe som også utgjør en svakhet i den endelige løsningen.

6.4 Forbedringer og andre valg

Dersom vi hadde gjort dette på nytt kunne vi ha forbedret prosjektet ved å formulere en klarere problemstilling fra begynnelsen, slik at vi kunne ha fokusert enda mer på detektorene. Dette ville ha frigjort mer tid i starten av prosjektet, som vi kunne ha brukt til ytterligere ressursinnsamling og generelt kommet raskere i gang. Vi undervurderte omfanget av å definere problemstillinger knyttet til utviklingen av en slik detektor.

Det kom frem i avgrensningene i kapittel 2.2 at vi utviklet kode for videosegmentering. Denne koden endte opp med å ikke være relevant for den endelige løsningen. Vi skulle ønsket å få tilgang til de endelige bildene som løsningen bruker tidligere i prosjektperioden. Hvis vi skulle forbedret noe, hadde vi ønsket å presisere tidligere overfor oppdragsgiver at vi hadde behov for dette. Dette kunne gitt oss bedre tid til å forbedre deteksjonen på bildene av dekk.

Vi burde hatt en grundigere dialog med oppdragsgiver angående funksjonaliteten til deres system og omfanget av oppgaven i starten av prosjektet. Vi opplevde en viss mangel på forståelse for oppgaven og de nye konseptene innledningsvis. Dette påvirket effektivitet negativt, og er et område vi skulle ønske vi hadde adressert tidligere. Gruppen skulle ha oppfordret til en omfattende samtale med oppdragsgiver, som kunne ha bidratt til å oppklare disse områdene.

I intervjuet under seksjonen "Generelt" forklarer oppdragsgiver at han ønsket å se eksempler og resultater fra eksisterende systemer for detektorene. Vi baserte i hovedsak våre deteksjonsmetoder på tilbakemeldinger fra oppdragsgiver og "on-the-fly" -løsninger som vi vurderte som passende for den aktuelle problemstillingen på det tidspunktet. Hvis vi skulle gjøre noe annerledes, ville det vært å trekke enda mer inspirasjon fra tidligere vellykkede detektorer og teste dem. Eksisterende løsninger kan også være bedre egnet til å håndtere store mengder data, da vi primært arbeidet med et begrenset antall bilder. Oppdragsgiver bemerket også i intervjuet at datasettet i eventuelt videre arbeid ville vært mye større. Dette er for å teste deteksjonen på så mange bilder som mulig, samt få så bra deteksjon som mulig.

7 KONKLUSJON OG VIDERE ARBEID

For å konkludere prosjektet vil vi diskutere i hvor stor grad problemstillingen og målene våre er oppfylt. Vi vil ta utgangspunkt i egne erfaringer og resultater fra dybdeintervjuet i evalueringen av prosjektet. Avsluttende ønsker vi også å vise til eventuelt videre arbeid på løsningen.

7.1 Konklusjon

Problemstillingen for prosjektet er som følger:
Hvordan kan vi automatisk detektere og varsle om feil på en klar og pålitelig måte for å redusere tiden det tar å manuelt kvalitetssjekke bildene?

Fra denne problemstillingen har vi utviklet en nettside, samt tre detektorer for "motion blur", dårlig lys og vanndråper. Sluttproduktet vi sitter igjen med dekker det meste av hva vi ønsket å oppnå, men mangler noe testing med større datasett samt implementasjon av "motion blur"-korrigering. Ut fra problemstillingen over definerte vi to mål som vi anså som essensielle for å tilfredsstille problemstillingen vår:

- Utvikle en brukervennlig nettside som oversiktlig viser detekterte feil
- Definere tydelige og pålitelige terskelverdier

Vårt system er en bildekvalitetsdetektor som inneholder 3 detektorer. Analysen i kapittel 5.4 viser at detektorene effektivt oppdager feil i bildene som ble undersøkt. Mer spesifikt viste resultatene at 91% av de dårlige bildene ble klassifisert som dårlige. Dette anser vi som gode resultater. Vi klarer å identifisere de aller fleste av de dårlige bildene i systemet vårt. Ettersom gruppen valgte en problemstilling som kunne blitt forbedret med mer testing og undersøkelse på andre metoder, har vi kommet i mål med system som kan automatisk detektere og varsle om feil i bildekvaliteten .

Etter mange evalueringer kontinuerlig gjennom prosjektet, samt sluttevalueringen med oppdragsgiver kan vi vise til disse positive resultatene:

- Definerte en relevant problemstilling for Counting Hero
- Utviklet en enkel og informativ nettside
- Viste til overbevisende resultater med bruk av innovative og kreative metoder

I sammenheng med problemstillingen vi har jobbet med gjennom hele prosjektet, spurte vi i intervjuet med oppdragsgiver om systemet sin relevans når det kommer til å redusere tiden det tar å kvalitetssjekke bildene. På dette spørsmålet gir oppdragsgiver uttrykk for at detektorene og nettsiden vil kunne bidra til å redusere tiden. Basert på resultatene i evalueringen gitt i intervjuet, velger vi å konkludere med et resultat som oppfyller de målene vi har satt. Gruppen mener likevel at prosjektet har et godt potensial for videreutvikling, noe vi utdyper mer i neste delkapittel.

7.2 Videre arbeid

Dersom prosjektet skal arbeides videre med, ville vi blant annet anbefale å jobbe mer med å gjøre nettsiden estetisk. Dette var som tidligere nevnt noe vi valgte å nedprioritere gitt tiden vi hadde til disposisjon, men anerkjenner at det kan være ønskelig. Nettsiden vi har utviklet kan virke ensfarget og statisk, noe som ikke er i tråd med slik Counting Hero sin hjemmeside [49] er i dag som er mer fargerik og dynamisk.

Det viktigste for videre arbeid med detektorene er å teste dem på større datasett. Testing på store datasett vil hjelpe til med å finjustere terskelverdiene til detektorene, noe som kan gjøre dem mer nøyaktige. Her kunne det også vært nyttig og utforsket forslagene for bedring av "motion blur" og vann-detektorene som diskutert i kapittel 6.3.

Videre ville vi utforsket bruk av eksisterende eksempler fra andre systemer implementert i vårt eget slik vi fikk tilbakemelding på i dybdeintervjuet. Dette ville gitt oss muligheten til å prøve gode og testede metoder som har gitt overbevisende resultater tidligere. Gruppen hadde også hatt god dokumentasjon og begrunnelser for metodene vi hadde prøvd og muligens implementert.

Avslutningsvis ville vi ha videreutviklet detektoren for vanndråper. Vi opplevde som tidligere nevnt noen problemer med den i områder med mye skygge. Her kunne detektoren i noen tilfeller anta skygger som dråper, og gi falske resultater. Dette ville vært en ganske relevant videreutvikling å jobbe med ettersom det vil kunne hjelpe med å redusere de falske-positive resultatene enda mer.

8 REFERANSER

[1] McKinsey & Company, (2023). The state of AI in 2023: Generative AI's breakout year [Online]. Tilgjengelig: https://www.mckinsey.com/capabilities/quantumblack/our-insights/the-state-of-ai-in-2023-generative-ais-breakout-year

[2] Statens vegvesen. IT - kunstig intelligens og datakvalitet [Online]. Tilgjengelig: https://www.vegvesen.no/om-oss/jobb/medarbeiderne-vare-forteller/it--kunstig-intelligens-og-datakvalitet/

[3] Wayback Machine, (27. sep. 2023). Counting Hero - About us [Online]. Tilgjengelig: https://web.archive.org/web/20230927120041/https://www.countinghero.com/about-1

[4] Wayback Machine, (10. des. 2023). Counting Hero - Home [Online]. Tilgjengelig: https://web.archive.org/web/20231210104948/https://countinghero.com/

[5] GitHub, (2024, Sept.). Image and Video Deblurring [Online]. Tilgjengelig: https://github.com/subeeshvasu/Awesome-Deblurring

[6] PyTorch, (2023). PyTorch Documentation [Online]. Tilgjengelig: https://pytorch.org/docs/stable/index.html

[7] P. Ongsulee. (2018, Jan. 18). Artificial intelligence, machine learning and deep learning. Presentert på *15th International Conference on ICT and Knowledge Engineering (ICT&KE) 2017*, Bangkok, Thailand. [Online]. Tilgjengelig: https://ieeexplore-ieee-org.galanga.hvl.no/document/8259629

[8] Coursera, (2024, Feb. 09). Deep Learning vs. Machine Learning: A Beginner's Guide [Online]. Tilgjengelig: https://www.coursera.org/articles/ai-vs-deep-learning-vs-machine-learning-beginners-guide

[9] L. Chen, F. Fang, T. Wang, G. Zhang. (2020, Jan. 09). Blind Image Deblurring With Local Maximum Gradient Prior. Presentert på *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) 2019,* Long Beach, CA, USA. [Online]. Tilgjengelig: https://ieeexplore-ieee-org.galanga.hvl.no/document/8953398

[10] V. Powell. Image Kernel [Online]. Tilgjengelig: https://setosa.io/ev/image-kernels/

[11] GIMP. Konvolusjonsmatrise [Online]. Tilgjengelig: https://docs.gimp.org/2.4/no/plug-in-convmatrix.html

[12] Y. Quan, P. Lin, Y. Xu, Y. Nan, H. Ji, (2021, Apr. 14). Nonblind Image Deblurring via Deep Learning in Complex Field. Presentert på *IEEE Transactions on Neural Networks and Learning Systems*. [Online]. Tilgjengelig: https://ieeexplore-ieee-org.galanga.hvl.no/document/9404870

[13] K. He, X. Zhang, S. Ren, J. Sun, (2016, Des. 2016). Deep Residual Learning for Image Recognition. Presentert på *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2016*, Las Vegas, NV, USA. [Online]. Tilgjengelig: https://ieeexplore-ieee-org.galanga.hvl.no/document/7780459

[14] O. Kupyn, V. Budzan, M. Mykhailych, D. Mishkin, J. Matas, (2018, Jun. 18-23). DeblurGAN: Blind Motion Deblurring Using Conditional Adversarial Networks. Presentert på *IEEE/CVF Conference on Computer Vision and Pattern Recognition 2018*, Salt Lake City, UT, USA. [Online]. Tilgjengelig: https://ieeexplore-ieee-org.galanga.hvl.no/document/8578952

[15] C. Paul, M. Godambe, (2021, Nov.). Image Downsampling & Upsampling [Online]. Tilgjengelig:

https://www.researchgate.net/publication/359772964_Image_Downsampling_Upsampling#full-text
[16] D. Li, X. Shi, Y. Zhang, K. C. Cheung, S. See, X. Wang, H. Qin, H. Li, (2023, Aug. 22). A Simple Baseline for Video Restoration with Grouped Spatial-Temporal Shift. Presentert på *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) 2023*, Vancouver, BC, Canada. [Online]. Tilgjengelig:
https://ieeexplore.ieee.org/document/10203329
[17] JetBrains, (2024). PyCharm [Online]. Tilgjengelig:
https://www.jetbrains.com/pycharm/
[18] Microsoft, (2024). Visual Studio Code [Online]. Tilgjengelig:
https://code.visualstudio.com/
[19] Medium, (2023, Jun. 16). Comparing PyCharm and VSCode for Python Development [Online]. Tilgjengelig:
https://medium.com/@teamcode20233/comparing-pycharm-and-vscode-for-python-development-6de85180d994
[20] A. Jalli (2023, Okt. 19). 11 Best Python IDEs and Code Editors Available [Online]. Tilgjengelig:
https://builtin.com/software-engineering-perspectives/python-ide
[21] OpenCV, (2024). OpenCV [Online]. Tilgjengelig:
https://opencv.org/
[22] Flask, (2010). Flask Documentation [Online]. Tilgjengelig:
https://flask.palletsprojects.com/en/3.0.x/
[23] Discord, (2024). Discord [Online]. Tilgjengelig:
https://discord.com/
[24] Meta, (2024). Messenger [Online]. Tilgjengelig:
https://www.messenger.com/
[25] GitHub, (2024). GitHub [Online]. Tilgjengelig:
https://github.com/
[26] GitHub, (2024). GitHub Desktop [Online]. Tilgjengelig:
https://desktop.github.com/
[27] Google, (2024). Google Disk [Online]. Tilgjengelig:
https://drive.google.com/
[28] Google, (2024). Google Dokumenter [Online]. Tilgjengelig:
https://docs.google.com/
[29] ClickUp, (2024). ClickUp [Online]. Tilgjengelig:
https://clickup.com
[30] Lucidchart, (2024). Lucidchart [Online]. Tilgjengelig:
https://www.lucidchart.com/
[31] Milient. Scrum vs Kanban: Velg riktig metode for prosjektet ditt [Online]. Tilgjengelig:
https://www.milientsoftware.com/no/blogg/scrum-vs-kanban (Hentet 25. feb. 2024)
[32] Pinterest. Agile metode [Online]. Tilgjengelig:
https://no.pinterest.com/pin/3096293484577030/
[33] A. Rolstadås, (2021, Jun. 29). Gantt-diagram [Online]. Tilgjengelig:
https://snl.no/Gantt-diagram

[34] P.S. Davidsen og K. Vollan, “Risikohåndtering av IT-prosjekt”, Oslo, april 1998. [Online]. Tilgjengelig: https://dfo.no/sites/default/files/r98-7-risikohandtering-av-it-prosjekter.pdf

[35] P. Simek (2019, Nov. 11). Unit-testing [Online]. Tilgjengelig: https://developerexperience.io/articles/unit-testing

[36] S. Ratnawati, L. Widianingsih, N. Anggraini, I. Marzuki Shofi, N. Hakiem, F. Eka M Agustin, (2020, Dec. 04). Evaluation Of Digital Library's Usability Using the System Usability Scale Method of (A Case Study). Presentert på *8th International Conference on Cyber and IT Service Management (CITSM) 2020*, Pangkal, Indonesia. Tilgjengelig: https://ieeexplore-ieee-org.galanga.hvl.no/document/9268801

[37] M. Sbert, C. Ancuti, C. O. Ancuti, J. Poch, S. Chen and M. Vila, (2021, Feb. 21). Histogram Ordering. Presentert i *IEEE Access*. [Online]. Tilgjengelig: https://ieeexplore.ieee.org/document/9352004

[38] Python Docs, (2024, Apr. 14). Pickle - Python object serialization [Online]. Tilgjengelig: https://docs.python.org/3/library/pickle.html

[39] Spring, (2024). Spring Boot [Online]. Tilgjengelig: https://spring.io/projects/spring-boot

[40] Canon, (2024). Slik mestrer du lukkerhastigheten [Online]. Tilgjengelig: https://www.canon.no/get-inspired/tips-and-techniques/how-to-use-shutter-speed/

[41] NumPy, (2022). What is NumPy? [Online]. Tilgjengelig: https://numpy.org/doc/stable/user/whatisnumpy.html

[42] OpenCV (2023). Operations on arrays [Online]. Tilgjengelig: https://docs.opencv.org/4.x/d2/de8/group__core__array.html#ga191389f8a0e58180bb13a727782cd461

[43] N. Sanchar, Sandeide bussterminal, Fyllingsdalen, Bergen, Norge. *Car passing by in a Highway - Royalty Free Stock Video / (Copyright Free) Download.* (Jun. 04, 2018). Hentet Feb. 05, 2024. [Online video]. Tilgjengelig: https://www.youtube.com/watch?v=K6xsEng2PhU&ab_channel=NauloSanchar

[44] School of Physics Sydney. Rolling: The physics of wheels [Online]. Tilgjengelig: https://www.animations.physics.unsw.edu.au/jw/rolling.htm

[45] University of Rochester. Rolling, torque and angular momentum [Online]. Tilgjengelig: http://teacher.pas.rochester.edu/phy121/lecturenotes/Chapter12/Chapter12.html

[46] PyTorch, (2023). torch.var [Online]. Tilgjengelig: https://pytorch.org/docs/stable/generated/torch.var.html#torch-var

[47] Scandinavian Photo, (2022, Okt. 12). Lukkertid - hvordan fungerer kameraets lukker? [Online]. Tilgjengelig: https://www.scandinavianphoto.no/kunnskap/lukkertid

[48] Prabhat, Nishant, D.K. Vishwakarma, (2020, Jun. 19). Comparative Analysis of Deep Convolutional Generative Adversarial Network and Conditional Generative Adversarial Network using Hand Written Digits. Presentert på *4th International Conference on Intelligent Computing and Control Systems (ICICCS) 2020*, Madurai, India. [Online]. Tilgjengelig: https://ieeexplore-ieee-org.galanga.hvl.no/document/9121178

[49] F. Preetham, (2024, Apr. 25). The Gaussian Distribution: A Mathematical Perspective [Online]. Tilgjengelig: https://medium.com/mathematical-musings/the-gaussian-distribution-a-mathematical-perspective-ab89fcfd29b7

[50] S. Nah, T.H. Kim, K.M Lee, (2017, Nov. 09). Deep Multi-scale Convolutional Neural Network for Dynamic Scene Deblurring. Presentert på *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2017*, Honolulu, HI, USA. [Online]. Tilgjengelig: https://ieeexplore-ieee-org.galanga.hvl.no/document/8099518

[51] Y. Peng, (2023, Jun. 14). Website Visual Communication Design Method Based on User Experience. Presentert på *6th International Conference on Wireless Communications and Applications (ICWCAPP) 2022*, Haikou, China. [Online]. Tilgjengelig https://ieeexplore-ieee-org.galanga.hvl.no/document/10148530

[52] Counting Hero. Revolutionizing Road Safety with Computer Vision [Online]. Tilgjengelig: https://www.countinghero.com/ (Hentet 21. feb. 2024)

[53] A. Kulkarni, D. Chong, F.A Batarseh, (2020). *Data democracy* [Online]. Tilgjengelig: https://www.sciencedirect.com/science/article/abs/pii/B9780128183663000058

[54] Pytest, (2024). pytest: helps you write better programs - pytest documentation [Online]. Tilgjengelig: https://docs.pytest.org/en/8.1.x/

[55] Medium, (2023, Mai 23). Mastering Unit Tests in Python with pytest: A Comprehensive Guide [Online]. Tilgjengelig: https://medium.com/@adocquin/mastering-unit-tests-in-python-with-pytest-a-comprehensive-guide-896c8c894304

[56] C. Boyce, P. Neale, (2006, Mai). Conduction in-depth interviews: A Guide for Designing and Conducting In-depth Interviews for Evaluation Input [Online]. Tilgjengelig: https://nyhealthfoundation.org/wp-content/uploads/2019/02/m_e_tool_series_indepth_interviews-1.pdf

[57] P. Singh, N. Singh, K.K. Singh, A. Singh, (2021). Machine Learning and the Internet of Medical Things in Healthcare [Online]. Tilgjengelig: https://www.sciencedirect.com/science/article/abs/pii/B9780128212295000033

9 VEDLEGG

Vedlegg 1: Prosjekthåndbok

Vedlegg 2: Visjonsdokument

Vedlegg 3: Kravdokument

Vedlegg 4: Systemdokumentasjon

Vedlegg 5: Modell for gradering av risikoprodukt

Sannsynlighet	Svært Høy (5)	5	10	15	20	25
	Høy (4)	4	8	12	16	20
	Middels (3)	3	6	9	12	15
	Lav (2)	2	4	6	8	10
	Svært Lav (1)	1	2	3	4	5
		Svært Lav (1)	Lav (2)	Middels (3)	Høy (4)	Svært Høy (5)
	Konsekvens					