Universitetet i Stavanger

DET TEKNISK-NATURVITENSKAPELIGE FAKULTET

MASTEROPPGAVE

Studieprogram/spesialisering: Automatisering og signalbehandling	Vårsemesteret, 2016 Åpen
Forfatter: Erik Sudland	…… (signatur forfatter)
Fagansvarlig: Erlend Tøssebro Veileder(e): Ivar Austvoll	
Tittel på masteroppgaven: Gjenkjenning av kjøretøy ved inn- og ut- kjøring av tunneler Engelsk tittel: Detection of cars entering and exiting a tunnel	
Studiepoeng: 30	
Emneord: Bildebehandling, Bakgrunnssubtraksjon, HOG, Trace Transform	Sidetall: 122 + vedlegg/annet: bilde, video, programkode (dropbox) Stavanger, 15/07/2016 dato/år

Sammendrag

Flere granskningsrapporter av tunnelbranner de siste årene [78, 79] har konkludert med at det savnes en teknologi som kan gi sanntids posisjonsoversikt over antall kjøretøy og tilhørende personer i tunnelen. En slik oversikt kan hjelpe nødetatene med å gjøre en effektiv vurdering av skadeomfanget og tilpasse redningsarbeidet mer målrettet deretter. I denne oppgaven drøfter vi muligheten for å anvende maskinsyn for et slikt formål. Det er blitt utarbeidet et forslag til et system som registrerer ulike egenskaper til kjøretøy ved tunnelinngang. Egenskapene brukes til å gjenkjenne kjøretøyet ved utgang, samtidig som det definerer kjøretøyet som lett eller tung klasse. Egenskapene som detekteres informerer om kjøretøyets farge, form og struktur. Sistnevnte egenskap brukes til å beskrive kjøretøyfronten. Denne egenskapen er vektlagt gjennom oppgaven. Kjøretøyets front representerer sterke karakteristiske trekk, og egnes derfor til å skille ulike kjøretøy fra hverandre.

Metodene HOG og trace transform benyttes til å hente de ulike strukturelle egenskapene. Den ene beregningen detekterer kantstrukturen i bildet, og den andre detekterer signaturen. Disse testes opp mot lysendring, okklusjon og geometriske transformasjoner som oppstår i filmopptaket. Det er blitt gjennomført filmopptak ved hjelp av to kamera som filmet kjøretøy på veien i et topp-ned perspektiv. Kameraene simulerer en plassering ved inn- og utgang på tunnel. Resultatet av testene viser at HOG-deskriptor er egnet til å danne en beskrivelse av kjøretøyfronten, da den viser seg å være både rask og lite sensitiv til støy i form av lysendringer. Antall kjøretøy som HOG-metoden matcher korrekt, er høyt nok til å tilfredsstille oppgavens formål. Metodens ulempe er at fronter med lik kantstruktur, men ulik farge ikke blir separert ved HOG-metoden. En mer omfattende frontbeskrivelse kan derimot oppnås ved å kombinere HOG sammen med en deskriptor som henter informasjon om intensiteten i bildet. I oppgaven ble trace transform brukt til dette, og et bedre resultat ble oppnådd. Ulempen er at metoden krever mye prosesseringskraft.

Utviklingen i verden er preget av en økende grad av automatisering. Teknologien kan bidra til tryggere omgivelser for fremtiden. Derfor er det viktig å starte utviklingen av et system som har potensiale til å redde liv i fremtiden. Denne oppgaven kan sees på som den første byggeklossen i et slikt system. God lesing!

Forord

Denne masteroppgaven ble skrevet som en avslutning til mastergradsstudiet informasjonsteknologi, signalbehandling og automatisering ved Universitetet i Stavanger våren 2016.

Flere bidragsytere skal takkes i forbindelse med oppgaven. Jeg vil utnevne en takk til min fagansvarlig Erlend Tøssebro som har kommet med verdifulle innspill og tilbakemeldinger. Når jeg trengte veiledning og råd underveis, bistod min veileder Ivar Austvoll med en fabelaktig innsats - tusen takk Ivar.

Jeg vil også takke Bjørn Arild Fossåen fra Statens vegvesen for anskaffelse av kameraet som var nødvendig for å gjennomføre oppgaven. Ove Njå, professor i Risk and Emergency Management, har vist stor entusiasme under oppgaveskrivingen og bidratt med ideer og forslag. Takk for det. Ståle Freyer har vært svært behjelpelig med utleie av utstyr som var essensielt for å kunne utføre eksperimentene i oppgaven. Tusen takk.

Til slutt vil jeg rette en stor takk til venner og familie for oppmuntring og støtte. Dere har motivert og inspirert meg gjennom dette semesteret.

Erik Sudland
15 Juli 2016

Innhold

Figurer

Tabeller

Forkortelser

Blob Binary large object

FAST Feature from Accelerated Segment Test

GMM Gaussian Mixed Models

HSV Hue, Saturation og Value

NCC Normalised Cross Correlation

VTS Vegtrafikksentralen

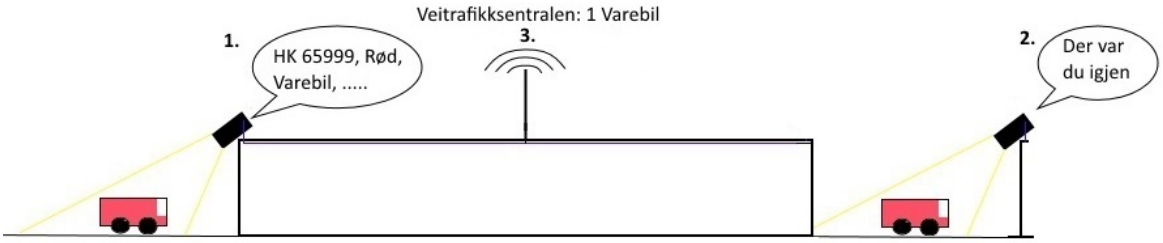
WDR Wide Dynamic Range

YCbCr Luminans, blå og rød krominanskomponenter

Kapittel 1

Introduksjon

Denne oppgaven handler om hvordan man kan registrere kjøretøy som er inne i en tunnel slik at nødetatene har en oversikt over dette ved en eventuell ulykke. I dag detekterer Statens vegvesen kjøretøy ved å bruke induktive sløyfer, laser, radarer og trykkfølsomme sensorer. Sensorene har fordeler og ulemper og disse er forklart i tilleggskapittelet. En økning i datakraft de siste årene har gjort maskinsyn til en attraktiv løsning for å detektere kjøretøy. Fordelen med denne teknologien i forhold til andre typer sensorer er at den har et større bruksområde. Teknologien kan brukes til å gjenkjenne et stort antall objekter som ulike kjøretøytyper, fotgjengere, og skilt som indikerer farlig last. Muligheten til å detektere kjøretøy i feil retning eller andre hendelser regnet som farlige, gjør teknologien ekstra interessant. Et ønsket systemet der overvåkingskamera er plassert inni/utenfor tunnel og det brukes maskinsyn-algoritme til å detektere kjøretøy, er vist under i Figur 1.1.



Figur 1.1 Visualisering av et deteksjonssystem i tunnel.

Systemet kan deles opp i 3 punkter:

1. Registrere egenskaper til innkjørende kjøretøy.
2. Registrere at kjøretøyet er ute av tunnelen.
3. Info til Vegtrafikksentralen (VTS) om antall og type kjøretøy som befinner seg i tunnelen. Denne delen etterlates som en mulig problemstilling i en senere masteroppgave.

I dette kapittelet presenteres det et forslag til et system som gjenkjenner og teller antall kjøretøy. For å kunne gjenkjenne det samme kjøretøyet ved to forskjellige observasjonspunkter, må ulike egenskaper som beskriver kjøretøyet detekteres. Jeg har bestemt at problemstillingen i denne oppgaven blir å finne egenskaper som kan knyttes til *formen*, *fargen* og *strukturen* ved kjøretøyet.

Kjøretøyfronten er et område som har sterke særegne trekk. Innhenting av egenskaper som forteller noe om strukturen ved dette området kan brukes til å gjenkjenne samme kjøretøy ved en senere anledning. En videre problemstilling blir derfor å finne metoder som henter ut forskjellige typer strukturegenskaper og som er robuste mot ulike bildetransformasjoner. Til dette har jeg anvendt to metoder som testes mot hverandre. Målet er å velge metoden som gjenkjenner flest kjøretøyfronter observert ved forskjellige punkter på veien korrekt. Egenskapen som omhandler struktur er derfor spesielt lagt vekt på i denne oppgaven.

I dette kapittelet vil jeg først introdusere tunnelsikkerhet generelt og deretter de aspektene ved tunnelsikkerhet som blir behandlet i oppgaven.

1.1 Tunnelsikkerhet

Norge er et av landene med flest tunneler i verden. Med ca. 1000 vegtunneler og en samlet lengde på over 800 km. Årlig bygges det 20-30 km nye tunneler her i landet [82]. Tunnelene utgjør en nøkkelrolle når det gjelder å holde områder tilgjengelige for folk og industrier som er avhengig av effektiv transport av varer og arbeidskraft. Å ha et sunt veinett er viktig for økonomien i tettsteder og regioner.

Tunnelsikkerhet har fått et økende fokus de siste årene. Brannen i Oslofjordtunnelen i 2011 hvor 32 mennesker ble transportert til sykehus for behandling [77], samt nye retningslinjer satt av EU med fokus på stigningsgrad, rømningsveier og havarilommer [50] har bidratt til dette. Ny teknologi innen TV-overvåking, automatisk hendelsesdetektering og automatisk branndeteksjon, vil kunne spille en nøkkelrolle for å møte sikkerhetskravene som vil stilles i fremtiden for å få et tryggere veinettverk.

Ulykkesfrekvensen er lavere i tunneler enn på åpen vei. De fleste dødsulykkene skjer ved utforkjøringer eller ved at et lett kjøretøy kommer over i motsatt kjøreretning [17]. Faremomentet for tunneler ligger ved at når en ulykke først skjer, kan konsekvensene være store. Det er særlig i de tilfeller der et kjøretøy begynner å brenne og skaper svært farlig røykutvikling. Antall tunnelbranner i Norge har ligget i snitt på rundt 21 i året [56]. Kartlegging av branner i tunnel foretatt av Transportøkonomisk institutt i 2014 [56], viser at de fleste branner oppstår i undersjøiske tunneler med høy stigninggrad. En viktig årsak til kjøretøybranner i tunnel, er tunge kjøretøy med tekniske problemer, illustrert i Figur 1.2.

Figur 1.2 Brannen i Oslofjordtunnelen 23. juni 2011 oppstod i motoren til det tunge kjøretøyet.
FOTO: Statens Vegvesen

Skulle en tunnelulykke oppstå, vil det være viktig for nødetatene å få raskest mulig oversikt over antall og typer kjøretøy som befinner seg i tunnelen. En slik oversikt vil kunne brukes til vurdering av skadeomfang og behov for ressurser. Dersom man har informasjon om at ingen kjøretøy befinner seg i tunnelen ved en brann, slipper brannvesenet å sende personell inn i farlige omgivelser for å lete etter overlevende. Ulykkesgranskningen gjort av Statens havarikommisjon for transport (SHT) av brann i vogntog i Gudvangatunnelen i 2013 [78], konkluderte med mangel av videoovervåking i tunnelen. SHTs analyse beskrev situasjonen slik:

" *Tunnelen var ikke utstyrt med noen form for overvåking eller telling av kjøretøy som ga kontinuerlig informasjon om hvor mange kjøretøy som befant seg i tunnelen. Vegtrafikksentralen (VTS) og brannvesenet hadde dermed ikke oversikt over hvor mange kjøretøy som befant seg på den siden av brannen som røyken ble styrt* ".

Fra anbefalingen til SHT, vil det være nyttig for vegtrafikksentralen å vite klassen til kjøretøyet. For eksempel vil det å vite om det er en lastebil eller personbil i tunnelen når en eventuell ulykke inntreffer, kunne si mye om antallet personer som befinner seg i tunnelen og mulig skadeomfang. Informasjon om kjøretøyets fart vil kunne brukes til å beregne hvor langt inne i tunnelen kjøretøyet befinner seg. Noe som vil bidra til å redusere søketiden etter overlevende. Store vogntog kan frakte farlig og eksplosiv last. Ved en eventuell brann vil det kunne være katastrofalt å sende brannmannskap inn i tunnelen. Derfor kan innhenting av informasjon fra oransje skilt som indikerer transport av farlig gods, bidra til økt sikkerhet for personell og i ytterste tilfelle redde liv.

1.1.1 Bruk av maskinsyn for å bedre tunnelsikkerhet

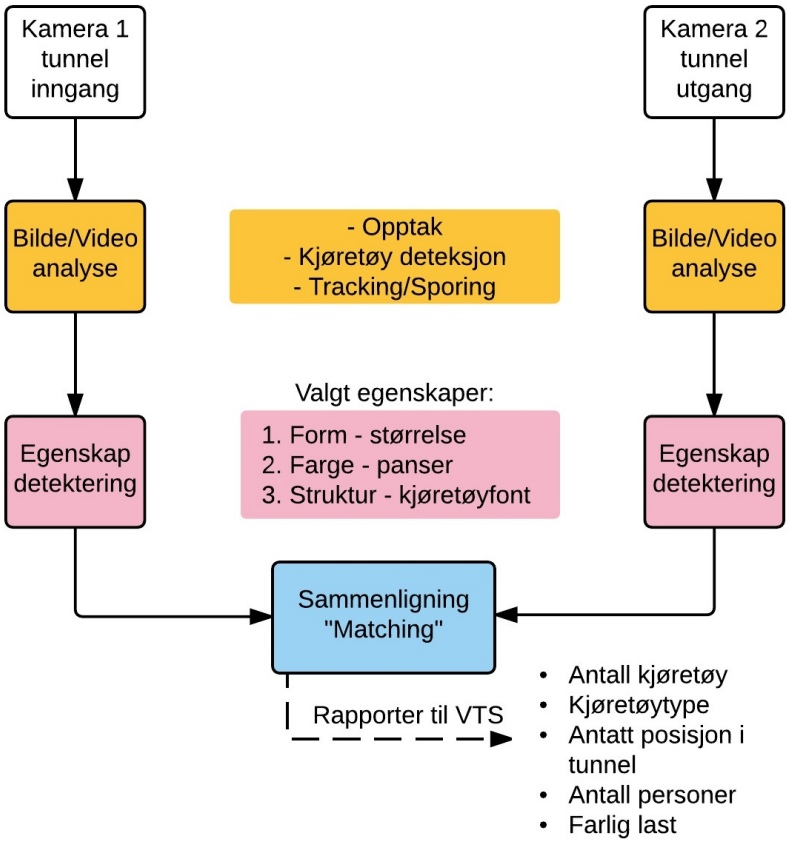
Maskinsyn er pga. senere års utvikling blitt en attraktiv teknologi for testing i trafikk. Teknologien vil kunne fungere som en sensor for flere ulike typer hendelsesdeteksjoner. Implementering av nye funksjoner utføres ved oppdatering av programvare og dermed slippes det å hindre trafikken. Dette er en stor fordel i tunneler hvor en stenging kan medføre kostnader.

Østerrike er et land som i likhet med Norge, har hatt flere store ulykker i tunnel. Som reaksjon har det blitt investert betydelige summer de siste årene for å forbedre sikkerheten. Et pilotprosjekt som involverte flere forskere, ble utført i 2005 av Østerrikes vegvesen (ASFINAG). En tunnel ble sperret av og maskinsyn algoritmer ble testet på overvåkningskameraer [68]. Målet var å detektere objekter som uautoriserte personer, mistet last og kjøretøytype sammen med ulike unormale trafikksituasjoner som trafikkork, kjøretøy i feil kjøreretning og deteksjon av røyk og brann. Prosjektet gikk ved navnet VITUS (Video Image analysis for Tunnel Safety) og konkluderte med at teknologien ikke var fullstendig moden til å gi pålitelige resultater.

Med drastisk forbedring av teknologi og datakraft de siste årene, startet ASFINAG i samarbeid med private selskap i slutten av 2014 et nytt landsomfattende sikkerhetprosjekt for igjen å teste teknologien. Her blir nye, avanserte 3D-maskinsynalgoritmer samt termisk kamera brukt for å oppnå bedre resultater. Prosjektet går under navnet Robust Sensor Systems for Advanced Traffic Applications (ROSSATA). For detektering av objekter i tunnelen, blir RGBD-karakterstikk (Rød, Grønn, Blå, Dybde informasjon for hver piksel [47]), passiv stereo og 3D-bevegelsesflyt for 3D-scene analyse brukt. Karakterisering av geometrien i form av dybde og intensitet brukes så til å klassifisere kjøretøyene. Videre blir trafikkdata analysert. Teknikker som gjør det mulig å ta individuell kontroll over trafikken, og dermed regulere den for å oppnå bedre flyt testes ut. Prosjektet er pågående.

1.2 Forslag til system

Her presenteres det et konsept for et system som kjenner igjen motorkjøretøy, samt valg av egenskaper og algoritmer. I Figur 1.3 vises systemet oppgaven realiserer.



Figur 1.3 Deteksjonssytemet som er blitt foreslått i oppgaven. Oppgaven har lagt vekt på områdene merket med oransje, blå og fiolett.

1.2.1 Deteksjonssystemet

Den mest nærliggende egenskapen for å kjenne igjen et kjøretøy er gjennom skiltdeteksjon. Grunnet skitne skilt, lav oppløsning og lysrefleksjoner er derimot skilt ikke alltid lesbar. Systemet oppgaven presenterer, detekterer derfor flere kjøretøyegenskaper i tilfeller der skilt er uleselig. Det settes som krav at algoritmen til egenskapsdetektering er rask og krever lite prosessorkraft. Dette muliggjør et smart-kamera system der prosessering kan gjøres lokalt og man slipper kostbare serverrom for å utføre deteksjonen.

Grunnet stor variasjon av type tunneler og begrensende muligheter til innhenting av data, tar ikke systemet stilling til plasseringen ute eller inne. På en annen side blir ulike forutsetninger vurdert som må til for at systemet skal virke. Disse er bedre beskrevet i resultat delen. Kamera kan enten velges å plasseres i høyden med filming i topp-ned perspektiv eller på bakkenivå som filmer trafikken langs bakkeplanet. Skifte i posituren til motivet (kjøretøyobjektet) som skapes ved disse ulike observasjonspunktene, er stor nok til at samme algoritme ikke vil fungere for begge kameraposisjonene.

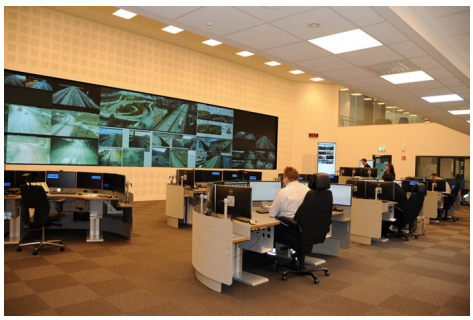
Med begrunnelse i at det allerede finnes utallige kamera plassert inne og utenfor tunnelåpningen i en høyde som gir et synsfelt med bredt dekningsområde, har valget falt på å fokusere på en kameraposisjon som filmer ovenfra og ned. Her vil Vegtrafikksentralen få en fordel ved at allerede eksisterende kameraer kan oppgraderes med smartfunksjon. I tillegg vil en kameralinse i høyden bli mindre utsatt for støv, salt og trykkbølger forårsaket av tung trafikk. En kameralinse fritt for smuss og forurensning, er en forutsetning for en velfungerende maskinsynalgoritme. Resultat av eksisterende kameraovervåkning på veier, samt videopanelet til VTS er vist i Figur 1.4.

En annen utfordring for et slikt system er at en krever vanligvis forskjellig algoritme på ulike tider av døgnet. Lysforholdene på kveldstid og i tunnel gjør at lite lys blir tatt opp av kameraet. Dette reduserer bildekvaliteten betraktelig. Ved labre lysforhold blir også metalloverflaten mer utsatt for lysrefleksjoner, noe som medfører at gjenkjenningen blir vanskeligere. I denne oppgaven har vi valgt å fokusere på en algoritme som fungerer i forhold med tilstrekkelig lys. Her åpnes det for at en i et evt. videre arbeid ser på algoritmer som fungerer i mørke.

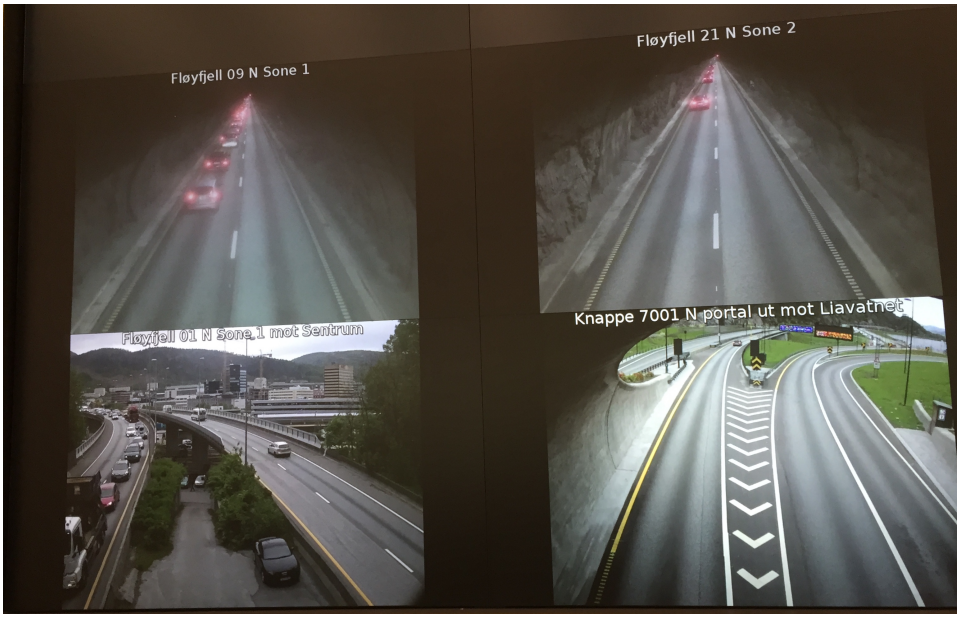
Selv om oppgaven ser for seg et at skilt leses når kjøretøy passerer, anbefales det at skiltlesing foregår med et eget kamera der fokuset er rettet mot skiltområde for å oppnå en høy bildekvalitet. For et HD kamera som danner et overblikk over veien, kan oppløsningen i flere tilfeller bli for lav for en tilfredsstillende avlesning av skilt. Av den overnevnte grunn, ferdig tilgjengelige systemer og at mye tidligere arbeid har blitt gjort m.h.p. automatisk skiltdeteksjon [41] [14], vil det ikke bli fokusert på skiltdeteksjon i denne oppgaven. Oppgavens hovedfokus er egenskapsdetektering. Til dette trenger vi bare å implementere en algoritme for å lokalisere skilt, ikke lese dem. Skilt brukes som referansepunkt til innhenting av egenskaper fra bilfronten og er bedre beskrevet i neste delkapittel.

Systemet som er presentert i oppgaven, detekterer og sporer kjøretøy til de ankommer en satt avstand fra kameraet, bestemt av et forhåndsdefinert deteksjonsområde. Dette området vises i Figur 1.5. Her hentes det ut et bilde av kjøretøyet hvor videre klassifisering av kjøretøytype gjøres, deteksjon av skilt og andre egenskaper uføres. Videre sendes egenskapsvektor

(a) Videoovervåking ovenfra-ned perspektiv.
FOTO: Masar Al-Reem networks



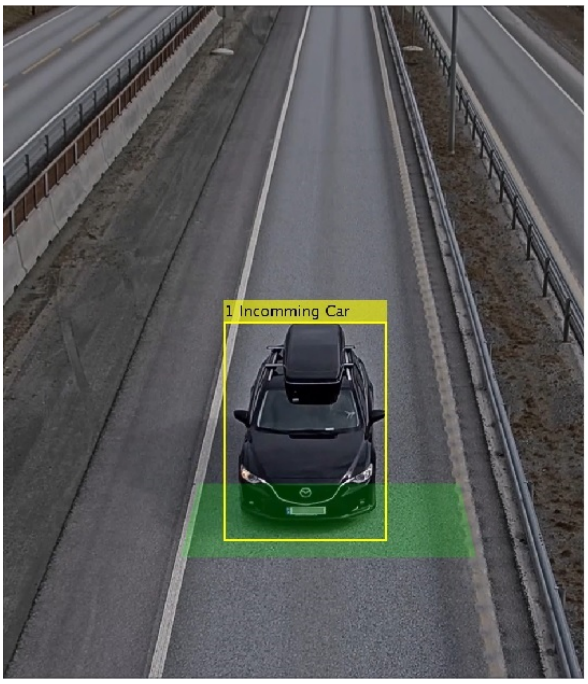
(b) Videoveggen ved VTS.
FOTO: Statens vegvesen



(c) Forstørret bilde av videoveggen til VTS. Ovenfra-ned perspektiv er mest vanlig ved overvåkning i tunnel og friluft.
FOTO: Statens vegvesen

Figur 1.4 Bildene illustrerer videoovervåkningen på veier.

til endekameraet som bruker informasjonen til å kjenne igjen samme kjøretøy ved ankomst. Resultatet av sammenligningen brukes til å holde vegtrafikksentralen oppdatere om antall og type kjøretøy som befinner seg i en tunnel. I neste avsnitt presenteres egenskapene som vi har valgt å detektere ved kjøretøyene.



Figur 1.5 Dette bildet viser deteksjonsområdet i grønt som kjøretøyet må krysse før egenskaper detekteres. En gul ramme er markert rundt kjøretøyet som spores gjennom billedsekvensen.

1.2.2 Valg av egenskaper

Det kan være svært utfordrende å finne stabile karakteristiske trekk som kan brukes til å beskrive et kjøretøy. Om et kjøretøy skifter kjørelys eller skrur på blinklys, fører det til endring av bilens utseende. Lyssignatur fra lykt vil dermed bli en upålitelig kilde for å beskrive bilen. For at de karakteristiske egenskapene skal være gjenkjennelige, må de forbli uforandret ved tidsskift og invariant mot bildetransformasjoner som skalaendring og rotasjon.

Vi er interessert i å detektere bildeegenskaper som tilfredsstiller kravene nevnt ovenfor og er særegne nok til å kunne skille kjøretøy fra andre. Siden denne oppgaven kan sees på som en ”prototype” til et system, har det blitt sett på relativt enkle egenskaper. Disse egenskapene kan deles inn i 3 kategorier; **form**, **farge** og **struktur**. De to første egenskapene skal brukes til å skille vekk usannsynlige kjøretøykandidater og sistnevnte tar for seg ”grovarbeidet” ved en sammenligning og skal være oppgavens hovedfokus.

1. Form

Størrelse er den mest hensiktsmessige måten å skille tyngre og lette kjøretøygrupper fra hverandre. Å klassifisere et kjøretøy inn i en kjøretøygruppe vil kunne redusere antall mulige kandidater som må sammenlignes, og forbedre dermed treffprosenten. Klassen til kjøretøyet

blir skilt mellom *lett* eller *tung* ut ifra størrelsen av bilderammen, (markeringsrammen) rundt det detekterte kjøretøyet vist i gult i Figur 1.5.

2. Farge
Farge er en av de mest unike egenskapene ved et kjøretøy. Metoden valgt for fargedeteksjon er enkel og lite ressurs krevende. Et lite utsnitt av et område på panseret skal hentes ut for å beregne en middelverdi av fargeintensiteten. Middelverdien sammenlignes mot andre kjøretøykandidater. Avstanden vurderes opp mot en terskelverdi (grenseverdi) som grupperer kjøretøyene til *lik*, eller *ulik*. Siden metoden har utfordringer ved sterkt lys og skygge, blir ulike fargerom testet for å finne hva som egner seg best til oppgaven. Formålet med fargesammenlikning er å skille eller "filtrere" vekk usannsynlige kandidater og dermed redusere antallet sammenlikninger.

3. Struktur
Lette kjøretøyklasser som for eksempel personbiler, har ofte store likheter i form og farge. Da kan informasjon som forteller noe om strukturformen til kjøretøyet være nyttig. Strukturmetodene som er plukket ut i denne oppgaven for å testes mot hverandre er: **Histogram of Oriented Gradients (HOG)** og **trace transformen**.

- Histogram of Oriented Gradients leter etter gradientorienteringer (kantstruktur) i et bilde og har som fordel å være invariant mot lokale geometriske og fotometriske (lysstyrke) transformasjoner [13].
- Trace transformen kan tenkes som utregning av projeksjoner langs ulike snitt i et bilde. I motsetning til Radon transformen der projeksjonene er basert på integralet til en funksjon over en rett linje, beregnes mange andre funksjoner ved trace transformen. Dermed kan trace transformen tenkes på som en generalisering av Radon transformen. Trace transformen henter ut bildeegenskaper i form av signaturer som er robuste mot rotasjon, skalering og globale lysendringer [35].

Mange artikler i tidligere arbeid har hatt et fokus på deteksjon av egenskaper som omhandler hele kjøretøyet[19][65]. I denne oppgavene er det valgt å fokusere på et bestemt bildeområde, nemlig kjøretøyfronten. Denne delen av kjøretøyet inneholder logo, grill, frontlykt og har derfor sterke karakteristiske trekk som er med på å gjøre området svært unikt fra andre kjøretøy. En annen fordel ved å sammenlikne strukturen i et fast område av kjøretøyet, i dette tilfelle fronten, er at vi kan oppnå en større invarians mot bildetransformasjoner og unngå støy der bakgrunnen blir en del av sammenlikningen. Fronten til ulike biler vises i Figur 1.6. Denne blir detektert ved å ta utgangspunkt i skiltposisjonen.

Figur 1.6 Ulike personbilfronter.

4. Skiltfarge

Kameraposisjonen som er valgt i oppgaven gjør at skilt detekteres ved lav bildeoppløsning. Dette gjør dem vanskelig å lese. Det er derimot fullt mulig å gjenkjenne fargen til skiltet som kan gi oss nyttig informasjon. Ved en forbipasserende varebil som har grønne skilt, kan vi anta færre passasjerer. Skiltfargen blir derfor en av egenskapene vi skal detektere i oppgaven.

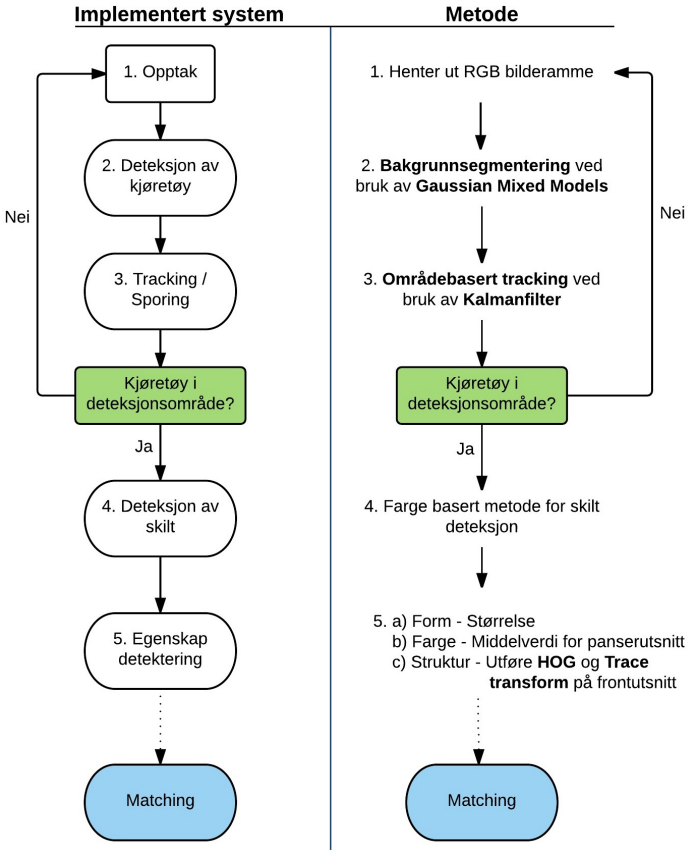
En liten oppsummering av egenskapene er vist i Tabell 1.1. I fremtiden kan systemet utvides med flere egenskaper og mønstergjenkjenning kan brukes for å finne hvilke kjøretøy som mest sannsynlig er like.

Tabell 1.1 Her vises egenskapene som detekteres, hva slags deskriptor som brukes og om informasjonen skal sendes videre til VTS.

Egenskap	Deskriptor	Til VTS
1. Størrelse	Binær	Klassifisering: Lett/Tung
2. Farge middelverdi	Skalar	Nei
3. Front signatur	1D vektor	Nei
4. Skilt farge	Farge navn	Varebil: J/N

1.2.3 Kort beskrivelse av systemet

Et flytskjema som beskriver systemet implementert i oppgaven fra start til slutt er vist i Figur 1.7.



Figur 1.7 Systemets flytskjema.

Teorien bak metodene er gitt i Kapittel 2. Matching av kjøretøy er forklart i Figur 4.20. Implementeringen av systemet er forklart i Kapittel 4.

1.2.4 Utfordringer

- Påvirkning fra miljø og vær skaper skygger, refleksjoner og lysendringer.
- Bilkø skaper okklusjon (tildekking).
- Geometrisk transformasjon av kjøretøyobjektet oppstår mellom kameraopptak filmet fra forskjellig observasjonspunkter.
- Systemet krever annen algoritme ved kjøring under svake lysforhold som for eksempel sent på døgnet.

- Kjøretid til algoritmen som detekterer egenskaper.
- Størrelse på egenskapvektoren.

1.3 Rapportens innhold og struktur

Kapittel 2 - Teori bak metodene
I dette kapittelet beskrives teorien bak metodene som brukes i oppgaven. Dette omhandler teori for detektering av kjøretøy ved bakgrunnssubtraksjon og "områdebasert" tracking ved bruk av Kalmanfilter. Videre beskrives også teorien bak egenskapmetodene som er valgt: trace transform, HOG, fargemiddelverdi av panserutsnitt samt forklaring av valgte fargerom.

Kapittel 3 - Skiltdeteksjon
I dette kapittelet forklares skiltdeteksjonsalgoritmen som er laget ved å kombinere tidligere arbeid med egne ideer for å tilpasse formålet i oppgaven.

Kapittel 4 - Implementering av metodene
I dette kapittelet blir oppbygningen og implementeringen av systemet i oppgaven beskrevet. Her vil egenimplementerte algoritmer som bidrar til løsning på utfordringer ved ulike del-ledd i systemet stå i fokus.

Kapittel 5 - Eksperimentell del og drøfting av resultat
I dette kapittelet testes HOG og trace transform deskriptorer mot hverandre under ideelle forhold, lysendring og okklusjon med mål å matche flest kjøretøyfronter korrekt. Det utføres også en test av hvilket fargerom som best egner seg til beregning av middelverdi for panser-utsnitt og et resultat av bakgrunnssubtraksjon legges frem. Resultatet vil også bli diskutert her.

Kapittel 6 - Konklusjon
I dette kapittelet vil det bli lagt frem en konklusjon ut ifra problemstilling og resultat.

Kapittel 2

Teori

Med henvisning til Figur 1.7 som viser et flyteskjema for systemet, skal teorien bak metodene som brukes i oppgaven utdypes i dette kapittelet. Algoritmen i den første delen av systemet har som oppgave å detektere et kjøretøyobjekt. Dette er en krevende oppgave som kan løses ut fra et stort utvalg av tilgjengelige metoder. I oppgaven er det valgt å detektere kjøretøy gjennom bakgrunnssubtraksjon ved bruk av Gaussian Mixed Models. Teorien bak denne metoden vil bli gitt i dette kapittelet. For å assosiere hvert kjøretøy med en unik identitet, må kjøretøyene ”trackes eller også kalt spores gjennom bildesekvensen. Her er det mulig å velge mellom flere ulike metoder. Disse vil bli presentert i dette kapittelet. Begrunnelse for valget av metoden samt implementeringen blir forklart i Kapittel 4.

Den neste delen av kapittelet handler om egenskapsdeteksjon. Kjøretøyets egenskaper skal detekteres når et valgt område på veibanen passeres. Disse egenskapene skal videre brukes til å kjenne igjen samme kjøretøy ved en senere anledning. Egenskapene er delt inn i 3 kategorier; Form, farge og struktur. Fordi deteksjon av egenskaper fra disse tre kategoriene er oppgavens problemstilling, legges det stor vekt på å presentere tidligere arbeid som omhandler deteksjon av disse egenskapene. Videre utdypes nødvendig teori for å kunne forstå metodene for egenskapsdeteksjon som er valgt i oppgaven. For egenskapen som omhandler farge, vil teorien bak fargerommene HSV og YCbCr bli presentert. Disse fargerommene skal i oppgaven testes for beregning av middelverdi av fargen på panseret. Teorien bak beregningen av middelverdien blir også gitt her. For metoder som detekterer strukturliknende egenskaper skal teorien bak HOG og trace transform presenteres. Disse metodene skal brukes til å hente ut egenskaper fra kjøretøyfronten til kjøretøyet.

2.1 Deteksjon av kjøretøy

I dette underkapittelet presenteres det metoder og teori som går under kategorien ”deteksjon av kjøretøy”. Dette er første ledd i prosessen som må utføres før vi kan hente ut kjøretøy-egenskaper. Videre blir teorien bak *bakgrunnssubtraksjon* som brukes til å detektere kjøretøy utdypet. Til slutt forklares teorien bak ”*tracking*” som brukes i systemet for å spore kjøretøy langs veistrekningen.

2.1.1 Bakgrunnssubtraksjon

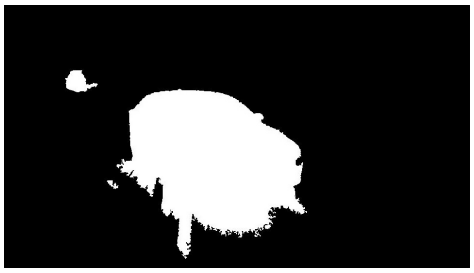
Deteksjon av objekter er en viktig del av applikasjoner som omhandler overvåkning. Dette er fordi det ofte letes etter et bestemt objekt. For at deteksjonen skal foregå raskest mulig, er det viktig å ha et lite leteområde. For eksempel ved detektering av skilt, vil det være raskere å lete i et mindre bildeområde som kun inneholder kjøretøyobjektet. Derfor er det veldig viktig å skille vekk den overflødige informasjonen som ligger i bakgrunnen som kan anses som støy.

Prosessen som omhandler kjøretøydeteksjon kan deles i to, deteksjon og verifisering av kjøretøykandidat. Til å detektere objekter som kan anses som kjøretøykandidater, finnes det i dag flere metoder. *Kunnskapsbaserte metoder* anvender ”priori” kunnskap for å danne en hypotese om kjøretøyets posisjon i bildet. Metoden kan bruke informasjon om symmmetri, farge, skygge, frontlyktene på kjøretøyet eller geometriske egenskaper som kanter og hjørner. Det finnes også *stereobaserte metoder* som danner et 3D-kart (scene) for å lete etter kjøretøyobjekter. Disse metodene kan være komplekse. En eklere metode er *bevegelsesbaserte metoder* som leter etter endringer i bakgrunnen. Felles for disse algoritmene er at de detekterer potensielle kjøretøykandidater i et bilde ut fra hypotese om mulig posisjon. For å bekrefte funn av kjøretøy, må kandidatene først verifiseres.

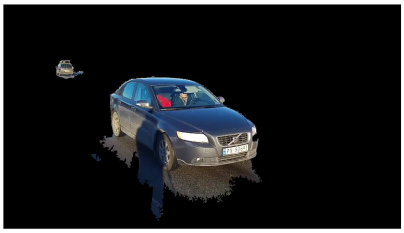
Verifisering av kjøretøy kan foregå ved å sammenlikne kandidaten mot en mal og beregne en korrelasjonskoeffisient. En annen brukt metode lærer karakteristikker fra et sett med treningsbilder og bruker en klassifiserer til å danne en beslutningsgrense mellom kjøretøy og ikke kjøretøy. Trening av en slik klassifiserer er ofte svært ressurskrevende på grunn av antallet treningsbilder som kreves for å oppnå en pålitelig beslutningsgrense. Informasjon om kjøretøydeteksjon og en gjennomgang av metoder for verifisering av kjøretøykandidater er hentet ut fra Z. Sun [85] og M.S.P. Patil [57]. Nylig ble også en av de mest omfattende studiene av ulike algoritmer til å detektere kjøretøy ved et kamera i bevegelse (montert på kjøretøy) gjort av R. M. Haralic [71].

For å detektere kjøretøykandidater er en *bevegelsesbasert* metode som kalles bakgrunnssubtraksjon blitt valgt. Dette er fordi metoden er enkel å implementere og lite ressurskrevende. Metoden er mye brukt for å finne objekter i bevegelse hvor statiske kamera brukes. For verifisering av kjøretøy er en enkel metode valgt som tar utgangspunkt i størrelsen av kjøretøyobjektet og forklares i Kapittel 4.4.1. Figur 2.1 viser noen av utfordringene med bakgrunnssubtraksjon, hvor skygger og refleksjoner fører til at mer enn bare kjøretøyet blir tatt med som forgrunn. Det er mulig å finjustere sensitiviteten til algoritmen, men dette kan føre til at den blir tregere til å oppfatte endringer.

(a) Bilderamme.



(b) Forgrunnsmaske.



(c) Bakgrunn fjernet.

Figur 2.1 Bakgrunnssubtraksjon.

Bakgrunnssubtraksjon har blitt studert mye siden 1990, hovedsakelig for applikasjoner innen videoovervåking [75]. I denne oppgaven brukes metoden fordi tyngre operasjoner som egenskapdetektering og tracking av kjøretøy krever en forgrunnmaske (kjøretøyet). Forgrunnen detekteres ved å sammenligne neste bilderamme med en modell av bakgrunnen, noe som resulterer i en binær maske vist i Figur 2.1b. Den enkleste måten å modellere bakgrunnen på er å bruke et bilde av bakgrunnen som ikke inneholder et bevegelig kjøretøyobjekt. Under kritiske situasjoner som mye trafikk, vær og endringer i belysning, kan det derfor være krevende å lage en maske. For at endringene i omgivelsene ikke skal bli misstolket som forgrunn, må modellen regnes ut over flere bilderammer, bli oppdatert iterativt eller etter en viss tid.

En enkel algoritme for bakgrunn subtraksjon ble først foreslått i 1987 av M.K. Leung, et. al. [43] der nåværende bilde I_t ved tid t blir subtrahert med bakgrunnsmodell B for å detektere forgrunn. En sekvens av tidligere bilder ved tid $t-1, t-2, t-3...$, blir brukt til å danne en historisk modell som beskriver bakgrunnen. Matematisk kan det forklares som følgende:

$$F(x,y) = \begin{cases} 1 & I_t(x,y) - B(x,y) \geq T \\ 0 & ellers \end{cases} \tag{2.1}$$

$I_t(x,y)$ er pikselverdien som funksjon av piksel-koordinatene x,y (horisontal og vertikal retning), F er den resulterende binære forgrunnsmasken. T er en satt terskelverdi som bestemmer om piksel endringen er stor nok til å kvalifiseres som bakgrunn eller forgrunn. Denne enkle metoden faller rask sammen når bakgrunnen er dynamisk.

Gaussian Mixed Models

Det eksisterer flere metoder for bakgrunnssubtraksjon. Disse har ulik ytelse i form av prosesseringskrav og robusthet mot støy. Statistiske modeller tilbyr større robusthet mot endringer av lysforhold og dynamikk i bakgrunnen [5]. En parametrisk sannsynlighet bakgrunnsmodell som både har vist seg å være svært populær og gi svært god ytelse [75] [74], er en blanding av gaussiske Modeller (Gaussian Mixed Models (GMM) eller Mixture of Gaussian models (MOG)) utviklet av Stauffer og Grimson [83] og senere forbedret av Hayman og Eklundh [26]. Siden miljømessige faktorer bidrar til at pikselverdiene i bakgrunnen endrer seg over tid, går hovedideen bak GMM ut på å danne en modell av historien til intensitsetendringene til hvert piksel i form av en gaussisk funksjon.

Alternativet til å bruke en enkel gaussisk funksjon for å beskrive pikselvariasjonen til bakgrunnsmodellen, er å bruke en blanding av flere gaussiske tetthetsfunksjoner. Områder i videoen som inneholder for eksempel viftende trær og rennende vann, har en varians som er i konstant endring. Derfor er bruken av flere modeller av bakgrunnen nødvendig. De gaussiske modellene av piksel-historikken som har høyest varians blir stemplet som forgrunnsmodell. Ved et nytt bilde I_t, bestemmes hver piksel til forgrunn eller bakgrunn ut fra hvilken av modellene som passer best med den nye pikselverdien. Som et resultat av metodens popularitet, har mange forbedringer av moden blitt gjort de siste årene. En grundig undersøkelse om forbedringene kan det leses mer om i [6]. Blant forbedringene er evnen til å håndtere raske variasjoner i lysforholdene.

Metoden fungerer ved at distribusjonen til pikselintensiteten blir modellert som en sum av vektet gaussiske distribusjoner med forskjellige median og standardavvik til et gitt fargerom. For at algoritmen skal være robust mot variasjoner i omgivelsene, blir bakgrunnmodellen oppdatert ettersom nye bilder prosesseres. Matematisk beskrives sannsynligheten for at piksler er en del av bakgrunnen ved tid t som:

$$P(I_t|B_t) = \sum_{n=1}^{N} \frac{\alpha_n}{(2\pi)^{d/2}|\sum_n|^{1/2}} e^{-\frac{1}{2}(I_t-\mu_n)' \Sigma^{-1}(I_t-\mu_n)} \tag{2.2}$$

Her er $I_0, I_1, ..., I_{t-1}$ pikselverdien ved tid t, B er bakgrunnsmodellen, d er dimensjonen for fargerommet (3 for RGB) og hver gaussisk funksjon n er beskrevet av sitt gjennomsnitt μ og kovarians matrisen $\sum_n$. Antall distribusjoner er gitt av N som bestemmes etter tilgjengelig minne og prosesseringskraft. Verdier mellom 3 og 5 blir brukt. α_n er veiefaktoren for de ulike gaussiske funksjonene til hver piksel. Summen av veiefaktoren α_n er 1. For å senke tiden det tar å uføre algoritmen, antar Stauffer og Grimson [83], at RGB komponentene er uavhengig av hverandre og har lik varians, selv om det i den virkelige verden ikke ville vært tilfelle. Kovariansmatrisen for RGB kanalene er gitt i likning (2.3).

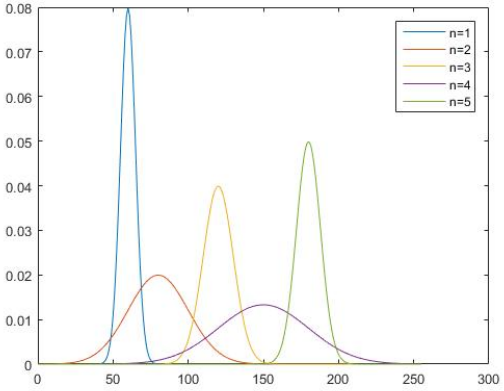
$$\sum = \sigma_n^2 I = \begin{pmatrix} \sigma_n^2 & 0 & 0 \\ 0 & \sigma_n^2 & 0 \\ 0 & 0 & \sigma_n^2 \\ \end{pmatrix} \tag{2.3}$$

Distribusjonen av nylig observerte pikselverdier kan beskrives som en blanding av gaussiske funksjoner, visualisert i Figur 2.2.

Hver ny pixel $I_t(x,y)$ ved tid t sammenlignes mot eksisterende N gaussisk distribusjon til en match er funnet. En match defineres når en piksel verdi er innen 2.5 standard avvik av en normaldistribusjon. Under viser formelen for beregningen av distansen der k gir en terskelverdi mellom 2 og 3 standardavvik.

$$||I_t - \mu_n|| < k\sigma_n \tag{2.4}$$

Denne varierende terskelverdi er veldig verdifull der forskjellige bilderegioner har ulik lysstyrke. Ved en uniform terskelverdi vil objekter forsvinner når de kommer inn i områder



Figur 2.2 Gaussike distribusjoner for piksel $I_t(x,y)$

med mye skygge. Hvis ingen av n distribusjoner matcher nåværende pikselverdi, blir distribusjonen med minst sannsynlighet erstattet med en distribusjon som bruker nåværende verdi $I_t(x,y)$ for μ, σ^2 settes høy og veiefaktoren α_n blir satt lav. Derimot om piksel $I_t(x,y)$ gir et sant utslag i likning (2.4), skal parameterne for distribusjonen som er matchet oppdateres. Oppdatering av vekt for distribusjonen er gitt følgende:

$$\alpha_{n,t} = (1-\delta)\alpha_{n,t-1} + \delta \tag{2.5}$$

δ er her læringskoeffisienten og beskrives som adapteringstiden til modellen. Videre blir de resterende parameterne for den nye observasjonen som har funnet en matchende distribusjonen, oppdatert på følgene vis:

$$\mu_{n,t} = (1-\rho_n)\mu_{n,t-1} + \rho_n I_t \tag{2.6}$$

$$\sigma^2_{n,t} = (1-\rho_n)\sigma^2_{n,t-1} + \rho_n (I_t - \mu_{n,t})^{'} (I_t - \mu_{n,t}) \tag{2.7}$$

der

$$\rho_n = \delta \eta (I_t | \mu_n, \sigma_n) \tag{2.8}$$

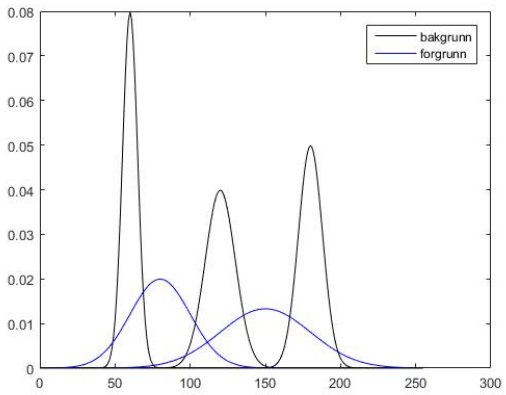
η her betyr normalfordeling. For de resterende distribusjonene, beholdes μ og σ parameterne ved like mens veiefaktoren α_n justeres. Dermed blir en viktig fordel med GMM

som nevnes av forfatterne at metoden muliggjør objekter til å bli en del av bakgrunnen uten å ødelegge eksisterende modeller. Hvis et objekt er stasjonert lenge nok til å bli en del av bakgrunnen, skulle bevege seg, vil distribusjoner som beskriver tidligere bakgrunn fortsatt eksistere, bare med lavere veiefaktor.

Videre må vi finne hvilke av gaussiske distribusjoner n som tilhører forgrunnen og bakgrunnen. Et objekt som skal detekteres er i bevegelse og derfor vil en distribusjon som representerer forgrunnen ha større varians og mindre veiefaktor. Stauffer og Grimson, [83], rangerer derfor N antall GMM etter forholdet α_n/σ_n. Denne verdien øker når variansen minker. Etter rangeringen, blir de første B gaussiske distribusjonene som overskrider en viss terskel T, stemplet som bakgrunn:

$$B = argmin_b(\sum_{n=1}^{b} \alpha_n > T) \tag{2.9}$$

Et eksempel på en mulig klassifisering ut fra Figur 2.2 vises i Figur 2.3.



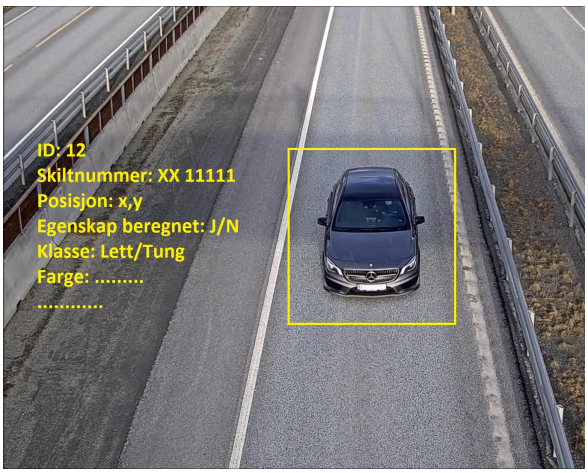
Figur 2.3 Klassefisering av GMM

Ulempen med metoden kan være at den har vanskeligheter med å takle plutselige og raske lysendringer og håndtering av okklusjon mellom overlappende kjøretøy. Implementering av bakgrunnssubtraksjon, validering av kjøretøykandidater og fjerning av skygge er forklart i Kapittel 4.4.1.

2.1.2 Tracking

I en scene er det ofte ønskelig å følge en eller flere objekter av interesse for å holde det under oppsyn eller å innhente posisjonen. Applikasjoner som omhandler ”tracking” eller også kalt ”sporing” er mange og representerer en av de mest grunnleggende oppgaver i maskinsyn. For å utføre en nøyaktig sporing i en video, kreves det en god deteksjon av objektet, noe som øker kompleksiteten til oppgaven. Videosporing bygger på å knytte sammen bestemte objekter i en bildesekvens. Ofte blir bevegelsesinformasjon brukt til å assosiere ulike objekter med hverandre.

For at systemet i oppgaven skal vite antall motorkjøretøy som kjører inn i en tunnel, må det holdes telling på kjøretøyene som passerer kameraet. Dette kan gjøres ved å spore bilene til et bestemt punkt på veien hvor en egenskapsdetektering så vil foregå. En annen fordel med å spore er at hvert kjøretøy kan assosieres med en unik id-vektor. Denne id-vektoren kan holde informasjon som om en egenskapsvektor er beregnet eller posisjonen til kjøretøy-objektet i tidligere bilderammer. Dermed muliggjøres fartsberegning om ønskelig senere. Eksempel på sporing er vist i Figur 2.4.



Figur 2.4 En id-vektor assosieres med hvert kjøretøy når det spores.

I applikasjoner for trafikkovervåking er sporing av kjøretøy svært utbredt. Flere forskjellig metoder har blitt foreslått med hver sine fordeler og ulemper. Med utgangspunkt fra litteratur om maskinsyn av B. Coifman, et. al. [11] og A. P. Shukla, et. al. [70], kan forskjellige trackingsteknikker bli delt opp som følgende:

Modellbasert tracking:
En mulighet til å spore kjøretøy, gjøres ved å matche ulike nettrammemodeller (wireframe-models) av kjøretøy i videobilder. En nettrammene er en geometrisk 3D-modell av et objekt. Eksempel på wireframe-tracking er gjort av J. M. Ferryman, et. al. [18], som kombinerer 3D-nettrammemodell med en intensitetsmodell av et kjøretøy for å lære en modell av utseende over tid. En ulempe med modellbasert tracking, er det store antallet modeller som kreves p.g.a. forskjellige kjøretøypositurer og kameravinkler. Å inneha en detaljert modell for alle kjøretøyene på en motorvei er således urealistisk.

Områdebasert tracking:
I denne trackingsmetoden blir en sammenbundet område i et bilde (en blob), identifisert og sporet over tid i videofilmen. Blob-området representerer kjøretøyobjekter og initialiseres ved å bruke bakgrunnssubtraksjon. Videre kan en områdebasert trackingsteknikk som Kalmanfilter brukes til å spore blob-områdene. Denne metoden fungerer bra i områder med lite trafikk, men får problemer med å isolere ulike kjøretøy fra hverandre når delvis okklusjon oppstår. Ved trafikkork kan kjøretøy bli gruppert sammen til en stor klump.

Aktiv konturbasert tracking:
Hovedideen her er å representere kjøretøyet gjennom sin kontur (grensekanten til kjøretøyet) som dynamisk oppdateres i påfølgende kjøretøybilder. Fordelen med metoden er en reduksjon av beregningskompleksitet sammenlignet med områdebasert tracking. Til ulempe gjenstår okklusjon som et problem og presisjonen er begrenset til presis lokasjon av konturen.

Egenskapbasert tracking:
Istedenfor å spore hele objektet som gjøres ved andre metoder, velges det her bare å spore fremtredende egenskaper. Egenskapene kan være et sett av hjørnepunkter som sammen grupperes i henhold til et bevegelsesmønster. Ved å spore ulike kjøretøyegenskaper, er metoden mer robust mot delvis okklusjon. Utfordring videre er å gruppere egenskapene til riktig kjøretøy. En metode som sporer kjøretøy i trafikk ved å detektere hjørneegenskaper er foreslått av B. Coifman, et. al. [11].

I oppgaven er det valgt å bruke områdebasert tracking med begrunnelse forklart i Kapittel 4.4.2. Her brukes et Kalmanfilter til å predikere posisjonen til kjøretøyet i nåværende bilderamme. Kalmanfilter kan brukes hvor vi har usikker informasjon om et dynamisk system og vil gjette seg frem til systemets neste steg. Filteret har fått navnet sitt i fra Rudolf E. Kalman som publiserte en artikkel om metoden i 1960 [37].

2.2 Generelt om egenskapsdeteksjon

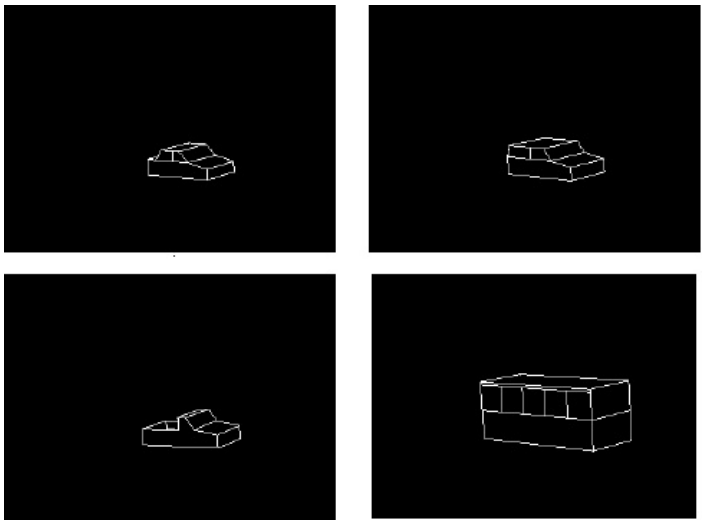
En bildeegenskap er et ganske bredt begrep, men det betyr som regel en del av et bilde som inneholder interessante egenskaper eller detaljer vi er interessert å detektere. Disse kan være alt fra visuelle egenskaper som farge, form, kanter, tekstur, symmetri, men også bevegelseinformasjon hentet ut fra en sekvens med bilder. Valg av hvilken interessant egenskap som skal hentes ut er avhengig av hva applikasjonen ønsker å oppnå. I maskinsyn er detektering av egenskaper en viktig prosess i applikasjoner som inkluderer overvåking, 3D-rekonstruksjon, objektgjenkjenning, bevegelsessporing og hendelsesdeteksjon. Flere maskinsynalgoritmer bruker egenskapsdetektering som en del av første steg og derfor har mange egenskapsdetektorer blitt utviklet. Når egenskapene er funnet, må en representere den i form av egenskapsvektorer. Disse vektorene kalles *deskriptorer* og kan sammenlignes med numeriske "fingeravtrykk", brukt til å finne den samme egenskapen igjen i et senere bilde. Dette gjøres for å redusere mengden av data som kreves for å beskrive et bildeområde som korresponderer til en egenskap.

Noen egenskapsvektorer er svært enkle. Eksempelvis kan en farge representeres med bare et ord eller tall. En mer kompleks arkitektur som bildeutsnitt og figurform kan representeres ved å beregne signaturer. En signatur er en 1D-funksjon brukt til å representere et utsnitt og kan genereres på forskjellige måter [21]. Den grunnleggende ideen er å redusere dimensjonaliteten til et bildeutsnittet ved å representere det på en enklere måte enn originalen. En enkel måte å beregne en signaturvektor på er ved å beregne snittdistansen fra senteret av bildeobjektet til kanten som funksjon av vinkelen.
Egenskapsdetektering kan også utføres ved å lete etter egenskapspunkter, også kalt interessepunkter. Disse kan finnes ved å bruke en egenskapsdetektor til å søke etter punkter i form av kanter, hjørner og blob regioner (blob er et sammenbundet område med like egenskaper). Til beskrivelsen av egenskapspunktet, brukes normalt en deskriptor som er dannet ut fra et lokalt nærområde rundt punktet i form av et histogram. Når en deskriptor eller egenskapsvektor er funnet, kan den sammenliknes med andre for å finne korresponderende interessepunkter i ulike bilder. Matching kan utføres ved å finne minste avstand mellom de ulike vektorene ved f.eks Mahalanobis- eller Euclidean-avstand [4]. For at sammenlikningen skal være rask og pålitelig, må vektoren være kort, robust mot støy og bildetransformasjoner. Bildetransformasjonene kan være skala endring forårsaket av forskjellig innstilling av zoom på kamera, rotasjon som skjer ut fra forskjellige observasjonspunkter, lys- og skygge-endringer, samt okklusjon der kjøretøy kan være tildekke av andre.

2.3 Form-egenskap

Formen er en viktig faktoren til å kunne skille ulike kjøretøy-typer også kalt -klasser fra hverandre. For eksempel kan størrelse enkelt brukes til å skille en personbil fra en lastebil. For å kunne gjenkjenne flere kjøretøytyper, kreves en mer avansert metode. Populært er dette gjort ved å danne en numerisk tredimensjonal modell ut fra geometrien på objektet som videre sammenliknes mot et sett av tredimensjonale bokser. En slik metode kalles ”wire-frame” modell.

Metoden fungerer på følgende vis. Ut fra kantene til kjøretøyet, dannes en tredimensjonal nettmodell som beskriver formen på objektet. Bestemmelse av kjøretøyklassen kan gjøres ved å sammenlikne kantlinjene eller ”trådene” i modellen mot et sett tredimensjonale bokser som representerer ulike kjøretøytyper (buss, pickup, personbil). Denne metoden har blitt foreslått i arbeidet til Ambardekar, et. al. [42], og Z. Chen, et al. [9]. Bilde av modellen er vist i Figur 2.5.



Figur 2.5 3D-”wire-frame” modell av ulike kjøretøytyper [42]

En mer avansert geometrisk kjøretøyskrivelse kan gjøres dersom alle kantene kan detekteres og deres 3D-lokalisering og orientering kan rekonstrueres ved å kombinere informasjonen om farge, albedo og tekstur. Dette er blitt gjort i arbeidet til Y. Guo, et. al. [24], som estimerer positurtransformasjonen til et kjøretøy og matcher den med tilnærmet tredimensjonale kjøretøymodeller. En samling av kantlinjer og punkter er også blitt brukt i Y. Guo, et. al. [23], til å matchet ulike kjøretøy under forskjellige belysninger. Derimot kan begrenset regnekraft og kameraoppløsning gjøre en troverdig 3D-rekonstruksjon vanskelig. En annen utfordring skjer når refleksjoner fra sola bidrar til å jevne ut kantrensene i bildet. Referanse til tidligere

metoder som bruker dimensjonen på kjøretøyet som lengde, bredde og høyde til klassifisering er gjort i arbeidet til H. Asaidi, et. al. [1].

Det er også populært å bruke en lokal egeskapsdetektor til å finne punkter i et bilde som kan brukes til å rekonstruere objektet eller finne tilsvarende punkt i et nytt bilde. Nøyaktigheten i bruken av egenskapsdetektor avhenger av at samme punkt detekteres til tross for endringer av observeringsvinkel. SIFT [51] (Scale Invariant Feature Transform) og SURF [4] (Speeded Up Robust Features) er to eksempler på populære egenskapsdetektorer. I arbeidet til Y. Guo et al.,[23], er det blitt konkludert at SIFT-lignende egenskaper ikke kan brukes på kjøretøybilder av lav oppløsning.

Selv om modellbasert kjøretøybeskrivelse er den mest populære måten å klassifisere ulike kjøretøytyper på [62], er kanskje den største utfordringen med metoden antallet modeller som trengs for å representere en enkel kjøretøytype. Årsaken til dette er geometriske transformasjonen som skapes ved forskjellige observasjonspunkter og den store variasjonen av kjøretøyformer. I oppgaven er det begrenset av tid til å implementere en slik modell og derfor er det valgt å bruke en rask og enkel metode for å dele et kjøretøy inn ito klasser; *lett* og *tung* kjøretøy. Klassen bestemmes ut fra størrelsen av kjøretøybilde (markeringsboksen) som detekteres ved bakgrunnsubtraksjon. Parametere og implementering av metoden er forklart i Kapittel 4.4.4.

2.4 Farge-egenskap

Farge er en mektig deskriptor som kan forenkle beskrivelsen til et objekt. Ulike farger på karosseriet, bidrar til å gjøre farge til en av de viktigste egenskapene som kan brukes for å skille et kjøretøy fra et annet. I de siste årene har derfor fargeegenskap blitt mye brukt til å detektere og gjenkjenne forskjellige kjøretøy. Det finnes likevel en del utfordringer; f.eks. bilde av en gul bil vil se hvit ut ved sterk sollys og motlys fra innkommende biler i en svakt opplyst tunnel, kan bidra til at feil farge oppfattes. Andre feilkilder som fører til et fargeskift i forskjellig observasjonspunkter er overflatereffleksjoner forårsaket av sola eller andre lyskilder og forskjellige kamerainnstillinger. Faktorene bidrar til utfordringer å bruke farge som en pålitelig egenskap i kjøretøygjenkjenning.

Det finnes flere måter å bruke fargeinformasjon i kjøretøygjenkjenning. En metode er å navngi et valgt område eller hele kjøretøyet som en ”visuell” farge, som eksempel rød, brun, gul, grønn, blå, svart, hvit eller grå/sølv. En fremgangsmåte foreslått av Weijer, et. al. [88], trener opp en algoritme som klassifiserer et bildeområde til en av 11 valgte basisfarger under varierende forhold skapt ved skygger og sollys. K. Kim, et. al. [39], har skrevet en artikkel

om gjenkjenning av fargenavn på et kjøretøy ut fra 7 mest populære bilfarger. Bestemmelse av fargen ut fra et område på panseret er blitt skrevet om i artikkelen av E. Dule, et. al. [16]. En mer avansert metode er blitt publisert av H. Gu og S. Lee [22], som omhandler å skille kjøretøy-kroppenfra bakgrunnen og videre bestemme kjøretøyfargen ut fra pikslene på kjøretøyrammen. En nylig artikkel som bruker "deep learning", på norsk "dyp læring" for å gjenkjenne kjøretøyfargen og sammenligne resultatet med tidligere arbeid, er blitt publisert av C. Hu, et al. [29].

Hva alle disse artiklene har til felles, er at bestemmelsen av fargen gjøres ved å trene opp en klassifiserer. En klassifiserer er en metode som brukes til å bestemme den mest sannsynlige klassen (f.eks: farge, biltype osv.) til et ukjent objekt. Trening av en slik klassifiserer behøver en treningsdata med mange bilder av kjøretøy under varierende lysforhold for å gjøre metoden robust. Derfor er en annen metode ofte brukt som ser på fargelikheten i et bestemt område på kjøretøyet. Dette er gjort i Y. Guo, et al. [23], som danner fargemaske av kjøretøyobjektet og bruker denne informasjonen til å kjenne igjen samme kjøretøy ved en annen anledning. Det finnes også enklere måter å bruke fargeinformasjon i kjøretøygjenkjenning. Fargen til et valgt område ved kjøretøyet, kan representeres i form av histogram, median eller korrelasjon av fargen (fargefordelingen beskrives som funksjon av distanse mellom to piksler [30]) og videre sammenliknes med ulike kandidater for å finne likheter.

For at en slik metode skal fungere problemfritt i et multi-kamera system der opptakene gjøres under ulike lysforhold og kameraparametere, må en farge og lysstyrke overførings-funkjson beregnes. Fremgangsmåten for overføring av lysstyrke mellom kameraer er blitt foreslått av Javed et. al. [33]. I artikkelen til F. Porikli [59], forelås en annen variant der korresponderende farger mellom de ulike kameraene kartlegges for å beregne forskyvningen. I virkeligheten er en slik fargeoverføringsmodell vanskelig å beregne grunnet til flere usikre variable. I oppgaven er det valgt å bruke en metode som beregner en middelverdi av fargen på et utsnitt av panseret. Begrunnelsen til valg av metode samt implementeringen er forklart i Kapittel 4.4.5.

2.4.1 Introduksjon til ulike fargerom

For å beregne middelverdien av pikselverdien i et område, er det viktig å velge et fargerom som er best egnet til oppgaven. Et fargerom er en matematisk modell for å representere farge i form av verdier [90]. Typisk har et fargerom 3 dimensjoner der hver dimensjon er en fargekanal. Det finnes mange ulike fargerom. Alle med side fordeler og ulemper. For å velge det fargerommet som er best egnet til bruk i en fargesammenlikning, må ulike fargerom testes. Det har derfor blitt gjort en test i Kapittel 5.4 der YCbCr fargerom viste seg å gi best

resultat. Videre i dette kapittelet vil ulike fargerom introduseres og teorien bak YCbCr og HSV fargerom blir lagt frem da disse brukes i oppgaven. Til slutt i kapittelet vil teorien for beregning av middelverdien og avstandsmål legges frem.

RGB fargerom

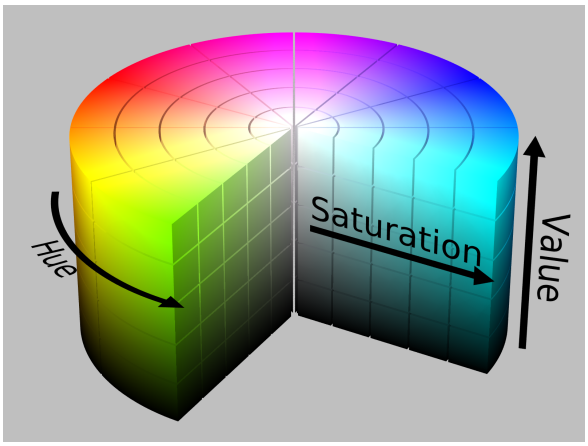
RGB fargerom som står for det samme på norsk og engelsk: red-green-blue eller rød-grønn-blå, utgjør disse 3 komponentene fargekanalene. Hovedulempen i bruk av RGB fargerommet i maskinsyn er den høye korrelasjonen mellom fargekanalene. Dette er fordi alle fargekanalene inneholder informasjon om luminans (mål på hvor lys en flate er [8]). En senking av kontrastnivået i et bilde fører til reduksjon av verdien i alle fargekanalen, dermed gjør det svært vanskelig å utføre en fargesammenlikning ved ulike lysvariasjoner. RGB fargerom representerer heller ikke farge slik vi mennesker oppfatter dem. Vi definerer ikke farge ut fra hvor mange prosent de inneholder av hver primærfarge.

Lab fargerom

Lab er et populært fargerom med dimensjonene L for lightness (lyshet), a (fargekomponent som gir fargen verdi fra grønt til rødt) og b (angir fargen mellom gult og blått) [10]. Lab er et fargerom med flere fordeler over RGB. En egen kanal L brukes til å skille ut lysstyrken i en scene, i tillegg inneholder Lab teoretisk alle farger mulig å oppfatte. En av de viktigste fordelene med Lab er uavhengighet fra utstyret det opereres på. Et utstyr uavhengig fargerom er et fargerom der valgte parametere produserer samme farge uavhengig av utstyret som brukes [12]. Ulempen med metoden er at konvertering fra RGB til Lab fargerom behøver utregning av kubikkroten og dermed er operasjonen beregningsmessig mer krevende enn transformasjon til HSV.

HSV fargerom

Med motivasjon å kunne danne en modell som beskriver farge nærmere slik den oppfattes av mennesker, ble HSV (Hue, Saturation, Value) modellen utviklet på midten av 1970 tallet ved New York Institute of Technology og Palo Alto Center Incorporated og først ble beskrevet i 1978 av Alvy Ray Smith i en utgave av Computer Graphics [72]. Andre variasjoner av fargerommet er gitt ved HSL (L for lightness), HSB (B for brightness) og HSI (I for intensity). HSV fargerom representeres i form av sylinderkoordinater vist i Figur 2.6.

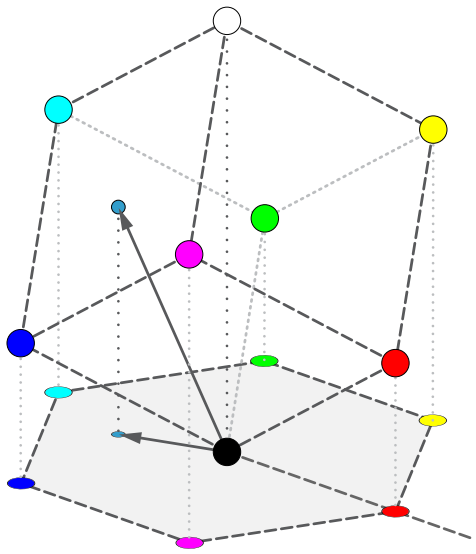


Figur 2.6 HSV sylinder. Bilde er hentet fra Wikipedia.

Vinkelen rundt den sentrale vertikale aksen korresponderer til fargekanalen *hue* som fremstiller fargene i form av en vinkelen i en kontinuerlig sirkel. Vinkelen gis i grader fra [0,360] eller normalisert til [0,1], der rød er både ved 0° og 360° [0 og 1]. Distansen fra sentrale vertikale akse korresponderer til *saturation* og måler fargens fortynning av hvitt [76]. Ved normalisering er *saturation* mellom [0,1]. Anstanden langs sentrale aksen korresponderer til *value* som inneholder informasjon om gråtonen og spenner fra 0 (svart) til 1 (hvit).

Fordelen med dette rommet er ikke bare en uavhengig kanal for fargeintensitet, mer intuitiv beskrivelse av fargen enn ved RGB, men også en rask konvertering til og fra RGB som kunne gjøres med maskinvare i sanntid på 70 tallet. Ideen bak konverteringen er å bruke en RGB kube bestående av rødt, grønt og blått lys og videre vinkle kuben slik at svart plasseres på bunnen og hvit rett ovenfor langs vertikale akse. Denne RGB kuben prosjekteres på et plan, kalt for ”fargeplan” som ligger vinkelrett på den nøytrale aksen. Projeksjonen tar form av en sekskant med rød, gul, grønn, cyan, blå og magenta i sine hjørner vist i Figur 2.7. Videre måles *hue* av fargen i kuben ved vektorvinkelen til et punkt i projeksjonen rundt dets akse og starter på rød ved 0°. Matematisk er beregningen av *hue* vist i Likning 2.10.

$$H = \begin{cases} 0^\circ & \Delta = 0 \\ 60^\circ \times (\frac{G'-B'}{\Delta} mode 6) & , C_{max} = R' \\ 60^\circ \times (\frac{B'-R'}{\Delta} + 2) & , C_{max} = G' \\ 60^\circ \times (\frac{R'-G'}{\Delta} + 4) & , C_{max} = B' \end{cases} \tag{2.10}$$



Figur 2.7 RGB til hue konvertering der fargeplanet er vist i grått. Bilde er hentet fra Wikipedia.

Her betyr ' symbolet at fargen er normalisert til området [0,1]. C_{max} definert som høyest verdi av (R',G',B') og C_{min} defineres som minste verdi av (R',G',B'). Et alternativ til *saturation* er fargemetning eller "chroma" som brukt i A. R. Smith [72]. Forskjellen er at *saturation* målerer renheten i fargen eller ikke-hvitheten mens chroma er absolutt. Δ som kalles for chroma, er definert om $\Delta = C_{max} - C_{min}$. Ved nøytrale farger (C=0) settes *hui* til verdi 0°.

HSL modellen skalerer fargen så den alltid passer i området [0,1] for alle kombinasjoner av *value* og kaller den nye egenskapen *saturation*. For å beregne *saturation*, divideres chroma med *value*. Matematisk beskrivelse er vist i likning (2.11).

$$S = \begin{cases} 0 & ,\Delta = 0 \\ \frac{\Delta}{V} & ,ellers \end{cases} \tag{2.11}$$

Value er definert i likning (2.12).

$$V = C_{max} \tag{2.12}$$

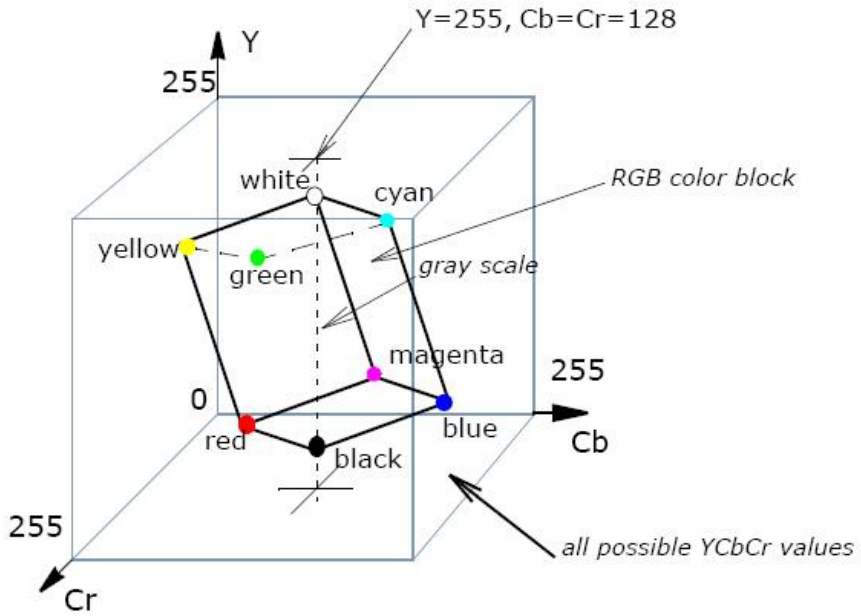
YCbCr fargerom

YCbCr fargerom ble utviklet som en del av ITU-R BT.601-5 standarten av ITU (International Telecommunication Union) [32]. Disse dokumentene definerer *YCbCr* som et fargerom for digital-tv. Fargerommet et populært for tv bruk fordi Y (luminans) komponenten kan

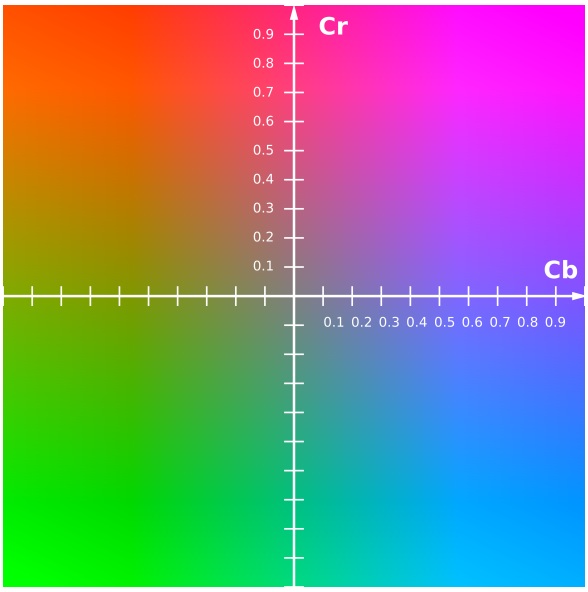
sendes med høy båndbredde, mens *CbCr* (blå og rød) krominans komponentene (delen av fargeinformasjonen som bestemmer fargen [84]) komprimeres siden det menneskelige øyet er mer sensitiv mot svart-hvit informasjon [60]. Dette fargerommet defineres av en matematisk koordinat transformasjon som er assosiert med RGB fargerom [60]. Matematisk er konvertering fra *RGB* til *YCbCr* vist i likning (2.13) [38]. Her er brukes $'$ symbolet til å vise at gamma korreksjon utføres.

$$\begin{bmatrix} Y' \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 65.481 & 128.553 & 24.966 \\ -37.797 & -74.203 & 112.0 \\ 112.0 & -93.786 & -18.214 \end{bmatrix} \begin{bmatrix} R' \\ G' \\ B' \end{bmatrix} + \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} \tag{2.13}$$

For at *YCbCr* fargerommet skal kunne representere alle fargeintensitetene til en *RGB* fargekube, der hver fargekanal er 8-bit, blir *Y* forskjøvet til ligge i et område mellom 16-235 og *CbCr* mellom 16-240. Dette resulterer i at *Y* kanalen har verdien 16 for svart og 235 for hvit. Dermed roteres og skaleres *RGB* fargekuben for å passe innenfor en større *YCbCr* fargekube, vist i Figur 2.8. Fargeplanet for $Y = 0.5$ er vist i Figur 2.9.



Figur 2.8 RGB fargekube i YCbCr fargerom [31].



Figur 2.9 *CbCr* planet ved konstant luminans Y'=0.5 [60].

2.4.2 Beregning av middelverdi av fargen

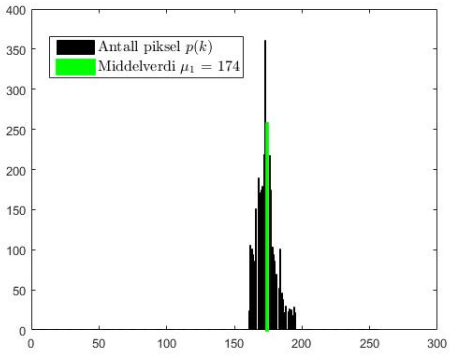
For å beregne middelverdien av fargen i pansersnittet, dannes først et fargehistogram for alle 3 fargekanalene. Fargehistogrammene viser fordelingen av ulike pikselverdier og gjør det derfor lett å beregne en middelverdi for fargen. Matematisk vises beregning av middelverdien i likning (2.14).

$$\mu_n = \frac{1}{T} \sum_{k=1}^{N} k \cdot p(k) \tag{2.14}$$

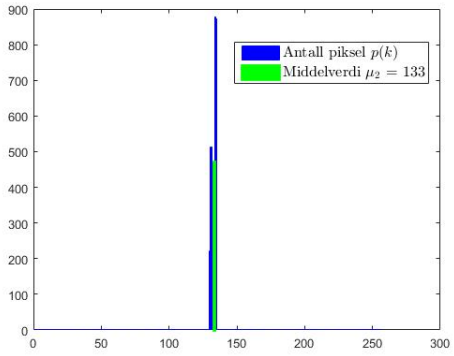
N står for antall mulige pikselverdier og har her verdi 256. T er totalt antall piksler i et bilde, d.v.s. summen av søylene i histogrammet. n er en bestemt fargekanal og μ_n er middelverdien for pikselintensiteten. Pikselverdiene er gitt i $k = 1,2,3...256$ og $p(k)$ er antallet piksler med den gitte verdien. Fargehistogram for Y, Cr og Cb som er dannet ut fra et panserutsnitt vises i Figur 2.10 sammen med middelverdien μ_n.

2.4.3 Avstandsmål for sammenligning av middelverdi

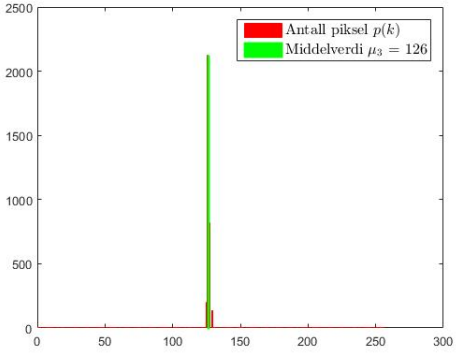
For å finne likheten mellom forskjellige middelverdier av panserutsnitt, kan det gjøres en beregning av avstand. Eksempel på avstandsmål er SAE (sum av absolutt avvik eller "error"), MAE (middel absolutt avvik), MSD (middel kvadrert eller "squarred" avvik) [3]. I oppgaven velges *middel absolutt avvik* eller på engelsk *mean-absolute error* med betegnelse d_{MAE}.



(a) Histogram av Y fargekanal



(b) Histogram av Cb fargekanal



(c) Histogram av Cr fargekanal

Figur 2.10 YCbCr fargehistogram dannet fra panserutsnittet vist i Figur 4.12

Denne metoden gir mer robusthet mot ekstremverdier enn metoder der kvadratrøtter beregnes. Avstanden er normalisert til [0,1] og matematisk er sammenlikning av fargemiddelverdier for et frontutsnitt par er gitt i likning (2.15) [7].

$$d_{MAE} = \frac{||\mu^{(1)} - \mu^{(2)}||}{N} = \frac{1}{N}\sum_{n=1}^{N} |\mu_n^{(1)} - \mu_n^{(2)}| \tag{2.15}$$

Her er $\mu_n^{(1)}$ middelverdien for frontutsnittet detektert i Kamera 1 og $\mu_n^{(2)}$ er for Kamera 2 ved fargekanal n. N er antallet fargerom og er 2 i de tilfellene en ønsker å separere vekk Y kanelen for å oppnå bedre robusthet mot lyseendringer.

2.5 Struktur-egenskap

Struktur er et vidt begrep og brukes her i forbinde med egenskaper som kan fortelle noe om strukturen i et bilde, ikke tekstur. Denne egenskapen kan være svært nytting der kjøretøy ofte har lik form og farge.

I de siste årene har mange artikler blitt skrevet som handler om gjenkjenning av kjøretøy i tunnel ved å se på egenskaper som forteller noe om strukturen. I en artikkel fra 2012 av R. Rios-Cabrera, et. al. [65], brukes Haar egenskaper til deteksjon, sporing og matching av kjøretøy. Haar består av firkant-formet funksjoner som regner ut gjennomsnittlig intensitet mellom forskjellige regioner [65]. Metoden har gitt gode resultater. I tillegg er lengden på egenskapsvektoren liten som reduserer kravet til båndbredde. Ulempen med denne metoden er at det kreves et stort antall bilder for å trene opp en klassifiserer for Haar egenskapene.

I 2013 ble en artikkel publisert av forskere fra Ghent universitet i Belgia [34] som brukte en algoritme av lav kompleksitet for å matche kjøretøy i tunnel. Signatur-vektorer, bestående av Radon transform-lignende projeksjoner, ble brukt til å representere et utseende av kjøretøyet. Metoden viste tilfredsstillende resultater ved gode forhold, men utfordringer kunne oppstå ved okklusjon og sterke lysrefleksjoner.

En ny forskningsartikkel ble publisert i 2015 [19], av flere fra den samme gruppen som tok for seg bruken av Radon transform til kjøretøy gjenkjenning i tunnel [34]. Her ble signaturer fra trace transformen brukt til å beregne en modellbasert beskrivelse av utseende til kjøretøyet. Fordelen med bruken av trace transformen er at den er mer robust mot belysningsstøy og geometriske transformasjoner. Forskningsartikkelen konkluderer at metoden gir svært gode resultater ved kjøretøygjenkjenning og utkonkurrerer flere algoritmer brukt i tidligere arbeid.

I denne oppgaven er det valgt å bruke Histogram of Oriented Gradients (HOG) metoden på et utsnitt av bilfronten. Metoden er valgt på grunn av sin gode robusthet mot lokale geometriske transformasjoner og lysendringer. Denne metoden ønskes det å teste opp mot trace transformen brukt av A. Frías-Velázquez, et. al. [19]. I artikkelen ble metoden brukt på hele kjøretøybilde og ikke på et bestemt område som ikke inkluderer bakgrunnsinformasjon. Resultatet av testene legges frem i Kapittel 5.5. I de neste underkapitlene forklares teorien bak HOG og trace transform.

2.5.1 Histogram of Oriented Gradients (HOG)

Histogram of Oriented Gradients (HOG) er en type egenskapsdeskriptor som beskriver retningen av gradienter eller kanter i lokale bildeområder. Deskriptoren brukes ofte til oppgaver som omhandler objektgjenkjenning [13]. HOG er relativt enkel å forstå sammenlignet med andre deskriptorer som for eksempel SIFT som ble foreslått av D. G. Lowe i [51]. Dette er

fordi HOG bruker globale egenskaper til å beskrive et objekt istedenfor lokale egenskaper. Det betyr at for eksempel at et kjøretøy beskrives med en enkel egenskapsvektor i motsetning til andre egenskapsdeskriptorer som tar for seg deler av kjøretøyet.

HOG deskriptoren ble først introdusert av Nevneet Dalal og Bill Triggs på en konfereanse om maskinsyn og mønstergjenkjenning, kalt CVPR i 2005, [13], med hovedfokus på deteksjon av mennesker. Hovedideen bak HOG var at form og utseende til lokale objekter i et bilde kan beskrives som en fordeling av lokale kant-retninger.

Før en egenskapsdetektor kan begynne å lete etter egenskaper, er det vanlig å utføre bildeforbedring for å fremheve ønskende detaljer. Denne delen kalles for pre-prosessering med formål her å normalisere fargen og gamma-verdier (luminans). I en artikkel av Dalal og Triggs [13], hvor deskriptoren ble brukt i menneskedeteksjon, er det konkludert at dette steget faller ut fordi normaliseringen gir beskjeden ytelsesforbedring. Dette er fordi etterfølgende normalisering av blokker (en større bildedel hvor et mål på intensiteten beregnes), oppnår like resultater. Derfor handler første steg om beregning av bildegradienten. Til dette finnes mange ulike type derivat-masker, også kalt filter-kjerner. I arbeidet til Dalal og Triggs [13], ble ulike typer masker testet for utregning som 3×3 Sobel maske, vist i Figur 3.6 og 2×2 diagonale masker. Best resultat ble oppnådd ved å bruke 1D-sentrert, punkt-diskret derivat-maske i både vertikal og horisontal retning. Filter-kjernene er gitt under i likning (2.16).

$$\begin{aligned} [-1,0,1] &\text{ - Beregner gradienten i horisontal retning} \\ [-1,0,1]^T &\text{ - Beregner gradienten i vertikal retning} \end{aligned} \tag{2.16}$$

Maskene konvolveres med bildet der responsene gir gradientens komponenter i horisontal og vertikal retning. Resultatet for operasjonen er vist i Figur 2.11b.

(a) Frontbilde av et kjøretøy. (b) Beregnet gradient av frontbilde.

Figur 2.11

Gradienten i bildet er en vektor med to komponenter, gitt i ligning (2.17), som beregnes ved bruk av maskene i (2.16). Tallverdien til gradienten er gitt av (2.18) og retningen er gitt av ligning (2.19). Det er tallverdien og retningen som brukes i HOG-deskriptoren.

$$I_x = \frac{\partial I}{\partial x} \qquad I_y = \frac{\partial I}{\partial y} \tag{2.17}$$

$$|\nabla I| = \sqrt{I_x^2(m,n) + I_y^2(m,n)} \tag{2.18}$$

$$\theta = arctan\frac{I_y}{I_x} \tag{2.19}$$

Neste steg involverer dannelsen av celle histogrammer. Selve cellene blir dannet ved å dele opp et bilde inn i et gitter, der hver celle i gitteret inneholder for eksempel 8×8 piksler. Her er det valgt å bruke celler med rektangulær form (mest vanlig), men har også mulighet til å være radielle.

Basert på verdiene beregnet i gradient-utregningen, gir hver piksel i sin celle en vektet stemme for en retningsbasertbasert histogram-kanal. Histogram-kanalene med informasjon om retningen, er jevnt spredd over 0 til 360 for "signert gradient" eller 0 til 180 grader for "usignert gradient", der her er sistnevnte valgt. Best antall histogram-kanaler ble funnet av Dalal og Triggs, [13], til å være 9 ut fra testing. Noe som gir 20° per kanal.

Når det gjelder vekten, kan bidraget for hver piksel velges ut fra en funksjon av tallverdien, kvadrert, kvadratroten eller en klippet versjon av tallverdien. I arbeidet til Dalal og Triggs, [13], gav tallverdien av magnituden det beste resultatet som gjør at sterkere gradienter har større effekt på histogrammet.

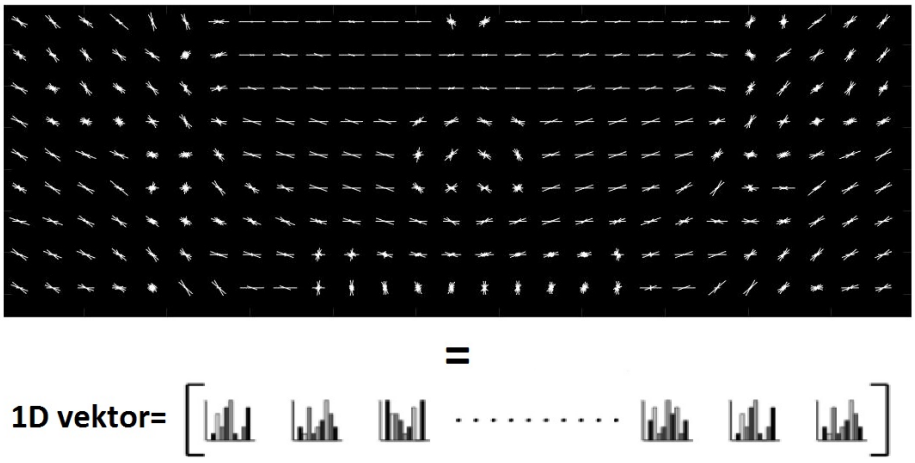
For å redusere effekten av lys og kontrastendringer, må histogrammene normaliseres. Istedenfor å normalisere hvert histogram individuelt, blir cellene gruppert i store overlappende blokker og normaliseres ut ifra alle histogrammene i blokken. Størrelsen på overlappingen pleier å bli satt til 50% overlapp. Dette resulterer i at hver celle bidrar mer enn en gang til den endelige deskriptoren. HOG deskriptoren blir så den sammensatte vektoren bestående av komponentene av normaliserte celle-histogrammer fra alle blokkområdene. Det finnes 2 typer av geometriske blokker; Rektangulære blokker (R-HOG) og sirkulære blokker (C-HOG). I oppgaven brukes R-HOG blokker som er firkantet gitter, representert av 3 parametre: antall histogram kanaler, antall piksler per celle og antall piksler per blokk. Her brukes 2×2 celler per blokk med 8×8 piksel per celler og 9 histogram kanaler siden dette gir generelt gode resultater. Oppdeling av bildet inn i blokker og celler er vist i Figur 2.12.

Figur 2.12 Her vises hvordan bildet deles opp i blokker og celler.

Neste del involverer blokk-normalisering som gjøres ved å lokalt normalisere styrken til gradienten. I Dalal og Triggs [13], ble 4 alternativer testet og den som gav best resultat er vist nedenfor i likning (2.20):

$$f = \frac{v}{\sqrt{||v||_2^2 + e^2}} \tag{2.20}$$

Her er v ikke-normalisert vektor som inneholder histogrammer i en gitt blokk, $||v||_k$ er dets k-norm for k=1,2 og e er en liten konstant. Visualisering av HOG deskriptoren er gitt i Figur 2.13.



Figur 2.13 Visualisering av HOG deskriptoren. De stjerneliknende objektene viser til gradientens retning i et lokalt bildeområde.

Endelige deskriptorstørrelse for et 64 $\times$ 128 bilde som deles opp i 7 blokker på langs og 15 blokker vertikalt (totalt 105 blokker), er 3 780 verdier. Denne verdien er fått ved 105 blokker $\times$ 4 celler per blokk $\times$ 9 kanaler per histogram.

Fordelene med HOG er at den fanger kant- eller gradient-strukturen som er veldig karakteristisk til en lokal form i et bilde. Metoden har i tillegg kontrollerbar grad av invarians mot lokale geometriske og fotometriske transformasjoner. Metoden er derimot ikke invariant mot globale transformasjoner.

2.5.2 Trace transform

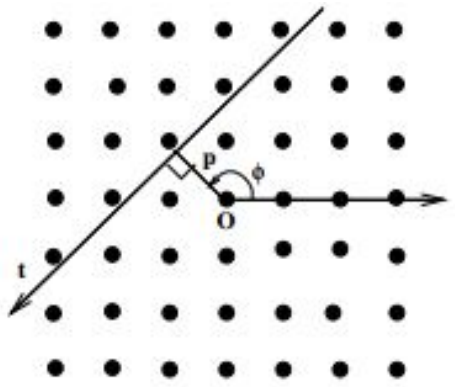
Trace transformen ble introdusert av A. Kadyrov og M. Petrou i 2001 [35], og er en radial projeksjons-transformasjon som gjør det mulig å konstruere bildeegenskaper som er invariant mot en valgt gruppe bildetransformasjoner [89]. Trace transformen kan sees på som en generalisering av Radon transformen som ofte brukes i forbindelse av tomografi. Prinsippet bak Radon er at 2D funksjoner kan bli fullt rekonstruert ut fra kunnskapen om dets integraler langs rette linjer. Trace transformen er lik med Radon transformen i den forstand at den også beregner funksjoner av et bilde langs linjer som på kryss og tvers krysser bildets domene. Forskjellen ligger i at Radon transformen beregner en bestemt funksjon, nemlig integralet.

Hovedmålet til trace transformen er å innhente bildedeskriptorer som *sinogram*, "*cirkus-*" eller *sirkel-funksjon* og *trippelfunksjon* til oppgaver som omhandler objektgjenkjenning. Slike deskriptorer blir beregnet i en kaskadeformet rekkefølge fra trace transformen der dimensjonaliteten reduseres for hvert steg i prosessen.

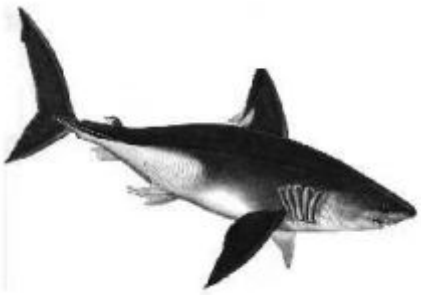
En trace transform deskriptor dannes først ved beregning av *sinogram*. Kort beskrevet så beregnes deskriptoren ved å anvende en funksjon kalt Trace- eller T-funksjon over snittlinjene i et bilde ved forskjellige orienteringer. Her blir bildet transformert fra kartesiske koordinater *(x,y)* til polare koordinater (ϕ, p). Figur 2.14 viser parametrene der t er snittlinjen, også kalt for *trace-linjen*. Her er ϕ $[0, 2\pi]$ og p er i området $[-p_{max}, p_{max}]$, der p_{max} ikke overstiger halve lengden av diagonalen til bildet.

T-funksjonene som beregnes over parameter t langs line (ϕ, p), er valgt med hensyn på å produsere egenskaper invariant mot bildetransformasjoner som rotasjon, translasjon, skalering. I A. Kadyrov, [36], presenteres 7 T-funksjoner vist i Tabell 2.1 som gir invarians når et bilde gjennomgår en affin transformasjon. Her er $f(t)$ signalet langs snittlinjen t. Hver av funksjonene beregner et *sinogram* $G(\phi, p)$ der resultatet vises i Figur 2.14c.

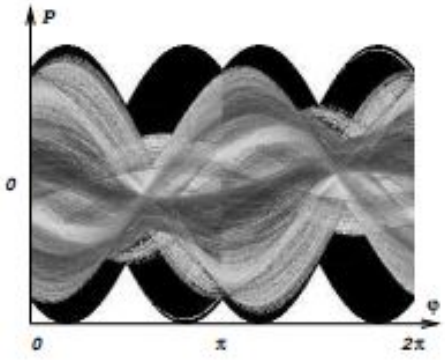
Videre gjøres en redusering av dimensjonen på trace deskriptoren gjennom bruk av diametral funksjon P. Funksjonene beregnes over radielle koordinater til et bilde ved forskjellige vinkel-orienteringer og returnerer en 1D-signatur, kalt *sirkelfunksjon*. Sagt på en annen måte så beregnes det en funksjon for hver kolonne av trace transformen, dvs. langs p retningen vist i Figur 2.14c, der resultatet danner en 1D-*sirkelfunksjon*. Navnet *sirkelfunksjon* er gitt grunnet til at "signaturen" som beregnes, er en funksjon med hensyn til parameter ϕ [36].



(a) Definering av parametre til en snittlinje i et bilde.



(b) Bildeobjekt.



(c) Anvending av T-funksjoner langs snittlinjer. Resultatet gir en *sinogram* deskriptor.

Figur 2.14 Bildene er hentet fra [35].

Tabell 2.1 Liste av T-funksjoner [89].

Nummer	Funksjon
1	$T(f(t)) = \int_{R+} rf(r)dr$
2	$T(f(t)) = \int_{R+} r^2 f(r)dr$
3	$T(f(t)) = \lvert \int_{R+} e^{i5log(r_1)} r_1 f(r_1)dr \rvert$
4	$T(f(t)) = \lvert \int_{R+} e^{i3log(r_1)} f(r_1)dr_1 \rvert$
5	$T(f(t)) = \lvert \int_{R+} e^{i4log(r_1)} r_1^{0.5} f(r_1)dr_1 \rvert$
6	$T(f(t)) = median_{tk} > 0\{\lvert r_1 f(r_1)\rvert, \lvert f(r_1)\rvert^{1/2}\}$
7	$T(f(t)) = median_{tk} > 0\{\lvert f(r)\rvert, \lvert f(r)\rvert^{1/2}\}$

$r = t - c$, $c = median(\{t_k\}_k, \{|f(t_k)|\}_k)$
$r_1 = t - c_1$, $c_1 = median(\{t_k\}_k, \{|f(t_k)|^{1/2}\}_k)$

Tre P-funksjoner gitt i A. F. Velazquez, et. al. [89], vises i Tabell 2.2 der *g(p)* presenterer en diametral "tracelinje" eller "snittlinje" av *sinogrammet* $G(p,\phi)$.

Tabell 2.2 Liste av P-funksjoner [89].

Nummer	Funksjon
1	$P(g(p)) = \sum_k \lvert g(p_{k+1}) - g(p_k)\rvert$
2	$P(g(p)) = median(\{g(p_k)\}_k, \{\lvert g(p_k)\rvert\}_k)$
3	$P(g(p)) = \int \lvert\mathcal{F}\{g(p)\}(\omega)\rvert^4 d\omega$

Ved å bruke alle 7 T-funksjonene vist i Tabell 2.1 og alle 3 P-funksjonene i Tabell 2.2, oppnår vi $7 \times 3 = 21$ *sirkelfunksjoner*. Det er mulig å redusere størrelsen på deskriptoren til bare en enkel verdi eller skalar, her kalt for *trippelegenskap* gjennom å bruke en annen funksjon, Φ-funksjon, over *sirkelfunksjonen* [35]. Dette steget blir ofte unngått siden *sirkelfunksjonen* gir en mer nøyaktig beskrivelse av objektet.

2.5.3 Matching av strukturbaserte-egenskapsvektorer

Populære avstandsmål for å måle likheten i en egenskapsvektor er normalisert-krysskorrelasjon (NCC), korrelasjons koeffisient, euklidsk distanse, sum av absolutt differanse (MAD), middel kvadrert distanse (MSD), Hausdorff distanse, osv... [53]. I oppgaven er NCC blitt valgt med begrunnelse gitt i Kapittel 4.4.7 og 4.4.8. Normalisert-krysskorrelasjon er vist nedenfor i likning (2.21):

$$\gamma(u,v) = \frac{\sum_{x,y}[f(x,y) - \overline{f}_{u,v}][t(x-u,y-v) - \bar{t}]}{\{\sum_{x,y}[f(x,y) - \overline{f}_{u,v}]^2 \sum_{x,y}[t(x-u,y-v) - t]^2\}^{0.5}} \tag{2.21}$$

Likningen er hentet fra arbeidet til J. P. Lewis [44], hvor f kan være et bilde eller en egenskapsvektor, t er malen eller "template" som det skal sammenlignes mot, $\overline{f}$ og $\bar{t}$ betyr middelverdien.

En stor verdi av $\gamma(u,v)$ betyr høy korrelasjon mellom vektor y og v der verdien varierer mellom [-1,1].

Kapittel 3

Deteksjon av skilt

I dette kapittelet forklares teorien bak skiltdeteksjon. Skiltposisjonen brukes som referanse for frontutsnittvinduet. Fra frontområdet skal strukturbaserte egenskaper detekteres.

3.1 Metode for skiltdeteksjon

Et bilskilt, også kalt kjennemerke er en plate med en kombinasjon av sifre og eventuelt bokstaver som tjener til identifikasjon av et kjøretøy [49]. Deteksjon av skilt kan gjøres ved automatisk skiltgjenkjenning (ANPR). Dette har i de siste årene vist seg å være en effektiv måte å registrere kjøretøy på, og har en rekke bruksområder. På en parkeringsplass kan for eksempel teknologien brukes til adgangskontroll eller til beregning av tiden en bil står parkert. Statens vegvesen har brukt teknologien til å finne kjøretøy med manglende EU-kontroll eller kjøretøy som er meldt stjålet [81]. Det var nettopp i forbindelse med sikkerhet at politiet i Storbritannia først oppfant teknologien i 1979 [69]. Med et stor utvalg av kunder fra politi til private virksomheter, har etterspørselen bidratt til et bredt utvalg av produkter på markedet.

Hovedmålet med skiltdeteksjonen i denne oppgaven er å bruke lokasjonen til skiltet som et referansepunkt for frontdeteksjon. Videre skal skiltfargen detekteres (hvit eller grønn) og brukes til å fortelle om kjøretøyet er en personbil eller varebil. Siden det tidligere har vært gjort mye arbeid innen dette feltet [14], har oppgavens fokus vært rettet mot andre problemstillinger knyttet til kjøretøygjenkjenning. En enkel algoritme som tar utgangspunkt i tidligere arbeid innen skiltgjenkjenning har derfor blitt valgt. Segmentering og tolkning av tegn er ikke en del av oppgaven, med begrunnelse forklart i Kapittel 1.2.1. Prosessen deles derfor inn i 3 ledd: Pre-prosessering (bildeforbedring), lokalisering og post-prosessering (validering).

Oppgaven har en liten utfordring som må taes hensyn til. Optiske kameraer som er beregnet for ANRP har mulighet for deteksjon av infrarødt lys. Dette gjør skilt lesbart på alle tider av døgnet. Samtidig taes skiltbildene i høy kvalitet da kameraet er fokusert på skiltområdet. I oppgaven må fokuset være slik at lastebiler kan detekteres i full størrelse. Skiltene som detekteres blir derfor av liten bildeoppløsning, men stor nok til å kunne leses. Dette må deteksjonsalgoritmen ta hensyn til. Det stilles også krav til at algoritmen er rask og robust for videopptaket den skal brukes på.

Flere teknikker for å detektere skilt har blitt utprøvd de siste årene. De mest anvendte metodene er formbaserte og leter etter kantene i bilskiltet [93] [28]. I Tabell 3.1 vises fordeler og ulemper for hver klasse av skiltdeteksjonsalgoritmer. Informasjonen er hentet fra Du et al. [14].

Tabell 3.1 Fordeler og ulemper med hver klasse av skiltdeteksjon algoritme.

Metode	Forklaring	Fordel	Ulempe
Bruke kantgrenseegenskaper.	Kantgrensen til et skilt er rektangulær.	Enklest og rask.	Vanskelig å anvende på komplekse bilder siden de er sensitive mot uønskede kanter.
Bruke globale egenskaper.	Finne komponenter som er sammenbundet, og som har like dimensjoner som et skilt.	Enkel og uavhengig av lokasjonen til skiltet.	Virker dårlig på bilder av lav kvalitet som kan resultere i forvrengte objekter.
Bruke teksturegenskaper.	Hyppige fargeoverganger ved skiltet.	Kan detektere selv om kanten er deformert.	Beregningsmessig kompleks når det er mange kanter.
Bruke fargeegenskaper.	Leter etter en bestemt farge ved skiltet.	Kan detektere bøyde og deformerte skilt.	Problem når kjøretøybakgrunn har samme farge som skiltet.
Bruke bokstav/tegn-egenskaper.	Det må være tegn på et skilt.	Robust mot rotasjon.	Tidskrevende å prosessere alle binære objekter. Kan produsere feil ved annen tekst i bildet.
Bruke en eller flere egenskaper.	Kombinering av egenskaper er mer effektivt.	Mer pålitelig.	Beregningsmessig kompleks.

Forslag til metode

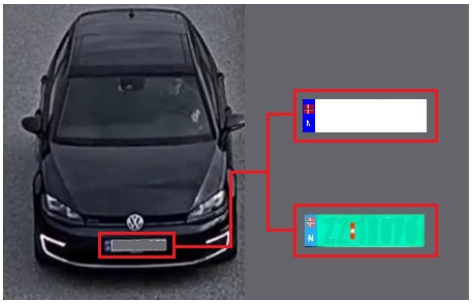
De fleste metodene brukt i tidligere arbeid er avhengig av en høy bildeoppløsning av bilskiltet. Dette medfører at metodene ikke er ideelle for denne oppgavens formål. Det er bestemt å implementere en metode som er inspirert av en kombinasjon av flere metoder fra tidligere arbeid. Resultatet er to metoder som brukes til å lokalisere skilt. Den mest pålitelige algoritmen utføres først. I tilfeller der den ikke detekterer et skilt, utføres en mindre pålitelig algoritme. Denne metoden har en bedre evne til å detektere skiltene ved lyse kjøretøybakgrunner. Metodene forklares nedenfor.

3.1.1 Skiltmetode 1

Under solrefleksjon eller ved lav bildeoppløsning er det vanskelig å detektere kantene i bilskilt-rammen. Derfor er det bestemt å bruke en metode som hovedsakelig baserer seg på fargeegenskaper. For deteksjon av hvite skilt tar metoden utgangspunkt i arbeidet til Zhai et al. [93], som har foreslått en løsning som bruker morfologiske åpne funksjoner til å eliminere

områder som ikke inneholder skilt. Denne metoden funger ikke på grønne skilt som har lav lysintensitet. Disse blir istedenfor detektert ved å segmentere grønne områder.

Algoritmen begrenser seg derfor til å detektere grønne og hvite skilt, illustrert i Figur 3.1. I Norge er disse de mest vanlige fargene, og det burde derfor ikke være et problem. Mangelen på testdata førte til at andre fager ikke ble inkludert. Om nødvendig, har algoritmen mulighet for utvidelse til å for eksempel detektere oransje skilt.



Figur 3.1 Skilt detekteres ved å lete etter lyse eller grønne områder.

Metoden forklares i større detalj i det følgende. For å lete etter hvite og grønne områder på kjøretøyet som kan inneholde skilt, har det blitt valgt å bruke HSV (Hue, Saturation og Value) fargerom. En nærmere forklaring av fargerommet er gitt i Kapittel 2.4. Fargerommet brukes fordi informasjon om intensiteten og fargen er adskilt i de ulike fargekanalene. Dette gjør at HSV er mer robust mot lysendringer enn RGB fargerom. En mulighet er også å bruke YCbCr fargerom da det også brukes til deteksjon av fargeegenskaper i oppgaven. Fordi fargerommet er gitt i to dimensjoner er det vanskeligere å definere hvilke verdier som svarer til grønt. Tidsbegrensinger la en stopper til å bruke dette fargerommet.

Deteksjon av grønne skilt gjøres ved å bruke *hue* fargekanalen i HSV fargerommet for å segmentere ut grønne blob-områder. Disse blob-områdene er representert som enere i et binært bilde og grupperes sammen til rektangulære bokser som kan inneholde et skilt. Hvite skilt detekteres ved å finne områder av kjøretøybildet som har høy pikselintensitet. Til dette brukes en *value* kanal. En utfordring ved å bruke fargebaserte algoritmer for å detektere hvite skilt er at kjøretøybakgrunnen ofte har samme farge som skiltet vi ønsker å detektere. Problemet vises i Figur 3.2.

Zhai et al. [93] har foreslått en løsning når kjøretøybakgrunnen har samme farge som skiltet. Denne løsningen skal vi benytte oss av. Her brukes morfologisk åpne funksjon med et rektangulært strukturelementer (SE) for å utjevne kjøretøybildet for bakgrunnsbelysning. Dette eliminerer områder som ikke inneholder et skilt. Morfologisk operasjon er en teknikk brukt til å finne samspillet mellom et bilde og et valgt strukturelement for å simplifisere

Figur 3.2 Hvit skilt på en lys kjøretøysbakgrunn (skiltet er sladdet).

og eliminere uønsket bildedata [25]. Det resulterende bildet kalles bakgrunn. I det neste steget utheves plateregionen ved å subtrahere bakgrunnsbildet fra det originale gråtonebildet (*value-kanal*). Resultatet av operasjonen er vist under i Figur 3.3.

(a) Binært kjøretøybilde av fronten til kjøretøyet fra Figur 3.2.

(b) Binært bilde etter at en morfologisk operasjon er utført.

Figur 3.3 Fra venstre: I dette bilde er det vanskelig å lokalisere skiltet fordi en stor del av kjøretøybildet er hvitt. I bildet til høyere er en morfologisk operasjon blitt utført for å filtrere lyse områder som ikke er en del av skiltet. Skiltet er representert i hvitt og er sladdet.

Etter at filtrering er utført, blir skiltkandidater detektert ved å gruppere hvite piksler til rektangulære markeringsbokser. Dette vises i Figur 3.4. For å bestemme hvilken av de grønne og hvite skiltkandidatene som faktisk er et skilt, brukes en valideringsmetode.

Figur 3.4 En validering må gjøres av skiltkandidatene vist i grønt.

Validering av skiltkandidater

Etter at potensielle kandidater er funnet, må falske skiltdeteksjoner fjernes ved å validere dem mot 3 forskjellige kriterier. Utgangspunktet for valideringsmetoden er hentet fra Duan et al. [15]:

- Forholdet mellom bredde og høyde
- Minimum og maksimum størrelse
- Antall objekter i horisontale kutt

Posisjonen til kjennemerker kan variere som eksempel for bilmerket Alfa Romeo hvor skiltet oftest er å finne litt til sides på kjøretøyet. Derfor brukes ikke posisjon til å evaluere potensielle kandidater. Nedenfor forklares valideringsmetodene i større detalj.

1. *Forholdet mellom bredde og høyde:*
For å fjerne potensielle feilkandidater, forkastes skiltkandidater som ikke oppfyller et bestemt forhold mellom bredden W_{LP} og høyden H_{LP} til et kjennemerke. Her står LP for ”licence plate”. Matematisk er det vist nedenfor i likning (3.1).

$$minWHForhold < \frac{W_{LP}}{H_{LP}} < maxWHForhold \tag{3.1}$$

De fleste kjennemerkene i Norge er rektangulære i formen med en bredde 52 cm og en høyde på 11 cm [49]. Firkantformede skilt eksisterer også, men finnes oftest på baksiden av motorkjøretøyet. Grunnet lite testdata av firkantede skilt, velges det kun å lete etter rektangulære skilt. Dette vil dermed øke påliteligheten til korrekt deteksjon av skilttypen som det finnes flest av. Selv om et vanlig forhold for $\frac{W_{LP}}{H_{LP}}$ er $\simeq$4.73 [49], blir tallet anderledes når deteksjonsprosessen fungerer i sanntid. Dette er på grunn av variasjoner skapt av forskjellige typer skiltrammer, eldre skilt som har et større flateareal og lyspåvirkninger som skaper refleksjon og skygge. Dette er noe som taes hensyn til ved bestemmelse av $\frac{W_{LP}}{H_{LP}}$ kriteriet. For grønne skilt blir et slakkere kriterie satt grunnet et større objektområde som detekteres på grunn av støy.

2. *Minimum og maksimum størrelse:*
I neste steg av valideringsprosessen blir skilt over og under en viss størrelse fjernet. Utrykk for størrelsesvalueringen er gitt i likning (3.2).

$$\begin{aligned} minH < H_{LP} < maksH \\ minW < W_{LP} < maksW \end{aligned} \tag{3.2}$$

3. *Antall objekter i horisontale kutt:*
For siste valideringstest brukes det en metode foreslått av Duan et al. [15]. Ideen er at symboler på et skilt danner mange vertikale kanter. En kant kan sees på som punkter i et bilde med sterke gradienter som er dannet ut fra skarpe forandringer i bildeintensiteten [21]. Siden antallet vertikale kanter er vesentlig høyere enn ved de fleste andre deler av fronten, kan en skiltkandidat velges basert på høyest antall. For å telle antall vertikale kanter, utføres 2 kutt nær senteret av kandidaten i horisontal retning som teller antall objekter. Middelverdien av kuttene regnes ut og kjennetegnskandidaten med flest objekter, velges som skilt. Antall objekter funnet må være over en satt grense for at et skilt skal bli detektert. Til deteksjon av kanter brukes et Sobel filter. Sobel filteret beregner en tilnærmet bildegradient for hver piksel ved å konvolvere bildet med et par av 3×3 filtere [73]. Denne operatoren gav et godt resultat ved testing og ble anbefalt av Duan et al. [15]. Bilde av operatoren er vist i Figur 3.5. Tabell 3.2 sammen med Figur 3.6 viser utførelsen av valideringsoperasjonen.

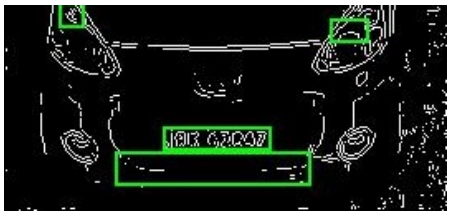
Figur 3.5 Sobel maske

-1	0	1
-2	0	2
-1	0	1

(a) Horisontal Sobel maske.

-1	-2	-1
0	0	0
1	2	1

(b) Vertikal Sobel maske.



Figur 3.6 Resultat av kantdeteksjon utført på en kjøretøyfront. Her er skiltkandidater som skal evalueres vist i grønt.

Skiltkandidat	Antall objekter	Resultat
	2.5	Ikke Skilt
	1.5	Ikke Skilt
	15	Skilt
	3	Ikke Skilt

Tabell 3.2 Validering av skiltkandidater skjer ved å telle antall objekter langs horisontale kutt.

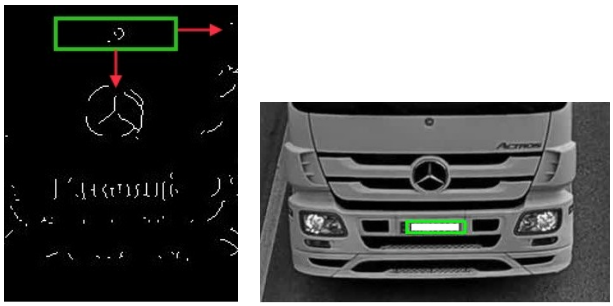
3.1.2 Skiltmetode 2

Skiltmetode 2 kan tenkes på som en plan ”B” som kjøres i tilfeller der metoden ovenfor ikke gir resultat. Denne metoden er aktuell i tilfeller der bakgrunnen har samme farge som skiltet og ble foreslått av Zheng et al. [95]. På grunn av tidsbegrensninger har implementeringen av algoritmen blitt ”dummet ned” der det har blitt unngått å implementere en foreslått metode for støyfjerning og en mer effektiv måte å søke etter skilt. Her har det også blitt valgt å

gjenbruke verifiseringsalgoritmen som ser etter horisontale kutt i objekter, til motsetning av metoden i arbeidet til Zheng et al.

Metoden i arbeidet til Zheng et al. [95] utnytter kunnskapen om at et skilt består av mange vertikale kanter. Dette er grunnet de mange tegnene og symbolene på skiltet (se Figur 3.7a). De vertikale gradientene detekteres ved å bruke en vertikal Sobel maske som konvolveres med kjøretøybilde. Videre blir kjennemerket detektert ved å skifte et rektangulært vindu med et flatemål på størrelse av forventet kjennemerke på tvers av kantbildet. Målet er å finne områder som inneholder flest horisontale kanter. I oppgaven er det blitt valgt å begrense leteområdet til senteret av kjøretøyfronten fordi lykter ofte kan ha mange kanter. Derfor har skiltmetode 2 en svakhet i få tilfellene der et skilt ikke er sentrert.

Hvis det totale antall kantpunkter i et vindu er over en terskelverdi, blir området vurdert som potensiell plateregion. Plateregionen som gir høyest respons av horisontale kutt i objekter, blir valgt som skilt. Videre beregnes det en middelverdi av fargen og intensiteten som brukes til å bestemme om skiltet er grønt, hvitt eller ingen av delene der hvor algoritmen vil vurdere skilt som ikke detektert. I Figur 3.7 vises en illustrasjon av søket etter skilt. Implementering av algoritmen er forklart i Kapittel 4.4.3.



(a) Leting etter flest horisontale kanter.

(b) Skiltet som er detektert.

Figur 3.7 Deteksjon av kjennetegn ved leting etter horisontale kanter.

Kapittel 4

Utstyr og Implementering

I dette kapittelet vil følgende deler av oppgaven bli lagt frem:

- Rammebetingelser for systemet
- Kamerautstyr brukt for opptak
- Programkode
- Implementering av metodene i systemet

Hovedfokuset i kapittelet er å forklare anvendt metode i de ulike leddene i systemet. Leddene består av kjøretøydeteksjon, skiltdeteksjon, kjøretøyklassifisering, fargedeteksjon, innhenting av egenskaper ved fronten og algoritme for kjøretøysammenligning. Det begrunnes hvorfor de ulike metodene i leddene er valgt.

Ved implementering av metodene har både egen- og ferdigalgoritme blitt anvendt. Her vil kapittelet ha fokus på egne løsninger som er implementert i de ulike leddene ved systemet. Det vil bli forklart hvordan ferdigalgoritmer anvendes og videreutvikles med ekstra funksjoner for å tilpasses systemet.

4.1 Rammebetingelser

Følgende rammebetingelser er utarbeidet for oppgaven:

- Autoeksponering er blitt slått av under opptak for å unngå raske endringer i lysintensiteten.
- Algoritmen er testet ut på vei med ett kjørefelt for å begrense innvirkningen av okklusjon og skygge.

- Bildestabilisering i kamera er slått på under opptak.
- Algoritmen er blitt testet på en rett veistrekning uten helning.
- Algoritmen er beregnet å fungere i dagslys og ved topp-ned perspektiv.
- WDR (Wide dynamice range) funksjonen er slått av under filmen da den skapte bildeforstyrrelser.

4.2 Kamerautstyr

Det er benyttet to ulike kamera for opptak av kjøretøy. Et for å simulere innkjørsel og et for utkjørsel. For å skape et skift i kontrast og perspektiv som oppstår i virkeligheten, har det blitt benyttet kamera med ulike innstillinger og posisjoner. I Figur 4.1 vises de ulike kameraene.

(a) Kamera 1: Nikon D7200
FOTO: What digital camera

(b) Kamera 2: Axis Q1615-E
FOTO: Axis Communication

Figur 4.1 Fra venstre: Kamera 1 simulerer opptak ved tunnelinngang og Kamera 2 simulerer opptak ved utgang.

Kamera 1, Nikon D7200 er et speilreflekskamera som ble lånt av Universitetet i Stavanger. Hensikten med dette kameraet er å ta opptak av kjøretøyobjektene fra et annet perspektiv for sammenligningsformål. For å spare tid, har det ikke blitt implementert kjøretøydetesjons-algoritmen på opptakene av dette kameraet. Det gjøres heller en manuell segmentering av kjøretøybildene fra opptakene. Kameraparametrene i dette kameraet har ikke vært i fokus.

Kamera 2, se Figur 4.1b, ble gått til innkjøp i forbindelse med oppgaven av Statens vegvesen sammen med et batteridrevet ethernet switch (Axis T8415) for drift av kamera.

Switchen fungerer som et trådløs kommunikasjonssystem som muliggjør styring av kamera fra et Android- eller Apple-basert operativsystem. Valget av dette kameraet forklares nærmere nedenfor.

4.2.1 Valg av kamera

Kjøpsvurderingens hovedkriterie er kameraets grensesnitt til å operere i et fremtidig system. Flere av potensielle kameratyper som finnes på marked i dag ble lagt til grunn i vurderingsprossessen. Alt fra web-, industri- til IP-/nettverks-kamera ble vurdert. Kravene som ble satt til kamera, tok utgangspunkt i en rapport publisert i 2005, [58], som omhandler krav til kameraegenskaper når maskinsynalgoritme i tunnel brukes. Egenskapene til det innkjøpte kameraet sidestilt med kravene som er gitt er vist i Tabell 4.1.

Tabell 4.1 Her vises kravene til kameraegenskapene som er satt.

Egenskap	Type	Krav	Valgt Kamera: Axis Q1615-E
oppløsning	optisk	$\geq$ SXGA	HDTV 1080p
progressive-scan	optisk	påbudt	ja
sensitivitet	optisk	$\geq$ 40 dB SNR med 10 lux, f1.4	farge: 0.36 lux, b/w: 0.08 lux, f1.3
bildefrekvens	optisk	25 bilder/s	25/30 bilde/s med WDR, 50/60 bilde/s uten WDR
dynamisk bredde	optisk	>60 dB	opptil 120 dB
montering	mekanisk	C-mount	C-mount
grensesnitt	mekanisk	PoE (Power over Ethernet)	PoE med støtte for vanlige nettverksprotokoller
fokal lengde	optisk	$\geq$ 12 mm for 2/3" bildesensor, horisontal visningsvinkel $\leqslant 41^\circ$	2.8-8 mm (avh. av zoom) for 1/2.8" bildesensor, horisontal visningsvinkel: 90-40° (avh. av zoom)

Selve opptakene ble gjort ved å plassere begge kamera på et kamerastativ (tripod) og filme i et topp-ned perspektiv over en gangbro. En forklaring av opptakene gjøres i Kapittel 5.1.

4.3 Programkode

Programmet Matlab er blitt valgt for testing av systemet. Dette programmet er enkelt i bruk for testing av algoritmer og inneholder stort bibliotek av ferdigfunksjoner innen bildebehandling. Bærebar datamaskin med en Intel Core i5-4200U prosessor på 1.6 GHz (4 kjerner) er brukt til testing. Videre i dette delkapittelet blir det en beskrivelse av hver enkel funksjon.

4.3.1 Åpen kildekode

1. **get_transform.m**
   Denne "skripten"/algoritmen beregner trace transform projeksjonen av et gråtonebilde i form av sinogram- og sirkelfunksjon-deskriptorer. Som inngangsparametre er det mulighet for å velge hvilke *T-* og *P-funksjoner* som skal brukes til å beregne deskriptoren, samt valg av vinkel-steglengde. Denne algoritmen konverterer i hovedsak bildet fra kartesiske koordinater (x,y) til polare koordinater (p,θ) og beregner funksjoner langs radielle akser. Denne algoritmen er blitt tilsendt av Andres Frias-Velazquez [20].

4.3.2 Kode fra MATLAB sine biblioteker

1. **detectFASTFeatures**
   Koden returnerer hjørnepunkter funnet ved FAST-detektoren.

2. **extractHOGFeatures**
   Koden returnerer HOG deskriptor av bildet.

3. **videotrafficgmm.m:**
   Denne koden er hentet fra Computer Vision System Toolbox i Matlab. I oppgaven brukes en modifisert utgave av koden til å utføre bakgrunnssubtraksjon ved GMM til å detektere kjøretøy. Egne funksjoner blitt implementert for å fjerne deteksjoner utenfor ønsket område, verifisere kjøretøy og fjerning av skygge. Parametrene er tilpasset opptaket forklart i Kapittel 5.1.1.

4. **multiObjectTracking.m**
   I denne funksjonen kjøres et skript som tracker objekter ved bruk av Kalmanfilter.

5. **vision.GammaCorrection**
   Denne funksjonen er hentet fra Computer Vision System Toolbox i Matlab og brukes til å utføre gammakorreksjon på kjøretøybildet.

4.3.3 Egentilpasset kode

1. **vehicle_detection.m**
   Denne filen kjører hovedprogrammet som kaller andre funksjoner. Hovedprogrammet er en videreutvikling av funksjonen *multiObjectTracking.m* fra Matlab sitt bibliotek som har blitt utvidet med flere egne funksjoner. Dette inkluderer bl.a. kjøretøyklassifisering, frontsegmentering, henting og matching av egenskaper når et kjøretøyobjekt krysser et forhåndsdefinert område.

2. **LP_detection.m**
   Denne funksjonen returnerer skiltposisjonen sammen med informasjon om hvit eller grønn skiltfarge. Som inngang krever funksjonen bl.a. et RGB bilde av kjøretøyobjektet. Implementering av koden er forklart i Kapittel 4.4.3.

3. **color_detect.m**
   Et skript som detekterer panseret og returnerer middelverdien for fargen. Som inngang krever skriptet et RGB bilde av kjøretøyet og informasjon om hjørnepunktene. Implementering av koden er forklart i Kapittel 4.4.5.

4.4 Implementering av metode

I dette delkapittelet beskrives implementeringen av algoritmen til systemet som skal brukes i to kamera på hver sin ende av tunnelen. Begge skal kjøre samme algoritme for deteksjon av kjøretøy og i tillegg skal kameraet ved tunnelutgangen også kunne matche egenskapsvektorene for å kjenne igjen kjøretøyene. Flyteskjema for systemet er vist i Figur 1.7 og en oversikt over hvordan dette delkapittelet er bygd opp forklares nedenfor.

Kamera 1: Tunnel inngang

1. Kjøretøydeteksjon
2. Egenskapdetektering

Kamera 2: Tunnel utgang

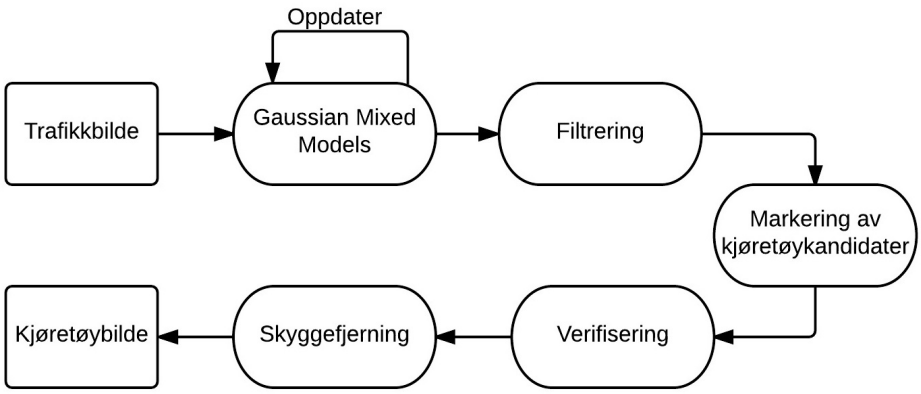
1. Kjøretøydeteksjon
2. Egenskapdetektering
3. Matching

Oversikt over hvor implementering av de ulike delene av algoritmen er beskrevet.

1. Kjøretøydeteksjon
   (a) Bakgrunnssubtraksjon og skyggefjerning: *Kapittel 4.4.1*
   (b) Tracking: *Kapittel 4.4.2*
2. Egenskapsdetektering
   (a) Skiltdeteksjon: *Kapittel 4.4.3*
   (b) Formbasert metode: *Kapittel 4.4.4*
      - Kjøretøyklassifisering
   (c) Fargebasert metode: *Kapittel 4.4.5*
      - Innhenting av panserutsnitt
      - Beregning av middelverdi av fargen
   (d) Strukturbasert metode:
      - Deteksjon av frontutsnitt: *Kapittel 4.4.6*
      - HOG: *Kapittel 4.4.7*
      - Trace transform: *Kapittel 4.4.8*
3. Matching
   (a) Kjøretøysammenligning *Kapittel 4.5*

4.4.1 Implementering av bakgrunnssubtraksjon

Bakgrunnssubtraksjon har som mål å detektere kjøretøyobjekter i et trafikkbilde. I oppgaven er GMM metoden valgt for bakgrunnssubtraksjon og teorien bak metoden er beskrevet i Kapittel 2.1.1. Algoritmen for GMM er en ferdigfunksjon i Matlab som har blitt kalibrert mot opptaket i Kapittel 5.1.1. Videre har egne funksjoner blitt implementert i algoritmen som fjerning av deteksjoner utenfor ønsket område, verifisering av kjøretøy og fjerning av skygge. Implementeringen av prosessen som handler om bakgrunnssubtraksjon er vist som blokkskjema i Figur 4.2. En detaljert forklaring av blokkene er gitt under.

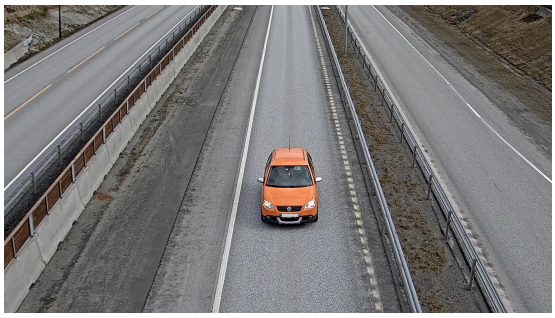


Figur 4.2 Blokkdiagram for utførelsen av bakgrunnssubtraksjon

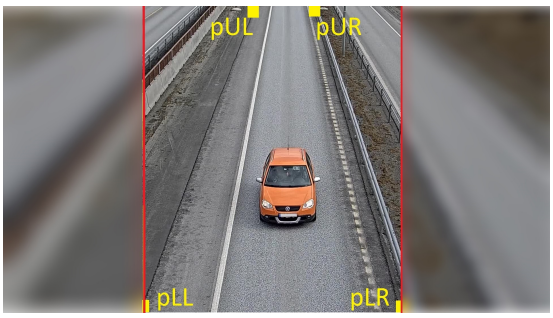
Trafikkbilde

Ved tid t hentes det ut et trafikkbilde fra opptaket. Fargerommet som er valgt for bildet er RGB fordi GMM ferdigfunksjonen som brukes i Matlab, krever det. Tidsbegrensninger gjorde det ikke mulig å endre funksjonen til å bruke YCbCr fargerom, som hadde vært ønskelig siden det er konkludert med at dette fargerommet gir best støyhåndtering ved bruk i GMM på grunn av numerisk stabilitet og uavhengig lysstyrke-kanal [40] [64]. En annen grunn til å bruke dette fargerommet er at det brukes ved fargesammenligning. Dette er forklart i Kapittel 4.4.5.

Etter at et trafikkbilde er hentet ut, beskjæres deler av bilderammen som ikke kan inneholde kjøretøy. Dette reduserer leteområdet til algoritmen. Beskjæring er ofte en funksjon som programmeres ved selve kalibrering av kameraet. Her er denne funksjonen blitt implementert i selve algoritmen i systemet. Beskjæringen resulterer i et rektangulært bilde vist i Figur 4.3b.



(a) Trafikkbilde.

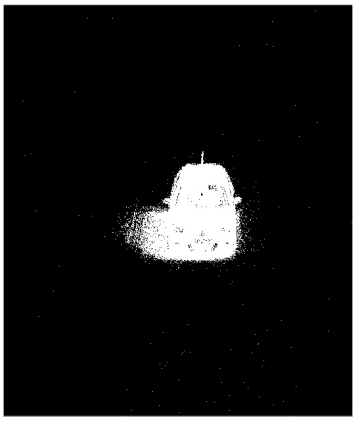


(b) Bildepunkter i gult velges ved kalibrering for å beskjære unødvendig bildeområde (glattet område) og til å fjerne deteksjoner utenfor kjørebanen.

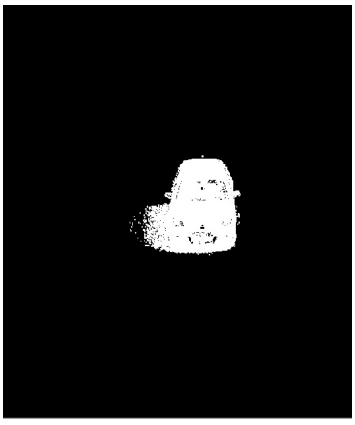
Figur 4.3

Gaussian Mixed Models

Her utføres bakgrunnssubtraksjon ved GMM og forgrunn detekteres. Parametrene til GMM er funnet ved testing og er forklart i Kapittel 5.3.1. Det binære bilde etter bakgrunnssubtraksjon er vist i Figur 4.4a.



(a) Forgrunnen i hvitt er detektert ved GMM bakgrunnssubtraksjon.



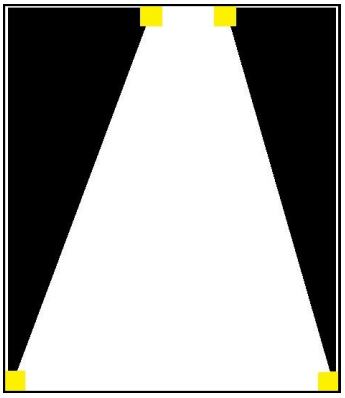
(b) Forgrunnen etter filtrering.

Figur 4.4 Forgrunn som detekteres ved bakgrunnssubtraksjon.

Filtrering

Små kameravibrasjoner og lyseendring kan forårsake salt og pepper støy i forgrunnsbilde. Disse blir fjernet ved å bruke morfologisk åpene operasjon med rektangulær 3×3 strukturelement. GMM algoritmen i Matlab sitt bibliotek er mest effektiv når den danner en

bakgrunnsmodell for hele bildet. Derfor kan piksler utenfor interesseområde bli feilaktig klassifisert som forgrunn. Disse deteksjonene blir fjernet ved å bruke masken vist i Figur 4.5. Resultat av filtrering er vist i Figur 4.4b.



Figur 4.5 Maskens formål er å fjerne deteksjoner utenfor interesseområde (området i svart). Masken initialiseres ved å utføre interpolasjon mellom punktene vist i gult. Punktene velges ved kalibrering.

Markering av kjøretøykandidater

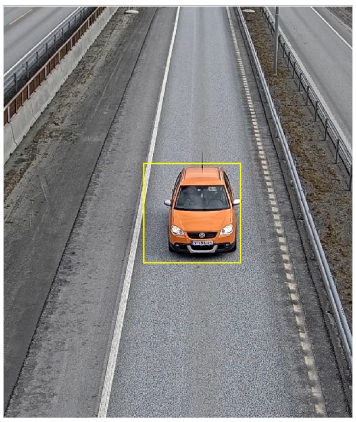
Her grupperes nærliggende forgrunnspiksler inn i rektangulære markeringsbokser. Dette gjøres ved å bruke Matlab-funksjonen *blobAnalyser*. Markeringsboksene kan sees på som kjøretøykandidater. Minimumsgrensen for antall piksler blobben må inneholde før den kan grupperes, er satt til 10 000 piksler i oppgaven. I Figur 4.6a vises en kjøretøykandidat i gul markeringsramme.

Verifisering av kjøretøy:

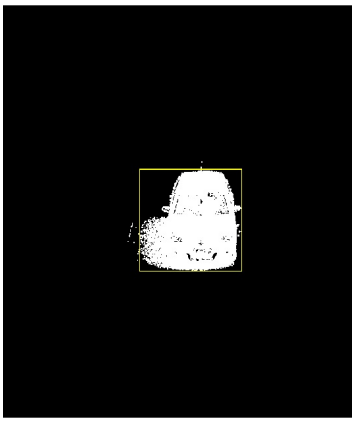
Bakgrunnssubtraksjon detekterer bare mulige områder et kjøretøy kan befinne seg i. For å finne selve kjøretøyet, må hver markeringsboks (kjøretøykandidat) k evalueres og eventuelle feildetekteringer fjernes. Foruten å ha tilgang til et stort treningsett med data for å trene opp en klassifiserer for kjøretøyevaluering, er det valgt å bruke den enkleste metoden. Falske kandidater skilles vekk ut fra følgende kriteriene:

1. Fjerner kandidater som har for stor høyde H_k i forhold til bredde W_k og motsatt. Grendseverdiene t_{min} er satt til 0.333 og t_{max} til 3.

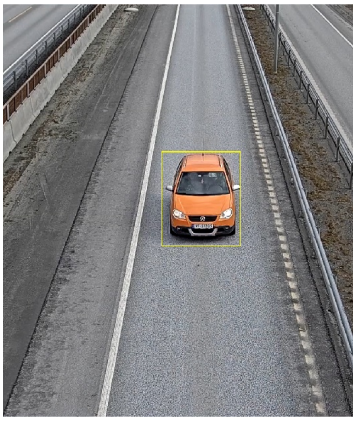
$$\begin{cases} BEHOLD & t_{min} < \dfrac{H_k}{W_k} > t_{max} \\ FJERN & ellers \end{cases} \tag{4.1}$$



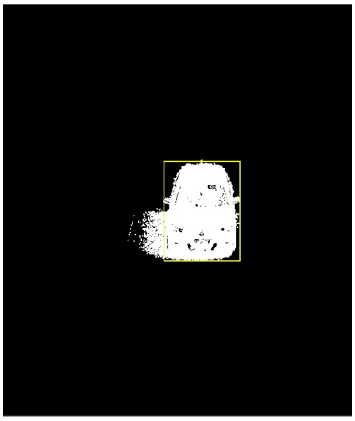
(a) Gul markeringsramme viser en kjøretøykandidat.



(b) Markeringsramme rundt blob området.



(c) Markeringsrammen er redusert etter skyggefjerning.



(d) Markeringsramme etter skyggefjerning.

Figur 4.6 Detektert kjøretøykandidat før og etter skyggefjerning.

2. Fjerner barkeringsbokser som har en størrelse under en satt terskelgrense $t_{size}(y)$. Denne terskelgrensen er avhengig av boksens posisjon langs y-aksen og kan ha 8 mulige verdier. I tilfelle en deteksjon er langt fra kamera og befinner seg øverst i bilderammen, brukes en lav terskelverdi t_{size}. Denne parameteren må kalibreres når systemet settes opp.

$$\begin{cases} BEHOLD & k_{size} > t_{size}(y) \\ FJERN & ellers \end{cases} \tag{4.2}$$

Store deteksjonsobjekter blir ikke fjernet da for eksempel et vogntog ofte bruker opp store deler av bilderammen.

Fjerning av skygge ved bruk av hjørnedeteksjon

Skygge som følger kjøretøyet er et problem fordi det vil bli en del av forgrunnsmasken (kjøretøyobjektet) ved en subtraksjon av bakgrunnen. Derfor har en løsning vært et tema i flere artikler [46] [55] [27] [57]. Det er blitt foreslått en løsning for fjerning av skygger fra kjøretøy som er detektert ved bruk av GMM i arbeidet til H. Asaidi, et. al. [1]. I oppgaven er derimot en egen løsning for fjerning av skyggen valgt som tar i bruk hjørnedeteksjon. Metoden er enkel og lite ressurskrevende.

Det er slik at et kjøretøyobjekt består av mange forskjellige strukturer der omfanget av hjørner er stort. En vei er en jevn flate og skal ideelt ikke inneholde hjørnerpunkter. Dette utnyttes i metoden ved å fjerne deler av kjøretøybildet som ikke inneholder hjørner. For å redusere sannsynligheten at hjørner detekteres i deler av kjøretøybilde som inneholder en veibane, glattes bilde først ved å bruke gaussisk lavpassfilter med $\sigma = 3$. For deteksjon av hjørner er FAST (Feature from Accelerated Segment Test) hjørnedetektor valgt på grunn av sin gode ytelse [66] [67]. Testing av andre hjørnedetektorer er ikke prioritert her der FAST gjør en tilfredsstillende jobb.

Det er ikke alltid alle hjørnepunkter på veibanen blir glattet vekk, og i noen tilfeller vil disse detekteres på en avstand fra selve kjøretøyet. Vi ønsker å fjerne disse hjørnene. Dette gjøres ved å beregne det som kalles *massesenter* som beskriver hvor i bildet konsentrasjonen av hjørnepunkter langs x-aksen (i horisontal retning) er. *Massesenter* finnes ut fra snittverdien μ_x for hjørne-koordinatene langs x-aksen. Hjørnepunkter $C(x_c, y_c)$ med koordinatverdi for x-aksen som er over en viss avstand fra massesenteret fjernes.

$$\begin{cases} FJERN & |x_c - \mu_x| > t_c \\ BEHOLD & ellers \end{cases} \tag{4.3}$$

Her er t_c satt til 150 pikselavstand for datasettet brukt i Kapittel 5.1.1. Denne parameteren må kalibreres ved endring av kameraposisjon. I neste steg er det ønskelig å fjerne områder i kjøretøybilde som inneholder skygge og bakgrunn. Dette gjøres ved å se om de mest venstre- og høyre-liggende hjørnene har en stor avstand til bilderammen. Hvis det er tilfellet, reduseres bilderammen til nærmeste hjørne. Dette er illustrert i figur 4.7.

Bilde kuttes bare hvis forholdet mellom lengden L (bilderammen til nærmeste hjørne) og totale bredden til bilderammen TL er under en viss terskelverdi t_{cut}. Dvs. $TL/L < t_{cut}$ der t_{cut} er satt til 8. Dette for å unngå å beskjære bildet når lengden L er liten. Samme operasjon

Figur 4.7 Fjerning av skygge: De to røde prikken er hjørnene som ligger nærmest høyre og venstre bilderamme. Grønn firkant viser til *massesenteret*. Lengde *L* avgjør hvor mye av bildet som skal kuttes. I oppgaven har en valgt å kutte *0.7*L* for å unngå tilfeller der deler av sidelyktene blir feilaktig kuttet.

utføres på begge sider av kjøretøyet. Metoden er også ment å virke under begrenset rotering av kjøretøyet i forhold til kameraet.

En annen utfordring ved bakgrunnssubtraksjon, er lysrefleksjoner dannet av frontlykt langs en vei. Disse kan bli feilaktig en del av forgrunnen. Dette viste seg ikke å være et problem ved opptakene beskrevet i Kapittel 5.1, der opptakene ble gjort på dagtid. Siden denne problematikken ikke har dukket opp i testdataene, har det i oppgaven blitt utelatt å prøve å finne en løsning for problemstillingen. Et forslag til løsning er derimot gjort i L. Unzueta, et. al. [87]. Resultat av bakgrunnssubtraksjonen er vist i Figur 4.8, hvor et kjøretøybilde detekteres.

Figur 4.8 Her vises resultatet av bakgrunnssubtraksjon.

4.4.2 Implementering av tracking

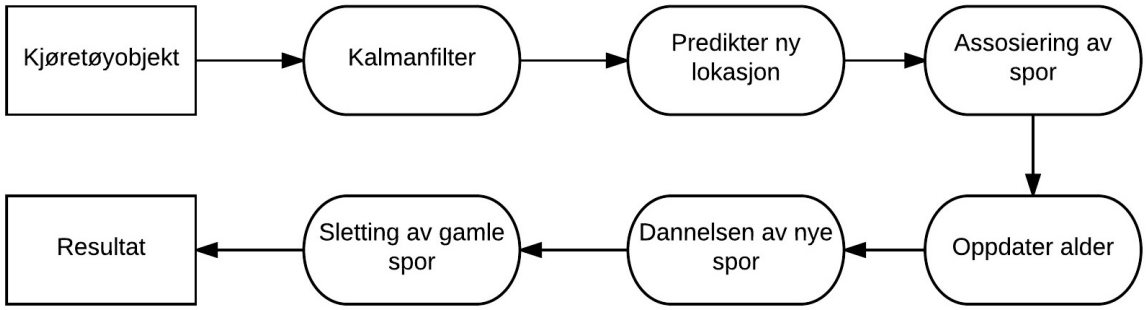
Teorien bak tracking og en forklaring av ulike trackingteknikker er gitt i Kapittel 2.1.2. Trackingteknikkene kan deles inn som følgende:

- Modellbasert tracking
- Områdebasert tracking
- Aktiv-konturbasert tracking
- Egenskapbasert tracking

Formålet med å tracke er å assosiere et kjøretøyobjekt (blob-objekt) med en identitet som opprettholdes så lenge kjøretøyet er synlig i videosekvensen. Dette kan enklest oppnås ved å bruke regionbasert tracking. Problematikken med okklusjon er i seg selv et hinder som er i hovedsak er knyttet til bakgrunnssubtraksjon der flere kjøretøyobjekter kan bli smeltet sammen til en. En egenskapsbasert trackingmetode vil ikke kunne løse denne problematikken. Derfor velges en enkel områdebasert trackingmetode og en løsning for okklusjonsproblematikken kan heller gjøres med algoritmer knyttet til bakgrunnssubtraksjon. En mulig løsning for okklusjonsproblemet er foreslått av W. Zhang, et. al., [94], der sammensmeltede kjøretøyobjekter isoleres fra hverandre. Tidsbegrensing har ikke gjort det mulig å implementere denne algoritmen i systemet. Videopptakene som er gjort i forbindelse med oppgaven, foregår på en rett veistrekning og derfor vil en metode som bruker bevegelsesinformasjonen til å predikere posisjonen av kjøretøyet i form av et Kalmanfilter, være optimal. Teori om Kalmanfiltret kan leses om i [37].

Algoritmen for sporing er implementert ved å bruke en modifisert utgave av en ferdigalgoritme fra Matlab. Endringene er gjort for å tilpasse koden til formålene i oppgaven. En slik endring har vært å utvide id-vektoren til å huske kjøretøyets posisjon i tidligere bilderammer. Informasjonen om posisjon i tidligere bilderammer benyttes til å validere om objektet er et mulig kjøretøy. Et blokkdiagram av Matlab koden for tracking er vist Figur 4.9.

Hver markeringsboks som er detektert ved GMM, blir assosiert med et spor (id-vektor) som kan tenkes på som en identitet. Assosieringen av spor til samme objekt er kun basert på bevegelsesinformasjon. Bevegelsen til hvert spor blir estimert med et Kalmanfilter som predikerer posisjonen til et spor i hver bilderamme og bruker sannsynlighetsregning for å assosiere sporene til kjøretøyobjektene. Vedlikehold av spor gjøres ved å oppdatere alderen til sporene for hver etterfølgende bilderamme. Ikke alle detekterte markeringsbokser blir assosiert med et spor. En deteksjon som ikke er tilordnet et spor, starter et nytt ett. Sporene som ikke er blitt assosiert med en deteksjon/objekt, antas å ha forlatt synsfeltet og derfor



Figur 4.9 Blokkdiagram for tracking av kjøretøy.

slettes etter en satt alder. Antall bilderammer et spor er tillatt å være ”usynlig” er satt i oppgaven til 5.

I oppgaven er id-vektoren eller ”sporet” som assosieres med kjøretøyet når det spores, blitt endret til å inneholde informasjon om deteksjon av egenskaper ved kjøretøyet er blitt utført. Beregning av egenskaper utføres når kjøretøyet krysser et gitt deteksjonsområde som her kalles ”greenZone”, vist i Figur 1.5. For at ikke en egenskapsdetektering skal utføres på feildeteksjoner, er det blitt dannet en sikkerhetssone, kalt ”redZone” som må passeres for at kjøretøyet skal bli verifisert. Dette området plasseres like før ”greenZone” og er vist i Figur 5.6. Minst 3 bilder av kjøretøyet må detekteres i ”redZone” før det godkjennes for en videre egenskapsdetektering.

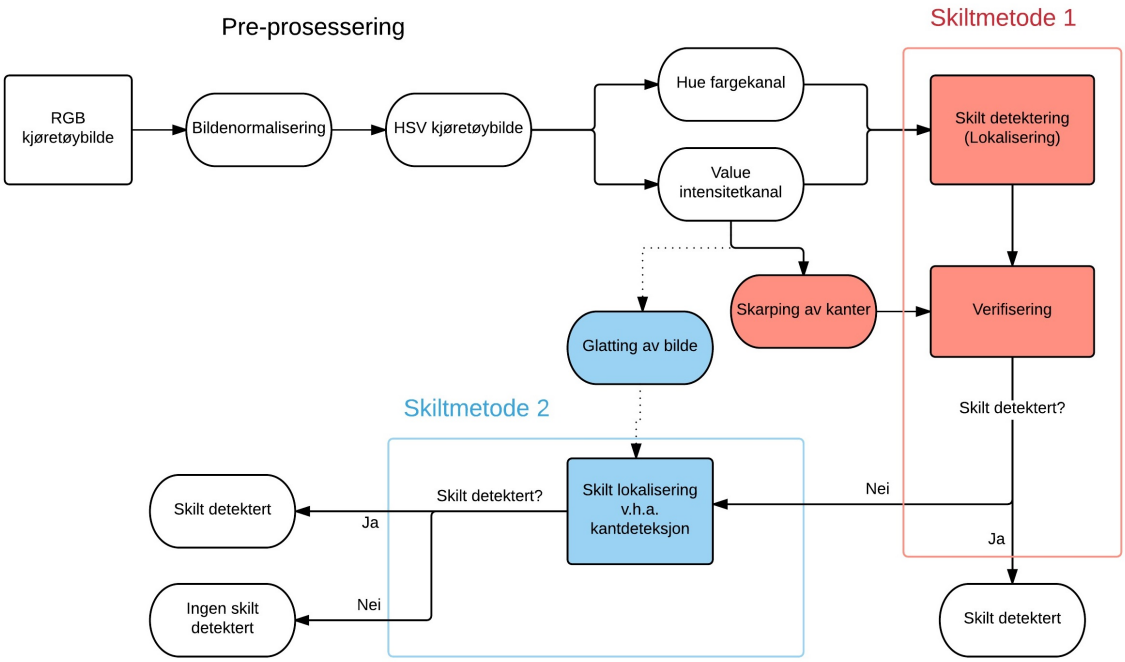
4.4.3 Implementering av skiltdeteksjon

Implementering av skiltdeteksjon er basert fra egen Matlab algoritme. Nedenfor i Figur 4.10 vises et blokkdiagram av operasjonen. Skiltdeteksjonsalgoritmen er delt i to metoder. Skiltmetode 1 skal fungere primært, mens skiltmetode 2 skal kjøres i de tilfellene første metode ikke detekterer skilt. Forklaring av metodene gjøres i Kapittel 3.1. Skiltdeteksjon utføres når et kjøretøy krysser deteksjonsområdet vist i Figur 1.5.

Grunnet at skiltdeteksjon ikke har vært fokus i oppgaven, er algoritmen for skiltdeteksjon kun blitt anvendt for opptaket forklart i Kapittel 5.1.1, der begge skiltmetodene detekterer alle bilskiltene ved de 15 kjøretøyene. Resultatet av deteksjonen er vist i Figur A.8. Nedenfor forklares noen av stegene i algoritmen for skiltdeteksjon.

Bildenormalisering

Metoden leter etter skilt kun i nedre $1/3$-del av kjøretøybildet. Dette reduserer søkeområdet og minker kjøretiden til metoden. Før en videre detektering av skilt kan utføres, må rotasjonen i bildet normaliseres. Dette er viktig grunnet til geometriske transformasjoner av skiltet som



Figur 4.10 Blokkdiagram for skiltdeteksjon.

oppstår ved forskjellig kameravinkel i forhold til veiplanet. Ved kalibrering av systemet, må derfor en vinkel α settes som bestemmer hvor mye kjøretøybildet skal roteres rundt x-aksen. Rotasjonsmatrisen er vist i likning (4.4). I oppgaven har vinkel α i rotasjonsmatrisen blitt satt lik θ. Vinkel θ er forklart i Kapittel 5.1.1.

$$R = \begin{bmatrix} cos\alpha & -sin\alpha \\ sin\alpha & cos\alpha \end{bmatrix} \tag{4.4}$$

Grunnet lysendringer og skygger fra omgivelsene, kan skiltområdet på et kjøretøybildet bli mørkelagt. Dette kan spesielt skape utfordringer ved deteksjon av grønne skilt. Derfor må det utføres en form for normalisering av bildekontrasten. Til dette brukes en gammakorreksjon-funksjon i Matlab sitt Vision biblioteket til å øke kontrastnivået. Gammaverdien ble kalibrert mot opptaket beskrevet i Kapittel 5.1.1 og satt til 1.5.

Skiltmetode 1

Strukturelementet som brukes i morfologisk åpne operasjoner til å fjerne bakgrunnsinformasjon fra kjøretøybildet, ble foreslått i X. Zhai, et. al. [93], å være på størrelse med skiltet,

men testing gav bedre resultater ved å sette en mindre verdi. Et rektangulært element med størrelse 11×20 ble derfor valgt.

En verifiseringsalgoritme bestående av 3 ledd, brukes til å skille ut falske skiltkandidater:

1. Første ledd sjekker kandidaten mot et bredde-høyde forhold kriterium forklart i likning (3.1). Forholdet $minWHForhold$ er satt til 3 og $maxWHForhold$ er satt til 6.5 ut ifra testing.
2. Neste ledd fjerner kandidater over og under en satt størrelse vist i likning (3.2). Denne størrelsen er bestemt ut fra forventet størrelse til skiltet og må settes ved kalibrering av kamera.
3. Siste verifiseringsledd sjekker antall vertikale kantobjekter i horisontale kutt. Her er det viktig at kantene er tydelige. Derfor utføres en "spissing" eller "styrking" av skarpe kanter. Teknikken som er valgt til å oppnå dette, finner differansen av bildet med en uskarp versjon for å øke kontrasten langs kanter hvor ulike farger møtes. For grønne skilt er dette spesielt viktig der en overgang fra grønn til svart (bakgrunn til tegn) ikke vil skape like stor gradient som ved overgang fra hvit til svart. Siden det allerede utføres gamma korreksjon, er det ikke behov å bruke histogramutjevning. Kantdeteksjon utføres ved bruk av Sobel filter vist i Figur 3.5. Et minimumskrav til antall objekter er satt til 7 før kandidaten kan vurderes som skilt.

Skiltmetode 2

Metoden flytter et vindu på langs og tvers av et kantbilde. Bildet er dannet ved å konvolvere gråtonebildet med en horisontal Sobel maske. Før kantdeteksjonen utføres, er det viktig å glatte bildet for støy. Til dette brukes et gaussisk glattefilter med $\sigma = 1.3$. Størrelsen på søkervinduet er bestemt ut fra en middelverdi for observerte skilt fra datasettet forklart i Kapittel 5.1.1. Denne verdien forandres ved kalibrering av kamera mot andre opptak. Søking etter skilt foregår ved senteret av kjøretøyfronten som er definert ved "*massesenteret*". For forklaring av begrepet, se Figur 4.7. For områder der antall kantpiksler overstiger en satt verdi, her 55, utføres det telling av antall objekter i horisontale kutt. Søkevinduet der flest objekter er detektert, velges til skiltområdet.

4.4.4 Implementering av kjøretøyklassifisering

Kjøretøy klassifisering gjøres enkelt ved å bruke størrelsen på markeringsboksen k_s som detekteres ved bakgrunnsubtraksjon til å klassifisere et kjøretøy til *lett* eller *tung*.

$$\begin{cases} Lett & k_s < t_s \\ Tung & ellers \end{cases} \tag{4.5}$$

Grenseverdien er kalibrert mot videoopptaket beskrevet i 5.1.1, der t_s er satt til $350 \times 500 = 175\,000$ (bredde $\times$ høyde).

4.4.5 Implementering av fargedeteksjon

Teori om ulike fargerom er gitt i Kapittel 2.4.

Valg av metode for fargedeteksjon

Komplekse fargemetoder, nevnt i Kapittel 2.4 kan gi større immunitet mot belysningsendring, men krever bruk av maskinlæring for å trene opp en klassifiserer. Til dette kreves et stort omfang av bildedata. Derfor ble det bestemt å bruke en enkel fargemetode til egenskapsdetektering som ser på fargelikheten ved to kjøretøybilder. Først ble det testet å bruke fargehistogram til å sammenlikne et fast område på et kjøretøy i forskjellige bilder. Kort forklart så representer et histogram fargefordelingen i et bilde. Det viste seg utfordrende å sammenlikne histogrammene grunnet fargeforskyvning som skjedde ved små lysendringer og kamerainnstillinger. Dette oppstod til tross for stor reduksjon av antall mulige fargeverdier. En metode som viste seg å være mer robust mot slike variasjoner, var å beregne middelverdi for pikselintensiteten (fargen) for et fast område av kjøretøyet. På grunn av sitt store flateareal, velges dette området til å være et lite bildeutsnitt av panseret. Uthenting av panserutsnitt er forklart i nedenfor i dette delkapittelet.

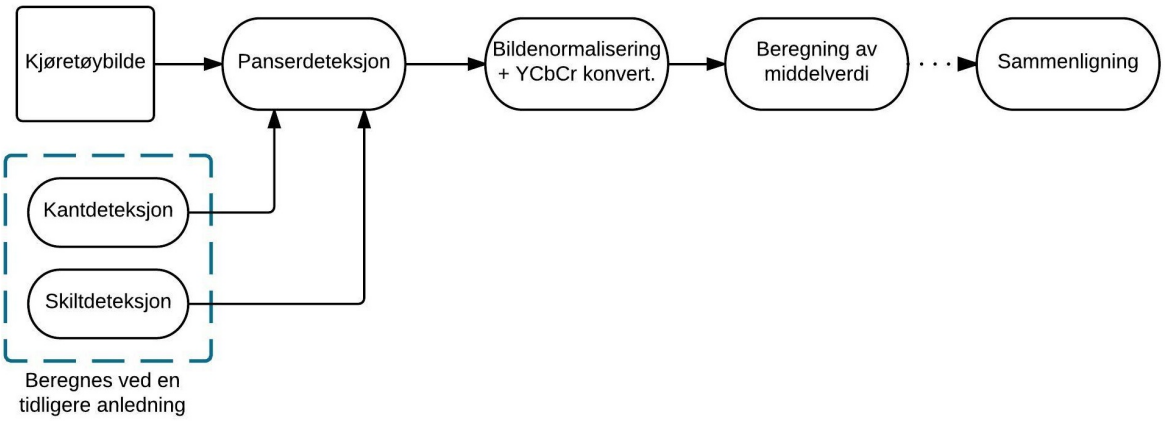
Siden flere kjøretøy har samme farge, er det umulig å finne eksakt kjøretøy ved å bare se på fargelikheten i et område. I oppgaven er det derfor valgt å bruke fargelikheten til å skille vekk kjøretøy som ikke likner. Dermed blir antallet resterende kjøretøy som må sammenlignes ved strukturbaserte metoder, færre.

Valg av fargerom

For at ikke feil kjøretøy skal elimineres, må et fargerom velges ut fra robusthet og invarians mot lysendringer. Derfor må et fargerom kunne skille ut luminansen fra fargene. YCbCr fargerommet ble valgt siden det ga best pålitelighet ut i fra testing, særlig på gråfarget kjøretøy, se Kapittel 5.4. I testen ble RGB, YCbCr og HSV fargerom vurdert. Lab fargerom ble ikke vurdert på grunn av krevende konvertering fra RGB.

Implementering

På grunn av at lastebiler har en mye mer kompleks panser som inneholder mange forskjellige strukturer, har implementeringen av panserutsnitt blitt avgrenset til å kun omfatte lette kjøretøy. Panseret på disse kjøretøyene har i de fleste tilfellene en jevn farge og dermed vil metoden fungere mer pålitelig på disse kjøretøyene. For å sammenlikne fargen i to panserutsnitt, er det viktig at området de er hentet ifra på kjøretøyene er identisk. Algoritmen finner et panserutsnitt ved å ta utgangspunkt i skiltposisjonen og sidespeilene. Sistnevnte detekteres ved hjelp av hjørnedeteksjon. Blokkdiagram for implementeringen er vist i Figur 4.11.



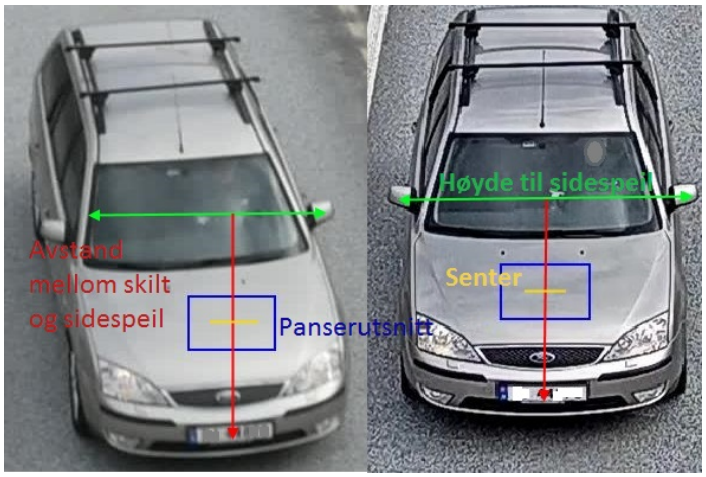
Figur 4.11 Blokkdiagram for fargedeteksjon.

Panserdeteksjon

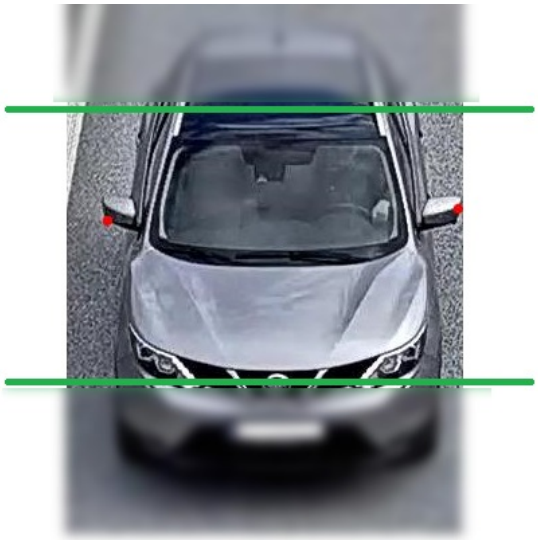
Til uthenting av panserutsnitt, dannes et adaptivt vindu som skaleres i forhold til skiltstørrelsen. Dette gir robusthet mot skalaendringer som oppstår grunnet til forskjellige kamera-zoom og -posisjoner. Bredden på panservinduet settes lik skiltbredden $W_{panser} = W_{skilt}$ og høyden er definert ved 3 ganger skiltbredden $H_{panser} = 3 \times H_{skilt}$. Vinduet er vist i Figur 4.12.

Senteret av panserutsnittvinduet settes i y-retning til å være midtveis mellom skiltsenteret og høydeposisjonen til sidespeilene. Vindusenteret i x-retning er funnet ved å ta utgangspunkt i skiltsenteret. I tilfeller der skilt ikke blir detektert, tas det utgangspunkt i området der kjøretøyet har sin ”tyngde” av hjørner i x-retning, også kalt *massesenteret* (frontsenteret). Dette begrepet er forklart i Figur 4.7.

Sidespeilene blir funnet ved å velge ut mest høyre- og venstre-liggende hjørne på bildet. Detektering av hjørner utføres ved skyggefjerning der FAST hjørnedetektor er valgt. For å forbedre sjansen til korrekt detektering, fjernes hjørner som representerer usannsynlig områder for sidespeil. Dette området er markert som uskarpt i Figur 4.13.



Figur 4.12 Deteksjon av panserutsnitt fra samme kjøretøy ved forskjellig observasjonspunkter.



Figur 4.13 Røde prikkene vises detektert sidespeil.

For de to detekterte sidespeilene, regnes det ut en middelverdi av høydeposisjonen. Om det er for stor høydeforskjell mellom detektert venstre og høyre hjørne, det vil si avstand $>$ 50 piksel, velges høyden til hjørnet som er nærmest bildesenteret. Høyden brukes så som referansepunkt sammen med skiltposisjonen for å sentrere vinduet som henter ut et panserutsnitt. Ved en kameraposisjon som gir en vinkling på kjøretøyet, velges bare et sidespeilet som referansepunkt.

Bildenormalisering og konvertering til YCbCr

For å fjerne høyfrekvent støy i panserutsnittet, er det viktig at bildet først blir glattet. Til

dette brukes et gaussisk glattefilter med $\sigma = 2$. Ved en sammenligning av farge, er det også viktig at kontrasten normaliseres. Til dette brukes gammakorreksjon. Hvitbalanse-filtrering ble også vurdert, men ulike artikler har skiftende mening om virkningen [61]. Axis Q1615-E nettverkskamera har en automatisk funksjon for hvitbalansering. Denne funksjonen fører til at farger i et bilde vises likt uansett temperatur til lyskilden og derfor unnlates implementering av denne funksjonen i Matlab. Videre konverteres RGB bildet av frontutsnittet til YCbCr fargerom. Verdien av *Y* fargekanalen er i området $[16, 235]$, mens *Cb* og *Cr* er i området $[16, 230]$. Disse blir normalisert til en verdi $[0, 1]$.

Beregning av middelverdi og sammenlikning

For hver fargekanal, dannes det et fargehistogram. Beregningen av middelverdien er forklart i likning (2.14). Ved matching, sammenlignes middelverdien mot andre fronter for å finne hvilke som har lik farge. Dette gjøres ved å beregne avstand i form av *middel absolutt avvik* forklart i likning (2.15). Her vil den største mulige avstanden ha verdien 1 og den miste vil gi 0. Vurdering av match som *lik* eller *ulik* gjøres videre ut fra likning (4.6) der avstanden d_{MAE} sammenliknes mot en grenseverdi t_{dist}.

$$\begin{cases} Lik & d_{MAR} < t_{dist} \\ Ulik & ellers \end{cases} \tag{4.6}$$

Her beregnes avstand for *Y* og *CbCr* adskilt. Avstand for *Y* beregnes først. Om resultatet gir en avstand over en satt verdi, regnes kjøretøyene som *ulik*. Derimot hvis en match klassifiseres som *lik*, vil videre en avstand for *CbCr* beregnes ut fra likning (2.15) hvor *N*=2. Å skille *Y* fra *CbCr* gir metoden større robusthet mot lysendringer.

4.4.6 Implementering av frontsegmentering

Fronten ved kjøretøyet inneholder mye informasjon som kan brukes til identifisering. Dette kan for eksempel være en logo som forteller noe om bilmerket. Derfor har flere metoder blitt foreslått til å detektere en front [2] [61]. I oppgaven skal fronten segmenteres ut for å ha et fast område på kjøretøyet som skal brukes til gjenkjenning. Her skal strukturbasert algoritme forklart i Kapittel 2.5 brukes til å danne en deskriptor av fronten og videre matches. I Figur 4.14 og 4.15 vises en bilfront ved to ulike kameraposisjoner.

Den kanskje største utfordringen med frontdeteksjon er å definere grensene. For at samme område skal bli segmentert selv ved ulike observasjoner, bør denne grensen være avhengig av så få variabler som mulig. Toppen av frontområdet ble først vurdert avgrenset

(a) Kjøretøy (b) Bilfront

Figur 4.14 Kjøretøy filmet med Kamera 1.

(a) Kjøretøy (b) Bilfront

Figur 4.15 Kjøretøy filmet med Kamera 2

ved lyktposisjonen, men det viste seg utfordrende å gjøre en pålitelig deteksjon av frontlykt i dagslys. Dette særlig for lyse kjøretøy. Metoden som viste seg å være bedre egnet, var å ta i bruk skiltposisjonen. Denne brukes til posisjonering av deteksjonsvinduet. Størrelsen av vinduet bestemmes ut ifra skiltstørrelsen, noe som gjør metoden robust mot skala endringer. Siden formen på frontflaten er anderledes for forskjellige kjøretøytyper, defineres størrelsen av frontvinduet ulikt for lette og tunge kjøretøytyper.

Bredden W_{roi} og høyden H_{roi} for deteksjonsvinduet som også kan kalles for ”region of interest” (roi), er definert for lette kjøretøytyper i likning (4.7).

$$\begin{aligned} W_{roi} &= 3 \times W_{skilt} \\ H_{roi} &= 5 \times H_{skilt} \end{aligned} \tag{4.7}$$

Størrelsen til frontvinduet for tunge kjøretøy er definert i likning (4.8).

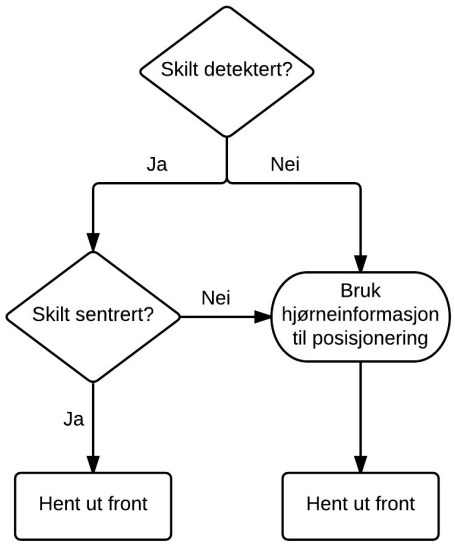
$$\begin{aligned} W_{roi} &= 3.5 \times \times W_{skilt} \\ H_{roi} &= 7 \times H_{skilt} \end{aligned} \tag{4.8}$$

Likning (4.7) og (4.8) er definert ut ifra kameraposisjon der planvinkel er $\theta = 0$. Denne vinkelen er forklart i opptaket gjort i Kapittel 5.3. Når et stort geometrisk skift av kjøretøyobjektene oppstår mellom opptakene, betyr det at kameraene er rotert forskjellig i forhold til veiplanet. Et eksempel på dette er vist i Figur 4.14a og 4.15a der Kamera 1 har planvinkel $\theta = 4$ og Kamera 2 har vinkel $\theta = 0$. En stor rotasjon av kjøretøyobjektet gjør at en mindre del av fronten blir observerbar. Parameterene som definerer størrelsen på deteksjonsvinduet må derfor tilpasses planvinkelen. Ved opptakene gjort av Kamera 1 med $\theta = 4$, har vindubredden blitt redusert med 5%. Posisjonen til deteksjonsvinduet bestemmes slik at senteret av nedre konturlinje matcher bunnen av bilskiltet. Dette er illustrert i Figur 4.16.

Figur 4.16 Definisjon av bilfront (roi) basert på posisjon av bilskiltet.

Før en front kan hentes ut, må bildetransformasjonen normaliseres. Dette gjøres ved å rotere bilde ved bruk av rotasjonsmatrise vist i likning (4.4). Rotasjonsvinkelen må settes ved kalibrering av systemet og ble her satt lik planvikel θ for opptak forklart i Kapittel 5.1.1. I tilfeller der et skilt ikke er detektert eller viser seg ikke å være sentrert, brukes hjørnedeteksjon til å danne referansepunkter. Dette er vist i Figur 4.17.

Algoritmen sjekker først om skiltet er i senter på kjøretøyet ved å sjekke hvor nært det befinner seg *massesenteret* og om det er langt fra senteret av bilderammen. I tilfellet der algoritmen vurderer skilt som ikke sentrert, brukes hjørnepunktene som er detektert ved bruk av FAST algoritmen til å sentrere deteksjonsvinduet. Lavest detektert hjørnepunkt i bilderammen definerer nedre del av frontområdet, altså vinduets posisjon i y-retning. Langs x-aksen *sentreres* frontvinduet i punktet som ligger midt mellom mest høyre- og venstre-

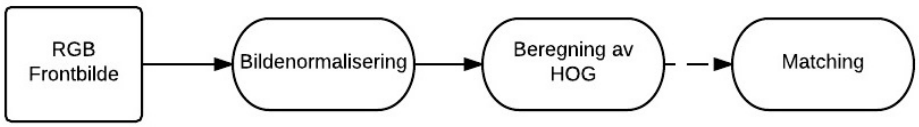


Figur 4.17 Implementering av frontdeteksjon.

liggende hjørnepunkt i nedre bildeområde. For å videre detektere senteret av fronten med en større nøyaktighet, beregnes det en middelverdi mellom punktet funnet over og *massesenteret*. Middelverdien blir dermed det nye senteret for vinduet.

4.4.7 Implementering av HOG

Delkapittelet forklarer hvordan bildet normaliseres før HOG beregnes og hvilken metode som er valgt for matching av deskriptoren. Teorien bak HOG er forklart i Kapittel 2.5.1. Implementering av HOG er vist i Figur 4.18.



Figur 4.18 Her vises implementering av HOG.

Beregning av HOG gjøres på et bilde av kjøretøyfronten som er hentet ut ved segmentering. For å matche HOG vektorene må lengden være lik. Dette gjøres ved å skalere ned bildet til en valgt bildestørrelse. Her er det valgt 160 piksler i lengden og 53 piksler i høyden. Denne verdien er valgt ut fra optimalisering av deskriptoren og forklart i Kapittel 5.5.1. Liten lengde på vektoren gir en rask kjøretid for matching av deskriptorene.

Det er normalt å glatte bildet før gradienten beregnes. Til dette brukes et gaussisk glattefilter med glattekoeffisienter $\sigma_1 = 0.9$ for frontutsnitt detektert i Kamera 1 og $\sigma_2 = 2.5$

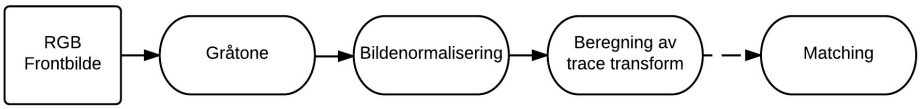
for Kamera 2. Valgene for verdiene er forklart i Kapittel 5.5.1. Etter bildenormalisering, beregnes en deskriptor for HOG.

Matching av HOG deskriptor

For matching av deskriptor velges **normalisert-krysskorrelasjon (NCC)** fordi det er en populær måte å matche HOG deskriptorer [45]. Det gir et godt resultat ved testing. Fordelen med metoden er at den er mindre sensitiv til lineære forandringer i tallverdien sammenlignet med vanlig krysskorrelasjon[86]. En annen fordel er at ulike korrelasjonskoeffisienter beregnes ved alle mulige skift av NCC. Dermed kan to egenskapsvektorer med forskjellige lengder matches. NCC er vist i likning (2.21).

4.4.8 Implementering av trace transform

Implementering av trace transform er vist i Figur 4.19.



Figur 4.19 Her vises implementering av trace transform.

Trace transform er beregningsmessig krevende. Defor må bildestørrelsen reduseres før beregning av signaturen skal gjøres. Her reduseres bildet til 110 piksler i bredden og høyden reduseres slik at størrelseforholdet i bildet bevares. Videre glattes frontbildene, som hentes ut fra Kamera 1 og 2, for høyfrekvent støy med et gaussisk glattefilter med glattekoeffisient $\sigma_1 = 0.7$ og $\sigma_2 = 2$. Parametrene er valgt ved optimalisering av deskriptoren i Kapittel 5.5.2. Teorien bak trace transformen er forklart i Kapittel 2.5.2.

Før en deskriptor beregnes for trace transformen, må vi velge hvilke T- og P-funksjoner som skal beregnes. Kombinasjonen av alle 7 T-funksjonene og 3 P-funksjonene gir totalt 21 sirkelfunksjoner og beregning for hver av disse krever mye datakraft. Flere funksjoner bærer på mye av den samme informasjonen og kan sees på som overflødig fordi de ikke bidrar til nye ”bevis” for å skille ulike bilder. Disse fører dermed til en missvisende likhet og danner en høy korrelasjonskoeffisient med hverandre. Derfor har det blitt foreslått å plukket ut 8 sirkelfunksjoner som returner lavest korrelasjonskoeffisienter [89]. Dette vil kunne redusere kjøretiden til trace transformen betraktelig. T-funksjonene som gir lavest korrelasjon seg imellom er nr. 2, 5, 6, 7 vist i Tabell 2.1 og P-funksjonene 1 og 2 vist i Tabell 2.2.

Det er også foreslått å bruke funksjoner som returnerer ikke-korrelerte ortogonale signaturer [89], men denne funksjonen er ikke tilgjengelig på dette stadiet i form av Matlab kode skriver forfatterne. For å videre redusere kjøretiden til algoritmen, er det valgt å sette vinkelsteget mellom snittene til 3° [89]. Mer om implementering av trace transform i ulike programmeringsspråk kan leses i A. Frias-Velazquez, et. al. [20].

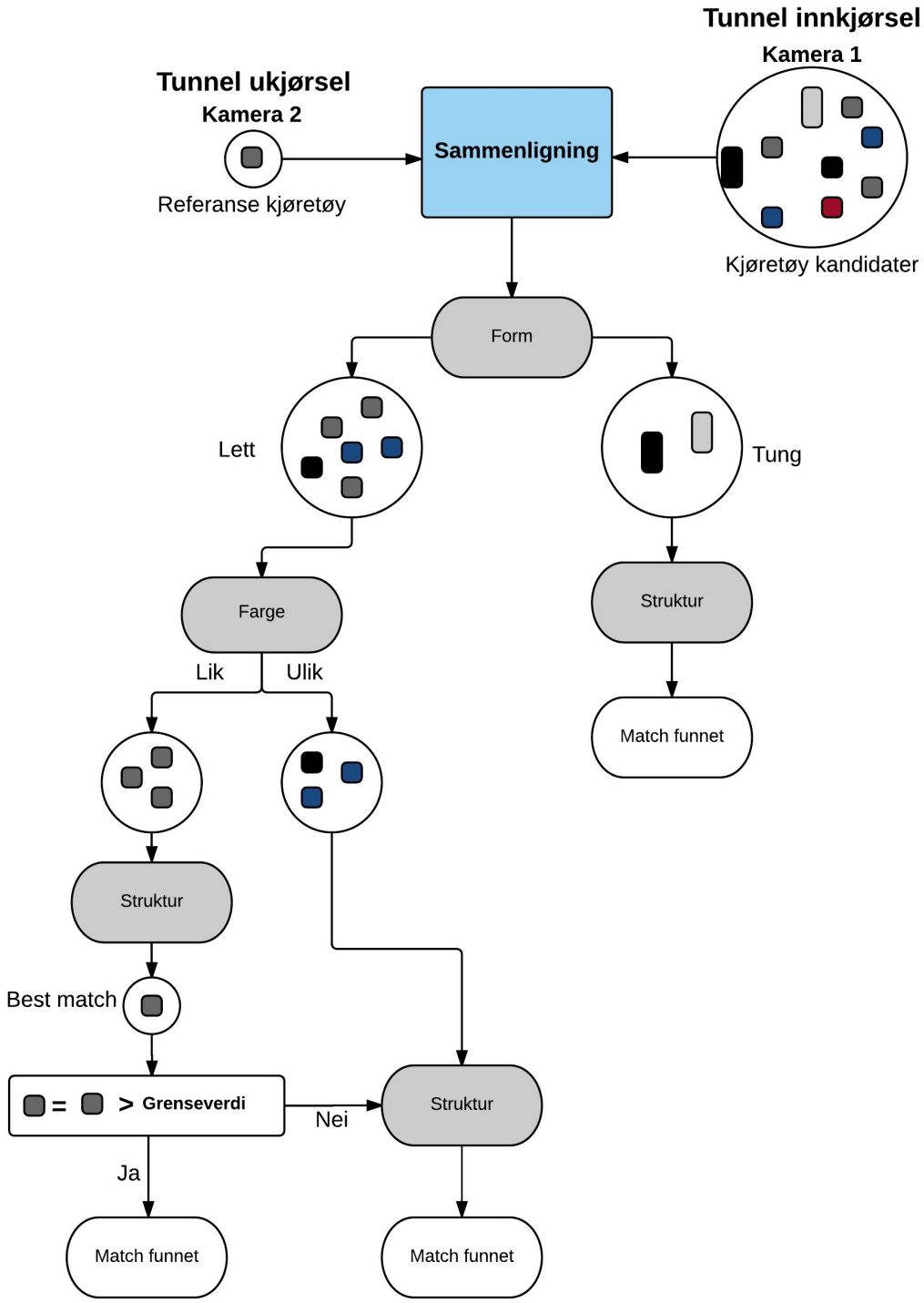
Matching av trace transform deskriptor

Ved matching av trace deskriptorer kan flere av de metriske målene som er nevnt tidligere brukes. Det mest anvendte i dag er å bruke NCC, forklaret i likning (2.21). Denne metoden ble anbefalt i arbeidet til A. F. Velazquez, et al. [89], og A. Kadyrov et al, [36]. Ved sammenligning av to signaturer beregnes ulike korrelasjonskoeffisienter ved alle mulige skift av NCC. Deretter velges den maksimale verdien ved alle skift der to signaturer er mest like idet korrelasjon maksimeres.

4.5 Algoritme for kjøretøysammenligning

Delkapitelet tar for seg hvordan egenskapsvektorer matches i systemet for å gjenkjenne kjøretøy. Ideen er å bruke parametere som beskriver farge og størrelse til å skille vekk usannsynlige kjøretøykandidater, og videre bruke strukturbaserte metoder for å utføre en mer grovere sammenlikning. Med farge menes både en sammenligning av middelverdi av fargen på panseret samt fargen på bilskiltet. I noen tilfeller kan feil farge detekteres som vil resultere i at kjøretøyet klassifiseres som *ulik* og dermed filtreres vekk. Derfor sjekker systemet alltid om høyest oppnådde korrelasjonskoeffisient ved strukturbaserte metoder er over en satt grenseverdi, her 0.75, og samtidig om avstanden til kjøretøyet som gir nest høyest likhetskoeffisient er over 0.1.

Det er også mulighet å bruke tid som en faktor til å redusere antall kjøretøy som skal matches. Dette er ikke gjort her grunnet at opptak som ble benyttet har for kort tidshorisont. Opptaket er forklart i Kapittel 5.1.1. Et flytskjema som beskriver hvordan egenskapene skal sammenliknes er vist i Figur 4.20.



Figur 4.20 Her vises hvordan et kjøretøy detektert ved utkjørselen av tunnel, matches mot ulike kjøretøykandidater observert ved innkjørselen. Først blir antall kandidater redusert ved å sammenlikne kjøretøyklassen. En videre reduksjon av kandidater foregår ved å se om fargen er *lik* eller *ulik*. Ut fra de resterende kandidatene brukes en strukturbasert algoritme for å finne den kandidaten som gir høyest match. Dersom korrelasjonskoeffisienten til matchen er under en gitt grenseverdi, sammenlignes også egenskapsvektoren med kjøretøyene som ble gruppert til *ulik* ved fargesammenlikning.

Kapittel 5

Eksperimentell del og drøfting av resultat

I dette kapittelet fremlegges resultatene fra testingen av algoritmene og diskusjonen rundt disse. Testing er blitt gjort av algoritmen for bakgrunnssubtraksjon, fargesammenligning av panserutsnitt og sammenligning av frontutsnitt ved bruk av HOG og trace transformen. Det gis også en forklaring av kameraoppsettet under opptak av testdata.

5.1 Oppsett av eksperiment

I oppgaven er videoopptak blitt gjort på offentlig vei ved bruk av to kamera som står nær hverandre. Rammebetingelsen for opptakene er forklart i Kapittel 4.1. Under opptak ble boksen rundt Axis Q1615-E nettverkskamera avmontert for å få plass til et kamerastativ. Bilde av utstyret under opptak er vist i Figur 5.1. Skilt og personer i opptakene har i ettertid blitt sladdet for å bevare anonymiteten til sjåførene. Opptakene er blitt gjort på Solasplitten fra en gangbro. Fartsgrensen var her ved start av filmperioden 40 km/t, men ble etterhvert hevet til 60 km/t.

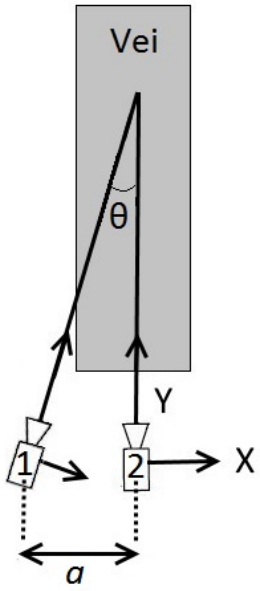
Kamera 1, se Figur 5.1b, simulerer opptak ved tunnelinngang. Kamera 2 representerer opptak gjort ved tunnelutgang. Kameraene brukt til opptak har forskjellig skarphet og fargeoppfatning (kamerainfo er gitt i Kapittel 4.2). Avstanden mellom kameraene skaper en geometrisk transformasjon av kjøretøyobjektet. Målet er å skape bildetransformasjoner som kan oppstå når et kjøretøyobjekt blir detektert ved forskjellige tunnelpunkter. Tegning av kameraoppsettet ved filming er vist i Figur 5.2. Opptakene som forklares under, er delt inn i tre grupper. Videopptak 1 og 3 inkluderer både filmopptak gjort av Kamera 1 og 2. Videopptak 2 inneholder kun opptak gjort av Kamera 2. Videoopptakene er brukt i forskjellige type tester i

dette Kapittelet.

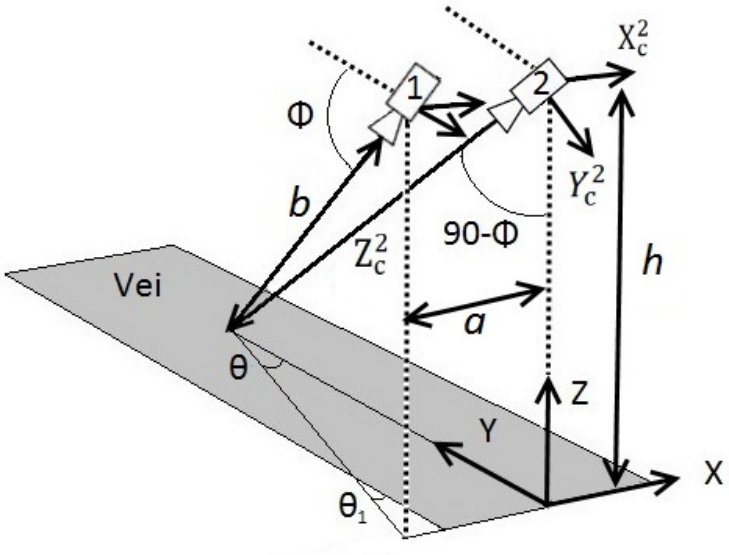
(a) Bildet viser Kamera 2.

(b) Kamera 1 til vestre illustrerer kamera ved tunnel inngang og Kamera 2 til høyre ved utgang.

Figur 5.1 Opptak gjort på offentlig vei.



(a) Sett ovenfra-ned.



(b) Samme scene fra venstre side.

Figur 5.2 Kamera 1 og 2 er plassert i høyde h meter over veien med tilt vinkel ϕ og plan vinkel θ. Under opptakene er θ lik 0 for Kamera 2. X, Y, Z beskriver verdenskoordinatsystem og X_c, Y_c, Z_c er kamerakoordinatsystem. a er avstanden mellom kameraene og b er avstanden fra det optiske senteret til der den optiske aksen krysser veiplanet.

5.1.1 Videoopptak 1

Info om opptaket er gitt i Figur 5.3. I videopptak 1 er 15 kjøretøypar blitt filmet. Algoritmen i systemet blir testet mot dette opptaket. Testen er forklart i Kapittel 5.6 og bilde av kjøretøyene er vist i Figur A.6 og Figur A.7. Videoopptaket er lagt som vedlegg. Opplysninger som kan være med på identifisere personer er sladdet.

Opptak Kamera 1	Opptak Kamera 2
Navn: Trafikk_kamera1.mp4 Varighet: 1.13s Bilde/s: 29	Navn: Trafikk_kamera2.mp4 Varighet: 1.08s Bilde/s: 25

Figur 5.3 Her vises info om videoopptak 1. Dette opptaket regnes som hovedopptaket i oppgaven.

Under opptaket med Kamera 1, oppstod det et uheldig moment som har blitt klippet bort. Derfor er ikke opptakene helt synkronisert. Kameraposisjonen og innstillingene under opptak ble bestemt ut fra ulike interesser. Zoom-innstillingen under innspilling måtte balanseres mellom høy optisk zoom for å detektere kjøretøyobjektet og skilt i tilfredsstillende kvalitet, og samtidig få plass til et vogntog i bilderammen. På samme måte ble helningsvinkelen på kamera bestemt. Kameraposisjonen under opptak er beskrevet ut fra parametrene i Tabell 5.1 der forklaring av parametrene er gjort i Figur 5.2.

Tabell 5.1 Her vises parametrene som beskriver kameraposisjonen under opptak av *videoopptak 1* og *2*. Avstanden mellom kameraene er funnet ved å bruke målebånd, mens avstanden mellom vei og kamera er funnet ved å bruke lasermåleren Cotech 400. Gradskive er brukt for å beregne vinkel ϕ og θ.

a=2.12m: +/-0.05m	h	b	ϕ	θ	zoom
Kamera 1:	8.55m: +/- 1m	19.80: +/- 2m	65°: +/- 3°	4°: +/- 1°	0
Kamera 2:	8.46m +/- 1m	25.78m: +/- 2m	71°: +/- 3°	0 °	2.29x

Opptakets formål:

Opptaket kan tenkes på som hovedopptaket som alle delleddene i systemet skal testes mot. Grunnet begrenset tid og utfordringer som oppstod under filming, har det blitt valgt å prioritere testing av den samlede algoritmen til systemet opp mot opptaket fra Kamera 2. Dette kameraet har de nødvendige egenskapene til å kunne brukes i et fremtidig system og det har derfor vært viktig at automatisk deteksjon av egenskaper fungerer på dette kameraet.

Ved matching, trenges en egenskapsvektor som er hentet fra en tidligere deteksjon av kjøretøyet. For å spare tid, har området på kjøretøyet der egenskapsdeteksjon skal foregå, blitt segmentert ut gjennom bruk av bilderedigeringsprogram fra opptakene gjort med Kamera 2. Formålet er å ha egenskapversvektorer som en kan sammenligne med og derfor er det ikke nødvendig å utføre en automatisk deteksjon av kjøretøy og skilt. Dermed slipper man å sladde mye ekstra videomateriell som legges ved i oppgaven.

5.1.2 Videoopptak 2

Videoopptak 2 er blitt gjort av Kamera 2 og har som mål å teste bakgrunnssubtraksjon under raske lysendringer, forklart i Kapittel 5.3.1. Opptaket er blitt lagt til som vedlegg. Kameraparametrene som beskriver kameraposisjonen under opptak er den samme som for Kamera 2 i Tabell 5.1. Info om opptakene er gitt i Figur 5.4.

Opptak Kamera 2

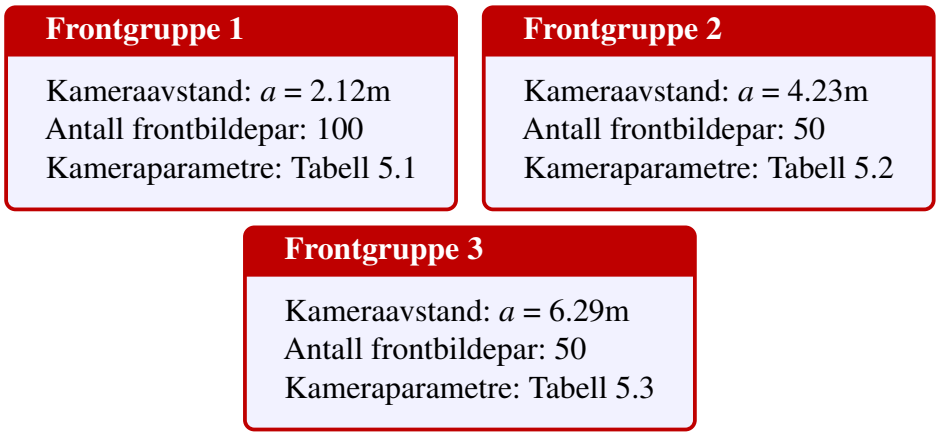
Navn: Trafikk_lysendring_kamera2.mp4
Varighet: 18s
Bilde/s: 25

Figur 5.4 Her vises info om videoopptak 2. Dette opptaket brukes i forbindelse med test av bakgrunnssubtraksjon.

5.1.3 Videoopptak 3

Videoopptak 3 har blitt gjort med det formål å hente ut nok data til frontgjenkjenning. Testen er forklart i Kapittel 5.5. Det vil være interessant å finne ut hvordan algoritmen fungerer under ulike lysendringer og geometriske transformasjoner. Til dette kreves et stort antall testdata av ulike frontutsnitt. For å spare tid, har frontutsnittene blitt segmentert ut manuelt fra opptak gjort under forskjellige værforhold og tider på døgnet. Hadde segmenteringen blitt gjort automatisk, måtte algoritmen blitt kalibrert for hvert opptak. Det hadde vært for tidskrevende og antall kjøretøyfronter som er nødvendig for å gjøre en grundig testing av den strukturbaserte algoritmen hadde ikke blitt nådd.

Opptakene gjort i forbindelse med frontsammenligning, kan deles inn i tre grupper som bestemmes ut ifra avstanden mellom kameraene. Disse bildegruppene skal testes mot hverandre. Målet er å se hvor stor geometrisk transformasjon algoritmen tåler før det oppstår for høy feilrate ved sammenligning. Frontgruppene er vist i Figur 5.5.



Figur 5.5 Her vises de tre gruppene av opptak gjort for å innhente nok data til å teste metoden for frontsammenligning.

Frontgruppe 1 består av 100 frontutsnittspar. Algoritmen har hentet ut 15 av utsnittene automatisk fra videopptaket beskrevet i Kapittel 5.1.1. De resterende frontutsnittene er segmentert ut manuelt fra opptak gjort under forskjellige tider på døgnet og kamerainnstillinger. Dette er gjort for å skape endringer i skala, bildeskarphet og kontrast. Slike endringer vil forekomme under reelle forhold. Det at kameraene filmer kjøretøyet fra forskjellige observasjonspunkter, har bidratt til å skape en bildetransformasjon i form av rotasjon. Ved testing av strukturbasert algoritme skal virkningen av rotasjonsnormalisering testes. Derfor er datasettet delt inn i to grupper. Ett der alle frontutsnittene er rotasjonsnormalisert og ett uten. Rotasjonsnormalisering er blitt utført ved å rotere bildet med vinkelen θ.

Tabell 5.2 Her vises parametrene som beskriver kameraposisjonen under opptak ved kameraavstand a=4.23m (Frontgruppe 2).

a=4.23m: +/-0.05m	h	b	ϕ	θ	zoom
Kamera 1:	8.55m: +/- 1m	20.23: +/- 2m	65°: +/- 3°	8.5°: +/- 1°	0
Kamera 2:	8.46m +/- 1m	25.78m: +/- 2m	71°: +/- 3°	0°	2.29x

Tabell 5.3 Her vises parametrene som beskriver kameraposisjonen under opptak ved kameraavstand a=6.39m (Frontgruppe 3).

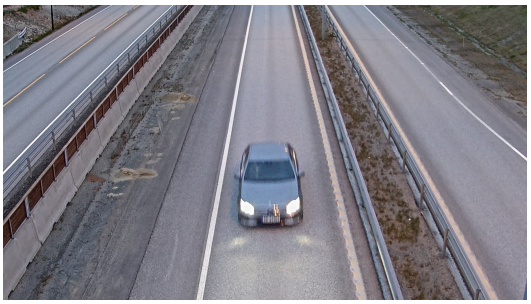
a=6.39m: +/-0.05m	h	b	ϕ	θ	zoom
Kamera 1:	8.55m: +/- 1m	20.68: +/- 2m	66°: +/- 3°	13°: +/- 1°	0
Kamera 2:	8.46m +/- 1m	25.78m: +/- 2m	71°: +/- 3°	0°	2.29x

5.2 Analyse av kamera under svake lysforhold

For å detektere skilt og egenskaper knyttet til panserutsnitt, settes det krav til kvaliteten på videoopptaket. Høy bildekvalitet kan være vanskelig å få til på kveldstid eller opptak gjort i dårlig belyst tunnel, selv med et avansert Axis Q1615-E forklart i Kapittel 4.2. Derfor er det utført en test under dårlige lysforhold for å foreta en "visuell" vurdering om skarpheten i bildet er tilfredsstillende nok til at en frontsammenligning kan utføres. I testen er det foretatt opptak av kjøretøy på vei kl 22.30, vist i Figur 5.7 og 5.8. Her testes også bruken av WDR-funksjonen (Wide dynamic range) på kameraet med mål om å forbedre detaljnivået i mørke områder. Funksjonen forbedrer eksponering når det er en betydelig kontrast mellom lyse og mørke områder i et bilde og samtidig styrker kameraets evne til å tåle motlys. En demonstrasjon av funksjonen er vist i Figur 5.6. Resultat av testen er vist i Figur 5.7 og 5.8.

(a) WDR av. (b) WDR på

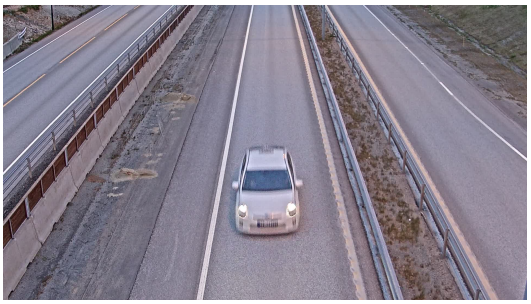
Figur 5.6 Test med og uten WDR. WDR gir her en høyere bildekontrast der flere detaljer kan skimtes, men har den ulempen at områder med høy intensitet blir missdannet som for eksempel frontlyktene på bilen.



(a) Filming av kjøretøy kl 22.30 med Axis Q1615-E nettverkskamera. WDR funksjon er slått av.

(b) Bilde av kjøretøyfronten. Det resulterende bilde er for uskarpt til at frontsammenligning kan utføres. Årsaken til uskarpheten er lang lukkertid på kameraet for å samle nok lys. Dette fører til dannelse av fenomenet kalt "motion blur". En raskere lukkertid vil derimot føre til mer støy ved lite lys. Her kan det også sees lysrefleksjoner på kjøretøyfronten forårsaket av en lyktstolpe. Ved en sammenligning, kan dette skape utfordringer.

Figur 5.7 Test av opptak under svake lysforhold. WDR er slått av.



(a) Filming av kjøretøy kl 22.30 med Axis Q1615-E nettverkskamera. WDR funksjon er slått på.

(b) Bilde av kjøretøyfronten. WDR funksjonen til kamera er ikke moden nok og forårsaker missdannelser i områder med høy intensitet.

Figur 5.8 Test av opptak under svake lysforhold. WDR er slått på.

Med denne testen har det blitt vist at frontdeteksjonsalgoritmen ikke vil kunne fungere i opptak under svak belysning med kameraet som brukes i oppgaven.

5.3 Analyse av bakgrunnssubtraksjon

Bakgrunnssubtraksjon er anvendt på videopptak 1. En forutsetning for opptakene er at autoeksponering er slått av. Autoeksponeringen forårsaker raske oscilleringer i bildekontrasten når lyssterke motorkjøretøy passerer. Dermed blir det vanskelig for algoritmen å tilpasse bakgrunnsmodellen raskt nok. Bakgrunnssubtraksjon er utført ved å bruke Gaussian Mixed Models (GMM) (ref. Kapittel 2.1.1) og implementering (ref. Kapittel 4.4.1).

Bakgrunnssubraksjonsmetoden blir testet ved å se hvor nøyaktig et kjøretøy blir segmentert fra bakgrunnen idet det passerer deteksjonsområdet vist i Figur 5.9b. Det resulterende kjøretøybilde som bakgrunnssegmenteringsmetoden henter ut, skal brukes videre for å teste ulike algoritmer som detekterer egenskaper. Det er derfor viktig at påvirkningen fra bakgrunnen blir minimal.

(a) Bakgrunn. (b) Kjøretøy.

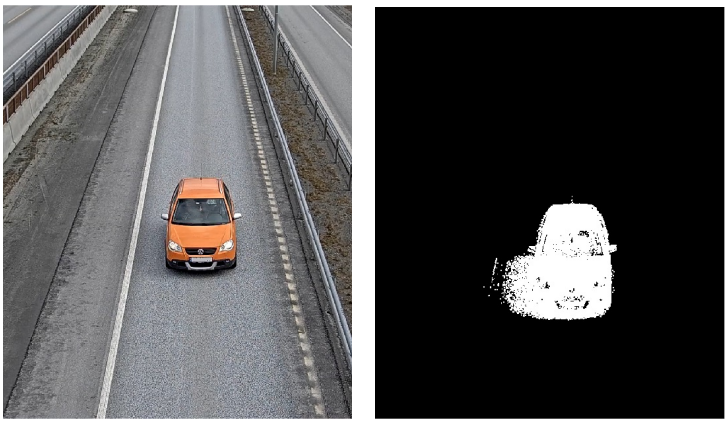
Figur 5.9 Kjøretøyet som skal segmenteres idet det krysser deteksjonsområdet vist i grønt.

Videre skal en robusthetstest utføres på videopptak 2 (ref. Kapittel 5.1.2). Her brukes et 18 sekunders langt videoopptak som inneholder en rask lysendring. Målet er å se hvilke parametre i algoritmen som kan tilpasses for å tåle raske endringer i lysforholdene.

5.3.1 Resultat av bakgrunnssubtraksjon

Resultatet av bakgrunnssubtraksjon er vist i Figur 5.10. Bilder av alle 15 kjøretøyene som er hentet ut ved bakgrunnssubtraksjon er vist i Figur A.7. Nedenfor gis en begrunnelse for valg av parametere for GMM.

En vei skal være statisk og derfor er det ingen grunn til å bruke mange gaussiske funksjoner N for å modellere bakgrunnen. Tre gaussiske funksjoner er tilstrekkelig. Læringsratekoeffisienten δ brukt i likning (2.4) må bestemmes. For initialisering av bakgrunnmodellen, setts δ til å være adaptiv for å danne en rask modell av bakgrunnen ved start. Antall treningsrammer settes til 150 der læringsraten δ endres til $\delta = 1$ /(nåværende bildenummer) så lenge bildet er en treningsramme. Om plutselig en gatelykt skulle bli skrudd på ved kveldstid, må



(a) Trafikkbilde uten bakgrunnssubtraksjon.

(b) Resulterende bakgrunnsmaske ved bakgrunnssubtraksjon.

Figur 5.10

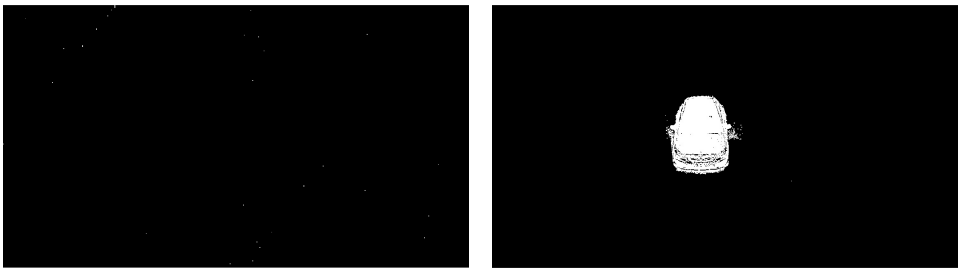
bakgrunnsmodellen kunne håndtere endringen. Derfor anbefales det at læringsraten δ settes høyt. Om parameteren settes til $\delta = 0.001$ vil et objekt ta $\log(T)/\log(1\text{-}\delta) = 105$ rammer for å bli en del av bakgrunnen når $T = 0.9$.
T-parameteren fra likning (2.9) bestemmer minimum andel av data som kan tilhøre forgrunnen uten å påvirke bakgrunnsmodellen. For liten T vil medføre at færre piksler blir klassifisert som forgrunn. Parametrene er valgt ut fra testing, og er vist under:

- $N = 3$
- $\delta = 0.005$
- $T = 0.6$

For å teste om algoritmen kan håndtere lysendring, blir det foretatt en robusthetstest. Her er det ønskelig at bakgrunnsmodellen skal kunne oppdateres raskt for å tilpasse seg de nye lysforholdene. Et godt resultat er avhengig av at algoritmen unngår å klassifisere deler av bakgrunnen som forgrunn. Resultatet av testen vises i form av to bilder tatt med 16 sekunder mellomrom i Figur 5.11 og 5.12. Figurene viser en rask tilpasning ved lysendringer der ingen bakgrunnspiksler blir klassifisert feil.

(a) (b)

Figur 5.11 Fra venstre: Bilde av bakgrunnsomgivelsene sammen med et bilde tatt 16s senere. Her vises tydelige lysendringer som danner skygger i bakgrunnen.

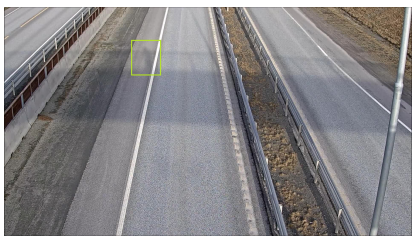


(a) (b)

Figur 5.12 Fra vestre: Her vises resultatet av bakgrunnssubtraksjonen utført på opptaket vist i Figur 5.11. Bildet til høyre viser delen som er klassifisert forgrunn i hvitt.

5.3.2 Utfordringer ved bakgrunnssubtraksjon

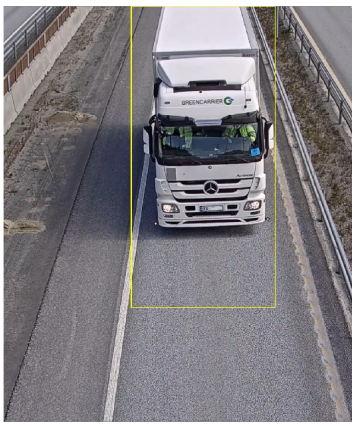
Det finnes tilfeller der metoden ikke har mulighet til å tilpasse seg raske endringer. Derfor er tester blitt gjort for å vise eksempler på utfordringer med metoden.



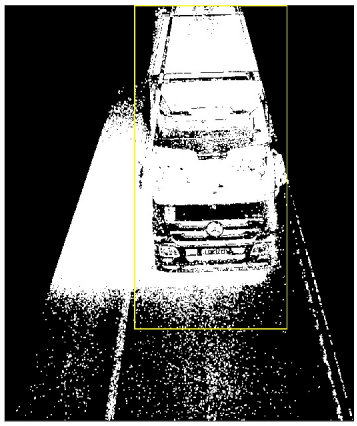
(a) En person går over en gangbru som danner en skygge på veien.

(b) Skyggen fra en person (markert i gult) som beveger seg blir klassifisert som forgrunn fordi endringene i bakgrunnen skjer så raskt at bakgrunnsmodellen ikke klarer oppdatere seg raskt nok.

Figur 5.13 Utfordring 1: Utenforliggende objekter som skaper endringer i bakgrunnen.

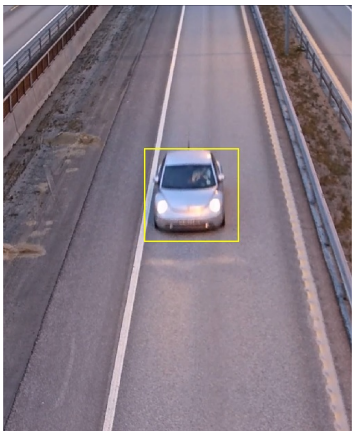


(a) I denne testen er autoeksponering blitt slått på. Det detekterte kjøretøyet vises i gult.

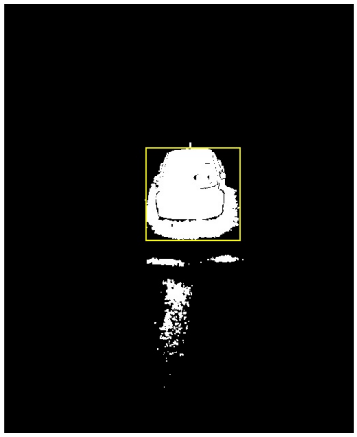


(b) Autoeksponeringen skaper uønskede endringer i bakgrunnen som fører til at GMM feilklassifiserer flere piksler som forgrunn. Dette fører til at større kjøretøyområde detekteres. En mindre aggressiv autoeksponering er derfor en fordel.

Figur 5.14 Utfordring 2: Autoeksponering.



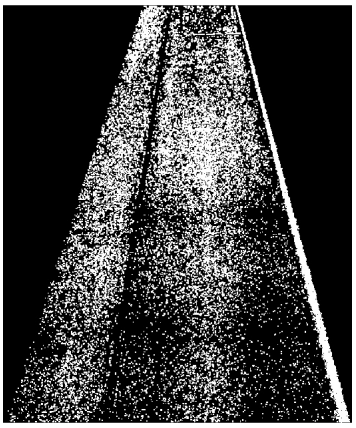
(a) Opptak gjort sent på kvelden. Ved lite lys blir lyktrefleksjoner et problem som kan tolkes feilaktig som forgrunn.



(b) Her blir lys-refleksjoner fra lykten en del av forgrunnen. Den gule markeringsrammen viser til det detekterte kjøretøyobjektet.

Figur 5.15 Utfordring 3: Lyktrefleksjoner.

(a) I denne testen har det blitt gitt et kraftig dytt til kamera med mål å teste hvordan algoritmen takler vibrasjoner.



(b) Den kraftige vibrasjonen skaper endringer i bakgrunnen. Likevel detekteres ingen kjøretøy (gul markeringsboks) grunnet verifiseringsalgoritmen som fjerner uønskede kandidater.

Figur 5.16 Utfordring 4: Vibrasjoner.

5.4 Analyse av fargesammenligning

Teorien bak fargesammenligning er forklart i Kapittel 2.4 og implementert i Kapittel 4.4.5. Fargedatasammenligning gjøres ved å hente ut et lite utsnitt fra fronten for å beregne en middelverdi av fargen. Middelverdien blir så sammenlignet mot andre kjøretøykandidater ved beregning av "middel absolutt avvik" d_{MAE}, forklart i likning (2.15). Lengen på avviket bestemmer om kjøretøyparet som sammenlignes skal klassifiseres som *lik* eller *ulik*.

Ved analyse av fargesammenligning skal det gjøres ulike tester for å finne ut hvilket fargerom som best egner seg til fargesammenlikning under varierende lysforhold. Det skal også testes hvor mange av totalt 16 kjøretøykandidater metoden i gjennomsnitt skiller vekk som *ulik* ved sammenligning når ulike fargerom brukes. Målet er å stå igjen med færrest kjøretøy som stemples *lik* uten å feilklassifisere korrekt kjøretøypar som *ulik*.

Valg av fargerom til analyse

Fargerommene som testes er:

- RGB
- HSV
- HS+V
- YCbCr

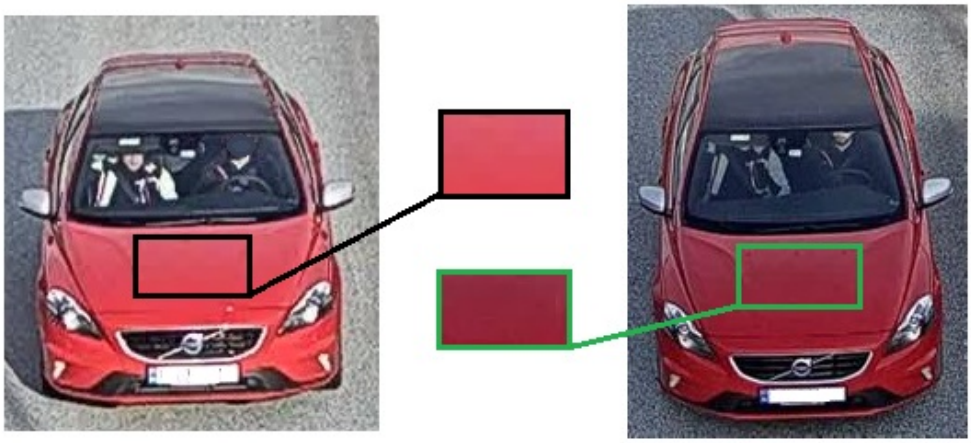
- Y+CbCr

Ved *HS+V* og *Y+CbCr* skilles intensitet-komponenten fra de resterende komponentene. Ideen bak denne separasjonen er å se om algoritmen kan oppnå en større belysningsinvarians ved å veie fargekanalen som inneholder informasjon om luminans mindre. Tre tester av fargerommene skal gjøres der resultatet skal danne et grunnlag til å velge fargerommet som gir størst pålitelighet. Testen er delt opp som følger:

- **Test1:** Test ved kontrastendring
- **Test2:** Test av forskyvning i gråtonesansen
- **Test3:** Test av antall kjøretøy metoden skiller ut

5.4.1 Test 1: Kontrastendring

Denne testen skal teste robustheten av fargerommet under lysvariasjoner. Testen tar for seg et panserutsnitt av samme kjøretøy ved to forskjellige punkter i en veistrekning. Et i sollys og det andre i skygge. Dette er vist i Figur 5.17.



Figur 5.17 Dette bildet viser utsnittet av panserfargen som skal brukes til å teste hvordan ulike fargerom takler lysendringer.

Målet er å finne ut hvilket fargerom som har størst robusthet mot lyseendringer. Det fargerommet som gir lavest middelverdiavstand d_{MAE} vil gi best resultat. Ved sammenligning av *HS+V* og *Y+CbCr*, unnlates intensitetkomponenten. Dette er fordi en sammenligning av intensitetverdi utføres som et eget ledd i systemet for å skille mørke og lyse kjøretøy fra

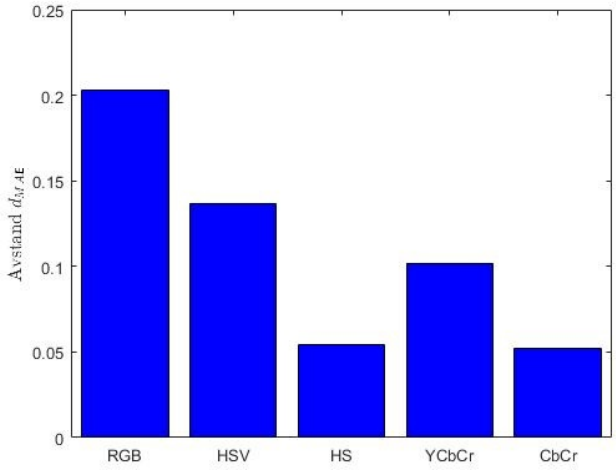
hverandre. Målet i denne testen er derimot å se hvordan lydendringene påvirker fargemiddelverdien i hovedleddet. For *Y+CbCr* er det hovedleddet *CbCr* som inneholder informasjon om fargen.

Resultat

Resultatet for eksperimentet er vist i Figur 5.18c.

(a) Kamera 1: Panserutsnitt

(b) Kamera 2: Panserutsnitt



(c) Resultat av sammenligningen. Høyden av søylene beskriver avstanden for middelverdien til fargen ved sammenligning. Avstanden er beregnet med *mean-absolute error*

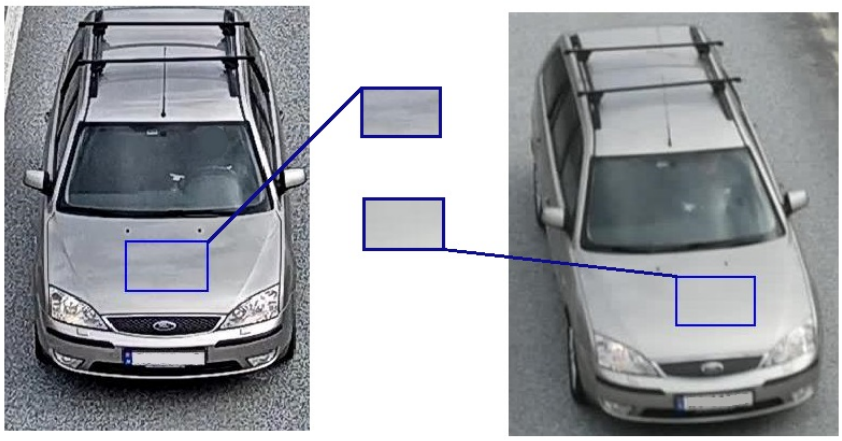
Figur 5.18 Øverst vises bilde av frontutsnittene som skal sammenlignes. Nederst vises resultatet. Søylene kan her sees på som størrelsen av endringen i middelverdien for fargen mellom to observasjonspunkter.

Resultatet viser at fargerommene der intensitet komponenten er separert, gir lavest avstand og dermed best resultat. Som forventet gir *RGB* dårligst resultat fordi alle komponentene inneholder informasjon om luminansen. Ved kontrastendringer oppstår det store forskyvninger i fargeintensiteten. Av fargevektorene som inneholder tre komponenter viser *YCbCr* til minst endring og derfor best resultat.

5.4.2 Test 2: Forskyvning i gråtonen

Flere studier har konkludert at grå/sølv er en av de vanskeligste bilfargene å gjenkjenne [63] [48]. Gråfargen er bare et mål på intensitet og derfor skal det ideelt ikke oppstå endringer i fargekomponenten under lysvariasjoner. Dette er ikke alltid tilfellet. Grå er en populær bilfarge og derfor skal det gjøres en test for å finne det fargerommet som gir størst pålitelighet.

To panserutsnitt brukt i denne testen er hentet fra opptak gjort av samme kjøretøy filmet med to forskjellige kameraer. Skift i gråtonen oppstår her på grunn av kameraenes forskjellige fargeoppfatning. Oppsettet for opptaket er forklart i Kapittel 5.1.1 og bilde av panserutsnittet samt kjøretøyet er vist i Figur 5.19.



Figur 5.19 Dette bildet viser utsnittet av panserfargen som skal brukes til å teste hvordan ulike fargerom takler en forskyvning i nyansen.

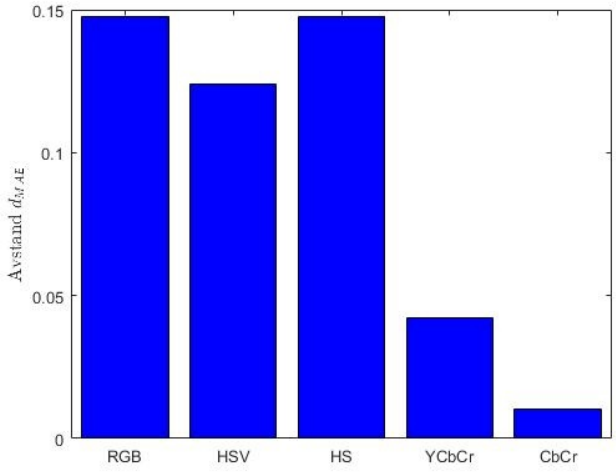
Resultat

Resultatet av eksperimentet er vist i Figur 5.20c. Høyden av søylene kan sees på som graden av endring i fargens middelverdi når forskyvning i nyanse oppstår.

Resultatet av testen viser at fargeintensiteten ved *YCbCr*, samt *CbCr* fargerom endres lite når en forskyvning i gråtoner oppstår. Siden gråtoner beskrives ut fra intensiteten, skapes det problemer for RGB fargerommet når alle fargekanalene inneholder informasjon om luminans. Overraskende nok viser *HSV* et dårlig resultat og enda dårligere når *V* komponenten fjernes. Dette skyldes at grå ikke er en farge som kan beskrives i *hue* (nøytal farge) og derfor kan verdien for *H* komponenten variere ved forskjellige nyanser. I *YCbCr* beskrives fargekomponentene i to dimensjoner og gir større robusthet i dette tilfellet.

(a) Kamera1: Panserutsnitt.

(b) Kamera2: Panserutsnitt.

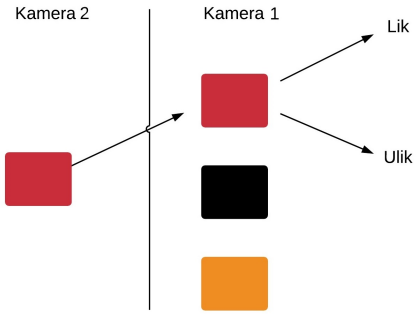


(c) Resultat av sammenligningen. Stolpediagrammet viser avstanden i farge-middelverdi mellom to panserutsnitt.

Figur 5.20 Her vises resultatet av Test 2.

5.4.3 Test 3: Antall kjøretøy som skilles ut

Målet med denne testen er å se hvor mange kjøretøy algoritmen kan avskrives som ulike ved forskjellige fargerom. Her blir hvert av de 16 kjøretøyene testet mot 16 andre kjøretøykandidater. Et godt resultat her vil være få antall kjøretøypar som stemples *lik* og at det samtidig unngås feilklassifisering. Jo flere kjøretøy som klassifiseres *ulik*, desto færre sammenlikninger trenges å utføres for de resterende algoritmene som foretar en grovere sammenlikning. Sammenlikningen er illustrert i Figur 5.21.



Figur 5.21 Sammenligning av frontutsnitt.

De fleste kjøretøyparene brukt i denne testen er hentet fra opptaket forklart i Kapittel 5.1.1 (kjøretøyparene er vist i Figur A.6 og A.7). Resultat av panserutsnittet er vist i Figur 5.22.

(a) Panserutsnitt av bil-ID 1 til 8.

(b) Panserutsnitt av bil-ID 9 til 16.

Figur 5.22 Resultat av panserutsnitt-deteksjon for ulike kjøretøy.

Utsnittsparene vurderes *lik* eller *ulik* ut fra likning (4.6) der parameteren t_{dist} brukes til å avgjøre når avstanden d_{MAE} er for stor til at parene kan være like. For et gitt fargerom settes terskelverdien kritisk lav slik at flest kjøretøy klassifiseres *ulik*, uten å gå på bekostning av feilklassifikasjoner. Parameteren settes ved å ta utgangspunkt i det sanne bildeparet som gir størst avstandsverdi d_{MAE} ved sammenligning, og sette grensen lik $t_{dist} = \text{maks}(d_{MAE}) + 0.01$ for et gitt fargerom. Dermed er terskelverdien t_{dist} satt for hvert fargerom: *RGB*=0.1823, *HSV*=0.125, *YCbCr*=0.0655.

For fargerommene der intensitetkomponenten er avskilt, utføres det to sammenligninger. Først en sammenligning av middelverdi til intensitetkanalen. Her settes en "slakk" terskelverdi t_{dist} for å kunne skille lyse og mørke kjøretøy fra hverandre. Om kjøretøyene vurderes *lik*, gjøres en videre sammenligning av de resterende fargekanalene. For *HS+V*-fargerom, er terskelverdien t_{dist} for *V*-sammenligning satt til 0.3 og HS-delen satt til 0.148. Ved *Y+CbCr*-sammenligning, klassifiseres kjøretøyene som like ved en middelverdiavstanden under 0.2025 og for *CbCr* er denne verdien satt til 0.0379.

Resultat

Resultatet for eksperimentet er vist i Tabell 5.4.

	Lik	Ulik
RGB	4.88 (30.5%)	11.1 (69.4 %)
HSV	3.63 (22.7%)	12.4 (77.5%)
HS+V	3.94 (24.6%)	12.1 (75.6%)
YCbCr	2.94 (18.4%)	13.1 (81.9%)
Y+CbCr	**2.69 (16.8%)**	**13.3 (83.1%)**

Tabell 5.4 Tabellen viser et gjennomsnitt antall kjøretøy som blir klassifisert *lik* og *ulik* ved en fargesammenlikning.

Testen viser hvor stor andel av kjøretøyene som blir skilt vekk ved sammenlikning. Her er parametrene satt kritisk der akkurat ingen kjøretøy blir klassifisert feil. Det er tydelig at *RGB* filtrer vekk færrest kjøretøy. Dette fordi at terskel verdien må settes høyt nok til å takle lysendringer. *YCbCr* viser til et godt resultat der mange kjøretøy filtreres vekk. Interessant gir *Y+CbCr* fargerommet et bedre resultat der separering av *Y* er ment til å gi større pålitelighet ved lysendringer, ikke en økning av ytelsen (reduksjon av antall like kjøretøy). Resultatet fra testene gjort i denne delen, viser at *Y+CbCr* fargerom gir størst robusthet mot lysendringer, samtidig som det klarer å skille flest kjøretøy.

Hvilke panserpar som blir klassifisert *lik* ved *Y+CbCr* fargerom er vist i Tabell A.1. Dette resultatet viser at fargene som er vanskeligst å skille fra hverandre er svart, hvit og grå. Ved sammenligning har disse fargene størst sannsynlighet for å bli stemplet like. Dette fordi bare *Y*-komponenten inneholder informasjon som kan skille "fargene" fra hverandre.

5.5 Analyse av struktur-utvinnende algoritmer

I denne delen skal testing av frontsammenligning gjøres ved bruk av HOG og trace transform under ulike forhold som okklusjon og lysendring. Målet er å se hvilken deskriptor gir best resultat ved sammenligning av frontutsnitt. Følgende deskriptorer skal testes mot hverandre:

- HOG
- Trace transform med 8 sirkelfunksjoner
- Trace transform med 21 sirkelfunksjoner
- HOG + Trace transform med 8 sirkelfunksjoner

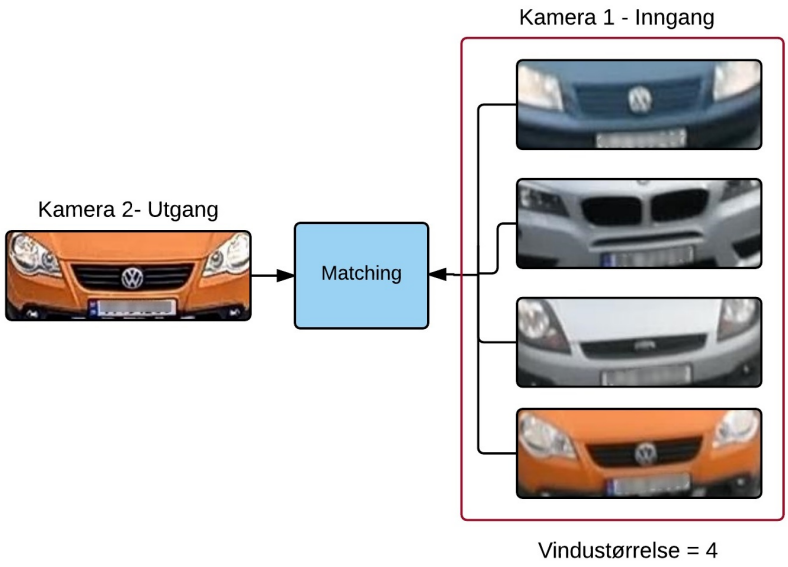
Implementering av HOG deskriptor er beskrevet i Kapittel 4.4.7 og trace transform er forklart i Kapittel 4.4.8. En sammenligning av egenskapsvektorene utføres gjennom beregning av normalisert-krysskorrelasjon. Som grunnlaget om at utregning av trace transform er prosesseringsmessig krevende, skal sammenligning gjøres ved både å bruke alle 21 funksjonene som er dannet ved å kombinere funksjonene i Tabell 2.1 og 2.2 og ved å velge ut de 8 minst korrelerte funksjonene beskrevet i A. F. Velazquez, et. al. [89].

Siden HOG og trace transform henter ut svært forskjellige bildeegenskaper, kan en kombinasjon gi en mer komplett deskriptor av kjøretøyfronten. Derfor skal det også testes om en samlet deskriptor vil kunne gi et bedre resultat. Implementering av en slik deskriptor er gjort ved å beregne en samlet middelverdi av korrelasjonskoeffisienten fra HOG og trace transform med 8 sirkelfunksjoner. Dermed veies bidragene fra begge deskriptorene like mye. Optimalisering av HOG og trace deskriptorene er gjort i Kapittel 5.5.1 og 5.5.2. Testene som skal utføres er gitt som følgende.

- Test 1: Sammenligning av HOG og trace deskriptor
- Test 2: Sammenligning ved lysendring
- Test 3: Sammenligning ved okklusjon
- Test 4: Sammenligning ved geometrisk transformasjon

Hvis systemet skulle blitt realisert i praksis, vil antallet kjøretøyfronter det sammenlignes mot, variere avhengig av trafikkflyten og tunnellengden. I følgende analyse av deskriptorene vil det derfor være hensiktsmessig å variere antallet kjøretøykandidater det matches mot i form av et matchvindu. Et lite matchvindu vil gi en større sannsynlighet å matche korrekte kjøretøy grunnet færre kandidater. I Figur 5.23 vises matching av frontutsnitt med en vindustørrelse på fire.

Kjøretøyfrontene som er brukt i testene er filmet ved ulike tider på døgnet og forskjellige kamerainnstillinger for å simulere de naturlige utfordringene som vil forekomme når systemer



Figur 5.23 Her vises matching av kjøretøyfront mot en kjøretøygruppe med vindustørrelse på fire.

kjøres i sanntid. I et sanntidssystemet skal bare kjøretøyfronter av samme kjøretøytype sammenlignes med hverandre. Siden vi er avhengig av et stort datasett for å gjøre en grundig test av metodene, har det blitt valgt å kun utføre test av frontsammenligning for lette kjøretøytyper. Forklaring av datasettet brukt i testen er gjort i Figur 5.5.

5.5.1 Optimalisering av HOG deskriptoren

Ved optimalisering av HOG deskriptoren, vil det fokuseres på å innstille parametre slik at flest kjøretøyfronter blir matchet riktig for datasettet beskrevet i Figur 5.5. Redusering av deskriptorens tidsbruk er også en viktig del av optimaliseringen. For å oppnå dette må følgende parametre endres:

- Gaussisk glattefilter med parameter σ.
- Bildestørrelse ved re-skalering.
- HOG parametre: cellestørrelse og blokkstørrelse.

Matlabkoden som brukes til detektering av HOG egenskaper, bruker en 1D-punkt diskret derivat-maske forklart i likning (2.16). Siden kantdeteksjon gjøres gjennom differensiering, er det sensitivt for støy. For å motvirke dette, er det vanlig å glatte bildet med et gaussisk filter. Kamera 1 filmer uskarpe bilder og trenger derfor å glattes mindre. Graden av bildeglattingen

for hvert kamera, er bestemt av parameterne σ_1 og σ_2. Kombinasjonen av standardavvikene som gir best resultat er bestemt ut fra en test der 100 frontutsnittpar sammenlignes. Resultatet er vist i Tabell 5.5.

Tabell 5.5 Her vises antall kjøretøy som matches korrekt ved ulike standardavvik for et gaussisk glattefilter. Ulike standardavvik brukes for opptak gjort av Kamera 1 og 2.

	$\sigma_2 = 1.9$	$\sigma_2 = 2.1$	$\sigma_2 = 2.3$	$\sigma_2 = 2.5$	$\sigma_2 = 2.7$	$\sigma_2 = 2.9$
$\sigma_1 = 0.5$	92	93	93	93	91	91
$\sigma_1 = 0.7$	92	93	**94**	**94**	91	91
$\sigma_1 = 0.9$	92	**94**	**94**	**94**	**94**	**94**
$\sigma_1 = 1.1$	92	**94**	93	93	93	93
$\sigma_1 = 1.3$	92	92	93	93	93	93

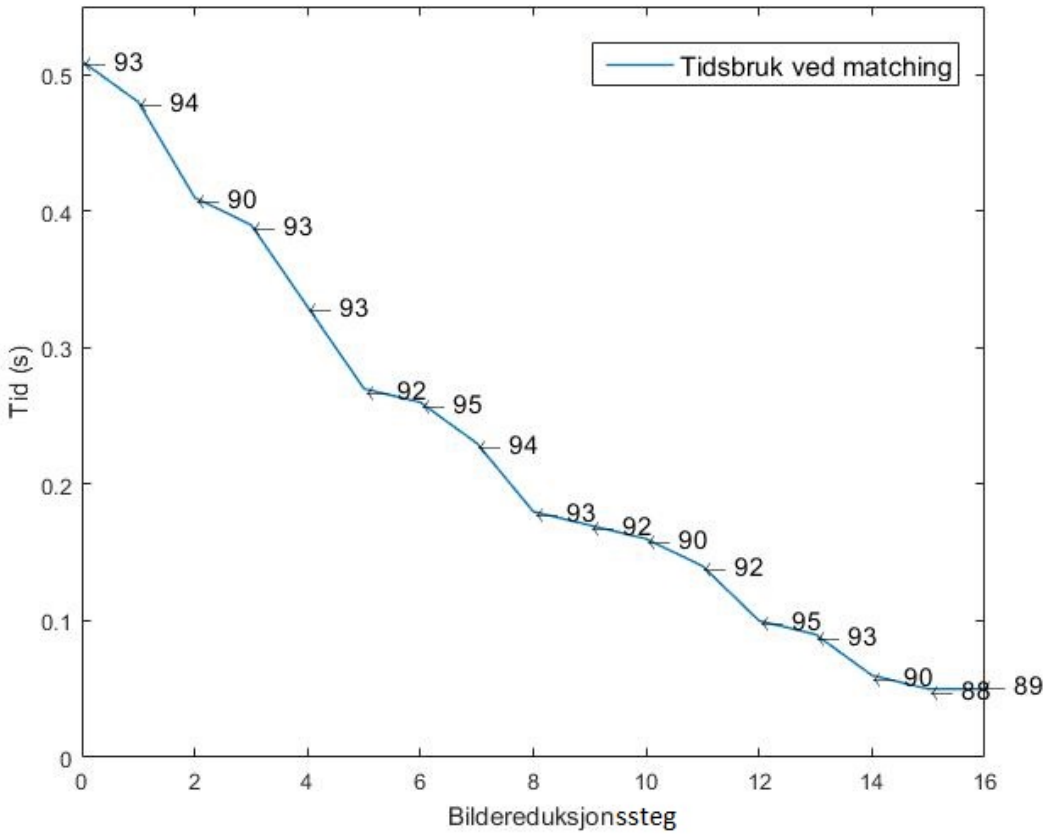
Resultatet i Tabell 5.5 viser at det må brukes en høy glattekoeffisient for opptak gjort av Kamera 2 grunnet et skarpere bilde. Koeffisientene $\sigma_1 = 0.9$ og $\sigma_2 = 2.5$ gir høyest antall korrekt match og velges som parametere i en optimalisert deskriptor.

Før en beregning av HOG deskriptor kan utføres, må bildestørrelsen normaliseres. En liten bildestørrelse reduserer lengden på HOG deskriptoren med fordel å kunne utføre en raskere matching. Matching av deskriptorer gjøres ved normalisert-krysskorrelasjon, se likning (2.21), og kan være beregningsmessig krevende når en korrelasjonskoeffisient skal beregnes for alle skift. Her gjøres det derfor en test for vurdering av hvor mye bildestørrelsen kan reduseres uten å minske sannsynligheten for korrekt match.

Segmentering av panserutsnitt ved bruk av parametrene beskrevet i likning (4.7), gir et tilnærmet størrelseforhold der bredden på panserbildet er tre ganger større en høyden. Største frontutsnittstørrelse hentet ut fra opptak er på 219×75 piksler. I testen skal alle frontutsnitt i utgangspunkt skaleres til denne bildestørrelsen for å beregne tiden det tar å matche en HOG deskriptor mot 100 andre kandidater. Denne bildestørrelsen skal gradvis reduseres for hvert steg med 10 piksler i x- og et antall piksler i y-retning som bevarer ønsket størrelseforhold. Resultatet av testen er vist i Figur 5.24. I Tabell 5.6 vises bildestørrelsen av frontutsnittet ved hver reduksjon.

Tabell 5.6 Her vises størrelsen på bildet og 1D-HOG deskriptoren ved hver reduksjon. For beregning av HOG deskriptoren er cellestørrelse 10×10, blokkstørrelse 2×2 og antall histogramkanaler satt til 9.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
x	220	210	200	190	180	170	160	150	140	130	120	110	100	90	80	70	60
y	73	70	67	63	60	57	53	50	47	43	40	37	33	30	27	23	20
1D	4536	4320	3420	3240	3060	2304	2160	2016	1404	1296	1188	720	648	576	252	216	180
t (s)	0.51	0.48	0.41	0.39	0.33	0.27	0.26	0.23	0.18	0.17	0.16	0.14	0.1	0.09	0.06	0.05	0.04



Figur 5.24 Her vises tiden det tar å matche 1 kjøretøyfront mot 100 andre ved forskjellige bildestørrelser. Bildestørrelsen ved forskjellig steg er forklart i Tabell 5.6. Tallene ved den blå linjen viser antall korrekte match ved sammenligning.

Resultatet viser at bildestørrelsen i liten grad påvirker presisjonen ved matchingen, men gir derimot et stort utslag i kjøretiden grunnet til drastisk reduksjon av størrelsen på deskriptoren. Bildestørrelse $x = 160$ og $y = 53$ gir et godt resultat. Større reduksjon kan være risikabelt da detaljer vil kunne unnlates.

Parametrene til HOG deskriptoren må videre justeres for å tilpasses bildestørrelsen. Parametere som cellestørrelse og blokkstørrelse må bestemmes. En økning av cellestørrelsen kan bidra til å miste småskala detaljer. En liten blokkstørrelse hjelper å dempe virkninger av lysforandringer på HOG egenskapene [54]. Antall histogramkanaler er vanlig å sette til 9 [13], og vil derfor ikke bli justert. I Tabell 5.7 vises antall kjøretøy av 100 som matches korrekt ved ulike parametre. Her gir cellestørrelse på 10×10 og blokkstørrelse på 2×2 best resultat.

Tabell 5.7 Her vises resultat av antall kjøretøy som matches korrekt ved ulike justeringsparametre.

Cellestørrelse	Blokkstørrelse	Korrekt antall match
[8 8]	[2 2]	89
[8 8]	[4 4]	92
[10 10]	[2 2]	95
[10 10]	[4 4]	93
[12 12]	[2 2]	93
[12 12]	[4 4]	91

5.5.2 Optimalisering av trace transform deskriptoren

Trace transformen blir optimalisert for å oppnå en høy matchprosent ved sammenligning og samtidig redusere kjøretiden. Sistnevnte er svært viktig der trace transformen er kjent å være beregningsmessig krevende. For å oppnå dette, må følgende parametre justeres:

- Parameter σ for gaussisk glattefilter.
- Reduksjon av bildestørrelse

Før en beregning av trace deskriptoren gjøres, må bildet glattes for å gjevne ut støy. Resultat av gaussisk glatting med forskjellige σ verdier er vist i Tabell 5.8.

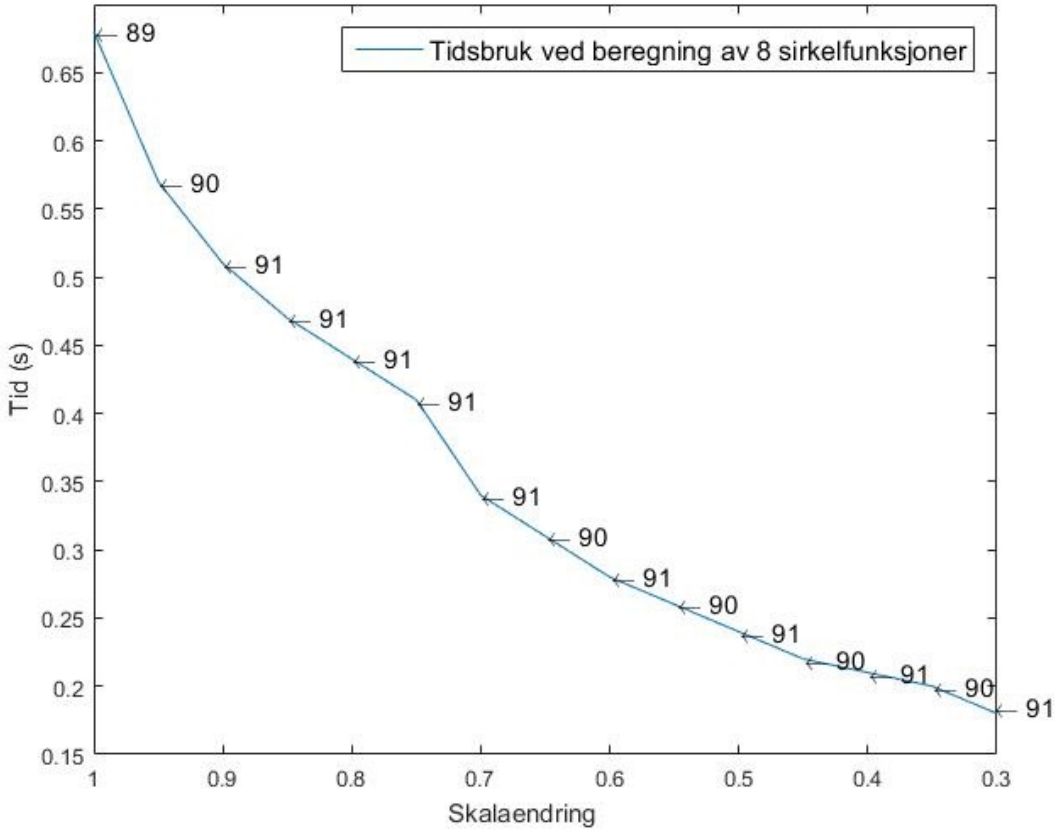
Tabell 5.8 Her vises antall kjøretøy som matches korrekt ved ulike glattekoeffisienter for et gaussisk glattefilter.

	$\sigma_2 = 1.5$	$\sigma_2 = 1.7$	$\sigma_2 = 1.9$	$\sigma_2 = 2.1$	$\sigma_2 = 2.3$
$\sigma_1 = 0.3$	85	88	88	88	87
$\sigma_1 = 0.5$	86	**89**	88	**89**	87
$\sigma_1 = 0.7$	86	88	**89**	**89**	87
$\sigma_1 = 0.9$	86	86	88	**89**	87
$\sigma_1 = 1.1$	84	87	88	88	88
$\sigma_1 = 1.3$	83	85	85	88	88

Ut ifra resultatet i Tabell 5.8, er det valgt å bruke $\sigma_1 = 0.7$ og $\sigma_2 = 2.1$.

Lengden av trace transform deskriptoren er uavhengig av bildestørrelsen, men til gjengjeld er trace transformen beregningsmessig krevende og en liten bildestørrelse vil derfor gi en stor fordel. Her gjøres det en test for å se hvor mye bildestørrelsen kan reduseres uten å påvirke resultatet. Her er det viktig at størrelseforholdet på bildet mellom bredde og høyde bevares selv når bildet nedskaleres. I denne testen er det valg å bruke 8 sirkelfunksjoner som gir en lenge på deskriptoren på 8×120. Tidsbuken for beregning av egenskapsvektorene ved

ulike bildestørrelser, er vist i Figur 5.25 og Tabell 5.9.



Figur 5.25 Her vises tiden det tar å beregne trace transform deskriptor av en kjøretøyfront ved forskjellige bildestørrelser. Egenskapsvektoren består av 8 sirkelfunksjoner. Her brukes det i utgangspunktet en bildestørrelse på $x = 219$ og $y = 75$ piksler som reduseres prosentvis. Bildestørrelsen ved forskjellig skalaendring er forklart i Tabell 5.9. Tallene ved den blå linjen viser antall korrekte match ved sammenligning.

Tabell 5.9 Denne tabellen viser tiden det tar å beregne en trace transform egenskapsvektor m.h.p. bildestørrelsen. Her er 8 sirkelfunksjoner brukt for å beregne en egenskapsvektor ved ulike bildestørrelser. Frontutsnittbildet skaleres ned 5% for hvert steg.

	1	**0.95**	**0.90**	**0.85**	**0.80**	**0.75**	**0.70**	**0.65**	**0.60**	**0.55**	**0.50**	**0.45**	**0.4**	**0.35**	**0.30**
x	219	208	197	186	175	164	153	142	132	121	110	99	88	77	66
y	75	71	68	64	60	56	53	49	45	41	38	34	30	26	23
t (s)	0.68	0.57	0.51	0.47	0.44	0.41	0.34	0.31	0.28	0.26	0.24	0.22	0.21	0.20	0.18

Resultatet viser at endringen i bildestørrelsen gir liten innvirkning på nøyaktigheten av frontsammenligningen. Ved en panserutsnittstørrelse på 110×38, tar det 0.24 sekund å beregne 8 sirkelsignaturer. Ved en mindre bildestørrelse, kan flere detaljer forsvinne. Derfor er det valgt å bruke denne bildestørrelsen ved optimalisert trace transform deskriptor.

5.5.3 Test 1: Sammenligning av HOG og trace deskriptor

I denne testen skal de fire forskjellige deskriptorer som vises under brukes til å matche 100 frontutsnittpar.

1. HOG
2. Trace transform med 8 sirkelfunksjoner
3. Trace transform med 21 sirkelfunksjoner
4. HOG + trace med 8 sirkelfunksjoner

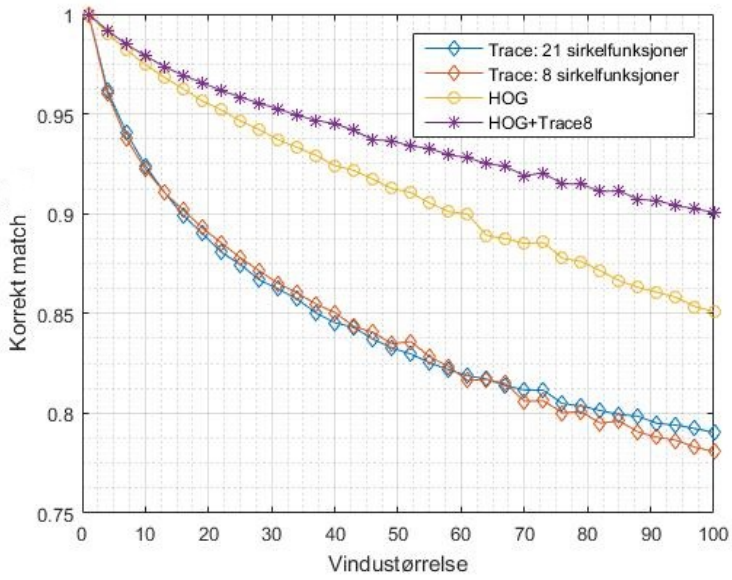
Deskriptorene som opptrer i denne testen skal vurderes ut fra:

- Sannsynligheten for korrekt match ved sammenligning (matchprosent).
- Kjøretid for beregning og matching av deskriptorene.
- Deskriptoren som viser til størst rotasjonsinvarians.

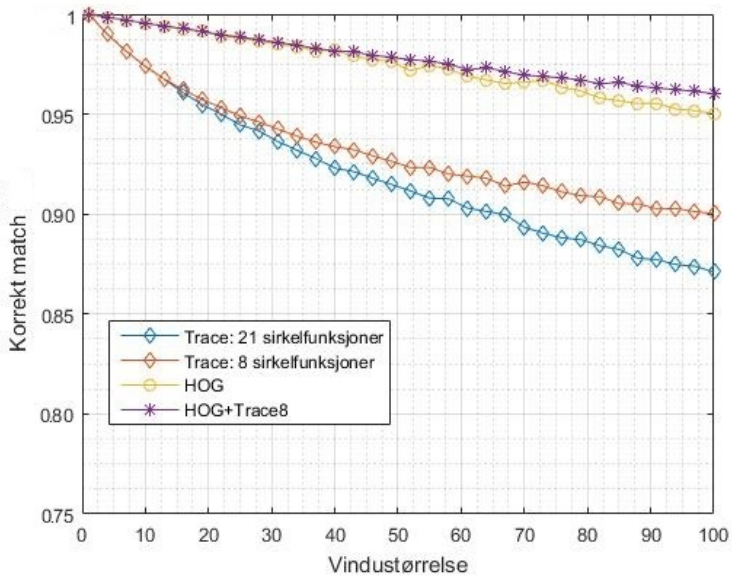
For å teste deskriptorens evne til å håndtere rotasjoner av kjøretøyobjektet, skal to ulike bildegrupper som hver inneholder 100 frontutsnittpar forsøkes matches. I den første bildegruppen er frontutsnittene rotasjonsnormalisert. I den andre gruppen er frontutsnittene ikke rotasjonsnormalisert. Bildesettet fra Kamera 1 er blitt rotasjonsnormalisert ved å rotere frontutsnittene 4° om x-aksen. Deskriptoren som gir minst endring i antall korrekte match når rotasjonen i bildene normaliseres, blir vurdert å gi best rotasjonsinvarians. Når antallet kandidater ett frontutsnitt sammenlignes med øker, reduseres sannsynligheten for korrekt match. I testen blir derfor antallet kandidater det matches mot variert i form av vindustørrelse. Frontutsnittene er hentet ut fra opptaket kalt Frontgruppe 1 og forklart i Figur 5.5.

Resultat

Resultatet av testen med og uten rotasjonsnormalisering er vist i Figur 5.26. Resultatet viser at selv uten rotasjonsnormalisering, kan en høy matchprosent oppnås. Når rotasjonen av frontene normaliseres i henhold til hverandre, øker sannsynligheten for korrekt match betraktelig. En nøyaktig normalisering blir derfor en viktig del av prosessen for å oppnå en høy grad av kjøretøygjenkjenning.



(a) Sannsynlighet for korrekt match ved forskjellig størrelse av match-vinduet. Frontutsnittene er ikke rotasjonsnormalisert.



(b) Sannsynlighet for korrekt match ved forskjellig størrelse av match-vinduet. Her er frontutsnittene blitt rotasjonssnormalisert.

Figur 5.26 Resultat av frontmatching ved ulike deskriptorer.

Deskriptoren som gir høyest sannsynlighet for korrekt match er en kombinasjon av HOG og trace transformen som videre kalles for ”HOG + Trace 8”. Denne kombinasjonen gir best resultat fordi begge deskriptorene henter ut forskjellige bildeegenskaper. HOG deskriptoren

ser bare på kantstrukturen i bildet mens trace deskriptoren inneholder informasjon om intensiteten. Ulempen med denne fremgangsmåten er at trace transformen alene gir et dårligere resultat sammenlignet med HOG og kan derfor ved enkelte sammenligninger påvirke resultatet negativt. Dette særlig i tilfeller der en sammenligning av HOG deskriptorer gir en liten avstand mellom gjennomsnittet for korrekt og feil korrelasjonskoeffisient.

Alene gir HOG et godt resultat sammenlignet med trace transformen. 95 av 100 kjøretøyfronter matchet korrekt. Feil skyldes i hovedsak at kjøretøyfronter med lik kantstruktur feilaktig ble matchet sammen. Trace transformen gav et dårligere resultat fordi den lettere ble påvirket av kontrastendringene forårsaket av forskjellige kamerainnstillinger. Dette er fordi trace transformen beregner funksjoner langs kutt i bilder og endringene i intensiteten vil gi utslag som påvirker resultatet negativt.

21 sirkelfunksjoner ved trace transformen henter ut mer informasjon om bildet enn 8 sirkelfunksjoner. Dette skal i teorien gi et bedre resultat. Dette viste seg å ikke være tilfellet da bruk av 8 sirkelfunksjoner ga et likt eller bedre resultat. Dette kan skyldes at flere sirkelfunksjoner bærer på lik informasjon, og kan sees på som overflødig fordi de ikke bidrar med nye ”bevis” til å kunne skille mellom ulike bilder. Derfor gir noen sirkelfunksjoner en høy korrelasjonskoeffisient med hverandre. De overflødige signaturene kan derfor føre til å bedra likheten i deskriptoren. Samme resultat er også rapportert i A. F. Velazquea, et. al. [89]. I Tabell 5.10 vises kjøretiden for beregning av deskriptor og korrelasjonskoeffisient.

Tabell 5.10 Her vises kjøretiden ved beregning av en enkel deskriptor og ved matching av to deskriptorer gjennom NCC. Tegnet * indikerer at kjøretøyutsnittene ikke er rotasjonsnormalisert.

	HOG	**Trace8**	**Trace21**	**HOG + Trace8**
*Korrekt match (100 kjøretøypar)	85%	78%	79%	90%
Korrekt match: (100 kjøretøypar)	95%	90%	87%	96%
Tid (s): Per deskriptor	0.0036	0.3045	0.4792	0.3081
Tid (s): Match 1 mot 1	0.0022	0.0040	0.0115	0.0062

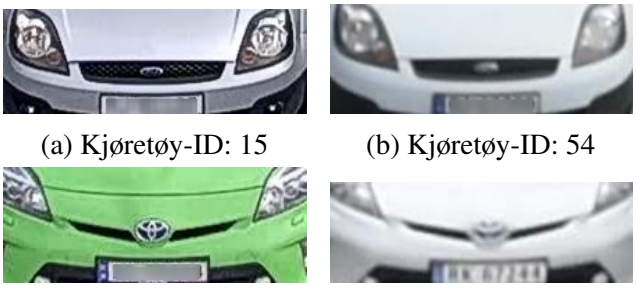
I følge Tabell 5.10, kan man se at selv om ikke matchprosenten økte ved bruk av 21 sirkelfunksjoner, økte derimot beregningskostnadene betraktelig sammenlignet med 8 funksjoner. HOG er på den andre siden en veldig rask metode med tidsbruk på 0.0036 sekund per deskriptor. Utfordringen ved kjøretiden til HOG kan oppstå når en lang deskriptor beregnes ut fra en stor bildestørrelse. Denne kan ta lang tid å matche ved normalisert-krysskorrelasjon.

Resultatet i Tabell 5.11 viser at HOG har høyest økning i sikkerhetsmarginen når bilderotasjonen normaliseres. Dette kan indikere at deskriptoren har minst rotasjonsinvarians. Sikkerhetsmarginen er definert her som avstanden mellom middelverdien for korrelasjonskoeffisientene som svarer til korrekt og ukorrekt match. HOG er også den deskriptoren som

Tabell 5.11 Her vises to gjennomsnittsverdier for korrelasjonskoeffisienter beregnet ut fra matching av korrekt og feil kjøretøypar. Målet er å vise hvor stor avstand det er mellom vekten av korrekt og feil korrelasjonskoeffisienter. En stor avstand gir en høy sikkerhetsmargin ved sammenligning som betyr at matchingen utføres med stor sikkerhet.

	HOG	Trace8	Trace21	HOG + Trace8
Uten rotasjon normalisering:				
Korrekt korr.koeff. gj.sn.	0.8187	0.7807	0.8411	0.7997
Feil korr.koeff. gj.sn.	0.5890	0.5717	0.6682	0.5803
Sikkerhetsmargin (1)	0.2297	0.2090	0.1729	0.2194
Med rotasjon normalisering:				
Korrekt korr.koeff. gj.sn	0.8695	0.8202	0.8737	0.8449
Feil korr.koeff. gj.sn.	0.5963	0.5760	0.6736	0.5862
Sikkerhetsmargin (2)	0.2731	0.2442	0.2001	0.2587
Økning i sikkerhetsmargin:				
(1)-(2):	0.0434	0.0352	0.0272	0.0393

viser til størst sikkerhetsmargin. Dette betyr at de kjøretøyfrontene som matches korrekt, matches med stor grad av sikkerhet. Med en høy sikkerhetsmargin, vil deskriptoren kunne tåle større bildetransformasjoner før bilder matches feil. I Figur 5.27 vises noen av frontbildene som matches feil ved HOG deskriptor og i Figur 5.28 vises det samme for trace deskriptor.



(a) Kjøretøy-ID: 15 (b) Kjøretøy-ID: 54

(c) Kjøretøy-ID: 22 (d) Kjøretøy-ID: 97

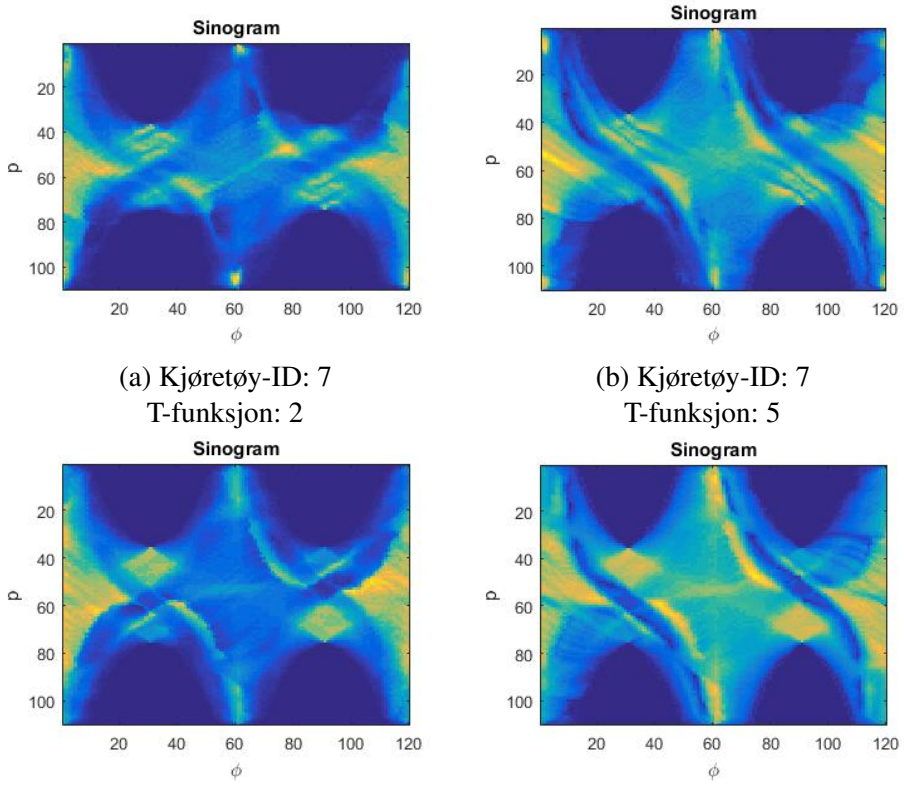
Figur 5.27 Frontutsnitt som gir feil ved matching av HOG deskriptor. Kjøretøy-ID 15 blir feilaktig matchet med Kjøretøy-ID 54 der den ene bilen er sølv og den andre er hvit. Kjøretøy-ID 22 blir feilaktig matchet med ID 97

(a) Kjøretøy-ID: 7 (b) Kjøretøy-ID: 72

(c) Kjøretøy-ID: 68 (d) Kjøretøy-ID: 56

Figur 5.28 Frontutsnitt som gir feil ved matching av trace deskriptor. Kjøretøy-ID 7 blir feilaktig matchet med kjøretøy-ID 72. Kjøretøy-ID 68 blir feilaktig matchet med ID 56.

En typisk grunn til at frontutsnittspar blir matchet feil ved HOG skyldes i hovedsak at frontutsnittene har like kantlinjer. Siden HOG deskriptoren ikke henter informasjon om fargeintensiteten, blir ofte kjøretøyfronter som har lik gradientstruktur, men ulik farge forvekslet med hverandre. Et typisk frontutsnittspar som feilaktig blir matchet sammen ved trace transformen, skyldes at de har lik intensitet i deler av bildet. Dette kan bedre illustreres ved å se på sinogrammet av kjøretøyfronten i Figur 5.29. Her vises et sinogram beregnet fra T-funksjonene 2 og 5 (Tabell 2.1) av kjøretøyfrontene ID 7 og ID 72 vist i Figur 5.28. Disse frontene er blitt feilaktig matchet sammen.



(a) Kjøretøy-ID: 7
T-funksjon: 2

(b) Kjøretøy-ID: 7
T-funksjon: 5

(c) Kjøretøy-ID: 72
T-funksjon: 2

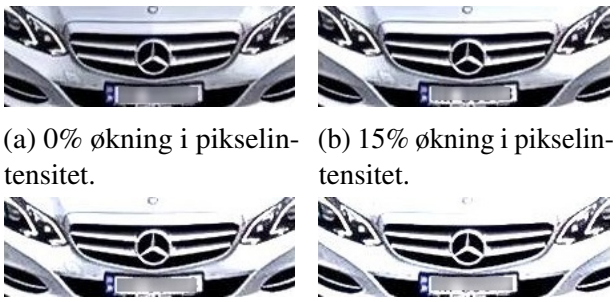
(d) Kjøretøy-ID: 72
T-funksjon: 5

Figur 5.29 Trace visualisering i 3D-rom for kjøretøy ved ID 7 og 72. Grunnet ferdigfunksjonen i Matlab som er brukt til visualisering av bildene, er ikke aksene helt korrekte. Steget for kuttvinkelen er satt til 3°, dermed svarer ϕ-aksen til $[0, 360]$ grader (selv om den vises $[0, 120]$ i figuren). P-aksen svarer til $[-55, 55]$ der 0 er midten av aksen (i figuren vises aksen som $[0, 110]$ der 110 er bildebredden). Visualiseringen viser responsen for en gitt T-funksjon ved ulike verdier av P og ϕ. Gul farge viser områder med høy intensitet, altså der T-funksjonen gir høy respons. De gule områdene på høyre og venstre siden av sinogrammet svarer til bilskiltet. Her ser vi at sinogrammene for begge panserutsnittene har svært lik struktur for T-funksjonene 2 og 5. Kun i enkelte områder er det en forskjell i intensiteten og derfor blir de feilaktig betraktet som like.

5.5.4 Test 2: Sammenligning ved lysendring

Lysendringer som oppstår når kjøretøy detekteres ved forskjellige lokasjoner, er en av de største utfordringene for et system som realiseres i praksis. Derfor er det viktig at deskriptorene som brukes, gir en god robusthet mot lysendringer. Denne testen har som mål å vurdere trace og HOG deskriptorens evne til å takle lysendringer. Lysendringene er oppnådd i testen ved å øke pikselintensiteten prosentvis fra $0 - 80\%$ på alle frontutsnittbildene hentet fra Kamera 2. Dermed oppstår et vesentlig skille i lysintensiteten mellom bildeparene. Et

godt resultat her vil gi en liten grad av endring i antall korrekte match når forskjellen i lysintensiteten øker. Frontutsnittene brukt i testen er forklart i Figur 5.5 og er de samme som er brukt i Test 1. Endring av pikselintensiteten med 0%, 15%, 40%og 70% er vist i Figur 5.30.



(a) 0% økning i pikselintensitet.

(b) 15% økning i pikselintensitet.

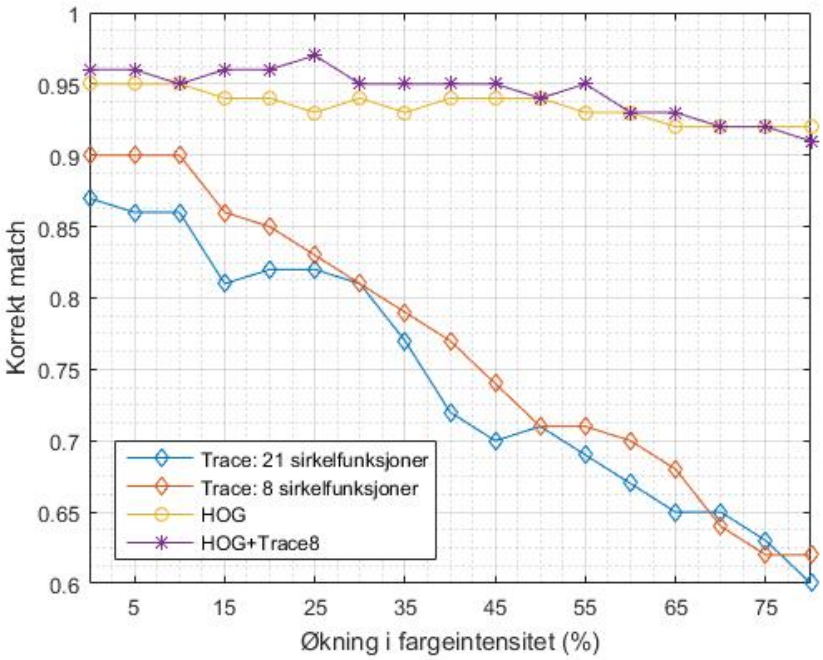
(c) 30% økning i pikselintensitet.

(d) 70% økning i pikselintensitet.

Figur 5.30 Lysendring.

Resultat

Resultatet for testen er vist i Figur 5.31 og i Tabell 5.12.



Figur 5.31 Frontsammenligning ved en gradvis økning av lysstyrken.

Når pikselverdien økes prosentvis i bildet, blir økningen i pikselintensiteten størst ved piksler som allerede har en høy verdi. Dermed øker skillen mellom lyse og mørke områder.

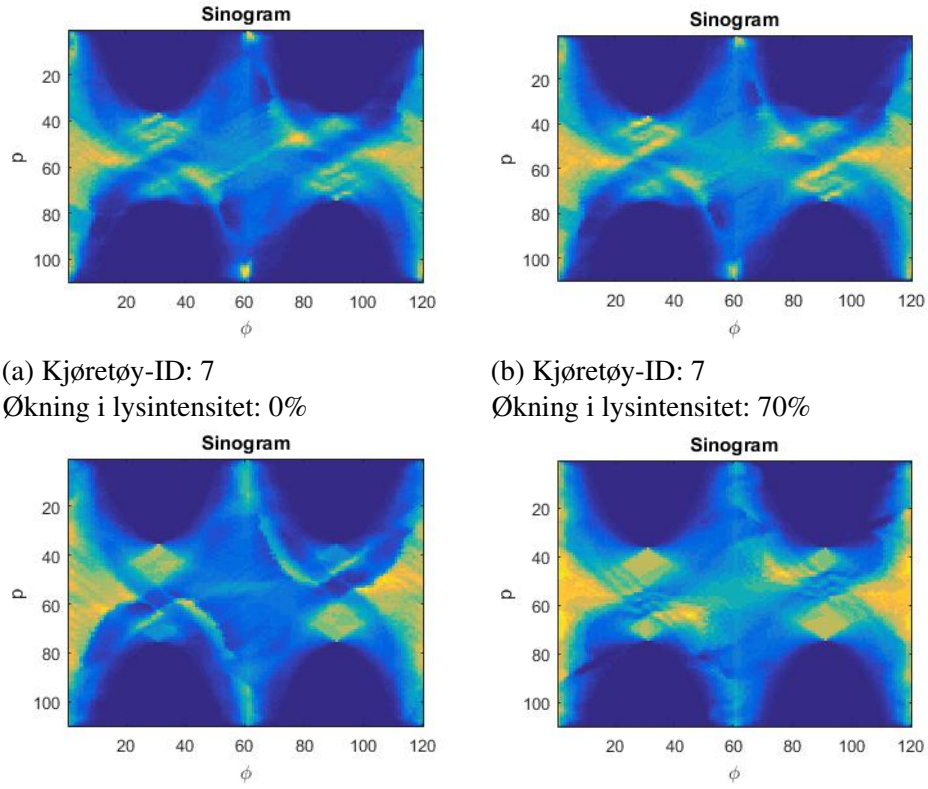
Tabell 5.12 Her vises endringen i antall kjøretøy som gjenkjennes når lysstyrken øker.

Lysendring [%]	**0**	**15**	**30**	**45**	**60**	**75**	**80**
Korrekt match [%]							
HOG	95	94	94	94	93	92	92
Trace 8	90	86	83	74	71	62	62
Trace 21	87	81	81	70	67	63	60
HOG + Trace 8	96	96	95	95	93	92	91
Reduksjon [%]							
HOG	0	-1	-1	-1	-2	-3	-3
Trace 8	0	-4	-7	-16	-19	-28	-28
Trace 21	0	-6	-6	-17	-20	-24	-27
HOG + Trace	0	0	-1	-1	-3	-4	-5

Endringene i pikselverdiene påvirker trace signaturene som beregnes langs kuttlinjene i bildet der flere områder får en økt respons som er vist i Figur 5.32. En økning i lysintensiteten skaper derfor et større skille mellom bildeparene og fører til at langt færre kjøretøy matches korrekt enn tidligere.

HOG henter ut egenskaper som forteller noe om bildegradienten. Denne blir i liten grad påvirket ved en global økning av lysintensiteten. Dette er noe resultatet i Tabell 5.12 bekrefter. Ved en 60% økning av lysintensiteten, blir bare 3% færre kjøretøyfronter gjenkjent. Økningen i lysintensiteten fører bare til at de svake gradientene forsvinner mens de sterkeste forblir uendret. Dermed oppnår HOG en god robuthet mot lysendringer.

HOG + Trace 8 deskriptoren gir her et interessant resultat. Trace transformen er mindre lysinvariant enn HOG og det var forventet et svakere resultat sammenlignet med å kun bruke HOG alene. I stedet er ytelsen ved denne kombinasjonen lik som ved å kun bruke HOG alene når pikselintensiteten økes med 75%. I testen danner HOG en høy korrelasjonskoeffisient når korrekte kjøretøyfronter matches og får derfor større innflytelse sammenlignet med trace transformen der avstanden mellom korrekte og ikke korrekte korrelasjonskoeffisienter minker sterkt ved lysendringer. Derfor får korrelasjonskoeffisienten til HOG en større vekt enn ved sammenligning av trace deskriptor. Sistnevnte får bare innflytelse i denne testen der HOG er ”usikker” og danner flere høye korrelasjonskoeffisienter mellom kjøretøyfronter som ligner. I tillegg har begge metodene sine fordeler og ulemper ved ulike typer av kjøretøyfronter. Resultatet er derfor i stor grad avheng av datasettet som brukes. Derfor er det for tidlig å konkludere at HOG og trace kombinasjon her vil gi en god ytelse under lysendringer uten videre testing.



(a) Kjøretøy-ID: 7
Økning i lysintensitet: 0%

(b) Kjøretøy-ID: 7
Økning i lysintensitet: 70%

(c) Kjøretøy-ID: 72
Økning i lysintensitet: 0%

(d) Kjøretøy-ID: 72
Økning i lysintensitet: 70%

Figur 5.32 Her vises hvordan økningen i lysintensiteten påvirker sinogrammet som er beregnet ut fra T-funksjon 2 (Tabell 2.1). Sinogrammet for kjøretøy ID 7 blir i liten grad påvirket når lysintensiteten økes. Dette kan sees på bildet der få nye områder går fra blå til gul farge. For kjøretøy ID 72 gir lysendringen et større utslag hvor flere områder som før var blå nå blir gule. Dette skyldes at kjøretøy ID 72 er lysere enn kjøretøy ID 7. Derfor blir kjøretøy ID 72 i større grad påvirket av en økning i lysintensiteten. Bilde av kjøretøyfrontene er vist i Figur 5.27.

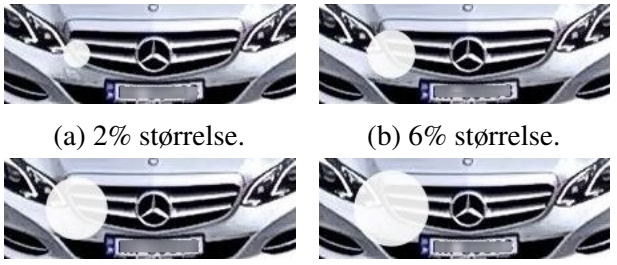
5.5.5 Test 3: Sammenligning ved okklusjon

Gjenskinn skapt av solrefleksjon kan ses på som en form av okklusjon der mye av bildeinformasjonen forsvinner. Fronten er et område på kjøretøyet som har lett for å danne refleksjoner grunnet de mange områdene og kantene som er vinklet forskjellig. I denne testen skal derfor solrefleksjon gjenskapes i form av okklusjon på frontutsnittet. Testen er delt i to der målet er å finne hvilken påvirkning okklusjon har på de forskjellige deskriptorene.

I den første delen av testen dannes en hvit sirkel på et fast område i et panserutsnittbilde som gradvis økes fra 0% til 15% av total bildestørrelse, illustrert i Figur 5.33. Dette bildet matches så med det originale der målet er å se hvor raskt korrelasjonskoeffisienten

minker når graden av okklusjon øker. Denne testen utføres for både trace transform med 8 sirkelfunksjoner og HOG. I begge tilfellene blir bildet filtrert med et gaussisk filter $\sigma = 2$.

I den neste delen av testen dannes en hvit sirkel på et fast bildeområde ved alle frontutsnitt som er hentet fra Kamera 2. Frontutsnittene er hentet fra samme datasettet som er brukt i Test 1 og 2 og er forklart i Figur 5.5. I testen skal videre okklusjonen økes i størrelse fra 0% til 15% av total bildestørrelse og vi er her interessert å se hvilken innvirkning dette får på matchprosenten når disse matches mot frontene fra Kamera 1.



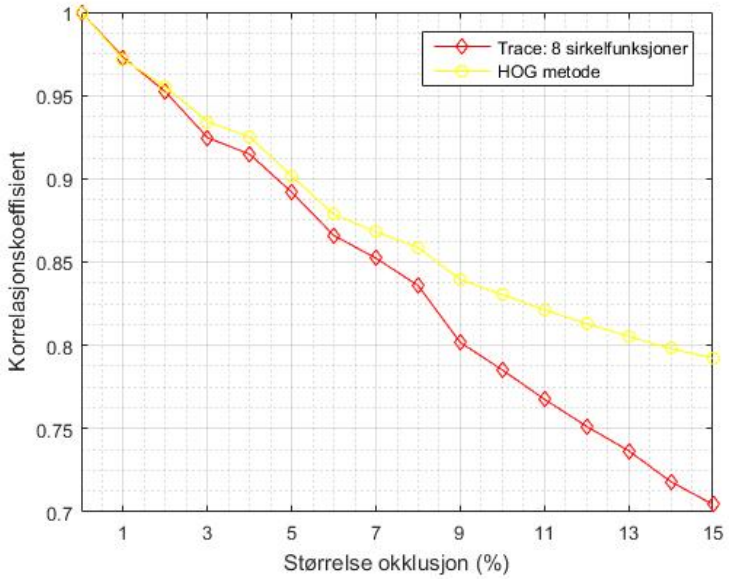
(a) 2% størrelse. (b) 6% størrelse.

(c) 10% størrelse. (d) 15% størrelse.

Figur 5.33 Første test: Lysrefleksjoner ved kjøretøyfronten kan ses på som en form av okklusjon.

Resultat

Resultatet av den første delen av testen er vist i Figur 5.34. Her vises hvordan korrelasjonskoeffisienten endres når størrelsen av okklusjonen øker.



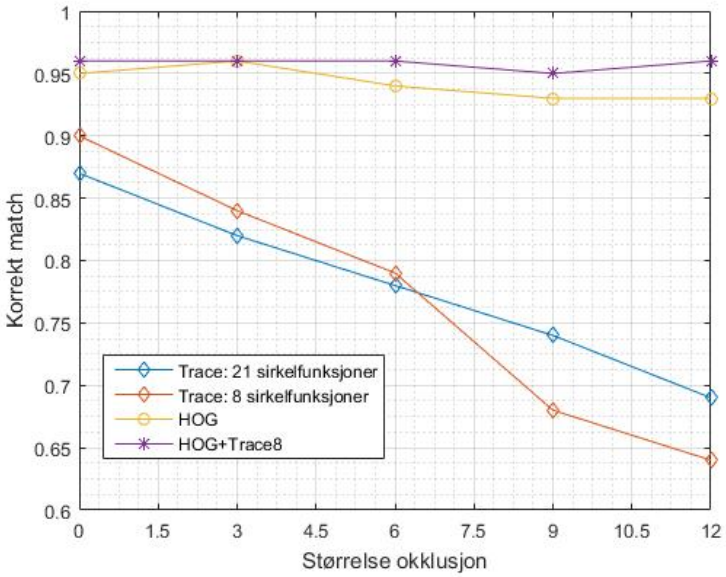
Figur 5.34 Korrelasjonskoeffisient ved en økende grad av okklusjon.

Resultatet i Figur 5.34 viser at korrelasjonskoeffisienten for HOG og trace transformen minker likt ved start når okklusjonen øker. Ved okklusjon over 7%, avtar fallhastigheten for korrelasjonskoeffisienten til HOG, mens korrelasjonskoeffisienten for trace transformen fortsetter å falle i samme lineære tempo som før. Trace transform deskriptoren blir derfor i større grad påvirket negativt av okklusjonen. Resultatet for den andre delen av testen er vist i Figur 5.35 og i Tabell 5.13.

Tabell 5.13 Her vises hvordan avstanden mellom korrekte korrelasjonskoeffisienter og ikke korrekte korrelasjonskoeffisienter som blir matchet endres når det oppstår en okklusjon på 12% av bildestørrelsen.

	HOG		Trace 8		Trace 21		HOG + Trace 8	
Grad okklusjon	0%	12%	0%	12%	0%	12%	0%	12%
Korrekt match	95%	93%	90%	64%	87%	69%	96%	96%
Korrekt korr. koeff.	0.87	0.63	0.82	0.71	0.87	0.79	0.85	0.67
Feil korr. koeff.	0.60	0.46	0.58	0.57	0.67	0.66	0.59	0.51
Sikkerhetsmargin	0.27	**0.17**	0.24	**0.14**	0.20	**0.13**	0.26	**0.16**

Resultatet i Tabell 5.13 viser at ved okklusjon blir det mindre avstand mellom korrelasjonskoeffisienten beregnet for korrekte og ukorrekte match. Dermed blir sikkerhetsmarginen

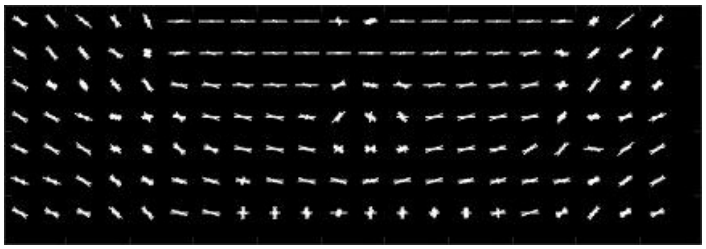


Figur 5.35 Her vises hvordan sannsynligheten for korrekt match reduseres når størrelsen på okklusjonen øker.

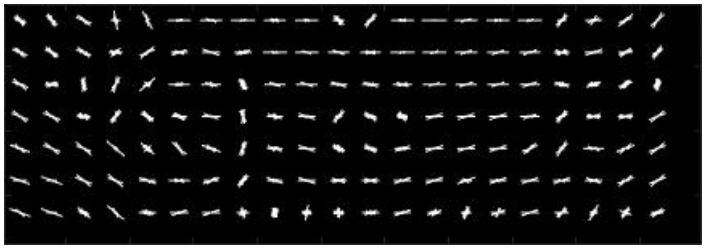
mindre og sannsynligheten for å matche feil øker. For trace transform med 8 sirkelfunksjoner faller presisjonen med 26% når okklusjonen øker til 12%. Dette viser at trace transform ikke gir en god invarians mot okklusjon. Dette er fordi linjene som skjærer gjennom okklusjonen ved beregning av transformen, får en høyere pikselintensitet, noe som påvirker responsen i sirkelfunksjonene.

Fra resultatet i Figur 5.35, gir HOG og HOG+Trace8 igjen best resultat. Grunnen til at HOG gjør det bra i denne testen er fordi alle kjøretøyfrontene i lik grad påvirkes av okklusjon. Størrelsen og posisjonen av okklusjonen er lik for alle kjøretøyfrontene og derfor blir informasjonen om gradientorienteringen i området overflødig fordi det ikke bidrar til å skille bildene fra hverandre. I et system som realiseres i praksis, vil graden av lysrefleksjoner variere for de ulike kjøretøyfontene. Dette er fordi forskjellige frontstrukturer reflekterer lyset ulikt. Denne tilfeldigheten gjør at i et ekte system kan lysrefleksjoner (okklusjon) medføre et større problem for HOG deskriptoren enn hva resultatet i testen tilsier. I Figur 5.36 vises det hvordan okklusjonen påvirker retningen til gradienten ved HOG deskriptor.

Best resultat gir igjen en kombinasjon av HOG og trace transformen. En mulig grunn for dette er at resultatet av HOG veies mer enn trace transform, grunnet til en sterkere korrelasjonskoeffisient ved korrekt match. Dette resultatet er også rapportert i Test 2: lysendring og forklares nærmere der. Videre testing må gjøres for å konkludere om det gode resultatet skyldes tilfeldigheter knyttet til datasettet eller om denne kombinasjonen av deskriptorer kan gi et bedre resultat enn ved HOG alene.



(a) Uten okklusjon



(b) 12% okklusjon

Figur 5.36 Bildet viser gradientorienteringen i lokale histogrammer for et bilde med og uten okklusjon. HOG deskriptoren er beregnet fra kjøretøyfronten vist i Figur 5.33. I venstre del av nederste bildet, vises det tydelig hvordan okklusjonen påvirker retningen på gradienten.

5.5.6 Test 4: Sammenligning ved geometrisk transformasjon

Grunnet svingninger i veibanen vil kameraene i hver sin ende av tunnelen være rettet forskjellig mot veibanen med ulik planvinkel θ. Denne vinkelen definerer kameraets rotasjon om veiplanet og er illustrert i Figur 5.2. Når kameraene har en forskjellig vinkel θ skapes en geometrisk transformasjon av kjøretøyobjektet mellom opptakene. I denne testen skal planvinkelen mellom kameraene økes, med mål å se hvordan dette påvirket sannsynligheten for at kjøretøyene matches korrekt. Under opptak, ble en økning av vinkelen oppnådd ved å øke avstanden mellom kameraene. Her ble det gjort opptak ved tre forskjellige kameraavstander. For hver av avstandene ble 50 frontutsnittspar segmentert ut. Segmenteringen foregikk manuelt ved å først foreta en rotasjonsnormalisering av bildet. Frontutsnittene deles inn i tre grupper bestemt ut fra kameraavstand og er forklart nærmere i Tabell 5.14. Kameraparametrene under opptakene er forklart i Figur 5.5.

Tabell 5.14 Hver kameraavstandgruppe inneholder 50 frontutsnittspar av forskjellige kjøretøy.

Kameraavstand 1	Kameraavstand 2	Kameraavstand 3
Kamera 1 $\theta = 4$	Kamera 1 $\theta = 8.5$	Kamera 1 $\theta = 13$
Kamera 2 $\theta = 0$	Kamera 2 $\theta = 0$	Kamera 2 $\theta = 0$
Kameraavstand, a = 2.12m	Kameraavstand, a= 4.23m	Kameraavstand, a= 6.29m

Her er det viktig å merke seg at de tre frontutsnittgruppene inneholder frontutsnitt fra forskjellige kjøretøypar. Derfor vil en direkte sammenligning av matchprosenten mellom de ulike kameraavstandene ikke være helt rettferdig. Derimot kan sammenligningen gi en indikasjon på hvor mye geometrisk transformasjon av kjøretøyobjektet deskriptoren takler før sannsynligheten for feil blir for stor. I Figur 5.37 vises den geometriske transformasjonen av kjøretøyet som forekommer under hver kameraavstand og i Figur 5.38 vises frontutsnittet som segmenteres etter at rotasjonsnormalisering er blitt utført.

Kamera 2:

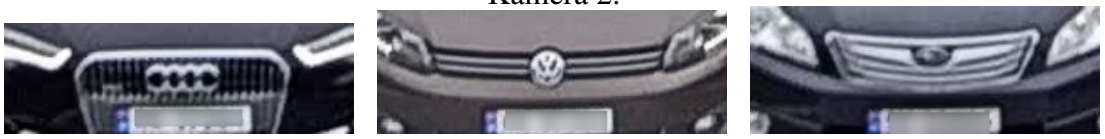
(a) Kameraavstand 1 (b) Kameraavstand 2 (c) Kameraavstand 3

Kamera 1:

(d) Kameraavstand 1 (e) Kameraavstand 2 (f) Kameraavstand 3

Figur 5.37 Her vises det hvordan forskjellige kameraavstander forårsaker geometrisk transformasjon av kjøretøyet.

Kamera 2:



(a) Kameraavstand 1 (b) Kameraavstand 2 (c) Kameraavstand 3

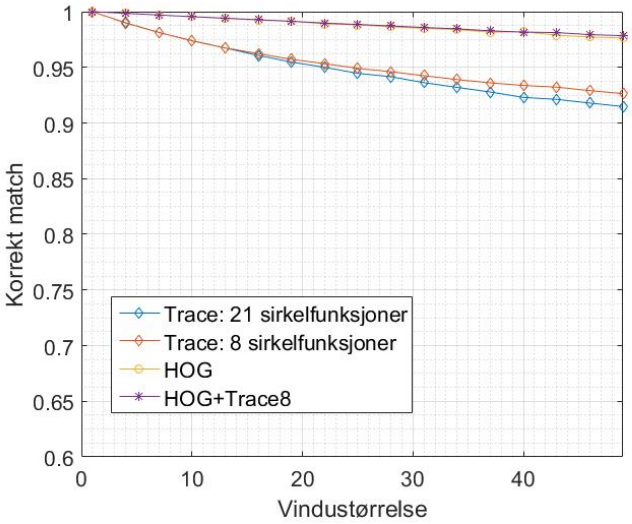
Kamera 1:

(d) Kameraavstand 1 (e) Kameraavstand 2 (f) Kameraavstand 3

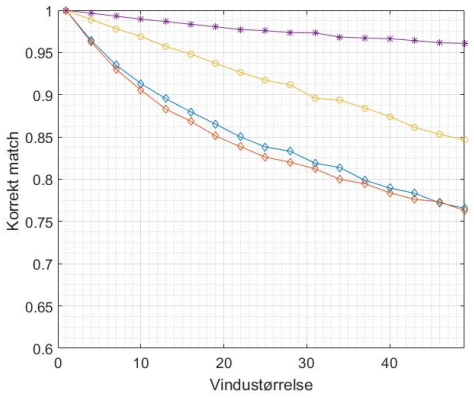
Figur 5.38 Frontutsnitt ved forskjellig kameraavstand.

Resultat

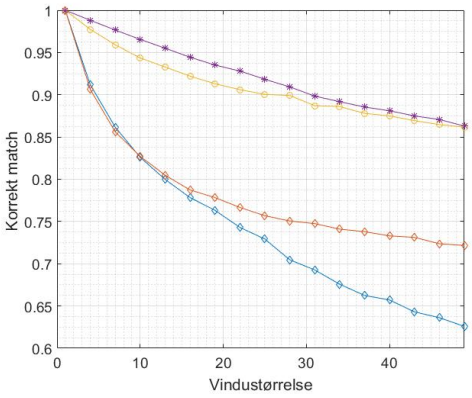
I Figur 5.39 og Tabell 5.15 vises resultatet av frontutsnittmatchingen ved tre forskjellige kameraavstander.



(a) Kameraavstand 1



(b) Kameraavstand 2



(c) Kameraavstand 3

Figur 5.39 Resultat av frontmatching ved ulik kameraavstand.

Tabell 5.15 Her vises sannsynligheten for korrekt match ved tre kameraavstander. 50 kjøretøypar er brukt i sammenligningen for hver kameraavstand.

	HOG	Trace8	Trace21	HOG + Trace8
Kameraavstand 1	97%	93%	91%	98%
Kameraavstand 2	84%	76%	76%	96%
Kameraavstand 3	86%	72%	62%	86%

Resultatet viser at en økning av kameraavstanden reduserer presisjonen ved hver av deskriptorene i testen. HOG deskriptoren gir derimot 86% match ved den største kameraavstanden, noe som kan ses på som et godt resultat grunnet den store bildetransformasjonen. For å oppnå et godt resultat med HOG, er det viktig at det gjøres en riktig rotasjonsnormalisering av bildet før frontene matches slik at gradientene får korrekt vinkel. Kombinasjonen av HOG og trace deskriptoren gir her igjen det beste resultatet grunnet "komplementære" egenskaper som samlet danner en kraftigere beskrivelse av kjøretøyfronten, selv når 8 signaturer ved trace transformen bidrar negativt og bare 72% av frontene matches korrekt.

5.5.7 Valg av struktur-utvinnende deskriptor

Basert på resultatet i Kapittel 5.5, vil det gjøres en vurdering av de ulike deskriptorene brukt i frontsammenligningen. Målet er å velge en deskriptor som kan tilfredsstille kravene som stilles for bruk i et sanntidssystem.

HOG deskriptor
HOG deskriptoren opptrer robust ved lysendringer og viser også til gode resultater ved okklusjon. Metoden har også den fordelen at beregninger av HOG deskriptoren foregår raskt. Ulempen med metoden oppstår når ulike kjøretøyfronter har en kantstruktur som ligner, og dermed blir matchet feil. Dette skjer fordi HOG ikke innhenter informasjon om intensiteten i bildet, og mister dermed evnen til å ta frontens fargeintensiteten med i vurderingen.

Trace deskriptor
Trace transformen viste til gode resultater ved matching av "ideelle" (Test1) kjøretøyfronter, men presisjonen falt raskt da det oppstod lysendringer eller okklusjon ved referansebildet som skulle matches. Resultatet viste også at de 8 mest korrelerte sirkelfunksjoner av trace transformen gav likt eller bedre resultat i testene enn ved å bruke 21 kjente sirkelfunksjoner. Den største forskjellen mellom de to var at beregningen av 8 sirkelfunksjoner tok 35% mindre tid enn ved 21 sirkelfunksjoner. Derimot er kjøretiden for 8 mest korrelerte sirkelfunksjoner er fortsatt høy, selv ved liten bildestørrelse.

Trace deskriptor + HOG
Denne deskriptoren viste til best resultat ved testing, særlig ved "ideelle" bilder. Sammenlignet med HOG klarte denne deskriptoren bedre å skille kjøretøy som har lik kantstruktur grunnet bidraget fra trace transformen. På grunn av ulike bildetransformasjoner er det derimot for tidlig å konkludere om denne metoden vil være best når systemet kjøres i sanntid. Dette særlig på grunn av at trace transform gir mindre robusthet ved lysendringer. Deskriptoren er

også beregningsmessig krevende.

HOG deskriptoren ble i denne oppgaven valgt fordi den er rask og invariant mot lysendringer. Deskriptoren har derimot en svakhet der informasjonen om fargen eller intensiteten av kjøretøyfronten ikke blir tatt med i vurderingen av likheten. Derfor er det en fordel å bruke en ekstra deskriptor som også henter ut informasjon om intensiteten i bildet. Bidraget fra denne deskriptoren bør veies mindre da den kan være mindre invariant mot lysendringer. Til dette kan trace transform brukes som et alternativ, men en ulempe er at den krever ekstra prosesseringskraft. En mulighet er også å bruke middelverdien av fargen til å skille kjøretøy som HOG ikke har mulighet til.

Det er viktig å understreke at resultatet oppnådd i testene må sees på som en indikasjon heller enn et endelig resultat. Om systemet skulle blitt realisert i praksis, ville vi fått et anderledes resultatet grunnet flere faktorer som ville spilt inn. Den største usikkerheten kommer fra nøyaktigheten av frontsegmenteringen og til dette er algoritmen avhengig av en presis deteksjon av skiltet. Dette er et område oppgaven ikke har fokusert på grunn av eksisterende arbeid og teknologi.

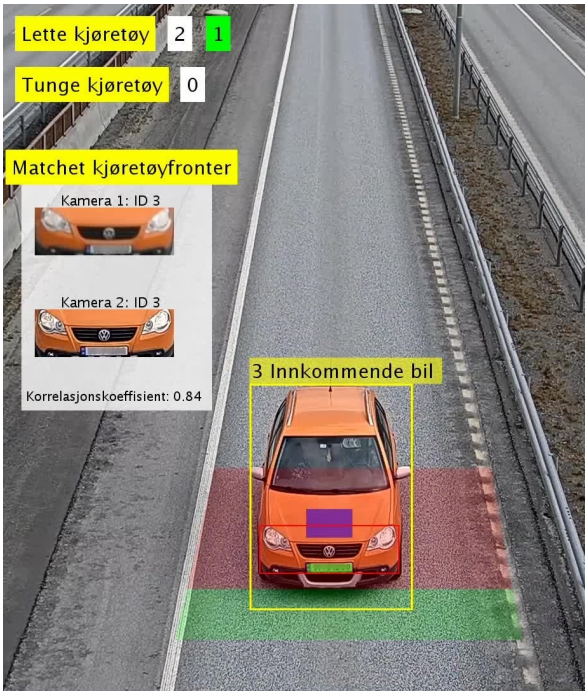
En faktor som derimot vil bidra positivt til et system som realiseres i praksis, er at begge kameraene som gjør opptak vil ha egenskaper som er tilpasset oppgaven kameraene skal utføre. I testene var bare ett kamera tilpasset veiovervåking. Det andre kameraet gjorde opptak ved en lavere skarphet med forskjellig kontrast og fargeoppfatning. Disse ulikhetene gjør det vanskelig å forutse antallet kjøretøy som vil matches korrekt i et sanntidssytem i motsetning til resultatet i disse testene. Dersom vi overfører resultatet fra test 1 og matcher referansebilde mot 25 kjøretøykandidater (antar at det er en trafikkert tid på døgnet) kan det forventes en match på 98.75% ved bruk av HOG. Test 4 viser at HOG fortsatt vil kunne gi et godt resultat ved større geometrisk transformasjon av kjøretøyet mellom opptakene.

5.6 Test av algoritme for kjøretøygjenkjenning

Algoritmen for kjøretøygjenkjenning har blitt testet mot opptaket gjort av Kamera 2 forklart i Kapittel 5.1.1. I testen blir både egenskaper fra kjøretøyfronten og middelverdien av fargen hentet ut fra 15 kjøretøy som krysser deteksjonsområdet. Egenskapene fra kjøretøyfronten blir hentet ut i form av en samlet deskriptor for HOG og trace transform med 8 sirkelfunksjoner. Denne matches videre mot deskriptorene beregnet fra kjøretøyene i Kamera 1. Samtidig utføres det telling av antall lette eller tunge kjøretøy som passerer kameraet. Telling av lette kjøretøy deles videre inn i antall kjøretøy som har grønne eller hvite skilt. Bilder av hvert kjøretøy med hver sin unike ID er vist i Figur A.6 for opptak av Kamera 1 og i Figur A.7 for

opptak av Kamera 2. Resultatet av testen er vist i to videoer som lagt i vedlegg. Her er alle kjøretøy blitt matchet korrekt.

- **System_resultat.m2v** - Resultat av algoritmen for kjøretøygjenkjenning. I denne videoen vises korrelasjonskoeffisienten samt bilde av kjøretøyfrontene som blir matchet som like når et kjøretøy krysser deteksjonsområdet. Et bildeutsnitt fra videoen er vist under i Figur 5.40.
- **System_resultat_gmm.mp4** - Resultat av bakgrunnssubtraksjon.



Figur 5.40 Her vises hvordan algoritmen for systemet virker i sanntid. Her telles antall lette kjøretøy med grønne og hvite skilt, samt tunge kjøretøy som passerer deteksjonsområdet. Fronten som detekteres ved kjøretøyet vises i bildet under teksten ”Kamera 2”. Fronten som kjøretøyet finner som match vises under teksten ”Kamera 1” i bildet. Området på panseret der middelverdien av fargen beregnes fra, er vist i lilla.

Kapittel 6

Konklusjon

I denne oppgaven har det blitt laget et prototype for et system som detekterer, gjenkjenner og klassifiserer kjøretøy ved begge endene i en tunnel. Det er satt hovedfokus på egenskapsdetektering som brukes til å gjenkjenne samme kjøretøy ved et senere tidspunkt. Egenskapene deles inn i form, farge og struktur der de to førstnevnte brukes til å skille vekk usannsynlige kjøretøykandidater.

Leting etter egenskaper som forteller noe om bildestrukturen gjøres her ved å se på et bestemt område av kjøretøyet, nemlig fronten. Fra dette området dannes en deskriptor i form av HOG og trace transform. Det er blitt konkludert med at HOG deskriptoren passer best til å danne en beskrivelse av kjøretøyfronten som er både rask og lite sensitiv for støy i form av lysendringer. En mer omfattende frontbeskrivelse kan derimot oppnås ved å kombinere HOG sammen med en deskriptor som henter informasjon om intensiteten i bildet. I oppgaven ble trace transform brukt til dette, og et bedre resultat ble oppnådd. Ulempen med trace transformen er at den krever mye prosesseringskraft.

6.1 Videre arbeid

- Matlab er et flott programmeringsspråk for testing av algoritme, men er lite egnet til å lage et program som kan implementeres i et sanntidssystem. Derfor er det anbefalt å gå over til et raskere språk som Python eller C++.

- Systemet i oppgaven kan deles inn i kjøretøydetektering, klassifisering og egenskapsdetektering. Hvert område har et stort potensiale for videreutvikling. Kjøretøydetektering er et av de viktigste, men også det vanskeligste leddet i prosessen, fordi man har

utfordringer som skygge eller objekter som feilaktig blir en del av forgrunnen. Derfor er det muligheter å implementere en metode som bedre detekterer og validerer kjøretøy.

- Objektgjenkjenning av flere kjøretøytyper som buss, trailer og motorsykkel ville også vært interessant.
- Siden en av de største usikkerhetene ved matching av frontutsnitt vil komme fra posisjonen av deteksjonvinduet, er det kanskje lurt å videreutvikle algoritmen til å bedre segmentere ut fronten, særlig i tilfeller der skiltposisjonen ikke kan brukes til segmentering av front.
- I oppgaven har det vært fokus på egenskapgjenkjenning ved dagslys, men i tunneler kan det ofte være dårlig belysning. En mulighet vil derfor være å finne egenskaper som kan brukes til å gjenkjenne kjøretøy under svake lysforhold.

Bibliografi

[1] Asaidi, H., Aarab, A., and Bellouki, M. (2014). Shadow elimination and vehicles classification approaches in traffic video surveiilance context. *Journal of Visual Languages and Computing*, 25(4):333–345.

[2] Baran, R., Rusc, T., and Fornalski, P. (2015). A smart camera for the surveillance of vehicles in intellegent transportation systems. *Springer*.

[3] Barbu, T. (2014). Pedestrian detection and tracking using temporal differencing and HOG features. *Computers and electrical engineering*, 40.

[4] Bay, H., Ess, A., Tuytelaars, T., and Gool, L. V. (2008). Speeded-up robust feature (SURF). *Computer Vision and Image understanding*, 110(Issue 3):346–359.

[5] Bouwmans, T. (2014). Traditional and recent approaches in background modeling for foregound detection: An overview. *ELSEVIER*.

[6] Bouwmans, T., El Baf, F., and Vachon, B. (2008). Background modeling using mixture of gaussians for foreground detection: A survey. *Recent Patents Comput. Sci. 1*, 3:219–237.

[7] Chai, T. and Draxler, R. R. (2014). Root mean square error (RMSE) or mean absolute error (MAE). *Geoscientific Model Development*.

[8] Chaves, J. (2015). Introduction to nonimaging optics. Second edition.

[9] Chen, Z., Ellis, T., and Velastin, S. A. (2011). Vehicle type categorization: A comparison of classification schemes. *Intellegent transportation systems*.

[10] CIE Publications (2004). International commission on illumination. [online] http://cie.mogi.bme.hu/cie_arch/kee/div1/tc148.pdf.

[11] Coifman, B., Beymer, D., McLauchlan, P., and Malik, J. (1998). A real-time computer vision system for vehicle tracking and traffic surveillance. *Transportation Reaserach Part C 6*, pages 271–288.

[12] D. J. Bora, A. K. Gupta, F. A. K. (2015). Comparing the performance of L*A*B and HSV color spaces with respect to color image segmentation. *International Journal of Emerging Technology and Advanced Engeneering*.

[13] Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. *Proc. IEEE Conf. Computer Vision and Pattern Recognition*.

[14] Du, S., Ibrahim, M., and Shehata, M. (2013). Automatic license plate recognition (ALPR): A state-of-the-art review. *IEEE Transactions on circuits and systems for video technology*, 23(2).

[15] Duan, T. D., Duc, D. A., and Du, T. L. H. (2004). Combining hough transform and contour algorithm for detecting vehicles license-plates. *Proceedings of 2004 International Symposium on Intellegent Multimedia, Video and Speech Processing*.

[16] Dule, E., Gokmen, M., and Beratoglu, M. S. (2010). A convenient feature vector construction for vehicle color recognition. *Recent advances in neural networks, fuzzy systems and evolutionary computing*.

[17] Engebretsen, A. (2013). Ulykker i norske tunneler siste ti år. *Statens vegvesen*.

[18] Ferryman, J. M., Worrall, A. D., and Maybanki, S. J. (1998). Learning enhanced 3d models for vehicle tracking. *In Proceedings of the British Machine Vision Conference*, pages 873–882.

[19] Frías-Velázquez, A., Van Hese, P., Pižurica, A., and Philips, W. (2015). Split-and-match: A bayesian framework for vehicle re-idetification in road tunnels. *Elsevier*.

[20] Frias-Velazquez, A., Philips, W., Besard, T., and Sutter, B. D. (2014). Case study of multiple trace transform implementations. *The International Journal of High Performance Computing Applications*.

[21] Gonzalez, R. C. and Woods, R. E. (2008). Digital image procesing. (Third Edition).

[22] Gu, H. and Lee, S. (2013). A view-invariant and anti-reflection algorithm for car body extraction and color classification. *Multimed Tools Appl*, 65:387–417.

[23] Guo, Y., Hsu, S., Sawhney, H. S., Kumar, R., and Shan, Y. (2007a). Robust object matching for persistent tracking with heterogeneous features. *IEEE Trans. Pattern Analysis and Machine Intellegence*, (29):824–839.

[24] Guo, Y., Hsu, S., Sawhney, H. S., Kumar, R., and Shan, Y. (2007b). Robust object matching for persistent tracking with heterogeneous fetures. *IEEE Trans. On Pattern Analsis And Machine Intelligence*, 29(5).

[25] Haralic, R. M., Sternberg, S. R., and Zhuang, X. (1987). Image analysis using mathematical morphology. *IEEE Translations on pattern analysis and machine intellegence*, PAM1-9(4).

[26] Heyman, E. and Eklundh, J. O. (2003). Statistical background subtraction for mobile observer. *IEEE International Conference on Computer Vision (ICCV)*, pages 67–74.

[27] Hsieh, J. W., Yu, S. H., Chen, Y. S., and Hu, W. F. (2006). Automatic traffic surveillance system for vehicle tracking and classification. *IEEE Transactions on intellegent transportation systems*, 7(2).

[28] Hsies, J., Yu, S., and Chen, Y. (2002). Morphology-basd license plate detection form complex schenes. *16th International Conference on Pattern Recognition*.

[29] Hu, C., Chen, L. Q. P., Xue, G., and Mei, L. (2015). Vehicle color recognition with spatial pyramid deep learning. *IEEE Transactions on intellegent transportation systems*, 16(5).

[30] Huang, J., Kumar, S. R., Mitra, M., Zhu, W., and Zabih, R. (1997). Image indexing using color correlograms.

[31] Intel (2016). Color models. [online] https://software.intel.com/en-us/node/503873.

[32] ITU (1995). *ITU-R BT.601-5*. Studio encoding parameters of digital television for standard 4:3 and widescreen 16:9 aspect ratio.

[33] Javed, O., Shafique, K., and Shah, M. (2005). Appearance modelling for tracking in multiple non-overlapping cameras. *International Conference on Computer Vision and Pattern Recognition*, 2(29):26–33.

[34] Jelača, V., Pižurica, A., Oswaldo, J., and Niño-Castañeda (2013). Vehicle matching in smart camera networks using image projection profiles at multiple instances. *Elsevier*.

[35] Kadyrov, A. and Petrou, M. (2001). The trace transform and its applications. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23:811–828.

[36] Kadyrov, A. and Petrou, M. (2003). Object signatures invariant to affine distortions derived from the trace transform. *Image Vision and Computing*, 21(13-14):1135–1143.

[37] Kalman, R. E. and Emil, R. (1960). A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82.

[38] Kaur, A. and Kranthi, B. V. (2012). Comparison between YCbCr color space and CIE-Lab color space for skin color segmentation. *International Journal of Applied Information Systems*.

[39] Kim, K., Park, S., and Choi, Y. (2008). Deciding the number of color histogram bins for vehicle color recognition. *IEEE Asia-Pasific Services Computing Conference*.

[40] Kristensen, F., Nilsson, P., and Öwall, V. (2006). Background segmentation beyond RGB. *ACCV 2006*, 2:1508–1511.

[41] Kulkarni, P., Khandebharad, A., Khope, D., and Chavan, P. (2012). License plate recognition: A review. *IEEE- Fourth International Conference on Advanced Computing*.

[42] l. Ambardekar, A., Nicolescu, M., and Bebis, G. (2008). Efficient vehicle tracking and classification for an automated traffic surveillance system.

[43] Leung, M. K. and Yang, Y. H. (1987). Human body motion segmentation in a complex scene. *Pattern Recognition*, (20):55–64.

[44] Lewis, J. P. (1995). Fast normalized cross-correlation. *Industrial Light and Magic*.

[45] Li, Q., Qu, G., and Li, Z. (2007). Matching between sar images and optical images based on hog descriptor. *IEEE*.

[46] Li, Y., Li, Z., Tian, H., and Wang, Y. (2011). Vehicle detecting and shadow removing based on edged mixture gaussian model. *Preprints of the 18th IFAC World Congress*.

[47] Litomisky, K. (2012). Consumer RGB-D cameras and their applications.

[48] Liu, P. C. W. (2014). Vehicle color recognition on urban road by feature context. *IEEE Transactions on Intellegent Transportation Systems*.

[49] Lovdata (1994). Forskrift om tekniske krav og godkjenning av kjøretøy, deler og utstyr (kjøretøyforskriften). [online] https://lovdata.no/dokument/SF/forskrift/1994-10-04-918.

[50] Lovdata (2007). Forskrift om minimum sikkerhetskrav til visse vegtunneler. [online] https://lovdata.no/dokument/SF/forskrift/2007-05-15-517.

[51] Lowe, D. G. (1999). Object recognition from local scale-invariant features. *Proceedings of the International Conference on Computer Vision*, page 1150–1157.

[52] Mahmassani, H. S., Haas, C., Zhou, S., and Peterman, J. (1998). Evaluation of incident detection methodologies. *Center of Transportation Research, The University of Texas at Austin*, (Technical Report FHWA/TX-00/1795-1).

[53] Marimon, D. and Ebrahimi, T. (2007). Orientation histogram-based matching for region tracking. *IEEE*.

[54] Matlab (2015). Hog features. [online] http://se.mathworks.com/help/vision/ref/extracthogfeatures.htm.

[55] Meher, S. K. and Murty, M. N. (2013). Efficient method of moving shadow detection and vehicle clasification. *International Journal of Electronics and Communications*.

[56] Nævestad, T.-O. and Meyer, S. F. (2012). Kartlegging av kjøretøybranner i norske vegtunneler 2008-2011. *Transportøkonomisk institutt*, 1205/2012:1–8.

[57] Patil, S. P. and Patil, M. B. (2014). Moving vehicle detection: A review. *International Journal of Computer Applications*.

[58] Pflugfelder, R., Bischof, H., Dominguez, G. F., Nolle, M., and Schwabach, H. (2005). Influence of camera properties on image analysis in visual tunnel surveillance. *Proceedings of the 8th International IEEE Conference on Intellegent Transportation Systems*.

[59] Porikli, F. (2004). Inter-camera color calibration using cross-correlation model function. *International Conference on Image Processing*, 2:133–136.

[60] Poynton, C. (1996). *A technical introduction to digital video*. John Wiley and Sons.

[61] Psyllos, A. and C. N. Anagnostopoulos, E. K. (2011). Vehicle model recognition from frontal view image measurements. *Computer Standards and Interfaces*.

[62] Rabiu, H. (2013). Vehicle detection and classification for cluttered urban intersection. *International Journal of Computer Science, Engeneering and Applications*, 3(1).

[63] Rachmadi, R. F. and Purnama, K. E. (2015). Vehicle color recognition using convolutional neural network. *Multimedia and Network Engineering Department, Institut Teknologi Sepuluh Nopember*.

[64] Ribeiro, H. and Gonzaga, A. (2006). Hand image segmentation in video sequence by GMM. *19th Brazilian Symposium on Computer Graphics and Image Processing*, pages 357–364.

[65] Rios-Cabrera, R., Tuytelaars, T., and Van Gool, L. (2012). Efficient multi-camera vehicle detection, tracking, and identification in a tunnel surveillance application. *Elsevier*.

[66] Rosten, E. and Drummond, T. (2005). Fusing points and lines for high performance tracking. *Proc. Int. Conf. Computer Vision*, 2:1508–1511.

[67] Rosten, E. and Drummond, T. (2006). Machine learning for high-speed corner detection. *European Conference on Computer Vision*, 1:430–443.

[68] Schwabach, H., Harrer, M., Bischof, H., and Nölle, M. (2005). Video based image analysis for tunnel safety - VITUS-1: A tunnel video surveillance and traffic control system.

[69] Sheldon, B. and Wright, P. (2010). Policing and technology.

[70] Shukla, A. P. and Saini, M. (1998). "moving object tracking of vehicle detection": A concise review. *International Journal of Signal Processing, Image Processing and Pattern Recognition*, 8:169–176.

[71] Sivaraman, S. and Trivedi, M. M. (2013). Looking at vehicles on the road: A survei of vision-based vehicle detection, tracking, and behaviour analysis. *IEEE Transactions on intellegent transportation systems*, 14(4).

[72] Smith, A. R. (1978). Color gamut transform pairs. *Computer Graphics Lab*.

[73] Sobel, I. (2014). History and definition of the so-called "sobel operator". *Reaseachgate*.

[74] Sobral, A. (2013). Comparing background subtraction algorithms.

[75] Sobral, A. and Vacavant, A. (2013). A comprehensive review of background subtraction algorithms evaluated with synthetic and real videos. *ELSEVIER, Computer Vision and Image Understanding*.

[76] Sonka, M., Hlavac, V., and Boyle, R. (1999). *Image processing analysis, and machine vision*. PWS publishing.

[77] Statens Havarikommisjon for Transport (2013). Rapport om brann i vogntog på Rv 23, Oslofjordtunnelen, 23. juni 2011. [online] http://www.aibn.no/Veitrafikk/Rapporter/2013-05.

[78] Statens havarikommisjon for transport (2015). Rapport om brann i vogntog på E16 i Gudvangatunnelen i Aurland 5. august 2013. [online] http://www.aibn.no/Veitrafikk/Avgitte-rapporter/2015-02.

[79] Statens havarikommisjon for transport (2016). Rapport om bussbrann i Gudvangatunnelen på E16 i Aurland 11. august 2015. [online] http://www.aibn.no/Veitrafikk/Avgitte-rapporter/2016-03.

[80] Statens vegvesen (2013). Evaluering av streknings atk i tunnel. [online] http://www.vegvesen.no/_attachment/529035/binary/851385?fast_title=Evaluering+av+streknings-ATK+i+tunnel+2013.pdf.

[81] Statens vegvesen (2015a). Automatisk skiltgjenkjenning (ANPR). [online] http://www.vegvesen.no/fag/Trafikk/Utekontroll.

[82] Statens vegvesen (2015b). Tunnelteknikk. [online] http://www.vegvesen.no/fag/Teknologi/Tunneler.

[83] Stauffer, C. and Grimson, E. (1999). Adaptive background mixture models for real-time tracking. *IEEE Conference on Computer Vision and Pattern Recognition*, page 246–252.

[84] Store Norske Leksikon (2016). Krominans. [online] https://snl.no/krominans.

[85] Sun, Z. (2006). On-raod vehicle detection: A review. *IEEE Transactions on pattern analysis and machine intellegence*, 28(5).

[86] Tsai, D. M. and Lin, C. T. (2003). Fast normalized cross correlation for defect detection. *Elsevier B.V.*

[87] Unzueta, L., Nieto, M., Cortes, A., Barandiaran, J., Otaegui, O., and Sanchez, P. (2012). Adaptive multicue background subtraction for robust vehicle counting and classification. *IEEE Transactions on intellegent transportation systems*, 13(2).

[88] van de Weijer, J., Schmid, J., Verbeek, C., and Larlus, J. (2009). Learning color names for real-world applications. *IEEE Trans. Image Process*, (18):1512–1523.

[89] Velazquez, A. F., Ortiz, C., Pizurica, A., Philips, W., and Cerda, G. (2012). Object identification by using orthonormal circus functions from the trace transform. *IEEE*.

[90] Wang, J. Z. (2001). Integrated region-based image retrieval. *Kluwer Academic Publishers*.

[91] Wikipedia (2015). The infra-red traffic logger. [online] https://en.wikipedia.org/wiki/TIRTL.

[92] Wikipedia (2016). Induction loop. [online] https://en.wikipedia.org/wiki/Induction_loop.

[93] Zhai, X., Benssali, F., and Ramaligam, S. (2010). License plate localisation based on morphological operations. *Int. Conf. Control, Automation, Robotics and Vision*, 11.

[94] Zhang, W., Wu, Q. M. J., Yang, X., and Fang, X. (2008). Multilevel framework to detect and handle vehicle occlusion. *IEEE Transactions on intellegent transportation systems*, 9(1).

[95] Zheng, D., Zhao, Y., and Wang, J. (2005). An efficient method of license plate location. *Pattern Recognition Letters*, 26:2431–3438.

Tillegg A

A - Startguide i bruken av kamera

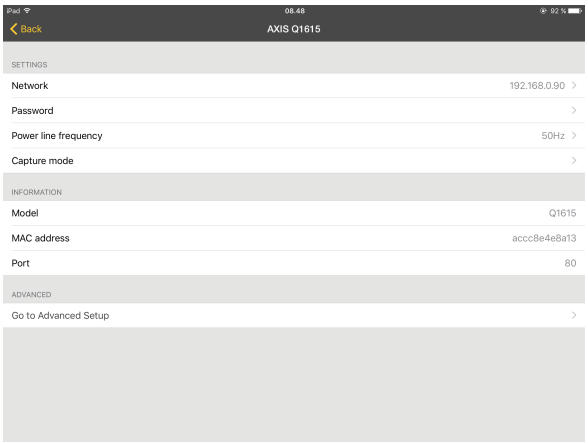
Her forklares det hvordan Axis Q1615-E nettverkskamera gjøres klart for opptak.

1. Koble kameraet til Axis T8415 (batteridrevet ethernet switch) gjennom en ethernet kabel.

2. Last ned appen ”AXIS T8415 Wireless Installation Tool” på et Android eller iOS operativsystem.

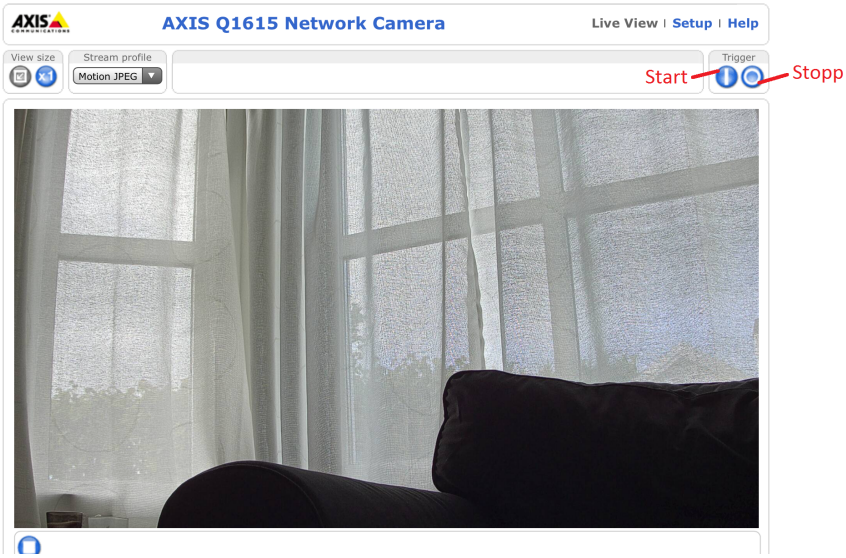
3. Koble så wifi-nettverket til Axis T8415 og åpne appen som nylig er lastet ned.

4. Gå inn på innstillinger og trykk videre på Advanced Setup som vist i Figur A.1.



Figur A.1 Innstilling meny ved AXIS T8415 Wireless Installation Tool

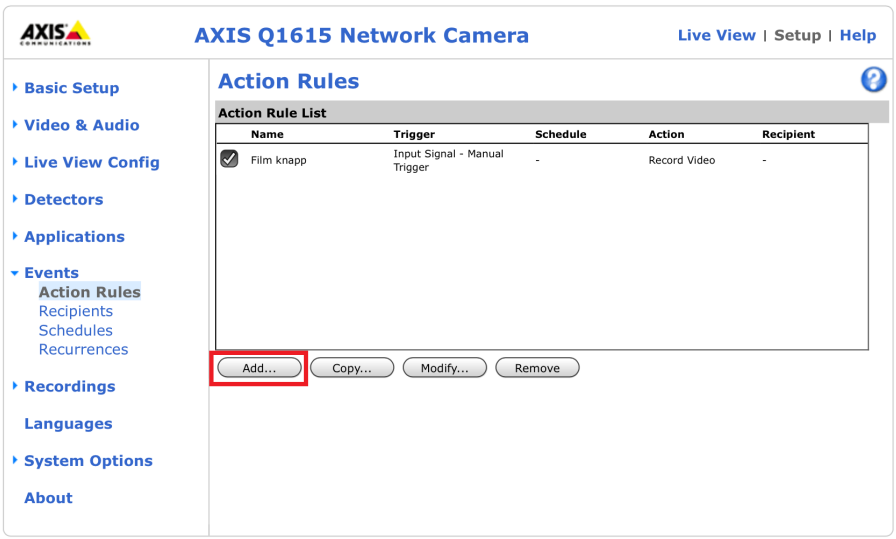
5. Tast inn brukernavn: root, passord: vegvesen. Om denne kombinasjonen ikke skulle virke, må kameraet nullstilles.



Figur A.2 Opptak styres med start og stopp knapper markert i rødt.

6. Opptak styres ved bruk av start og stopp knapp vist i Figur A.2.

7. I tilfelle styringsknappene ovenfor ikke er å finne på Live View bildet, trykk på Setup, så videre på Events og Action Rules i venstre meny. Trykk deretter på add-knappen vist i Figur A.3.



Figur A.3 I rødt vises add knappen som brukes til å danne start og stopp knapp ved opptak.

B - Eksisterende sensorer brukt i trafikk

I dette underkapittelet presenteres eksisterende teknologier for deteksjon av kjøretøy.

Teknologi for detektering av kjøretøy har eksistert en stund. Spørsmålet stilles om maskinsyn algoritmer i intelligente kameranettverk, kan forbedre deteksjonsraten over eksisterende teknologier.
I dag eksisterer et stort mangfold av ulike type sensorer som hver kan detektere ulike egenskaper ved kjøretøy. Disse egenskapene kan være varmestråling, lyd, magnetisk felt, vekt eller refleksjon av elektromagnetisk stråling. Slike signaturer til kjøretøyet er spesielt nyttig når tåke, røyk eller svake lysforhold gjør "synlige" observasjoner umulig.
En gjennomgang av de mest populære sensorene er beskrevet nedenfor, sammen med en vurdering gjort av H. S. Mahmassani, et. al., [52] av fordeler og ulemper.

- **Induktive sløyfer** er et elektromagnetisk deteksjonssystem hvor det brukes en bevegelig magnet til å indusere strøm i elektriske ledere som er forankret 5-10 cm ned i veien. Kjøretøy som passerer over sløyfen induserer strømmer i lederne og reduserer deres induktans. Teknologien kan både brukes til deteksjon og klassifisering av kjøretøygrupper [92]. Fordelen med teknologien er at den er billig, væruavhengig og har god ytelse. Ulempene er at den lett kan bli skadet ved tung trafikk noe som vil kreve stenging av veien for å repareres.

- **Trykkfølsomme sensorer** brukes til å måle kjøretøyets fart, akselavstand og vekt. Sensoren ligger i veibanen og har derfor mange av de samme ulempene som induktive sløyfer.

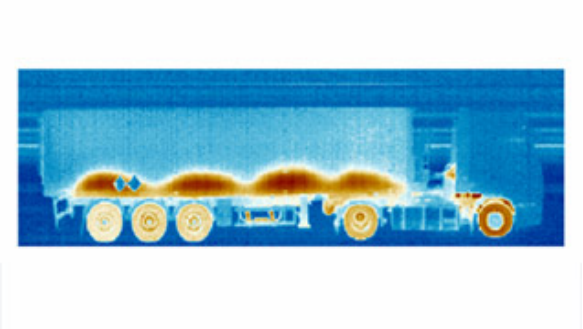
- **Radar** har mulighet til å måle fart, telle antall kjøretøy og kan "se" (opptil 150 meter) i all slags vær. Radar er også billig og lett å reparere. Teknologien kan derimot ha problemer i flerfelts applikasjoner der en maskering av kjøretøy oppstår. Teknologien kan ha en grad av falske alarmer.

- **Laser** blir brukt til deteksjon og fartsmåling, men er ikke robust mot dårlig vær som mye regn, tåke og snø. Laserskannere har muligheten til å klassifisere ulike kjøretøygrupper. Teknologien regnes som dyr.

- **Ultralyd** sender lydsignal høyere enn hva det menneskelig øre kan høre. Sensoren kan måle volum på kjøretøy og fart, men apparatet er støysensitivt når det skjer endringer i omgivelsene som temperatur, luft-fuktighet og -turbulens.

- **Bildeprosessering** har blitt brukt til trafikkovervåking siden 1970 tallet i blant annet USA, Japan og Frankrike [52]. Typisk blir videokamera med bildeprosessering brukt til kjøretøydetektering, men økningen i datakraft de siste årene åpner for utvidet bruk av teknologien. I dag har flere tunneler i verden kamera system som automatisk varsler om unormale forhold som fotgjenger i tunnel, kjøretøy i feil veibane, kø, røyk, mistet last og varmegang i kjøretøy. Systemet går under navnet automatic incident detection (AID). Teknologien har også mulighet til å detektere trafikk i flere kjørefelt som faller innfor synsfeltet.

  For å takle utfordringer som kan oppstå ifm. okklusjon, sterkt sollys og skygger vil det kreves avanserte algoritmer som igjen krever mye datakraft. Forskjellige algoritmer blir derfor ofte brukt ved natt og dagtid. Videre egner teknologien seg ikke i vær som gir dårlig sikt. For direkte bildeoverføring kreves det omfattende kommunikasjonsutstyr. Nylig har bildeprosessering blitt brukt i termiske kameraer for deteksjon av varmgang. I Østerrike brukes termisk kamera i dag til å detektere temperaturen i tunge kjøretøy fra begge sider, noe som er vist i Figur A.4. I etterkant brukes programvare for analyse og deteksjon av overoppheting der kjøretøyet kan bli tatt av veien for nedkjøling.



Figur A.4 Termoskanning av tungt kjøretøy.
FOTO: Østerrikes Vegvesen (ASFINAG)

- **Infrarødt lys** fungerer bedre enn sensor for synlig-lys sensorer i tåke. Et eksempel på IR-teknologi er TIRTL (The Infra-Red Traffic Logger). Sensoren kan brukes til trafikktelling, som fartsensor, sensor for rødt lys, sporing av tungt kjøretøy og høydesensor [91]. Teknologien fungerer ved at IR-stråling blir sendt til en mottaker vinkelrett av kjøreretningen på motsatt side av veien. Det åpner for klassifisering, fart- og volum-måling av kjøretøy. Teknologien har vært for salg siden 2002. Ulempen er at kostnadene for en slik sensor kan være høye og at den er utsatt for værpåvirkning.

Tilgjengelig utstyr for gjenkjenning av kjøretøy

Allerede i dag bruker Statens vegvesen utstyr til gjenkjenning av kjøretøy ved 2 forskjellige plasser på en veistrekning. Systemet kalles automatisk trafikkontroll (ATK) og måler gjennomsnittsfarten over en distanse. Systemet har også blitt testet i tunnel og det er blitt anbefalt at fotobokser bør plasseres enten ca. 50-100 m utenfor tunnelmunningen eller trekkes tilsvarende inn i tunnelen [80]. Samme teknologi kan i prinsippet brukes til å telle antall kjøretøy som befinner seg i tunnelen til enhver tid. En illustrasjon av ATK er vist i Figur A.5.

Figur A.5 Strekningsmåling av fart.
FOTO: Statens Vegvesen

Teknologien virker slik at fotoboks A vist i Figur A.5 tar bilde av skiltnummeret og sjåføren til kjøretøyet som passerer. Skitne skilt og refleksjoner fra sola gjør at skiltene ikke alltid er leselige. Derfor plasseres det også trykkfølsomme sensorer i veibanen som detekterer kjøretøyets fart, vekt og akselavstand. Nøkkeldata som tidspunkt for passering, akselavstand, vekt og registreringsnummer blir så sendt over til fotoboks B og brukt til å gjenkjenne kjøretøyet. En av teknologiens ulemper er blitsen som brukes til å ta bilder med, kan virke sjenerende på bilister. Ved å bare ta bilder når et kjøretøy passerer, begrenses bruksområdet fra å innføre andre smartfunksjoner som er mulig i videoovervåkning, eksempelvis fotgjenger i tunnel eller andre hendelsesdeteksjoner.
I denne masteroppgaven videreføres denne ideen: Bilenes skiltnummer skal brukes som hovedegenskap til gjenkjenning. Andre data vil bidra til å kjenne igjen kjøretøyet når lesing av skilt mislykkes.

C - Fargesammenligning ved Y+CbCr fargerom

Tabell A.1 viser resultatet av Test 3 forklart i Kapittel 5.4.3. Her brukes *Y+CbCr* fargerom til å skille vekk panserutsnittparer som ikke likner hverandre. I tabellen vises 1 som *lik* og - som *ulik*. Horisontal retning beskriver panserutsnittene hentet fra 16 kjøretøy som er detektert av Kamera 1 og vertikal retning viser til utsnitt av identisk kjøretøy hentet fra Kamera 2. For å unngå feilklassifiseringer må diagonalen være 1.

Tabell A.1 Tabellen viser hvilke panserutsnitt betegnes som like

Kamera 2 ↓ \ Kamera 1 →	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1	-	-	-	-	-	1	-	1	-	-	1	-	1	1	-
2	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-
3	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-
4	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-
5	1	-	-	-	1	-	1	-	1	-	-	1	-	1	1	-
6	-	-	-	-	-	1	-	1	-	-	-	-	-	-	-	-
7	1	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-
8	-	-	-	-	-	1	-	1	-	-	-	-	-	-	-	-
9	1	-	-	-	1	-	1	-	1	-	1	1	-	1	1	-
10	1	-	-	-	1	-	1	-	1	1	1	1	-	1	1	-
11	1	-	-	-	1	-	1	-	1	-	1	1	-	1	1	-
12	1	-	-	-	1	-	1	-	1	-	1	1	-	1	1	-
13	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-
14	1	-	-	-	1	-	1	-	1	1	1	1	-	1	1	-
15	1	-	-	-	1	-	1	-	1	1	1	-	-	1	1	-
16	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1

D - Bildesett

D1 - Kamera 1 bildesett

Bildene er blitt manuelt klippet ut fra et 1.13min langt opptak gjort av Kamera 1 forklart i Figur 5.3. Bildesettet inneholder 15 kjøretøy. Frontutsnittene fra disse kjøretøyene er blitt matchet i testen forklart i Kapittel 5.6.

(a) ID 1 (b) ID 2 (c) ID 3 (d) ID 4 (e) ID 5 (f) ID 6 (g) ID 7 (h) ID 8

(i) ID 9 (j) ID 10 (k) ID 11 (l) ID 12 (m) ID 13 (n) ID 14 (o) ID 15

Figur A.6 Enkelte kjøretøybilder hentet fra opptak i Kamera 1.

D2 - Kamera 2 bildesett

Bildesettet her er et resultat av bakgrunnssubtraksjon som utføres i testen forklart Kapittel 5.6. Opptaket er blitt gjort av Kamera 2, der detaljer om opptaket er gitt i Figur 5.3.

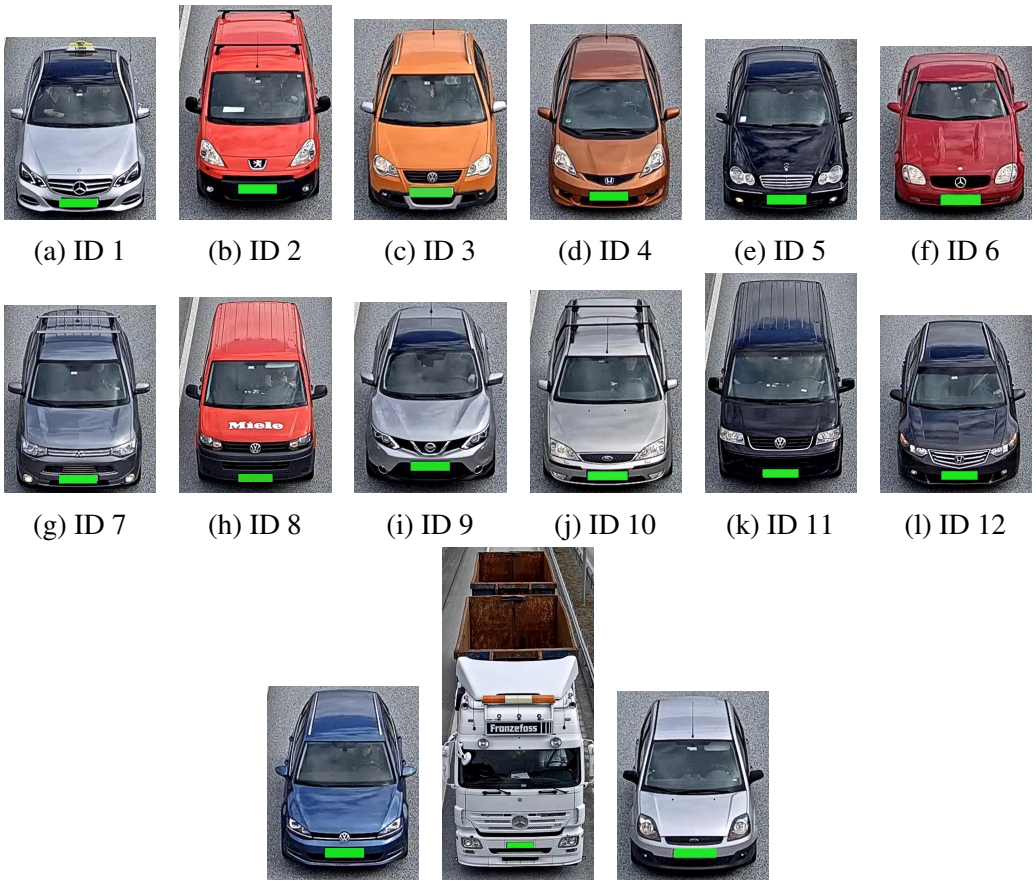
(a) ID 1 (b) ID 2 (c) ID 3 (d) ID 4 (e) ID 5 (f) ID 6 (g) ID 7 (h) ID 8

(i) ID 9 (j) ID 10 (k) ID 11 (l) ID 12 (m) ID 13 (n) ID 14 (o) ID 15

Figur A.7 Enkelte kjøretøybilder som er segmentert ved bakgrunnssubtraksjon utført på opptaket gjort av Kamera 2.

D3 - Skiltdeteksjon

Her vises resultat av skiltdeteksjon som utføres i testen forklart i Kapittel 5.6.



(a) ID 1 (b) ID 2 (c) ID 3 (d) ID 4 (e) ID 5 (f) ID 6

(g) ID 7 (h) ID 8 (i) ID 9 (j) ID 10 (k) ID 11 (l) ID 12

(m) ID 13 (n) ID 14 (o) ID 15

Figur A.8 Resultatet for skiltdeteksjon er vist i området i grønt.

E - Vedlegg

Kildekoden er plassert i zipfilen. En oversikt over funksjonene er gitt her:

- vehicle_detection.m
- LPdetect.m
- color_detect.m
- get_transform.m
- Trace transform.zip - Implementering av trace transform for ulike språk [20].

Videofiler er plassert i zipfilen. En oversikt over filen er gitt her:

- System_result.m2v - Resultatet av testen i Kapittel 5.6
- System_result_gmm.mp4 - Resultatet av bakgrunnssubtraksjonen i testen i Kapittel 5.6
- Trafikk_kamera1.m2v - Sladdet versjon av opptaket forklart i Kapittel 5.3
- Trafikk_kamera2.m2v - Sladdet versjon av opptaket forklart i Kapittel 5.3
- Trafikk_lysendring_kamera2.m2t - Sladdet versjon av opptaket forklart i Kapittel 5.4

Bildefiler er plassert i zipfilen. Bildene av kjøretøyene og frontene er forklart i Kapittel 5.1.3. En oversikt over filen er gitt her:

- Frontgruppe 1 - Både kjøretøybilder og frontutsnitt med og uten rotasjonsnormalisering.
- Frontgruppe 2 - Frontutsnittbilder med rotasjonsnormalisering.
- Frontgruppe 3 - Frontutsnittbilder med rotasjonsnormalisering.