NORGES TEKNISK-NATURVITENSKAPELIGE UNIVERSITET

FAKULTET FOR INFORMASJONSTEKNOLOGI, MATEMATIKK OG ELEKTROTEKNIKK

MASTEROPPGAVE

Kandidatens navn: Thor Marius Henrichsen & Erik Åldstedt Sund

Fag: Datateknikk

Oppgavens tittel (norsk): Automatisk Trafikkanalyse

Oppgavens tittel (engelsk): Automatic Traffic Analysis

Oppgavens tekst:

Videoovervåkning av veikryss blir stadig mer vanlig. Likevel finnes få verktøy for automatisk analyse av det resulterende videomaterialet. Det skal i denne oppgaven utvikles en prototyp for automatisk uttrekning av relevant informasjon fra slike overvåkningssekvenser. Man skal videre kunne spesifisere interessante situasjoner og få informasjon om slike med utgangspunkt i den forutgående analysen.

Analysesystemet skal inkludere prosesser som bevegelsessegmentering, skyggefjerning, klassifisering og følging. Resultatene fra videoanalysen skal struktureres i en romlig-temporal datamodell på en måte som legger til rette for gjenfinning av interessante situasjoner. Gjenfinningssystemet skal gi brukeren mulighet til å spesifisere konfigurasjoner av interessante hendelser og returnere disse i et passende brukergrensesnitt

---

Oppgaven gitt:	20. januar 2005
Besvarelsen leveres innen:	16. juni 2005
Besvarelsen levert:	16. juni 2005
Utført ved:	Institutt for datateknikk og informasjonsvitenskap
Veileder:	Roger Midtstraum

8. juni 2005

Roger Midtstraum
Faglærer

Forord

Denne rapporten dokumenterer implementasjon av og resultater fra Thor Marius Henrichsen og Erik Åldstedt Sunds hovedoppgave ved Norges teknisk-naturvitenskapelige universitet (NTNU) våren 2005.

Oppgaven ble formelt gitt av seksjon for data- og informasjonsforvaltning ved Institutt for Datateknikk og Informasjonsvitenskap (IDI) januar 2005 under tittelen "Automatisk Trafikkanalyse". Oppgaven er basert på en idé unnfanget av kandidatene selv i desember 2004. Oppgavebeskrivelsen er utarbeidet av kandidatene i samarbeid med hovedveileder Roger Midtstraum ved IDI.

Vi understreker at denne hovedoppgaven er et selvstendig prosjekt som ikke viderefører eller bygger på noe annet arbeid ved instituttet.

Vi vil rette en spesiell takk til vår veileder Roger Midtstraum for konstruktiv tilbakemelding i forbindelse med arbeidet med oppgaven. Videre har Arvid Aakre i faggruppe for Veg og Samferdsel ved Institutt for Bygg, Anlegg og Transport bidratt med nyttige kommentarer i forbindelse med bakgrunn, oppgavens relevans og eksisterende systemer. Vi vil også nevne bidrag fra Ketil Bø ved IDI som initielt inspirerte oss samt fra Trond Aalberg ved IDI for bidrag knyttet til hvordan data kan lagres og gjenfinnes.

Trondheim, 8. juni 2005.

Thor Marius Henrichsen

Erik Åldstedt Sund

Sammendrag

Videoovervåkning av veistrekninger i reguleringsøyemed blir stadig mer vanlig. Registrering av objektbevegelse, svingemønster og dermed også konfliktsituasjoner gjøres idag manuelt. Automatisering av dette er interessant både ut i fra et økonomisk og sikkerhetsmessig synspunkt. Mens tradisjonelle sensorteknologier bare registrerer begrensede aspekter ved scenen, registrerer man med video all visuell informasjon. På denne måten kan skjulte mønstre identifiseres og kvaliteten i avviklingen forbedres.

Denne rapporten beskriver prototyp implementasjoner for automatisk uttrekning og gjenfinning av relevant informasjon i slike overvåkningssekvenser. Implementasjonen er primært ment å behandle informasjon om kjøretøys bevegelser og svingemønster gjennom veikryss. Sekundært kan det benyttes til å analysere bevegelse langs en vilkårlig veistrekning.

Analysesystemet, X-Analyzer, inkluderer moduler for bevegelsessegmentering, skyggefjerning, klassifisering, objektfølging og datalagring. Systemet er generisk i den forstand at det kan anvendes på videomateriale fra vilkårlige veikryss. Brukeren definerer selv scenen i form av interessante regioner. Scenedefinisjonen danner grunnlaget for lagring i en romlig-temporal datamodell som legger til rette for gjenfinning av interessante situasjoner og sekvenser. Gjenfinningssystemet, X-Retriever, gir mulighet til å spesifisere konfigurasjoner som definerer konfliktsituasjoner i krysset.

Vi mener vi har lykkes med å implementere ideen presentert i oppgavebeskrivelsen. Vår prototyp for automatisk analyse opererer effektivt og implementerer all funksjonaliteten i vår målsetning. Gjenfinningssystemet returnerer de datafragmentene som samsvarer med brukerens spørringer. Vi konkluderer rapporten med at implementasjonen var vellykket, men ytterligere testing og videre utvikling vil være nødvendig dersom systemet skal benyttes i praksis.

Innhold

Figurer

Tabeller

Algoritmer

Kapittel 1

Innledning

1.1 Problemstilling

Videoovervåkning av veistrekninger blir stadig mer vanlig. Likevel gjøres registrering av objektbevegelse, svingemønster og dermed også konfliktsituasjoner fremdeles manuelt. Automatisering av dette er interessant både ut i fra et økonomisk og sikkerhetsmessig synspunkt. Mens tradisjonelle sensorteknologier bare registrerer begrensede aspekter ved scenen, registrerer man med video all visuell informasjon. På denne måten kan skjulte mønstre identifiseres og kvaliteten i avviklingen forbedres.

En applikasjon for å analysere videomateriale må segmentere den bevegelige forgrunnen fra den stillestående bakgrunnen. Videre må objektrepresentasjonen forbedres gjennom skyggefjerning. Skygge kan representere betydelig unøyaktighet i objektkonturen siden den ofte segmenteres som en del av det tilhørende forgrunnsobjektet. Regioner klassifiseres som ulike typer trafikanter eller støy. De klassifiserte regionene fra hvert bilde må så korreleres i tid for å danne bevegelsesbaner. Den uttrekte informasjonen må deretter struktureres i en romlig-temporal datamodell som legger til rette for effektiv gjenfinning.

Datamodellen definerer protokollen for hvordan data kan gjenfinnes. Et verktøy for å gjenfinne interessant informasjon fra videoanalysen bør ha uttrykkskraft til å spesifisere vilkårlige situasjoner gjennom et intuitivt brukergrensesnitt.

1.2 Vår løsning

Vi har sett på hvordan et system for automatisk trafikkovervåkning kan implementeres. Med bakgrunn i kjente metoder fra litteraturen har vi implementert en prototyp, X-Analyzer, for videoanalyse. X-Analyzer er et modulbasert system for analyse av videomateriale fra vilkårlige veistrekninger. Brukeren definerer selv en scene i form av interessante regioner. Scenedefinisjonen danner grunnlaget for analysen. Analysen inkluderer moduler for segmentering, skyggefjerning, klassifisering, følging og datalagring.

Gjennom X-Retriever har vi implementert et gjenfinningsverktøy for å finne igjen interessant informasjon i de uttrekte dataene. X-Retriever er et intuitivt brukergrensesnitt for å formulere

spørringer knyttet til statiske situasjoner og dynamiske sekvenser.

1.3 Rapportens struktur

Kapittel 2, *Bakgrunn*, presenterer motivasjonen for denne oppgaven fra et trafikkteknisk ståsted. En oppsummering av den overordnede strukturen, modulene og dataflyten i våre prototyper for analyse (X-Analyzer) og gjenfinning (X-Retriever) er gitt i kapittel 3, *Systembeskrivelse*. Generelle egenskaper ved inndata til en slik applikasjon og datasettene vi har benyttet til testing er beskrevet i kapittel 4, *Datasett*.

Hvert av kapitlene

- 5 *Segmentering*
- 6 *Skyggefjerning*
- 7 *Objektgjenkjenning*
- 8 *Følging*
- 9 *Datalagring*

beskriver en prosesseringsmodul i X-Analyzer. Kapitlene er likt strukturert: Først presenteres generell bakgrunn, deretter beskrives et knippe kjente metoder fra litteraturen. Videre følger en detaljert beskrivelse av vår implementasjon og resultater. Hvert av kapitlene avsluttes med en kort konklusjon.

Kapittel 10, *Brukergrensesnitt*, presenterer brukergrensesnittene vi har utviklet i forbindelse med denne oppgaven. I kapittel 11, *Evaluering*, drøftes resultatene fra vår implementasjon, relevans, mulige utvidelser og fremtidig arbeid. Kapittel 12, *Konklusjon*, konkluderer rapporten. Her samles trådene fra problemstillingen definert innledningsvis.

Vedlegg A, *Aktive konturer*, beskriver en metode vi eksperimenterte med for skyggefjerning som ikke ga tilfredsstillende resultater. I vedlegg B, *UML*, beskrives implementasjonen i form av UML-klassediagrammer. I vedlegg C, *XML*, presenteres grammatikken i XML-protokollen mellom X-Analyzer og X-Retriever som et XML Schema.

Kapittel 2

Bakgrunn

Dette kapittelet presenterer motivasjonen bak intelligente transportsystemer (ITS). Vi oppsummerer ulike former for trafikkovervåkning, alternative sensorteknologier og relatert arbeid innenfor ITS. Avslutningsvis beskrives kort vårt bidrag til dette feltet.

2.1 Motivasjon

Den overordnede målsetningen til ITS for trafikkovervåkning er at de skal være i stand til å innhente informasjon fra og automatisk analysere en scene slik et menneske ville gjort. Denne informasjonen kan bidra til å ivareta sikkerheten i krysset eller effektivisere avviklingen gjennom det. Å registrere alt som foregår på en bestemt veistrekning er en umulig oppgave. Man må derfor spørre mer spesifikt hvorfor en vil overvåke trafikken. Ifølge [17] har overvåkningsdata slik som trafikktellinger, trafikkintensitet (eng.: *flow rate*) og belegg (eng.: *occupancy*) tradisjonelt blitt brukt til å bistå trafikkplanlegging og administrasjon med å:

- estimere gjenværende kapasitet på motorveier og i kryss.
- finne sykliske luker i trafikken.
- forutse avviklingskvalitet i forbindelse med utbygging av motorvei og bomring.
- tilfredsstille myndigheters krav til rapportering.

2.1.1 Overvåkningstyper

Man kan skille mellom to ulike former for trafikkovervåkning:

- kontinuerlig, og
- midlertidig.

Kontinuerlige overvåkningssystemer er ofte kritiske i den forstand at de er med på å regulere trafikkavviklingen på motorveier og gjennom kryss. Systemene er permanente og integreres gjerne som en del av veistrekningen. Slike systemer krever høy nøyaktighet i observasjonene og pålitelige resultater. Robusthet er også viktig da de må være i stand til å operere kontinuerlig, uavhengig

av skiftende trafikk-, vær- og lysforhold. Permanente systemer brukes i forbindelse med både isolert og sammenkoblet regulering av lyskryss [17]. Med isolert regulering menes regulering i kun ett enkelt kryss i motsetning til sammenkoblet regulering, der trafikken gjennom en serie kryss i et område reguleres sammen for optimal avvikling. Slike systemer eksisterer også i forbindelse med datainnsamling fra motorveier og av-/påkjøringsramper.

På den andre siden eksisterer overvåkningssystemer som plasseres ut på midlertidig basis, for å samle informasjon. Denne informasjonen er ofte av samme type som nevnt over. Hovedforskjellen ligger i formålet for innsamlingen og hvordan registreringen er foretatt. Denne typen systemer har som regel ingen reguleringsoppgave. De er utplassert for å samle data som siden kan analyseres i forsknings-/utbyggingsøyemed eller benyttes av myndighetene for håndhevelse av trafikkreglement. Slik overvåkning omfatter blant annet videoovervåkning samt radar- og laserkontroller for hastighetsregistrering. Midlertidig overvåkning er vanligvis implementert over asfalten.

Vårt system tilhører den siste klassen av overvåkningssystemer. Hensikten er å analysere video fra et ukalibrert kamera posisjonert over et tilfeldig kryss. Det at systemet ikke er integrert i krysset vanskeliggjør analyseoppgaven, siden det ikke kan gjøres antakelser om kameraets plassering, kryssets geometri, oppbygning og metning/belegg.

2.1.2 Informasjon

Mye informasjon kan hentes ut med trafikkovervåkningssystemer. Et skille går mellom systemer som overvåker en veistrekning og systemer som overvåker veikryss. Vi gir her en oppsummering av informasjon det kan være aktuelt å hente ut med et system som overvåker veikryss. Kilder i trafikkforskningsmiljøet[1] har hjulpet oss med å identifisere det følgende:

Svingemønster. Ved å følge objekter gjennom krysset og på denne måten registrere svingemønsteret, får man en oversikt over hvilken kombinasjon av tilfarter/utfarter som er hyppigst brukt. Dette vil være nyttig i forbindelse med dimensjonering av krysset eller omlegging av krysset til for eksempel rundkjøring.

Ventetider. Både for lysregulerte og ikke-lysregulerte kryss vil det være nyttig å registrere ventetider for trafikanter. Dette gjelder både fotgjengere og kjøretøy. Informasjon om ventetid kan bidra til bedre regulering av krysset enten direkte gjennom en automatisk tilbakemeldingsløkke eller indirekte i form av omstrukturering av krysset.

Tidsluker. En tidsluke er tidsintervallet mellom to etterfølgende objekter. Dette henger tett sammen med det forrige punktet. Det er ofte interessant å se på hvordan tidsluker utnyttes for å optimalisere avviklingskvaliteten. Dette gjelder spesielt rundkjøringer der utnyttelse av tidsluker er vesentlig for trafikkavviklingen.

Konfliktsituasjoner. Konfliktsituasjoner kan oppstå mellom kjøretøy (f.eks. høyreregel) eller mellom kjøretøy og fotgjengere. Det finnes idag få eller ingen systemer for automatisk deteksjon av konfliktsituasjoner i kryss.

Hastighet. Ofte vil det være interessant å registrere kjøretøys hastighet i tilfarten inn mot krysset. Hastigheten inn mot krysset er relevant i forbindelse med ulykkesstatistikk og konfliktsituasjoner.

[1] Amanuensis Arvid Aakre ved faggruppe for veg og samferdsel, Institutt for Bygg, Anlegg og Transport, NTNU.

2.2 Innsamlingsmetoder

Ulike innsamlingsmetoder benyttes avhengig av overvåkningstype og hva slags data som skal samles. For trafikktelling i et punkt benyttes tradisjonelt induktive sløyfer nedfelt i asfalten eller trykkfølsomme kabler på tvers av veibanen [17]. Punktregistrering av hastighet gjøres gjerne ved hjelp av radar eller laser. Mer komplekse bevegelsesmønstre over større regioner, kan gjøres med elektroniske sendere i hvert kjøretøy som leses av basestasjoner langs veien (f.eks. bomring) [17]. Videoovervåkning kan benyttes som en selvstendig overvåkningsmetode, eller i kombinasjon med noen av de andre innsamlingsmetodene nevnt i dette delkapittelet.

Induktive sløyfer er permanente installasjoner i veibanen. De krever lite vedlikehold, men de ødelegges etter hvert som asfalten over slites [17]. De ligger derfor sjelden i mer enn ti år av gangen. Installasjon krever at trafikken stoppes midlertidig på den aktuelle veistrekningen [17]. Sløyfene registrerer objektbevegelse ved at et metallobjekt på overflaten induserer et magnetisk felt. Dette feltet kan så registreres. På denne måten telles kjøretøy direkte med en puls for hvert objekt. Dersom flere kjøretøy passerer over en sløyfe samtidig, kan dette føre til at kun en puls genereres [24]. Induktive sløyfer krever åpenbart at objektet som skal telles er av metall og at det er stort nok til å indusere et magnetfelt i sløyfen. Dette er ikke alltid tilfelle for lettere trafikanter som for eksempel syklister og fotgjengere.

Trykkfølsomme kabler på tvers av veibanen [24] registrerer akseltrykk fra kjøretøy som passerer. Antall kjøretøy kan estimeres ved å dividere antall akselpasseringer på en faktor som representerer det mest vanlige antall aksler på et kjøretøy. På samme måte som for induktive sløyfer i veibanen, forutsettes det at objektet har fysiske egenskaper som fører til generering av en puls ved passering (vekt).

Begge disse sensorene opererer med begrenset informasjon. De registrerer kun tilstedeværelsen av et objekt i et punkt og ingen andre attributter knyttet til objektet eller dets bevegelse. De kan ikke klassifisere objekttypen eller innhente informasjon om avvikling og bevegelsesbaner gjennom et veikryss. Til dette har man tradisjonelt benyttet manuell registrering. Slik registrering er nyttig for å kartlegge konfliktsituasjoner, men også veldig ressurskrevende.

Bruk av video for å samle trafikkinformasjon er et relativt nytt (kilde fra 1998), men potensielt nyttig, konsept [24]. Mens andre sensorteknologier bare registrerer begrensede aspekter ved det som foregår, registrerer man med videoovervåkning alt av aktivitet i scenen. Videre kan videoovervåkning benyttes som støtteinformasjon til innsamlede data fra for eksempel induktive sløyfer. I [24] nevnes noen av de parametrene man gjerne overvåker ved hjelp av video.

- Trafikkavvikling
- Svingebevegelser
- Hastighet
- Metning og forsinkelser
- Parkering
- Fotgjengerbevegelser

Video kan benyttes for å overvåke interaksjonen av disse faktorene. På denne måten kan skjulte mønstre og konfliktsituasjoner identifiseres. Dessuten er videoovervåkning relativt kostnadsef-

fektivt, i og med at det krever lite vedlikehold. Siden videoovervåkningsutstyr må plasseres ut over bakken, kan slike installasjoner bli utsatt for sabotasje og hærverk. Dette er med på å gjøre systemene til dels upålitelige. Pålitelighet er kritisk i forbindelse med automatisk regulering av for eksempel signaler i lyskryss.

Analysen av videomaterialet kan gjøres både manuelt og automatisk. Automatisk analyse inkluderer prosessering i form av bevegelsesegmentering, følging og klassifisering. Slik prosessering er beregningskrevende og sensitivt mot visuell støy.

2.3 Eget bidrag

Siden det per idag ikke eksisterer generelle systemer for å automatisk detektere konfliktsituasjoner og følge trafikanter gjennom overvåkede veistrekninger og kryss, har vi sett nærmere på dette. Vårt system er ment å kunne analysere video fra et ukalibrert kamera posisjonert over et tilfeldig kryss. Kameraet er ukalibrert i den forstand at attributter som observasjonsvinkel og kamerakoordinater ikke er tatt med i prosesseringen. Systemet samler data som ikke er ment å regulere avviklingen i krysset direkte, men snarere lette analyse av situasjoner og identifisering av konflikter. Selv om systemet primært er ment å overvåke veikryss, kan det også analysere data fra en tilfeldig veistrekning. En veistrekning kan betraktes som et kryss med kun to tilfarter.

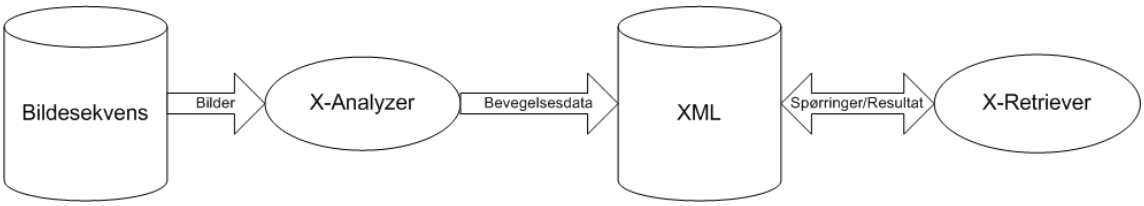
Konfliktsituasjoner eller bevegelsesmønstre som uttrykker konflikt, skal kunne spesifiseres av brukeren. Et typisk eksempel på en konfliktsituasjon i et kryss vil være et kjøretøy som ikke viker for en bil som kommer fra høyre i et kryss regulert ved vanlig høyreregel.

Vår tilnærming er basert på automatisk uttrekning av informasjon fra videooveråkningsmateriale. Visuell overvåkning er den eneste innsamlingsmetoden som muliggjør følging av trafikanter gjennom scenen og registrering av svingemønstre. Videoovervåkning og automatisk analyse blir mer og mer vanlig ettersom økt prosesseringskraft og stadig mer sofistikerte mekanismer for etterbehandling dukker opp. Ved å segmentere, klassifisere og følge objekter gjennom scenen, vil vi forsøke å generere en representasjon av det som foregår i scenen. Denne informasjonen skal så struktureres på en måte som letter gjenfinning av interessant informasjon. Gjenfinningssystemet benytter resultater fra analysen for å finne situasjoner og sekvenser av objektbevegelse som kan tyde på en konfliktsituasjon.

Kapittel 3

Systembeskrivelse

Dette kapittelet beskriver funksjonalitet og arkitektur i vår implementasjon. Vi har bygd en applikasjon, X-Analyzer, som analyserer bildesekvenser fra vilkårlige veikryss. Med bakgrunn i en brukerdefinert scenedefinisjon trekker X-Analyzer relevant informasjon ut av en bildesekvens. Denne informasjonen struktureres på en måte som fasiliterer gjenfinning av interessante hendelser. Gjenfinning er implementert i X-Retriever. Sistnevnte er et enkelt brukergrensesnitt for å formulere spørringer mot XML-data generert av X-Analyzer. Figur 3.1 illustrerer den løse koblingen mellom disse to applikasjonene.



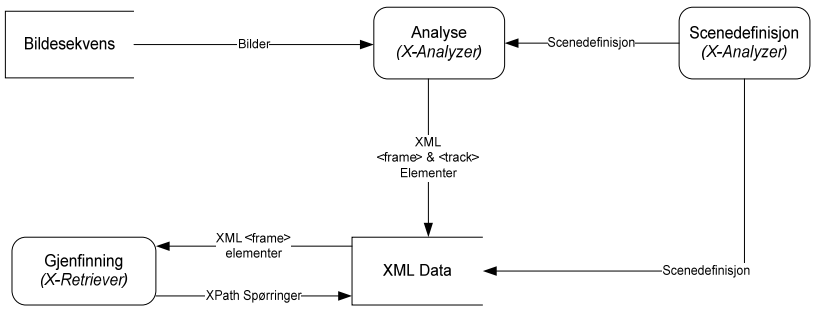
Figur 3.1: Hovedkomponenter

For utdypende UML pakke- og klassediagrammer henviser vi til vedlegg B.

Fra brukerens ståsted består systemet av tre kronologiske hovedprosesser:

1. Scenedefinisjon
2. Analyse
3. Gjenfinning

Sammenhengen mellom disse prosessene er illustrert i figur 3.2. Først spesifiserer brukeren en scenedefinisjon. Denne danner rammeverket for hvordan bildeinformasjonen analyseres og struktureres. Analysemodulen genererer en XML-representasjon av interessant informasjon i bildesekvensen. Denne XML-representasjonen danner så grunnlaget for gjenfinning av aktuelle situasjoner/sekvenser.



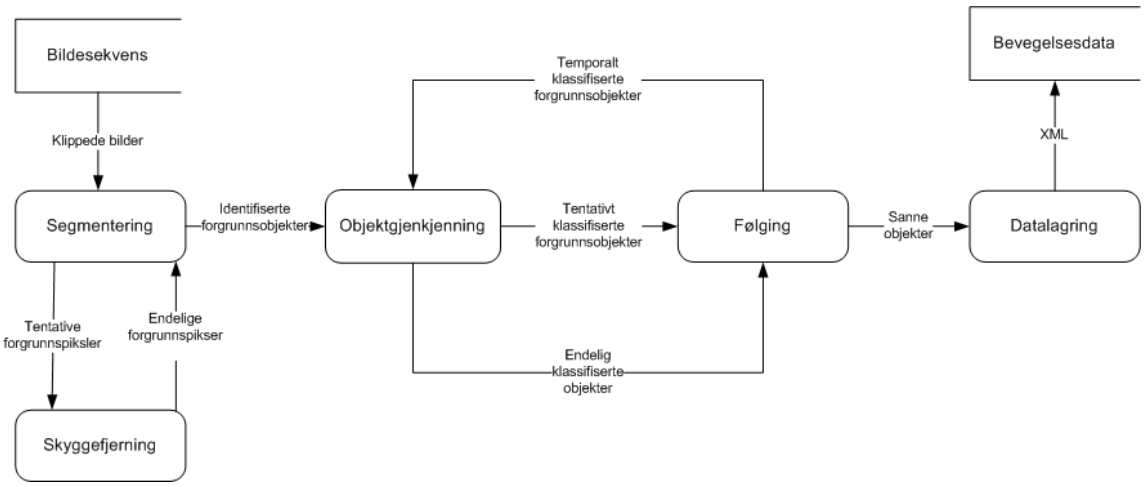
Figur 3.2: Hovedprosesser

3.1 Scenedefinisjon

Scenedefinisjonen beskriver kryssets geometri i form av et klippevindu og et polygon som avgrenser selve krysset. Polygonet deles i regioner med brukervalgte merkelapper for hver region. Scenedefinisjonen skrives til XML og sendes til analyseprosessen. Scenedefinisjonsverktøyet (SDT), er implementert som en del av X-Analyzer. Interaksjonssekvensen for scenedefinisjon er nærmere beskrevet i kapittel 10.

3.2 Analyse

I analysen trekkes relevant informasjon ut av bildesekvensen og lagres til XML i samsvar med scenedefinisjonen. Dette er hovedfunksjonaliteten i X-Analyzer. Modulene som utgjør analyseprosessen, er illustrert som runde rektangler i figur 3.3. Funksjonaliteten til hver av disse er beskrevet kort i det følgende.



Figur 3.3: Moduler i analysen

3.2.1 Segmentering

I segmenteringsmodulen identifiseres forgrunnsregioner i hvert bilde. Segmenteringsmodulen er beskrevet i detalj i kapittel 5. Modulen vedlikeholder en statistisk bakgrunnsmodell for hver piksel. Bakgrunnsmodellen danner grunnlaget for pikselklassifisering (forgrunn/bakgrunn) og oppdateres for hvert bilde i sekvensen. Inndata til segmenteringsmodulen er bildene i sekvensen. Utdata er sett med punkter som beskriver regioner med sammenhengende forgrunnspiksler.

3.2.2 Skyggefjerning

Skyggefjerning er implementert som en del av segmenteringsmodulen. Funksjonalitetsmessig er den likevel beskrevet som en egen modul i kapittel 6. Modulen opererer på pikselbasis. For hver piksel klassifisert som forgrunn i segmenteringsmodulen, bestemmer skyggefjerningsmodulen om denne pikselen egentlig representerer skygge. Hvis skyggefjerningsmodulen identifiserer pikselen som skygge, omklassifiseres pikselen som bakgrunn. Skyggefjerningsmodulen er derfor med på å forbedre forgrunnsobjektene. Inndata til skyggefjerningsmodulen er piksler tentativt klassifisert som forgrunn, mens utdata er piksler som er endelig klassifisert som forgrunn.

3.2.3 Objektgjenkjenning

Forgrunnsregionene identifisert i segmenteringen, klassifiseres deretter som kjøretøy eller fotgjenger i objektgjenkjenningsmodulen. Denne er nærmere beskrevet i kapittel 7. For hvert objekt i hvert bilde, beregnes en representasjon basert på fasong. Hvert objekt klassifiseres tentativt ut ifra denne representasjonen. Siden en regions fasong i et enkelt bilde ikke nødvendigvis er representativ for objektets sanne fasong, benyttes multiple hypoteser for den endelige klassifiseringen. Objektgjenkjenningsmodulen samarbeider med følgingsmodulen. Følgingsmodulen akkumulerer bevis for et objekts klasse, for hvert bilde objektet blir fulgt gjennom. Når tilstrekkelig bevis taler

for klassifisering som enten fotgjenger eller kjøretøy, klassifiseres objektet som en av disse. Inndata til objektgjenkjenningsmodulen er identifiserte forgrunnsregioner, mens utdata er tentativt klassifiserte forgrunnsobjekter.

3.2.4 Følging

Følgingsmodulen etablerer temporal koherens mellom objekter fra etterfølgende bilder. Modulen er nærmere beskrevet i kapittel 8. Følging gjøres prediktivt i form av et estimat av objektets posisjon i neste bilde og påfølgende verifikasjon av dette estimatet. Estimeringen gjøres med bakgrunn i objektets hastighetsvektor. Videre er følgingsmodulen med på den temporale klassifiseringen av objekter ved å akkumulere tentative klassifiseringsresultater for hvert objekt. Inndata til følgingsmodulen er lister med regioner som beskriver temporalt ukorrelerte objekter i individuelle bilder. Utdata er en beskrivelse av objektenes bevegelsesbaner.

3.2.5 Datalagring

Datalagringsmodulen har som oppgave å omdanne hver sti fra følgingsmodulen til en XML-representasjon. Denne modulen er beskrevet i kapittel 9. Datalagring gjøres på per bilde basis (`<frame>`-elementer) og per sti basis (`<track>`-elementer). For hvert bilde lagres informasjon om objektene i bildet. For hver sti lagres informasjon om hvordan et objekt har beveget seg gjennom scenen. På denne måten lagres informasjon om objekters posisjon flere ganger og vi innfører redundans i XML-representasjonen. Denne redundansen hjelper oss til enkelt å identifisere interessante hendelser i gjenfinningsapplikasjonen.

3.3 Gjenfinning

Når en bildesekvens er analysert og relevant informasjon i scenen er formatert til XML, kan brukeren finne igjen situasjoner og sekvenser ved hjelp av verktøyet X-Retriever. Dette er et enkelt skall for å formulere spørringer grafisk. Spørringene eksekveres som X-Path uttrykk i XML-representasjonen. X-Path spørringene returnerer XML-elementer som beskriver interessante bilder i sekvensen. To ulike spørringstyper kan formuleres:

Situasjonsspørringer. En situasjon beskriver en statisk konfigurasjon av objekter. En situasjon er statisk i den forstand at den ikke uttrykker objektbevegelse over tid, men er et øyeblikksfotografi (eng.: *snapshot*) av scenen. En situasjon er definert ved et sett objekter med en klasse (kjøretøy/fotgjenger), posisjon og retning. Situasjonsspørringer tar utgangspunkt i `<frame>`-elementene i XML-representasjonen.

Sekvensspørringer. En sekvens uttrykker en dynamisk konfigurasjon av objekter. En sekvens er dynamisk i den forstand at den har en temporal dimensjon. En sekvensspørring er definert ved et sett objekter og en angivelse av disse objektenes bevegelse. Sekvensspørringer tar utgangspunkt i `<track>`-elementene i XML-representasjonen.

Siden X-Retriever hovedsaklig er et brukergrensesnitt for grafisk spørringsformulering, er det nærmere beskrevet i kapittel 10 som omhandler brukergrensnitt og interaksjon.

Kapittel 4

Datasett

I dette kapittelet diskuteres datasettene vi har brukt til testing av X-Analyzer. Vi vil først drøfte generelle egenskaper knyttet til ulike formater og begrunne våre valg med tanke på valgt format. Deretter følger en diskusjon av kriterier knyttet til datasett for vår applikasjon. Til slutt vil vi beskrive egenskaper knyttet til innholdet i de spesifikke datasettene vi har valgt å benytte.

4.1 Formater

Video er som kjent en rekke bilder som vises på skjermen i en sekvens som gir illusjonen av kontinuerlig bevegelse. Antall bilder som vises per sekund i et videoklipp kalles bilderaten (eng.: *frame rate*). PAL standarden som benyttes i blant annet Europa, Kina og Australia har en bilderate på 25 bilder per sekund, mens NTSC standarden som benyttes i blant annet USA og Japan benytter 30 bilder per sekund [20]. Videodata er enormt plasskrevende. Dersom vi antar et 30 sekunders videoklipp med oppløsning 720×576 piksler, bilderate 30 bilder/sekund og 3 byte (RGB) per piksel gir dette et plassbehov på rundt 911MB. Slike datavolum er åpenbart for stort for de fleste applikasjoner. Som et første steg i datareduksjonen valgte vi derfor å plukke ut hvert femte bilde fra videosekvensene. Dette valget ble tatt etter eksperimentering med ulike bilderater. Det viste seg at 5 bilder per sekund ga en fornuftig avveining mellom datavolum og temporal detalj for tilfredsstillende segmentering og følging.

Det eksisterer en rekke former for komprimering som reduserer plassbehovet ytterligere. Prisen man betaler er en liten reduksjon av videokvaliteten. De fleste komprimeringsalgoritmene utnytter at det er liten forskjell i piksler mellom etterfølgende bilder. Istedenfor å eksplisitt lagre hvert bilde kan man derfor lagre kun forskjellen mellom etterfølgende bilder. MPEG standardene (MPEG-1/2/4) for videokomprimering utnytter dette for prediktiv koding av bilder og en frekvenstransformasjon (DCT) som reduserer datavolumet ytterligere [20].

For å lese ut individuelle bilder fra et videoklipp eksperimenterte vi med Java Media Framework (JMF) API-et fra Sun [23]. Vi implementerte et enkelt preprosesseringssteg som returnerte individuelle bilder fra videoklippet. Siden vi støtte på en rekke problemer ved bruk av JMF, valgte vi heller å representere våre testsett gjennom en sekvens av bildefiler. Konvertering av video til bilder gjorde vi ved hjelp av et videoredigeringsprogram (VirtualDub [34]).

Eksperimentering med ulike bildeformater viste at JPEG formatet ga adekvate resultater selv om denne komprimeringen ikke er tapsfri (eng.: *lossless*) [20]. I utgangspunktet var råmaterialet vårt fargevideo med romlig oppløsning 720×576 piksler. Vi reduserte den romlige oppløsningen på datasettet til 360×288 piksler. Implementasjonen fungerer med vilkårlig fargedybde og romlig oppløsning. For at skyggefjerningsmodulen beskrevet i kapittel 6 skal fungere, forutsettes RGB-fargebilder. For innlesing av bilder og intern representasjon har vi benyttet API-et Java Advanced Imaging (JAI) fra SUN [22]. JAI tilbyr effektive implementasjoner av vanlige bildebehandlingsoperasjoner optimalisert gjennom plattformspesifikk kode.

4.2 Kriterier

For at X-Analyzer skal gi tilfredsstillende resultater, er det flere kriterier knyttet til systemets inndata. Ideelt bør klippene være filmet med et fullstendig stillestående kamera. Dersom kameraet beveger seg, forstyrrer dette segmenteringen som er basert på intensitetsforskjeller i individuelle piksler mellom etterfølgende bilder.

Kameraet bør være plassert i en så loddrett vinkel som mulig over scenen som skal analyseres. Dette er et vesentlig poeng for denne typen applikasjoner siden bildene kun er todimensjonale perspektivprojeksjoner av den virkelige scenen. Ved å plassere kameraet rett over scenen, reduseres problemet med overlappende objekter (f.eks. biler i ulike felt), problemer knyttet til at objektene forandrer seg gjennom krysset avhengig av vinkel (front, bak, side) og problemer knyttet til størrelsesforandring gjennom scenen (perspektiv). En slik kameraposisjon er vanskelig å få til i praksis. Det må derfor gjøres en avveining mellom vinkel og hvor det faktisk er mulig å få posisjonert kameraet. Vi har i stor grad prøvd å finne steder der vi har vært i stand til å filme rett ned på krysset.

Videre bør videoklippene i så stor grad som mulig være frie for støyende objekter slik som fallende blader, snø, regn, etc. Skygge som følge av lav sol og refleksjoner i veibanen fra frontlys som følge av våt asfalt, er også med på å gjøre videoanalysen vanskeligere.

4.3 Våre datasett

Filmingen er gjort med et ukalibrert Canon digitalt videokamera (modell MV30i) plassert på et stillestående stativ. Kameraet er ukalibrert i den forstand at kameraets koordinater ikke blir tatt hensyn til i den videre prosesseringen. Vi deaktiverte kameraets autofokusfunksjon for å unngå støy i bildene introdusert ved fokusering da dette var en feilkilde i [29]. Vi deaktiverte også kameraets mekanisme for interlacing og automatisk regulering av lysforhold.

Vi har filmet to testsett med ulike egenskaper. Det første testsettet er i et oversiktlig lyskryss med relativt lite trafikk. Det andre testsettet er bare ca. 65 % av lengden til det første, men til gjengjeld betraktelig mer komplisert. Begge testsettene ble filmet på ettermiddagen den 10. januar 2005, med tilnærmet like lys og skyggeforhold. Lysforholdene er dunkle og objektene beveger seg på svart, våt asfalt. Dette kompliserer vår oppgave da våt asfalt forsterker refleksjoner fra kjøretøyenes lykter. Begge testsettene inneholder en rekke ulike kjøretøy (buss, bil, lastebil), fotgjengere og syklister.

4.3.1 Høgskoleringen

Figur 4.1: Høgskoleringen

Dette klippet består av 4491 bilder. Med en bilderate på 5 bilder per sekund, representerer dette totalt 14 minutter og 48 sekunder overvåkning. Totalt 114 objekter beveger seg gjennom det utsnittet av datasettet vi studerer. Figur 4.1 viser krysset uten noen objekter. Høgskoleringen er vårt minst komplekse datasett. Krysset er oversiktlig, det er lite trafikk og det er en god vinkel på kameraet. Testsettet er filmet fra en bro i Øvre Allé. At kameraet er plassert på en bro fører til problemer med bevegelse i kameraet, når kjøretøy passerer over broen. Dette resulterer i støy ved segmentering av enkelte bilder. Trafikklyset som endrer farge med jevne mellomrom kan oppfattes som objektbevegelse. Som man kan se har vi noen steder i scenen problemer med bakgrunnsobjekter (skilt, trær, lyktestolpe) som kommer i forkant og dekker til de bevegelige objektene vi ønsker å segmentere. Siden testsettet ikke er filmet rett ovenfra har vi også problemer med at forgrunnsobjekter tildekker hverandre.

4.3.2 Samfundet

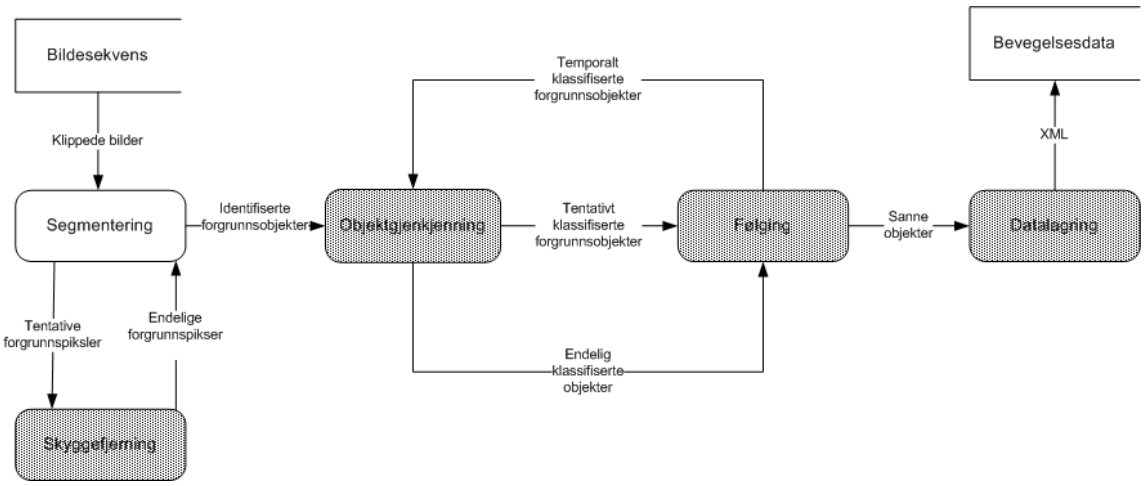
Dette klippet består av 2932 bilder. Med en bilderate på 5 bilder per sekund, representerer dette totalt 9 minutter og 46 sekunder overvåkning. Totalt 252 objekter beveger seg gjennom det utsnittet av datasettet vi studerer. Klippet er filmet i nordlig retning fra gesimsen over inngangen på Studentersamfundet i Trondheim. Figur 4.2 viser ett av bildene fra settet. Dette klippet er mer komplisert med tanke på segmentering og følging. Ettersom krysset er mer trafikkert er problemet med overlappende objekter større. Som i avsnitt 4.3.1 har vi også her problemer med trafikklys som endrer farge og dermed registreres som objektbevegelse. Perspektivet er et større problem i denne scenen. Objektene er relativt små i det de kommer inn/-forsvinner øverst i bildet, mens de får en større representasjon lenger ned i bildet.

Figur 4.2: Samfundet

Kapittel 5

Segmentering

Dette kapittelet beskriver segmenteringsmodulen i X-Analyzer. Som det fremgår av figur 5.1, samarbeider segmenteringsmodulen med skyggefjerningsmodulen om å identifisere piksler som forgrunnsregioner.



Figur 5.1: Segmenteringsmodulens plassering

Segmenteringsmodulen er den første prosesseringsmodulen i X-Analyzer. Målsetningen til segmenteringsmodulen er å finne alle forgrunnsobjektene. Forgrunnsobjekter er i denne sammenhengen regioner som flytter på seg fra et bilde til det neste. Segmenteringsmodulen danner dermed grunnlaget for videre skyggefjerning, klassifisering og følging. Modulen beregner posisjonen og utstrekningen til objektene i hvert enkelt bilde i en sekvens av bilder. Resultatet av segmenteringen er objektene i hvert enkelt bilde. Objektene representeres ved sett av sammenhengende piksler tilordnet en merkelapp.

5.1 Generelt

En måte å se segmenteringsoppgaven på, er at bakgrunnen fjernes og kun forgrunnsobjekter gjenstår. Dette gir opphav til begrepet bakgrunnssubtraksjon. Bakgrunnssubtraksjon går i enkelhet ut på å beregne et bakgrunnsbilde uten forgrunn. Estimatet av bakgrunnen kan så subtraheres fra bilder, for å isolere forgrunnsobjektene.

McIvor nevner i [30] tre parametre knyttet til bevegelsessegmentering. Disse parametrene danner grunnlaget for den videre diskusjonen i dette kapittelet.

Forgrunnsdeteksjon. Hvordan selve deteksjonen av forgrunnsobjekter fungerer.

Bakgrunnsoppdatering. Hvordan bakgrunnsmodellen oppdateres over tid.

Postprosessering. Hvordan segmenterte objekter er postprosessert for å fjerne støy og falske positiver (eng.: *false positives*).

Bevegelsessegmentering kan sies å være lettere enn tradisjonell segmentering, siden vi har mer informasjon tilgjengelig gjennom den temporale dimensjonen. Det er likevel en rekke faktorer som kompliserer segmenteringsoppgaven. Noen av disse problemene er knyttet til at bakgrunnen ikke er statisk. Pikselfordelingen i bildet kan endres uten at dette skyldes bevegelse av objekter man ønsker å segmentere. En god segmenteringsmetode må derfor være robust mot følgende situasjoner:

- Lysendringer kan skje gradvis ved skumring/soloppgang eller mer plutselig når for eksempel skyer skygger for solen.
- Høyfrekvente bevegelser i bakgrunnsobjekter, som f.eks. grener, blader, bølger, nedbør, etc., kan gi opphav til multimodale bakgrunnsmodeller. Dette er nærmere forklart i avsnitt 5.2.2.
- Bevegelse av kameraet vil kunne oppfattes som bevegelse av alle piksler i hele bildet.
- Når for eksempel biler parkerer, blir de statiske objekter. Disse må etterhvert inkluderes i bakgrunnen.

Andre problemer er knyttet til forgrunnsobjektene:

- Okklusjon. Siden det er vanskelig å få filmet scenen ovenfra, får vi problemer med at objekter ofte er delvis skjult bak andre objekter i scenen. Det viser seg å være vanskelig å skille overlappende objekter fra hverandre i segmenteringsmodulen. Dette bør derfor gjøres i følgingsmodulen, der ytterligere informasjon om objektenes bevegelse og historie gjennom scenen er tilgjengelig.
- Lokale lysendringer. Objekter kaster skygger som vil bevege seg med det segmenterte objektet gjennom scenen. Skygger er et problem i forbindelse med klassifisering og følging av objekter. Teknikker for skyggefjerning er integrert i segmenteringsmodulen og er diskutert i kapittel 6.
- Forgrunnsobjekter med pikselverdier tilnærmet lik bakgrunnen er vanskelig å segmentere riktig. Dette problemet er størst i forbindelse med gråtonebilder.

Også feil i kameraet knyttet til fokus eller andre fotoelektriske fenomener kompliserer segmenteringen [29].

5.2 Kjente metoder

I dette delkapittelet vil vi se på hovedprinsippene i kjente metoder for bevegelsessegmentering. Disse vil bli tatt for seg i avsnittene under og inkluderer:

Analyse av differansebilder. Prinsippet er å beregne et stasjonært bakgrunnsbilde, for så å identifisere bevegelse som de pikslene som skiller seg signifikant ifra bakgrunnen [7].

Statistiske modeller. Observerte verdier klassifiseres i henhold til en sannsynlighetsfordeling, som beskriver hver piksellokasjons bakgrunnsverdier. Bakgrunnsmodellen oppdateres kontinuerlig.

Optisk flyt. Segmentering ved optisk flyt, beregner bevegelse i form av hastighetsvektorer for hver piksel i bildet.

5.2.1 Differansebilder

I sin enkleste form går differansebildemetodene ut på å beregne differansebildet, D_t, ved å subtrahere et estimat av bakgrunnen, B, fra hvert enkelt bilde, I_t, fra sekvensen.

$$D_t = |I_t - B| \tag{5.1}$$

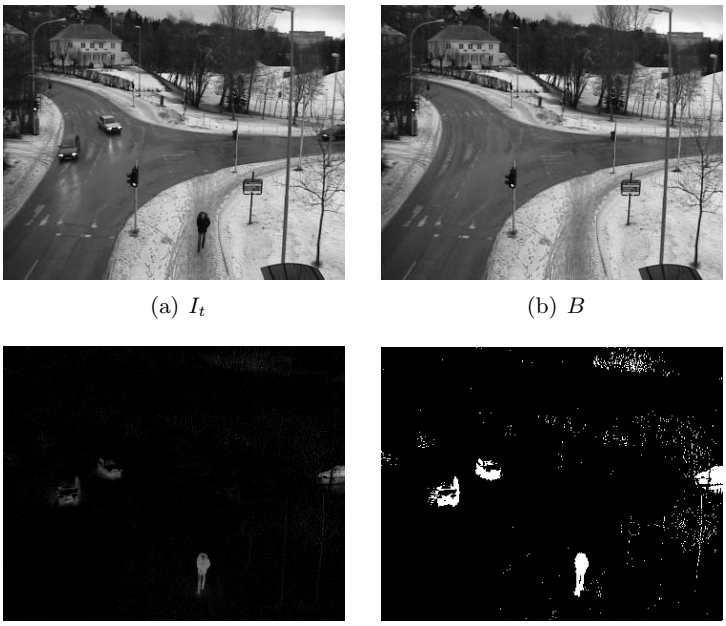
Differansebildet, D, vil, bortsett fra i det ideelle tilfellet, inneholde mer informasjon enn forgrunnsobjektene. For å skille ut denne informasjonen fra forgrunnen benyttes tersklingsoperasjonen på D. Terskelverdien, τ, bestemmes ut ifra mengden støy og bakgrunnsendringer [29]. Som regel er τ en statisk verdi som ikke endres etter initialisering. Tersklingsoperasjonen er formalisert i ligning 5.2.

$$S_t(x,y) = \begin{cases} 0, & D_t(x,y) < \tau \\ 1, & D_t(x,y) \geq \tau \end{cases} \tag{5.2}$$

Resultatet av tersklingen er et binærbilde, S_t, der 1 angir forgrunn og 0 angir bakgrunn. S_t kan benyttes som maske, for å rekonstruere forgrunnsobjekter. Til dette benyttes binæroperasjonen S_t AND I_t.

Felles for alle metoder basert på differansebilder og terskling, er at de er sensitive med hensyn på valg av τ. For liten τ vil gi for mange falske positiver, mens for stor verdi vil gi en for restriktiv segmentering. En for restriktiv segmentering resulterer i diskontinuiteter i de segmenterte objektene.

Figur 5.2 viser resultatet av en segmentering som beskrevet ovenfor.



(a) I_t (b) B

(c) $D_t = I_t - B$ (d) S_t $\tau = 20$

Figur 5.2: Enkel segmentering ved bruk av differansebilder. Figur 5.2(a) og figur 5.2(b) viser operandene i ligning 5.1. Figur 5.2(c) viser resultatet av subtraksjonen. Figur 5.2(d) viser binærbildet etter tersklingen beskrevet i ligning 5.2.

Bakgrunnsoppdatering

I trafikkerte kryss er man sjelden i den situasjonen at man har ett bakgrunnsbilde, B, som kan subtraheres fra hvert enkelt bilde, slik som i figur 5.2. Dessuten bør bakgrunnsmodellen være adaptiv, slik at lysendringer og nye statiske objekter gir konvergens mot nye stabile tilstander. Bakgrunnsmodellen for hver piksel bør derfor baseres på pikselens historie. Vi ser derfor på B_t, det vil si bakgrunnsmodellen ved tidspunkt t.

Foregående bilde Den enkleste måten å estimere en slik bakgrunnsmodell, er å sette den til å være foregående bilde. Ligning 5.1 blir da $D = |I_t - I_{t-1}|$. Denne tilnærmingen virker bare under spesielle antakelser om objektenes hastighet og antall bilder per sekund [29]. Differansen mellom etterfølgende bilder, vil dessuten segmentere hovedsaklig konturen av objektene. Dersom heller ikke denne er kontinuerlig, krever metoden mye postprosessering for å knytte sammen linjesegment som utgjør objektkontur.

Gjennomsnitt En annen måte å estimere bakgrunnen, er å beregne B_t som *gjennomsnittet* eller *medianen* av de siste n bildene i sekvensen. På denne måten vil bevegelige objekter etter hvert viskes ut av bakgrunnen. Det er også vanlig å beregne et løpende gjennomsnitt (running average) av de foregående bilder i sekvensen. Dette benyttes blant annet i [11, 42, 7]. Ligning 5.3 beskriver bakgrunnsoppdateringen som et førsteordens rekursivt filter, som oppdateres for hver iterasjon gjennom bildesekvensen.

$$B_{t+1} = \alpha I_t + (1 - \alpha) B_t \tag{5.3}$$

I ligning 5.3 representerer α læringskoeffisienten. Denne bestemmer hvor mye innflytelse nye bilder i sekvensen skal ha på bakgrunnsoppdateringen. Jo større α er, desto raskere blir endringer i scenen inkludert i bakgrunnsmodellen. På denne måten avtar hvert enkelt bildes bidrag til bakgrunnen eksponensielt, etterhvert som det forsvinner inn i historien. Eksponensiell glemming er ekvivalent med å benytte Kalman-filter for å spore bakgrunnsbildet [7]. Dersom α velges for stor, kan dette føre til at kunstige haler dannes i bak-kant av objekter. For å unngå dette, men allikevel få en adaptiv oppdatering når nye statiske bakgrunnsobjekter dukker opp, benytter [11] to korreksjoner:

1. Dersom en piksel er markert som forgrunn i mer enn m av de siste M bildene, så oppdateres bakgrunnen som $B_{t+1} = I_t$. Dette kompenserer for brå lysendringer og nye statiske objekter (biler som parkerer).
2. Dersom en piksel endrer tilstand fra forgrunn til bakgrunn hyppig, tas den ikke med i forgrunnsberegninger. Dette kompenserer for svaiende grener og andre høyfrekvente endringer.

Parametrisk oppdatering W^4, beskrevet i [10], benytter en helt annen metode for å modellere bakgrunnen. For hver piksellokasjon vedlikeholdes tre parametre. Disse oppdateres kun for piksler i innkommende bilder som klassifiseres som bakgrunn.

- $M(x, y)$ representerer minimumsverdi for bakgrunnspiksler i (x, y).
- $N(x, y)$ representerer maksimumsverdi for bakgrunnspiksler i (x, y).
- $D(x, y)$ representerer den største intensitetsforskjellen mellom etterfølgende bilder for bakgrunnspiksler i (x, y).

Et piksel markeres som forgrunn dersom:

$$|M - I_t| > D \text{ eller } |N - I_t| > D \tag{5.4}$$

Parametrene estimeres initielt ut ifra de første sekundene med video. En svakhet ved denne, og andre metoder basert på differansebilder, er at de forutsetter en tom bakgrunn for initialisering. Underveis oppdateres M, N og D periodisk for de delene av scenen som klassifiseres som bakgrunn. Dette kalles *selektiv* oppdatering (i motsetning til *blind* oppdatering). Selektiv oppdatering kan generaliseres til de fleste metodene basert på differansebilder. Hver piksel i hvert innkommende bilde, $I_t(x, y)$, kan før bakgrunnsoppdateringen klassifiseres som enten forgrunns- eller bakgrunnspiksel. Dersom $I_t(x, y)$ klassifiseres som forgrunn, ignoreres den i bakgrunnsmodellen. På denne måten unngår vi å få bakgrunnsmodellen forurenset av piksler som ikke tilhører bakgrunnen. Ligning 5.5 uttrykker selektiv oppdatering basert på løpende gjennomsnitt.

$$B_{t+1}(x, y) = \begin{cases} \alpha I_t(x, y) + (1 - \alpha) B_t(x, y) & \text{hvis } I_t(x, y) \text{ er bakgrunn} \\ B_t(x, y) & \text{hvis } I_t(x, y) \text{ er forgrunn} \end{cases} \tag{5.5}$$

Klassifiseringen i ligning 5.5 kan gjøres basert på statiske terskelverdier som definerer gruppetilhørighet for ulike pikselverdier. En annen form for selektiv bakgrunnsoppdatering er beskrevet i [45].

Andre klassifiseringsmodeller tar i bruk statistiske metoder og beregner sannsynlighetsfordelinger for hver piksellokasjon. Man kan så for eksempel benytte hypotesetester for å avgjøre om pikselen tilhører forgrunnen eller bakgrunnen [45]. Dette leder oss over i neste avsnitt som omhandler statistisk bakgrunnsanalyse.

5.2.2 Statistisk bakgrunnsanalyse

Statistiske bakgrunnsmodeller beregner sannsynlighetsfordelinger for hver piksellokasjon. For hver sannsynlighetsfordeling beskriver $P(X = x)$ sannsynligheten for å observere pikselverdi x på denne lokasjonen. Statistiske modeller oppdateres adaptivt for hvert nye innkommende bilde. Vi har valgt å se nærmere på to typer statistiske bakgrunnsmodeller, nemlig unimodale og multimodale bakgrunnsmodeller.

Unimodale bakgrunnsmodeller

Unimodale bakgrunnsmodeller beskriver hver piksellokasjon som én sannsynlighetsfordeling. Sannsynlighetsfordelingen for hver piksel oppdateres adaptivt for hvert innkommende bilde. For å beskrive en normalfordeling, trengs estimater for forventningsverdi, μ, og varians, σ^2. Siden disse parametrene endres etterhvert som nye bilder prosesseres, trenger vi adaptiv oppdatering av disse. Det er vanlig å benytte de k første bildene i sekvensen til å initialisere parametrene. Deretter oppdateres de fortløpende ved bruk av samme rekursive filter som i ligning 5.3.

Også i forbindelse med statistiske bakgrunnsmodeller, vil selektiv oppdatering være nyttig, for å unngå å forurense bakgrunnsmodellen med forgrunnspiksler. De fleste modeller benytter derfor et klassifiseringssteg forut for oppdatering, slik at kun piksler som klassifiseres som bakgrunnspiksler bidrar til oppdatering av bakgrunnsmodellen. En enkel klassifikator kan etableres for hver piksellokasjon, ved å se på forventningsverdien til bakgrunnens sannsynlighetsfordeling. Man kan da anta at forgrunnspiksler, ligger lenger unna forventningsverdien for bakgrunnen enn bakgrunnspiksler. Det vil også være interessant å vurdere en innkommende piksel i forhold til variansen. Man kan derfor enkelt beregne en maske S som segmenterer forgrunnsobjekter ved

$$S(x,y) = \begin{cases} 0 & \text{hvis } |I_t(x,y) - \mu_{t-1}(x,y)| < \lambda\sigma_{t-1} \\ 1 & \text{ellers} \end{cases} \tag{5.6}$$

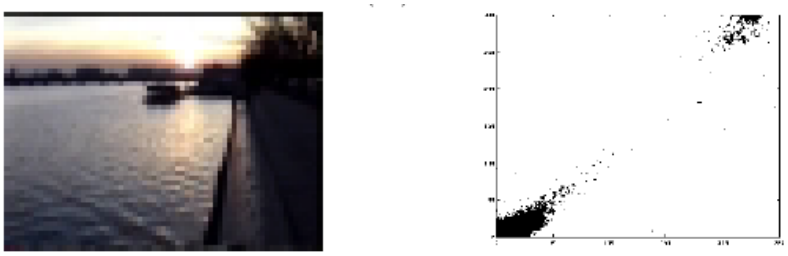
der λ representerer hvor mange standardavvik man tillater piksler i et innkommende bilde, I_t, å avvike fra forventningsverdien.

Multimodale bakgrunnsmodeller

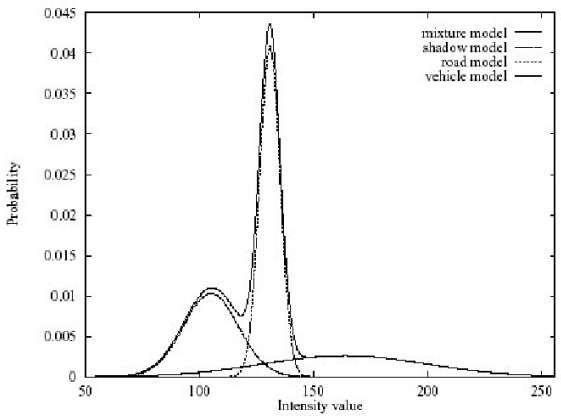
Multimodale bakgrunnsmodeller beskriver bakgrunner med flere forventningsverdier enn én. For eksempel vil blader/grener som vaier i vinden, flagg, blinkende neonskilt, etc., ikke kunne beskrives ved én normalfordeling. Dette er fordi pikselverdiene endres for mye (mellom to modi)

på kort tid. Figur 5.3 illustrerer en bimodal bakgrunn. Til høyre ser vi et bilde av et vann og til venstre et bimodalt spredt plot av én av vannpikslenes verdier over tid. Det er tydelig fra figur 5.3 at en vannpiksel inntar verdier innenfor hovedsaklig to klynger (eng. *clusters*). Dette skyldes solens refleksjoner i vannet som skifter mellom hovedsaklig to modi. En enkel normalfordeling er ikke i stand til å modellere multimodale bakgrunner som dette.

Miksturmodeller benyttes for å modellere multimodale bakgrunner. En miksturmodell er en blandingsmodell bestående av flere normalfordelinger (eng.: *mixture-of-Gaussians*). I [7, 14, 27, 33] modelleres multimodal bakgrunn som en mikstur av K normalfordelinger. Hver av de K komponentene beskriver én bakgrunnsmodalitet. I en multimodal fordeling, klassifiseres hver piksel i henhold til den komponenten som best modellerer denne pikselens modus. Miksturkomponenten for hver klasse oppdateres med bakgrunn i sannsynligheten for tilhørighet. Som regel velges K et sted mellom 3 og 5 [33].



Figur 5.3: Bimodal bakgrunn [33]



Figur 5.4: Plot av miksturmodell, $K = 3$ [7]

I [7] settes $K = 3$, og de ulike komponentene av miksturmodellen antas å modellere asfalt, skygge og kjøretøy. Ved å definere en sannsynlighetsfordeling for skygge, oppnår man automatisk skyggefjerning. Figur 5.4 viser et plot av miksturmodellen som benyttes til klassifisering i [7].

Matematisk modelleres historien til hver piksel, $X_1, X_2, \ldots, X_t$, som en K-komponents miksturmodell av normalfordelinger. Sannsynligheten for å observere pikselverdien, X_t, på en gitt pikselokasjon, er i [33] gitt som

$$P(X_t) = \sum_{i=1}^{K} \omega_{i,t} \eta(X_t; \mu_{i,t}, \Sigma_{i,t}) \quad (5.7)$$

der $\eta(X_t; \mu_{i,t}, \Sigma_{i,t})$ beskriver normalfordelingen til den i'te komponenten gitt ved parametrene gjennomsnittet, $\mu_{i,t}$, og kovariansmatrisen, $\Sigma_{i,t}$. $\omega_{i,t}$ representerer bidraget fra den $i$te normalfordelingen.

Det finnes ulike metoder for å trene disse modellene. I [7] benyttes en inkrementell versjon av Expectation Maximization (EM) algoritmen. I [33] poengteres det at en eksakt EM-implementasjon over et vindu av nylig observerte data er uforholdsmessig kostbart. Vi henviser derfor til [7] for en nærmere beskrivelse av EM. Istedenfor antyder [33] en versjon av k-means algoritmen, der en piksel klassifiseres til å tilhøre den miksturkomponenten den ligger nærmest. Denne metoden er beskrevet i algoritme 5.1.

Algoritme 5.1 Pikselklassifisering gjennom k-means

1. Hver nye innkommende pikselverdi, X_t, sjekkes mot hver av de K komponentene i miksturmodellen. Sjekkingen gjøres i henhold til ligning 5.6 med $\lambda = 2.5$. Hvis X_t matcher den $i$te komponenten av miksturmodellen (X_t innenfor λ standardavvik fra $\mu_{i,t}$), oppdateres komponentens parametre gjennom ligning 5.8–5.10.

$$\mu_{i,t} = (1-\rho)\mu_{i,t-1} + \rho X_t \tag{5.8}$$

$$\sigma^2_{i,t} = (1-\rho)\sigma^2_{i,t-1} + \rho(X_t - \mu_{i,t})^T(X_t - \mu_{i,t}) \tag{5.9}$$

$$\omega_{i,t} = (1-\alpha)\omega_{k,t-1} + \alpha \tag{5.10}$$

   der $\rho = \alpha\eta(X_t|\mu_{i,t-1}, \Sigma i, t-1)$ og α representerer læringskoeffisienten.
2. Komponenter av miksturmodellen som ikke matcher X_t, oppdateres gjennom ligning 5.11–5.13.

$$\mu_{i,t} = \mu_{i,t-1} \tag{5.11}$$

$$\sigma^2_{i,t} = \sigma^2_{i,t-1} \tag{5.12}$$

$$\omega_{i,t} = (1-\alpha)\omega_{k,t-1} \tag{5.13}$$

3. Dersom ingen av de K modellene matcher X_t, forkastes den minst sannsynlige modellen, og erstattes av en ny modell med parametrene $\mu = X_t$, initielt høy σ^2 og lav ω.

Etterhvert som miksturmodellene for hver pikselllokasjon endres, er vi interessert i å finne ut hvilke komponenter av modellen som representerer bakgrunnen. Dersom en innkommende piksel, $X_t(x, y)$, representerer en bakgrunnspiksel, vil den øke vekten, ω, for bakgrunnskomponenten den klassifiseres under. Dersom en piksel representerer et forgrunnsobjekt, vil dette enten resultere i dannelse av en ny komponent med lav ω, eller økning i varians for en eksisterende komponent. Det er derfor naturlig å velge de komponentene med høyest ω og lavest σ, som bakgrunn [33]. Komponentene sorteres derfor etter verdien på ω/σ. De B første komponentene velges så til å representere bakgrunnen. I [7] benyttes en enkel heuristisk merking av miksturkomponentene, for å avgjøre hvilken komponent som beskriver hvilken type objekt. Den mørkeste miksturkomponenten (lavest μ_i) merkes som skygge. Av de to resterende komponentene, merkes den med størst varians som forgrunnskomponent, og den siste som veibane.

5.2.3 Optisk flyt

Optisk flyt (eng.: *optical flow*) er en betegnelse på den visuelle bevegelsen man registrerer når en linse beveger seg relativt til et objekt eller omvendt. For eksempel vil verden sett fra et bilvindu, bevege seg bakover etterhvert som bilen beveger seg fremover. Objekter nærmere linsen beveger seg raskere i linsen enn objekter lengre borte, selv om linsens hastighet er konstant [4]. Optisk flyt kan derfor sies å være den optiske komponenten av bevegelse.

I denne rapporten benytter vi begrepet optisk flyt som en samlebetegnelse på en rekke algoritmer som estimerer tredimensjonal bevegelse ut ifra en projeksjon på en todimensjonal bildesekvens [1]. Altså reflekterer optisk flyt endringer i en kontinuerlig bildesekvens, $I(x, y, t)$, i løpet av et tidsintervall, dt.

Bevegelsen estimeres ved å kalkulere et optisk flytfelt (eng. *flow field*) ut ifra bilder i sekvensen. Flytfeltet er et sett med hastighetsvektorer som representerer tredimensjonal objektbevegelse. Flytfeltet kan altså uttrykke bevegelse slik som translasjon langs, og rotasjon rundt, vilkårlig(e) akse(r). Optiske flytfelt er ofte passende bevegelsesdeskriptorer, og kan egne seg til å rekonstruere tredimensjonal bevegelse ut ifra en projeksjon på en bildesekvens. Det er essensielt at dette feltet kun representerer bevegelsesrelaterte intensistetsendringer, i og med at andre endringer identifiseres som støy [32]. Den åpenbare anvendelsen av optisk flyt i segmenteringsøyemed, er å etablere bakgrunnsmodellen som de delene av flytfeltet med korte hastighetsvektorer. For segmenteringsoppgaver poengterer [45] at disse metodene ofte gir for høy beregningsmessig kompleksitet til sanntidsberegninger. Et annet problem er at det er vanskelig å bestemme bevegelse for store uniforme flater, uten å måtte se på store deler av bildet [29]. Vi vil derfor kun gi en kort beskrivelse av algoritmer for optisk flyt.

Optisk flyt baserer seg på to antakelser [32] :

1. Den observerte intensiteten i et objektpunkt er konstant over tid, og
2. nærliggende piksler i bildet beveger seg på samme måte.

Altså er det slik at nabolaget til en piksel, (x, y), har forflyttet seg en liten distanse, (dx, dy), iløpet av tidsintervallet, dt. Med bakgrunn i disse antakelsene er det dermed mulig å bestemme dx, dy og dt som tilfredsstiller:

$$I(x + dx, y + dy, t + dt) = I(x, y, t) \tag{5.14}$$

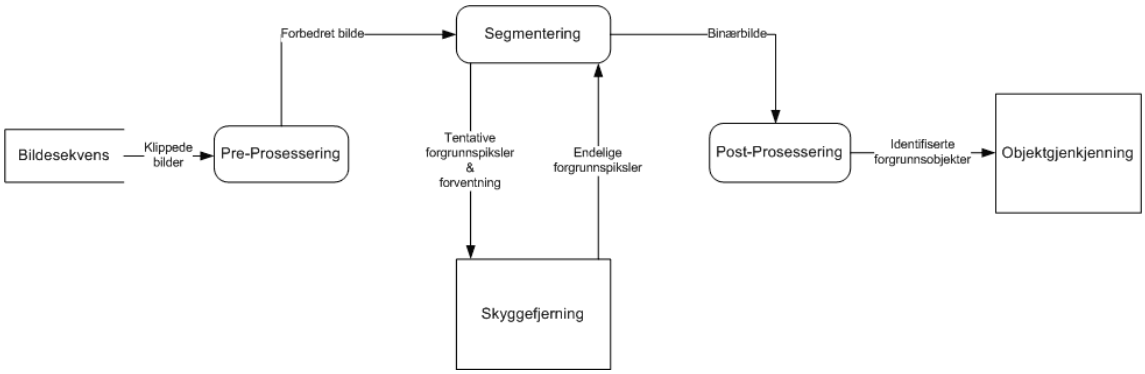
Målet videre er å beregne de to komponentene av hastighetsvektoren, $\mathbf{v} = (v_x, v_y)$. Som kjent er hastighet gitt som den tidsderiverte av posisjon. Altså er vi ute etter

$$\mathbf{v} = (v_x, v_y) = (\frac{dx}{dt}, \frac{dy}{dt}) \tag{5.15}$$

Siden vi ikke har benyttet optisk flyt i X-Analyzer, vil vi ikke gi en mer detaljert utledning her. For mer om hvordan man beregner $\mathbf{v}$ for hver piksel, og detaljerte beskrivelser av konkrete algoritmer, henviser vi til [1, 32].

5.3 Valgt metode

Dette delkapittelet beskriver segmenteringsmodulen i X-Analyzer. Figur 5.5 viser dataflyten gjennom denne modulen som dikterer strukturen i dette delkapittelet.



Figur 5.5: Dataflyt i segmenteringsmodulen

Først diskuteres nødvendigheten av preprosessering av bildene i sekvensen. Deretter følger en detaljert beskrivelse av selve segmenteringsalgoritmen. Til slutt beskrives noen filtre vi anvender på resultatet, for å eliminere støy og fjerne falske positiver. Skyggefjerning er implementert som en delprosess i segmenteringen, men er beskrevet i kapittel 6.

5.3.1 Preprosessering

Forut for segmenteringen ville vi prøve å eliminere tilfeldig støy i bildesekvensen. Støy i denne sammenhengen er individuelle piksler i bakgrunnsmodellen som brått endrer intensitet, slik at de feilaktig blir klassifisert som forgrunnsobjekter. Figur 5.6 viser det opprinnelige bildet og segmenteringen av dette bildet, uten forutgående støyreduksjon. Legg merke til de individuelle hvite pikslene i bildet til høyre som feilaktig blir segmentert som forgrunn.

Figur 5.6: Ingen filtrering

For å jevne ut gråtoner og dermed fjerne årsaken til denne typen støy, eksperimenterte vi med *medianfiltrering* og *boksfiltrering*.

Medianfilter

Responsen i piksel, (x, y), ved medianfiltrering av et bilde, $I(x, y)$, er gitt som medianen i $n \times m$ nabolaget til (x, y). Dette er illustrert i figur 5.7(a) der $n = m = 3$. Medianen i dette tilfellet er 111.

(a) Medianfilter (b) Medianfiltrert

Figur 5.7: Medianfilter og resultat

Figur 5.7(b) viser samme situasjon som figur 5.6, men her har det opprinnelige bildet blitt medianfiltrert for å eliminere støy i segmenteringen. Støyreduksjonen fremkommer tydelig til høyre.

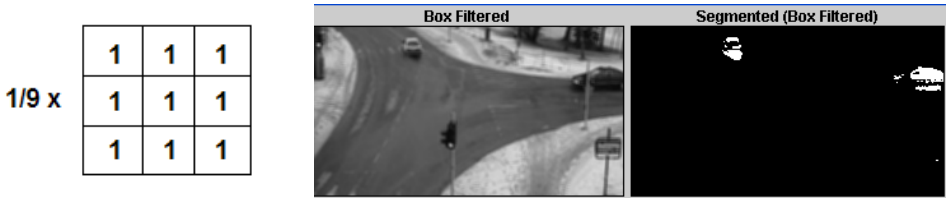
Boksfilter

Vi prøvde også å redusere støyen med et boksfilter. Et boksfilter er et lavpassfilter, der hvert element i filtermasken har samme verdi. Responsen til et boksfilter er gjennomsnittet i et $n \times m$ nabolag rundt hver piksel i bildet. I praksis flyttes en filtermaske, $H(s, t)$, med størrelse $n \times m$, over hvert punkt i bildet, $I(x, y)$. For hver piksellokasjon, (x, y), beregnes responsen av filteret som summen av produktene av filterkoeffisientene og de underliggende pikselverdiene. Denne operasjonen er i [8] gitt som

$$G(x, y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} H(s, t) I(x + s, y + t) \tag{5.16}$$

der $a = (m - 1)/2$ og $b = (n - 1)/3$. For å få gjennomsnittet, divideres til slutt resultatet av filtreringen på summen av koeffisientene i filteret. Vi benyttet et boksfilter med størrelse, $m = n = 3$, illustrert i figur 5.8(a).

Resultatet av boksfiltrering og segmentering av den samme scenen som figur 5.6 og figur 5.7(b), er gitt i figur 5.8(b). Støyen er her i stor grad eliminert, og kun de interessante forgrunnsobjektene gjenstår.



(a) Boksfilter (b) Boksfiltrert

Figur 5.8: Boksfilter og resultat

Oppsummering

Vi mener å kunne registrere minst støy gjennom preprosessering ved bruk av boksfilter fremfor medianfiltrering. På den andre siden viser det seg at initiell filtrering er med på å ødelegge konnektiviteten i de segmenterte objektene. Dette er noe synlig i de segmenterte objektene i figur 5.7(b) og 5.8(b). Støyfjerning kan også gjøres i etterkant av segmenteringen. Siden vi anser konnektivitet i de segmenterte objektene som viktigere enn støyfjerning på dette tidspunktet i prosessen, har vi valgt å avstå fra støyfjerning i preprosesseringsfasen.

5.3.2 Segmentering

Vi har implementert en statistisk adaptiv bakgrunnsfjerningsalgoritme. Adaptive metoder for bakgrunnsfjerning er mer robuste enn naiv bakgrunnssubtraksjon, men de er også beregningsmessig mer komplekse, siden bakgrunnsmodellen må oppdateres for hver iterasjon.

Implementasjonen tar utgangspunkt i Pfinder [42] og Stauffer & Grimsons algoritme, beskrevet i [33]. Stauffer & Grimson modellerer hver piksellokasjon som en miksturmodell bestående av K normalfordelinger. De K normalfordelingene beskriver hver en bakgrunnsmodalitet. For mer om bakgrunnsmodaliteter se avsnitt 5.2.2. Hver komponent av miksturmodellen er beskrevet ved en forventningsverdi, μ, og varians, σ^2, som definerer en normalfordeling. I [33] varieres K i intervallet, $K \in [3, 5]$. Vi eksperimenterte med to implementasjoner: En implementasjon der vi varierte antall komponenter i miksturmodellen, og en implementasjon med kun én enkel normalfordeling. I vårt tilfelle ga implementasjon med kun én normalfordeling både best resultater og ytelse. Den multinormale implementasjonen gav ikke like gode resultater og var uforholdsmessig beregningskrevende. Siden en rekke forfattere rapporterer om gode resultater ved bruk av miksturmodeller [7, 33, 14, 27], utelukker vi ikke at våre dårlige resultater kan skyldes faktorer vi har oversett, eller feil i implementasjonen. Den videre diskusjonen i dette avsnittet er knyttet til den best fungerende implementasjonen med én normalfordeling.

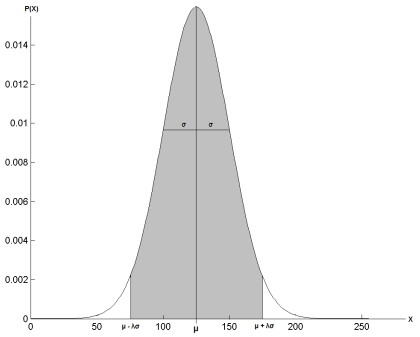
Siden vi opererer med kun én normalfordeling, reduseres algoritmen fra [33] til noe lignende som det [42] beskriver. Algoritmen opererer på pikselnivå i to steg:

1. Klassifisering av piksel (bakgrunn/forgrunn).
2. Oppdatering av μ og σ^2.

For å avgjøre om en piksel i et innkommende bilde tilhører forgrunnen eller bakgrunnen, sjekkes bakgrunnsmodellen. Vi har valgt samme klassifiseringsmetode som i [33]. Den innkommende

pikselverdien, x_{t+1}, klassifiseres som *bakgrunn* dersom den tilfredsstiller ligning 5.17.

$$\mu_t - \lambda\sigma_t < x_{t+1} < \mu_t + \lambda\sigma_t \tag{5.17}$$



Figur 5.9: Normalfordeling ($\mu = 125$ $\sigma = 25$).

Klassifiseringen baserer seg på antakelsen om at forgrunnsobjekter består av piksler med signifikant avvik fra bakgrunnsmodellen. Piksler med høyt standardavvik faller utenfor dette intervallet og blir segmentert som forgrunn. Piksler med lavt standardavvik stabiliseres som bakgrunnspiksler. Signifikans i avviket bestemmes av λ. λ settes normalt i intervallet $\lambda \in [1.0, 3.0]$. Høyere λ gir en mer restriktiv segmentering, siden dette medfører at flere piksler klassifiseres som bakgrunn. Figur 5.9 illustrerer dette. Langs x-aksen ligger pikselverdier mellom 0 og 255. $P(X = x)$ er sannsynligheten for å observere pikselverdi x.

Dersom fargebilder (RGB) benyttes, modelleres hver av fargekomponentene som uavhengige normalfordelinger. For å klassifisere en fargepiksel som bakgrunn, krever vi at minst to av de tre fargekomponentene tilfredsstiller ligning 5.17.

Vi har benyttet *selektiv oppdatering* av parametrene i normalfordelingen (se avsnitt 5.2.1). Dersom pikselen x_{t+1} blir klassifisert som bakgrunn, oppdateres μ og σ^2 for hver fargekomponent gjennom ligning 5.18 og ligning 5.19.

$$\mu_{t+1} = \alpha\mu_t + (1-\alpha)x_{t+1} \tag{5.18}$$

$$\sigma^2_{t+1} = \alpha\sigma^2_t + (1-\alpha)(x_{t+1} - \mu_{t+1})^2 \tag{5.19}$$

der α er en parameter til segmenteringsmodulen. På denne måten vil innkommende bakgrunnspiksler bidra med en vekt lik $(1-\alpha)$ til modellen. Dersom σ^2 etter oppdatering faller under en verdi σ^2_{min} settes $\sigma^2 = \sigma^2_{min}$. Hvis pikselen klassifiseres som forgrunn skjer ingen oppdatering.

Ved oppstart settes μ_0 til første innkommende pikselverdi x_0. σ^2 settes initielt høyt til $\sigma^2 = \sigma^2_{init}$. Algoritmen krever N initialiseringsbilder før den produserer en fornuftig segmentering. Siden det i initialiseringsfasen ikke finnes en modell å klassifisere ut i fra, oppdateres μ_{t+1} og σ^2_{t+1} blindt. Algoritmen er derfor sensitiv mot forgrunnsobjekter i denne perioden. I initialiseringsfasen beregnes gjennomsnittet ved hvert steg, t, gjennom ligning 5.20. σ^2 beregnes som ved normal prosessering (ligning 5.19).

$$\mu_{t+1} = \mu_t + \frac{x_{t+1} - \mu_t}{t+1} \tag{5.20}$$

Noen ganger blir stillestående objekter inkludert i bakgrunnsmodellen. Når slike objekter igjen beveger seg, avviker pikslene i denne regionen signifikant fra bakgrunnsmodellene. Denne regionen kan da bli feilaktig segmentert som forgrunn. Dette kalles gjenferd (eng.: *ghosts*). Modellen er spesielt sensitiv mot gjenferd i initialiseringsfasen. På den andre siden er det problematisk at det iblant tar for lang tid for nye statiske objekter å bli inkludert i bakgrunnen. For å unngå gjenferd, og samtidig inkludere nye statiske objekter, benyttes en korreksjon som ligner den beskrevet i [11]. Dersom en piksellokasjon, (x, y), har blitt identifisert som forgrunn i mer enn F_{max} etterfølgende bilder, tilbakestilles variansen for denne pikselen til $\sigma^2(x, y) = \sigma^2_{init}$. Dette øker akseptanseintervallet for bakgrunnsklassifisering (ligning 5.17). Derfor klassifiseres neste piksel i (x, y) som bakgrunn. Vi tilbakestiller ikke $\mu(x, y)$ siden den innkommende pikselen i (x, y) kan være forgrunn. Tilbakestilling av $\mu(x, y)$ til verdien i (x, y) vil isåfall forurense modellen uforholdsmessig mye.

5.3.3 Postprosessering

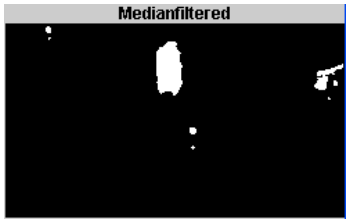
Når segmenteringen er fullført, er det essensielt at det resulterende binærbildet postprosesseres. Da kan forgrunnsobjektene forbedres, og støy i form av falske positiver fjernes. Det påpekes i [27] at siden en statistisk signifikant del av inputpikslene vil ligge i halene av sannsynlighetsfordelingen (figur 5.9), vil resultatet av segmenteringen inneholde mange små regioner som ikke representerer de interessante forgrunnsobjektene. Figur 5.10 viser resultatet av segmenteringsoperasjonen uten postprosessering. Noen av de hvite regionene i bildet til høyre representerer støy, og er uinteressante forgrunnsobjekter. Legg blant annet merke til at begge lyssignalene i krysset har blitt segmentert som forgrunn. Dette er fordi intensitetsendringen i disse pikslene ved lysskifte, feilaktig har blitt gjenkjent som objektbevegelse. På den andre siden vil en innsnevring av akseptanseintervallet (mindre λ i ligning 5.17), gi en for restriktiv segmentering. Dette får konsekvenser for konnektiviteten i forgrunnsobjektene og representerer et større problem.

Figur 5.10: Segmentering uten postprosessering

Vi har implementert en postprosesseringssekvens bestående av:

1. Medianfiltrering.
2. Morfologisk lukking.
3. Regionmerking (eng: *connected component labeling*).
4. Regionterskling (eng: *area thresholding*).

Medianfiltrering



Figur 5.11: Medianfiltrert

Medianfiltrering er beskrevet detaljert i avsnitt 5.3.1. Vi vil derfor kun illustrere resultatet av medianfiltrering av segmenteringen fra figur 5.10. Dette er vist i figur 5.11 der bildet har blitt prosessert med et 3×3 medianfilter.

Medianfiltreringen har eliminert mange av de små regionene, og de faktiske forgrunnsobjektene har blitt glattet. Fremdeles henger feilsegmenteringen av lyssignalene igjen. Disse regionene er for store til å bli eliminert av et 3×3 filter.

Morfologisk lukking

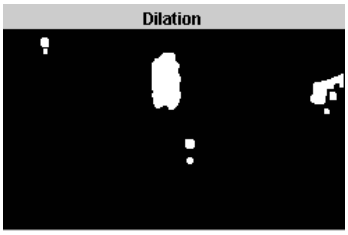
Matematisk morfologi benyttes i bildebehandling for å trekke ut bildekomponenter som er nyttige i representasjon og beskrivelse [8]. Morfologiske operatorer uttrykkes i sett-teori. Et sett punkter kan representere et objekt i bildet. Matematisk morfologi er et omfattende fagfelt. Vi har kun benyttet en enkel kombinasjon av to elementære morfologiske operatorer, nemlig dilasjon og erosjon. Disse involverer et binærbilde, A, og et strukturelement, B. Strukturelementet føres over hvert punkt i A, hvor den morfologiske operasjonen utføres. Vi har benyttet en enkel 3×3 maske med 1'ere som strukturelement.

Dilasjon Dilasjonen av bildet, A, med et strukturelement, B, er i [8] definert som:

$$A \oplus B = \{z|(\hat{B})_z \cap A \neq \varnothing\} \tag{5.21}$$

der $\hat{B}$ uttrykker *refleksjonen* av settet B, altså $\hat{B} = \{w|w = -b,\ b \in B\}$, og $(B)_z$ uttrykker *translasjon* av B med punktet, z, altså $(B)_z = \{c|c = b + z,\ b \in B\}$. Dersom B er symmetrisk om sitt origo, er det i praksis likegyldig om man benytter B eller $\hat{B}$ til dilasjonen.

Ligning 5.21 uttrykker settet av alle punkter, z, slik at $\hat{B}$ og A overlapper med *minst ett* element. Dilasjon er derfor nyttig når små diskontinuiteter i objekter skal tettes. Dilasjonen av bildet fra figur 5.11 er vist i figur 5.12. De segmenterte regionene har blitt ytterligere glattet. Videre har konnektiviteten i pikslene til bilen, som kommer inn fra høyre, blitt forbedret. Feilsegmenteringen som følge av lyssignalet har blitt forsterket.



Figur 5.12: Dilasjon

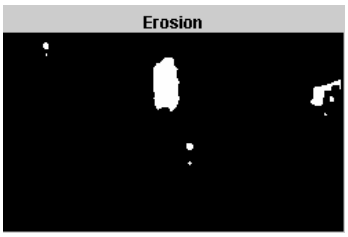
Erosjon Videre er erosjonen av et bilde, A, med et strukturelement, B, definert som

$$A \ominus B = \{z|(B)_z \subseteq A\} \tag{5.22}$$

Erosjonen av A med B, er settet av punkter z slik at B translert med z, er fullstendig inneholdt i A. Erosjon vil altså eliminere små irrelevante regioner i bildet.

Lukking Morfologisk lukking av A med B, er definert som dilasjonen av A med B, etterfulgt av erosjon av resultatet med B:

$$A \bullet B = (A \oplus B) \ominus B \tag{5.23}$$



Figur 5.13: Morfologisk lukking

Figur 5.13 viser erosjonen av bildet fra figur 5.12, og dermed den morfologiske lukkingen av det medianfiltrerte bildet fra figur 5.11. Hensikten med lukkeoperasjonen er å glatte konturer, smelte sammen små diskontinuiteter og tette hull [8]. Generelt eksisterer det ingen optimal kombinasjon av morfologiske operatorer. I W^4 beskrevet i [10] og [21], beskrives andre sekvenser med morfologisk postprosessering av segmenteringsresultater. Lukking er altså bare en av mange måter å forbedre resultatet på. I vårt tilfelle mener vi dette ga gode resultater.

Åpning Vi eksperimenterte også med morfologisk åpning av segmenteringen fra figur 5.11. Morfologisk åpning av et bilde, A, med et strukturelement, B, er definert i ligning 5.24 som det omvendte av lukking [8].

$$A \circ B = (A \ominus B) \oplus B \tag{5.24}$$

Åpning er med på å eliminere støy, bryte tynne grener og glatte konturen. Dette er fordelaktig, men i vårt tilfelle var åpning med på å bryte konnektiviteten. Resultatet av åpning er illustrert i figur 5.14. Den ene bilen er segmentert som to objekter og bilen som kommer inn fra høyre er forsvunnet helt. Dersom konnektiviteten i objektene ødelegges, skaper dette problemer for følgingsmodulen beskrevet i kapittel 8.

Figur 5.14: Morfologisk åpning

Regionmerking

Hensikten med regionmerking er å identifisere sammenhengende regioner i settet av forgrunnspiksler. Regionmerking endrer i så måte ikke resultatet, men hjelper oss å isolere hvert enkelt forgrunnsobjekt. Hver region representeres ved et objekt bestående av et sett koordinater og en merkelapp (eng.: *label*). Vår implementasjon er hentet fra [32, Kap. 6.1] og er beskrevet i algoritme 5.2. Regioner identifiseres i fire-konnektivitet, altså vurderes sammenheng mellom piksler

kun nord, øst, sør og vest for en piksel. Diagonal konnektivitet (åtte-konnektivitet) vurderes ikke.

Algoritme 5.2 Regionmerking i fire-konnektivitet

1. Scan bildet, R, rad for rad fra venstre til høyre. For hver piksel, $R(i,j) \neq 0$, tilordnes en verdi, $v \neq 0$. v bestemmes ut ifra merkingen til vestlig, $R(i-1,j)$, og nordlig, $R(i,j-1)$, nabo på følgende måte:
   - Hvis vestlig og nordlig nabo er umerket (bakgrunn), tilordnes $R(i,j)$ en ny og hittil ubrukt merkelapp.
   - Hvis kun én av naboene i vestlig eller nordlig retning er merket, tilordnes $R(i,j)$ denne merkelappen.
   - Hvis både nordlig og vestlig nabo er merket, tilordnes $R(i,j)$ en av disse merkelappene. Dersom nordlig nabos merking er ulik vestlig nabos merking, lagres disse merkelappene som ekvivalente i en ekvivalenstabell.
2. Alle regionpiksler ble identifisert i første gjennomløp, men noen regioner har piksler med ulik (ekvivalent) merking. Hele bildet scannes derfor på nytt, og piksler hvis merkelapp tilhører en ekvivalensklasse merkes på nytt med den laveste merkelappen fra ekvivalensklassen.

Algoritmen er optimalisert slik at den unngår et andre gjennomløp av hele bildet for å løse merkingskonflikter. Istedenfor å iterere gjennom bildet, itererer algoritmen gjennom ekvivalenstabellen. For hver ekvivalensklasse returneres regionene som er registrert i denne ekvivalensklassen. Siden regioner er lagret som en liste med punkter, går sammenslåingen ut på å slå sammen listene med punkter fra de ulike regionene knyttet til hver ekvivalensklasse. Deretter velges den minste merkelappen som merkelapp for den fullstendige regionen.

Regionterskling

Når alle regioner er isolert, lukes de minste regionene vekk gjennom terskling på regionstørrelse. Terskelverdien settes ut ifra hvor store regioner vi ønsker å identifisere som forgrunn øverst i bildet (dvs. lengst bak i scenen). Siden vi i de fleste tilfeller ikke er i stand til å filme rett ovenfra, vil et objekt i bunnen av bildet, ved Y_{MAX}, fremstå større enn det samme objektet i toppen av bildet ($y = 0$). Vi eksperimenterte derfor med en metode der posisjonen til regionen ble tatt hensyn til ved regiontersklingen. Vi prøvde å finne en dynamisk terskelverdi, $T(y)$, som aksepterte mindre regioner som forgrunnsobjekter desto høyere opp i bildet de forekom. For hver region beregnet vi den minste y-koordinaten. En region R med minste y-koordinat, y, ble akseptert som forgrunnsobjekt dersom $size(R) > T(y)$. Funksjonen $T(y)$ definerte vi som ved ligning 5.25.

$$T(y) = T_0(1 + \beta\frac{y}{Y_{MAX}}) \tag{5.25}$$

T_0 forteller hvor små regioner vi aksepterer som forgrunnsobjekt i toppen av bildet ($y = 0$). β er en normaliseringsfaktor som sier noe om hvordan kameraet er plassert i forhold til scenen som filmes. Større β gir et større akseptansekrav desto lenger ned i bildet (nærmere kamera) regionen befinner seg.

Den dynamiske terskelverdien fra ligning 5.25 fungerte dessverre ikke slik vi ønsket. Det ble et problem å finne passende verdier for β slik at arealet på regionene ble normalisert. Vi endte opp med å miste regioner som skulle være identifisert som forgrunnsobjekter nederst i bildet. Vi implementere derfor kun en enkel statisk terskel der $T(y) = T_0$ og $T_0 \in [50, 150]$ avhengig av scenen. Regionene vi sitter igjen med etter denne tersklingsoperasjonen, identifiseres som faktiske forgrunnsobjekter.

5.4 Resultater

Dette delkapittelet dokumenterer resultatene fra implementasjonen beskrevet i forrige delkapittel. Vi presenterer resultater ved ulikt valg av parametre, og hva vi fant å være den optimale parameterkombinasjonen. Parameterkombinasjonen er optimal i subjektiv forstand: Den definerer hva vi, ut ifra visuell inspeksjon, mener gir best segmentering. Videre diskuteres problemer og feilkilder knyttet til segmentering av vanskelige scener.

5.4.1 Statiske parametre

Dette avsnittet diskuterer resultater ved ulikt valg av statiske parametre for segmenteringsmodulen. Med statiske parametre forstås parametre som initialiseres til en konstant verdi ved oppstart.

λ

λ er avvikskoeffisienten fra ligning 5.17 (antall standardavvik fra forventningsverdien bakgrunnspiksler ligger innenfor). Siden sannsynlighetsfordelingen modellerer bakgrunn, vil for lav λ føre til at for få piksler klassifiseres som bakgrunn. Dette betyr at mye støy vil klassifiseres som forgrunn. Dette er illustrert i figur 5.15 der $\lambda = 1.0$.

Figur 5.15: For lav λ

På den andre siden vil for høy λ medføre en for restriktiv segmentering, der for mange piksler blir klassifisert som bakgrunn. Figur 5.16 viser segmentering av det samme bildet som i figur 5.15, men med $\lambda = 4.0$. De tre forgrunnsobjektene har ikke blitt segmentert riktig. Høy λ er også med på å ødelegge konnektiviteten i objekter.

Eksperimentering med ulike λ-verdier viste at $\lambda = 2.5$ fungerte best. Resultat fra segmentering med denne verdien er vist i figur 5.17. De tre forgrunnsobjektene er her korrekt segmentert uten diskontinuiteter.

Figur 5.16: For høy λ

Figur 5.17: Optimal λ

α

α er læringskoeffisienten fra ligningene 5.18–5.19 (hvor mye modellen skal vektlegges i forhold til nye bilder, dvs. adaptiviteten til metoden).

For lav α vil tillegge nye bilder for mye vekt. Dette fører til at objekter som stopper midlertidig, inkluderes i bakgrunnsmodellen. Når disse objektene igjen beveger seg oppstår gjenferd på stedet objektet stod. Figur 5.18 viser segmentering med $\alpha = 0.80$. Bilen som i disse bildene svinger ned til venstre i krysset har forut for dette stått noen sekunder og ventet på grønt lys oppe til høyre. Regionen der bilen har stått har blitt segmentert som forgrunn, pga. for lav α. Enda lavere α resulterer i at kunstige haler dannes i bak-kant av de segmenterte objektene [30].

Figur 5.18: For lav α

Figur 5.19 illustrerer hva som skjer dersom α settes for høyt ($\alpha = 0.99$). Segmenteringen av bilen som svinger ned til venstre er blitt delt. Vi mister altså konnektiviteten i forgrunnsobjektene ved for høy α. Dette er fordi modellen blir ufølsom for endringer.

Vi fant at $\alpha = 0.95$ fungerte optimalt. Ved denne verdien fikk vi en god avveining mellom adaptivitet og konnektivitet. Et resultat fra segmentering med denne verdien er vist i figur 5.20.

Figur 5.19: For høy α

Figur 5.20: Optimal α

Andre parametre

N representerer antall initialiseringsbilder pikselfordelingene trenger før de produserer pålitelige resultater. For lav N gjør at pikselfordelingene initialiseres for dårlig. Settes N for høyt forurenses fordelingene, siden blind oppdatering benyttes i initialiseringsfasen. $N = 10$ fungerte godt.

T_0 er regionterskelen som bestemmer hvor stor en region med sammenhengende piksler må være for å identifiseres som forgrunnsobjekt. For lav T_0 gjør at små regioner som egentlig representerer støy blir identifisert som forgrunnsobjekter. En for høy verdi gjør at objekter forsvinner fra scenen etterhvert som de blir mindre. $T_0 = 100$ fungerte godt.

F_{max} representerer maksimalt antall etterfølgende ganger en piksellokasjon kan identifiseres som forgrunn, før den klassifiseres som bakgrunn og variansen tilbakestilles til σ^2_{init}. For lav F_{max} gjør at piksler i forgrunnsobjekter som bare stopper midlertidig blir tilbakestilt. For høy F_{max} resulterer i at gjenferd ikke blir fjernet, samtidig som nye statiske objekter ikke blir inkludert i bakgrunnen. F_{max} avhenger av bilderaten. Med 5 bilder pr. sekund ga $F_{max} = 100$ gode resultater.

σ^2_{min} representerer minste tillatte varians. Dersom variansen σ^2 til en piksel synker under σ^2_{min}, settes $\sigma^2 = \sigma^2_{min}$. Siden variansen er med å bestemme akseptanseintervallet i ligning 5.17, vil for lav σ^2_{min} medføre at for mye støy blir segmentert som forgrunn. For høy verdi gir en for restriktiv segmentering. $\sigma^2_{min} = 150$ gav gode resultater.

σ^2_{init} er initialiseringsverdi for σ^2. Denne settes initielt høyt til $\sigma^2_{init} = 900$. Konsekvensen av dette er restriktiv segmentering initielt. Vi ønsker dette fordi pikselfordelingene i begynnelsen er veldig sensitiv mot støy. Dersom mye støy blir identifisert som forgrunn i oppstarten, vil modellene få et galt utgangspunkt. Etterhvert som modellene lærer å skille forgrunn fra bakgrunn synker kravet til høy σ^2.

5.4.2 Feilkilder

Dette avsnittet diskuterer hva vi anser for å være de største feilkildene og problemene knyttet til segmenteringsmodulen. Skyggeproblematikk er i denne sammenhengen også vesentlig, men dette er behandlet eksplisitt i kapittel 6.

Konnektivitet

Figur 5.21: Konnektivitet

Brudd på konnektivitet er illustrert i figur 5.21. Her er konnektiviteten brutt i begge forgrunnsobjektene. Bakgrunnsobjekter som okkluderer forgrunnen er mye av årsaken til disse diskontinuitetene. I figuren kommer to lysstolper som egentlig er en del av bakgrunnen foran bilene. Diskontinuiteter i forgrunnsobjektene skaper problemer for senere klassifisering og følging.

Okklusjon

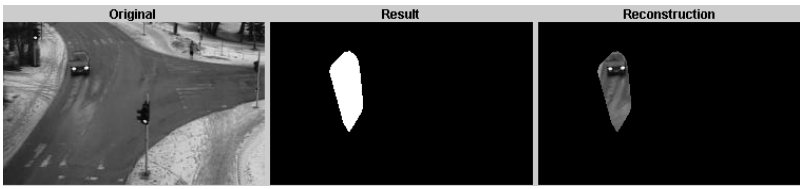
Delvis og fullstendig okkluderende objekter segmenteres konsekvent som ett sammenhengende objekt. Dette er illustrert i figur 5.22 der bussen okkluderer bilene i motgående kjørefelt. Årsaken til okklusjonen er vinkelen scenen er filmet i. Følgingsmodulen er til en viss grad robust mot dette problemet.

Figur 5.22: Overlappende objekter

Refleksjoner

Lys fra frontlykter som blir reflektert i bakgrunnen er, i likhet med skygge, med på å vanskeliggjøre definisjonen av en objektkontur. Dette er vist i figur 5.23. For å illustrere problemet har vi beregnet det konvekse hullet som omslutter de segmenterte regionene. Til høyre i figuren har vi også tatt med en rekonstruksjon av forgrunnen. Det er tydelig at lys fra bilens lykter utgjør en større del av det segmenterte forgrunnsobjektet enn selve bilen. Siden objektgjenkjenningsmodulen opererer med det konvekse hullet som representasjon av de segmenterte regionene,

vanskeliggjør dette klassifiseringsoppgaven. Samtidig forsterker det okklusjonsproblemene for følgingsmodulen.



Figur 5.23: Lysproblematikk

En mulig løsning på dette problemet er å benytte multimodale sannsynlighetsfordelinger slik som i [7]. Refleksjoner vil da kunne modelleres som en egen bakgrunnsmodalitet og fjernes automatisk.

Gjenferd

Problemet med gjenferd er vist i figur 5.24, der en bil har stått stille i bakgrunnsmodellens initialiseringsfase, for så å bevege seg. Dette problemet er størst i initaliseringsfasen siden modellen er spesielt sensitiv mot støy da.

Figur 5.24: Gjenferd

Som diskutert tidligere kan gjenferd elimineres ved å justere α på bekostning av konnektivitet. Vi benytter i tillegg parameteren F_{max} som en korreksjon på segmenteringen.

5.5 Konklusjon

Vi har implementert en statistisk, unimodal metode for bevegelsessegmentering. Metoden er basert på bakgrunnsmodellering av individuelle piksler gjennom normalfordelinger. Metoden er adaptiv i den forstand at bakgrunnsmodellen tilpasser seg endringer i scenen som følge av lysendringer og nye bakgrunnsobjekter. Dette er ikke tilfelle for de enkleste metodene for bevegelsessegmentering, som kun baserer seg på å subtrahere en tom bakgrunn fra hvert bilde i en videosekvens. Implementasjonen gir gode resultater for både farge- og gråtonebilder.

Vi implementerte også segmentering gjennom en multimodal miksturmodell, for å motvirke effekter som kommer av skygger og refleksjoner. Implementasjon av denne metoden gav ikke tilfredsstillende resultater og var uforholdsmessig beregningskrevende. Vi tror våre dårlige erfaringer med dette skyldes feil vi ikke fikk avdekket i vår egen implementasjonen, da flere av våre referanser dokumenterer gode resultater.

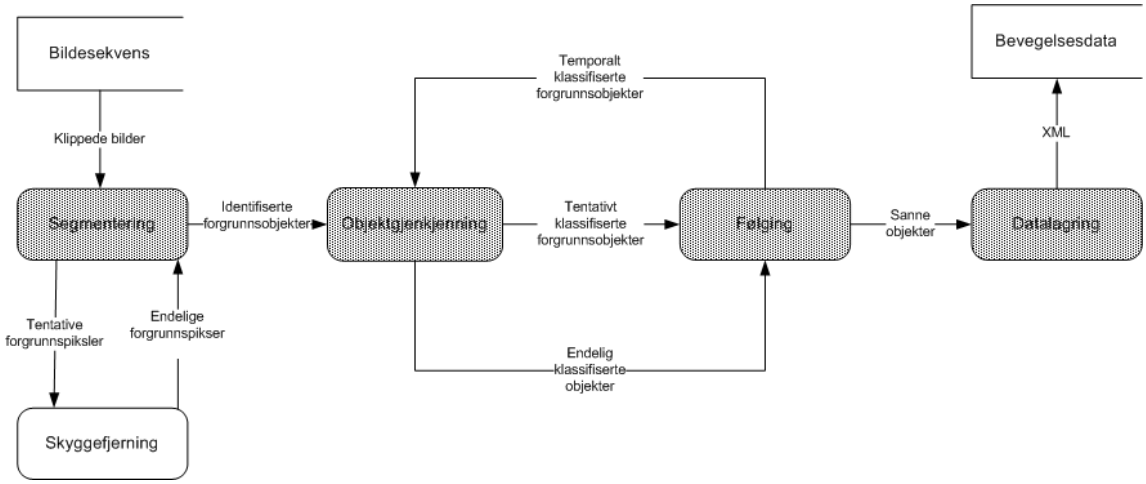
Vi eksperimenterte med preprosessering i form av median- og boksfiltrering for å fjerne støy. Filtreringen var til en viss grad vellykket. Siden preprosesseringen førte til lavere grad av konnektivitet, avstod vi fra dette. Støyfjerning implementerte vi istedenfor i et postprosesseringssteg bestående av medianfiltrering, morfologisk lukking og terskling på regionstørrelse. At regionterskelen settes statisk er en svakhet, siden objekter har ulik størrelse avhengig av posisjon i bildet.

Vurdering av segmenteringsmodulens resultater og parametervalg, er basert på visuell inspeksjon. Vi eksperimenterte med ulike parametre for modulen, og kom frem til en kombinasjon av disse som ga gode resultater for våre testsett. Suksesskriterier i denne sammenhengen har vært konnektivitet i objekter, lavt antall falske positiver og en nøyaktig definert objektkontur. Siden testsettene er filmet delvis fra siden, under lav sol og på fuktig asfalt, har vi problemer med henholdsvis okklusjon, skygge og refleksjoner fra frontlykter.

Kapittel 6

Skyggefjerning

Dette kapittelet diskuterer metoder for fjerning av skyggepiksler fra de segmenterte objektene. Figur 6.1 viser grensesnittet til skyggefjerningsmodulen i X-Analyzer.



Figur 6.1: Skyggefjerningsmodulens plassering

Skyggefjerningsmodulen opererer på pikselnivå i samarbeid med segmenteringsmodulen. De pikslene segmenteringsmodulen identifiserer som forgrunn, sendes til skyggefjerningsmodulen. Skyggefjerningsmodulen tar så den endelige avgjørelsen om en piksel representerer ekte forgrunn eller skygge.

6.1 Generelt

Bevegelsessegmentering i videosekvenser utnytter at bakgrunnen er tilnærmet stabil. Når forgrunnsobjekter beveger seg gjennom scenen, forandres pikselverdiene der de befinner seg. Skygger som forgrunnsobjektene kaster, skiller seg dessverre også signifikant fra bakgrunnen. Disse blir derfor segmentert sammen med de reelle forgrunnsobjektene. I noen situasjoner utgjør skygge

mer enn 50% av pikslene til et reelt forgrunnsobjekt. For videre prosessering av forgrunnsobjektene, vil en unøyaktig definert objektkontur skape problemer. For det første blir objektene vanskeligere å gjenkjenne og klassifisere. For det andre vil okklusjonsproblemer oppstå hyppigere i følgingsmodulen, siden skygger ofte kastes over på andre objekter.

6.2 Kjente metoder

I dette delkapittelet presenteres kjente metoder for skyggefjerning fra litteraturen.

6.2.1 Pikselfordelinger

Friedman & Russel rapporterer i [7] om en metode som benytter miksturmodeller bestående av flere normalfordelinger for å modellere bakgrunnen. Denne miksturmodellen har tre komponenter: en som modellerer bakgrunnspiksler, en som modellerer skyggepiksler og en som modellerer objektpiksler. Denne metoden har problemer med objekter med pikselverdier som ligger nært verdien til en skyggepiksel. Dette fører til diskontinuiteter i de segmenterte objektene. I segmenteringsmodulen eksperimenterte vi med miksturmodeller for bakgrunnsmodellering, men valgte å forkaste forsøket siden vi ikke oppnådde tilfredsstillende resultater i en prøveimplementasjon.

6.2.2 Aktive konturer

Vi trodde aktive konturer (eng. *"snakes"*), ville egne seg godt til skyggefjerning siden skyggen til et objekt som regel ikke danner en skarp kontur mot bakgrunnen. En aktiv kontur initialiseres rundt den segmenterte regionen, og er ment å krympe inn mot selve objektkonturen. Objektkonturen kan gjenkjennes som en kant i bildet med en signifikant intensitetsforskjell.

Ressem rapporterer i [29] om gode resultater ved bruk av aktive konturer. Vi investerte derfor mye tid i en prøveimplementasjon av dette. Arbeidet med denne implementasjonen er dokumentert i vedlegg A. Som det går frem av vedlegg A.4, ga prøveimplementasjonen dårlige resultater. I tillegg til dette identifiserte vi tre andre faktorer som gjorde at vi valgte å avstå fra bruk av aktive konturer til skyggefjerning i X-Analyzer:

- Metoden var beregningsmessig kompleks, og ville medført en betydelig ekstra beregningstid for hvert bilde. Våre tester ble kjørt på små bilder med ett objekt. Selv disse testene tok flere sekunder å prosessere.
- Initialisering av de aktive konturene var vanskelig å få til bra nok og var beregningsmessig tungt.
- Mye mer ressurser måtte brukes på å lage en velfungerende implementasjon.

6.2.3 Kromatografi

Porikli & Tuzel [26] rapporterer om gode resultater i skyggefjerning ved bruk av kromatografi, også kjent som fargeanalyse i HSI-rommet. HSI er en fargemodell som skiller seg fra den konvensjonelle RGB-modellen, ved at de tre komponentene er fargetone (Hue), fargemetning (Sat-

uration) og intensitet (Intensity). Prinsippet er enkelt: En skyggepiksel er en bakgrunnspiksel med lavere intensitet og fargemetning enn det en bakgrunnspiksel forventes å ha [26].

Gonzales & Woods [8] gir oss følgende likninger for konvertering fra RGB til HSI:

- Fargetonen er gitt ved

$$H = \begin{cases} \theta & \text{if } B \leq G \\ 360 - \theta & \text{if } B > G \end{cases} \tag{6.1}$$

der

$$\theta = \arccos\left\{ \frac{\frac{1}{2}\big((R-G)+(R-B)\big)}{\sqrt{(R-G)^2+(R-B)(G-B)}} \right\}$$

- Fargemetningen er gitt ved

$$S = 1 - \frac{3}{R+G+B} \cdot \min(R,G,B) \tag{6.2}$$

- Intensiteten er gitt ved

$$I = \frac{1}{3}(R+G+B) \tag{6.3}$$

6.3 Valgt Metode

Med bakgrunn i resultatene ved bruk av aktive konturer, valgte vi å implementere idéen beskrevet i delkapittel 6.2.3 istedenfor. Dette delkapittelet dokumenterer vår implementasjon av metoden.

Figur 6.2: Dataflyt i skyggefjerningsmodulen

Figur 6.2 viser dataflyten gjennom skyggefjerningmodulen. Skyggefjerningsmodulen er i praksis implementert som en integrert del av segmenteringsmodulen. Modulen mottar tentativt identifiserte forgrunnspiksler og forventningsverdiene til disse pikslene, fra segmenteringsmodulen. Etter konvertering fra RGB- til HSI-verdier, tas en avgjørelse om denne pikselen representerer en sann forgrunnspiksel eller en skyggepiksel. Resultatet returneres til segmenteringsmodulen som oppdaterer segmenteringen med bakgrunn i avgjørelsen.

6.3.1 Algoritme

Metoden for skyggefjerning ved hjelp av kromatografi presenteres i algoritme 6.1.

Algoritme 6.1 Shadow-Removal($foregroundPixels$)

```
1: for all pixel_i ∈ foregroundPixels do
2:   if INTENSITY(μ_i) > INTENSITY(pixel_i) then
3:     if SATURATION(μ_i) > SATURATION(pixel_i) then
4:       REMOVE(pixel_i, foregroundPixels)
5:     end if
6:   end if
7: end for
8: return foregroundPixels
```

Algoritmen tar inn et sett med piksler, $foregroundPixels$. Dette er pikslene segmenteringsmodulen har identifisert som forgrunn. μ_i representerer forventningsverdien for bakgrunnspiksel i. For hver piksel i $foregroundPixels$ sjekkes det om intensiteten (I) og fargemetningen (S) til μ_i er større enn intensiteten og fargemetningen til $pixel_i$. Dersom begge disse betingelsene er oppfylt, fjernes $pixel_i$ fra settet $foregroundPixels$. Til slutt returneres settet med ekte forgrunnspiksler. Algoritmen benytter tre funksjoner:

INTENSITY returnerer intensitetskomponenten (I) i henhold til ligning 6.3.

SATURATION returnerer fargemetningskomponenten (S) i henhold til ligning 6.2.

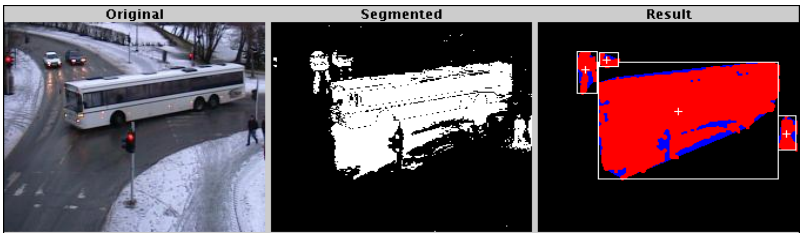
REMOVE Fjerner et element ($pixel_i$) fra et sett ($foregroundPixels$).

6.4 Resultater

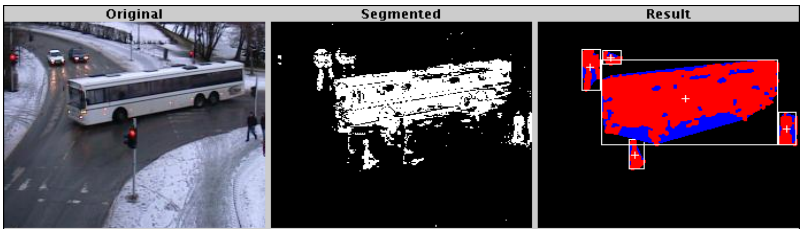
Figur 6.3 og figur 6.4 viser skyggefjerningsalgoritmen beskrevet i delkapittel 6.3, anvendt på to bilder fra testsettet Høgskoleringen.

Figur 6.3 viser en buss i krysset. Vi ser i figur 6.3(a) at en stor andel av de segmenterte pikslene egentlig tilhører bussens skygge. Algoritmen som beregner det konvekse hullet, legger til flere piksler i forgrunnsobjektet og forverrer situasjonen ytterligere. I figur 6.3(b) er det meste av skyggen fjernet fra regionen som beskriver selve segmenteringen (rødt). Det konvekse hullet (blått) fyller også nå inn piksler mellom ekstremalpunktene i den sammenhengende regionen, men situasjonen er allikevel bedre enn uten skyggefjerning.

Figur 6.4 viser en grønn bil som svinger til høyre i krysset. Skyggen kastet fra denne bilen gir unøyaktighet i segmenteringen (figur 6.4(a)). Figur 6.4(b) illustrerer skyggefjerningsalgoritmen anvendt på segmenteringen av det samme bildet. En god del skygge har blitt eliminert i dette tilfellet. Noe av skyggen som algoritmen har fjernet, fylles igjen av algoritmen som beregner det konvekse hullet.

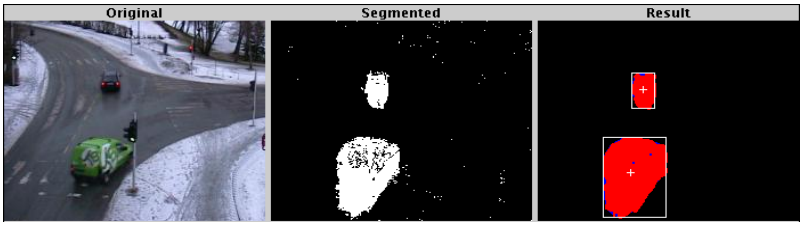


(a) En buss *uten* skyggefjerning

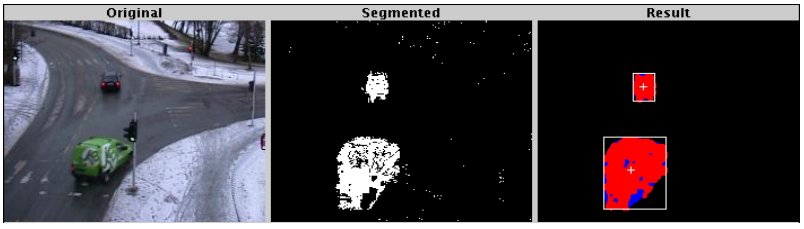


(b) En buss *med* skyggefjerning

Figur 6.3: Resultater fra skyggefjerning på en buss. I bildet lengst til høyre vises segmenteringen i rødt, det konvekse hullet rundt segmenteringen i blått, det omsluttende rektangelet i hvitt og objektets massesenter som et hvitt trådkors.



(a) En bil *uten* skyggefjerning



(b) En bil *med* skyggefjerning

Figur 6.4: Resultater fra skyggefjerning på en bil

6.4.1 Mulige Forbedringer

Porikli [26] benytter seg av nabolagsinformasjon for skyggefjerning. Alle pikslene besøkes to ganger. Den første iterasjonen er tilsvarende vår implementasjon. Andre iterasjon ser på sannsynligheten for at en piksel representerer skygge, ut i fra sannsynligheten for at naboene er skyggepiksler.

Fargetonen (H) kan også benyttes for å styrke skyggeestimatet. En skyggepiksel vil ha tilnærmet samme fargetone som forventningsverdien til en bakgrunnspiksel [26].

Videre kunne skyggefjerningsmodulen ha vært integrert tettere med algoritmen som beregner det konvekse hullet. På denne måten kunne man ha kuttet utstikkende deler fra en sammenhengende region. Dermed kan man unngå at det konvekse hullet motvirker skyggefjerningen.

6.5 Konklusjon

Skygge er et problem i denne typen applikasjoner, fordi skyggen segmenteres som en del av objektet den er kastet fra. I noen tilfeller består den segmenterte skyggen av et større antall piksler enn forgrunnsobjektet i seg selv. Dette gir en uriktig objektkontur som fører til problemer i forbindelse med klassifisering og følging.

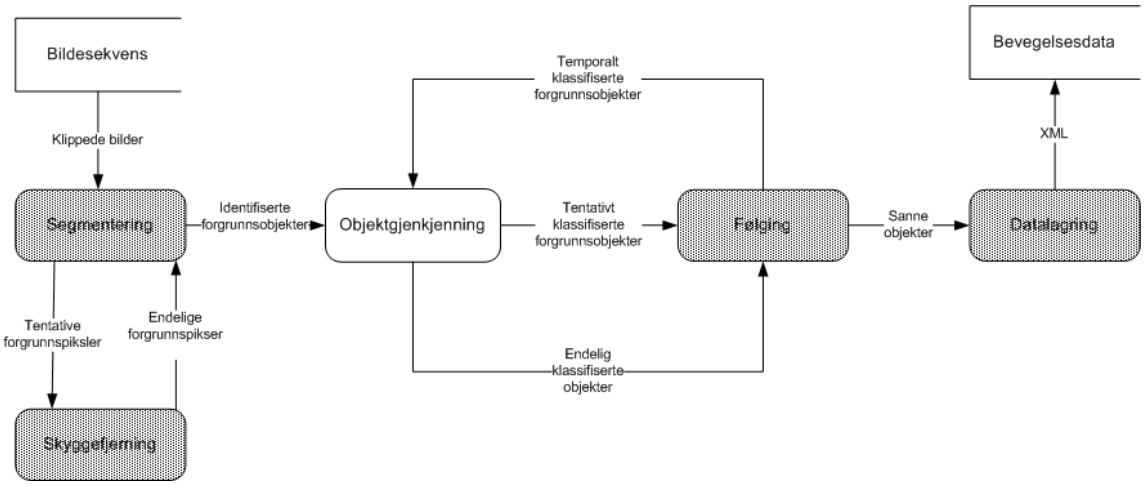
Vi har evaluert ulike metoder som kan bidra til enten å løse eller å motvirke dette problemet. Vi eksperimenterte først med en metode basert på aktive konturer. Aktive konturer initialiseres rundt den segmenterte regionen, og er ment å krympe inn mot selve objektkonturen. Objektkonturen kan gjenkjennes som en kant i bildet med en signifikant intensitetsforskjell. Vi fikk ikke de aktive konturene til å krympe inn mot den reelle objektkonturen på en tilfredsstillende måte og forkastet derfor dette forsøket.

Vi valgte istedenfor en metode basert på kromatografi. Prinsippet er å fjerne deler av forgrunnen med egenskaper tilsvarende skygge. En sammenligning i HSI-rommet av en forgrunnspiksel og en bakgrunnspiksel, danner grunnlaget for å identifisere skyggepiksler. En åpenbar svakhet ved metoden er at den kun fungerer for fargebilder. I segmenterte regioner med mye skygge, ga implementasjonen gode resultater. Vi gjorde også forsøk med å anvende HSI-analyse for å eliminere refleksjoner fra kjøretøyenes frontlykter. HSI-analyse fungerte ikke for denne anvendelsen.

Kapittel 7

Objektgjenkjenning

Dette kapittelet beskriver hvordan vi gjenkjenner segmenterte objekter i X-Analyzer. Som det fremgår av flytdiagrammet i figur 7.1, samarbeider denne modulen med følgingsmodulen. Objekter klassifiseres tentativt i hvert bilde. Følgingsmodulen etablerer temporal koherens mellom objekter fra etterfølgende bilder. Den endelige klassifiseringsbeslutningen tas dermed med bakgrunn i observasjoner av samme objekt over tid.



Figur 7.1: Objektgjenkjenningsmodulens plassering

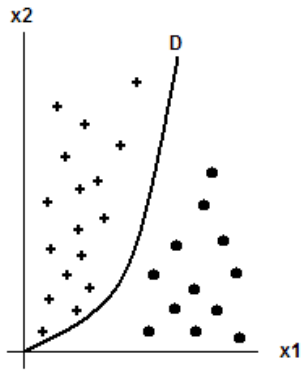
Først beskrives ulike kjente metoder for representasjon og klassifisering. Deretter følger en beskrivelse av vår implementasjon med påfølgende resultater og konklusjon.

7.1 Generelt

Vi har valgt å inkludere enkel objektgjenkjenning da det i mange tilfeller er nyttig å bestemme hvilken klasse segmenterte objekter tilhører. X-Analyzer er utviklet primært for å detektere konfliktsituasjoner mellom kjøretøy i veibanen. Å kartlegge fotgjengeres bevegelsesmønster er et underordnet mål. Vi antar derfor i utgangspunktet at objekter er kjøretøy. Dersom tilstrekkelig

bevis taler for at et objekt representerer en fotgjenger, klassifiserer vi objektet som dette. Implisitt i dette ligger en antakelse om at de segmenterte objektene representerer enten fotgjengere eller biler. Andre klasser som for eksempel syklister, fugler, støy etc., er ikke tatt hensyn til.

Gjenkjenning er komplisert ved at objektene ikke er filmet rett ovenfra. Dette gir opphav til ulik størrelse på objektene, avhengig av hvor langt fra kameraet de befinner seg. Ideelt bør vi derfor klassifisere ut ifra egenskaper ved objektene som er invariante med hensyn på størrelse. Et annet og større problem, er den ulike fasongen det samme objektet kan ha avhengig av orientering. Det er stor forskjell på en bil sett bakfra og ifra siden. Likeledes er det betydelig forskjell i geometrien til en fotgjenger avhengig av høyden kameraet er plassert i.



Figur 7.2: Vektorrom

Som antydet ovenfor er gjenkjenningsoppgaven todelt. Først må de segmenterte regionene representeres på en måte som er hensiktsmessig for videre prosessering [8, Kap. 11]. Deretter må ulike representasjoner sammenlignes i en klassifikator. Gjenkjenning av objekter krever representasjon i form av eksakte *deskriptorer*. Disse må reflektere objektenes egenskaper og være diskriminerende i den forstand at de skiller objekter i ulike klasser. Målet er å finne et færrest antall deskriptorer som karakteriserer objektet tilstrekkelig for utvetydig klassifikasjon. Deskriptorer kombineres gjerne i n-dimensjonale egenskapsvektorer (eng.: *feature vectors*) som benyttes til klassifikasjon. Figur 7.2 illustrerer en rekke slike vektorer representert ved deskriptorene (x_1, x_2) i et todimensjonalt vektorrom. Den euklidske avstanden mellom vektorene kan tas som et mål på likhet mellom de ulike objektene.

Det andre problemet knyttet til gjenkjenning er å få datamaskinen til å dra veksler på tidligere observerte objekter og nye objekter. Dette er det vi kaller klassifisering. Målet er å tilordne hvert objekt en klasse ut ifra dets representasjon. Klassifisering involverer ofte trening av en klassifikator. Klassifikatoren lærer først et sett med referanseformer i en treningsperiode. Etter treningen skal modellen være utviklet til å generalisere slik at den kan kjenne igjen objekter den hittil ikke har sett nøyaktig maken til, men som ligner på objektene fra treningen. Den åpenbare antakelsen er at treningssettet representerer det som er typisk for objekter fra hver klasse [13, Kap. 3]. Med bakgrunn i figur 7.2 går klassifisering ut på å finne funksjonen, D, som skiller klassene for de ulike datapunktene.

7.2 Kjente metoder

I dette delkapittelet vil vi først beskrive kjente metoder for representasjon. Videre beskrives kort noe teori som gir bakgrunn for selve klassifiseringen.

7.2.1 Representasjon

De vanligste objektdeskriptorene, og alle deskriptorer vi diskuterer, er basert på fasong. Disse kan hovedsaklig deles i to klasser: [32, 8]:

- Regionbaserte deskriptorer.
- Konturbaserte deskriptorer.

Videre kan deskriptorer være numeriske og ikke-numeriske [32]. Numeriske deskriptorer er ofte enkle skalarer som trekkes ut ifra objektet. Disse inkluderer areal, statistiske momenter etc. n numeriske deskriptorer kombineres enkelt i n-dimensjonale egenskapsvektorer som så klassifiseres i et høydimensjonalt vektorrom (eng.: *feature space*). Ikke-numeriske deskriptorer gir opphav til mer komplekse strukturerer som f.eks. grafrepresentasjoner, syntaktiske konturrepresentasjoner etc.

Regionbaserte deskriptorer

Vi vil her beskrive noen vanlige numeriske regionbaserte deskriptorer og deres anvendbarhet i X-Analyzer.

Areal. Dette er kanskje den enkleste deskriptoren og kalkuleres enkelt med å telle antall piksler som utgjør objektet. Areal er ikke en robust deskriptor i X-Analyzer, siden perspektivproblemene medfører at objekter får større areal desto nærmere kameraet de befinner seg. En normaliseringsoperasjon slik som beskrevet i [2] vil kunne bedre dette.

Høyde/Bredde. Dette er også en enkel deskriptorer som beregnes effektivt. Høyde/breddeforholdet er dessuten invariant med hensyn på størrelse. Det kan derfor fungere godt som diskriminator i X-Analyzer.

Eulers Tall. Eulers tall, ϑ, er en topologisk invariant deskriptor som kan benyttes for objekter bestående av mer enn én region. Dersom S beskriver antall sammenhengende regioner i et objekt og N bestemmer antall hull, er ϑ i [32] gitt som

$$\vartheta = S - N \tag{7.1}$$

ϑ kan være en nyttig deskriptor f.eks. i forbindelse med tekstgjenkjenning der et bilde av en A åpenbart vil gi $\vartheta = 0$, mens et bilde av en B resulterer i $\vartheta = -1$ [8]. For X-Analyzer gir ikke denne deskriptoren noen mening side vi forutsetter at objekter kun består av en region uten hull.

Projeksjoner. Projeksjoner kan gjøres horisontalt $p_h(i)$ (beskriver høyde) og vertikalt $p_v(j)$ (beskriver bredde) i binære regioner. Horisontal og vertikal projeksjon av en region er i [32] definert som henholdsvis

$$p_h(i) = \sum_j f(i,j) \qquad p_v(j) = \sum_i f(i,j) \tag{7.2}$$

Projeksjoner kunne vært passende deskriptorer for objekter i X-Analyzer, siden disse sier noe om den horisontale og vertikale utstrekningen til objektene. Problemet med projeksjoner er, slik vi ser det, at de i seg selv definerer en hel vektor, noe som vanskeliggjør enkel klassifisering. I sin enkleste form er de heller ikke invariante med hensyn på størrelse.

Hovedakser. Hvert objekt har to hovedakser: l_{major} og l_{minor}. Førstnevnte definerer den lengste aksen gjennom to punkter i objektet, mens l_{minor} representerer den lengste aksen i regionen vinkelrett på l_{major}. Disse aksene definerer også den minste ellipsen som kan

tilpasses regionen. Forholdet mellom disse to aksene definerer objektets eksentrisitet (eng.: *eccentricity*). Dette er en skalar verdi som enkelt kan benyttes i senere klassifisering. Videre er objektets orientering, gitt ved vinkelen, θ, mellom l_{major} og x-aksen, en nyttig deskriptor. Klassifisering med bakgrunn i egenskaper trekt ut ifra disse aksene, er benyttet blant annet i [2, 31]. Retning og eksentrisitet er dessuten invariante med hensyn på størrelse. I X-Analyzer er mennesker representert som høye, tynne objekter, mens biler har et lavere høyde/bredde-forhold. Derfor er hovedakser og egenskaper avledet fra disse, gode deskriptorer for X-Analyzer.

Kompakthet. Kompakthet er et mål på forholdet mellom kvadratet av omkretsen og areal. Den mest kompakte regionen er en sirkel. Kompakthet er en veldig enkel og vanlig deskriptor, men dessverre for sensitiv til støy og ujevnheter i objektkonturen til å benyttes i X-Analyzer. I [31] rapporteres det om lignende erfaringer.

Statistiske momenter. Statistiske momenter, μ_{pq}, av grad, $(p+q)$, er enkle skalare deskriptorer, som i [32] er definert ved

$$\mu_{pq} = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} (i - x_c)^p (j - y_c)^q f(i, j) \tag{7.3}$$

der (x_c, y_c) er koordinatene til regionens massesenter. Statistiske moment kan gjøres både rotasjons- og skaleringsinvariante ved normalisering. Vi har ikke implementert denne deskriptoren i X-Analyzer, da vi ikke synes de gir en intuitiv forståelse av objektets fasong.

Konturbaserte deskriptorer

Her beskrives et utvalg konturbaserte deskriptorer og deres anvendbarhet i X-Analyzer:

Omkrets. Objekters omkrets er den enkleste konturbaserte deskriptoren, men den sier i seg selv veldig lite om konturen til objektet. Den er invariant mot rotasjon, men ikke mot skalering. Derfor egner den seg dårlig til klassifisering i X-Analyzer.

Konturkoding. Konturkoder (eng.: *chain/freeman codes*) er ikke-numeriske deskriptorer som gir opphav til syntaktiske beskrivelser av objekters eksterne fasong [8]. Disse konstrueres ved å følge konturen og representere hver retning langs denne med et symbol fra alfabetet $\Sigma = \{N = 0, E = 1, S = 2, W = 3\}$. Koden kan så normaliseres med hensyn på startpunkt, ved å skifte den inntil strengen representerer minste numeriske verdi. Avstanden mellom referanseobjekter og observerte objekter kan så uttrykkes i Levenshtein-distanser [25] med en passende kostnadsfunksjon. Konturkoder er ikke invariante med hensyn på størrelse, og de krever referanseobjekter for å representere hver klasse. Dette gjør dem vanskelig å benytte som deskriptorer i X-Analyzer.

Krumning. En regions krumning (eng.: *curvature*) er en skalar, som definerer forholdet mellom den totale omkretsen av regionen og piksler der retningen i konturen endres signifikant [32]. Jo færre brå endringer i konturen, desto mindre er krumningen. Krumning kan beregnes effektivt ut ifra et objekts konturkode.

7.2.2 Klassifisering

Siden klassifiseringsmekanismen i X-Analyzer kun baserer seg på enkel sammenligning av objekters deskriptorer, kommer vi ikke til å gi en omfattende presentasjon av klassifiseringsmetoder her. Vi vil heller presentere et abstrakt rammeverk som danner utgangspunktet for noe av teorien rundt klassifiseringsmetoder.

Gonzales skiller i [8] mellom to hovedmetoder for klassifisering av objekter: *beslutningsteoretiske metoder* (eng.: *decision theoretic*) og *strukturelle metoder*. Den første av disse inkluderer tradisjonelle klassifiseringsmetoder basert på kvantitative egenskaper trekt ut fra objekter. Dette kan være deskriptorer som areal, omkrets, etc. Strukturell klassifisering er basert på kvalitative egenskaper. For eksempel er en konturkode en kvalitativ representasjon som kan klassifiseres med bakgrunn i formelle grammatikker. Vi vil kun se nærmere på beslutningsteoretiske metoder, da disse danner bakgrunnen for klassifiseringsfunksjonaliteten vi har implementert.

Beslutningsteoretiske metoder er basert på bruk av beslutningsfunksjoner (eng.: *decision functions*) [8, Kap. 12.2]. Disse klassifiserer egenskapsvektorer, $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T$, i én av W klasser. Målet er å finne beslutningsfunksjoner, $d_1(\mathbf{x}), d_2(\mathbf{x}), \ldots, d_W(\mathbf{x})$, som har egenskapen at dersom en vektor, $\mathbf{x}$, tilhører en klasse, ω_i, så gjelder

$$d_i(\mathbf{x}) > d_j(\mathbf{x}) \qquad j = 1, 2, \ldots, W;\ j \neq i. \tag{7.4}$$

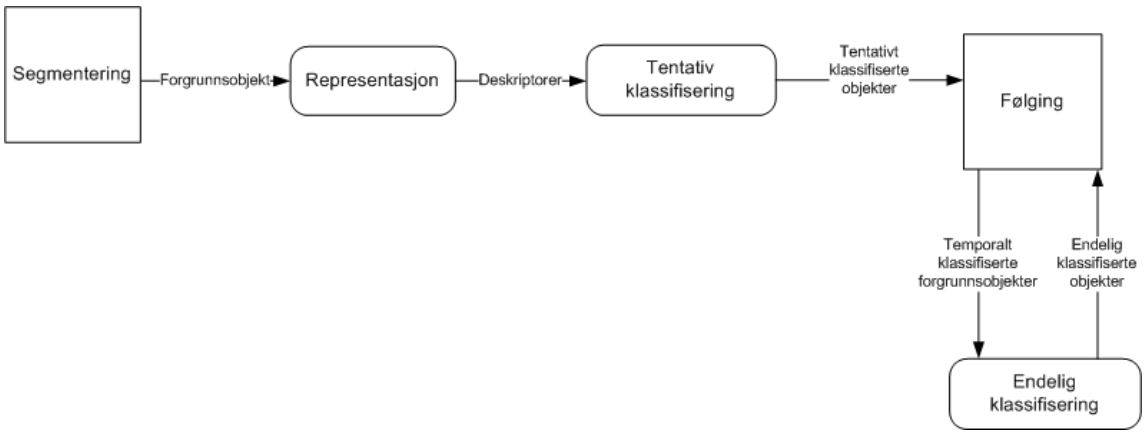
Med andre ord tilhører $\mathbf{x}$ klassen hvis beslutningsfunksjon gir høyest numerisk verdi. Beslutningsfunksjonene for hver klasse definerer klasseskiller når $d_{ij}(\mathbf{x}) = d_i(\mathbf{x}) - d_j(\mathbf{x}) = 0$ (D i figur 7.2).

En enkel beslutningsfunksjonen er minste distanse klassifikatoren (eng.: *minimum distance classifier*). Denne tar utgangspunkt i distansen mellom gjennomsnittet, $\mathbf{m}_j$, for hver klasse og observerte verdier, $\mathbf{x}$. Den beste matchen er klassen hvis $\mathbf{m}_j$ ligger nærmest (definert f.eks. ved euklidsk avstand) $\mathbf{x}$. Videre kan $\mathbf{m}_j$ oppdateres til å inkludere $\mathbf{x}$. Det kan vises [8] at denne prosedyren er ekvivalent med å evaluere funksjonen i ligning 7.5, for høyest numerisk verdi.

$$d_j(\mathbf{x}) = \mathbf{x}^T\mathbf{m}_j - \frac{1}{2}\mathbf{m}_j^T\mathbf{m}_j, \qquad j = 1, 2, \ldots, W. \tag{7.5}$$

7.3 Valgt metode

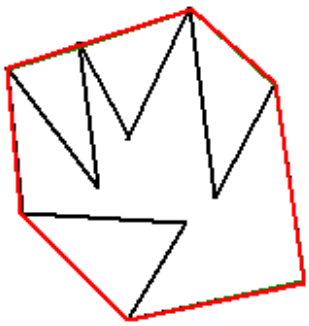
Dataflyten gjennom denne modulen er illustrert i figur 7.3. Forgrunnsobjektene mates inn fra segmenteringsmodulen. Først beregnes en rekke geometriske deskriptorer for objektrepresentasjon. Med bakgrunn i disse tas en tentativ beslutning om hvilken klasse hvert objekt tilhører. Den tentative klassifiseringen mates så inn i følgingsmodulen, som med bakgrunn i temporal korrelasjon mellom objekter fra etterfølgende bilder, beregner den endelige klassifiseringen for hvert objekt.



Figur 7.3: Dataflyt i objektgjenkjenningsmodulen

7.3.1 Representasjon

Når representasjon skal velges, må det gjøres visse antakelser om fasong for å finne diskriminerende deskriptorer. Kameravinkel og bevegelse gjør at objekter endrer geometriske egenskaper gjennom scenen. Dette gjelder spesielt kjøretøy som svinger. Derfor vil en representasjon som fungerer bra sett rett ovenfra, ikke fungere like bra dersom objektene er sett fra siden, slik som i vårt tilfelle.



Figur 7.4: Konvekst hull

Våre deskriptorer er basert på det konvekse hullet som omslutter hver segmenterte region. Et konvekst hull for en region er det minste polygonet som inneholder hele regionen. Samtidig må to vilkårlige punkter i regionen kunne knyttes sammen med et linjesegment som ikke faller utenfor polygonet [32]. Det konvekse hullet kan uformelt sies å være det polygonet som oppstår hvis man spenner ett strikk rundt alle punktene i regionen. Figur 7.4 illustrerer et polygon i svart og det omsluttende konvekse hullet i rødt.

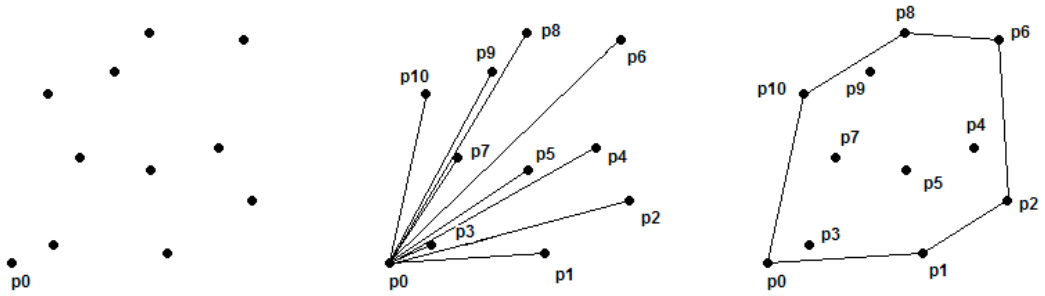
Ved å konstruere et konvekst hull rundt hver forgrunnsregion tettes topologiske hull, og ujevnheter langs randen av regionene utjevnes. Det finnes flere algoritmer for å beregne konvekse hull. Vi har valgt å implementere en versjon av Graham-scan fra [5]. Hovedpunktene i Graham-scan er skissert i algoritme 7.1 og illustrert i figur 7.5.

Algoritme 7.1 gir en polygonrepresentasjonen av det konvekse hullet for en forgrunnsregion.

For objektrepresentasjon er vi også interessert i den eksakte punktgeometrien til det konvekse hullet. Med punktgeometri mener vi en liste med koordinatene til de pikslene som representerer konturen til det konvekse hullet. Punktgeometrien til polygonet finner vi med algoritme 7.2 fra [32, Kap. 5]. Algoritmen tar utgangspunkt i det minste rektangelet som omslutter hvert konvekse hull, søker etter første objektpiksel og sporer konturen rundt objektet ut i fra denne. Hvert punkt på konturen lagres i en liste.

Algoritme 7.1 Graham scan

1. Finn punktet, p_0, med lavest y-koordinat. Hvis flere punkter har denne verdien, velg det punktet av disse med lavest x-koordinat. p_0 er garantert å være en del av det konvekse hullet.
2. Sorter de resterende punktene etter økende vinkel relativt til p_0.
3. Besøk hvert punkt p_i i sortert rekkefølge. Bygg polygonet som definerer det konvekse hullet ved å legge til tentative kanter så lenge det gjøres venstresvinger. Fjern kanter og backtrack til p_i ved høyresving.



(a) Punkter (b) Vinkler (c) Konvekst hull

Figur 7.5: Illustrasjon av Graham Scan algoritmen. Figur 7.5(a) viser pivot punktet, p_0, og punktene vi vil finne det konvekse hullet for. Figur 7.5(b) illustrerer punktene sortert på stigende vinkel. Figur 7.5(c) viser det konvekse hullet generert av algoritmen.

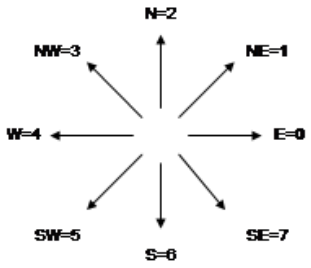
Retningene som omtales i algoritmen er kodet som i figur 7.6. Alle punktene i det konvekse hullets ytterkant er ved terminering av algoritme 7.2, gitt ved $p_0, \ldots, p_{k-2}$. Dette er i figur 7.7 vist som de grønne konturene som omslutter de blå regionene.

Vi har valgt å skille mellom to typer forgrunnsobjekter: Fotgjengere og kjøretøy. Fotgjengere er typisk langstrakte objekter, har en tilnærmet vertikal hovedakse og lite areal. Kjøretøy er mer klumpete, har en mer varierende vinkel på hovedaksen og som regel større areal. Dette er tydelig i figur 7.7.

Algoritme 7.2 Sporing av indre objektkontur

1. Finn punktet, p_0, med lavest y-koordinat. Hvis flere punkter har denne verdien, velg det punktet av disse med lavest x-koordinat. p_0 er startpiksel for regionsporingen. Legg til p_0 i listen L som definerer konturen. Definér en retningsvariabel, DIR, som lagrer retningen til siste bevegelse langs konturen. Initialiser DIR = 7 (SE).
2. Let mot klokken i 3×3 nabolaget til gjeldende piksel etter neste konturpiksel, p_k. Begynn søket etter p_k i retning:
   - (DIR + 7) mod 8 hvis DIR er jevn
   - (DIR + 6) mod 8 hvis DIR er odde

   Legg til p_k i L og oppdater DIR med retningen der p_k ble funnet.
3. Stopp hvis $p_k = p_1$ og $p_{k-1} = p_0$, hvis ikke gå til 2.



Figur 7.6: Retninger

Figur 7.7: Deskriptorer

Med bakgrunn i disse observasjonene har vi valgt å representere hvert forgrunnsobjekt ved en vektor, $\mathbf{v} = (v_\alpha, v_\beta, v_\gamma, v_\delta)^T$, bestående av følgende geometriske deskriptorer:

Areal, α. α beskriver arealet omsluttet av det konvekse hullet, hvis massesenter er representert ved hvite trådkors i figur 7.7. Areal er egentlig en uheldig deskriptor ettersom dette endres avhengig av hvor i bildet objektet observeres. Siden de andre deskriptorene er invariante med hensyn på areal, kan store langstrakte objekter slik som busser feilaktig identifiseres som fotgjengere. Siden busser alltid er representert ved stort areal uavhengig av posisjon og fotgjengere ved, til sammenligning, lite areal, vil α virke diskriminerende i dette tilfellet.

Formfaktor, β. Formfaktoren er et mål på rundheten til et objekt. Denne er i [9] definert som

$$\beta = \frac{4\pi \text{areal}}{(\text{konveks omkrets})^2} \tag{7.6}$$

Vi benytter i ligning 7.6 arealet av regionen, representert ved blått, og omkretsen av det konvekse hullet, representert ved grønt i figur 7.7. På denne måten ignoreres lokale irregulariteter i konturen. Formfaktor er en passende deskriptor siden fotgjengere ofte er representert ved langstrakte regioner ($\beta < 1.0$), mens biler ofte er mer klumpete ($\beta \sim 1.0$).

Orientering, γ. Et objekts hovedakse er definert ved to punkter, (x_1, y_1) og (x_2, y_2), på konturen med lengst distanse, $d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$, mellom seg. I figur 7.7 er hovedaksene for hvert objekt markert med røde linjesegment. Vinkelen γ mellom hovedaksen og x-aksen finnes ved hjelp av ligning 7.7.

$$\gamma = \arctan\left(\frac{y_2 - y_1}{x_2 - x_1}\right) \tag{7.7}$$

I praksis tas det negative argumentet i ligning 7.7, siden det er mer intuitivt å tenke på γ som en vinkel iforhold til den horisontale bunnen av bildet. Dersom $\gamma < 0$ settes $\gamma = \gamma + 180$ slik at $\gamma \in [0, 180)$. For fotgjengere er objektets hovedakse tilnærmet vertikal, mens for kjøretøy varierer denne mer. Dette er tydelig i figur 7.7.

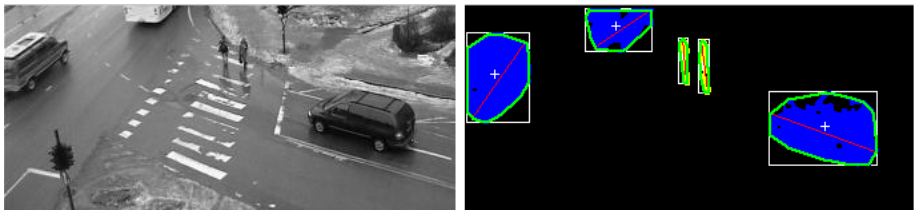
Bredde/høyde-forhold, δ. Forholdet mellom bredde og høyde (eng.: *aspect ratio*) på det omsluttende rektangelet parallelt med bildets hovedakser, er definert ved det minste punktet, (x_{min}, y_{min}), og det største punktet, (x_{max}, y_{max}), på konturen. Dette rektangelet er tegnet i hvitt rundt objektene i figur 7.7. Fotgjengere gjenkjennes ved at de normalt har større høyde enn bredde.

7.3.2 Klassifisering

Klassifiseringen i X-Analyzer benytter korrelasjon mellom objekter i etterfølgende bilder til å bygge klassifikasjonshypoteser. For hvert objekt, x, i hvert bilde, t, beregnes objektets representasjon som vektoren, $\mathbf{v}_t(x)$. Ut i fra $\mathbf{v}_t(x)$ tas en tentativ beslutning om objektets klasse. Denne beslutningen gjøres med bakgrunn i en tersklingsvektor, $\tau = (\tau_\alpha, \tau_\beta, \tau_\gamma, \tau_\delta)^T$, der hvert element beskriver en terskelverdi for hvert element i $\mathbf{v}(x)$. τ må settes manuelt siden optimale terskelverdier varierer fra scene til scene. x klassifiseres tentativt som *fotgjenger* dersom $\mathbf{v}_t(x)$ tilfredsstiller ligning 7.8. Hvis ikke klassifiseres objektet som *kjøretøy*.

$$v_\alpha < \tau_\alpha \text{ AND } v_\beta < \tau_\beta \text{ AND } v_\gamma \in [\frac{\pi}{2} - \tau_\gamma, \frac{\pi}{2} + \tau_\gamma] \text{ AND } v_\delta < \tau_\delta \tag{7.8}$$

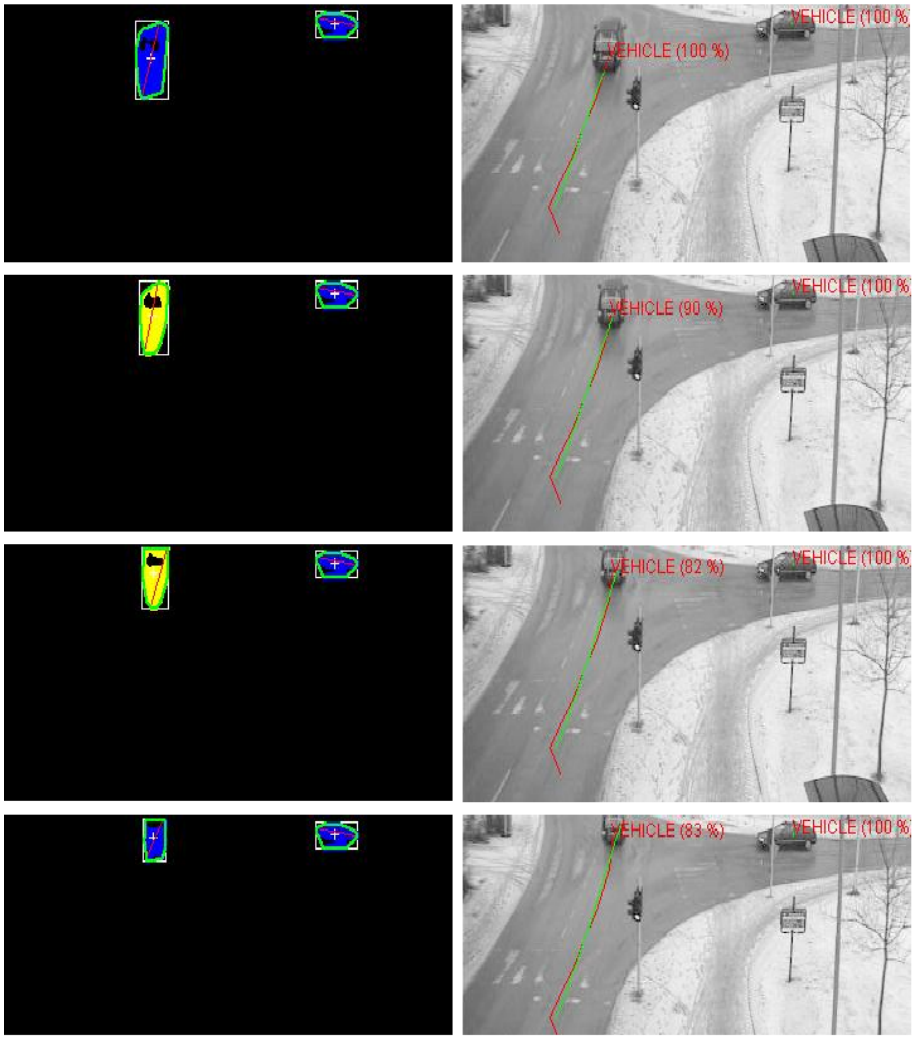
Figur 7.8 viser den tentative klassifiseringen av en rekke objekter i en scene. Objekter representert ved blå regioner er tentativt klassifisert som kjøretøy mens gule regioner representerer objekter klassifisert som fotgjengere.



Figur 7.8: Tentativ klassifisering

I følgingsmodulen etableres temporal koherens mellom objekter fra etterfølgende bilder. De tentative beslutningene for hvert objekt samles her, inntil en endelig statistisk avgjørelse om objektets klassifikasjon kan treffes. Å benytte temporal konsistens mellom segmenterte forgrunnsobjekter refereres i [19] som *"multiple hypothesis approach"*. Antakelsen er at dersom et objekt vedvarer over tid, er det en god kandidat for endelig klassifisering. Objekter som ikke kan følges et visst antall bilder forkastes som støy og klassifiseres dermed ikke. Slik bygges klassifikasjonshypoteser for hvert objekt over tid, inntil tilstrekkelig bevis taler for endelig klassifikasjon. Dette er nyttig fordi én instans av en bevegelsesregion, ikke nødvendigvis er representativ for objektet representert ved regionen. For eksempel vil to mennesker som delvis overlapper i ett bilde, lett kunne klassifiseres som et kjøretøy. For at et objekt skal klassifiseres endelig, kreves det at følgingsmodulen har greid å følge objektet gjennom et visst antall bilder. Dette er nærmere

forklart i kapittel 8. For hvert objekt følgingsmodulen greier å følge mellom etterfølgende bilder oppdateres sannsynligheten for at objektet tilhører hver av de to klassene. Når følgingsmodulen så har et komplett og vellykket spor, klassifiseres objektet til den klassen det er mest sannsynlig at det tilhører. Forbigående bevegelse som ikke beskriver reelle forgrunnsobjekter (svaiende trær, lys fra biler) og forgrunnsobjekter som plutselig forsvinner eller smelter sammen forkastes på denne måten automatisk, siden de ikke kan følges over tid.



Figur 7.9: Temporal klassifisering

Figur 7.9 viser en sporingssekvens med tentativ (venstre kolonne) og løpende klassifisering (høyre kolonne) av hvert forgrunnsobjekt. Den røde kurven bak hvert objekt i den høyre kolonnen viser den sporede stien, mens den grønne kurven viser midlere sti.

I øverste rad i figur 7.9 er bilen på vei oppover, klassifisert med 100% sikkerhet som kjøretøy (blått objekt). I andre og tredje rad klassifiseres bilen tentativt som fotgjenger (gult objekt). Dette gjør at den løpende sannsynligheten for at objektet er kjøretøy, reduseres til henholdsvis 90% og 82%. I det siste bildet klassifiseres objektet igjen som kjøretøy og sannsynligheten for kjøretøy

stiger til 83%. Den andre bilen i scenen som kommer inn fra høyre har en stabil klassifisering som kjøretøy med 100% sikkerhet i hele eksempelet.

7.4 Resultater

Dette delkapittelet dokumenterer resultater og terskelverdier ved klassifisering av objekter i de to datasettene. Terskelverdier vi fant å fungere godt for den tentative klassifiseringen i begge settene, er gitt i tabell 7.1.

Parameter	*Verdi*	*Beskrivelse*
τ_α	1000	Areal (antall piksler)
τ_β	0.9	Formfaktor (rundhet)
τ_γ	20.0	Orienteringsavvik (grader)
τ_δ	0.6	Bredde/Lengde (aspect ratio)

Tabell 7.1: Terskelverdier for tentativ klassifisering

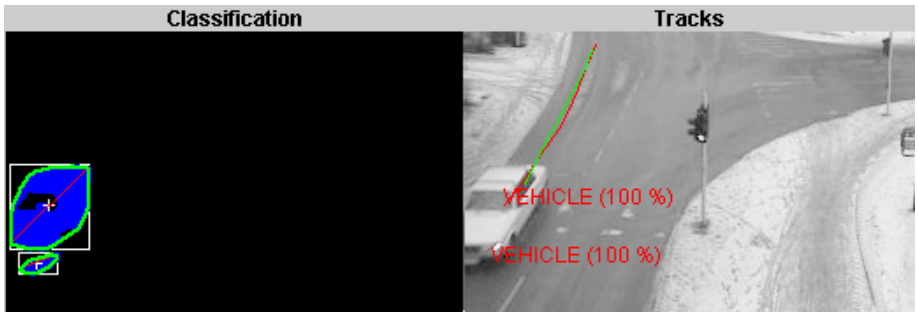
Den tentative klassifiseringen er i stor grad prisgitt forutgående segmentering og skyggefjerning. Dårlig segmenterte objekter gir dårlige representasjoner som resulterer i uriktig tentativ klassifisering. Dette er illustrert i figur 7.10. Til venstre i figuren er den tentative klassifiseringen, til høyre den temporale klassifiseringen. Fotgjengeren i bunnen av det venstre bildet har fått blå farge. Dette indikerer at objektet er tentativt klassifisert som kjøretøy. Årsaken til dette er den runde formen som gir opphav til $\beta \sim 1.0$ og høy δ. I den temporale klassifiseringen er fotgjengeren klassifisert som kjøretøy med hele 90% sannsynlighet.

Figur 7.10: Dårlig representasjon

Et annet problem er overlappende objekter. I figur 7.11 har to fotgjengere blitt segmentert som én region og dermed klassifisert som kjøretøy. Dette er et gjennomgående problem i våre testsett. Siden representasjonen av objektene er basert på det konvekse hullet som omslutter regionen, er disse to fotgjengerne feilaktig klassifisert som ett kjøretøy med 65% sikkerhet. Bildene i figur 7.11 viser to fotgjengere, men gir kun opphav til ett spor. Siden vi kun teller objekter med vellykkede spor i klassifiseringsstatistikken, har vi valgt å telle slik feilsegmentering som at én fotgjenger har blitt feilaktig segmentert som ett kjøretøy.

På den andre siden vil usammenhengende regioner som i virkeligheten tilhører samme objekt, gi opphav til flere stier og dermed flere endelige klassifiseringer. Dette er illustrert i figur 7.12. Her har en bil blitt segmentert, fulgt og klassifisert som to regioner. Siden det i virkeligheten kun er snakk om ett objekt, har vi i statistikken valgt å ignorere de(n) regionen(ene) som representerer objektet dårligst.

Figur 7.11: Overlappende objekter



Figur 7.12: Feilsegmentering

Dersom vi får en vellykket sti (og dermed endelig klassifisering på et objekt) for så å miste det før objektet er ute av scenen teller vi denne klassifiseringen med i statistikken. Dersom vi igjen får et nytt vellykket spor på det samme objektet telles også denne andre klassifiseringen med. Vi teller altså distinkte klassifiseringer hver gang vi får et vellykket spor på et objekt, selv om det i realiteten er snakk om det samme objektet.

Klassifiseringen beskrevet her er gjort på objekter i hele rektangelet som beskriver interesseområdet. Vi har altså utelatt å definere et kryss som ekskluderer fortausregioner. Dette er gjort for å få et høyere antall fotgjengere representert i resultatene enn vi ville fått dersom vi definerte scenen til kun å prosessere objekter i veibanen.

Tabell 7.2 viser resultatene av klassifiseringen for det enkleste av testsettene våre. Totalt klassifiseres 95,97% av objektene i dette testsettet riktig. Et problem knyttet til dette testsettet er at det er for få fotgjengere i det, i forhold til kjøretøy.

Faktisk klasse	*Totalt antall*	*Klassifisert som*		*Korrekt klassifisert*
		Fotgjenger	*Kjøretøy*	
Fotgjenger	12	7	5	58,33%
Kjøretøy	137	1	136	99,27%
Total	149	8	141	95,97%

Tabell 7.2: Klassifisering Høgskoleringen

Tabell 7.3 viser resultatene fra et vanskeligere testsett med flere objekter og hyppigere objektforekomst. Dette testsettet er bare halvparten så langt (antall bilder) som det forrige, men

inneholder nesten dobbelt så mange objekter. I dette testsettet er derfor problemet knyttet til overlappende objekter mye større enn i forrige sett. Likevel klassifiseres over 90% av objektene riktig.

Faktisk klasse	*Totalt antall*	*Klassifisert som*		*Korrekt klassifisert*
		Fotgjenger	*Kjøretøy*	
Fotgjenger	77	52	25	67,53%
Kjøretøy	219	3	216	98,63%
Total	296	55	241	90,54%

Tabell 7.3: Klassifisering Samfundet

7.5 Konklusjon

Vår antakelse om at alle objekter i krysset er enten kjøretøy eller fotgjengere, kan virke brutal. En kan forestille seg klasser for syklister, lastebiler, busser, motorsykler, støy, fugler, etc. Siden representasjonen i mange av disse klassene er nokså lik har vi valgt å begrense oss til to klasser, der det er mulig å diskriminere objekter på fasong. Antakelsen om at fotgjengere er representert ved langstrakte små objekter, mens kjøretøy er mer klumpete og til sammenligning større, har vist seg å være gyldig.

Representasjonen vi valgte tok utgangspunkt i fasongen til de segmenterte regionene. Ujevnheter i randen og topologiske hull gjorde at vi baserte representasjonen på det konvekse hullet til hvert objekt. Dette eliminerte mye støy, men dessverre også noe av detaljinformasjonen. Diskriminantene vi identifiserte i fasongen var nesten alle basert på forholdet mellom lengde og bredde i objektene. Deskriptorene er derfor ikke uavhengige av hverandre. Å benytte areal som deskriptor er egentlig uheldig siden objektene har ulikt areal utifra posisjon i bildet. Likevel fant vi at det fungerte bra til å skille veldig store og langstrakte objekter (som f.eks busser) fra mindre og langstrakte objekter (som f.eks. fotgjengere). Ingen av de andre deskriptorene var i stand til inkludere dette.

Den tentative klassifiseringen vi har valgt kan synes overdrevent enkel, siden den kun er basert på statisk terskling av skalare verdier. Likevel ser det ut til at den gir adekvate resultater i kombinasjon med en løpende temporal klassifisering. For kjøretøy gir modulen gode resultater i begge våre testsett med 99,27% og 98,63% korrekt klassifisering for henholdsvis Høgskoleringen og Samfundet. Fotgjengere identifiseres ikke like godt, med henholdsvis 58,33% og 67,53% korrekt, for hvert av de to testsettene. Årsaken til dette er todelt. For det første er forekomsten av fotgjengere i begge kryssene veldig lav, relativt til forekomsten av kjøretøy. Lavt antall observasjoner kan være årsak til uriktig klassifisering.

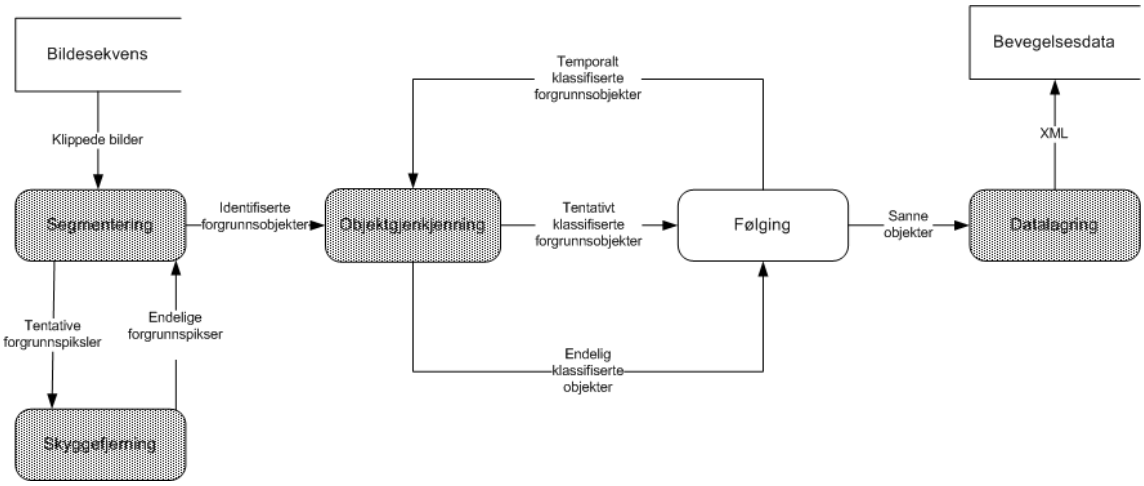
For det andre er applikasjonens hovedoppgave å detektere situasjoner som involverer kjøretøy. Siden scenedefinisjonen normalt vil ekskludere fortausregioner, vil den normale prosesseringen kun gjelde veibanen. I veibanen er normalt forekomsten av andre objekter enn kjøretøy lav. Derfor er terskelverdiene bevisst vridd mot å gi fordelaktig klassifisering som kjøretøy. Vi ser det altså som viktigere å få en høy *andel* korrekt klassifiserte kjøretøy enn et høyt *antall* korrekt klassifiserte fotgjengere, siden førstnevnte er overrepresentert i våre testsett.

Kapittel 8

Følging

"Følging" har vi avledet fra det engelske ordet *"tracking"*. Vi bruker *"følgingsmodulen"* for det engelske ordet *"the tracker"*, dvs. den modulen som utfører selve følgingen. Videre bruker vi det norske ordet *"sti"* for det engelske ordet *"track"*.

Figur 8.1 viser følgingsmodulen som det fjerde prosesseringssteget i X-Analyzer. For hvert bilde tar den inn ett sett segmenterte forgrunnsregioner. Oppgaven er å etablere temporal korrelasjon ved å koble segmenterte *regioner* fra et bilde sammen med *objekter* i foregående bilde. Resultatet er sanne objekter og bevegelsesbaner. Vi refererer konsekvent til regioner som resultatet fra segmenteringen, mens med objekter forstås en serie korrelerte regioner.



Figur 8.1: Følgingsmodulens plassering

Som det fremgår av figur 8.1, samarbeider følgingsmodulen med objektgjenkjenningsmodulen om den endelige klassifiseringen av objektene. Klassifiseringen av de temporalt korrelerte objektene bygger på de tentativt klassifiserte regionene fra foregående steg.

Ettersom segmenteringen ikke alltid er korrekt må følgingsmodulen være robust mot at

- Objekter kan forsvinne fra enkeltbilder og dukke opp senere.

- Støy kan segmenteres som forgrunn.
- Objekter kan okkludere eller komme så nær hverandre at de blir segmentert som samme region.

8.1 Generelt

Følging kan implementeres med utgangspunkt i:

1. Data fra én kameravinkel.
2. Data fra flere kameraposisjoner som ser scenen fra forskjellige vinkler. Ut ifra dette estimeres stier med større sikkerhet.

Vi har valgt å benytte opptak fra kun én kameravinkel. Vi hadde verken tid eller tekniske ressurser til å gjøre opptak og korrelere data fra flere kameravinkler.

8.1.1 Én kameravinkel

Det største problemet forbundet med å følge objekter fra én kameravinkel er okklusjon. Okklusjon oppstår når man projiserer et tredimensjonalt system, hvor et objekt står foran et annet, på en todimensjonal flate. Bilder fra én kameravinkel vil alltid være en todimensjonal projeksjon av et tredimensjonalt system. Ettersom objektene til enhver tid observeres fra én side, har man verken informasjon om hele formen til objektene eller informasjon nok til å unngå eventuelle okklusjonsproblemer.

Okklusjon kan få konsekvenser for følging fordi objekter (delvis) forsvinner fra bildet og dermed ikke blir gjenkjent. Et okkludert objekts sti blir ufullstendig og i noen tilfeller direkte feil.

Okklusjonsproblemet kan løses på to måter:

1. Ved bruk av heuristikker og resonnering som motvirker effekten av okklusjon.
2. Ved bruk av redundante data fra flere kameravinkler.

8.1.2 Flere kameravinkler

Med data fra flere kameravinkler kan man unngå okklusjon. På den andre siden øker kompleksiteten siden:

1. Data fra de forskjellige kameraene må synkroniseres i tid.
2. Kameraene må kalibreres på en konsistent måte for å transformere pikselkoordinater til verdenskoordinater.
3. Transformasjonen av pikselkoordinater til verdenskoordinater er beregningsmessig tungt.
4. n videostrømmer gir n ganger så mye data.

Kang gir i [15] løsninger på noen av disse problemene. Det rapporteres her om gode følgingsresultater med utgangspunkt i data fra flere kameravinkler.

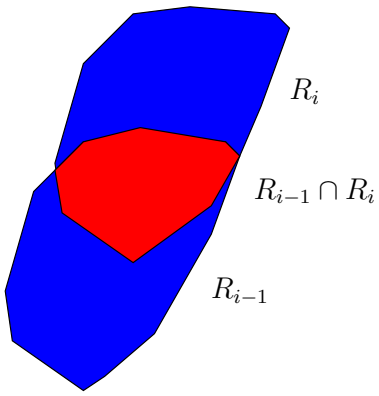
8.2 Kjente metoder

Dette delkapittelet presenterer kjente metoder som kan benyttes for å følge objekter i segmenterte videosekvenser.

- Ressem gjorde i [29] en antakelse om at segmenterte regioner som tilhører samme objekt overlapper i etterfølgende bilder.
- Lipton [19] korrelerer segmenterte regioner med maler som kontinuerlig blir oppdatert.
- Kalman-filtre fra [32] har blitt benyttet for å estimere stier.

Optisk flyt kan også benyttes for følging. Denne metoden ble gjennomgått i avsnitt 5.2.3. Siden vi ikke har benyttet optisk flyt for segmentering, er det heller ikke naturlig å bruke det for følging. Metoden blir derfor ikke nærmere diskutert.

8.2.1 Overlappende objekter



Figur 8.2: Overlappende objekter

Hvis en sampler videosekvensen med høy nok frekvens, vil alle objekter som beveger seg gjennom en scene (delvis) overlappe seg selv i etterfølgende bilder [29]. Hvis et objekt representeres som en region, R_i i bilde i, kan dette formaliseres som:

$$R_i \cap R_{i-1} \neq \emptyset \tag{8.1}$$

Figur 8.2 viser hvordan objekter kan følges mellom bilder. I figur 8.2 er regionene R_i og R_{i-1} markert i blått. Snittet, $R_i \cap R_{i-1}$, av de to regionene er markert i rødt. Metoden vil korrelere disse to regionene som samme objekt. Initialisering og terminering av en sti gjøres enkelt:

1. Dersom det eksisterer en region R_i og ingen region R_{i-1}, initialiseres R_i som et nytt objekt.
2. Dersom det eksisterer en region R_{i-1} og ingen region R_i, termineres følgingen. Objektet antas å ha forlatt scenen.

Algoritmen får problemer når ulike objekter, A og B, blir segmentert som samme region.

- Sett at algoritmen følger A og B som to objekter. I bilde f_i blir objektene segmentert som én region. I bilde f_{i+1} blir objektene igjen korrekt segmentert. Dette fører til at:
  1. Posisjonen til regionen i f_i er feil. Den representerer verken A eller B.
  2. Algoritmen må i bilde f_{i+1} velge hvilke av regionene som representerer hvert av objektene A og B.
- Sett at algoritmen følger A og B som to objekter. I bilde f_i–f_{i+k} blir objektene segmentert som én region. Algoritmen vil da følge A og B som om det var ett objekt. Hvis regionen

som representerer A og B i tillegg forsvinner ut av scenen før f_{i+k}, vil objektene kunne telles som kun ett objekt.

- Sett at algoritmen frem til og med bilde f_i følger A og B som om det var ett objekt. I f_{i+1} og utover segmenteres objektene korrekt. Dette fører til at:
  1. Posisjonen til både A og B frem til og med f_i er feil.
  2. Stien til ett av objektene blir feil siden den vil begynne der regionen har massesenter i f_{i+1}.
  3. Følgingsmodulen må velge hvilket objekt som skal tilordnes stien frem til og med f_i.

8.2.2 Korrelasjon

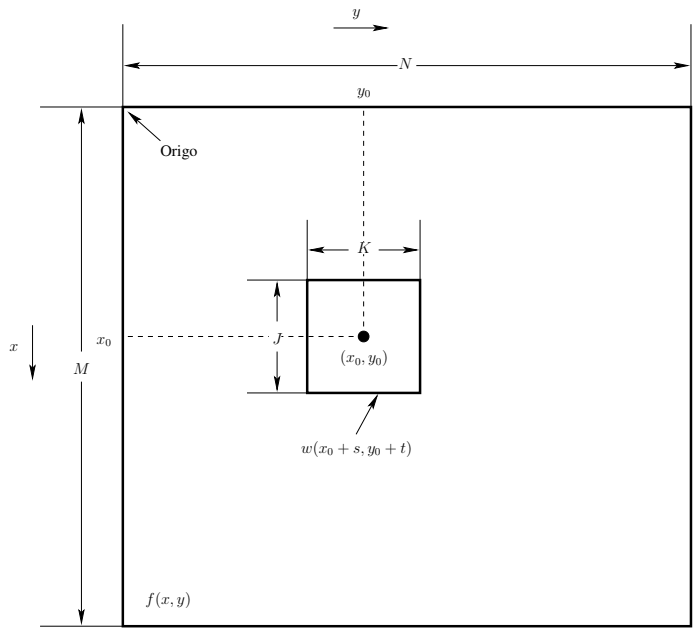
Lipton [19] sammenligner segmenterte objekter med maler (eng.: *templates*). Det benyttes en korrelasjonskoeffisient som uttrykker korrelasjon mellom en mal og et segmentert objekt. Malen oppdateres gjennom et IIR-filter som vi kommer tilbake til i neste avsnitt. Korrelasjonskoeffisienten gjennomgås i avsnittet under.

Korrelasjonskoeffisienten

Gonzales & Woods gir i [8] følgende ligning for å beregne korrelasjonskoeffisienten, γ:

$$\gamma(x,y) = \frac{\sum_s \sum_t [f(s,t) - \bar{f}(s,t)][w(x+s,y+t) - \bar{w}]}{\left\{ \sum_s \sum_t [f(s,t) - \bar{f}(s,t)]^2 \sum_s \sum_t [w(x+s,y+t) - \bar{w}]^2 \right\}^{\frac{1}{2}}} \tag{8.2}$$

der $x = 0, 1, 2, \ldots, M-1$ og $y = 0, 1, 2, \ldots, N-1$. M og N er størrelsen på bildet f i henholdsvis x- og y-retning. $\bar{w}$ er gjennomsnittet av pikslene i vinduet med malen w som glides over f. $\bar{f}$ er gjennomsnittet av pikslene i f som sammenfaller med pikslene i w. Summene tas over koordinatene, (s,t), som er felles for f og w. $\gamma(x,y)$ normaliseres i intervallet $[-1,1]$ [8]. Figur 8.3 illustrerer de ulike parametrene i ligning 8.2.



Figur 8.3: Parametre i korrelasjonskoeffisienten [8]

IIR-filter

Et IIR-filter (eng.: *Infinite Impulse Response filter*) oppdaterer malen, R_n, på omtrent samme måte som bakgrunnsmodellen i segmenteringen (kapittel 5):

$$R_n = \alpha M_n + (1 - \alpha) R_{n-1} \tag{8.3}$$

M_n er piksler fra regionen i bildet som ligner mest på malen. Korrelasjonen mellom en region fra bildet og malen bestemmes ut fra korrelasjonskoeffisienten γ (ligning 8.2). $\alpha \in [0, 1)$ bestemmer i hvilken grad nye data fra region M_n i bildet får påvirke malen.

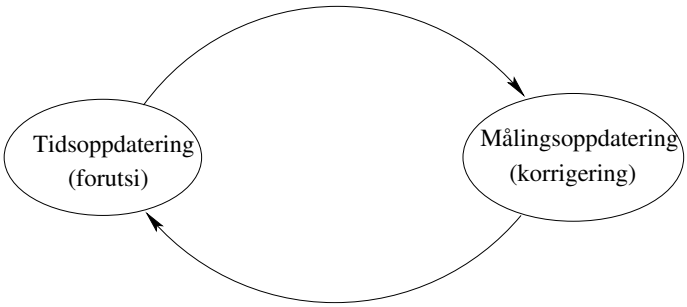
Prøveimplementasjon

Vi lagde en prøveimplementasjon av denne metoden i X-Analyzer. Korrelasjonsmetoden viste seg beregningsmessig intensiv og ga dårlige resultater. Vi bestemte oss derfor for ikke å videreføl ge denne ideen.

8.2.3 Kalman-filtre

Sonka [32, Kap. 15] beskriver bruk av Kalman-filtre for bevegelsesanalyse. De fleste applikasjoner for bevegelsesanalyse modellerer objekter, $\mathbf{x}$. Vi bygger modeller av objektene basert på observasjoner, $\mathbf{z}_k$, over tid. $\mathbf{x}$ og $\mathbf{z}$ er egenskapsvektorer (eng.: *feature vectors*). Flere observasjoner, $\mathbf{z}_1, \mathbf{z}_2, \ldots$, av det samme objektet kan utnyttes slik at man får et bedre estimat av modellen $\mathbf{x}$.

Dessuten vil estimatet for modellen ved tid k kunne gi en prediksjon for den neste observasjonen $\mathbf{z}_{k+1}$.



Figur 8.4: Feedback-mekanismen i et Kalman-filter. Figuren er hentet fra [32, Kap. 15].

Kalman-filtre tilbyr en feedback-mekanisme, illustrert i figur 8.4, for akkurat dette. Feedback-mekanismen består av følgende steg:

1. Observere $\mathbf{z}_k$.
2. Estimere $\mathbf{x}_k$.
3. Forutsi $\mathbf{z}_{k+1}$.
4. Observere $\mathbf{z}_{k+1}$ ved å utnytte informasjon fra prediksjonen.
5. Oppdatere estimatet for modellen $\mathbf{x}_{k+1}$.

Når man bruker et Kalman-filter må følgende antakelser holde:

- Systemet vi observerer er lineært.
- Observasjoner er lineære funksjoner av den underliggende tilstanden.
- Støyen, både i systemet og i observasjonene, er hvit og normalfordelt.

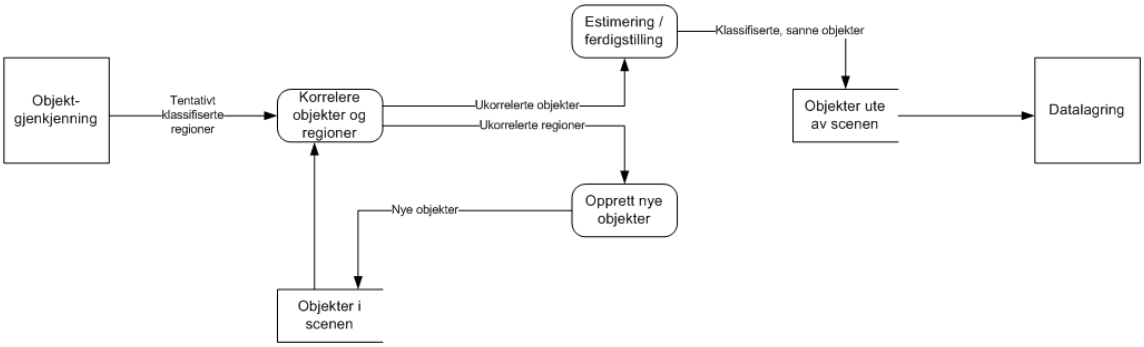
Koller[18] bruker aktive konturer (se avsnitt 6.2.2) i form av kubiske splines med tolv kontrollpunkter til å beskrive konturen til en bil. Kontrollpunktene brukes som egenskapsvektor i målingsfunksjonen i et Kalman-filter. Dermed blir konturen (estimatet av den underliggende modellen) forbedret etterhvert som man får flere observasjoner. Nok et Kalman-filter benyttes for å estimere posisjonen. En lineær affin transformasjon benyttes som et estimat for hvordan konturen flytter på seg. Den affine transformasjonen er sammensatt av en skalering[1] og en translasjon.

Kalman-filtre er nærmere behandlet i [32].

8.3 Valgt metode

Vi baserte vår implementasjon på ideen om overlappende objekter fra [29]. I tillegg inkluderer vi et resonneringssteg for å løse problemer knyttet til ufullstendig segmentering. Stiprediksjon gjøres ved bruk av et Kalman-filter.

[1] Skaleringens fortegn er avhengig av om bilen flytter seg mot kameraet eller fra det.



Figur 8.5: Dataflyt i følgingsmodulen

Dataflyten gjennom modulen er illustrert i figur 8.5. Vi refererer til *regioner* som resultatet fra segmenteringen ved hvert steg. Med *objekter* mener vi objekter vi følger gjennom scenen. Oppgaven er å korrelere de nye regionene med objekter i scenen. Dersom et objekt ikke korrelerer med en ny region, gis objektet enten et stiestimat eller det antas at objektet har forlatt scenen. Dersom man gir objektet et stiestimat, kan man forsøke å gjenoppta stien ifra neste steg. Regionene som ikke korrelerer med noen av objektene kan kvalifisere som nye objekter. Disse blir forsøkt fulgt videre i neste bilde.

Modulen samarbeider også med objektgjenkjenningsmodulen om temporal klassifisering av objekter som enten fotgjengere eller kjøretøy. For hvert objekt bygges klassifiseringshypoteser med bakgrunn i den tentative klassifiseringen for hver region. Mer om dette i kapittel 7.

8.3.1 Følgingsalgoritme

Algoritme 8.1 implementerer følging av objekter basert på prinsippet om overlappende regioner og Kalman-filter.

Algoritme 8.1 består av tre steg:

1. Den første for-løkken oppdaterer stiene til objektene i *currentObjects* som overlapper med de nye regionene i *segments*.
2. Den andre for-løkken behandler objektene i *currentObjects* som ikke overlapper med noen av de nye regionene i *segments*.
3. Den tredje for-løkken behandler regionene i *segments* som ikke overlapper med noen av objektene i *currentObjects*. Disse initialiseres som nye objekter i *currentObjects* eller klassifiseres som støy.

På linje 4 sjekkes det om et objekt overlapper med en region. Objektet i denne betingelsen er den sist tilordnede regionen. Dersom det sist tilordnede punktet på objektets sti var et estimat, sjekkes den den nye regionen mot objektet, translert til estimatets posisjon. Vi observerer altså et objekt og estimerer dets posisjon gjennom massesenteret. Deretter forutsier vi neste punkt på stien og observerer neste posisjon, for så å oppdatere estimatet. Dette beskriver et enkelt Kalman-filter. I linje 6 legges den overlappende regionens massesenter til objektets sti. Massesenteret representerer objektets koordinat. I de to neste linjene fjernes den overlappende

Algoritme 8.1 Tracker(*segments*)

```
1: unprocessed ← currentObjects
2: for all object ∈ currentObjects do
3:    for all segment ∈ segments do
4:       if object.segment overlaps segment then
5:          object.segment ← segment
6:          add segment's centroid to object's track
7:          remove segment from segments
8:          remove object from unprocessed
9:       end if
10:   end for
11: end for
12: for all object ∈ unprocessed do
13:   if object is moving out of scene then
14:      remove object from currentObjects
15:      add object to completeTracks
16:   else if object's confidence > c_min then
17:      add prediction to object's track
18:   else
19:      remove object from currentObjects
20:   end if
21: end for
22: for all segment ∈ segments do
23:   if segment is close to the edge of the scene then
24:      add new object to currentObjects
25:      object.segment ← segment
26:   end if
27: end for
```

regionen fra settet av regioner, *segments*, og det prosesserte objektet fjernes fra settet av objekter *unprocessed*. På denne måten blir verken en region eller et objekt prosessert mer enn én gang for hvert bilde.

En regions posisjon defineres ved massesenteret (x_c, y_c). Dette er gitt ved [32]:

$$\begin{aligned} x_c &= \frac{m_{10}}{m_{00}} \\ x_y &= \frac{m_{01}}{m_{00}} \end{aligned} \tag{8.4}$$

der m_{pq} er statistiske moment gitt ved:

$$m_{pq} = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} i^p j^q f(i,j) \tag{8.5}$$

I ligning 8.5 er f bildefunksjonen og i, j en regions koordinater.

Linjene 12–21 behandler objektene som ikke overlapper med noen av de nye regionene. Dette kan skyldes ufullstendig segmentering eller okklusjon. Delkapittel 8.4 diskuterer problemer som propagerer fra segmenteringsmodulen og skaper problemer for følgingen. Ukorrelerte objekter behandles på en av tre måter:

1. Objektet kan være på vei ut av scenen. Hvordan det bestemmes at objektet er på vei ut av scenen, beskrives i avsnitt 8.3.2. Hvis betingelsen er oppfylt, fjernes objektet fra *currentObjects* og legges til i settet av objekter som har blitt fulgt vellykket gjennom scenen, *completeTracks*.
2. Dersom det er sannsynlig at et spor ikke er falskt (linje 16), fortsettes følgingen ved å estimere nye punkter på stien. Estimering av nye punkter og konfidens av stier beskrives i avsnitt 8.3.2.
3. Hvis ingen av de foregående betingelsene er oppfylt, antas objektets sti å være falsk. Objektet fjernes da fra *currentObjects* og forkastes.

På dette tidspunktet er alle objekter i scenen behandlet. Regionene som korrelerer disse objektene er fjernet fra *segments*. Gjenværende regioner i *segments* er derfor kandidater for nye objekter. For at en region skal kvalifisere som et nytt objekt, må det oppstå i kanten av bildet. Dette sjekkes på linje 23. De gjenværende regionene antas å representere støy og behandles ikke videre.

8.3.2 Heuristikker

Dette delkapittelet presenterer heuristikker vi har benyttet. Heuristikkene presenteres kort i tabell 8.1 og blir grundigere gjennomgått i de etterfølgende avsnittene.

Type	Beskrivelse	Anvendelse
Gjennomsnittlig retning	Et objekts gjennomsnittlige retningsvektor beregnet over et visst antall observasjoner	Hjelpestørrelse for de andre heuristikkene.
Posisjon	Posisjonen til et objekt	Bestemmer om et objekt er "nært kanten" i scenen.
Retningsendring	Retningen til et objekt kan ikke endre seg altfor mye mellom etterfølgende bilder.	Bestemmer om et nytt punkt tilhører en sti.
Hastighetsendring	Hastigheten til et objekt kan ikke endre seg altfor mye mellom etterfølgende bilder	Benyttes til å bestemme om et nytt punkt tilhører en sti.
Stiestimat	Estimat for nye punkter langs en sti ut fra et objekts gjennomsnittlige retningsvektor	Benyttes når et objekt ikke korrelerer med noen regioner. På denne måten kan følgingsmodulen håndtere objekter som av og til forsvinner i segmenteringen.
Stikonfidens	Sannsynlighet for at en sti er ekte.	Benyttes for bestemme om en sti skal forkastes.

Tabell 8.1: Heuristikker i følgingsmodulen

Gjennomsnittlig retning

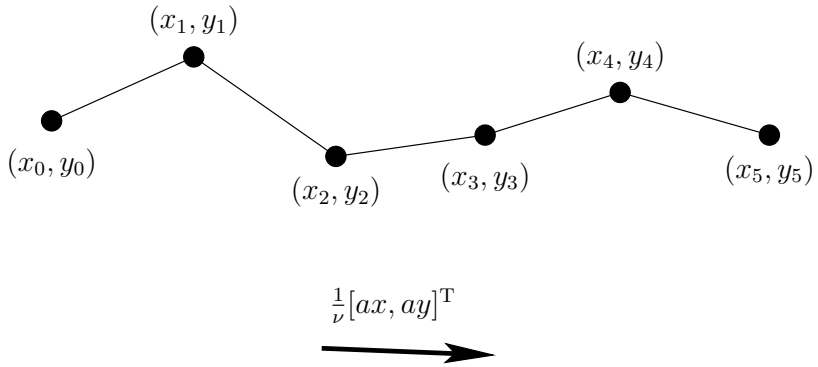
Gjennomsnittlig retning for en sti, $\mathbf{v}$, kan defineres som gjennomsnittlig endring i x- og y-koordinat i stien for de siste ν bildene. $\mathbf{v}$ kan benyttes til å:

- Oppdage unaturlig retningsendring.
- Oppdage unaturlig hastighetsendring.
- Estimere neste punkt på stien.

Algoritme 8.2 Average-Direction(*points*)

1: $points \leftarrow$ last ν points of $points$
2: **for all** $(x_i, y_i) \in points$ **do**
3: $\quad ax \leftarrow ax + (x_i - x_{i-1})$
4: $\quad ay \leftarrow ay + (y_i - y_{i-1})$
5: **end for**
6: **return** $\frac{1}{\nu}[ax, ay]^{\mathrm{T}}$

Algoritme 8.2 beregner gjennomsnittlig retning for ν punkter ved å summere retningsendringer i x- og y-retning og dividere på ν. Figur 8.6 viser en sti og dens gjennomsnittlige retning for de siste $\nu = 5$ punktene.



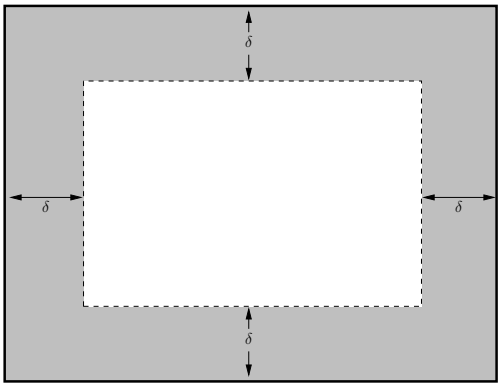
Figur 8.6: Gjennomsnittlig retning

Posisjon

Posisjon benyttes til å:

- Bestemme om en region skal kvalifisere som et nytt objekt.
- Bestemme hvordan ukorrelerte objekter skal behandles.

For at en region skal kvalifisere som et nytt objekt, må den befinne seg nær kanten av bildet. Regioner som oppstår midt i scenen forkastes som objekter fordi intet sikkert kan sies om hvor de kom inn i scenen. Nært bildekanten er definert ved en konstant grense på δ piksler. Dette er illustrert i figur 8.7. Ukorrelerte regioner med massesentrum innenfor δ piksler fra kanten blir initialisert som nye objekter.



Figur 8.7: Definisjon av δ

Objekter som ikke lenger korrelerer med noen av regionene fra segmenteringen kan forsvinne fordi:

1. Objektet har forlatt scenen, eller
2. Objektet blir klassifisert som støy.

Følgingsmodulen klassifiserer ukorrelerte objekter som en av disse ved hjelp av posisjon og gjennomsnittlig retning. Dersom et objekt befinner seg nær en kant i bildet *og* dets retningsvektor peker mot den samme kanten, klassifiseres objektet som at det har forlatt scenen. Hvis ikke forsøker følgingsmodulen å estimere videre sti basert på retningsvektoren. På denne måten kan stien gjenopptas på et senere tidspunkt dersom prediksjonen viser seg å korrelere med regioner i etterfølgende bilder. Dersom stikonfidensen (forklart senere i dette delkapittelet) blir for lav, klassifiseres objektet som støy.

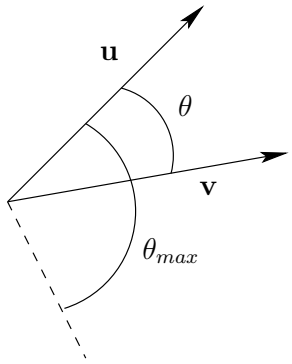
Retningsendring

Unaturlige svingebevegelser er med på å diskvalifisere regioner til å korrelere med objekter. Vi kan derfor sette en begrensning, θ_{max}, på vinkelen, θ, mellom vektoren, $\mathbf{u}$, som representerer gjennomsnittsretningen til et objekt og vektoren, $\mathbf{v}$, som dannes av det siste punktet på stien og massesenteret til en kandidatregion fra segmenteringen.

Dersom $\theta > \theta_{max}$ diskvalifiseres regionen fra stien.

θ beregnes fra cosinus-setningen som i ligning 8.6. Størrelsene er illustrert i figur 8.8.

$$\theta = \arccos \frac{\mathbf{u} \cdot \mathbf{v}}{|\mathbf{u}||\mathbf{v}|} \tag{8.6}$$



Figur 8.8: Retningsendringer

Hastighetsendring

Når et nytt punkt skal legges til på en sti, er det usannsynlig at objektet endrer hastigheten brått. Dersom vi sammenligner lengden til den gjennomsnittlige retningsvektoren, $|\mathbf{u}|$ med avstanden, d, fra det siste punktet på stien, p_i, til kandidaten, p_{i+1}, bør ikke disse være veldig forskjellige. Siden dette også er et mål på nærhet benytter vi den samme konstanten som vi brukte til å definere nærhet til bildekanten, δ. Dersom $|d - |\mathbf{u}|| > \delta$ diskvalifiseres regionen representert ved p_{i+1} fra stien.

Stiestimat

Dersom et objekt midt i bildet ikke lenger kan korreleres med en region fra segmenteringen, forsøker følgingsmodulen å estimere nye punkter på stien. Estimatet kan korreleres med regioner i etterfølgende bilder og på denne måten kan følgingen gjenopptas. Stiprediksjoner gjøres så lenge det fins tilstrekkelig stikonfidens (se neste avsnitt). Den gjennomsnittlige retningsvektoren, $\mathbf{u}$, og siste punkt på stien, p_i, benyttes til å estimere neste punkt, $\hat{p}_{i+1}$:

$$\hat{p}_{i+1} = p_i + \mathbf{u} \tag{8.7}$$

Stikonfidens

Stikonfidens, c, sier noe om hvor sikker algoritmen er på at en sti tilhører et faktisk objekt, og ikke er en falsk sti. c initialiseres likt for alle objekter og synker hver gang algoritmen må estimere neste punkt på stien. Når konfidensen kommer under et visst minimum, $c < c_{min}$, forkastes objektet. c reinitialiseres når et estimat av en sti igjen korrelerer med en region fra segmenteringen.

8.4 Resultater

Dette delkapittelet presenterer resultater fra våre to datasett. Parameterkombinasjonen vi fant å være optimal finnes i tabell 8.2. Avsnitt 8.4.2 diskuterer resultater med andre parameterkombinasjoner.

Parameter	Verdi	Beskrivelse
ν	5 bilder	Tidsvindu for beregning av gjennomsnittlig retningsvektor.
δ	15 piksler	Definerer nærhet til kant i scenen og punkt på sti.
θ_{max}	$\frac{2\pi}{3}$ radianer	Maksimal tillatt retningsendring for en sti.
c_{min}	0.2	Minste tillate stikonfidens før stien forkastes.

Tabell 8.2: Optimal parameterkombinasjon

Tabell 8.2(a) presenterer resultatene fra datasettet Høgskoleringen. 65 % av alle objektene ble korrekt fulgt. Omtrent 84 % av objektene ble adekvat fulgt. Med adekvat forstås at resultatet med tanke på inngangs- og utgangsregion er korrekt selv om følgingen igjennom krysset kan avvike fra objektets faktiske sti. Korrekte stier utgjør derfor et subsett av adekvate stier.

Omtrent 16 % av stiene gikk tapt, mens tre falske stier ble introdusert. Prosentandelene er basert på andel av de faktiske stiene. Siden falske stier ikke er faktiske stier men støy, utgjør de ikke en prosentandel, representert ved "N/A". Resultater fra datasettet Samfundet er presentert i tabell 8.2(b).

(a) Høgskoleringen

Kategori	Antall	Andel
Korrekte stier	74	64.9 %
Adekvate stier	96	84.2 %
Tapte stier	18	15.8 %
Falske stier	3	N/A

(b) Samfundet

Kategori	Antall	Andel
Korrekte stier	168	66.7 %
Adekvate stier	196	77.8 %
Tapte stier	56	22.2 %
Falske stier	18	N/A

Tabell 8.3: Resulater fra følgingsmodulen

8.4.1 Stier

Den følgende diskusjonen tar utgangspunkt i kategoriene fra tabell 8.2(a)– 8.2(b). Illustrasjonene er fra vårt primære datasett, Høgskoleringen. Diskusjonen er likefullt overførbar til datasettet Samfundet. Figurene som illustrerer stier benytter følgende innskrift for fulgte objekter:

Rød sti Den faktiske stien dannet ut i fra massesenter.

Grønn sti Den glattede stien.

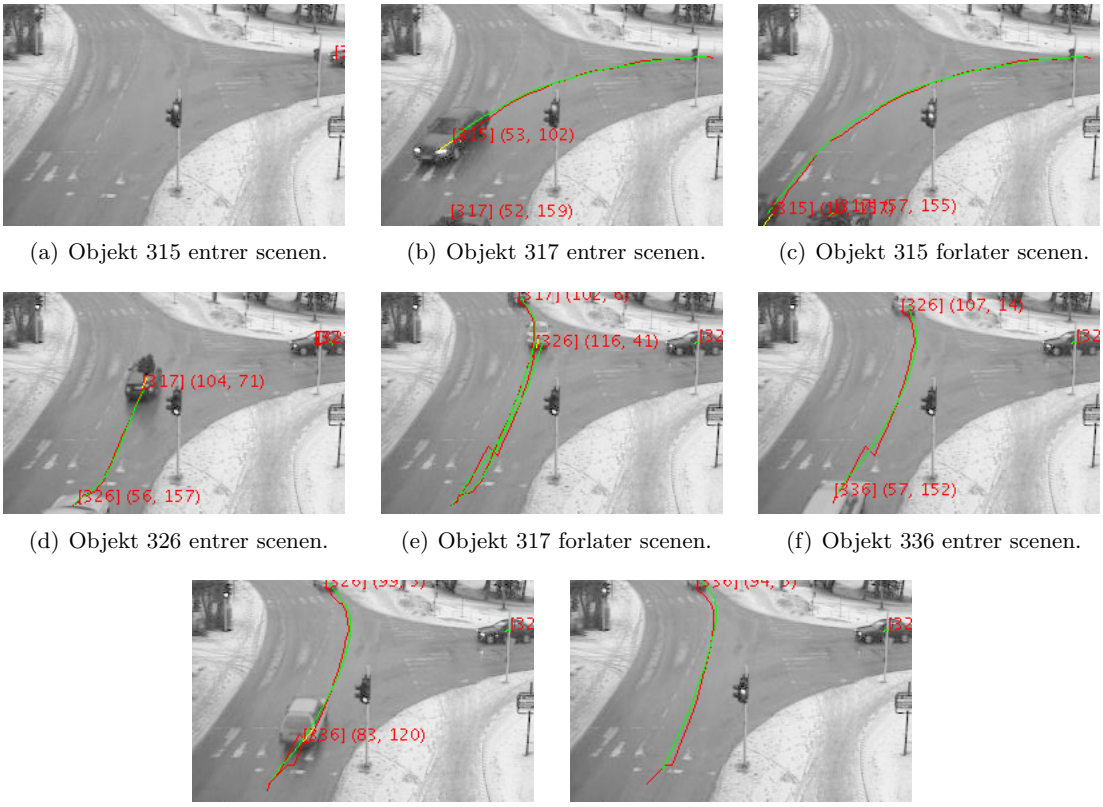
Blå sti Den estimerte stien. Kun til stede dersom følgingsmodulen må estimere neste punkt.

Gul strek Gjennomsnittlig retningsvektor. Kun synlig i noen tilfeller.

Rød skrift ID og massesenterets posisjon i (x, y) koordinater.

Korrekte stier

Figur 8.9 viser en bildesekvens fra Høgskoleringen med fire korrekte stier fra objekt 315,317,326 og 336.



(a) Objekt 315 entrer scenen. (b) Objekt 317 entrer scenen. (c) Objekt 315 forlater scenen.

(d) Objekt 326 entrer scenen. (e) Objekt 317 forlater scenen. (f) Objekt 336 entrer scenen.

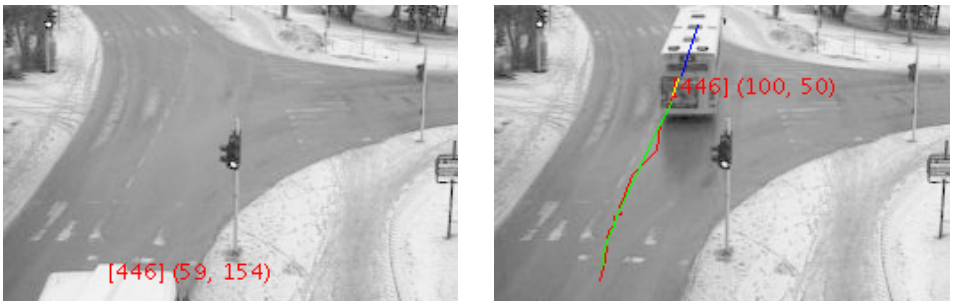
(g) Objekt 326 forlater scenen. (h) Objekt 336 forlater scenen.

Figur 8.9: Korrekt følging

Adekvate stier

For et objekt som følges gjennom scenen er vi hovedsaklig interessert i objektets svingebevegelser. Med dette mener vi hvilken region objektet kommer inn i og hvilken region objektet forsvinner fra scenen. Alle stier som kommer fra riktig region og forlater scenen i riktig region regnes som adekvate i X-Analyzer. Korrekte stier danner derfor et subsett av adekvate stier. 84 % og 78 % av stiene fra henholdsvis Høgskoleringen og Samfundet klassifiseres som adekvate.

Figur 8.10(a) viser et objekt (446) idet det kommer inn i scenen. Følgingsmodulen følger bussen korrekt gjennom mesteparten av scenen. I figur 8.10(b) mistes stien. Følgingsmodulen estimerer videre sti (blått) og godkjenner estimatet av stien når den kommer nært nok kanten. Estimert sti ender i riktig region og er dermed en adekvat sti.



(a) Objekt 446 entrer scenen (b) Objekt 446 forlater scenen

Figur 8.10: Adekvat sti

Tapte stier

Det er tre hovedårsaker til at stier går tapt. Tabell 8.4 oppsummerer tapte stier for testsettet Høgskoleringen. Et eksempel for hver av de tre årsakene presenteres kort.

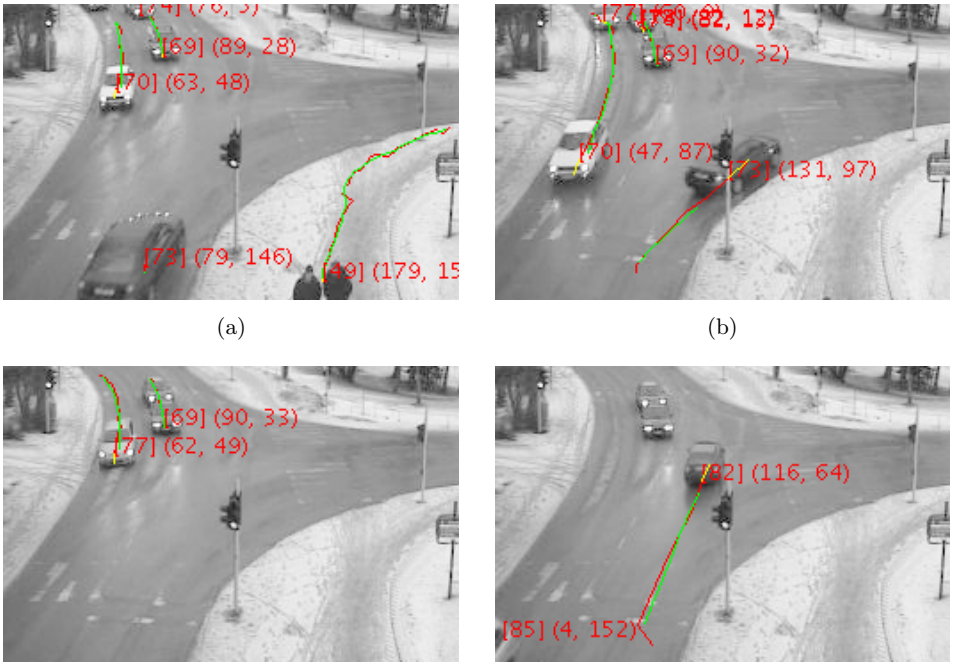
Antall	**Årsak**
7	Okklusjon
7	Brudd på konnektivitet
4	Objekter for små for segmentering

Tabell 8.4: Årsaker til at stier går tapt

Figur 8.11 viser en sekvens der flere biler okkluderer eller kommer så nært hverandre at de segmenteres som samme region. I figur 8.11(a) og figur 8.11(b) er alle objektene i scenen tilordnet stier. I figur 8.11(b) er to av bilene øverst i bildet veldig nære hverandre. I figur 8.11(c) er den bakre bilen delvis okkludert av bilen foran. Det oppstår usikkerhet siden de to regionene flyter sammen. Stien til den bakre bilen estimeres fremover. Siden den ikke finner noen korrelerende region blir den til slutt forkastet. Etterhvert mistes også den andre stien som opprinnelig var tilordnet den fremre bilen (figur 8.11(d)).

Figur 8.12 illustrerer en sti som forsvinner på grunn av brudd i objektets konnektivitet. Figuren viser en sekvens i tre steg. I figur 8.12(a) og figur 8.12(b) går alt som det skal. Objektet segmenteres som én region med én sti. I det neste steget splittes regionen fire regioner. Det er nå vanskelig å avgjøre hvilken av disse regionene som korrelerer best med objektet. Den flate regionen nærmest bildekanten velges som mest sannsynlig. Denne regionen har i figur 8.12(d) en hastighetsvektor (gul strek) som peker rett til høyre. Neste punkt estimeres å ligge langs den nedre bildekanten. Ingen regioner korrelerer med dette estimatet, og stien forkastes (figur 8.12(f)).

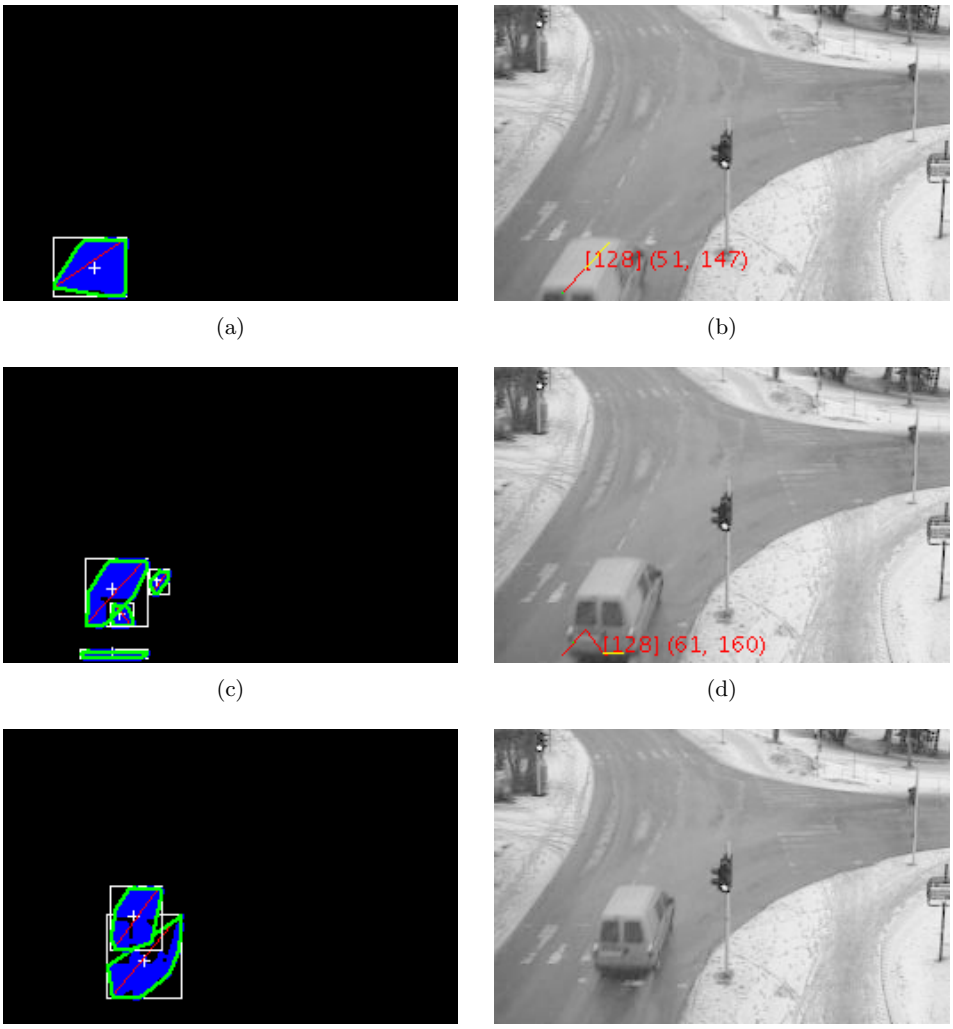
Modulen får også problemer når elementer i scenen er for små til at de blir segmentert som forgrunn. Figur 8.13 illustrerer dette. I figur 8.13(a) kommer en fotgjenger inn fra høyre. Den røde streken i forbindelse med fotgjengeren viser at objektet har blitt tilordnet en sti. Siden fotgjengeren beveger seg vekk fra kamera, gjør perspektivet at objektet blir mindre. I figur 8.13(b) har fotgjengeren fått en for liten representasjon til å bli segmentert. Følgingsmodulen kan derfor ikke lenger vedlikeholde en sti for objektet.



(a) (b)

(c) (d)

Figur 8.11: Tapte stier på grunn av okklusjon



(a) (b)

(c) (d)

(e) 7 (f)

Figur 8.12: Tapte stier på grunn av objektsplitting

(a) (b)

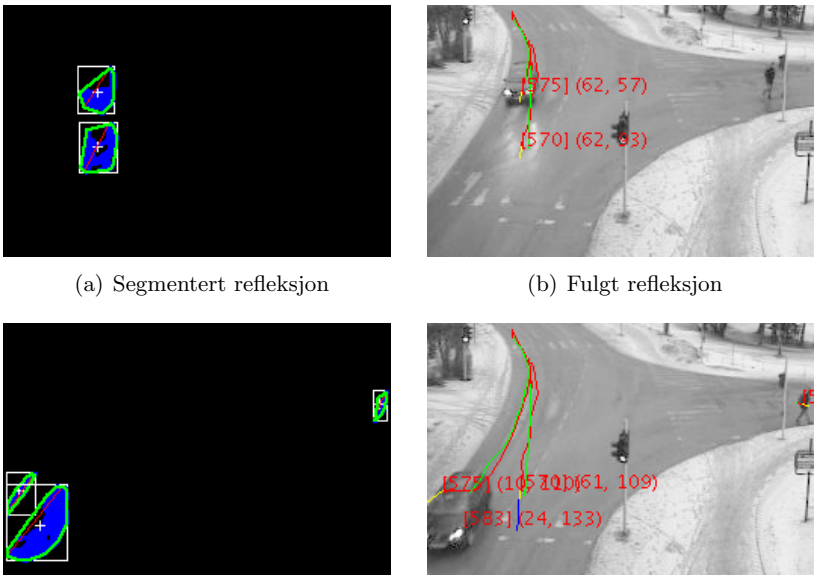
Figur 8.13: Tapte stier på grunn av for små objekter. Den røde sirkelen er tegnet på i ettertid.

Falske stier

Falske stier er stier følgingsmodulen har godkjent som ikke kommer av faktiske objekter. Dette har to hovedårsaker:

- Refleksjoner fra frontlykter i våt asfalt. Dette er illustrert i figur 8.14.
- Delvis okkluderte objekter i kanten av scenen gir konnektivitetsproblemer. Figur 8.15 illustrerer dette.

Figur 8.14 illustrerer lysproblematikken som fører til falske stier. Figur 8.14(a) og figur 8.14(b) viser henholdsvis segmenteringen og stien til to objekter (570 og 575). Objektet representert ved 570 kommer av refleksjoner i veibanen fra bilens (575) frontlykter. I figur 8.14(c) er ikke lenger refleksjonene segmentert som en egen region. Likevel illustrerer figur 8.14(d) et videre estimat (blått) for objekt 570. Siden prediksjonen fortsetter helt ut i kanten av bildet, blir objekt 570 tatt for å være en sti. Dette er åpenbart en falsk sti siden den ikke representerer et ekte objekt.

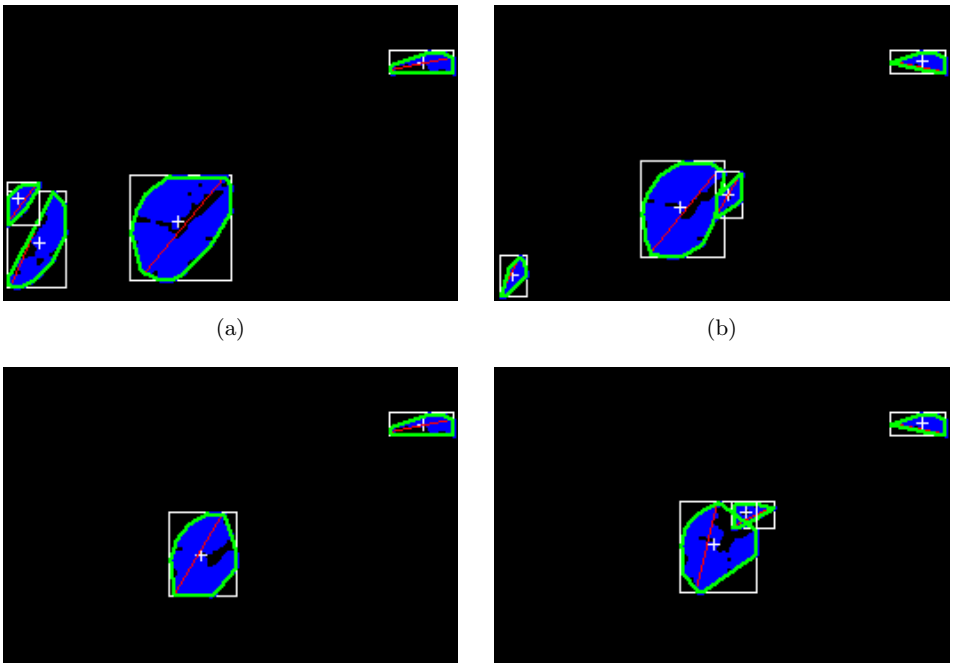


(a) Segmentert refleksjon (b) Fulgt refleksjon

(c) Segmentering uten refleksjon (d) Estimat av refleksjonsregion

Figur 8.14: Lysproblematikk

Figur 8.15 illustrerer et problem som oppstår når et delvis okkludert objekt står i ro ved kanten av bildet. Massesenteret til objektet oppe til høyre i figur 8.15(a)–8.15(d) oscillerer frem og tilbake. Følgingsmodulen estimerer punkter på stien til objektet. Estimatet forflytter seg mot kanten, helt til det er nært nok til at følgingsmodulen godkjenner stien som ekte. Siden objektet fortsatt er nært kanten vil følgingsmodulen opprette en ny sti og forsøke følge den. Den forrige stien blir dermed en falsk sti.



(a) (b)

(c) (d)

Figur 8.15: Kantproblematikk

8.4.2 Parametre

Her presenteres resultater fra følgingsmodulen med ulike parameterkombinasjoner. Resultatene tar utgangspunkt i den samme scenedefinisjonen som for resultatene presentert innledningsvis i dette delkapittelet. Når en parameter varieres, holdes de andre som i tabell 8.2. Vi har kun inkludert resultater fra datasettet Høgskoleringen. Poengene som gjøres her er likevel overførbare til datasettet Samfundet

ν

ν bestemmer antall punkter som skal benyttes ved beregning av den gjennomsnittlige retningen til et objekt. Lavere ν (tabell 8.4(a)) gir større sensitivitet for retningsendringer. Høyere ν (tabell 8.4(b)) gjør at retningen fastslås med større sikkerhet. For lav ν introduserer flere falske stier og stier som går tapt. For høy ν resulterer i like mange falske stier som ved optimal verdi. I begge tilfellene får vi færre adekvate stier enn ved optimal verdi ($\nu = 5$).

(a) $\nu = 3$

Kategori	**Antall**	**Andel**
Korrekte stier	64	56.1 %
Adekvate stier	93	81.6 %
Tapte stier	21	18.4 %
Falske stier	8	N/A

(b) $\nu = 7$

Kategori	**Antall**	**Andel**
Korrekte stier	68	59.6 %
Adekvate stier	93	81.6 %
Tapte stier	21	18.4 %
Falske stier	3	N/A

Tabell 8.5: Resultater ved varierende ν

δ

δ bestemmer hvor mange piksler det kan være mellom to punkter som karakteriseres som nær hverandre og hvor mange piksler det må være fra et objekt til en kant for at objektet skal klassifiseres som på vei ut av scenen. Lavere δ (tabell 8.5(a)) resulterer i færre falske stier på grunn av objekter som står i ro i kanten av bildet. Ved høyere δ (tabell 8.5(b)) får vi flere falske stier, samtidig som færre stier går tapt.

(a) $\delta = 10$

Kategori	**Antall**	**Andel**
Korrekte stier	33	28.9 %
Adekvate stier	75	65.8 %
Tapte stier	39	34.2 %
Falske stier	4	N/A

(b) $\delta = 20$

Kategori	**Antall**	**Andel**
Korrekte stier	60	52.6 %
Adekvate stier	99	86.8 %
Tapte stier	15	13.2 %
Falske stier	9	N/A

Tabell 8.6: Resultater ved varierende δ

θ_{max}

θ_{max} bestemmer hvor stor vinkel et punkt på stien kan ha i forhold til den gjennomsnittlige retningen til objektet. For lav verdi (tabell 8.6(a)) gjør algoritmen mer restriktiv på hvilke punkter som blir lagt til langs en sti. Dette gir færre falske stier. For høy θ_{max} (tabell 8.6(b)) gjør at flere punkter blir lagt til på stien.

(a) $\theta_{max} = \frac{\pi}{2}$)

Kategori	**Antall**	**Andel**
Korrekte stier	63	55.3 %
Adekvate stier	93	81.6 %
Tapte stier	21	18.4 %
Falske stier	6	N/A

(b) $\theta_{max} = \pi$

Kategori	**Antall**	**Andel**
Korrekte stier	72	63.2 %
Adekvate stier	93	81.6 %
Tapte stier	21	18.4 %
Falske stier	1	N/A

Tabell 8.7: Resultater ved varierende θ_{max}

c_{min}

c_{min} bestemmer hvor lenge følgingsmodulen kan følge et objekt gjennom estimater. Tabell 8.7(a) viser at en lav verdi for c_{min} ikke gjør noen forskjell i forhold til resultatene oppnådd ved bruk av den optimale verdien. Høyere c_{min} (tabell 8.7(b)) gjør at flere stier går tapt.

(a) $c_{min} = 0.1$

Kategori	**Antall**	**Andel**
Korrekte stier	74	64.9 %
Adekvate stier	96	84.2 %
Tapte stier	18	15.8 %
Falske stier	3	N/A

(b) $c_{min} = 0.5$

Kategori	**Antall**	**Andel**
Korrekt stier	71	62.3 %
Adekvate stier	89	78.1 %
Tapte stier	25	21.9 %
Falske stier	4	N/A

Tabell 8.8: Resultater ved varierende c_{min}

8.5 Konklusjon

I dette kapittelet har vi definert følgingsmodulens oppgaver. Kompleksiteten ved å benytte data fra flere kameravinkler gjør at har valgt å bruke data fra kun én kameravinkel i X-Analyzer. Vi har diskutert problemer en følgingsmodul må være robust mot. Dette inkluderer okklusjonsproblemer, problemer knyttet til ufullstendig segmentering og støy.

To metoder for følging ble evaluert:

1. Følging ved korrelasjon mellom regioner og maler.
2. Følging ved prinsippet om overlappende regioner.

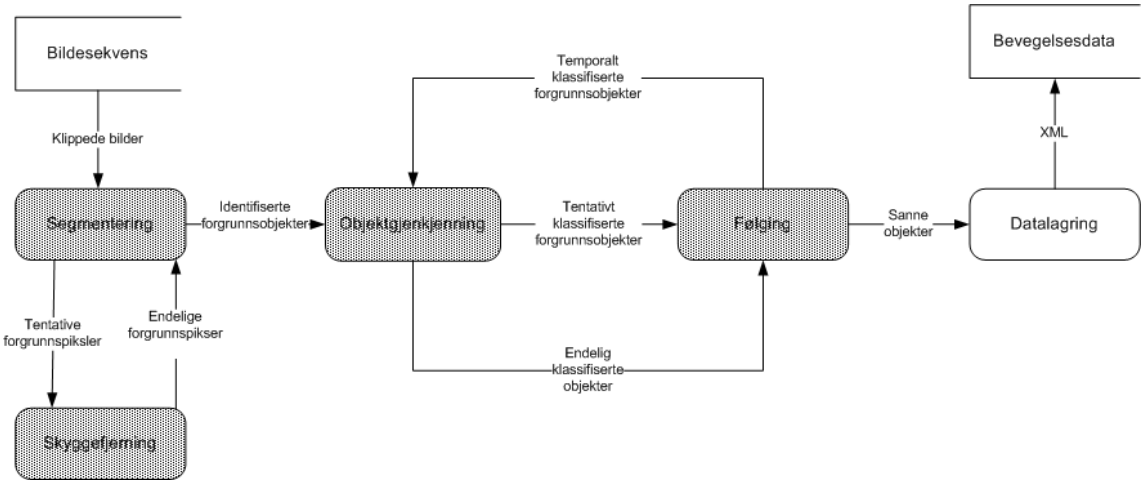
Følging ved korrelasjon viste seg å være uforholdsmessig beregningskrevende. Vi forkastet derfor denne idéen etter å ha eksperimentert med en prøveimplementasjon. Vi valgte istedenfor å følge objekter med bakgrunn i prinsippet om overlapp. For å gjøre metoden mer robust har vi benyttet en rekke heuristikker i følgingen. Vi har i tillegg integrert et Kalman-filter for å estimere stier i tilfeller hvor det ikke er mulig å finne noen korrelerende region.

Den valgte metoden ga gode resultater for vår applikasjon. 75–85 % av alle objektene i to testsett ga adekvate stier. Den implementerte metoden tar fire parametre. Vi fant verdier for disse parametrene som ga best resultat for testsettet Høgskoleringen. Diskusjonen rundt resultatene er også overførbar til testsettet Samfundet.

Kapittel 9

Datalagring

Figur 9.1 viser datalagringsmodulen som et element i dataflyten gjennom X-Analyzer. Modulen er det siste prosesseringssteget. Den definerer protokollen for informasjonsutveksling med applikasjoner som skal gjenfinne data generert av X-Analyzer. Dette gjelder for eksempel gjenfinningsverktøyet X-Retriever nevnt i kapittel 3.



Figur 9.1: Datalagringsmodulens Plassering

9.1 Generelt

Data må lagres i et strukturert format som gjør det lettere å finne igjen situasjoner og sekvenser. I en trafikkovervåkningsapplikasjon vil det kunne være interessant å lagre:

- Posisjonsangivelser.
- Et objekts klassifisering.
- Hastighetsvektorer i ulike posisjoner.

- Avstand mellom objekter.
- Kalibreringsdata for kamera.
- Geometriske egenskaper ved objektet (størrelse, omkrets).
- Visuelle egenskaper (farge, kjennetegn).

Vi har gjort valg underveis som gjør det vanskelig å lagre noe av denne informasjonen. Eksempelvis har vi valgt å ikke konsentrere oss om kalibrering av kameraet og transformasjon av kamerakoordinater til verdenskoordinater. Dette gjør det vanskelig å si noe om den virkelige hastigheten til et objekt eller et objekts virkelige avstand til et annet objekt i scenen. Det kunne også vært interessant å identifisere kjøretøy utifra registreringsnummer. Oppslag i sentrale databaser kunne da gitt ytterligere informasjon om kjøretøy og eier. Vi anser dette å være på siden av vår målsetning og heller ikke gjennomførbart med det utstyret vi har hatt tilgjengelig for datainnsamling.

9.2 Kjente metoder

Det finnes to hovedmåter å lagre data: Strukturert eller semi-strukturert. Vanlige relasjonsdatabaser benyttes ofte for å lagre strukturerte data etter et bestemt skjema. Semi-strukturerte data lagres gjerne som eXtensible Markup Language (XML).

9.2.1 Strukturerte data

Den vanligste formen for lagring av strukturerte data er tradisjonelle databaser. Med strukturerte data menes data som struktureres strengt av et forhåndsdefinert databaseskjema. Det finnes en rekke kommersielle leverandører av ulike typer databaser. Den mest utbredte formen for databaser er *relasjonsdatabaser*.

En tradisjonell databaseløsning er for omfattende for vår type applikasjon. Vi trenger ikke all funksjonaliteten et slikt system tilbyr i forbindelse med transaksjonshåndtering, loggføring og tilgjengelighet. Dersom systemet hadde vært utviklet for å overvåke kryss kontinuerlig, og på denne måten generert større datamengder, hadde det vært mer naturlig og benyttet et konvensjonelt databasesystem. Vi vil ikke gå i mer detalj på dette området siden vi har valgt å benytte lagring i form av XML.

9.2.2 Semi-strukturerte data

XML er en standard for semi-strukturerte data definert av World Wide Web Consortium (W3C) [35]. Med semi-strukturerte data menes data som ikke er strukturert like strengt som strukturerte data i for eksempel en relasjonsdatabase. Dersom man ikke har en verdi for et felt i et XML-dokument, kan man utelate å si noe om at dette feltet eksisterer. Denne muligheten har man ikke i tradisjonelle databasesystemer der et felt må gis en verdi (f.eks. NULL i dette tilfellet).

XML er utvidbart i den forstand at man ikke trenger å følge noen standard. Egne elementer og tilhørende semantikk defineres i hvert enkelt tilfelle. W3C har utviklet standarder for hvordan

en kan definere subsett av XML. Det legges begrensninger på hvordan man kan uttrykke seg innenfor forskjellige *navnerom*. W3C arbeider også med standarder for hvordan en kan gjenfinne data i og transformere XML-dokumenter. Noen disse spesifikasjonene presenteres senere i dette delkapittelet.

Representasjon

Det finnes to spesifikasjoner for representasjon og aksess i XML-dokumenter gjennom programmeringsspråk. Disse er Simple API for XML (SAX) og Document Object Model (DOM) [40]. Begge disse spesifikasjonene er plattform- og språkuavhengige. SAX er basert på seriell aksess i gjennom en datastrøm fra XML-dokumentet. Det er derfor effektivt for aksesser som skal lese dokumentet sekvensielt. Seriell aksess gjør at SAX' representasjon av XML krever lite minne.

DOM er en W3C spesifikasjon. Denne er basert på representasjon av hele XML-dokumentet som et tre i minnet. DOM er derfor minnekrevende og best egnet i applikasjoner med et høyt antall tilfeldige aksesser og oppdateringer.

Dokumentdefinisjon

For dokumentdefinisjon benyttes Document Type Definition (DTD) eller XML Schema Definition (XSD). En dokumentdefinsjon er en grammatikk som definerer lovlige byggeblokker og deres sammensetning i et XML-dokument [36]. Dette kan være nyttig i flere sammenhenger:

- Det uttrykker en beskrivelse av hvordan dokumentet er oppbygd og formatet til dokumentet.
- Uavhengige grupper kan enes om å bruke en felles dokumentdefinisjon for datautveksling.
- Dokumentdefinisjoner kan brukes for automatisk validering av XML-dokumenter.

XSD er mer uttrykksfullt enn DTD:

- XSD er utvidbar i forhold til framtidige tillegg.
- XSD er skrevet i XML.
- XSD støtter datatyper.
- XSD støtter navnerom.

Transformasjoner

Transformasjon mellom ulike XML-representasjoner kan gjøres gjennom XML Stylesheet Language Transformations (XSLT) [39]. XSLT benytter XPath for å hente informasjon fra et kildedokument. Et eksempel er transformasjon fra et semantisk merket XML-*kildedokument* til et HTML-*resultatdokument*. Transformasjoner mellom ulike XML-representasjoner er dessuten nyttig for applikasjoner som trenger samme data strukturert på ulike måter. Vi henviser til [39] for mer om dette.

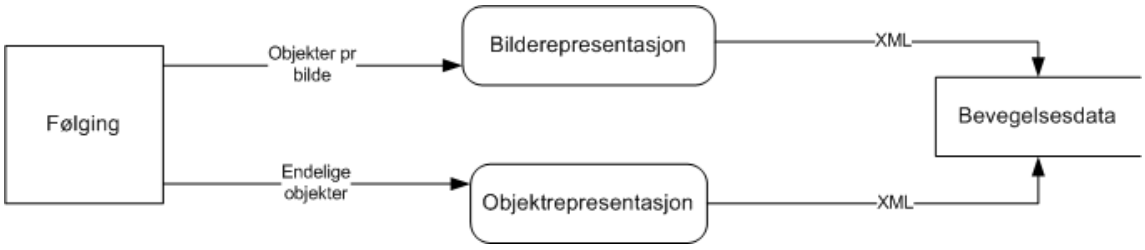
Spørringer

Spørringer i XML kan uttrykkes gjennom XPath eller XQuery. XPath er en spesifikasjon utviklet for å gjenfinne informasjon i XML-dokumenter [37]. Et XPath-uttrykk tilbyr navigering, seleksjon og ekstraksjon, gjennom en sti bestående av aritmetiske/logiske operatorer og sammenligningsuttrykk [6]. Returtypen er XML-elementer fra kildefilen som matcher spørringen.

XQuery er en annen spesifikasjon for spørringer i XML [38]. Til tross for at den i skrivende stund fortsatt ikke er en W3C-standard, eksisterer flere ufullstendige implementasjoner. XQuery er et sterkt typet spørrespråk. Det består av XPath-uttrykk, for-let-where-return (FLWR) utrykk og sort-by uttrykk. FLWR-uttrykk kan brukes til å utføre joins. I motsetning til XPath, gir XQuery mulighet til å konstruere den returnerte XML-representasjonen. Videre kan man definere egne funksjoner [6]. XQuery-spørringer er den mest uttrykksfulle formen for XML-spørringer.

9.3 Valgt metode

Figur 9.2 viser dataflyten gjennom datalagringsmodulen. For hvert bilde i videosekvensen sender følgingsmodulen informasjon om objektene i scenen til datalagringsmodulen. Til slutt, når alle bilder har blitt prosessert i resten av systemet, mottar datalagringsmodulen en liste med alle objekter som følgingsmodulen har fulgt vellykket gjennom scenen. Kun disse objektene får en XML-representasjon.



Figur 9.2: Dataflyt-diagram for datalagringsmodulen.

XML-representasjonen av den analyserte videosekvensen er redundant siden objektbevegelse blir lagret både for hvert bilde og for hvert objekt. Siden XPath ikke støtter join-operasjoner, var det enklest å lagre den samme informasjonen strukturert på forskjellige måter. Et alternativ hadde vært å benytte XSLT transformasjoner mellom ulike representasjoner.

For XML-representasjon har vi benyttet JDOM som er et DOM API for Java [12].

Tabell 9.1 oppsummerer hovedelementene X-Analyzer lagrer. Hvert av de fire hovedelementene beskrives i de neste avsnittene.

9.3.1 Interesseområde

Interesseområde (Region Of Interest) er den delen av scenen brukeren er interessert i å prosessere. Dette defineres som et rektangulært utsnitt av bildet. Informasjon som lagres er:

- Et punkt, (x, y), i bildekoordinater som markerer interesseområdets øvre venstre hjørne.

Element	XML	Beskrivelse
Interesseområde	`<roi>`	Lagrer startpunktet for øverste venstre hjørne i bildekoordinater samt bredden og høyden i antall piksler.
Krysset	`<junction>`	Lagrer de regionene brukeren definerer som kryss. Regionene er representert ved `<region>`-elementer. Hver region definerer et polygon og tilordnes en unik identifikator.
Bilder	`<frames>`	Lagrer informasjon om hvert bilde i sekvensen som `<frame>`-elementer. For hvert bilde lagres filnavn, en identifikator og hvilke objekter som er i hvilke regioner av krysset i dette bildet.
Objekter	`<tracks>`	Lagrer objekter som `<track>`-elementer. Hvert objekt er beskrevet ved en identifikator, hvilken klasse det tilhører og hvilke regioner det har beveget seg gjennom.

Tabell 9.1: Lagrede data

- Høyden til rektangelet i antall piksler.
- Bredden til rektangelet i antall piksler.

9.3.2 Krysset

Krysset er den delen av interesseområdet brukeren er interessert i å hente data ut ifra. Selv om hele det rektangulære interesseområdet prosesseres, er det kun observasjoner i utsnittet som beskriver krysset man er interessert å lagre informasjon ifra. Et kryss defineres som et sett regioner. Hver region er beskrevet ved et polygon og merkes med en av fire himmelretninger eller som midtregion. Ikke alle himmelretninger trenger å være representert. Midtregionen eksisterer alltid. Regionene lagres som `<region>`-elementer. For hver region lagres:

- Identifikator. En av tekststrengene “north”, “south”, “east”, “west” eller “center”.
- Polygonet som beskriver regionen. Dette består av minst tre ordnede punkter i interesseområdekoordinater.

9.3.3 Bilder

Her lagres en liste med informasjon om hvert enkelt bilde i sekvensen. For hvert bilde lagres en beskrivelse av forgrunnsobjektene og hvilke regioner objektene tilhører. Denne informasjonen lagres i `<frame>`-elementer bestående av:

- Et sekvensnummer.
- Filnavnet til bildefilen.
- For alle regioner med minst ett objekt lagres:
  - Regionens identifikator.

  - For hvert objekt i regionen lagres:
    * Objektets identifikator.
    * Identifikatoren til den neste regionen objektet drar gjennom ("out" dersom objektet er på vei ut av krysset, eller "none" dersom objektet står i ro).
    * Objektets klasse.

9.3.4 Objekter

Her lagres en liste med informasjon om alle objektene i videosekvensen. Et objekt er enten et kjøretøy eller en fotgjenger med en vellykket sti gjennom krysset. Objekter lagres som `<track>`-elementer bestående av:

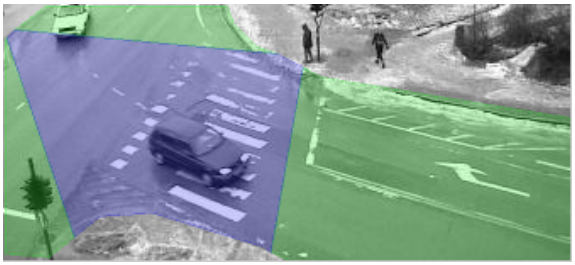
- Et sekvensnummer.
- Objektets klasse. Enten "vehicle" eller "pedestrian".
- For hver av regionene objektet har beveget seg gjennom lagres:
  - Regionens identifikator.
  - Hvilket bilde objektet kom inn i regionen.
  - Hvilket bilde objektet forlot regionen.

For hvert bilde underveis i prosesseringen sender følgingsmodulen alle objektene den har oppdaget til datalagringsmodulen. Når alle bildene er prosessert sender følgingsmodulen en liste over alle vellykkede spor. Disse skrives som `<track>`-elementer. Datalagringsmodulen fjerner ufullstendig fulgte objekter fra `<frame>`-elementene når den får de endelige objektene fra følgingsmodulen. Følgelig vil et objekt som ikke eksisterer som et `<track>`-element, heller ikke være til stede i et `<frame>`-element.

9.4 Resultater

Den komplette XSD-en som beskriver grammatikken for XML-dokumenter i X-Analyzer finnes i vedlegg C.1. De genererte XML-dokumentene fra begge testsettene samsvarer med denne definisjonen. Begge testsettene genererte i overkant av 53000 linjer XML.

Siden disse dokumentene er for store til å presentere i denne rapporten, har vi inkludert et eksempel på hvordan et bilde representeres som et `<frame>`-element, og hvordan objektene i dette bildet representeres som `<track>`-elementer. Bildet som skal representeres som XML er vist i figur 9.3. Figuren viser et utsnitt (interesseområde) fra et bilde (sekvensnummer 1219) i testsettet Samfundet.



Figur 9.3: Bilderepresentasjon med kryssdefinisjon

I figur 9.3 er kryssdefinisjonen inntegnet. Den blå regionen representerer midtregionen. De grønne regionene på sidene er merket som nordlig, sørlig og østlig region. Innenfor regionene som avgrenser krysset befinner det seg to biler. XML-koden under beskriver situasjonen i dette bildet ved et `<frame>`-element.

XML-representasjon av et bilde

```
<frame>
  <seq>1219</seq>
  <filename>00001349.jpg</filename>
  <regions>
    <region>
      <id>north</id>
      <tracks>
        <track>
          <id>1948</id>
          <heading>center</heading>
          <class>vehicle</class>
        </track>
      </tracks>
    </region>
    <region>
      <id>center</id>
      <tracks>
        <track>
          <id>1947</id>
          <heading>east</heading>
          <class>vehicle</class>
        </track>
      </tracks>
    </region>
  </regions>
</frame>
```

I nordlig region er en bil (ID 1948) på vei mot midtregionen. I midtregionen er en bil med (ID 1947) på vei østover. Fotgjengerne i øvre del av bildet er ikke representert som XML siden de befinner seg utenfor kryssdefinisjonen. Dersom en fotgjenger hadde befunnet seg innenfor en av regionene som definerer krysset, ville denne fått en lik representasjon med unntak av klassen.

XML-koden på forrige side refererer til to objekter med (ID 1947 og 1948). Disse objektene finner vi igjen lenger ned i XML-dokumentet, representert ved `<track>`-elementer. Disse elementene er gjengitt under.

XML-representasjon av objekt 1947

```
<track>
  <id>1947</id>
  <class>vehicle</class>
  <regions>
    <region>
      <id>north</id>
      <entry>1202</entry>
      <exit>1207</exit>
    </region>
    <region>
      <id>center</id>
      <entry>1208</entry>
      <exit>1221</exit>
    </region>
    <region>
      <id>east</id>
      <entry>1222</entry>
      <exit>1226</exit>
    </region>
  </regions>
</track>
```

XML-representasjon av objekt 1948

```
<track>
  <id>1948</id>
  <class>vehicle</class>
  <regions>
    <region>
      <id>north</id>
      <entry>1216</entry>
      <exit>1219</exit>
    </region>
    <region>
      <id>center</id>
      <entry>1220</entry>
      <exit>1230</exit>
    </region>
    <region>
      <id>east</id>
      <entry>1231</entry>
      <exit>1235</exit>
    </region>
  </regions>
</track>
```

Den ene bilen (ID 1947) har begynt i nordlig region (bilde 1202 – 1207), beveget seg til midtregionen der den befinner seg i figur 9.3 (bilde 1208 –1221) og over i østlig region (bilde 1222 – 1226).

Den andre bilen (ID 1948) har beveget seg fra nordlig region (bilde 1216 – 1219), der den befinner seg i figur 9.3, til midtregionen (bilde 1220 – 1230) og over i østlig region (bilde 1231 – 1235).

9.5 Konklusjon

Vi har implementert en datalagringsmodul som ut ifra data fra følgingsmodulen, genererer en XML-representasjon av aktiviteten i krysset. Selve krysset lagres som et sett regioner. For hvert bilde lagres en representasjon av aktivitet i de ulike regionene som utgjør krysset. For hvert objekt lagres informasjon om klasse og hvordan objektet har beveget seg gjennom krysset.

Lagringen er bevisst redundant. Siden XPath ikke har støtte for joins, gjør dette spørringene enklere. Et alternativ til å lagre redundante data er å benytte transformasjoner av XML for ulik representasjon. Siden de genererte XML-dokumentene ikke er ment å oppdateres, har ikke redundansen negative konsekvenser bortsett fra økt datavolum.

Kapittel 10

Brukergrensesnitt

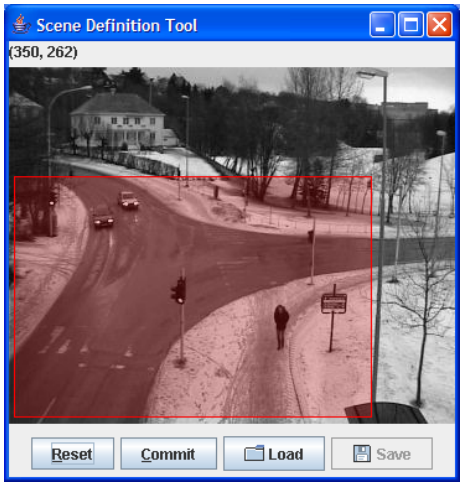
Vi har utviklet tre brukergrensesnitt. For X-Analyzer har vi utviklet Scene Definition Tool og X-Monitor. Scene Definition Tool gir brukeren mulighet til å spesifisere geometriske egenskaper ved scenen som skal analyseres. I X-Monitor kan man følge prosesseringen av bildene. X-Retriever benyttes til å formulere spørringer knyttet til aktivitet i scenen.

10.1 X-Analyzer

I dette delkapittelet presenteres de to brukergrensesnittene i X-Analyzer.

10.1.1 Scene Definition Tool

I Scene Definition Tool (SDT) kan brukeren formulere en scenedefinisjon. Scenedefinisjonen sier noe om hvordan X-Analyzer skal mappe koordinater i videosekvensen til XML. En scenedefinisjon består av et klippevindu, kryssgeometri og regioninndeling. En scenedefinisjon kan lagres som et eget XML-dokument og kan lastes inn ved en senere anledning.



Figur 10.1: Interesseområde

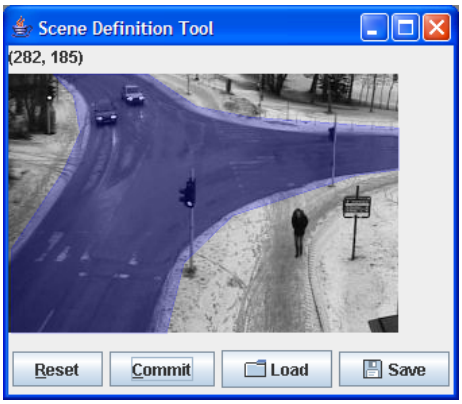
Interesseområde

Spesifikasjon av interesseområde (region of interest) er det første steget i en scenedefinisjon. Før prosesseringen kan begynne, klippes hvert bilde til et rektangulært interesseområde. Fornuftig valg av interesseområde reduserer beregningene i påfølgende prosesseringssteg, siden store deler av datasettet som regel faller utenfor interesseområdet. Et velplassert interesseområde vil dessuten være med på gi mer korrekte resultater. Dersom det plasseres slik at objekter langt borte klippes

vekk, vil perspektivproblemene i videosekvensen reduseres. Et interesseområde spesifiseres ved å merke et rektangulært område med musen. Det røde transparente rektangelet i figur 10.1 viser en seleksjon i SDT. Seleksjonen bekreftes eller kanselleres ved henholdsvis "Commit"- eller "Reset"-knappen.

Kryssgeometri

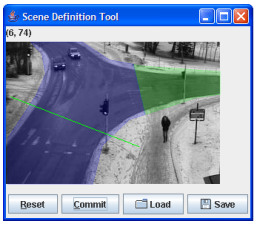
Når interesseområdet er bekreftet må kryssets geometri spesifiseres. Årsaken til dette er at vi kun vil registrere objektbevegelse i visse deler av scenen. Selv om hele bildet prosesseres, er vi ikke interessert i å registrere bevegelse utenfor selve krysset. Kun aktivitet som faller innenfor polygonet vi har spesifisert som kryss, vil bli skrevet til XML. Et kryss i denne sammenhengen er et polygon som brukeren selv definerer med musen. Kryssgeometriseleksjonen tegnes i transparent blått som vist figur 10.2. Seleksjonen bekreftes eller kanselleres ved bhenholdsvis "Commit"- eller "Reset"-knappen.



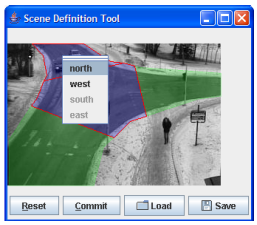
Figur 10.2: Kryssgeometri

Regioninndeling

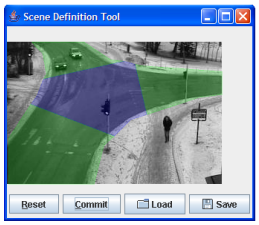
Når polygonet som definerer kryssets geometri er spesifisert, må brukeren dele dette inn i regioner. Disse regionene danner utgangspunktet for hvordan objekter og stier skrives til XML. En region defineres ved at brukeren slår en linje med musen, som deler krysspolygonet i to deler. En slik linje er vist i figur 10.3(a). Når en linjeseleksjon som splitter krysset er blitt gjort, tegnes splitten som to røde omriss. Brukeren kan så markere en av de to regionene med musen. Dette er vist i figur 10.3(b). En meny gir da mulighet til å sette en merkelapp på regionen. Denne merkelappen blir siden brukt som verdi for regionelementer i XML-representasjonen (kapittel 9). Vi har begrenset oss til at et kryss maksimalt kan deles i 4 regioner ("north", "south", "east", "west") samt én midtregion ("center"). Den umerkede regionen tas automatisk for å være midtregion. Figur 10.3(c) viser en fullstendig regioninndeling. Regionene med tilordnet merkelapp er representert ved transparente grønne polygon. Midtregionen er tegnet i transparent blått.



(a) Linjeseleksjon



(b) Regionseleksjon



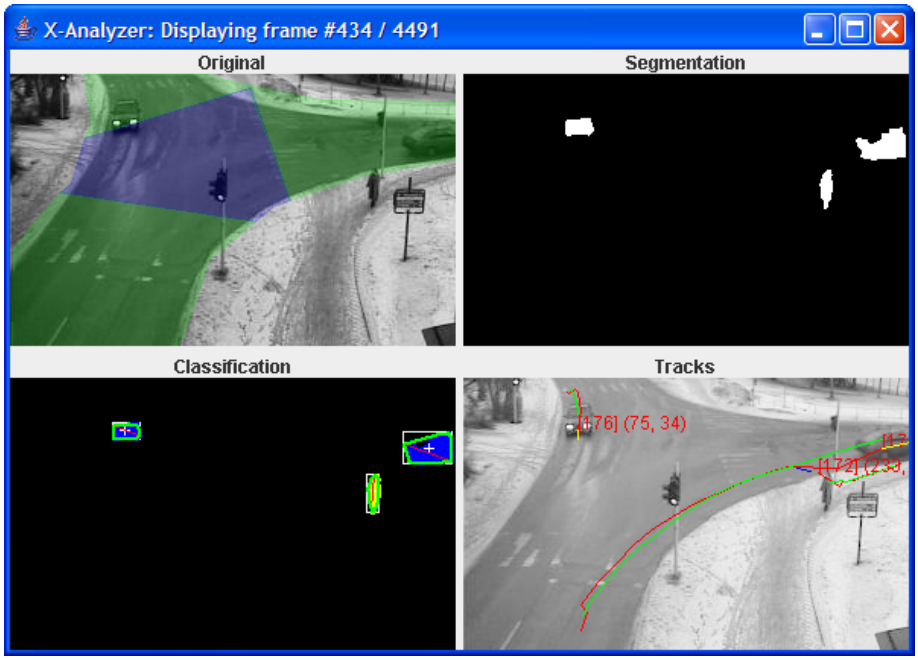
(c) Resultat

Figur 10.3: Regioninndeling

Regioninndelingen bekreftes ved å trykke "Commit"-knappen. Dette starter prosesseringen i X-Analyzer og åpner X-Monitor.

10.1.2 X-Monitor

X-Monitor lar brukeren følge resultatene fra de ulike modulene i prosesseringen. Figur 10.4 viser et skjermbilde. Øverst til venstre er det opprinnelige bildet med scenedefinisjonen inntegnet. Til høyre for dette er resultatet av segmentering og skyggefjerning (kapittel 5 og kapittel 6). Alle hvite objekter i dette bildet er identifisert som sanne forgrunnsobjekter. Nede til venstre er resultatet fra gjenkjenningsmodulen (kapittel 7) illustrert. Objektenes farge viser den tentative klassifiseringen i gjeldene bilde. Blått objekt representerer kjøretøy mens gult objekt representerer fotgjenger. Videre er objektenes massesenter markert med hvite trådkors, hovedaksen (eng.: *major axis*) som en rød linje og den segmenterte regionens konvekse hull i grønt. Nede til høyre er resultatene fra følgingsmodulen (kapittel 8). Hvert objekt med et vellykket spor har påtegnet en ID og (x, y) koordinater. Objekter uten ID er ikke fanget opp av følgingsmodulen. Stier gjennom scenen er tegnet i rødt og midlere sti er tegnet i grønt. Blå stier indikerer stipredikteringer for tapte objekter. Den lille gule streken på hvert objekt representerer objektets hastighetsvektor.



Figur 10.4: Skjermbilde X-Monitor

Brukeren kan når som helst pause prosesseringen, ved å trykke *P*. Ved å benytte høyre piltast i pausemodus kan brukeren hoppe ett og ett bilde frem i sekvensen. Hvilken som helst annen tast gjenopptar vanlig prosessering.

10.2 X-Retriever

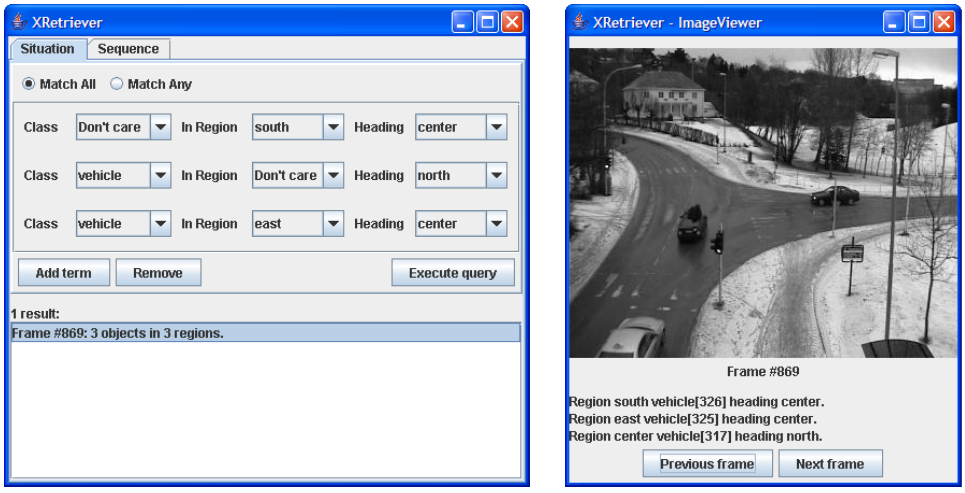
X-Retriever er et spørresystem for å gjenfinne situasjoner i XML-representasjonen fra analyserte kryss. X-Retriever er basert på XPath-spørringer. Interessante situasjoner og sekvenser formuleres grafisk som spørringer. Disse returnerer `<frame>`-elementer som beskriver de matchende bildene i sekvensen.

10.2.1 Situasjonsspørringer

En situasjon er en statisk konfigurasjon av objekter i scenen. Situasjonsspørringer returnerer matchende `<frame>`-elementer fra XML-representasjonen. Figur 10.5(a) viser spørrevinduet for situasjonsspørringer. Hvert objekt i en situasjon er beskrevet ved klasse, region og retning:

- Klasse sier hva slags objekt vi ønsker å finne (fotgjenger/kjøretøy).
- Region er en av regionene definert i SDT (se avsnitt 10.1.1).
- Retningen forteller hvor i scenen objektet er på vei. Dette kan enten være en region, "none" (objektet står stille) eller "out" (objektet er på vei ut av scenen).

Verdier for feltene velges fra rullegardinmenyene i brukergrensesnittet. Menyene initialiseres med verdier fra de respektive feltene i XML-representasjonen. I tillegg legges det i hver meny til en jokeropsjon ("Don't care") som, hvis valgt, medfører at feltet ikke blir tatt hensyn til i spørringen.



(a) Spørring (b) Resultat

Figur 10.5: Situasjonsspørring og resultat

En spørring, Q, består av et vilkårlig antall termer, q_i. Termer kan legges til eller fjernes fra spørringen med knappene "Add term" og "Remove". Spørringen definert i figur 10.5(a), består av tre termer:

- q_1: Vilkårlig objekt ("Don't care") i sørlig region ("south") på vei mot midtregion ("center").

- q_2: Kjøretøy ("vehicle") i vilkårlig region ("Don't care") på vei nordover ("north").
- q_3: Kjøretøy ("vehicle") i østlig region ("east") på vei mot midtregion ("center").

Når "Execute query" trykkes, vises resultatene i resultatlisten i bunnen av vinduet. I figur 10.5(a) gir spørringen kun ett resultat. Hvert av elementene i resultatlisten kan åpnes for nærmere inspeksjon ved å dobbelklikke i listen. Man får da opp et nytt vindu som viser det aktuelle bildet i sekvensen samt en beskrivelse av situasjonen i scenen. Resultatet av spørringen i figur 10.5(a) er vist i figur 10.5(b). Brukeren kan hoppe frem og tilbake i datasettet fra gjeldende posisjon med knappene "Next frame" og "'Previous frame".

Termer kombineres i valgt kombinasjonsmodus. Kombinasjonsmodus velges ved hjelp av opsjonsknappene øverst i brukergrensesnittet. I kombinasjonsmodus "ANY" dannes Q som unionen av spørringstermene:

$$Q = q_1 \cup q_2 \cup \ldots \cup q_n \tag{10.1}$$

I kombinasjonsmodus "ALL" dannes Q som snittet av spørringstermene:

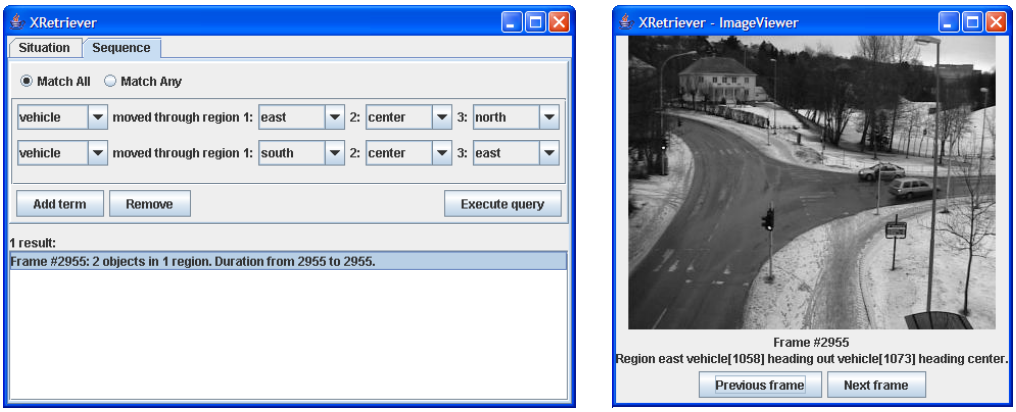
$$Q = q_1 \cap q_2 \cap \ldots \cap q_n. \tag{10.2}$$

10.2.2 Sekvenspørringer

En sekvens består av et objekt og opp til tre regioner dette objektet har passert gjennom (i sekvens). Brukeren kan spesifisere objektklasse (vehicle/pedestrian) og hver av de tre regionene (north/south/east/west/center). Også her kan man sette jokeropsjon ("Don't care") for en eller flere av regionene. Sekvensspørringene tar utgangspunkt i `<track>`-elementene i XML-representasjonen. Resultatet av en sekvensspørring er `<frame>`-elementene som matcher `<track>`-elementene. Figur 10.6(a) viser en sekvensspørring bestående av to termer i kombinasjonsmodus ALL:

- q_1: Kjøretøy ("vehicle") med sti gjennom region øst, midtregion og nord
- q_2: Kjøretøy ("vehicle") med sti gjennom region sør, midtregion og øst

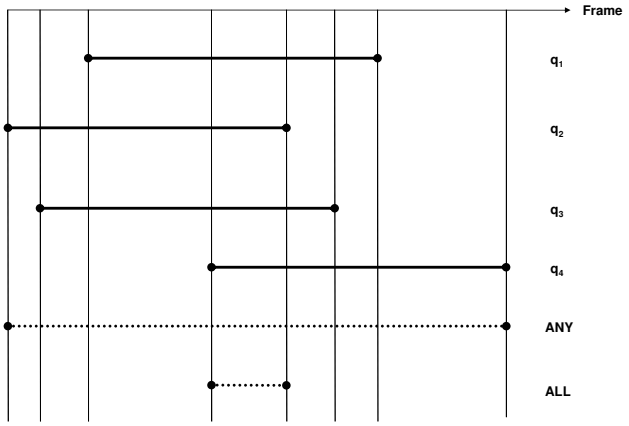
På samme måte som for situasjonspørringer kommer resultatene opp i resultatlisten nederst i brukergrensesnittet. Resultater fra sekvensspørringer har varighet (eng.: *duration*) knyttet til seg. Varigheten sier noe om hvilket intervall av datasettet som matcher spørringen. Denne spørringen ga kun ett resultat med 1 bilde varighet (bilde 2955 – 2955). Resultatet er nærmere illustrert i figur 10.6(b). I dette bildet svarer bilen som er på vei inn i scenen fra høyre til den første spørringstermen, q_1. Bilen som er på vei ut av scenen til høyre svarer til q_2.



(a) Spørring (b) Resultat

Figur 10.6: Sekvensspørring og resultat

I kombinasjonsmodus "ANY" returneres alle `<frame>`-elementer som overlapper med minst én av termene. I kombinasjonsmodus "ALL" returneres det utsnittet av `<frame>`-elementer som overlapper alle termene. Siden sekvensspørringer har en temporal dimensjon, må kombinasjonsmodus "ALL" eller "ANY" tolkes noe annerledes enn for situasjonsspørringer. Figur 10.7 illustrerer dette. Her vises varigheten av resultater fra fire termer, q_1, q_2, q_3, q_4, langs en tidslinje. Nederst vises varigheten av resultatet ved kombinasjonsmodus "ANY" og "ALL".



Figur 10.7: Kombinasjon av sekvenstermer

Kapittel 11

Evaluering

I dette kapittelet evalueres implementasjonen. Vi diskuterer resultater, anvendbarhet og fremtidige utvidelser.

Det ville vært naturlig å sammenligne våre resultater med resultater fra lignende systemer. At det ikke finnes lignende systemer, gjør evalueringen vanskelig. Siden registrering av svingemønster idag gjøres manuelt, er vårt sammenligningsgrunnlag data innhentet ved manuell registrering av de faktiske hendelsene i testsettene.

11.1 Diskusjon

Dette delkapittelet diskuterer resultatene fra hver prosesseringsmodul i X-Analyzer og gjenfinningssystemet X-Retriever. Til slutt diskuteres problemer knyttet til å registrere fotgjengeres bevegelse.

Med utgangspunkt i målet for oppgaven er resultatene fra både X-Analyzer og X-Retriever gode. Målet var todelt:

1. Å implementere en prototyp som ut i fra videodata fra et ukalibrert kamera genererer informasjon om forgrunnsobjekter i en trafikksituasjon.
2. Å implementere en prototyp for å gjenfinne interessante situasjoner i de analyserte videoklippene.

Systemet skulle implementeres fra bunnen av og kjøre på vanlige PC-er. Vår oppgave var definert uten konkrete kvantitative krav til løsningen. Oppgaven spenner vidt. Siden vi anså selve ideen om et komplett system for videoanalyse av trafikk å være det unike, valgte vi heller å fokusere bredt enn dypt. Vi forsøkte å implementere så mye funksjonalitet som mulig fra den initielle oppgavebeskrivelsen, istedenfor å gå dypt inn i noen av delfunksjonene i et slikt system. Likevel har segmentering og følging fått mest oppmerksomhet siden denne funksjonaliteten er mest kritisk for en slik applikasjon.

Kapitlene 5–9 dokumenterer jevnt over gode resultater fra X-Analyzer. X-Retriever (beskrevet i kapittel 10) kan benyttes til å gjenfinne sekvenser av objektbevegelse og situasjoner. Mye av

funksjonaliteten i X-Retriever erstatter manuell analyse av data fra trafikktellinger. Prototypen kan enkelt utvides og gjøres mer uttrykksfull.

11.1.1 Datainnsamling

To valg ble tatt i forbindelse med innsamling av videodata. Vi valgte å bruke data fra kun én kameravinkel fremfor å korrelere data fra flere kameravinkler. Dette har konsekvenser først og fremst i forbindelse med følging. Vi bestemte også å bruke et konvensjonelt digitalt videokamera framfor å samle data med et kamera som registrerer infrarøde bølger. Dette fikk konsekvenser for segmenteringen.

11.1.2 Segmentering

I segmenteringsmodulen modellerte vi bakgrunnen som unimodale sannsynlighetsfordelinger for hver piksellokasjon. Bakgrunnsmodellen er adaptiv med hensyn på endringer. Segmenteringen viste seg å fungere godt. Metoden har noen problemer med refleksjoner i asfalten og gjenferd. Okklusjonsproblemer fører iblant til at flere objekter segmenteres som samme region. Videre registrerer vi enkelte diskontinuiteter i objekter med pikselverdier tilnærmet lik bakgrunnsmodellen.

Data registrert gjennom infrarøde (IR) sensorer ville sannsynligvis gjort segmenteringen lettere. Motoriserte kjøretøy avgir varme i form av infrarød stråling som kan registreres. Caspi matcher i [3] videosekvenser fra et konvensjonelt videokamera og et infrarødt kamera. Deretter segmenteres resultatet ved bruk av bakgrunnssubtraksjon. Objekter med farge tilnærmet lik bakgrunnen er lettere å segmentere med bakgrunn i IR-data [3]. Et åpenbart problem ved bruk av IR er varm asfalt.

11.1.3 Skyggefjerning

For skyggefjerning benyttet vi en metode basert på kromatografi. Metoden identifiserer skyggepiksler i settet av forgrunnspiksler fra segmenteringsmodulen. En svakhet med metoden er at den kun fungerer for fargebilder. Metoden kan utvides ved å ta hensyn til sannsynligheten for at nabopiksler representerer skygge.

Det rapporteres i [29] om gode resultater ved bruk av aktive konturer for skyggefjerning. Vi eksperimenterte med dette uten særlig hell. Det er nærliggende å tro at vårt forsøk mislyktes som følge av feil i implementasjonen. Vi innså at en velfungerende implementasjon av aktive konturer kom til å bli uforholdsmessig ressurskrevende. Siden målet for oppgaven var å berøre alle aspektene ved et system for automatisk trafikkovervåkning, valgte vi å ikke gå videre med skyggefjerning gjennom aktive konturer.

11.1.4 Objektgjenkjenning

Forgrunnsobjekter klassifiseres enten som kjøretøy (harde trafikanter), fotgjengere (myke trafikanter) eller støy. Det kunne vært nyttig å skille mellom ulike typer kjøretøy (busser, lastebiler,

personbiler, etc.) og klasser av myke trafikanter (syklister, fotgjengere, etc). Vi valgte en enklere inndeling siden objekter kun diskrimineres ved fasong. Vår inndeling er likevel nyttig siden den definerer et skille mellom harde og myke trafikanter. De andre objektklassene er, fra et trafikkteknisk synspunkt, undergrupper av enten harde eller myke trafikanter. Det er dessuten et tydelig skille i hvilke trafikkregler som gjelder for disse to hovedgruppene.

Utgangspunktet for objektgjenkjenningen var enkel representasjon og tentativ klassifisering av hvert objekt i hvert bilde. Representasjonen var basert på objektenes fasong og deres konvekse hull. Den tentative klassifiseringen for hver region ble akkumulert temporalt på objektbasis. Den endelige beslutningen om et objekts klasse ble derfor tatt med utgangspunkt i gjentatte observasjoner av objektet.

Til tross for sin enkelhet fikk objektgjenkjenningsmodulen gode resultater for kjøretøy. Resultatene var ikke fullt så gode for fotgjengere.

11.1.5 Følging

Følgingen er basert på en antakelse om overlapp i regioner som representerer samme objekt i etterfølgende bilder. Metoden er følsom mot okklusjon og objekter som kommer nær hverandre. Disse situasjonene behandles gjennom estimater av objekters lokasjon. Dersom vi hadde benyttet optisk flyt for segmentering, kunne vektorfeltet til en segmentert region blitt benyttet til å estimere neste objektlokasjon mer nøyaktig. Til tross for metodens enkelhet, fikk vi gode resultater av følgingsmodulen.

At følgingen er basert på data fra kun én kameravinkel er også en grunn til at påliteligheten til følgingsmodulen ikke er bedre. Okklusjon er det største problemet. Hadde vi korrelert data fra flere kameraposisjoner, kunne dette problemet vært løst på bekostning av høyere kompleksitet.

11.1.6 Datalagring

Datalagringsmodulen genererer en XML-representasjon av aktiviteten i krysset. Data lagres både på bildebasis (hvilke objekter som er i hvilken region i et gitt bilde), og på objektbasis (hvilke bilder et objekt befant seg i hvilke regioner). Redundansen i XML-representasjonen var nødvendig for enkel implementasjon av X-Path spørringer igjennom X-Retriever. Krysspesifikasjonen i form av regioner lagres også. Vi benyttet et DOM API (JDOM) til å bygge en XML-representasjon av aktiviteten i scenen. XML-representasjonen av datasettene ble validert mot en XSD som definerte dokumentenes grammatikk. Hvordan datalagringsmodulen strukturerer informasjonen, har konsekvenser for X-Retriever.

11.1.7 X-Retriever

X-Retriever muliggjør grafisk utforming og eksekvering av spørringer. X-Retriever demonstrerer nytteverdien av automatisk trafikkanalyse. Fleksibiliteten i spørringstypene er noe begrenset i vår prototype. Vi identifiserte og implementerte støtte for to typer spørringer. Situasjonsspørringer spesifiserer konfigurasjoner av objekter, deres plassering i krysset og retning. En situasjonsspørring kan dermed definere konfigurasjoner som for eksempel beskriver brudd på høyreregel.

Sekvensspørringer spesifiserer forflytning gjennom regionene som definerer krysset. På denne måten kan en sekvensspørring returnere alle bilder av objekter med et bestemt svingemønster.

X-Retriever implementerer spørringene i form av XPath-uttrykk. XPath har ikke støtte for joins. Dette er årsaken til redundans XML-representasjonen.

11.1.8 Fotgjengere

Vi valgte tidlig å fokusere på deteksjon og følging av kjøretøy fremfor fotgjengere. Vårt fokus på kjøretøy gir seg utslag i hvordan vi har definert krysset i våre eksempler. Krysset er definert til å omfatte veibane og ikke fortau.

Systemet har støtte for deteksjon og følging av fotgjengere. Denne funksjonaliteten er dessverre ikke tilfredsstillende. Fotgjengere har ofte for liten representasjon til at de blir korrekt segmentert. I tillegg tar objektgjenkjenningsmodulen oftere feil i klassifisering av fotgjengere enn i klassifiseringen av kjøretøy.

Mye relevant informasjon kan trekkes ut fra fotgjengeres atferd og deres interaksjon med kjøretøy i trafikken. Derfor er bedre segmentering, følging og gjenkjenning av fotgjengere et sentralt utvidelsespunkt.

11.2 Anvendbarhet

Vårt system kan benyttes til å analysere videosekvenser fra vilkårlige veikryss og vanlige veistrekninger. En veistrekning kan sees på som en generalisering av et veikryss med kun to tilfarter. Å overvåke en veistrekning rett ovenifra for å kartlegge avviklingskvalitet, belegg og metning i hver retning på ulike tidspunkt er en enkel men nyttig anvendelse av systemet.

Selv om systemet fungerer tilfredsstillende for våre testsett, ville det fungert enda bedre dersom vi hadde fått filmet rett ovenifra. På denne måten ville okklusjonsproblemene blitt redusert. Dette ville gitt mer korrekt segmentering og dermed også mer pålitelige følgingsresultater. Vi registrerer at stier går tapt ettersom vår implementasjon ikke greier å følge alle objekter i tungt trafikkerte kryss. Påliteligheten og dermed anvendbarheten svekkes altså proporsjonalt med hvor mye trafikk det er i krysset som overvåkes.

Ettersom vår implementasjon ikke er utviklet for sanntidsprosessering, vil ikke systemet egne seg for kontinuerlig overvåkning. Det egner seg imidlertid godt for i ettertid å analysere overvåkningssekvenser. Analysen er nyttig i forbindelse med kartlegging av trafikkparametre som avvikling, trafikkvalitet, svingemønster og konfliktsituasjoner. Disse parametrene er vesentlige i forbindelse med utbygging av veianlegg.

Systemet kan også benyttes for å håndheve myndigheters regelverk. Det er enkelt å identifisere atferd i strid med trafikkreglement som for eksempel sving mot forbudsskilt, påbudt stopp, feil kjøreretning i enveiskjørte gater etc. Med enkle utvidelser kan dessuten systemet anslå hastighet til objekter i scenen.

11.3 Utvidelser

Vi har identifisert en rekke områder der implementasjonen kan forbedres. Dette delkapittelet oppsummerer det vi anser som de viktigste utvidelsene til de nåværende prototypene.

I tillegg til utvidelsene nevnt her, er forbedring av prosesseringsmodulene i X-Analyzer essensielt. Vi understreker at formålet med oppgaven var å rekke over så mye funksjonalitet som mulig uten å gå altfor mye i dybden i hvert prosesseringsteg. Siden vi ikke har hatt ressurser til å eksperimentere med ulike metoder for hver modul, ligger det åpenbart et ukjent forbedringspotensiale i å velge andre algoritmer og metoder enn de vi har valgt.

11.3.1 Kalibrering

En naturlig utvidelse er kalibrering av kameraet. Med kalibrering forstås at man registrerer kameraets koordinater (tredimensjonalt) i forhold til scenen. Ved å ta disse med i beregningene og i tillegg benytte et sett referansekoordinater, vil man kunne transformere todimensjonale pikselkoordinater til tredimensjonale verdenskoordinater. Med denne informasjonen vil man kunne registrere mer nyttig informasjon om aktivitet i scenen.

Hastighet. Ved å beregne et objekts posisjon i verdenskoordinater i to etterfølgende bilder, kan man enkelt kalkulere objektets hastighet. På denne måten kan effektiviteten og sikkerheten i et veikryss evalueres med hensyn på objekters hastighet. Dette kan også benyttes til å håndheve regelverk. Hastighet kan beregnes i den nåværende implementasjonen ved å måle tiden et objekt bruker på å forflytte seg mellom referansepunkter i krysset med kjent avstand.

Avstand. Avstanden mellom objekter (tidsluker) er nyttig for å evaluere avviklingskvaliteten. Tidsluker kan benyttes til å kvantifisere avviklingen i et veikryss. Med utgangspunkt i hvordan en gjennomsnittlig trafikant utnytter tidsluker, kan tiltak iverksettes for å bedre avviklingen. Avstanden mellom objekter kan også benyttes for håndheving av trafikkregler som definerer minste tillatte avstand mellom etterfølgende biler.

Objektstørrelse. Gjennom normalisering av koordinater kan man estimere objekters reelle størrelse uavhengig av perspektiv. Normalisert objektstørrelse vil gi en mer korrekt klassifisering siden fotgjengere er representert ved mindre objekter enn kjøretøy.

Segmentering Med bakgrunn i ideen om normalisert objektstørrelse vil segmenteringen kunne dra nytte av et kalibrert kamera. Segmenteringen benytter en statisk terskelverdi for hvor store regioner må være for å klassifisere som forgrunn. Ved å benytte en dynamisk terskelverdi normalisert i forhold til hvor i bildet regionen befinner seg, vil segmenteringen kunne gi mer nøyaktige resultater.

11.3.2 Kjennemerker

En innlysende utvidelse i X-Analyzer er å implementere gjenkjenning av kjennemerker (registreringsnummer). Ved å overvåke flere veistrekninger samtidig, kan dette benyttes til å innhente statistikk om kjøremønster. Man kan dermed kartlegge bilers bevegelser over større områder.

I dag benyttes elektroniske sendere (mobiltelefoner, autopassbrikker, etc.) i kjøretøy og basestasjoner langs veien til dette.

11.3.3 Fotgjengere

Fotgjengere bør kunne segmenteres, gjenkjennes og følges med større pålitelighet. Da vil man også kunne analysere konfliktsituasjoner mellom kjøretøy og fotgjengere bedre. Tidsluker mellom fotgjengere er dessuten interessant fra et trafikkteknisk ståsted.

11.3.4 Datainnsamling

Ved å utvide X-Analyzer til å korrelere video fra synkrone kamera, vil man kunne eliminere okklusjonsproblemer. Dette vil gi bedre estimater på objekters posisjon, størrelse og fasong. Høyere oppløsning i bildene kan benyttes ved analyse av data over et større område. Man kan også korrelere data fra forskjellige sensorer (f.eks. konvensjonelle videokamera, infrarødt kamera, induktive sløyfer, radar, laser, elektroniske sendere).

11.3.5 X-Retriever

Gjenfinningsapplikasjonen kan utvides på en rekke områder. En åpenbar utvidelse er å tillate kombinasjoner av situasjons- og sekvensspørringer. På denne måten kan man definere mer uttrykksfulle spørringer som kombinerer statiske konfigurasjoner med dynamiske sekvenser.

Spørringstermer kan kombineres i kun én kombinasjonsmodus ("All" eller "Any"). Definisjon av et spørretre med vilkårlige kombinasjoner av spørringer med ulik kombinasjonsmodus vil være interessant. Dette kan implementeres ved rekursive spørringer som består av spørringer i seg selv og ikke bare spørringstermer.

I den nåværende implementasjonen må brukeren selv formulere spørringen som definerer en konflikt. Det kan også implementeres støtte for automatisk deteksjon av spesifikke atferd- og konfliktmønstre.

Kapittel 12

Konklusjon

Analyse av trafikk er nyttig i forbindelse med kartlegging av trafikkparametre som belegg, metning og svingemønster. Disse parametrene er vesentlige i forbindelse med utbygging av veianlegg. Informasjonen kan benyttes for å iverksette tiltak som bedrer sikkerhet og avviklingskvalitet. Mye datainnsamling og analysearbeid gjøres idag manuelt. Kostnadseffektive metoder for automatisk analyse er derfor viktig.

Vi har sett på hvordan et system for automatisk trafikkovervåkning kan implementeres. Med bakgrunn i kjente metoder fra litteraturen, har vi implementert en modulbasert prototyp for videoanalyse av vilkårlige veistrekninger. Brukeren definerer selv en scene i form av interessante regioner. Scenedefinisjonen danner grunnlaget for analysen. Vi registrerer i analysen at noen stier går tapt ettersom vår implementasjon ikke greier å følge alle objekter i tungt trafikkerte kryss. Påliteligheten og dermed anvendbarheten svekkes altså proporsjonalt med hvor mye trafikk det er i krysset som overvåkes. Dette er åpenbart en svakhet.

Gjennom X-Retriever har vi implementert et gjenfinningsverktøy for å finne igjen interessant informasjon i de uttrekte dataene. X-Retriever er et intuitivt brukergrensesnitt for å formulere spørringer knyttet til statiske situasjoner og dynamiske sekvenser. Fremtidige utvidelser av X-Retriever vil kunne styrke uttrykkskraften.

Vi mener vi har lykkes med å implementere idéen presentert i oppgavebeskrivelsen. Vår prototyp for automatisk analyse opererer effektivt og implementerer all funksjonaliteten i vår målsetning. Fotgjengere kunne vært behandlet bedre, men god støtte for fotgjengere har ikke vært vårt primære mål. Våre prototyper har nytteverdi i kraft av at de muliggjør automatisk deteksjon av situasjoner som kan representere konflikter. Systemet kan også benyttes for å håndheve regelverk knyttet til atferd i trafikken.

Vi har også identifisert en rekke punkter hvor vår implementasjon kan utvides. Vi konkluderer derfor med at implementasjonen var vellykket, men ytterligere testing og videre utvikling vil være nødvendig dersom systemet skal benyttes i praksis.

Referanser

[1] S. S. Beauchemin og J. L. Barron. The computation of optical flow. *ACM Comput. Surv.*, 27(3):433–466, 1995.

[2] Lisa M. Brown. View independent vehicle/person classification. I *VSSN '04: Proceedings of the ACM 2nd international workshop on Video surveillance & sensor networks*, side 114–123. ACM Press, 2004.

[3] Yaron Caspi, Denis Simakov og Michal Irani. Feature-Based Sequence-to-Sequence Matching. http://www.wisdom.weizmann.ac.il/∼vision/VideoAnalysis/Demos/Traj2Traj/ir_pal.htm, May 2002. Example: Multi-sensor alignment (infra-red vs. visible).

[4] Centeye. Optic Flow, Mars 2005. http://www.centeye.com/pages/techres/opticflow.html.

[5] Thomas H. Cormen, Clifford Stein, Ronald L. Rivest og Charles E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2001.

[6] Mary Fernandez og Paul Cotton. XPath-XQuery Review. http://www.w3.org/2002/Talks/www2002-xpath-xquery/, May 2002. Presentation held at 11^{th} International World Wide Web Conference (WWW2002) at Hawaii.

[7] Nir Friedman og Stuart J. Russell. Image Segmentation in Video Sequences: A Probabilistic Approach. I *UAI*, side 175–181, 1997.

[8] Rafael C. Gonzales og Richard E. Woods. *Digital Image Processing*. Prentice-Hall, Inc., andre utgave, 2002.

[9] Kyung-Ah Han og Sung-Hyun Myaeng. Image organization and retrieval with automatically constructed feature vectors. I *SIGIR '96: Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, side 157–165, New York, NY, USA, 1996. ACM Press.

[10] Ismail Haritaoglu, David Harwood og Larry S. Davis. W4: Who? When? Where? What? A Real Time System for Detecting and Tracking People. I *FG*, side 222–227, 1998.

[11] Janne Heikkilä og Olli Silvén. A Real-Time System for Monitoring of Cyclists and Pedestrians. I *VS '99: Proceedings of the Second IEEE Workshop on Visual Surveillance*, side 74. IEEE Computer Society, 1999.

[12] Jason Hunter og Brett McLaughlin. JDOM, Mai 2005. http://www.jdom.org.

[13] Nils J. Nilsson. *Artificial Intelligence: A New Synthesis*. Morgan Kaufmann Publishers, 1998.

[14] P. KaewTraKulPong og R. Bowden. An Improved Adaptive Background Mixture Model for Real-time Tracking with Shadow Detection, 2001.

[15] Jinman Kang, Isaac Cohen og Gérard Medioni. Multi-views tracking within and across uncalibrated camera streams. I *IWVS '03: First ACM SIGMM international workshop on Video surveillance*, side 21–33. ACM Press, 2003.

[16] M. Kass, A. Witkin og D. Terzopoulos. Snakes: Active Contour Models. *International Journal of Computer Vision*, 1(4):321–331, 1988.

[17] Lawrence A. Klein. *Sensor Technologies and Data Requirements for ITS*. Artech House, London, 2001.

[18] Dieter Koller, Joseph Weber og Jitendra Malik. Robust Multiple Car Tracking with Occlusion Reasoning. I *ECCV (1)*, side 189–196, 1994. http://citeseer.ist.psu.edu/article/koller93robust.html.

[19] A. Lipton, H. Fujiyoshi og R. Patil. Moving target classification and tracking from real-time video. I *IEEE Image Understanding Workshop*, side 129–136, 1998. http://citeseer.ist.psu.edu/lipton98moving.html.

[20] Guojun Lu. *Multimedia database management systems*. Artech House, Inc., Norwood, MA, USA, 1999.

[21] Alessandro Bevilacqua Member. Effective Object Segmentation in a Traffic Monitoring Application.

[22] Sun Microsystems. Java Advanced Imaging, Mars 2005. http://java.sun.com/products/java-media/jai/forDevelopers/jai-apidocs/index.html.

[23] Sun Microsystems. JMF 2.1.1e Software, Februar 2005. http://java.sun.com/products/java-media/jmf/2.1.1/download.html.

[24] P. Matthews M.Slinn og P.Guest. *Traffic Engineering Design*. Arnold Publishers, 2002.

[25] Gonzalo Navarro. A guided tour to approximate string matching. *ACM Comput. Surv.*, 33(1):31–88, 2001.

[26] F. Porikli og O. Tuzel. Human body tracking by adaptive background models and mean-shift analysis. I *IEEE Int. Conference on Computer Vision Systems*, 2003. Workshop on PETS.

[27] Segmentation Wayne Power. Understanding Background Mixture Models for Foreground.

[28] R. K. Raman. Active Contour Models for Object Tracking. Hovedfagsoppgave, St. Edmunds college, Mai 2003. Computer Science Tripos Part II.

[29] Arnstein Ressem. Segmentering, følging og klassifisering i videoanalyse av trafikk. Hovedfagsoppgave, Norwegian University of Science and Technology, 2000.

[30] Alan Ivor Reveal. Background subtraction techniques.

[31] Ehud Rivlin, Michael Rudzsky, Roman Goldenberg, U. Bogomolov og S. Lepchev. A Real-Time System for Classification of Moving Objects. I *ICPR (3)*, side 688–691, 2002.

[32] Milan Sonka, Vaclav Hlavac og Roger Boyle. *Image Processing, Analysis, and Machine Vision*. PWS Publishing, International Thomson Publishing Inc., andre utgave, 1999.

[33] Chris Stauffer og W. E. L. Grimson. Adaptive Background Mixture Models for Real-time Tracking. I *CVPR*, side 246–252, The Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139, Juni 1999.

[34] Welcome to vitualdub.org!, Januar 2005. http://www.virtualdub.org.

[35] Extensible Markup Language (XML). http://www.w3.org/XML/, April 2005.

[36] DTD Tutorial. http://www.w3schools.com/dtd/, April 2005.

[37] XPath Tutorial. http://www.w3schools.com/xpath/, April 2005.

[38] XQuery Tutorial. http://www.w3schools.com/xquery, April 2005.

[39] XSLT Tutorial. http://www.w3schools.com/xsl/, April 2005.

[40] Wikipedia, Mai 2005.
http://www.wikipedia.org.

[41] Donna J. Williams og Mubarak Shah. A Fast Algorithm for Active Contours and Curvature Estimation. *CVGIP: Image Underst.*, 55(1):14–26, 1992.

[42] Christopher Richard Wren, Ali Azarbayejani, Trevor Darrell og Alex Pentland. Pfinder: Real-Time Tracking of the Human Body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):780–785, 1997.

[43] Chenyang Xu og Jerry L. Prince. GVF.m. http://iacl.ece.jhu.edu/projects/gvf/code/GVF.m. Matlab-code implementing gradient vector field.

[44] Chenyang Xu og Jerry L. Prince. Snakes, Shapes and Gradient Vector Flow. *IEEE Transactions on Image Processing*, 7(3):359–369, Mars 1998.

[45] Tao Yang, Stan Z. Li, Quan Pan og Jing Li. Real-time and accurate segmentation of moving objects in dynamic scene. I *VSSN '04: Proceedings of the ACM 2nd international workshop on Video surveillance & sensor networks*, side 136–143. ACM Press, 2004.

Vedlegg A

Aktive konturer

Med bakgrunn i resultatene rapportert om i [29], lagde vi en prøveimplementasjon av aktive konturer for skyggefjerning. Siden vi ikke oppnådde tilfredsstillende resultater i denne implementasjonen, valgte vi å presentere prøveimplementasjonen, resultater og teorien bak aktive konturer som et vedlegg til rapporten.

Aktive konturer, eller slanger (eng.: "snakes"), ble først benyttet av Kass i [16]. Metoden går ut på at en bruker initialiserer et sett med kontrollpunkter rundt en kontur i et bilde. Punktene kontrollerer splines som tilpasses et objekts kontur. Algoritmen forsøker å minimere et globalt energimål for konturen. I Kass' opprinnelige algoritme ble gradientbildet benyttet som energifunksjon.

A.1 Energitermer

Energien som skal minimeres, E_{tot}, består av to termer: *Intern* energi, E_{int}, og *ekstern* energi, E_{ekst}. Dersom konturen representeres parametrisk ved $\mathbf{v}(s) = (x(s), y(s))$, $s \in [0, 1]$, kan E_{tot} uttrykkes som i ligning A.1.

$$E_{tot} = \int_0^1 E_{int}(\mathbf{v}(s)) \ + \ E_{ekst}(\mathbf{v}(s)) \ ds \tag{A.1}$$

Termene som inngår i ligning A.1 forklares i det følgende.

A.1.1 Intern energi

Konturens interne energi kan uttrykkes ved:

$$E_{int}(\mathbf{v}(s)) = E_{elastisitet}(\mathbf{v}(s)) \ + \ E_{krumning}(\mathbf{v}(s)) \tag{A.2}$$

$$E_{elastisitet}(\mathbf{v}(s)) = \frac{1}{2}\alpha(s)|\mathbf{v}_s(s)|^2 \tag{A.3}$$

$$E_{krumning}(\mathbf{v}(s)) = \frac{1}{2}\beta(s)|\mathbf{v}_{ss}(s)|^2 \tag{A.4}$$

der $\mathbf{v}_s(s) = \frac{d\mathbf{v}}{ds}$ og $\mathbf{v}_{ss}(s) = \frac{d^2\mathbf{v}}{ds^2}$

Elastisitetsenergien styres av $\alpha(s)$ og et førsteordens ledd. Denne integreres i totalen og dermed minimeres den aktive konturens lengde. Krumningsenergien styres av $\beta(s)$ og et andreordens ledd. Ved å minimere dette uttrykket, sikrer vi at $\frac{d^2\mathbf{v}}{ds^2}$ ikke øker for mye. En lav verdi for $\frac{d^2\mathbf{v}}{ds^2}$ betyr at kurven bøyer seg sakte [28].

A.1.2 Ekstern energi

E_{ekst} er energien bildet, I, påfører den aktive konturen. Den kan uttrykkes ved gradienten, ∇I som i ligning A.5.

$$E_{ekst}(\mathbf{v}(s)) = -|\nabla I(\mathbf{v}(s))|^2 \tag{A.5}$$

Gradienten til en todimensjonal funksjon er en vektor som peker i den retningen den retningsderiverte til funksjonen har størst verdi. Energien dytter den aktive konturen mot kantene i bildet. I et bilde vil gradienten ha størst verdi og retning vinkelrett på steder med raske transisjoner i intensitet (kanter). Ligning A.5 gir derfor et energiminima dersom den aktive konturen ligger på kantene i bildet.

A.2 Algoritme

Flere algoritmer implementerer aktive konturer. Kass' opprinnelige algoritme [16] har svakheter diskutert i [28]. Vi valgte derfor å implementere Williams & Shahs metode fra [41]. Denne presenteres som algoritme A.1.

Algoritmen tar to parametre:

- En initialisert kontur representert ved et ordnet sett punkter (x_i, y_i). Konturen må være initialisert tett rundt objektet i bildet.
- Et bilde, I. Algoritmen bruker gradientbildet, ∇I, i alle beregninger.

Tre konstanter styrer algoritmen:

α styrer hvor stor innvirkning elastisitetsenergien har på energien til et punkt på konturen.

β styrer hvor stor innvirkning krumningsenergien har på energien til et punkt på konturen.

γ styrer hvor stor innvirkning bildeenergien har på energien til et punkt på konturen.

Konturen forbedres iterativt. Antall iterasjoner styres av en terskel, $Threshold_2$. Denne setter en grense for hvor mange punkter som kan endres når konturen er i likevekt. Likevektsstillingen er tilstanden med lavest energi. Energitermene beregnes for alle punktene i nabolaget, $\mathcal{N}$, og lagret i todimensjonale tabeller. I hver iterasjon forsøker algoritmen å minimere energien i hvert punkt, (x_i, y_i). På linje 6–8 forsikrer algoritmen seg om at $Max - Min > Threshold_1$. $Max - Min$ benyttes i linje 12 for utregning av energibidraget fra bildet i nabolaget $\mathcal{N}$. Linje 9–13 beregner energibidragene for punktene i $\mathcal{N}$. Tilnærminger til energibidragene benyttet i implementasjonen beskrives under.

Algoritme A.1 Williams-Shah($contour, I$)

1: **repeat**
2: **for all** points $(x_i, y_i) \in contour$ **do**
3: consider a neighbourhood $\mathscr{N}$ of size $m \times m$ centered at (x_i, y_i)
4: $Max \leftarrow \max_{(x,y)\in\mathscr{N}} \nabla I(x, y)$
5: $Min \leftarrow \min_{(x,y)\in\mathscr{N}} \nabla I(x, y)$
6: **if** $Max - Min < Threshold_1$ **then**
7: $Min \leftarrow Max - Threshold_1$
8: **end if**
9: **for all** points $(x, y) \in \mathscr{N}$ **do**
10: $E_{elastic}(x, y) \leftarrow d_{avg} - |v_i - v_{i-1}|$
11: $E_{bending}(x, y) \leftarrow |v_{i-1} - 2v_i + v_{i+1}|^2$
12: $E_{image}(x, y) \leftarrow (Min - \nabla I(x, y))/(Max - Min)$
13: **end for**
14: $EMax_{elastic} \leftarrow \max_{(x,y)\in\mathscr{N}} E_{elastic}(x, y)$
15: $EMax_{bending} \leftarrow \max_{(x,y)\in\mathscr{N}} E_{bending}(x, y)$
16: **for all** points $(x, y) \in \mathscr{N}$ **do**
17: $E_{elastic}(x, y) \leftarrow E_{elastic}/EMax_{elastic}$
18: $E_{bending}(x, y) \leftarrow E_{bending}/EMax_{bending}$
19: $E_{total}(x, y) \leftarrow \alpha E_{elastic}(x, y) + \beta E_{bending}(x, y) + \gamma E_{image}(x, y)$
20: **end for**
21: Find $(x', y') \in \mathscr{N}.\forall(x'', y'') \in \mathscr{N}.E_{total}(x', y') < E_{total}(x'', y'')$
22: $(x_i, y_i) \leftarrow (x', y')$
23: **end for**
24: **until** Number of points moved $< Threshold_2$
25: **return** contour

Linjene 16–20 normaliserer energibidragene innenfor $\mathscr{N}$. Disse adderes inn i en ny tabell som holder på totalenergien til pikslene i nabolaget. Linje 21 finner punktet $(x', y') \in \mathscr{N}$ med lavest energi. På linje 22 settes konturpunktet (x_i, y_i) til (x', y'). Alt dette gjentar seg inntil mindre enn $Threshold_2$ punkter endres i løpet av en iterasjon.

A.2.1 Tilnærminger

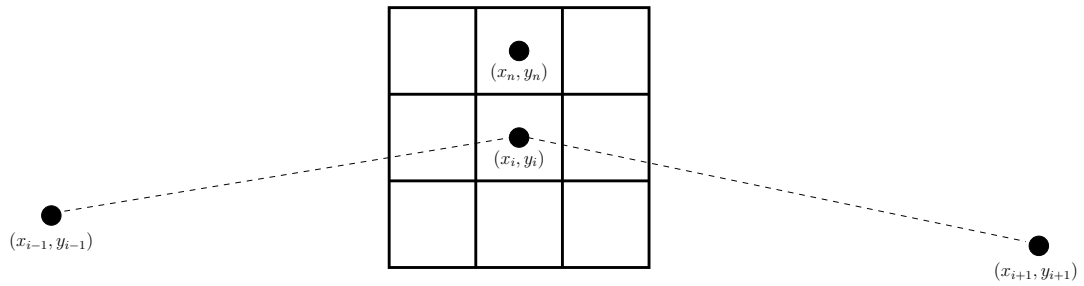
Tilnærming til elastisitets- og krumningsenergien er nødvendig, siden disse benytter henholdsvis den førstederiverte og den andrederiverte med hensyn på konturen. Figur A.1 viser punktene som er med i tilnærmingene. (x_{i-1}, y_{i-1}) tilsvarer forrige punkt på konturen, og (x_{i+1}, y_{i+1}) er neste punkt på konturen.

Med utgangspunkt i figur A.1, tilnærmes $E_{elastisitet}$ og $E_{krumning}$ som i ligning A.6–A.7 fra [28].

$$E_{elastisitet}(x, y) = \overline{d} - \sqrt{(x_n - x_{i-1})^2 + (y_n - y_{i-1})^2} \quad (A.6)$$

$$E_{krumning}(x, y) = \sqrt{(x_{i+1} - 2x_i + x_{i-1})^2 + (y_{i+1} - 2y_i + y_{i-1})^2} \quad (A.7)$$

der $\overline{d}$ er gjennomsnittsavstand mellom punktene i den aktive konturen.



Figur A.1: Punkter brukt i tilnærmingen til energitermene [28].

A.3 Prøveimplementasjon

Vi lagde en prøveimplementasjon av aktive konturer for å fjerne skyggepiksler fra segmenterte objekter. Resultatet er en er en kontur bestående av ordnet sett kontrollpunkter som beskriver en spline. Disse kontrollpunktene omformes til sammenhengende kurver.

A.3.1 Opptegning

For å representere konturen definert ved et sett kontrollpunkter benyttet vi Catmull-Clarks subdivisjonsalgoritme [28]. Denne er gjengitt som algoritme A.2.

Algoritme A.2 Catmull-Clark(*contour*)

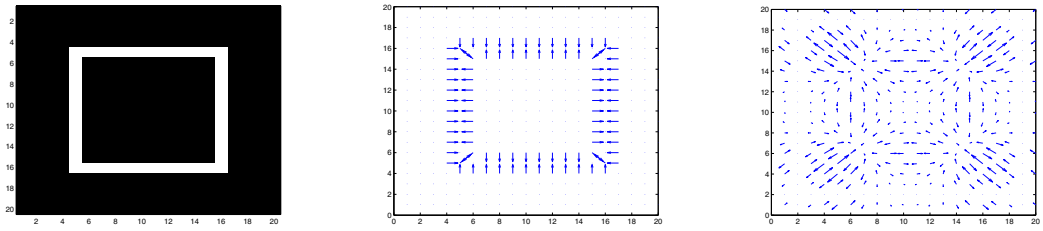
```
1:  repeat
2:     oldContour ← contour
3:     newContour ← new Set
4:     for all points p_i ∈ contour do
5:        ADD(newContour, 2i, 1/8 p_{i-1} + 6/8 p_i + 1/8 p_{i+1})
6:        ADD(newContour, 2i + 1, 1/2 (p_i + p_{i+1}))
7:     end for
8:     remove duplicate points from newContour
9:     contour ← newContour
10: until contour.COUNT = oldContour.COUNT
11: return contour
```

Algoritme A.2 tar inn et ordnet sett, *contour*, kontrollpunkter, p_i. På linje 5 og 6 brukes ADD. ADD legger til det tredje argumentet på posisjonen til det andre argumentet i settet definert ved det første argumentet. På linje 10 brukes COUNT for å returnere antall punkter i et sett. Konturen returneres som et ordnet sett sammenhengende punkter.

A.3.2 Gradient-vektor-flyt

Vi eksperimenterte også med gradient-vektor-flyt (GVF) felt som som input til algoritmen. Et GVF-felt er et bedre utgangspunkt for å beregne aktive konturer [44].

Figur A.2(c) illustrerer et GVF vektorfelt. I figur A.2(b) er gradientbildet. Begge disse figurene er generert ut i fra rektangelet i figur A.2(a). I motsetning til gradientfeltet i figur A.2(b), strekker vektorfeltet i figur A.2(c) seg utover hele bildet. Et slikt vektorfelt gjør det mulig å initialisere konturen i lengre avstand i fra objektet samtidig som konturen ikke forringes som ved vanlig glatting.



(a) Bilde av et rektangel. (b) Gradientbilde (c) GVF-felt

Figur A.2: Gradient-vektor-flyt felt

Metoden som genererer GVF-feltet presenteres som algoritme A.3 fra [43]. Algoritmen tar tre parametre:

F er bildet representert som en matrise.

μ er en konstant som settes høyere desto mer støy det er i bildet.

iterations er antall iterasjoner algoritmen skal kjøre.

Algoritme A.3 GVF(F, μ, *iterations*)

1: $(m, n) \leftarrow$ SIZE(F)
2: NORMALIZE(F)
3: FX $\leftarrow$ GRADIENT$_x$(F)
4: FY $\leftarrow$ GRADIENT$_y$(F)
5: U $\leftarrow$ FX
6: V $\leftarrow$ FY
7: **for all** pixels (i, j) **do**
8: $\quad \text{SMF}_{ij} \leftarrow \text{FX}_{ij}^2 + \text{FY}_{ij}^2$
9: **end for**
10: **for** $k \leftarrow 1$ **to** *iterations* **do**
11: $\quad$ **for all** pixels (i, j) **do**
12: $\quad\quad \text{U}_{ij} \leftarrow \text{U}_{ij} + \mu \cdot 4 \cdot \nabla^2 \text{U}_{ij} - \text{SMF}_{ij} \cdot (\text{U}_{ij} - \text{FX}_{ij})$
13: $\quad\quad \text{V}_{ij} \leftarrow \text{V}_{ij} + \mu \cdot 4 \cdot \nabla^2 \text{V}_{ij} - \text{SMF}_{ij} \cdot (\text{V}_{ij} - \text{FY}_{ij})$
14: $\quad$ **end for**
15: **end for**
16: **return** (U, V)

SMF holder kvadratet til gradientvektorene (FX, FY). ∇^2 betegner Laplace-operatoren. Algoritme A.3 benytter fire eksterne funksjoner:

SIZE returnerer størrelsen på en matrise.

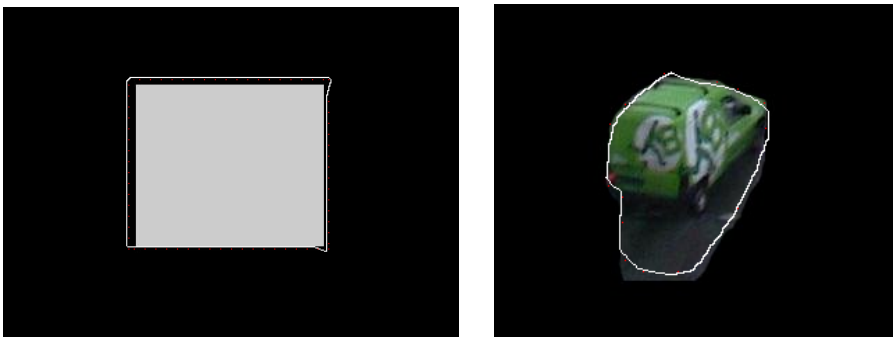
NORMALIZE normaliserer verdiene i en matrise i intervallet $[0, 1]$.

GRADIENT$_x$ og GRADIENT$_y$ regner ut henholdsvis gradientens x- og y-komponent.

A.4 Resultater

Figur A.3 viser resultatet av prøveimplementasjonen anvendt på to testbilder. Konturen i testbildene er tegnet som en hvit heltrukken linje. Initialiseringen (kontrollpunktene) er markert som røde prikker. Figur A.3(a) viser et overdrevent enekelt tilfelle med et grått rektangel på svart bakgrunn. Slangen legger seg inntil rektangelet på undersiden. Også på høyre side nærmer den aktive konturen seg. På venstre side og på toppen har konturen derimot fjernet seg fra rektangelet.

Figur A.3(b) viser en realistisk test hvor vi har forsøkt å initialisere konturen rundt en bil og dens skygge. Den aktive konturen har ikke greid å nærme seg bilen nevneverdig.



(a) Enkel test. (b) Realistisk test.

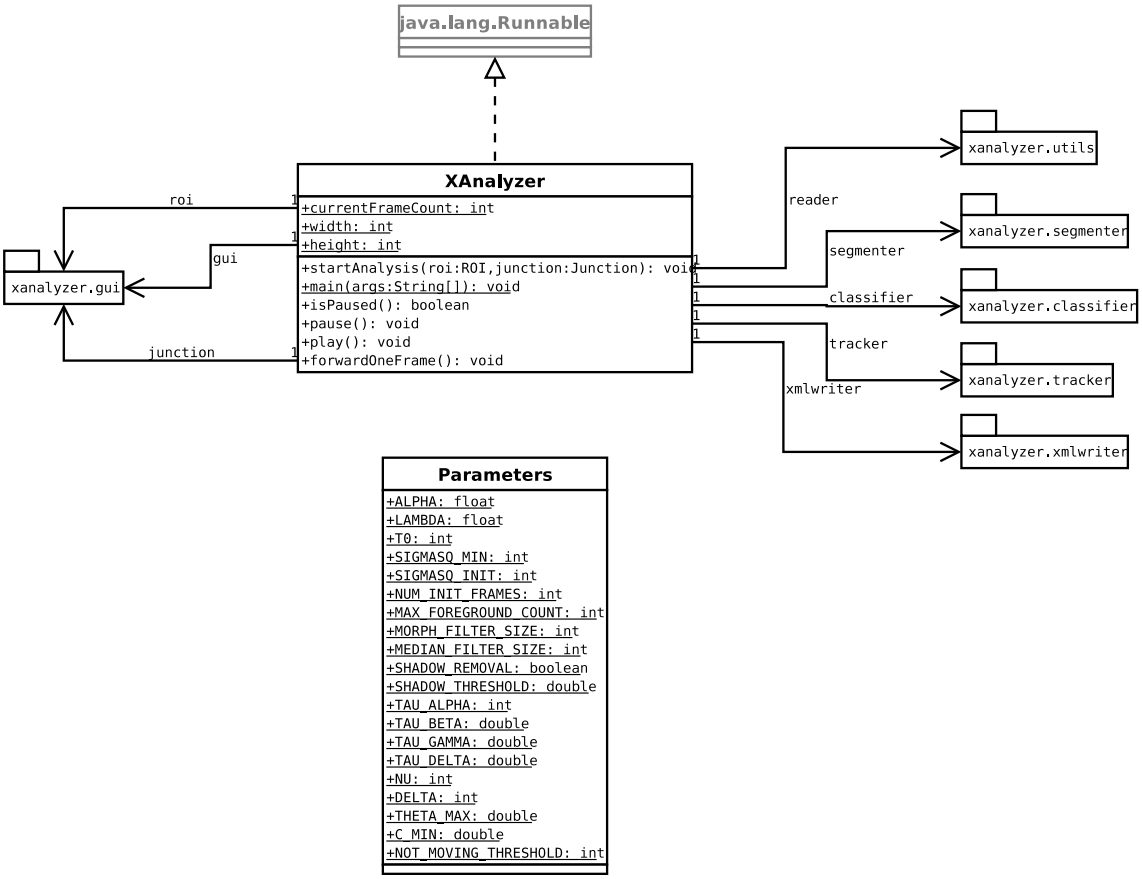
Figur A.3: Den resulterende aktive konturen er illustrert som hvit heltrukken linje. Initialiseringspunkter illustrert ved røde prikker.

Disse resultatene og deres beregningsmessige intensivitet gjorde at vi bestemte oss for å forkaste prøveimplementasjonen og heller benytte andre metoder for skyggefjerning.

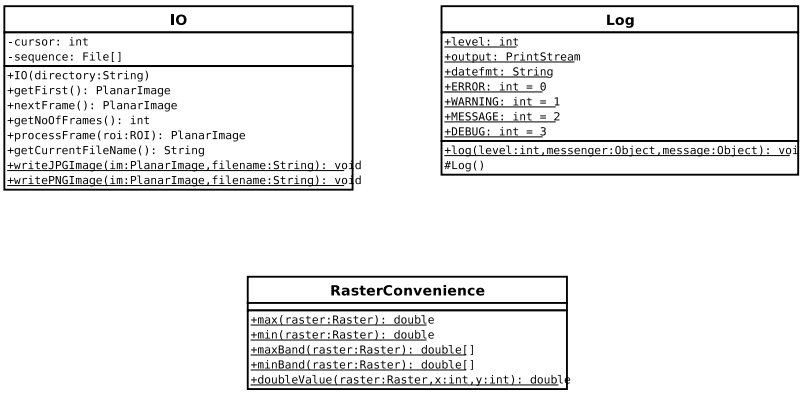
Vedlegg B

UML

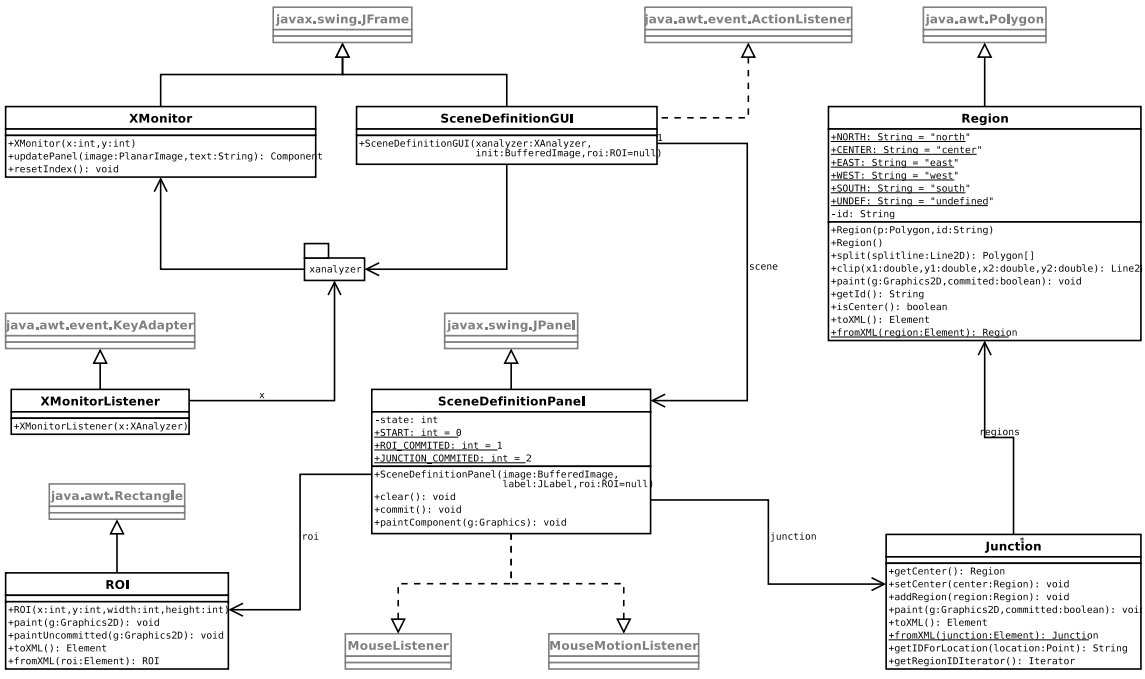
B.1 X-Analyzer



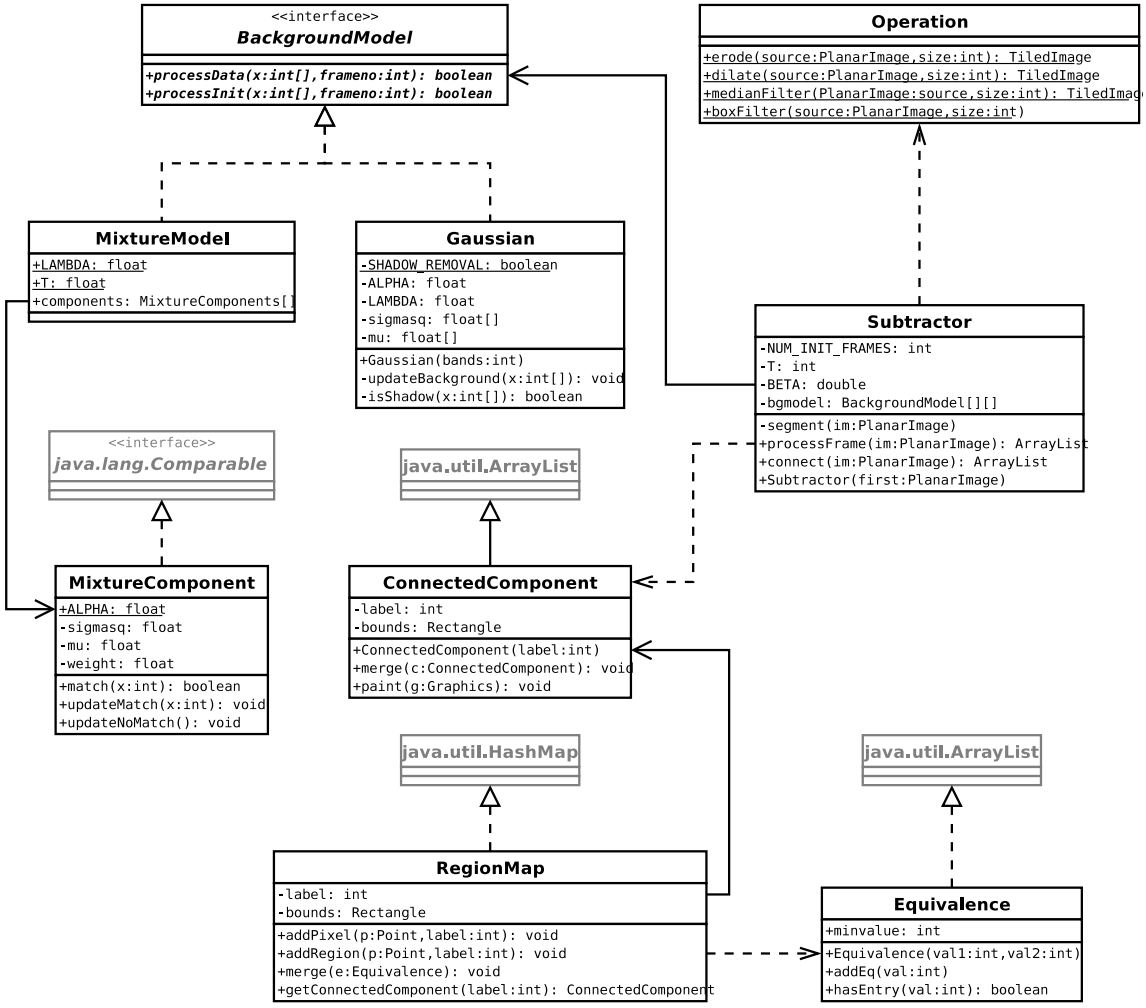
Figur B.1: UML klassediagram for xanalyzer



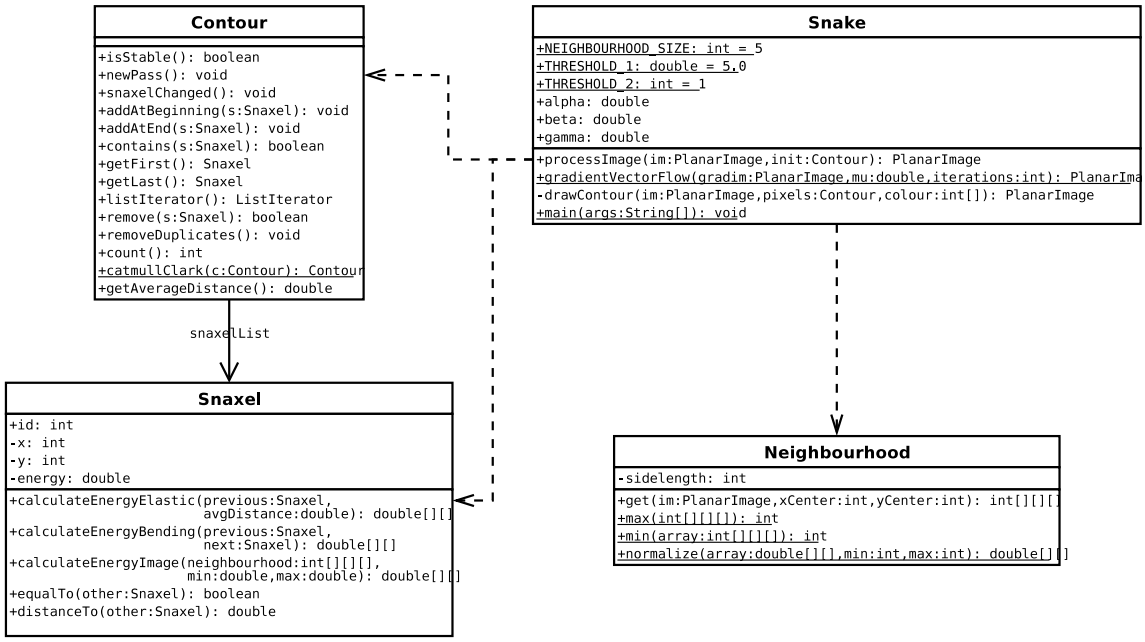
Figur B.2: UML klassediagram for xanalyzer.utils



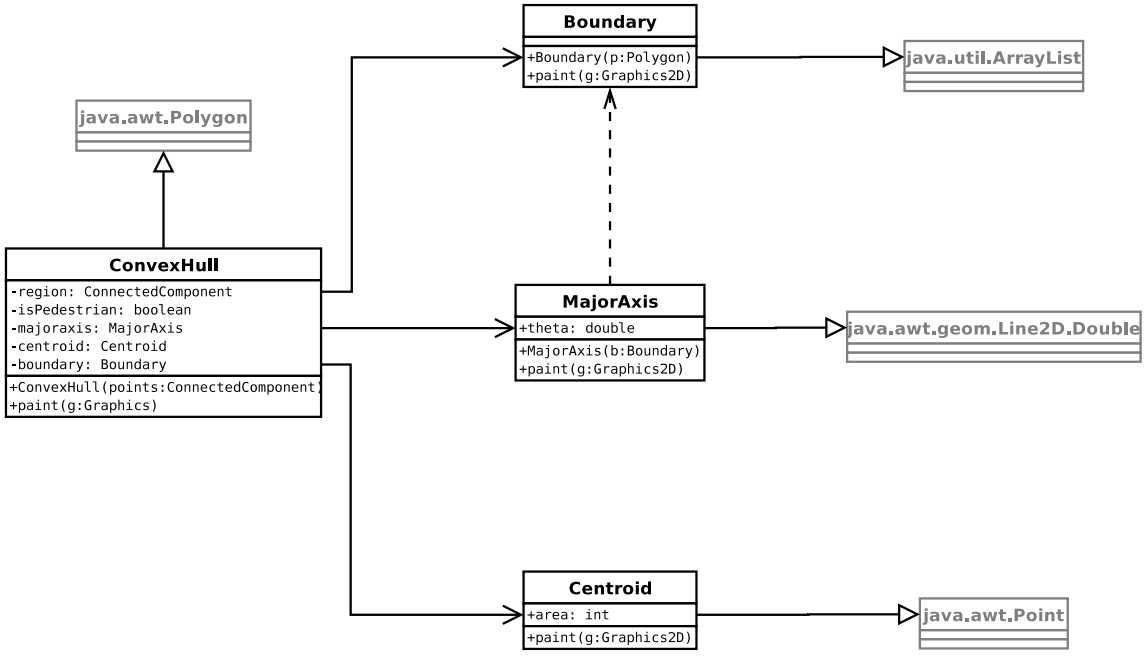
Figur B.3: UML klassediagram for xanalyzer.gui



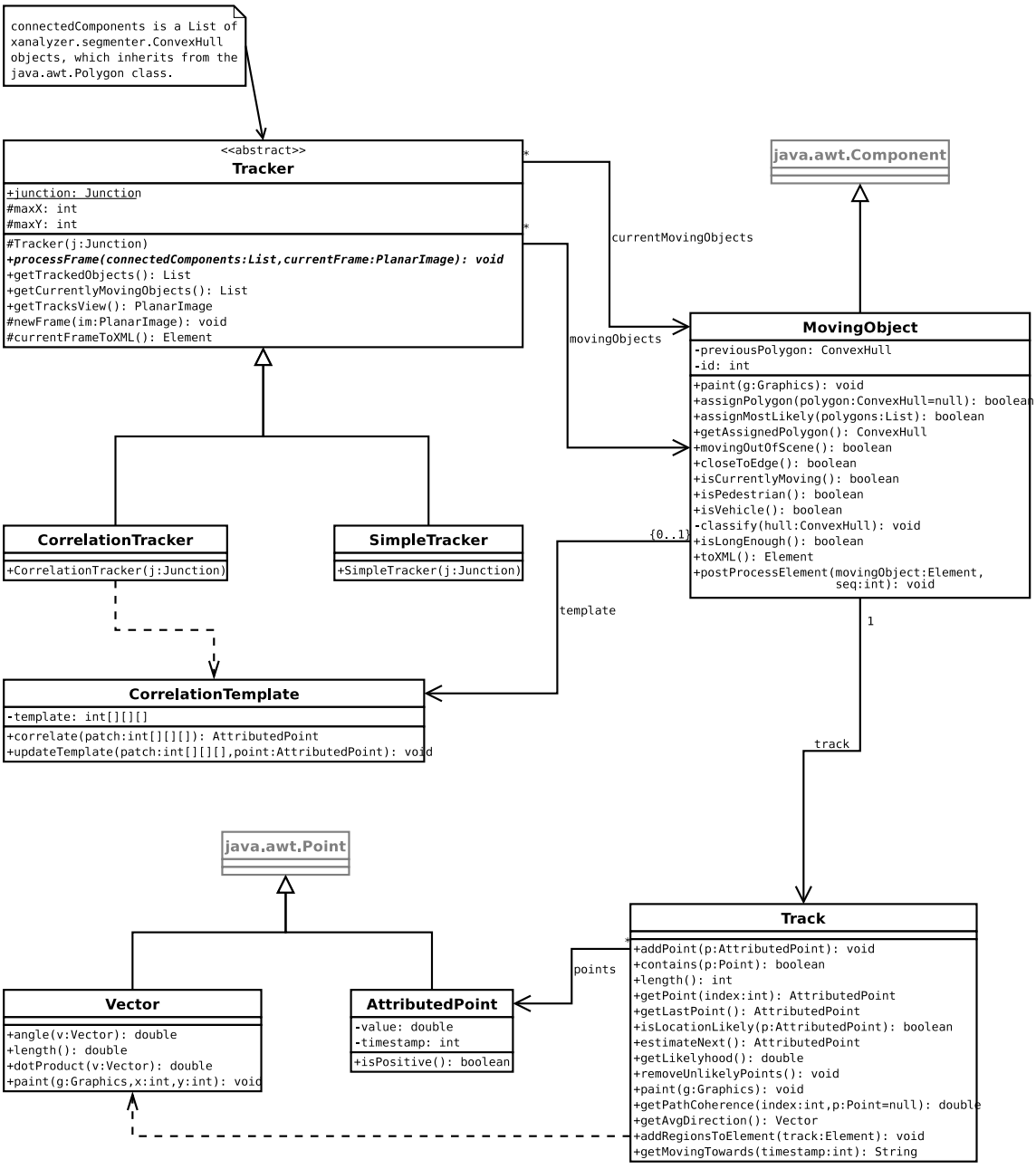
Figur B.4: UML klassediagram for xanalyzer.segmenter



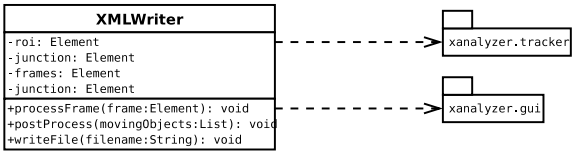
Figur B.5: UML klassediagram for xanalyzer.segmenter.snake



Figur B.6: UML klassediagram for xanalyzer.classification

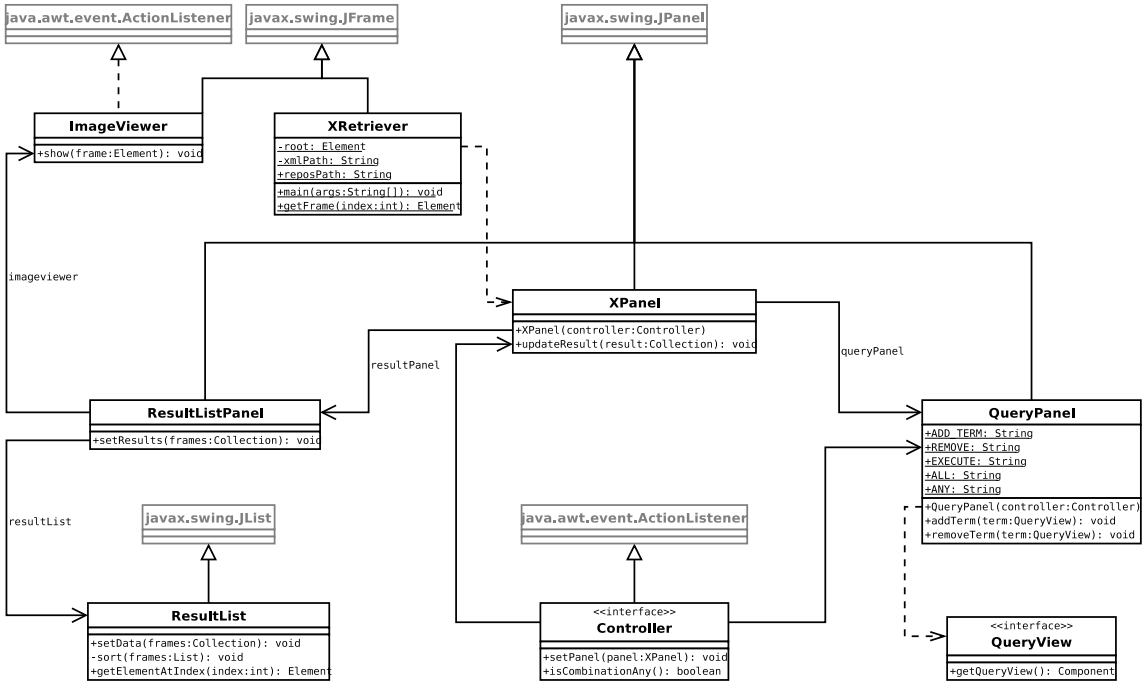


Figur B.7: UML klassediagram for xanalyzer.tracker

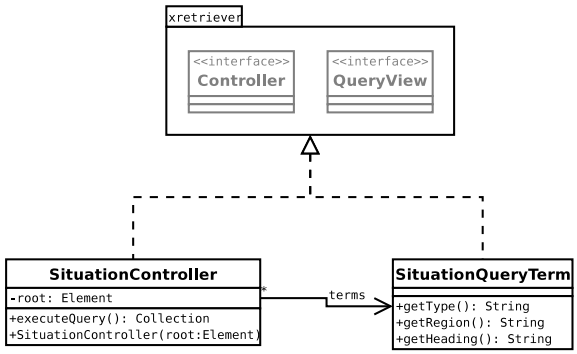


Figur B.8: UML klassediagram for xanalyzer.xmlwriter

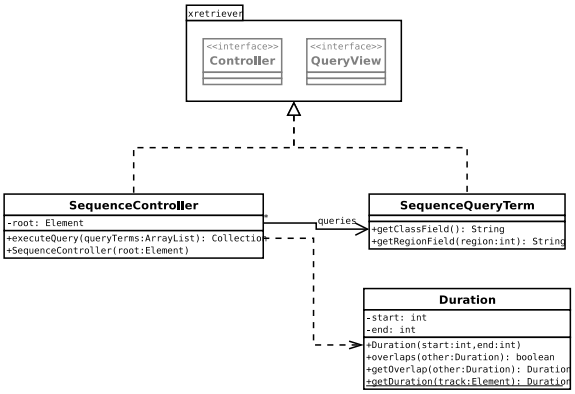
B.2 X-Retriever



Figur B.9: UML klassediagram for xretriever



Figur B.10: UML klassediagram for xretriever.situation



Figur B.11: UML klassediagram for xretriever.sequence

Vedlegg C

XML

C.1 X-Analyzer Schema

```
<?xml version="1.0"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:complexType name="pointType">
  <xs:all>
   <xs:element name="x" type="xs:nonNegativeInteger" minOccurs="1"/>
   <xs:element name="y" type="xs:nonNegativeInteger" minOccurs="1"/>
  </xs:all>
 </xs:complexType>

 <xs:complexType name="polygonPointType">
  <xs:all>
   <xs:element name="seq" type="xs:integer" minOccurs="1"/>
   <xs:element name="x" type="xs:nonNegativeInteger" minOccurs="1"/>
   <xs:element name="y" type="xs:nonNegativeInteger" minOccurs="1"/>
  </xs:all>
 </xs:complexType>

 <xs:simpleType name="regionIdType">
  <xs:restriction base="xs:string">
   <xs:enumeration value="south"/>
   <xs:enumeration value="north"/>
   <xs:enumeration value="east"/>
   <xs:enumeration value="west"/>
   <xs:enumeration value="center"/>
  </xs:restriction>
 </xs:simpleType>

 <xs:simpleType name="headingType">
  <xs:restriction base="xs:string">
   <xs:enumeration value="south"/>
   <xs:enumeration value="north"/>
   <xs:enumeration value="east"/>
   <xs:enumeration value="west"/>
   <xs:enumeration value="center"/>
   <xs:enumeration value="none"/>
   <xs:enumeration value="out"/>
  </xs:restriction>
 </xs:simpleType>

 <xs:simpleType name="classType">
  <xs:restriction base="xs:string">
   <xs:enumeration value="vehicle"/>
```

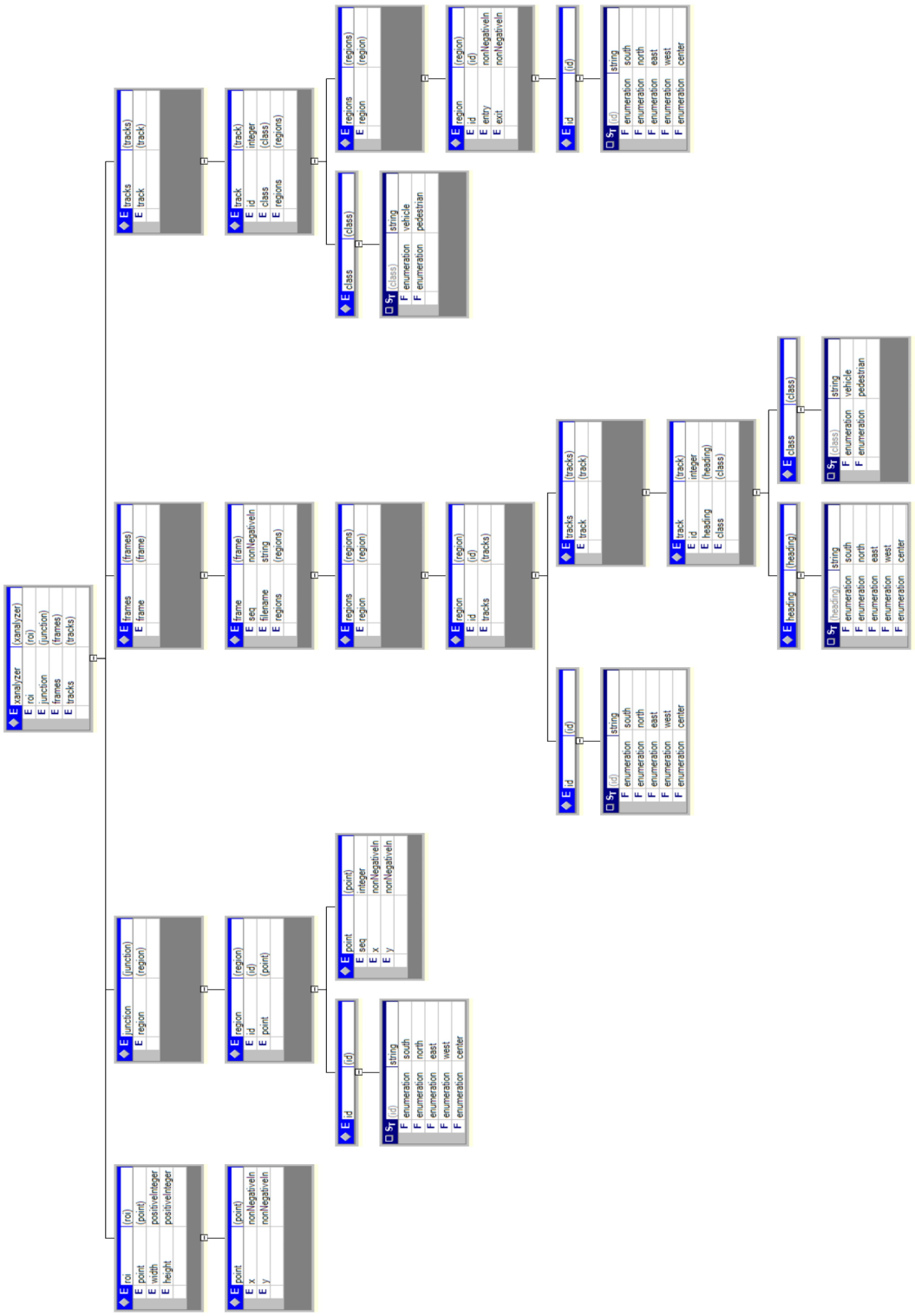
```
   <xs:enumeration value="pedestrian"/>
  </xs:restriction>
 </xs:simpleType>

 <xs:element name="xanalyzer">
  <xs:complexType>
   <xs:all>
    <xs:element name="roi">
     <xs:complexType>
      <xs:all>
       <xs:element name="point" type="pointType" minOccurs="1"/>
       <xs:element name="width" type="xs:positiveInteger" minOccurs="1"/>
       <xs:element name="height" type="xs:positiveInteger" minOccurs="1"/>
      </xs:all>
     </xs:complexType>
    </xs:element>
    <xs:element name="junction">
     <xs:complexType>
      <xs:sequence>
       <xs:element name="region" minOccurs="1" maxOccurs="unbounded">
        <xs:complexType>
         <xs:sequence>
          <xs:element name="id" minOccurs="1" maxOccurs="1"
          type="regionIdType"/>
          <xs:element name="point" minOccurs="3" maxOccurs="unbounded"
          type="polygonPointType"/>
         </xs:sequence>
        </xs:complexType>
       </xs:element>
      </xs:sequence>
     </xs:complexType>
    </xs:element>

    <xs:element name="frames">
     <xs:complexType>
      <xs:sequence>
       <xs:element name="frame" minOccurs="1" maxOccurs="unbounded">
        <xs:complexType>
         <xs:all>
          <xs:element name="seq" type="xs:nonNegativeInteger" minOccurs="1"
          maxOccurs="1"/>
          <xs:element name="filename" type="xs:string" minOccurs="1"
          maxOccurs="1"/>
          <xs:element name="regions" minOccurs="1" maxOccurs="1">
           <xs:complexType>
            <xs:sequence>
             <xs:element name="region" minOccurs="0" maxOccurs="unbounded">
              <xs:complexType>
               <xs:all>
                <xs:element name="id" minOccurs="1" maxOccurs="1"
                type="regionIdType"/>
                <xs:element name="tracks" minOccurs="1" maxOccurs="1">
                 <xs:complexType>
                  <xs:sequence>
                   <xs:element name="track" minOccurs="1"
                   maxOccurs="unbounded">
                    <xs:complexType>
                     <xs:all>
                      <xs:element name="id" type="xs:integer"
                      minOccurs="1" maxOccurs="1"/>
                      <xs:element name="heading"
                      minOccurs="1" maxOccurs="1" type="headingType"/>
                      <xs:element name="class"
                      minOccurs="1" maxOccurs="1" type="classType"/>
                     </xs:all>
                    </xs:complexType>
                   </xs:element>
```

```
                </xs:sequence>
               </xs:complexType>
              </xs:element>
             </xs:all>
            </xs:complexType>
           </xs:element>
          </xs:sequence>
         </xs:complexType>
        </xs:element>
       </xs:all>
      </xs:complexType>
     </xs:element>
    </xs:sequence>
   </xs:complexType>
  </xs:element>

  <xs:element name="tracks">
   <xs:complexType>
    <xs:sequence>
     <xs:element name="track" minOccurs="0" maxOccurs="unbounded">
      <xs:complexType>
       <xs:all>
        <xs:element name="id" type="xs:integer" minOccurs="1" maxOccurs="1"/>
        <xs:element name="class" minOccurs="1" maxOccurs="1" type="classType"/>
        <xs:element name="regions" minOccurs="1" maxOccurs="1">
         <xs:complexType>
          <xs:sequence>
           <xs:element name="region" minOccurs="0" maxOccurs="unbounded">
            <xs:complexType>
             <xs:all>
              <xs:element name="id" minOccurs="1" maxOccurs="1"
              type="regionIdType"/>
              <xs:element name="entry" type="xs:nonNegativeInteger"
              minOccurs="1" maxOccurs="1"/>
              <xs:element name="exit" type="xs:nonNegativeInteger"
              minOccurs="1" maxOccurs="1"/>
             </xs:all>
            </xs:complexType>
           </xs:element>
          </xs:sequence>
         </xs:complexType>
        </xs:element>
       </xs:all>
      </xs:complexType>
     </xs:element>
    </xs:sequence>
   </xs:complexType>
  </xs:element>
 </xs:all>
</xs:complexType>
</xs:element>
</xs:schema>
```



Figur C.1: XML Schema