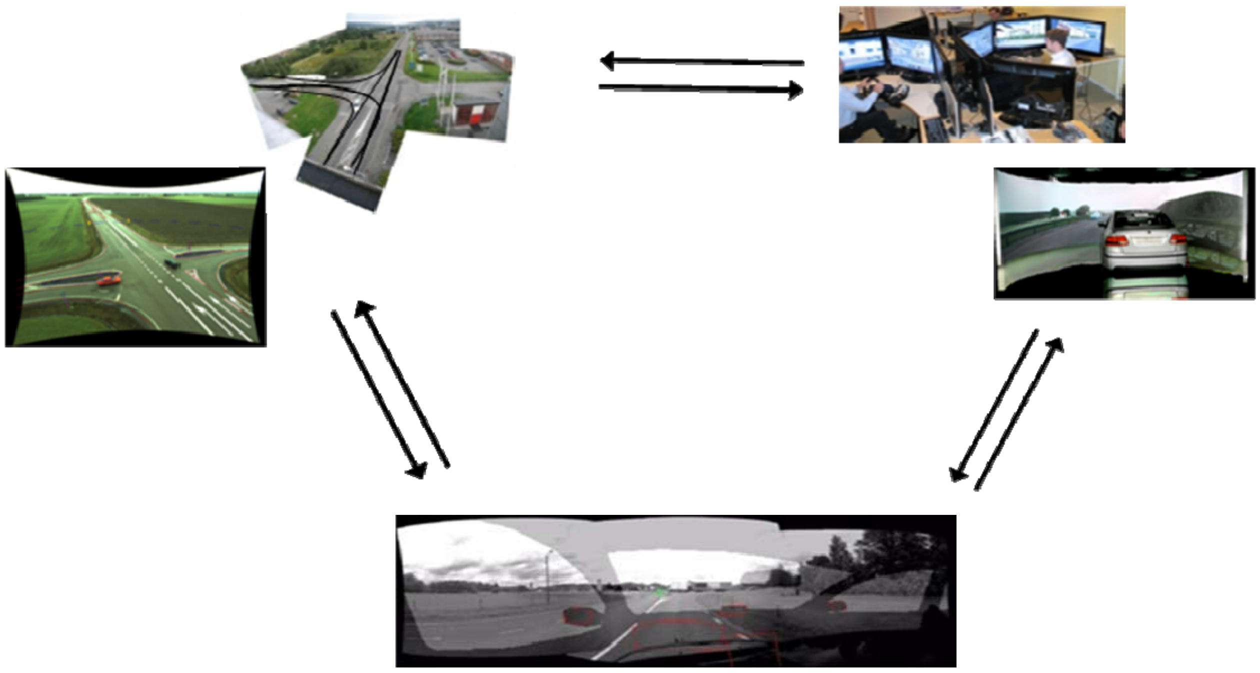
Intelligent Vehicle Safety Systems



IVSS Intersection accidents: Analysis and Prevention

Final Report – Volume 2 (appendices)

IVSS Intersection accidents: Analysis and Prevention

Final Report – Volume 2 (appendices)

Leif Franzén
Gösta Granlund
Björn Johansson
Björn Johansson
Fredrik Moeschlin
Linda Renner
András Várhelyi

Title of the report: IVSS Intersection Accidents: Analysis and Prevention, Final Report – Volume 1 (appendices)
Reference number (IVSS): AL80 A 2008:73343
ISBN: 978-91-633-5031-3
Authors: Leif Franzén, Gösta Granlund, Björn Johansson, Björn Johansson, Fredrik Moeschlin, Linda Renner, András Várhelyi
Publication date: June 30, 2009
Project manager and contact person: Jonas Bärgman, Autoliv Development AB, Vårgårda, Sweden
IVSS contact: Torbjörn Biding, Vägverket, Sweden
Printed: Chalmers Reproservice, Göteborg, Sweden, 2009

Introduction

Driving in intersections is a task that most people are familiar with and perform on a daily basis. This project was initiated to advance the understanding of driver actions and behavior in intersections as a basis for developing in-vehicle active safety systems. We used several different approaches and were able to extract useful and corroborating results that industry and academia can use to enhance traffic safety by developing active safety systems for cars. The results range from new methods and taxonomies for further research and development to data that can be used in the design of strategies for warnings and interventions in in-vehicle active safety systems. This report contains a set of results I hope will be a welcome contribution to the research community in better understanding driving in intersections. This is the appendices part of the report, covering some of the basic research and intermediate reports and studies produced in the project.

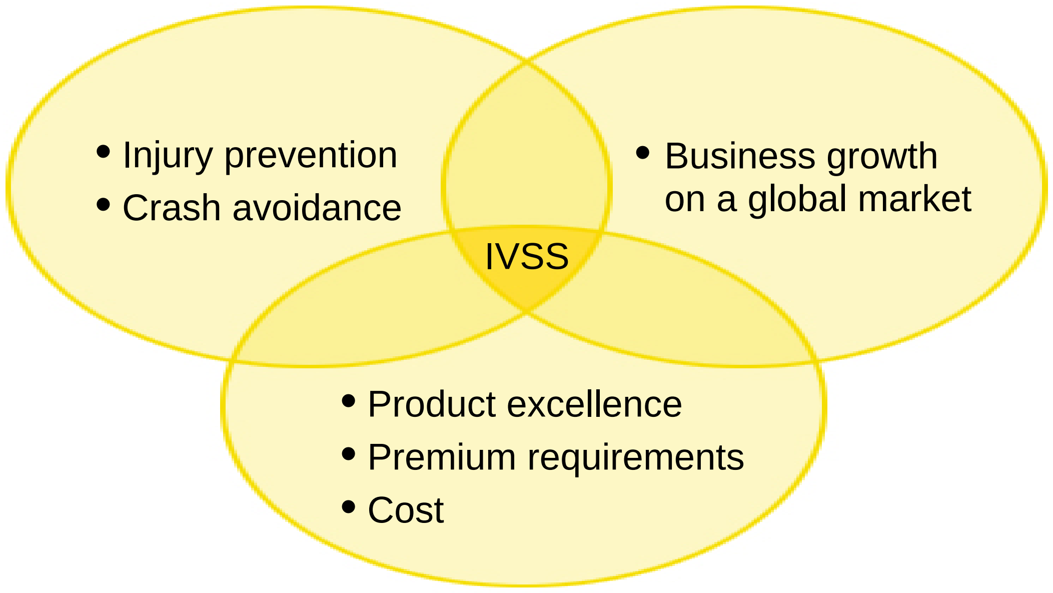
The project was funded partially by the Swedish government (via IVSS – see next page) and partly by the Swedish automotive industry. As the project manager I would like to thank all participating partners for their valuable contributions, with special thanks to the principle investigator, professor Kip Smith, for his devotion to the project and large contribution to this report. I would also like to thank IVSS for their funding and support. I know the knowledge gained by the partners due to this project are being and will be used in many direct and indirect ways.

Jonas Bärgman
Project manager, Autoliv Development AB

The IVSS Program

The IVSS program was set up to stimulate research and development for the road safety of the future. The end result will probably be new, smart technologies and new IT systems that will help reduce the number of traffic-related fatalities and serious injuries.

IVSS projects shall meet the following three criteria: road safety, economic growth and commercially marketable technical systems.



Three interacting components - for better safety, growth and competitiveness:

The human being

Preventive solutions based on the vehicle’s most important component.

The road

Intelligent systems designed to increase security for all road users.

The vehicle

Active safety through pro-active technology.

Table of Contents
Volume 2 - Appendices

KONFLIKTSTUDIER I LANDSVÄGSKORSNINGEN MELLAN VÄG E20 OCH 2623 UTANFÖR JUNG

András Várhelyi
Leif Franzén

2006

Lunds Tekniska Högskola
Institutionen för Teknik och samhälle
Trafik och väg

CODEN:LUTVDG/(TVTT-7208)1-32/2006

András Várhelyi
Leif Franzén

Konfliktstudier i landsvägskorsningen mellan väg E20 och 2623 utanför Jung

2006

Ämnesord:
Landsvägskorsning, trafikkonflikt, interaktion, hastighet,

Referat:
Konfliktobservation och hastighetsmätningar med radar genomfördes i landsvägskorsningen mellan väg E20 och 2623 utanför Jung. Tolv konflikter registrerades. Fortkörning är en vanlig företeelse genom korsningen. Det förekommer även andra regelbrott, t.ex. då fordon inte stannar vid stopplinjen.

English title:
Conflict studies at the intersection of rural roads E20 and 2623 outside Jung

Abstract:
Conflict observations and speed measurements with radar gun were carried out at the intersection of rural roads E20 and 2623 outside Jung. Twelve conflicts were registered. Speeding among drivers on the main road through the intersection was usual. Other violations of traffic rules are also common, e.g. not stopping at the stop line.

Citeringsanvisning:
András Várhelyi, Leif Franzén, Konfliktstudier i landsvägskorsningen väg 20/2623 utanför Jung. Lunds Tekniska Högskola, Institutionen för Teknik och samhälle. Trafik och väg 2006.

Institutionen för Teknik och samhälle
Lunds Tekniska Högskola
Trafik och väg
Box 118, 221 00 LUND

FÖRORD

Denna studie har genomförts på beställning av Autoliv Development AB i Vårgårda i syfte att komplettera pågående insamling av trafik- och olycksdata för den aktuella platsen. Resultaten från undersökningen skall ligga till grund för diskussioner om den fortsatta inriktningen för analysen av trafiksäkerhetssituationen i korsningen.

András Várhelyi Maj 2006

INNEHÅLL

SAMMANFATTNING

Syftet med fältmätningarna var att kartlägga vilka är de kritiska situationerna i landsvägskorsningen mellan E20 och väg 2623 utanför Jung.

Konfliktobservation och hastighetsmätningar med radar genomfördes i korsningen. Tolv konflikter (9 allvarliga) registrerades under observationsperioden bestående av 35 timmar. Alla konflikter involverade ett fordon som kör ut från östra sekundärvägen. Majoriteten av konflikterna är av typen då ett fordon kör ut från östra sekundärvägen (antingen med korsande eller vänstersvängande riktning) och det kommer i konflikt med ett fordon som passerar korsningen från syd-väst. Näst största konflikttyp är då svängande och raktframkörande fordon från var sin sekundärvägstillfart konkurrerar om den korta tidluckan flödet längs primärvägen ger och då kommer i konflikt med varandra.

Fortkörning är en vanlig företeelse genom korsningen. Mer än 65 % av förarna körde med en hastighet över hastighetsgränsen och 50 % av förarna körde fortare än 74 km/h.

Det förekommer frekventa köer på sekundärvägstillfarterna. Lastbilsförare som pratar i mobiltelefon är mer regel än undantag när fordonen passerar korsningen. Det förekommer att lastbilsförare har ena foten i vilställning uppe på instrumentbrädan när fordonet passerar korsningen. När fordon kör ut från sekundärvägen efter stopp och svänger vänster och det står en bil mittemot vid stopplinjen förekommer det frekvent att det vänstersvängande fordonet kör först fast det skall lämna företräde åt rakt fram korsande fordon från motsatta hållet. Många fordon stannar aldrig helt vid stopplinjen.

SUMMARY

The aim of the field observations was to explore which are the critical situations at the intersection of rural roads E20 and 2623 outside Jung.

Conflict observations and speed measurements with radar gun were carried out at the intersection. Twelve conflicts (9 serious) were registered during the observation period of 35 hours. All conflicts involved a vehicle driving out from the East secondary road. The majority of the conflicts were of the type where a vehicle drives out from the East secondary road (either crossing or left turning movement) and it comes into conflict with a vehicle passing the intersection from South-West. The next largest conflict type occurred when vehicles from the two secundary roads (turning and passing straight), competing for the short time gap in the flow of vehicles on the primary road, got in conflict with each other.

Speeding among drivers on the main road through the intersection was usual. More than 65 % of the drivers drove with a speed over the speed limit and 50 % of the drivers drove faster than 74 km/h.

There are frequent queues on the approaches of the secondary roads. Lorry drivers talking in their mobile phone when passing the intersection are more a rule than exception. It is not unusual that lorry drivers have one of their feet resting on the dashboard when their vehicle passing the intersection. When a vehicle drives out from the secondary road and turns left and there is a vehicle from the opposite approach to pass straight through the intersection, the left turning vehicle frequently do not give priority. Many vehicles do not stop at the stop line at all.

1 BAKGRUND

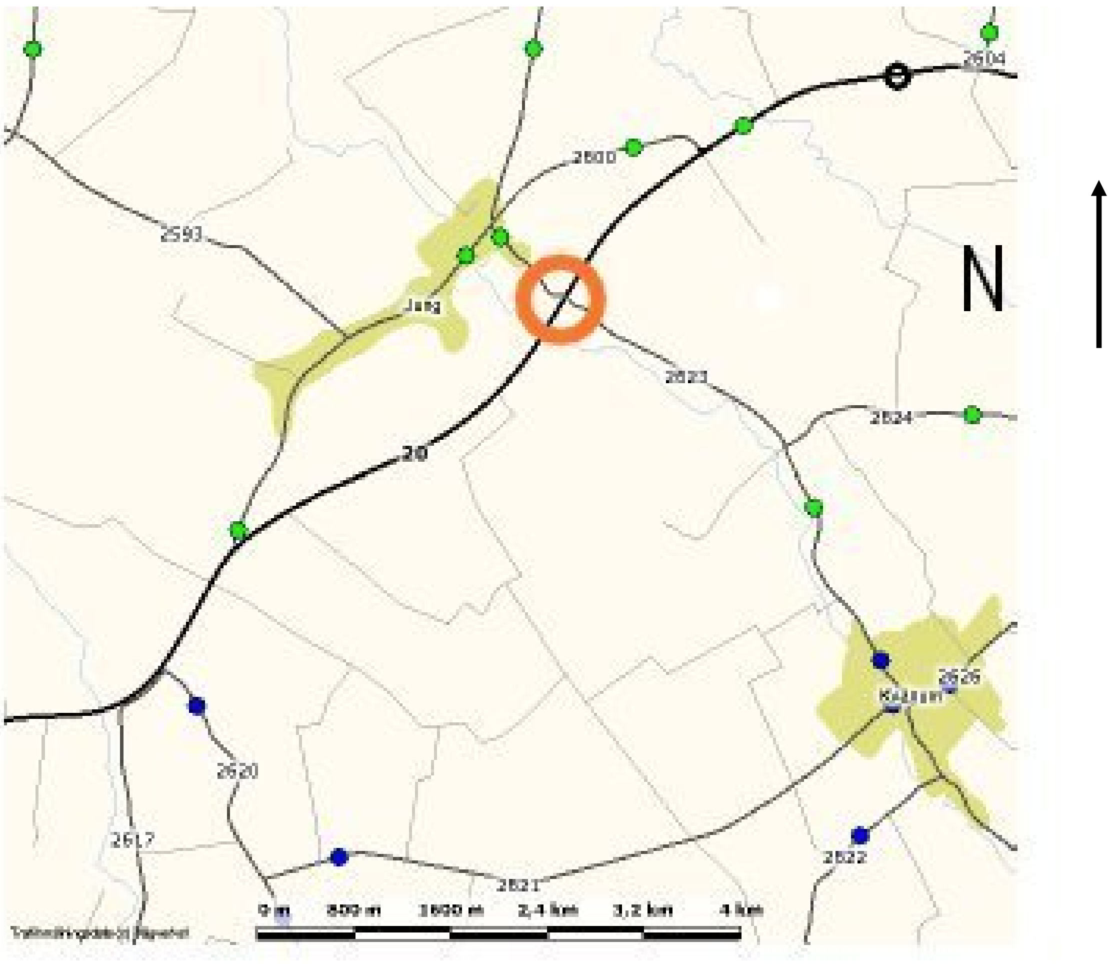
Autoliv Development AB i Vårgårda genomför trafiksäkerhetsstudier i landsvägskorsningar. Som en del av insamling av trafik- och olycksdata för den aktuella platsen uppstod behovet att i fält observera förarbeteende som kan resultera i trafikkonflikter. Som ett led i denna analys har Institutionen för teknik och samhälle vid Lunds Tekniska högskola genomfört konfliktstudier i en av målkorsningarna.

2 SYFTE

Syftet med fältmätningarna var att kartlägga vilka är de kritiska situationerna i landsvägskorsningen mellan E20 och väg 2623.

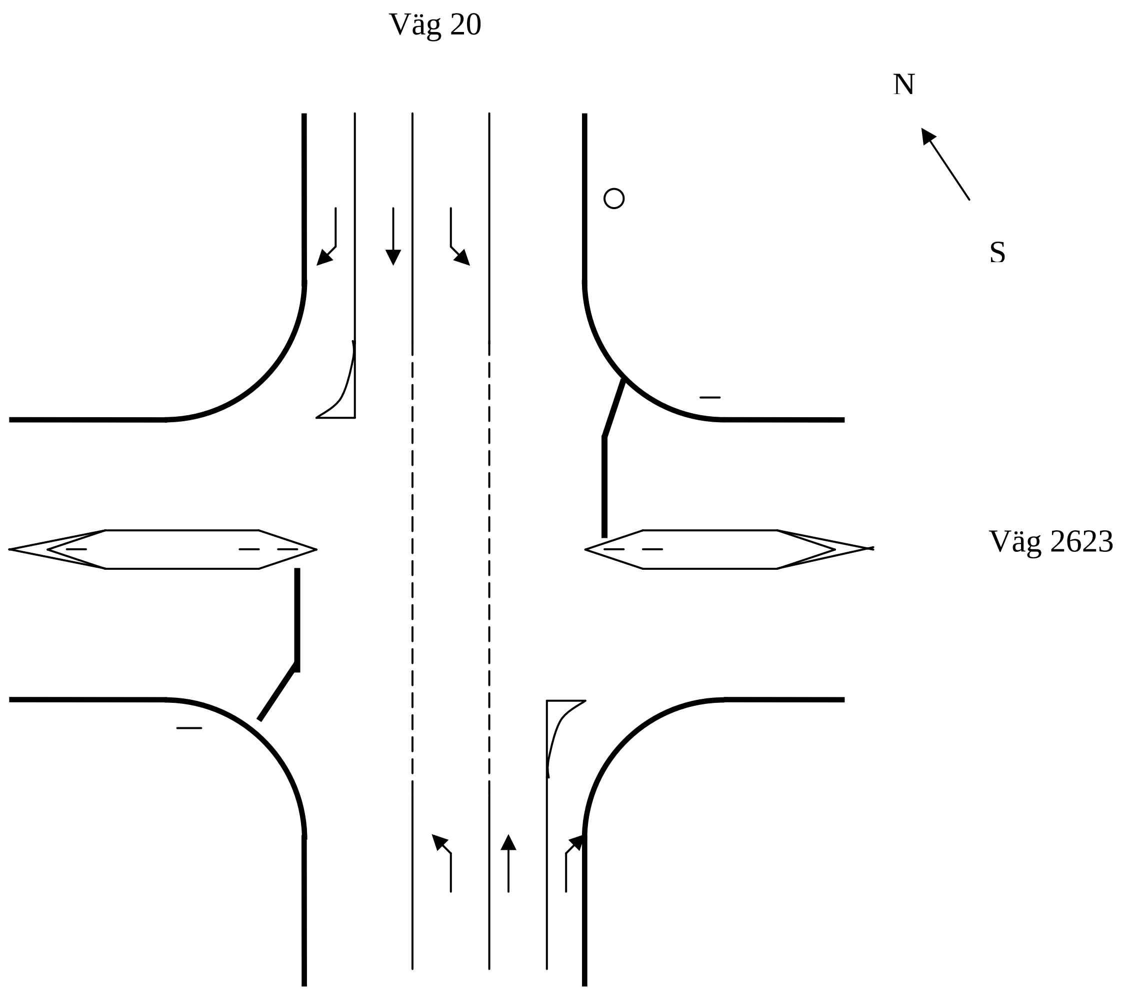
3 KORSNINGEN

Studieobjektet är en fyrvägskorsning på landsväg mellan E20 och väg 2623. Väg E20 har en bredd på 13 meter, medan den korsande sekundärvägen nr 2623 är 9 m bred. Korsningens läge visas i figur 1 nedan.



Figur 1. Situationskarta med landsvägskorsningen mellan E20 och väg 2623.

Båda huvudvägstillfarterna har ett körfält för varje riktning (högersväng, rakt fram, vänstersväng) genom korsningen. Medan sekundärvägen har ett körfält i korsningsmynningen. Stopplikt gäller på tillfarterna från sekundärvägarna. Fordonsflödena på huvudvägen är 8000ÅDT och 1200 / 1600ÅDT på sekundärvägarna. Den skyltade hastighetsgränsen är 70 km/tim på alla tillfarter. Huvudvägen utanför korsningsområdet har hastighetsbegränsning 90 km/t, medan sekundärvägen 2623 mot Jung har 70 km/t och samma väg sydöst mot Kvänum har 90 km/t hastighetsgräns utanför korsningsområdet. Längs huvudvägen före korsningen står varningsskylten ”Övrig fara” med tilläggstext ”Olycksdrabbad korsning”. Korsningens schematiska planritning visas i figur 2 nedan.



Figur 2.   Schematisk planritning för korsningen mellan E20 och väg 2623.

4 METOD OCH GENOMFÖRANDE

4.1 Konfliktobservationer

För att analysera trafiksäkerhetssituationen i korsningen har, den vid institutionen utvecklade, konflikttekniken använts (Hydén, 1987). Konflikttekniken grundas på sambandet mellan konflikter (olyckstillbud) av en viss allvarlighetsgrad och olyckor. En konflikt är en situation där två trafikanter befinner sig på kollisionskurs och en olycka hade inträffat om de båda inblandade hade fortsatt utan att ändra riktning eller hastighet.

Konflikttekniken är överlägsen olycksbaserade analyser vid analys trafiksäkerhetssituation vid enskilda latser:

- Antalet olyckor är ofta litet och därmed behäftade med stora slumpmässiga variationer. För att kunna använda antalet olyckor som mått på säkerheten måste man vänta många år för att kunna samla in "tillräckligt många" olyckor. Under en så lång tid kan förhållandena ändras på platsen, vilket gör det svårbedömt om de eventuellt uppvisade effekterna berodde på åtgärderna eller på andra orsaker. Med hjälp av konflikttekniken kan man samla in tillräckligt antal konflikter på en vecka för att kunna bedöma säkerhetsläget.
- Ett annat problem med analyser baserade på olyckor är att inte alla olyckor rapporteras.
- En trafiksäkerhetsåtgärd är ofta införd p.g.a. att antalet olyckor på platsen var relativt stort. Om antalet olyckor året efter blir mindre kan det bero på regressionseffekten, d.v.s. att antalet återgått till det "normala" medelvärdet.
- Konfliktstudierna ger mer detaljerad information än olycksanalyserna, bl a beträffande orsaker till situationernas uppkomst och om de inblandade trafikanternas hastigheter.

Konflikterna registreras av speciellt tränade observatörer direkt i trafiken. Förutom att alla grundläggande data (inblandade trafikanter och deras färdriktningar, hastigheter etc.) registrerades gjordes också en beskrivning av händelseförloppet som föregick konflikten och de faktorer som påverkade händelseutvecklingen enligt observatörens bedömning. Formulär där registreringen gjordes finns i bilaga 1.

Observationstider har varit måndag till fredag kl. 7:00-9:00, 10:00-11:30, 13:30-15:00, 16:00-18:00. Den totala observationstiden uppgick till 35 timmar.

4.2 Hastighetsmätningar med radar

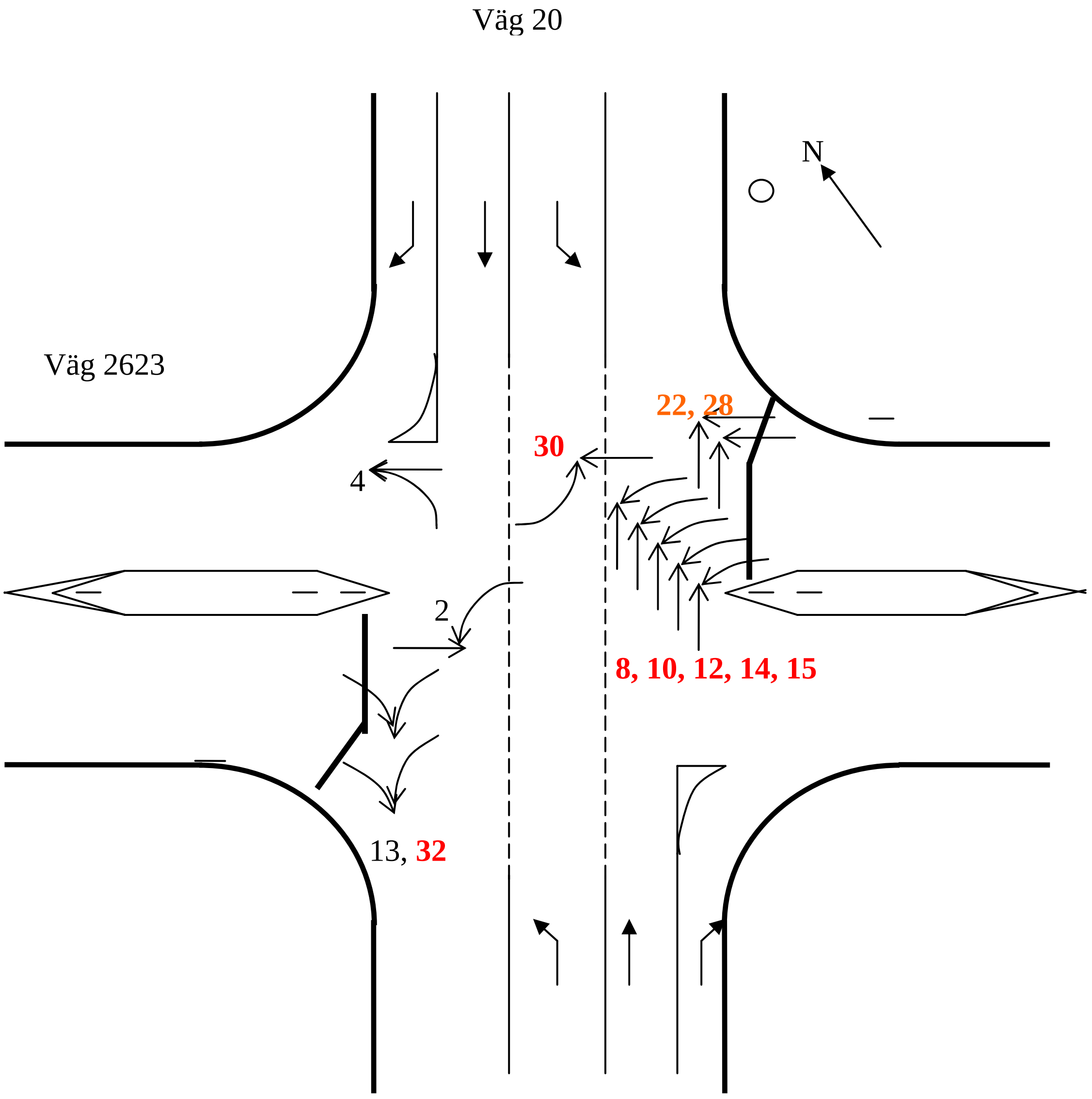
Hastigheter mättes med radarpistol på slumpmässigt valda "fria" fordon, dvs fordon som "fritt" kan välja sin hastighet längs huvudvägen. Hastigheterna mättes rakt framifrån med c:a 100 fordon per mätsnitt. Mätsnitten låg c:a 100 meter före korsningen.
Mättider har varit måndag till fredag mellan kl. 13:00 och kl. 18:30.

Det genomfördes även hastighetsmätningar med tidtagarur (vissa notoriska fortkörare kan ha radarvarnare i bilen och därmed kommer ej med i radarmätningarna).

5 RESULTAT

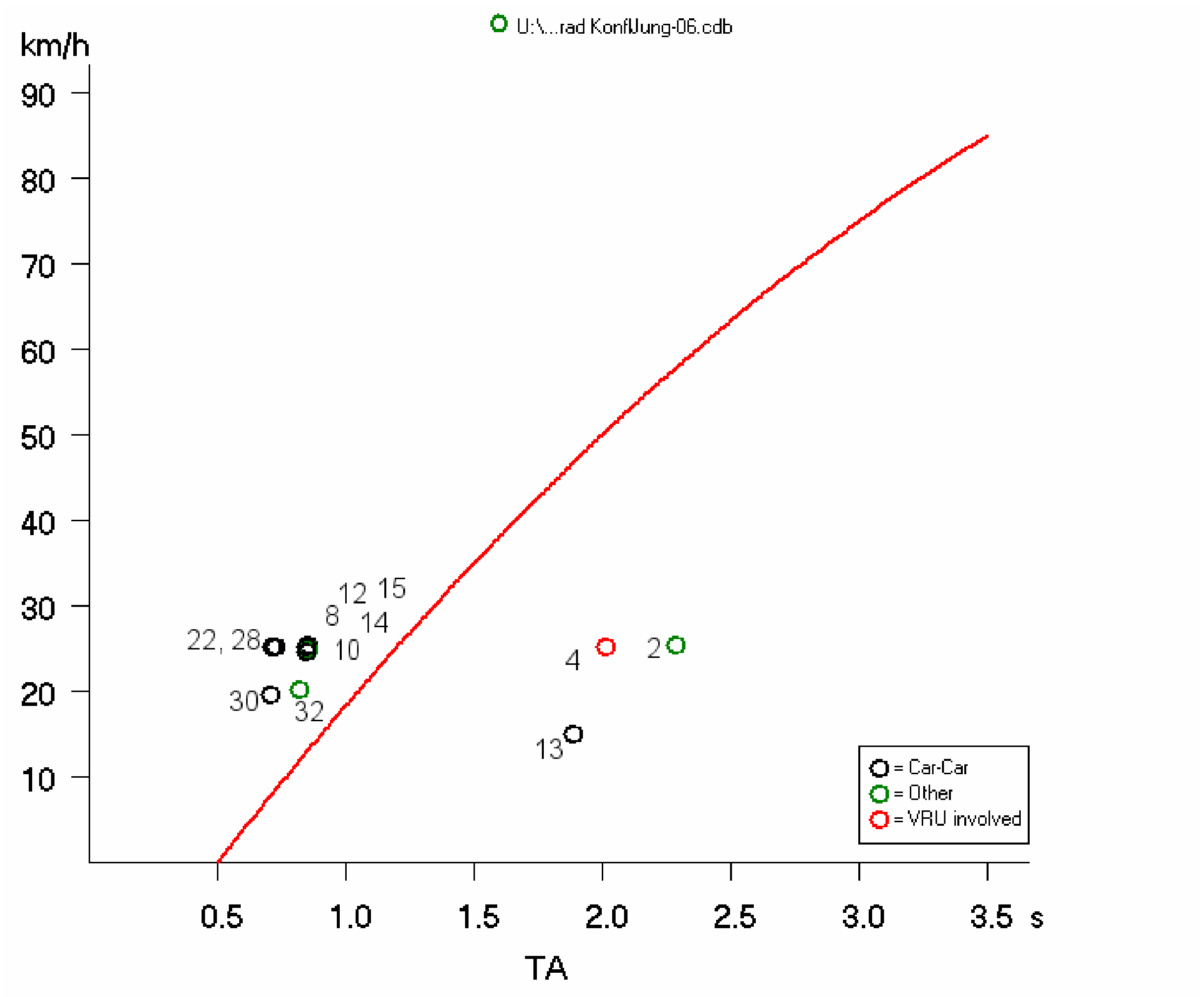
5.1 Konflikter

Tolv konflikter registrerades under observationsperioden bestående av 35 timmar. En sammanställning av observerade konflikttyperna visas i figur 3 nedan (Varje enskild konfliktsituation presenteras i bilaga 2). Som det framgår av figur 3 involverade alla konflikter ett fordon som kör ut från östra sekundärvägen. Majoriteten (7 st) av konflikterna är av typen då ett fordon kör ut från östra sekundärvägen (antingen med korsande eller vänstersvängande riktning) och det kommer i konflikt med ett fordon som passerar korsningen längs primärvägen från sydväst. Näst största konflikttyp (4st) är då svängande och raktframkörande fordon från var sitt sekundärvägstillfart konkurrerar om den korta tidluckan flödet längs primärvägen ger och då kommer i konflikt med varandra. Det förekommer även en konflikt där ett vänstersvängande fordon från primärvägen kommer i konflikt med ett fordon som kör ut från sekundärvägen på högra sidan (konflikt nr 4).



Figur 3. Sammanställning av konflikterna (Numren på allvarliga konflikter är i fet stil).

Nio konflikter var allvarliga, se diagram i figur 4 nedan. Allvarliga konflikter definieras av att den tidsmarginal som återstår då avväjningen påbörjas är högst lika med bromstiden vid häftig inbromsning på lätt fuktig vägbana plus en halv sekund. Konflikthastigheterna i dessa allvarliga konflikter verkar vara relativt låga, vilket beror på att den avvärjande parten är föraren som kör ut från sekundärvägen. Detta innebär inte att en olycka av samma typ inte skulle leda till mycket allvarliga personskador eftersom det andra involverade fordonet har en hög hastighet längs huvudleden och en kollision mellan dessa fordon kunde leda till stora skador.



Figur 4. Allvarlighetsgraden av konflikterna (konflikterna till vänster och ovan den röda gränslinjen är allvarliga).

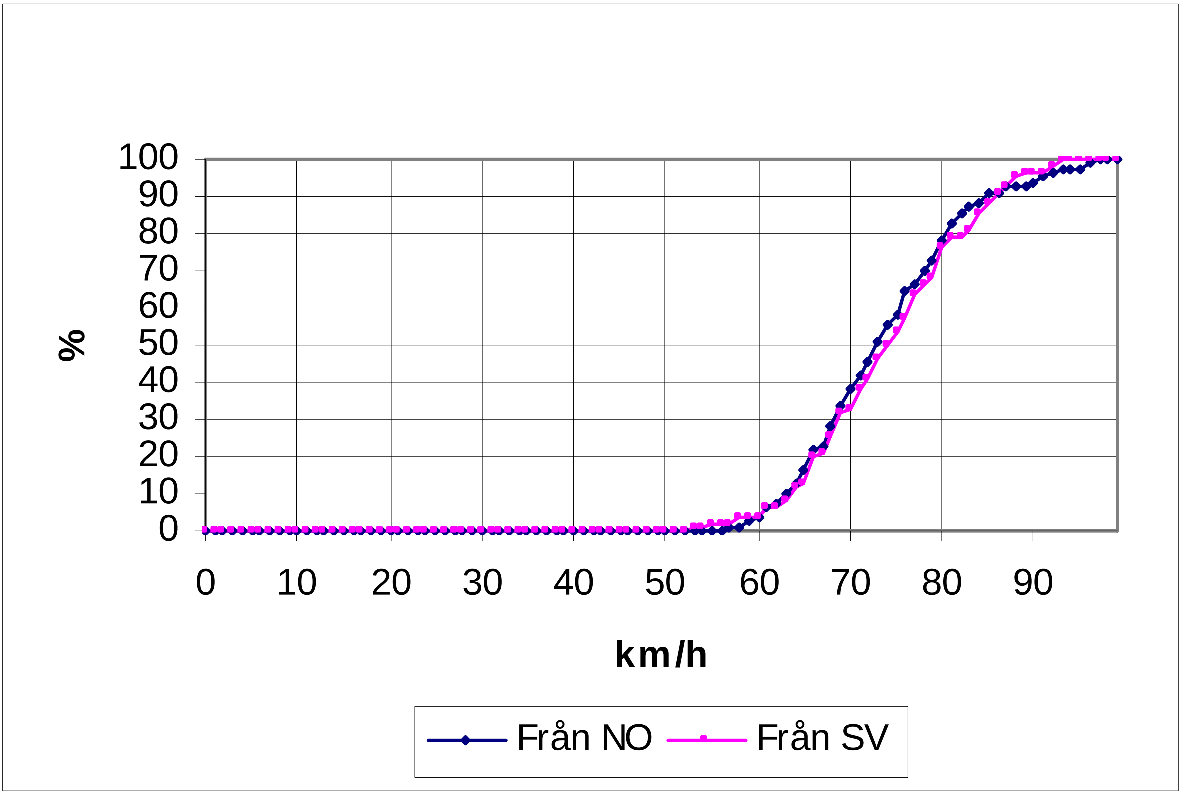
5.2 Hastigheter

Medelhastighet och dess standaravvikelse visas i tabell 1 och hastighetsfördelningskurvan i figur 5 nedan.

Tabell 1. Medelhastighet och standaravvikelse för fria fordon längs huvudvägen c:a 100 meter före korsningen.

	Från NO	Från SV
Antal mätningar	100	100
Medelhastighet (km/h)	75,1	75,7
Standardavvikelse (km/h)	8,78	8,71
Medianhastighet (km/h)	74,0	75,5

Som det framgår av tabell 1 och figur 5 är fortkörning en vanlig företeelse genom korsningen. Majoriteten av förare kör över hastighetsgränsen (70 km/t) genom korsningen. Mer än 65 % av förarna kör med en hastighet över hastighetsgränsen och 50 % av förare kör fortare än 74 km/h.



Figur 5. Hastighetsfördelningsdiagram för fria fordon längs huvudvägen c:a 100 meter före korsningen.

Hastighetsmätningarna med tidtagarur visade extrema hastigheter. Det mättes över 30 fordon som körde med en hastighet över 100 km/h, den högsta hastigheten som uppmättes var 140 km/h genom korsningen.

5.3 Övriga observationer

Det förekommer frekventa köer på sekundärvägstillfarten, upp till 15 bilar i kö på östra tillfarten och upp till 7 bilar på västra tillfarten. Väntetiderna för korsande trafik är långa p.g.a. bl.a.:

- Mycket trafik på E20,
- Många tunga fordon på E20,
- Höga hastigheter på E20,
- Många tunga fordon på korsande väg. Utfart från Preem till E20 har mycket långtradare och trailers men även Bussar är vanligt förekommande.
- En del fordonsförare har lång reaktionstid innan de korsar, ibland beroende på användande av mobiltelefon.

E20 utgör en barriär mellan Jung och Kvänum (boende och arbete/boende, arbete och skola). Enstaka korsande cyklister och någon fotgängare förekommer i korsningen

Lastbilsförare som pratar i mobiltelefon är mer regel än undantag. Det förekommer att lastbilsförare har ena foten i vilställning uppe på instrumentbrädan när de passerar korsningen.

När fordon kör ut från sekundärvägen efter stopp och svänger vänster och det står en bil mittemot vid stopplinjen förekommer det frekvent att det fordonet kör först fast det skall lämna företräde åt rakt fram korsande fordon från motsatta hållet.

Många fordon stannar aldrig helt vid stopplinjen. De bromsar ner fordonet till nästan stilla eller rullar sakta förbi stopplinjen. Bilar och bussar kör ut en bit vid stoppet när passerande fordon på huvudled närmar sig från vänster som om de inte först observerat det ankommande fordonet, och då de upptäcker detta stoppar planerad rörelse.

Vid utfart från sekundärvägen kör ofta högersvängande upp jämsides med fordon med annan körriktning för att på så sätt komma ut på huvudvägen. Vägrenen används då fullt ut.

6 DISKUSSION, SLUTSATSER

Tolv konflikter (9 allvarliga) registrerades under observationsperioden bestående av 35 timmar. Alla konflikter involverade ett fordon som kör ut från östra sekundärvägen (antingen med korsande eller vänstersvängande riktning).

Fortkörning är en vanlig företeelse genom korsningen. Det förekommer hastigheter upp till 140 km/h genom korsningen.

Det är frekventa köer på sekundärvägstillfarterna eftersom väntetiderna för korsande trafik är långa. Lastbilsförare som pratar i mobiltelefon är mer regel än undantag. Det förekommer att lastbilsförare har ena foten i vilställning uppe på instrumentbrädan när de passerar korsningen. När fordon kör ut från sekundärvägen efter stopp och svänger vänster och det står en bil mittemot vid stopplinjen förekommer det frekvent att det fordonet kör först fast det skall lämna företräde åt rakt fram korsande fordon från motsatta hållet. Många fordon stannar aldrig helt vid stopplinjen. Vid utfart från sekundärvägen kör ofta högersvängande upp jämsides med fordon med annan körriktning för att på så sätt komma ut på huvudvägen.

Den fortsatta studien av trafiksäkerhetssituationen i denna korsning bör fokusera på situationerna där från östra sidovägen utkörande fordon kommer i konflikt med fordon antingen kommande från söder på huvudvägen eller fordon som kör ut från motsatta sekundärvägstillfart.

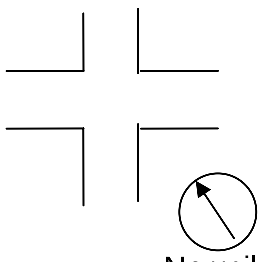
REFERENSER

Hydén, C. (1987) "The development of a method for traffic safety evaluation: The Swedish Traffic Conflicts Technique". Institutionen för trafikteknik, LTH.

BILAGA 1. FORMULÄR FÖR REGISTRERING AV KONFLIKTER

Observatör: ________________ Datum:___________ Tid:____________ Löpnr:_____

Stad: Vårgårda



Norrpil

Korsning: Väg 20/2623 (vid Preem bensinstation SO Jung)

Väderlek: Sol ☐ Mulet ☐ Regn ☐

Vägbana: Torr ☐ Våt ☐

Tidsperiod ☐ 7:00-9:00 ☐ 10:00-11:30 ☐ 13:30-15:00 ☐ 16:00-18:00

	Primär Trafikant I	Primär Trafikant II	Sekundär Trafikant III
Privatbilist	☐	☐	☐
Cyklist	☐	☐	☐
Fotgängare	☐	☐	☐
Övrig	_________	_________	_________
Kön (fotg)	M ☐ K ☐	M ☐ K ☐	M ☐ K ☐
Ålder (fotg)	_________	_________	_________
Hastighet	_____ km/h	_____ km/h	_____ km/h
Avst. till kollisionspunkten	_____ m	_____ m	
TO värde	_____ sek	_____ sek	
Avvärjande manöver			
Inbromsning	☐	☐	
Väjning	☐	☐	
Acceleration	☐	☐	
Möjlighet att väja	ja ☐ nej ☐	ja ☐ nej ☐	
		ja ☐ ☐ nej ☐ ☐	

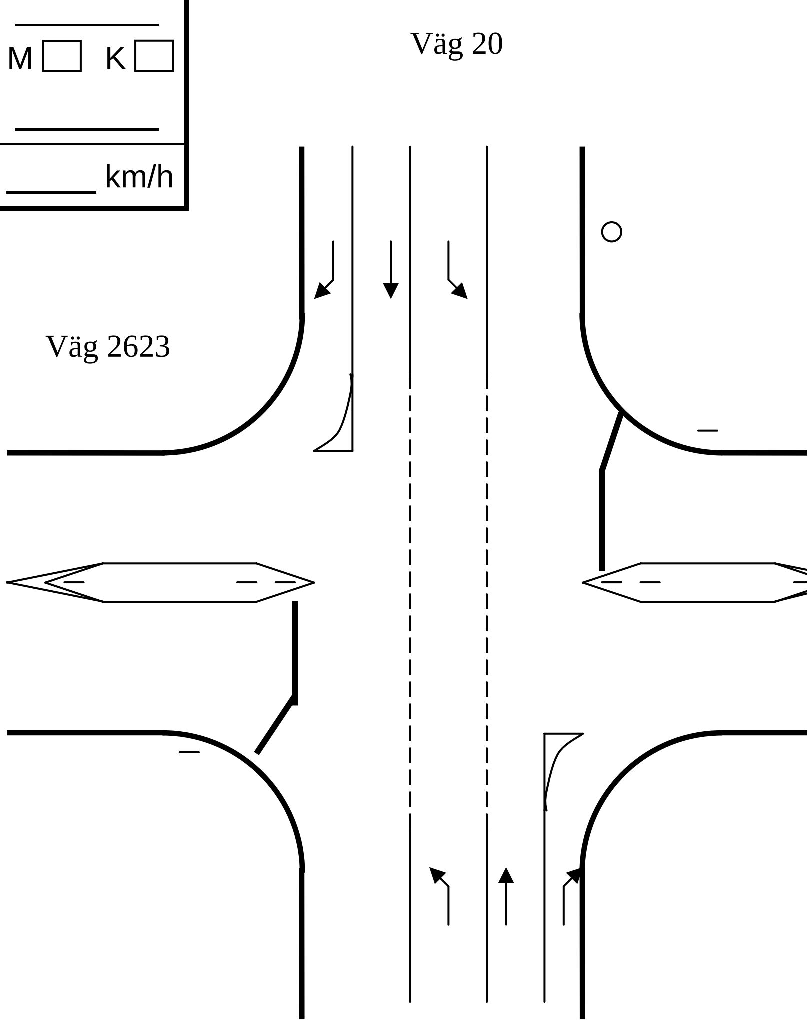
Beskrivning av situationen:

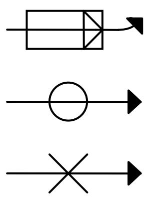
Fortsättning på nästa sida ☐ ⇒

Skiss över de inblandades läge.

Markera din position med ⊗

(Markera videokamerans position med)





Privatbil, Lastbil, Buss.

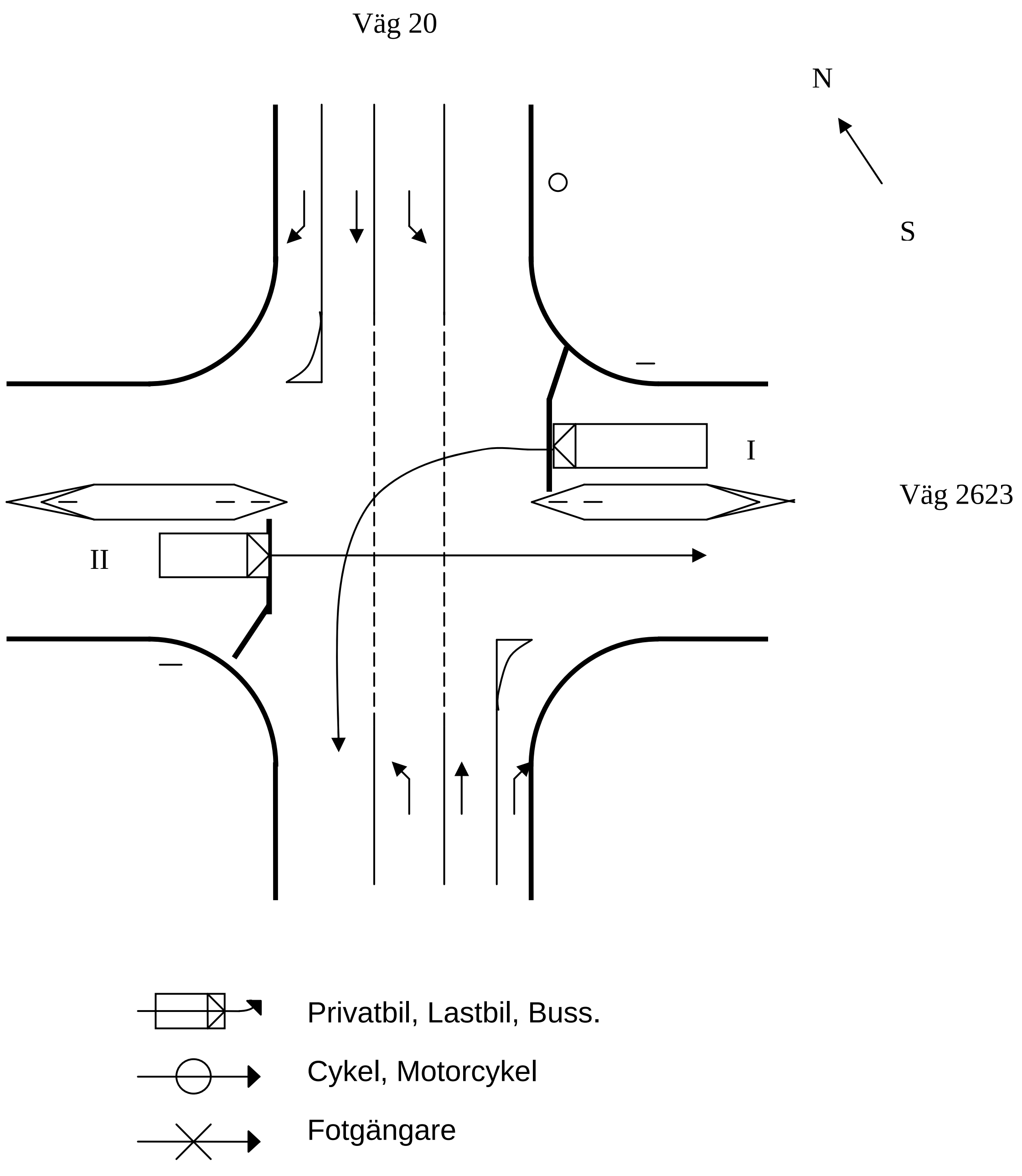
Cykel, Motorcykel

Fotgängare

BILAGA 2. ENSKILDA KONFLIKTSITUATIONER

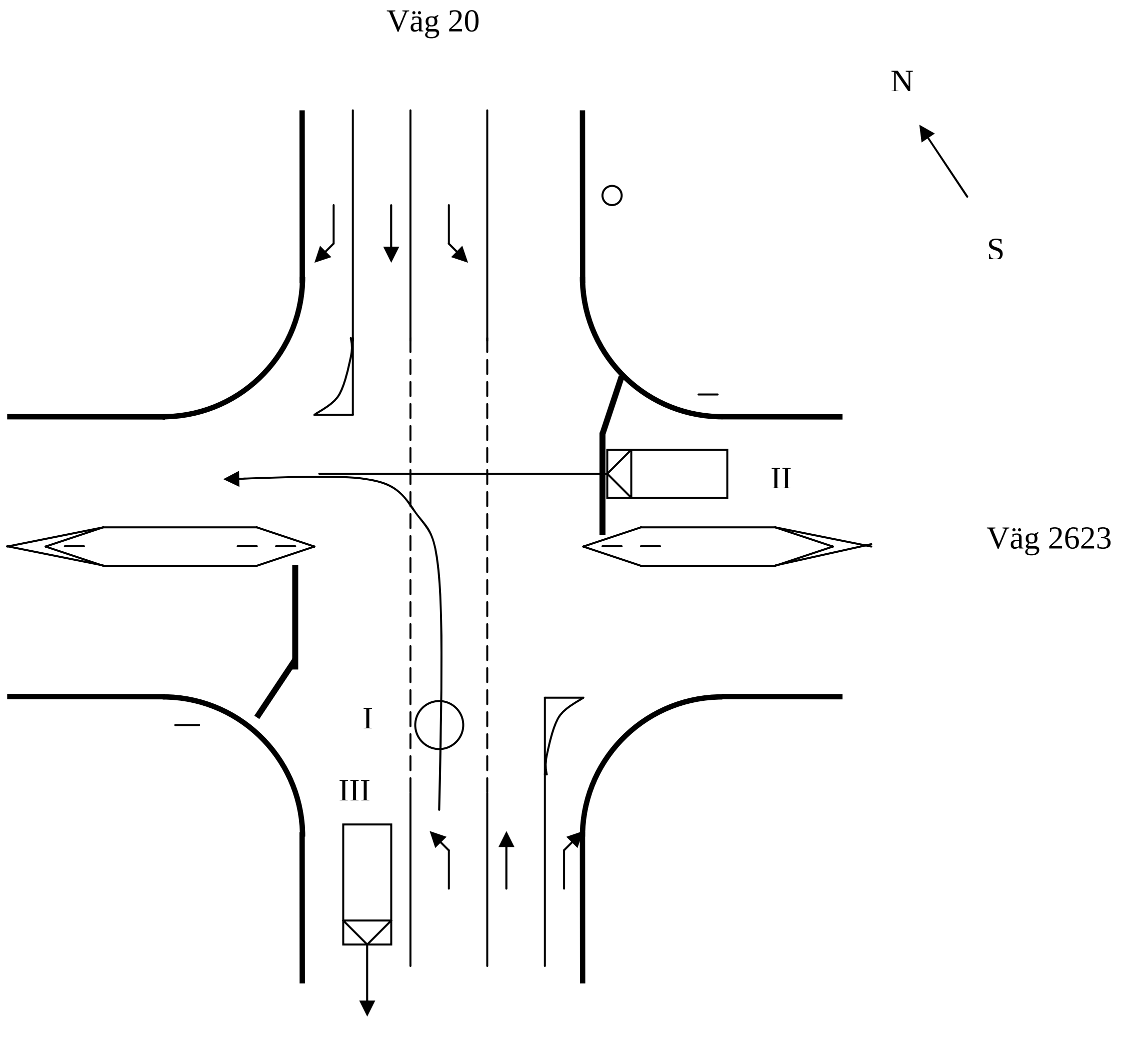
Konflikt, löpnr 2:
Vänstersvängande personbil (I) från sidoväg med stopplikt kör utan att stanna,
buss (II) från sidoväg med stopplikt står vid stopplinje i färd med att göra vänstersväng.

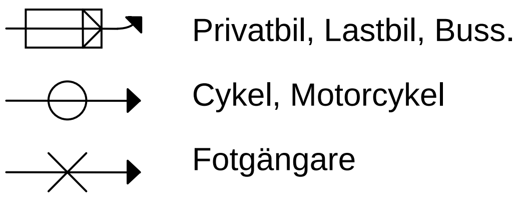
Fordon	I	II
Hastighet [km/h]	25	25
Avstånd [m]	16	4
TO-värde	2,3	



Konflikt, löpnr 4:
Rakt-fram-körande personbil (II) från sidoväg med stopplikt, väjer ej för till vänster väntande tung motorcykel med passagerare vänstersvängande på huvudled.

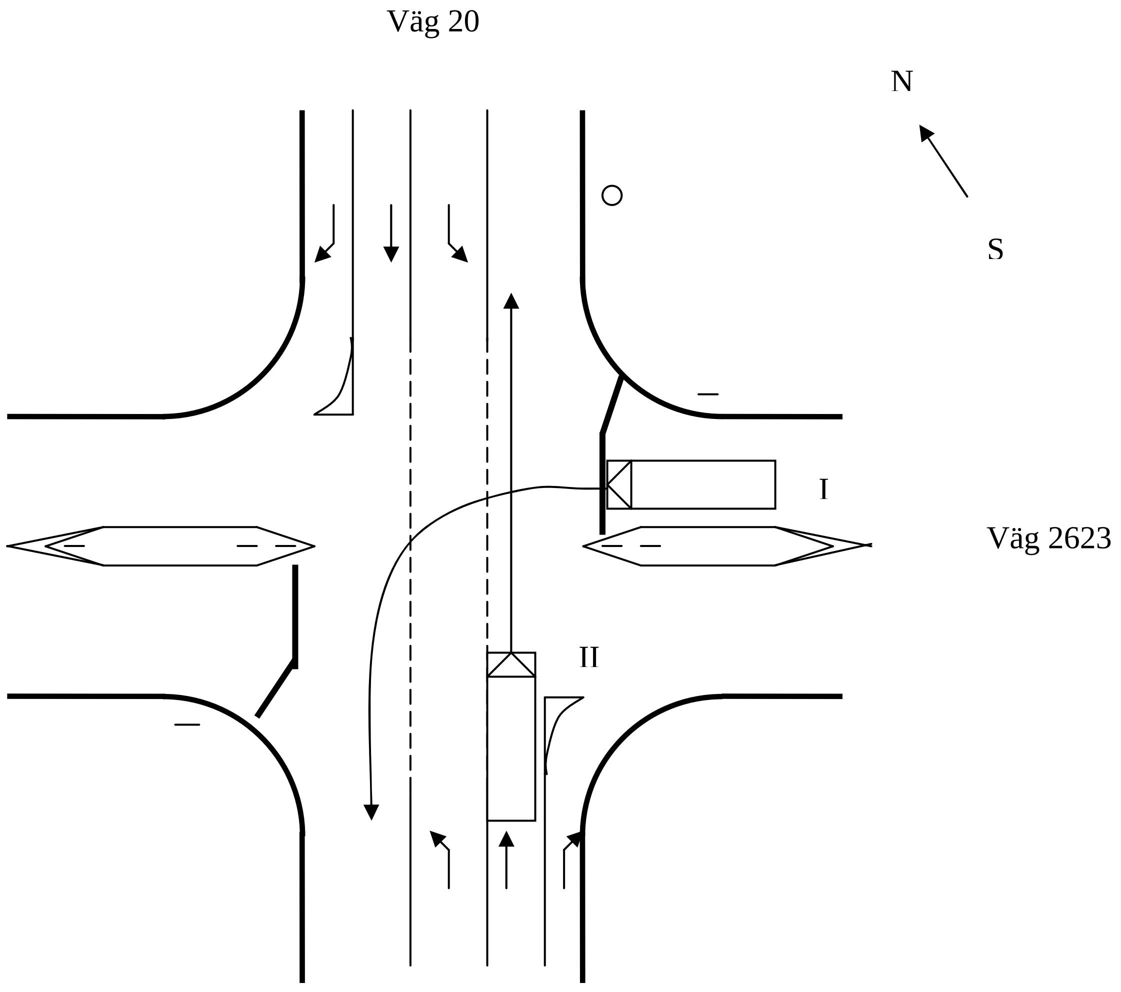
Fordon	I	II
Hastighet [km/h]	25	25
Avstånd [m]	14	16
TO-värde	2,0	

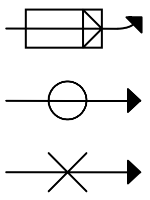




Konflikt, löpnr 8:
Vänstersvängande skolbuss (I) står på sidoväg med stopplikt, ser ut som ej ser rakt-fram-körande från vänster kommande tung lastbil (II) på huvudled.

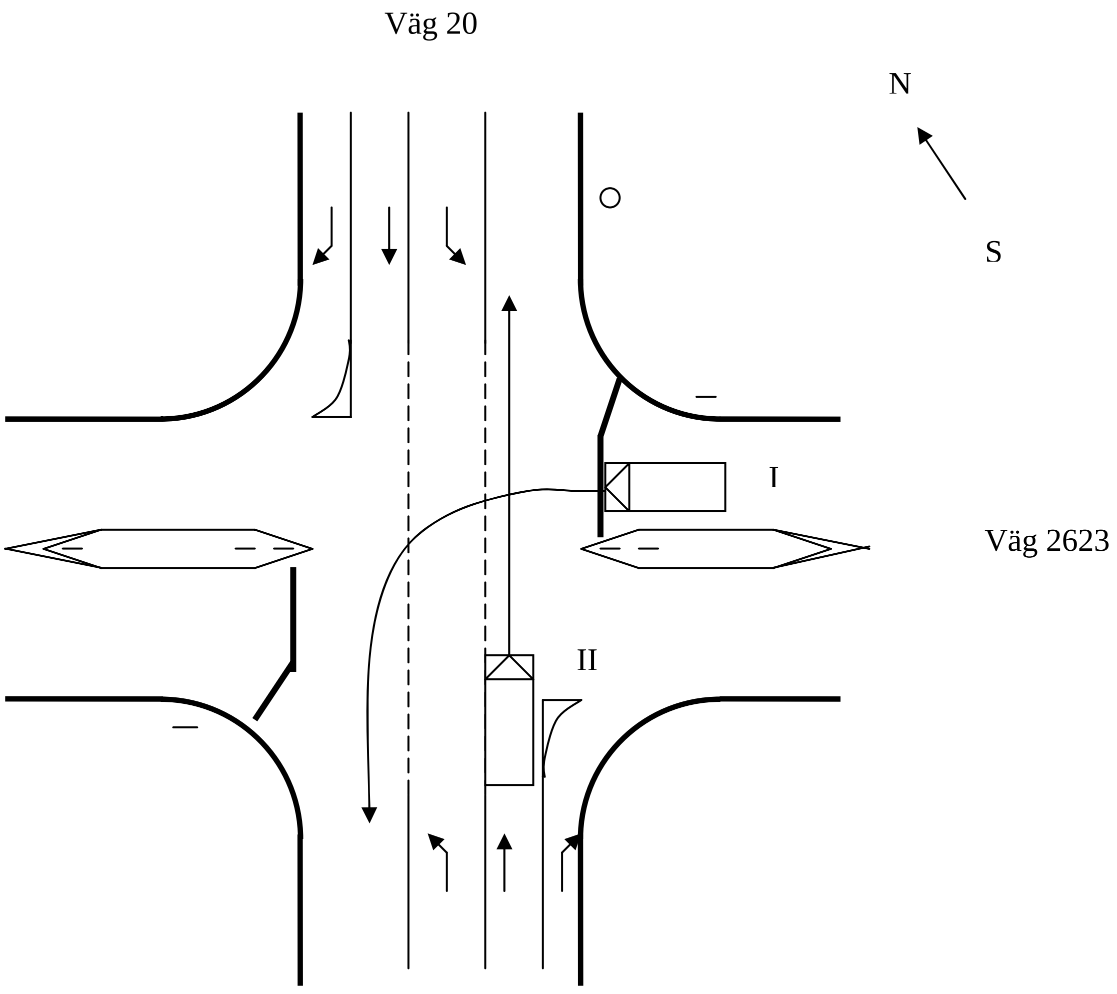
Fordon	I	II
Hastighet [km/h]	25	75
Avstånd [m]	6	40
TO-värde	0,9	

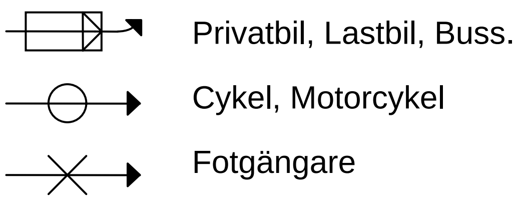




Konflikt, löpnr 10:
Vänstersvängande personbil (I) från sidoväg med stopplikt kör ut efter lastbil.
Det kommer en rakt-fram-körande personbil från vänster (II) på huvudled i relativt hög hastighet.

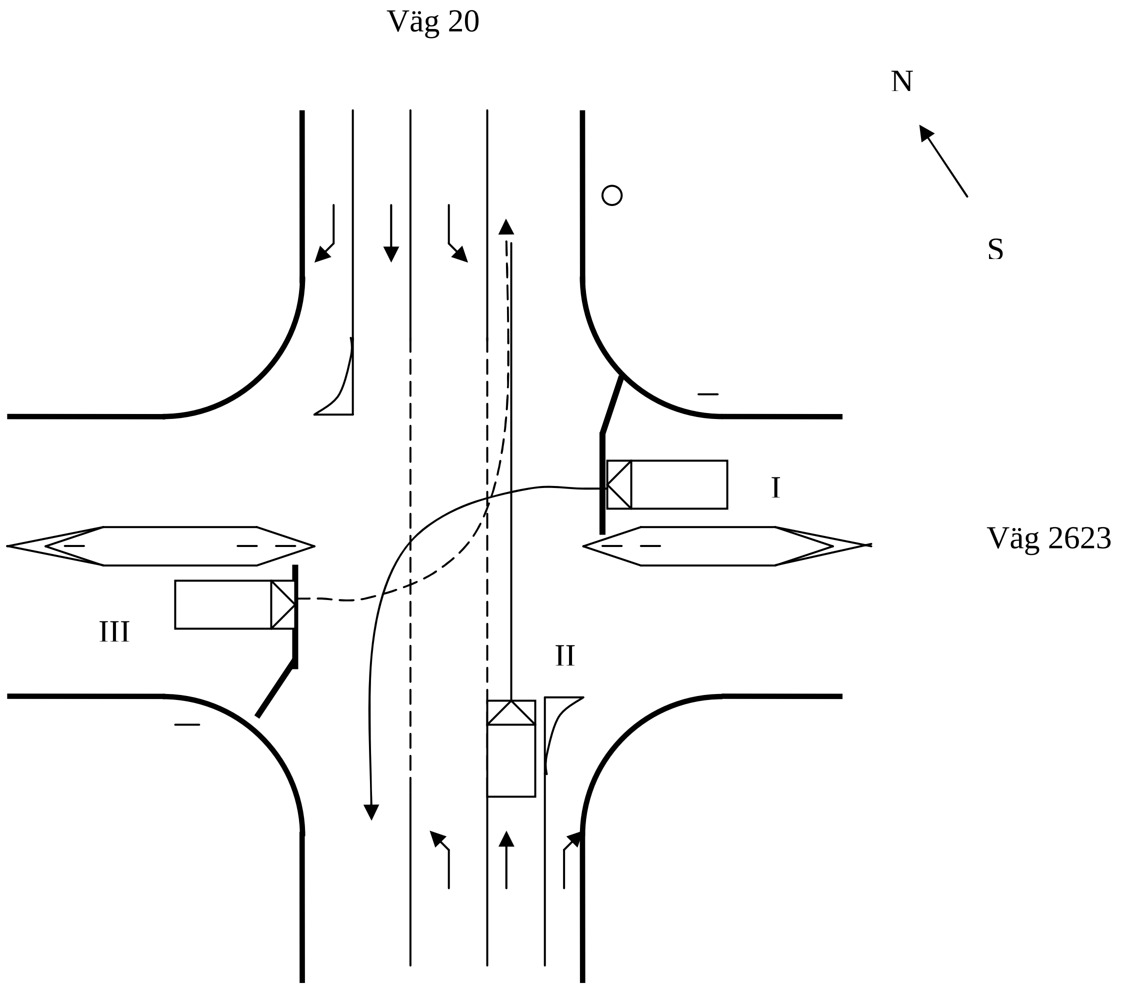
Fordon	I	II
Hastighet [km/h]	25	90
Avstånd [m]	6	60
TO-värde	0,9	

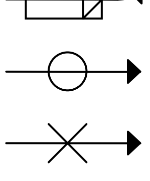




Konflikt, löpnr 12:
Vänstersvängande skåpbil från sidoväg med stopplikt kör ut, rakt-fram-körande personbil från vänster på huvudled kommer i relativ hög fart. Personbil från motsatt håll på sidoväg skall svänga vänster.

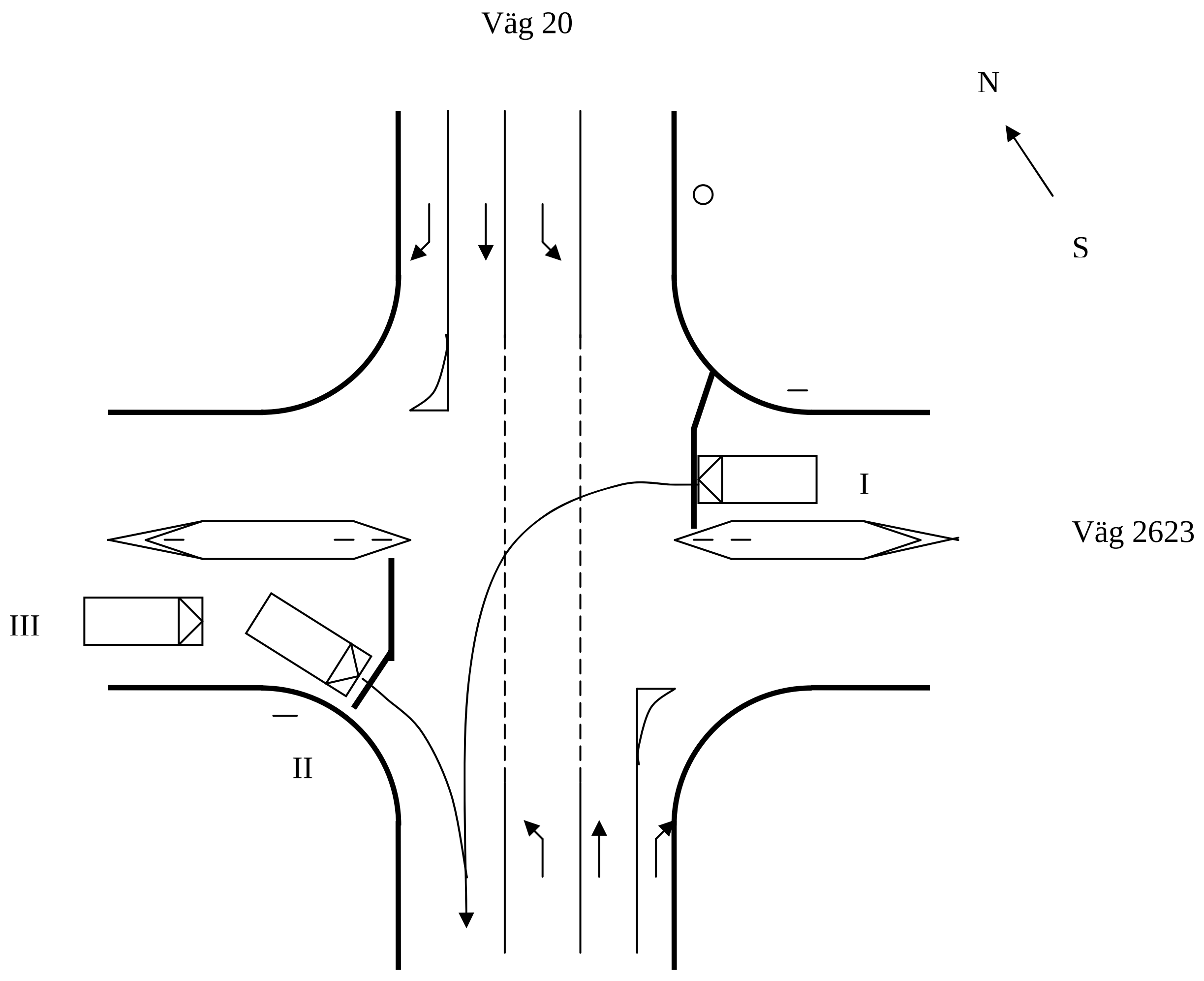
Fordon	I	II	III
Hastighet [km/h]	25	90	0
Avstånd [m]	6	70	-
TO-värde	0,9		-

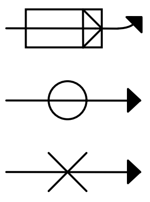




Konflikt, löpnr 13:
Stor stadsjeep (I) svänger vänster från sidoväg med stopplikt, högersvängande personbil (II) på motsatt sidoväg med stopplikt börjar också köra, men saktar in, stadsjeepen kör först.

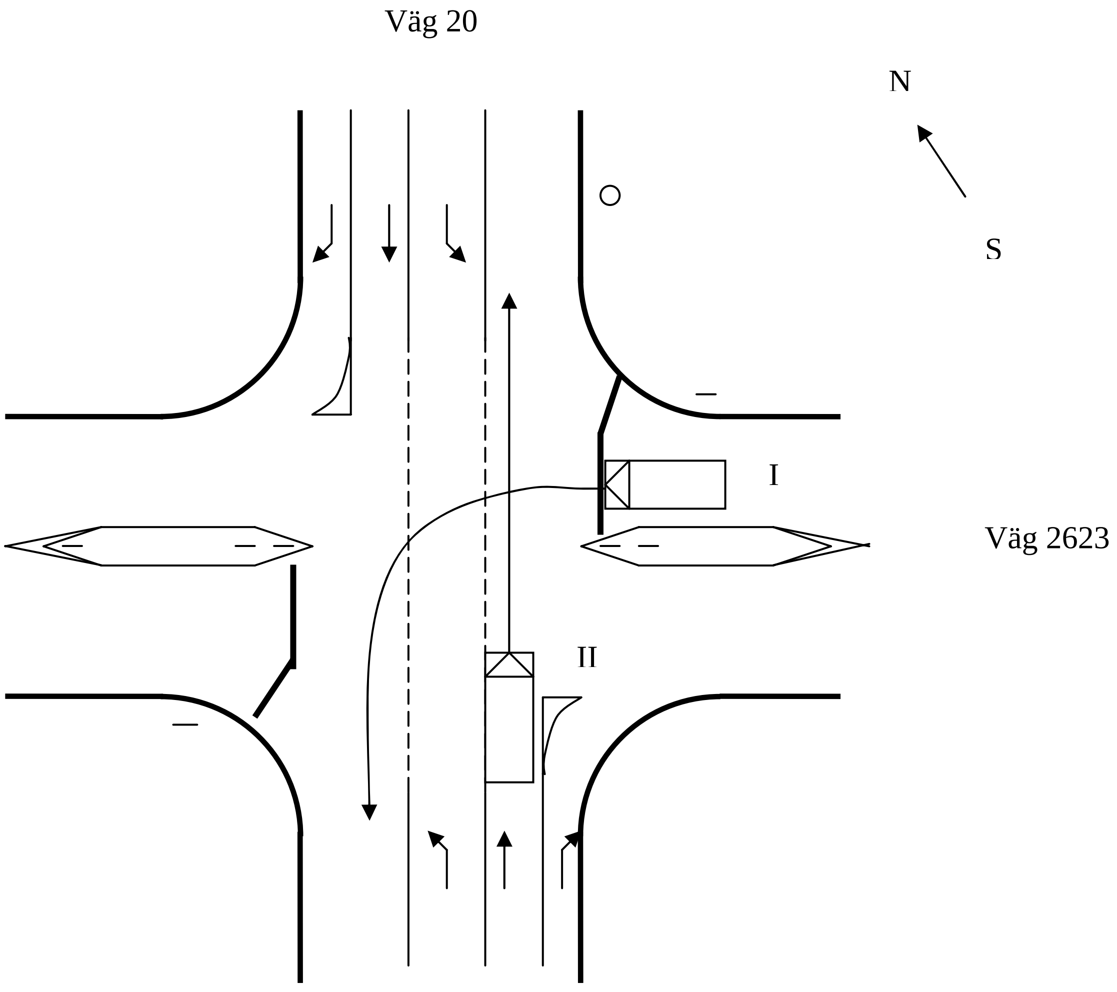
Fordon	I	II	III
Hastighet [km/h]	30	15	0
Avstånd [m]	28	8	-
TO-värde		1,9	-

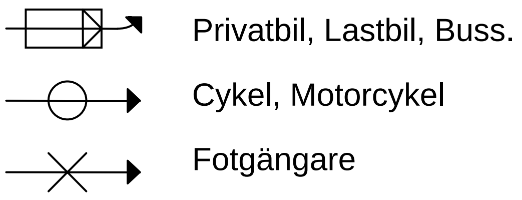




Konflikt, löpnr 14:
Vänstersvängande personbil (I) från sidoväg med stopplikt kör ut, kommer rakt-fram-körande personbil (II) från vänster på huvudled i relativt hög hastighet.

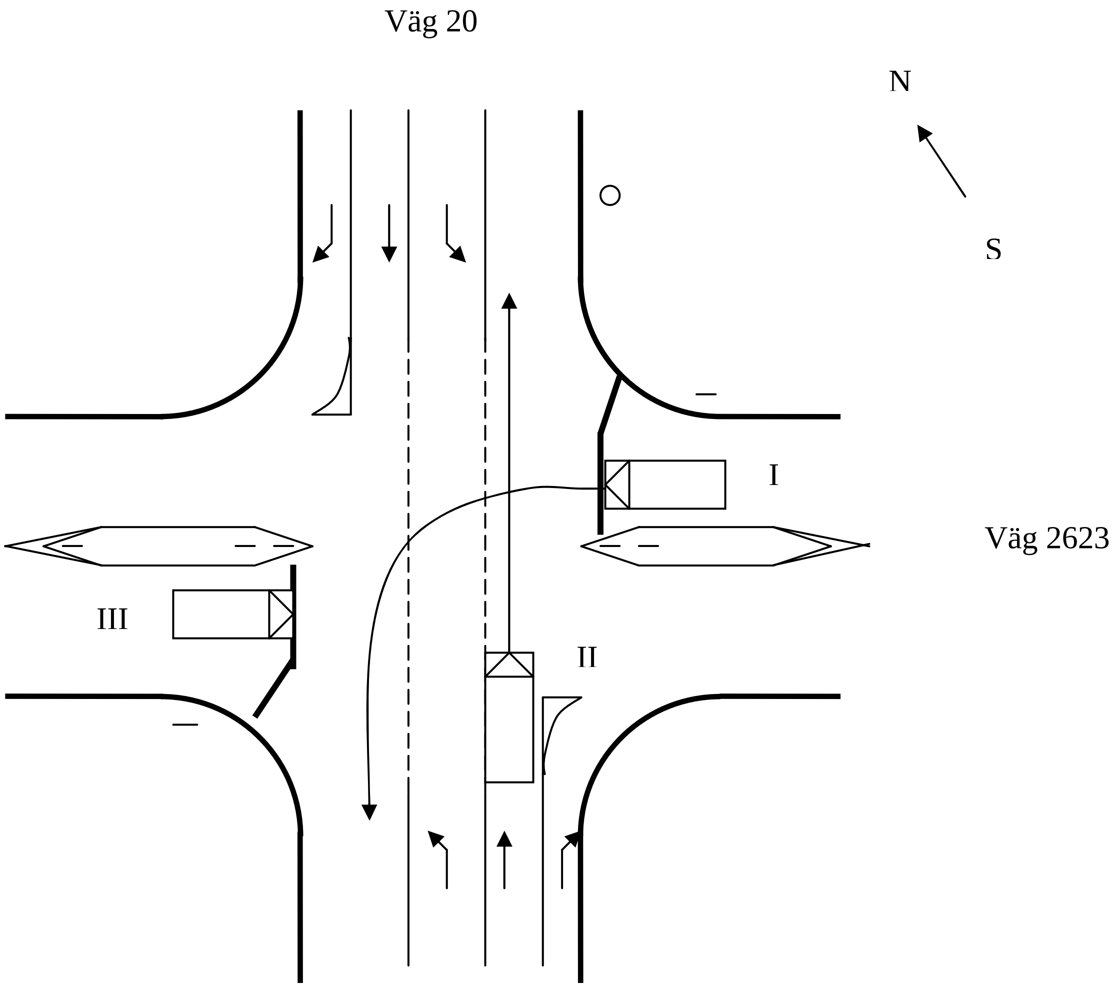
Fordon	I	II
Hastighet [km/h]	25	90
Avstånd [m]	6	60
TO-värde	0,9	

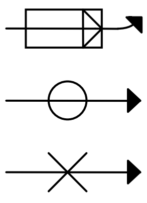




Konflikt, löpnr 15:
Vänstersvängande personbil (I) från sidoväg med stopplikt kör ut, kommer rakt-fram-körande personbil (II) från vänster på huvudled i relativt hög hastighet. Skåpbil (III) står vid stopplinje på motsatt sidoväg.

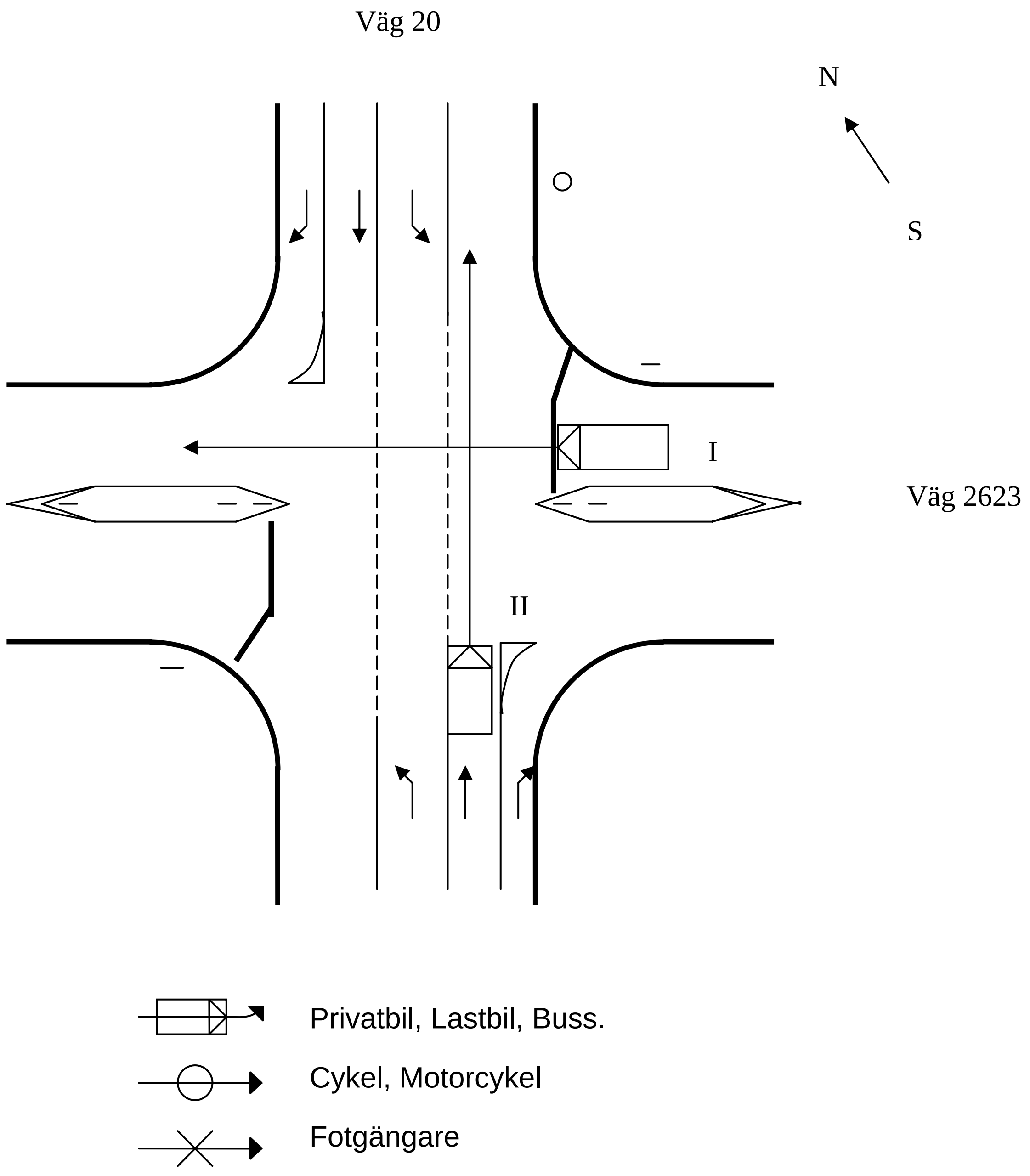
Fordon	I	II	III
Hastighet [km/h]	25	90	0
Avstånd [m]	6	60	-
TO-värde	0,9		-





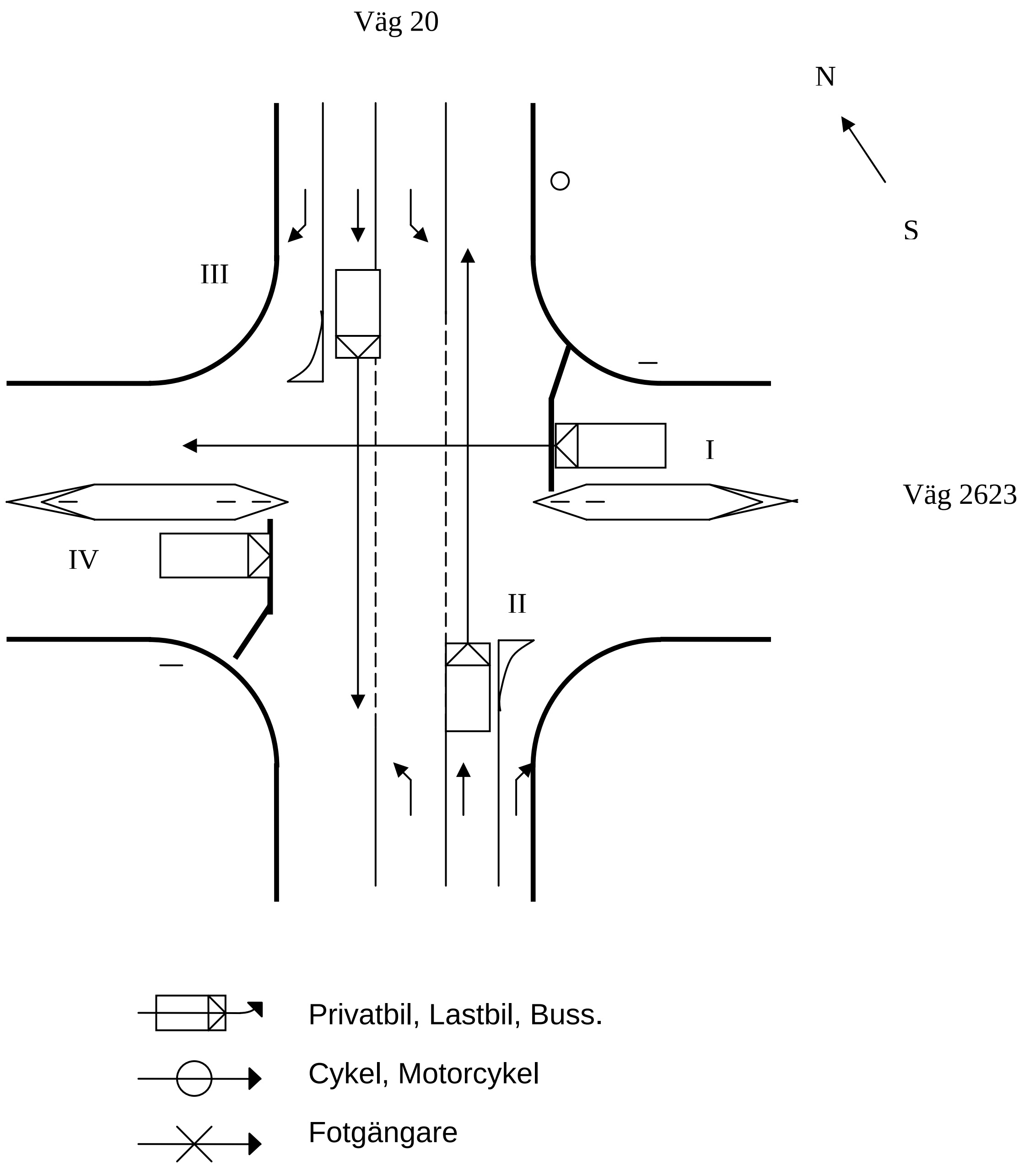
Konflikt, löpnr 22:
Personbil (I) på sidoväg med stopplikt, som skall köra rakt fram, verkar först inte se rakt-fram-körande personbil (II) som kommer från vänster på huvudled.

Fordon	I	II
Hastighet [km/h]	25	70
Avstånd [m]	5	60
TO-värde	0,7	



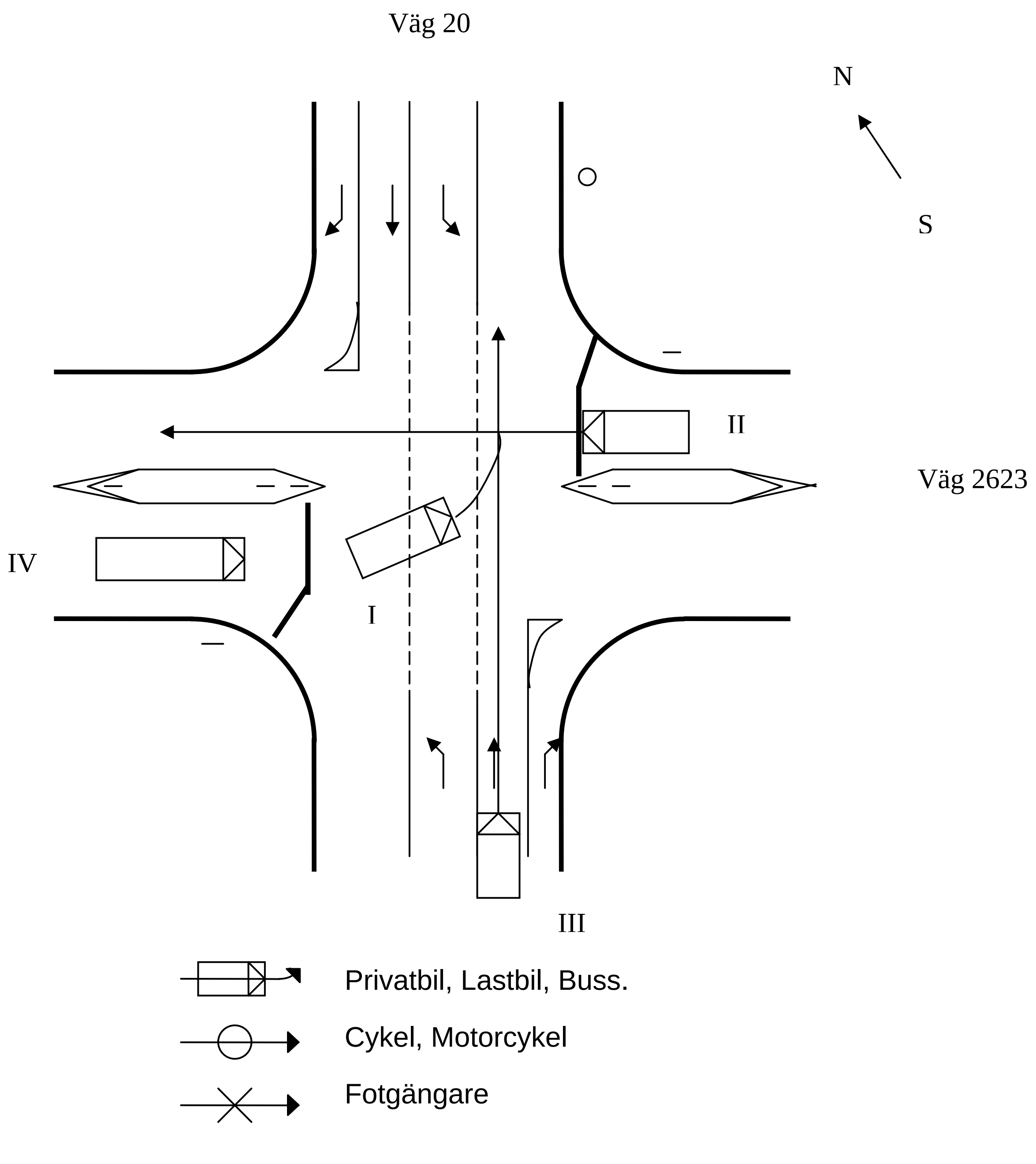
Konflikt, löpnr 28:
Personbil (I) står vid stopplinje på sidoväg med stopplikt, skall köra rakt fram. Ser ut som tänkt köra, när det kommer rakt-fram-körande personbilar (II) och (III) från höger och vänster på huvudled.

Fordon	I	II	III
Hastighet [km/h]	25	70	70
Avstånd [m]	5	60	-
TO-värde	0,7		-



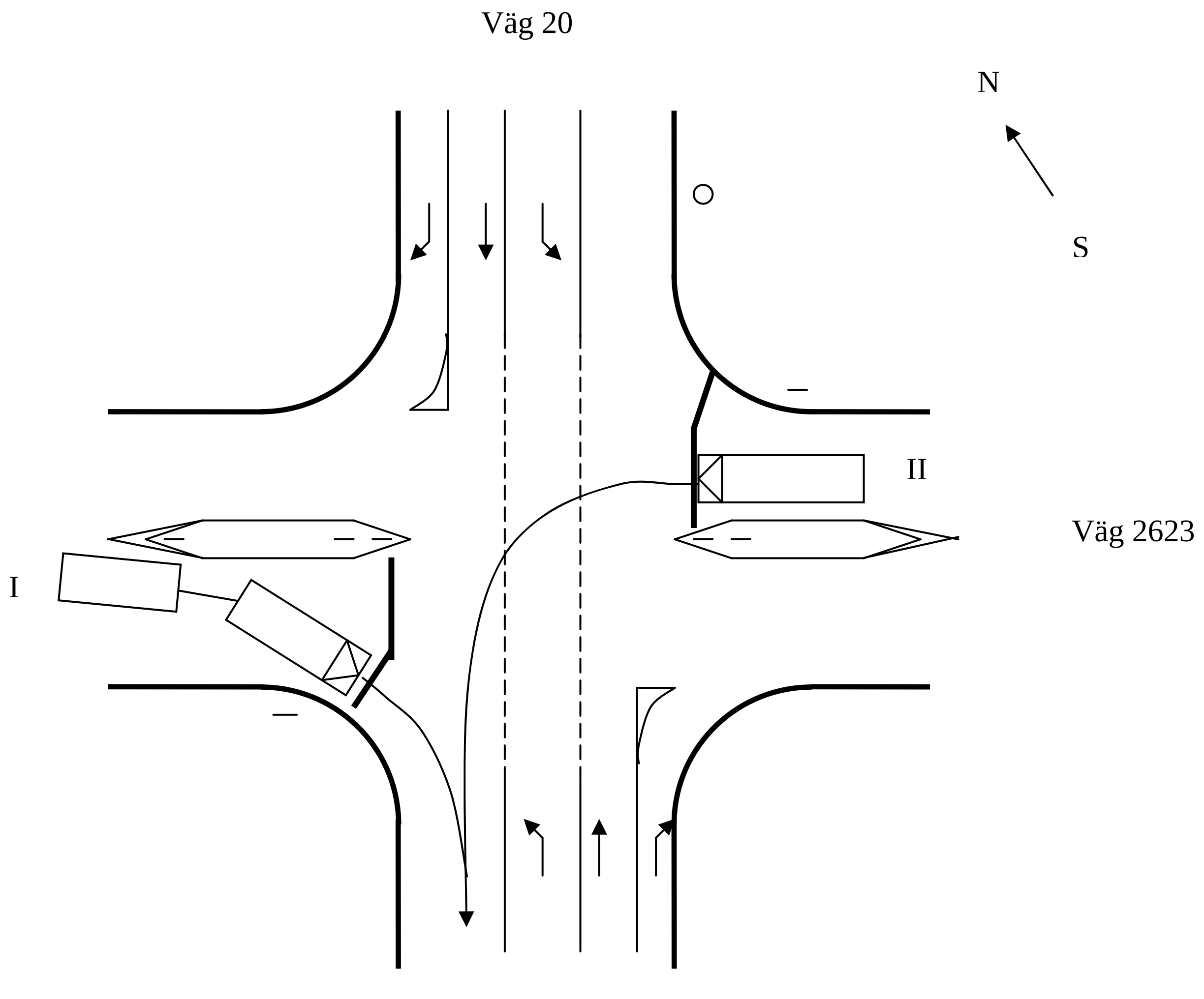
Konflikt, löpnr 30:
Rakt-fram-körande personbil (II) börjar röra på sig. Från motsatt sidoväg med stopplikt kommer
vänstersvängande personbil (I), som försöker köra först men stannar. Den andra bilen börjar backa
tillbaka till stopplinjen. Det kommer rakt-fram-körande personbil (III) sydöst ifrån på huvudled.

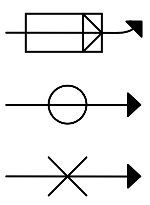
Fordon	I	II	III
Hastighet [km/h]	20	20	70
Avstånd [m]	4	4	60
TO-värde		0,7	



Konflikt, löpnr 32:
Lastbil med släp (I) svänger höger från sidoväg med stopplikt. Buss (II) svänger vänster från motsatt sidoväg med stopplikt. Lastbil bromsar, stannar då buss fortsätter mot lastbil.

Fordon	I	II
Hastighet [km/h]	20	20
Avstånd [m]	4,5	20
TO-värde	0,8	





Driver Coordination in Complex Traffic Environments

Linda Renner
Ph.D. Student
Department for Computer and Information Science
Linköping University
SE 581 83 Linköping
linkr@ida.liu.se

Björn Johansson
Ph.D.
Department for Computer and Information Science
Linköping University
SE 581 83 Linköping
bjojo@ida.liu.se

ABSTRACT

Even though many situations in driving involve more than one road user and interaction between those road users, most car driving models utilize a single driver perspective. Collisions between cars constitute a significant part of the total number of crashes each year. A consequence is that driver modeling should move beyond single driver behavior and aim at explaining interaction between drivers. In this paper we will present an approach that merges Hollnagel's Extended Control Model with Clark's Joint Action perspective on coordination. The purpose is to suggest a basic model to help explain coordination in traffic.

Keywords

Coordination, intersection, driving, driver models, traffic safety, Cognitive Systems Engineering, Joint Action, Joint Action Control Model

INTRODUCTION

The field of driver modeling aims at explaining performance of drivers. A large number of models of vehicle driving are based on either cybernetics/control theory, cognitive psychology as information processing or a mix thereof. To explain how control[1] can be achieved in a driving situation has been the major purpose of driver models (Gibson & Crook, 1938; McRuer, 1977; Dounges, 1978; Michon, 1985). What all of these models share is that they approach the task of driving from a single driver perspective or in some cases the driver and the vehicle as a system. Crashes with more than one driver involved constitute a significant part of all crashes each year, but as the current models work very well as a point of departure, they are insufficient per se to explain the dynamics of driver-driver interaction.

[1] 'Control', in this context, refers to the ability to keep the vehicle(s) within an acceptable performance envelope in relation to the goal(s) of the driver(s). Control can further be described in relation to goals on multiple levels, see the Extended Control Model below.

STATE OF THE ART

Cognitive Systems Engineering (CSE) (Hollnagel & Woods, 1983; 2005) is an approach that aims at analysing and evaluating the performance of complex systems consisting of both humans and technical artefacts. Taking a stance in cybernetics, CSE has extended the basic cybernetic theories about control by taking cognitive aspects into account. A basic premise in CSE is that all humans act in a socio-technical context that primarily shapes their behaviour, and thus performance. According to Cognitive Systems Engineering it is possible to view a person and the equipment he or she use as a *Joint Cognitive System* (JCS), meaning that the system as a whole strives toward a goal and that the system can modify its behavioural pattern on the basis of past experience to achieve anti-entropic ends. How can then the boundaries of a JCS be defined? Hollnagel & Woods (2005) use the following example: a driver and his/her car is a JCS, but a driver, his/her car and other drivers and their cars on a road are also a JCS, although on a different analytical level (see figure 1).

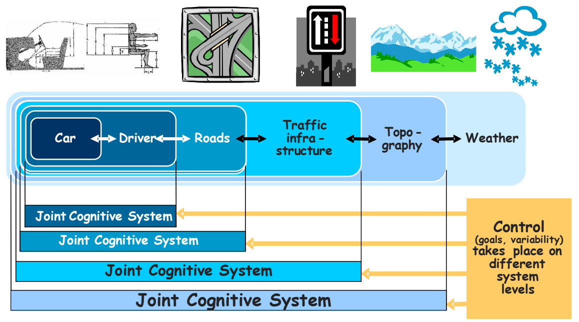


Figure 1. Joint Cognitive systems for driving. (Hollnagel & Woods, 2005)

In order to define if a constituent should be a part of the JCS, it is firstly necessary to identify if the function of it is important for the system, i.e. if the constituent represents a significant source of variety for the JCS – either the variety to be controlled or the variety of the controller (Hollnagel & Woods, 2005). The variety of the controller refers to constituents that allow the controller to exercise his/her variety, thus different kinds of mediators. Secondly, it is necessary to know if the system can manipulate the constituent, or its input, so the result is a specific outcome. If not, the constituent should be seen as a part of the environment. In the case of driving, Hollnagel & Woods states that weather clearly is a part of the environment rather than the JCS, since it is beyond the control of the driver-vehicle system. The driver-vehicle JCS can only modify its behaviour so that it adapts to the weather conditions, it cannot modify the weather.

The Contextual Control Model

Human performance is largely determined by the situation. The environment, our cognitive limitations, and the temporal aspects of our activities constrain possible actions. If we consider a common task like driving to work, we quickly realize that even though it mostly works out in the desired way, there is a large number of things that possibly can go wrong, and we perpetually make adaptations to the surroundings while driving, i.e. other drivers have a strong influence on the way we drive. The context has a strong influence since it to a great extent structures the driving task. Imagine driving to work without any roads, traffic rules or signs? The road has the contextual feature of limiting the area we drive on, the traffic rules help us to manoeuvre in traffic. By constantly reducing the number of choices within the system of 'traffic', it becomes possible to move large vehicles at extensive speeds close to each other with a surprisingly low accident rate. The Contextual Control Model (COCOM) (Hollnagel, 1993) provides a framework for examining control in different contexts, see figure 2.

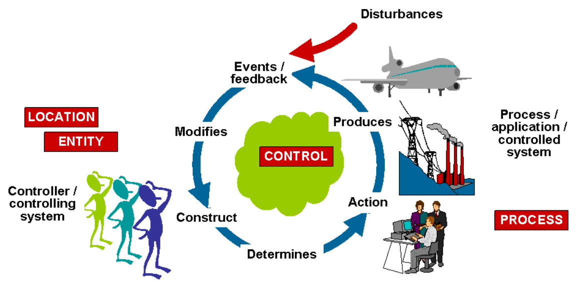


Figure 2. The Contextual Control Model (Hollnagel, 1998).

In figure 2, the controller, who is assumed to have a goal, a desired state that is to be achieved, takes action based on an understanding, a *construct*, in his/her effort to achieve or maintain control over a target system[2]. This action produces some kind of response from the target system. These responses are the *feedback* to the controller. It is not self-evident that the observable reactions are purely a consequence of the controller's action; they may also be influenced by external events. The controller will then maintain or change his/her construct depending on the feedback, and take further action.

The Extended Control Model

Although compensatory (feedback driven) control models apply in many situations, we know that there are great differences in driving style and performance, even when all aspects of the current situation are identical. Hollnagel et al. (2003) have suggested an extension of COCOM, aiming at describing driving control with even higher levels of control, the Extended Control Model (ECOM). The ECOM does not only try to explain the closed loop behaviours, but also tries to introduce high-level cognitive activities like planning. By providing more specific purposes with the driving activity, we may be able to understand why drivers perform different actions in seemingly similar situations. In the ECOM, control is seen as several parallel activities that give input to each other, see figure 3. Higher level cognitive processes, such as goal setting, will thus affect the lower level cognitive processes.

The levels of ECOM applied to driving[3] could be summarised as follows:

Targeting At the targeting level, the driver's own expectations of what will happen in the future derive plans and goals, like, for instance reaching destinations. Short and long term goals are set up and prioritized, which affects lower levels.

Monitoring At the monitoring level, the system keeps track of the traffic environment and the vehicle, and makes plans both from feedback from lower levels and expectations from the higher level. The plans generated are used by the regulating and tracking loops.

Regulating At the regulating level, tasks like target speed, specific position and movement relative to other traffic elements are controlled, often involving a number of tracking sub-loops.

[2] We use the term 'system' in this case. It should be noted that the term 'process', as in 'driving process', could also be suitable.

[3] Also called the Driver In Control model by Hollnagel et al. (2003).

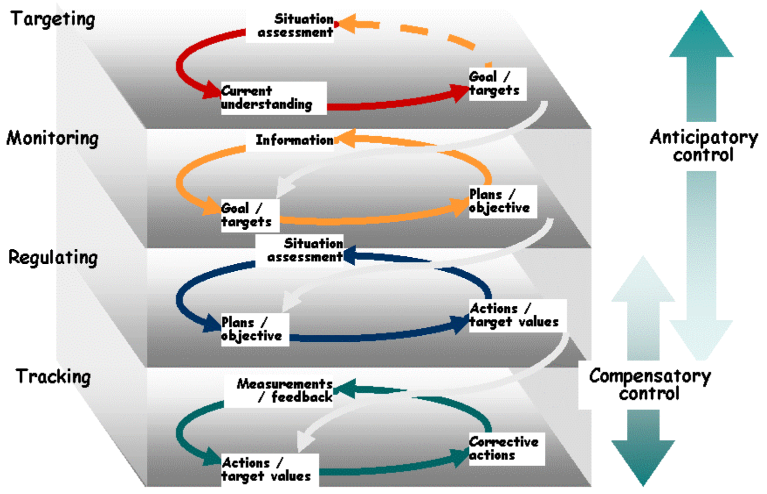


Figure 3. The Extended Control Model (ECOM) (Hollnagel et al., 2003).

Tracking At the tracking level, feedback tasks controlled by higher levels' goals and targets are performed, e.g. keeping the car on the road and shifting gears, maintaining speed, distance to the car in front/behind, and relative or absolute lateral position. By an experienced driver, these actions are performed almost automatically.

From the perspective of ECOM, goal properties, like going from A to B as fast as possible, will serve as a gain directly to the lower control level, affecting the overall behaviour of the Joint Driver Vehicle System. However, goals will always be weighted against each other. For example, fast against safe or speed against accuracy. As noted by Hollnagel et al. (2003), situations may also occur where the driver deliberately gives up control on one or more cognitive levels, in order to handle more severe difficulties on other. Depending on more intricate characteristics of the driver, we may find the weighting between goals different.

In this paper, ECOM is seen as a state-of-the-art model of driving behavior though the model does not only aim at explaining feedback driven behaviour, but also anticipatory, feedforward driven control. This is a significant contribution to driver modelling since it explains the variance in performance due to higher level cognitive activities, which is not explained by the basic feedback-driven models.

COORDINATION

A drawback of almost all models based on the cybernetic approach (ECOM included) is that they apply a single driver perspective. Any skilled driver recognises however that driver behaviour is also shaped by dynamic contextual features, i.e. by interaction with other drivers. The ECOM helps us explain driver behaviour on the levels of goal-setting (targeting/ monitoring) and manoeuvring (regulating/ tracking), but in its current form, it does not really provide an answer to how drivers coordinate their driving in traffic situations with multiple drivers. For example, when two drivers approach each other in a traffic situation, both of them will make a rapid assessment of the other driver's intention, and, if necessary, modify their behaviour. Wilde (1976) states that crucial concepts in the pattern of social interaction in driving are communication and perception of intent, both severely constrained by the design of vehicular signalling systems. The highly structured traffic environment adds more constraints, both in design and by traffic rules and 'social' norms of driving. These constraints help drivers to coordinate their actions, despite the limitations of the communication facilities in automobiles.

The ECOM is a useful conceptual model, or a 'skeleton' of driver-vehicle systems, upon which other theories can be connected to the different levels of control. The compensatory control levels (regulating, tracking) have already been described in detail (see for example Gibson & Crook, 1938 or McRuer, 1977), but the anticipatory control levels, especially when it concerns how monitoring is shaped by driver assumptions about other drivers and the general traffic situation, has been given less attention. We will suggest such a model, based on Clark's work on joint activities (1996) that can expand the scope of the ECOM and help describing how coordination is achieved between drivers.

Expanding the scope

A driver on an empty rural road performs autonomous actions, but participates in joint actions when approaching locations where he/she has to coordinate with other drivers, such as in intersections or in dense traffic. Thus neither individual models nor models of interaction in isolation are sufficient if one wants to understand interactive traffic behaviour. Rather, there is a need to merge these two perspectives, taking a holistic approach on driving where individual goals, assumptions, as well as the situational preconditions in terms of the traffic situation and the means for coordination are taken into account, see figure 4.

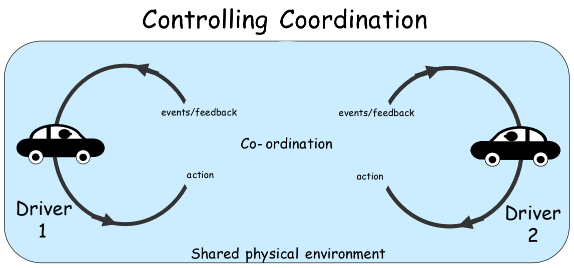


Figure 4. Driver control in coordination

Clark (1996) contrasts between individual actions and joint actions when describing communication or 'use of language' as the later, e.g. joint actions. Individual actions are carried out in isolation, and thus require no communication or coordination with others. Joint actions are performed with a common goal, which is based on the involved participants' assumptions of the purpose and procedure of the action, and may or may not demand communication, but always coordination. Driving is an activity where each driver constantly moves between individual actions and joint actions, depending on the traffic situation.

Klein et al. (2004) generalise the concept of joint activity when describing key aspects of coordination. They propose that joint activity requires a *basic compact* that constitutes a level of commitment for all parties to support the process of coordination, an agreement to participate and to carry out the required coordination responsibilities. One aspect of this basic compact is to align goals, typically to relax some short-term goals in order to permit more long-term goals, individual or shared, to be addressed. For instance, when approaching an intersection a short-term goal of making a left turn, might be relaxed for the long-term goal of reaching the destination safely. To achieve this, one may have to coordinate actions with other drivers to avoid danger, and, hence, wait until it is possible to make a safe left turn.

A second aspect of the basic compact is to try to detect and correct any loss of *common ground* that might disrupt the joint activity. Common ground (Clark & Brennan, 1991) refers to knowledge, beliefs and assumptions that support a joint action and describes what it is that make joint actions work. Common ground is described in three basic stages in a joint activity (Clark, 1996). *Initial common ground* includes all the essential knowledge and prior history that the parties bring to the joint activity, their shared general knowledge of the world and the conventions associated with the task, as well as what they know about each other. *Public event so far* includes knowledge of the joint actions taken so far. *The current state of the activity* provides cues in the physical scene to the parties to enable prediction of actions and the form of coordination.

In driving, joint actions are performed by strangers who will likely never meet, who lacks personal initial common ground about each other. The coordination process often takes place in high speeds in short time windows limiting the common ground of public event so far. This is compensated by an initial common ground based on an understanding of traffic rules, 'social' norms of driving and a shared understanding of the ability to communicate intent by signalling, positioning of the vehicle, etc. It is also compensated by the highly scripted common ground of the current state of the activity. Highly structured by the, for driving, specially designed physical scene with road lanes and signs supporting the shared understanding of traffic rules, norms and conventions.

Peculiarities of communication in driving – a comparison with face-to-face interaction

In contrast to driver-driver interaction, everyday interaction is largely based on verbal communication, but also highly supported by non-verbal cues, such as gestures, facial expressions, gaze etc. The specific preconditions in traffic interaction differ greatly from face-to-face human interaction, see table 1.

Table 1. Characteristics of driver-driver and face-to-face communication.

Time available	Very limited.	Usually not critical.
Means for communication	Vehicle positioning, vehicle signaling.	Language, gestures, facial expression, gaze etc, supports interpretation of quality of statements and attenational focus.
Consequence of missunderstanding	Great. Is difficult to recover due to limitations in time and means of communication and misunderstanding may lead to incidents or crashes.	It can easily be recoverd. Humans have a large repertoir of established mechanisms for recovering from communicative misunderstandings.
Settings	Highly scripted, i.e. Well established understanding of regularities of behaviour.	Level of scriptedness varies depending on the context and situation.

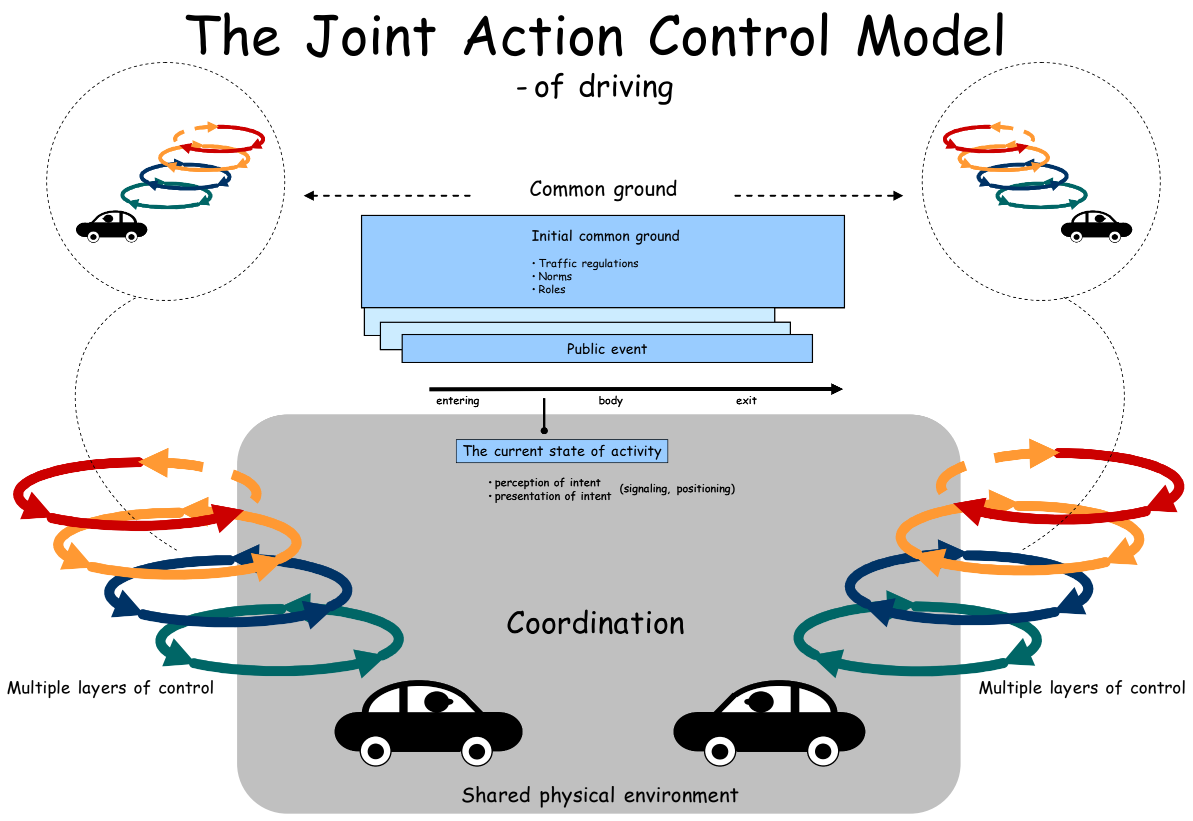


Figure 5. The Joint Action Control Model of driving as a framework for explaining driver coordination.

The time available for interaction and coordination in traffic is often spars and the means for communicating often limited. At very low speeds, the drivers may in the best case be able to make eye contact with each other or do some gestures, but at higher speeds, the drivers have to rely solely on positioning and the signal systems of their vehicles. Furthermore, the consequence of a misunderstanding is often great. If two drivers misunderstand each other, it may lead to a collision, inducing both economical loss and endangering the lives of the drivers. As in all social interactions, the crucial concepts in driver interaction would be perception of intent and presentation of intent. As mentioned above, limitations of communication and available time in traffic interaction restricts the possibility of achieving verification of intent between the involved drivers. However, the very scripted settings of traffic, where drivers have a well established understanding of the regularities and roles of traffic behaviour in various situations allows many conflicts to be managed successfully, despite the limitations on communication.

A joint action is by Clark (1996) described as a phase with an entry, a body and an exit. Phases are what actually get coordinated. The limitations of communication in driving situation, of course, make it difficult for drivers to interpret whether there is an agreement to enter a phase or not, making it a part of their common ground. Instead, conventions play an important role to decide whether or not the other driver/s follows the basic compact and supports the coordination phase/s. We would like to argue that drivers bring to the situation this structure of coordination. A structure not unique for driving, but originated from other human-human activities, which make drivers successfully accomplish coordination.

THE JOINT ACTION CONTROL MODEL – OF DRIVING

Even though for instance Gibson's and Crook´s (1938) idea about field of safe travel partly explains coordination from the perspective of individual drivers, integrating joint actions with ECOM enables a higher-level understanding of coordination and provides a structure for factors assumed to influence coordination. Many of the factors mentioned as essential in driver-driver interaction were summarised already by Wilde (1976), although with a focus on a single driver's internal states. Instead, we advocate for the importance of studying the dynamic interactive behaviour of drivers. Integrating the work on joint activity with

ECOM provides a framework for analysing driver interaction in relation to goals on multiple control levels. As stated, driver behaviour is affected both by low-level and high-level goals, as well as the ability to coordinate driving with other drivers, which to a large extent is affected by individual assumptions made by the drivers in a situation. When entering a situation demanding coordination, as in a road intersection, drivers make assumptions about each others intentions based on traffic rules, traffic 'norms', the type of vehicles involved and the observed behaviour of other drivers. The pre-requisite for successful coordination is that the drivers assume that they have entered the same type of activity, or at least, if uncertain of other drivers intentions, that they have a preparedness to compensate for other drivers actions if they are in conflict with the individual course of action. As a synthesis of the models presented above, we suggest a model of Joint Action Control (Joint Action Control Model, JACOM), see figure 5.

As can be seen in the model, each driver entering a joint activity has to control his/her vehicle while taking the physical environment as well as the actions of other drivers into account. The coordination of action follows the same stages of entering, body and exit as in other human-human activities, which in terms of the individual drivers consist of a set of assumptions about other drivers intentions that are modified during the course of public events. Although driving in dense traffic or intersections surely is a complex task, everyday traffic proves that successful coordination takes place almost constantly. The explanation for the success is probably two-folded. Firstly, all drivers have about the same background knowledge in terms of traffic regulations and traffic norms, described in the model as initial common ground. Secondly, humans are very good at verifying that the basic compact is fulfilled, i.e. coordinating actions with others and compensating for each others mistakes. Improvements in the traffic environment in terms of traffic lights, roundabouts, lane separation etc. have also decreased the demands on driver interaction, reducing the risk for misunderstandings and collisions.

To summarise, models of driving has focused mainly on the single driver, and thus neglected the dynamic character of coordination. This has in turn led to a focus on isolating drivers from each other rather than enhancing the capacity for coordination. This paper emphasise the need to expand the single-driver models of driving toward multi-driver models of driving. The Joint Action Control Model of driving can be seen as a first step in this direction.

ACKNOWLEDGMENTS

This research was founded by the IVSS-project (In Vehicle Safety Systems) on Intersection Accidents: Analysis and Prevention. We thank the partners involved in the project: Autoliv Research, Chalmers, Saab Automobile, the Swedish Road Administration and Volvo Car Corporation. We would also like to thank Torbjörn Falkmer for all good comments.

REFERENCES

Clark, H. H. (1996) *Using Language*. Cambridge: Cambridge University press.

Clark, H. H. & Brennan, S. E. (1991) Grounding in Communication. In L. B. Resnick, J. M. Levine & S. D. Teasly (Eds.), *Perspectives on socially shared cognition.* Washington DC: APA.

Dounges, E. (1978) A Two-Level Model of Driver Steering Behavior. Human Factors, 20(6), 691-707.

Gibson, J. J. & Crook, L. E. (1938) A theoretical field-analysis of automobile-driving. *The American Journal of Psychology*, LI(3), 453-471.

Hollnagel E. (1993) *Human Reliability Analysis: Context and Control*. London: Academic Press.

Hollnagel, E., Nåbo, A. & Lau, I. V. (2003) A systemic model of driver-in-control. *Proceedings of the second international driving symposium on human factors in driver assessment, training and vehicle control,* Park City, Utah, 86-91.

Hollnagel, E. & Woods, D. D. (1983) Cognitive Systems Engineering: New wine in new bottles. *International Journal of Man-Machine Studies*, 18, pp 583-600.

Hollnagel, E. & Woods, D. D. (2005) *Joint Cognitive Systems. Foundations of Cognitive Systems Engineering*. Boca Raton, FL: Taylor & Francis.

Klein, G., Feltovich P. J., Bradshaw J. M. & Woods, D. D. (2005) Common ground and coordination in joint activity. In W. B. Rouse & K. R. Boff (Eds.) *Organisational Simulation.* U.S.A: John Wiley and Sons LTD.

McRuer, D. T., Allen, R. W. & Klein, R. H. (1977) New Results in Driver Steering Control Models. *Human Factors*, 19(4), 381-397.

Michon, J. A. (1985) A critical view of driver behavior models: What do we know, what should we do? In L. Evans and R. C. Schwing (Eds.). *Human Behavior and Traffic Safety*. New York: Plenum Press; Pp. 485-520.

Wilde, G. J. S (1976) Social Interaction Patterns in Driver Behavior: An Introductory Review. *Human Factors*, 18(5), 477-492.

IVSS Intersection Accidents

Automatic repair and quality measuring of vehicle trajectories

Applied on image processing data from the Sävenäs intersection

Issuer/CDS ID	91413, Fredrik Moeschlin, fmoeschl@volvo.com
Issue Date	2008-03-25
Approved by	John-Fredrik Grönvall, jgronval@volvocars.com
File name	Report - Auto Repair_no_logo_BW.doc
Security Class	Public

1. Background and need for automatic processing

The IVSS Intersection Accidents project involves automatic video surveillance of an intersection at Sävenäs close to Göteborg. Through image processing the moving objects in the intersection are identified and represented as trajectories. Several data files are generated in Matlab format, typically containing about two hours of tracking data from the intersection. However, the data from the image processing does not perfectly represent the vehicles travelling through the intersection. Some of the main problems are:

- Trajectories broken into separate parts, thus representing one vehicle with several trajectories,
- Trajectories jumping from one vehicle to another, i.e. representing several vehicles,
- Trajectories not representing vehicles (flags are one prominent issue)

In order to perform reliable analyses, the trajectories from the image processing need to better match the vehicles travelling through the intersection. This was the setting and goal for the development of an automatic repair function. The task was divided in two parts: establishing quality measures and developing methods for repairing trajectories. Quality measures are required to identify defect trajectories and to monitor the effect of methods for automatic repair. In this report, the development of quality measures will precede the description, usage, performance and illustration of the scripts developed for automatic repair. All scripts were developed in Matlab and full documentation is available in the code. It should be stressed that the automatic repair does not include any image processing; the repair algorithms are operating purely on existing trajectories.

2. Quality measurements and noise level

Two categories of quality measures have been developed: trajectory quality and noise over time. A trajectory quality describes how well a single trajectory performs and the noise over time is a measure of the image processing performance at a given video frame. Further on, these two categories will be referred to as quality and noise. During the development of quality and noise, the measures have been manually evaluated using the original video and the integrated trajectory view available in the Hedvig software. One of the corner stones behind the quality measures are principal trajectories that are further described below.

2.1. Principal trajectories

Vehicles travelling through an intersection typically follow a distinct pattern in space; they appear on one road edge, drive through the intersection centre and disappear on another edge. The idea behind principal trajectories is to identify the most common path leading from one distinct edge to another. For instance, a four-way intersection will have three principal paths leading from each road edge (if u-turns are neglected). Several objects (trajectories) travelling along the same principal path are used to form the principal trajectory. The principal trajectory is created through an averaging where each trajectory's position in space (rather than time) is used for synchronisation. Details on how principal trajectories are created can be found in appendix A.

When a complete set of principal trajectories has been created for an intersection, it can be used as guidelines for normal vehicle behaviour. In this project, the principal trajectories are used to measure trajectory quality, noise level over time and to guide the automatic repair process when trajectories are split and merged. An example of principals generated from data from the Sävenäs intersection is shown in figure 1.

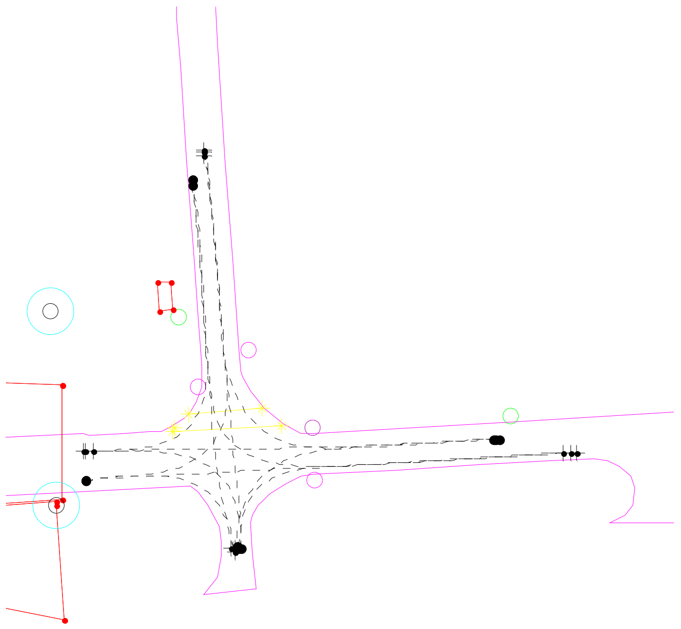


Figure 1: Principal trajectories (dotted black) generated for and plotted in the intersection at Sävenäs.

2.2. Trajectory quality

A number of quality measures have been developed within the project for different purposes. To find a single quality measure that could be estimated for every trajectory in the intersection an attempt was made to summarise and evaluate the previously used quality measures. The definition used for trajectory quality is the probability that the trajectory is fully representing one vehicle's entire journey through the intersection. Thus, high quality requires trajectories to represent vehicles, to represent only single vehicles and to represent their entire journey through the intersection.

Trajectory quality: probability of representing one vehicle's entire journey through the intersection.

In total nine measures were identified as useful for describing different aspects of trajectory quality. Listed without ranking the measures identified are:

1. Number of frames (points) (proportional to duration in time).
2. Length in meters [m].
3. Largest span in space ($||(\text{span}(x), \text{span}(y))||_2$ where $\text{span}(x) = |\max(x) – \min(x)|$) [m].
4. Rotation quality measured as max total rotation/length in meters [m^{-1}].
5. Largest jump between two subsequent points ($\max dr = \max ||(dx, dy)||_2$) [m].
6. Minimum absolute distance to intersection centre [m].
7. Average distance in space to the best matching principal trajectory [m].
8. Confidence for classification according to the best matching principal.
9. Length in parallel to the best matching principal, divided by the principal's length.

These nine partial quality measures, available after repair in the field *Objects.qualities*, were considered individually and evaluated on several different data sets. The results and distributions for each quality measure can be found in appendix B. To aggregate the partial measures into one single quality measure, each partial measure was translated linearly with a method described in appendix B. Briefly, this method applies predefined thresholds for in which interval a partial measure is most precise. In this way the method will extract the most relevant data from each partial measure. The result of this process is shown in figure 2 where the distribution of the aggregated quality measure is shown for 22 212 trajectories from different days. The aggregated quality measure is available after automatic repair in the field *Objects.quality*.

As can be seen in figure 2, the objects are separated in two piles at about 0.2 and 0.9. Through manual verification it was found that no trajectory with a quality below 0.5 appeared to be fully representing one vehicle's journey through the intersection (as quality was defined above). Thus, the qualities residing in the lower end are invalid or broken trajectories that should be excluded from the analysis. However, this does not imply that all vehicles are fully detected and represented with high qualities. If for instance one vehicle is split up in several short trajectory pieces each piece will be given a low quality since it can hardly be distinguished from the shadow of a flag. This is where the automatic repair comes in, providing methods for split and merge.

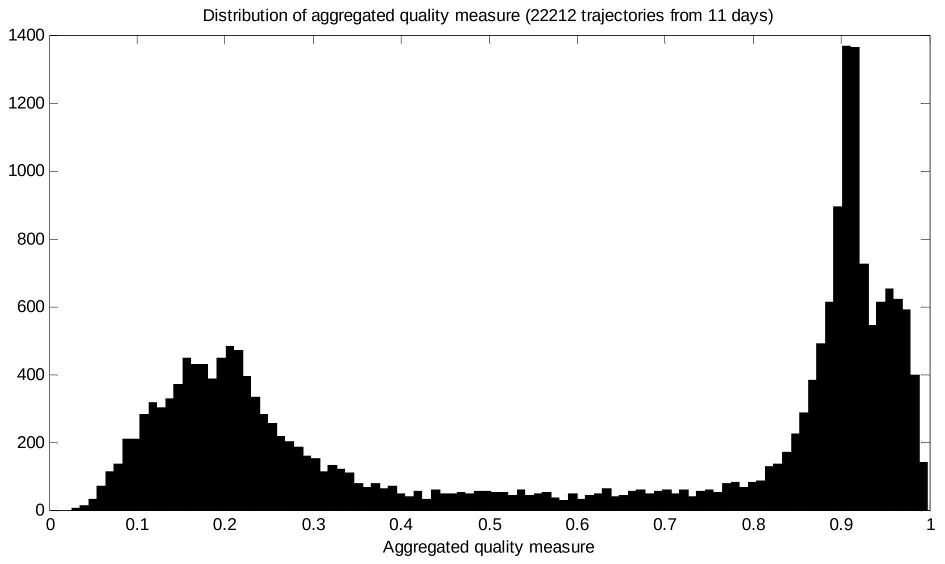


Figure 2: Aggregated quality measure for 22 212 trajectories from different days and times with various qualities.

2.3. Noise level

The performance of the image processing depends largely on the time because of shifting weather conditions (sunlight and wind) and variations in traffic density. To estimate the quality over time, the noise level measure was developed. The noise level is an estimation of how many broken trajectories there are in each time frame. Whether a trajectory is broken or not is determined by some of its quality measures. After counting the number of invalid trajectories for each frame, the noise level is achieved by filtering with a Gaussian low-pass filter. Finally, a threshold can be applied to separate noisy data from good data as illustrated in figure 3. The noise level after automatic repair is available in the field *ImLabels.noise*.

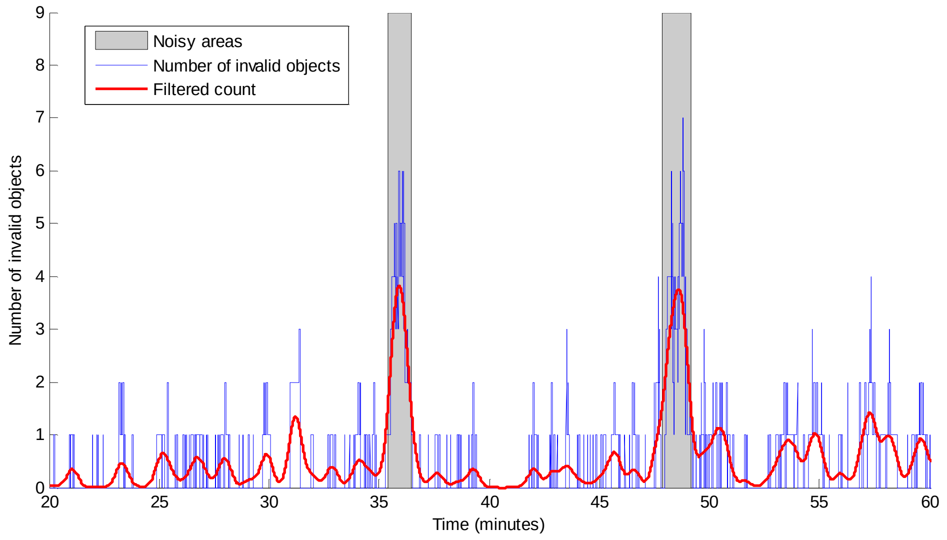


Figure 3: Illustration of noise level calculation for a period of time. The blue stepping curve corresponds to the number of invalid trajectories, the red curve is the low-pass filtered count (the noise measure) and the gray bars represent sets of frames with noise greater than 1.5.

3. Automatic trajectory repair

The automatic trajectory repair will process the data from image processing and try to repair trajectories that appear to be invalid. The script is executed from the Matlab command *auto_repair*. Help on how to use a specific script is provided by executing the Matlab command 'help *scriptname*'.

The following steps are carried out during automatic repair:

- Pre-processing of trajectories
- Quality and noise measurements
- Split of trajectories representing more than one vehicle
- Merge of several trajectories representing one vehicle
- Estimation of automatic repair performance

The very first step however is to generate principal trajectories for the intersection. Preferably this is done in advanced in order that the principal trajectories can be loaded from a file during the automatic repair (see appendix A for details on how to create principals). The pre-processing will calculate a number of trajectory properties that are required in several of the following steps. Quality and noise measurements are used to estimate which trajectories that are representing vehicles and to estimate how the image processing has performed over time (this is likely to affect the performance of automatic split and merge). No data with low quality is deleted during the automatic repair; instead the quality level is delivered as a property of each trajectory. The following fields are added to the structure *Object* during automatic repair:

The automatic split algorithm will detect trajectories that appear to be jumping from one vehicle to another. This is done by monitoring the deviation from the closest principal trajectory. If the vehicle trajectory suddenly starts to deviate more and more from the principal's path it is considered to have jumped. The point where the sudden deviation starts is used as point of split. This algorithm is able to handle multiple splits, e.g. where a trajectory jumps between several vehicles or objects in the video. Detailed information on the algorithm behind split is provided in appendix C.

During automatic merge an algorithm will identify trajectories that resemble valid vehicle journeys through the intersection but are shorter in distance. These trajectories are referred to as cut pieces. The algorithm will then step through all the cut pieces and try to find other trajectories to connect them with. In the search forward, the paths provided by principal trajectories are used as hints for the algorithm. A number of requirements are set to ensure that the merged trajectory has a vehicle-like behaviour. Detailed information on the algorithm behind merge is provided in appendix D.

During automatic repair, the following fields are added to each trajectory in the *Object* structure:

dist	distance vector for the trajectory, starts at 0, unit [m]
mLength	length of total merged part of this trajectory [m]
principal	index of the best matching principal trajectory
qualities	vector of quality measures (see quality measures above)
quality	aggregated quality measure in the range [0, 1]
source	indices to original trajectory in the original *Object* structure
sourcei	original positions (indices) within the original trajectory

When the trajectory repair is finished, the script will estimate the effect on the trajectories. This is done by measuring how the quality and noise have changed. In the ideal case the noise will be reduced and the quality increased. The aggregated quality is measured as the average trajectory quality and the noise as the time in seconds where the noise level is above 1.5. All measures of repair performance are put in a structure, auto_repair_info, which is passed along with the repaired data. A description of this structure is provided below.

Field name in auto_repair_info	**Description**

noise_t	Seconds with a noise level greater than 1.5 after repair.
noise_ratio	Fraction of time with noise > 1.5 after repair (noise_t/total_t).
splits	Number of trajectories split during automatic repair.
merges	Number of merges during automatic repair.
noise_t_raw	Seconds with a noise level greater than 1.5 before repair.
noise_ratio_raw	Fraction of time with noise > 1.5 before repair (noise_t_raw/total_t).
d_noise	Decrease in total noise during repair $((\sum noise_{before} – \sum noise_{after})/\sum noise_{before})$.
d_quality	Increase in average quality during repair $(mean(quality_{before}) – mean(quality_{after}))/mean(quality_{before})$.

4. Results from automatic repair

4.1. Illustration of the process

The best way to illustrate how the automatic repair operates is by an example. Figure 4a to 4c below shows a situation where the image processing has a limited performance but where the automatic repair manages to mend most of the trajectories. The images were cropped from the Hedvig software and only trajectories representing the two blue vehicles in the centre are shown. In figure 4a, it is obvious that the yellow trajectory (starting in the bottom of the picture) jumps from the original car on to the vehicle arriving from the left. In figure 4b this trajectory has been successfully split (the abruptly turning piece at the end is discarded due to its lack of data points). After automatic merge, in figure 4c, the split trajectory is mended with the last part that was lost in both figure 4a and 4b. Unfortunately the image processing was confused by the many vehicles arriving from the left and the last piece is drifting towards the opposite lane until it is finally lost (long before the vehicle leaves the camera view).

The situation shown in figures 4a to 4c is difficult for the automatic image processing because several vehicles arrive at the same time and queues are built up from the left. The tracking is lost, generating a large amount of trajectory pieces which makes the noise level rise. After automatic repair, it was found that the noise level for this scenario was above 3, which is to be considered a high noise level (cf. the limit of 1.5).

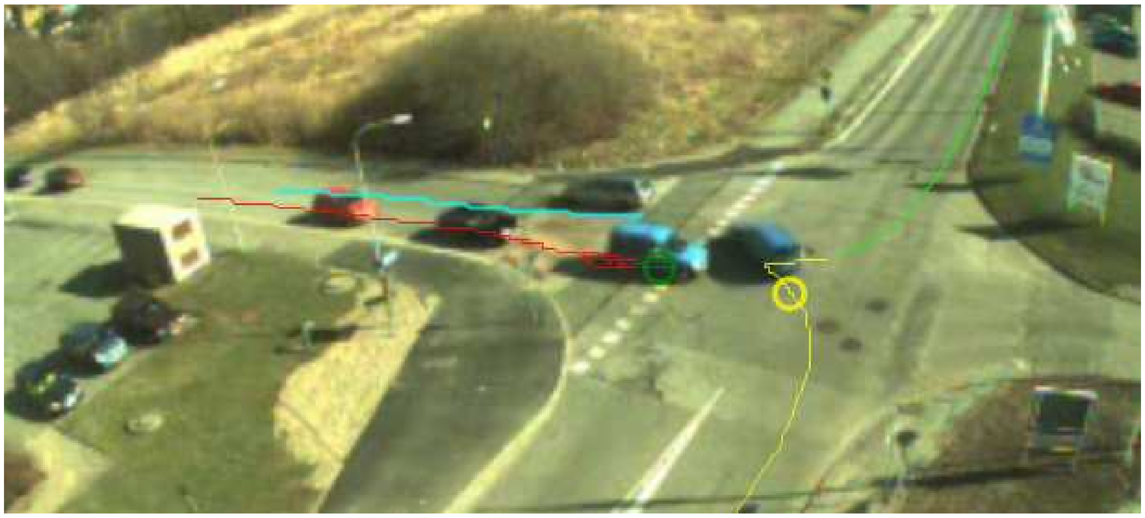


Figure 4a: Trajectories in original data (from coarse image processing), only trajectories representing the two blue vehicles in the centre are shown. The circle indicates the current vehicle position.

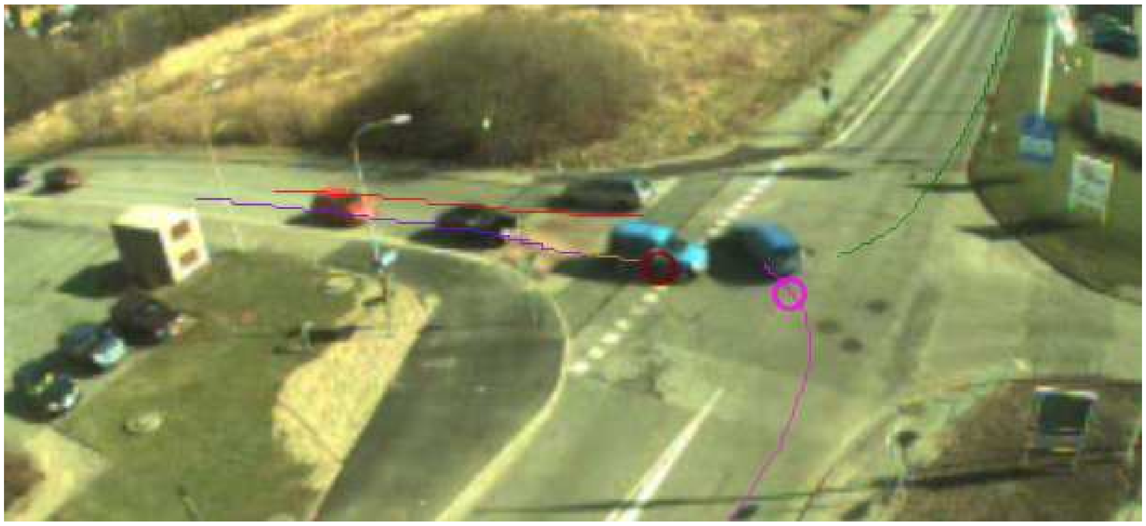


Figure 4b: Trajectories in figure 4a after automatic split.

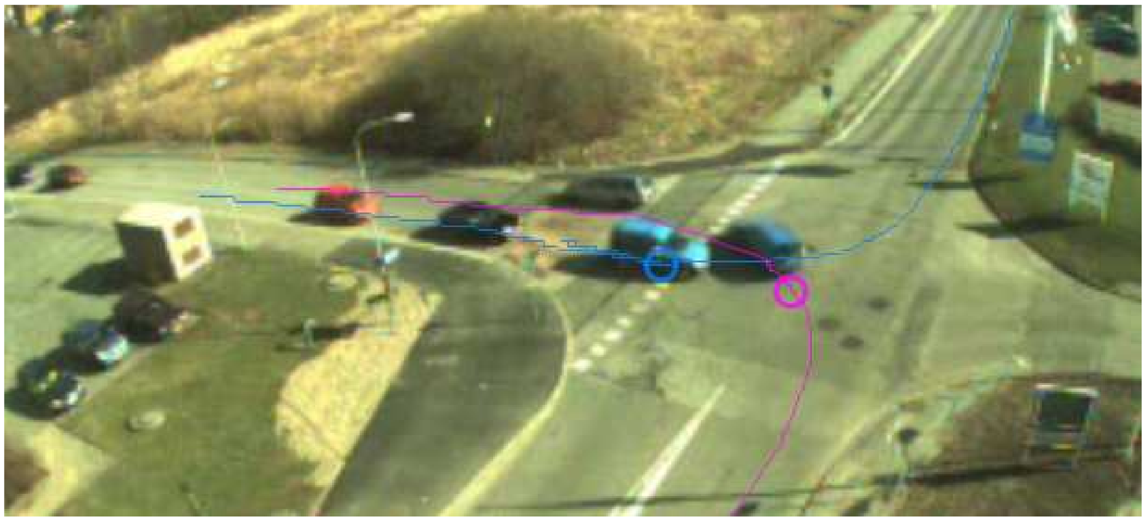


Figure 4c: Trajectories after automatic split and automatic merge (entire automatic repair).

4.2. Effects from automatic repair on quality and noise

To monitor the effects from automatic repair on data quality, the measures of trajectory quality and noise level were estimated before and after automatic repair. To illustrate the effects on noise, the automatic repair was applied on a data set from a normal day with steady light conditions where the image processing performed well most of the time (2nd of April 2007). Prior to repair, 379 seconds of data had a noise level greater than the limit of 1.5. After automatic repair, only 296 seconds of data had a noise greater than the limit. This change might appear minor but it is actually highly significant since the only operation applied was mending – no trajectories with low quality were discarded, flags and shadows remain. Figure 5 gives an example of how the noise level is changed by automatic repair for a limited segment of data.

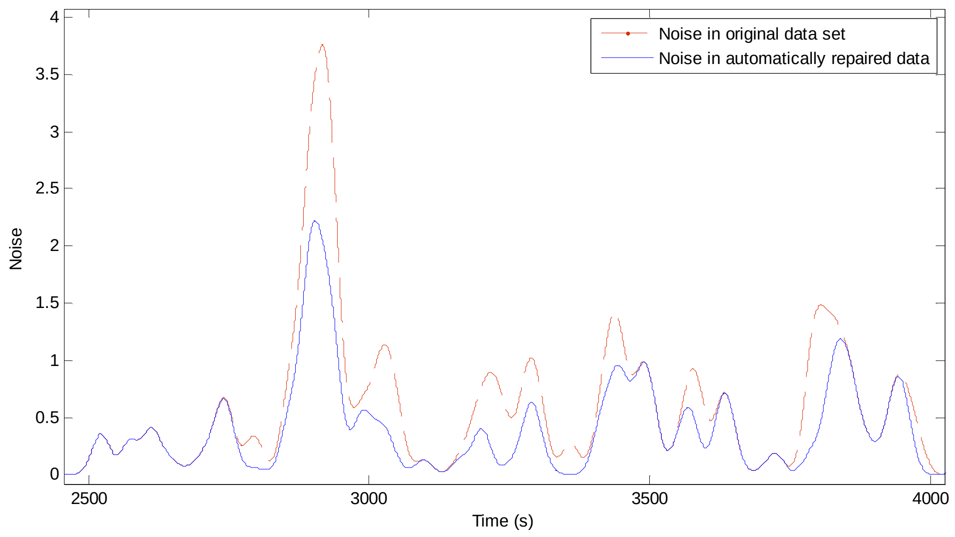


Figure 5: Noise level for a section of data prior to and after automatic repair.

The effects from automatic repair on trajectory qualities can be detected by studying the change in quality values. Figure 6 shows how the distribution of trajectory quality is changed during automatic repair. Negative values in the figure indicate that trajectories with this quality level were decreased in number during automatic repair and vice versa for positive. Thus it is apparent that the automatic repair in general reduces the number of trajectories with low quality and increases the number of trajectories with high quality, i.e. the automatic repair increases quality. When studying the distribution in figure 6, there seem to be more data below zero than above. If the number of trajectories remained constant during repair the distribution should be balanced, e.g. a trajectory removed from one quality interval should be added to another. However, in the automatic merge algorithm two or more trajectories can be combined into one which causes a reduction in the total count. This explains why more trajectories are removed at low quality than are added at high quality. It also shows that the automatic repair algorithm has a positive net effect on the trajectory quality.

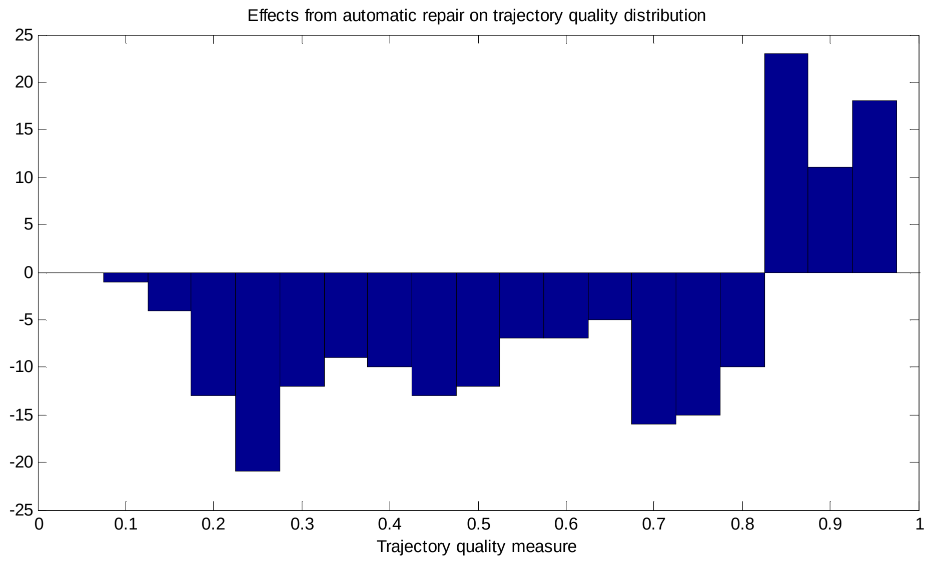


Figure 6: Changes in trajectory quality during automatic repair on a two-hour data set. The distribution indicates the increase or decrease of number of trajectories with specific qualities.

4.3. Summary of results on the image processed data

The results in *auto_repair_info* provide a summary of the performance of automatic repair on the specific data set. To summarise the general performance a number of files (154) from coarse image processing were randomly selected and automatically repaired and the information from *auto_repair_info* was extracted. It turned out that the automatic repair in average increases the trajectory quality by 2.1 % and reduces the total noise by 18 %. The analysis also shows that in average 12 % of a data set will be above the noise limit of 1.5 after the automatic repair. Further details are provided in the table below.

Parameter in auto_repair_info	**Average value for the 154 data sets**
Total noise after repair	872 s.
Ratio of noise after repair	0.120
Number of splits	117
Number of merges	115
Total noise prior to repair	1019 s.
Ratio of noise prior to repair	0.140
Ratio of total noise change	0.182
Ratio of total quality change	0.0208

5. Discussion and conclusion

The developed measures for quality and noise will constitute an important basis for further analysis of the data. It is crucial to understand and estimate the data quality to assure scientifically sound results. The automatic measures will also be useful during manual trajectory repair where they provide a hint on which trajectories and which time sections to consider for repair. Further on, the noise measure and specifically the total noise for a data file can be a criterion for including it in the analysis or not. However, even though the trajectory quality measure developed here was able to separate most trajectories into clusters of high and low quality (as in figure 2), there was still an interval where the quality was uncertain (about 0.5 to 0.8). It appears as if the trajectory quality could be improved further if an in-depth quality assessment would be carried out. Another indicator of this is that some of the partial quality measures are significantly correlated. Still it should be stressed that the limit of 0.5 is appear to be a reliable criterion for filtering out trajectories that do not fully represent one single vehicle travelling through the entire intersection.

During the automatic repair several trajectories were mended which generated an increase in quality and a reduction of noise. Thus it performs as intended and even manages to repair some rather complex situations. This automatic processing will be profitable for the time-consuming manual trajectory mending. Still the manual work will be needed since the automatic repair only operates on existing trajectories. If there is not enough data available from the image processing the automatic repair will not be able to create the necessary trajectories since the video is not used as input. For the manual repair, it is thus strongly recommended that the user can select sections to repair based upon arbitrary limits of noise and quality.

Appendix A – Principal trajectories

A principal trajectory is created from image processing data – trajectories – of vehicles travelling along the same path, i.e. from one specific road end to another. One way to identify trajectories travelling along the same path is by studying their entrance and exit points in the intersection. This can be done automatically using the script *get_principals_advanced.m*. The script will use a selection of trajectories, provided by the caller, and display all the trajectories' entrance and exit points in a figure, generating clusters of points. The user will then be asked to draw boxes to identify the entrance and exit zones on each road end. After the manual identification, the principals will be automatically calculated and returned as trajectories. An illustration of the process of selecting entrance and exit zones is shown in figure A1 where the generated principals also are plotted. Further details on using the script are provided below.

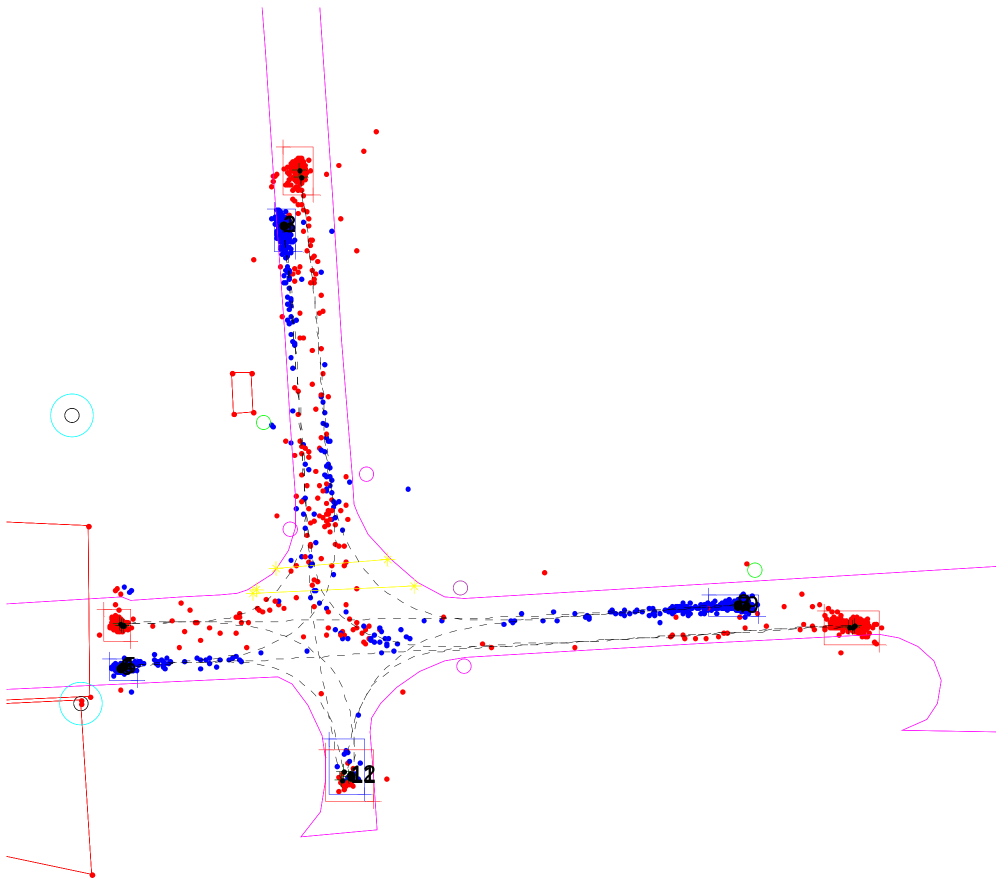


Figure A1: Illustration of using *get_principals_advanced.m*. Trajectory entrance points are blue and exit points are red. The boxes are manually entered with two points each. The principal trajectories generated through this process are shown as the dotted black curves. No trajectories were found travelling from south to north, thus no principal trajectory could be generated for this path.

Details on using *get_principals_advanced.m*

After a rough filtering, a number of entrance and exit points are plotted in the intersection in a new figure. Entrance points are plotted blue and exit points red. Once the points are plotted, the user will be asked for input of the boxes. Each box is entered by clicking two corners opposite each other on the diagonal (e.g. lower left and upper right). After selecting the four boxes for input zones, the four exit zones are selected in the same manner. If there are less than four zones (e.g. in a three-way intersection) it is possible to stop entering boxes by pressing enter. The operation will then shift from entrance zones to exit zones or from exit zones to done.

NOTICE: The zones for entrance and exit must be entered in the same order. For instance:

Entrance zones:	1. North. 2. West. 3. East. 4. South.
Exit zones:	1. North. 2. West. 3. East. 4. South.

Once the manual input is done the entered zones will be used for calculation of principals using *get_principals.m*. The entered zones will be display in the Matlab command window in a format that can be directly copied to *get_principals.m* for permanent use.

Appendix B – Trajectory quality measure evaluation

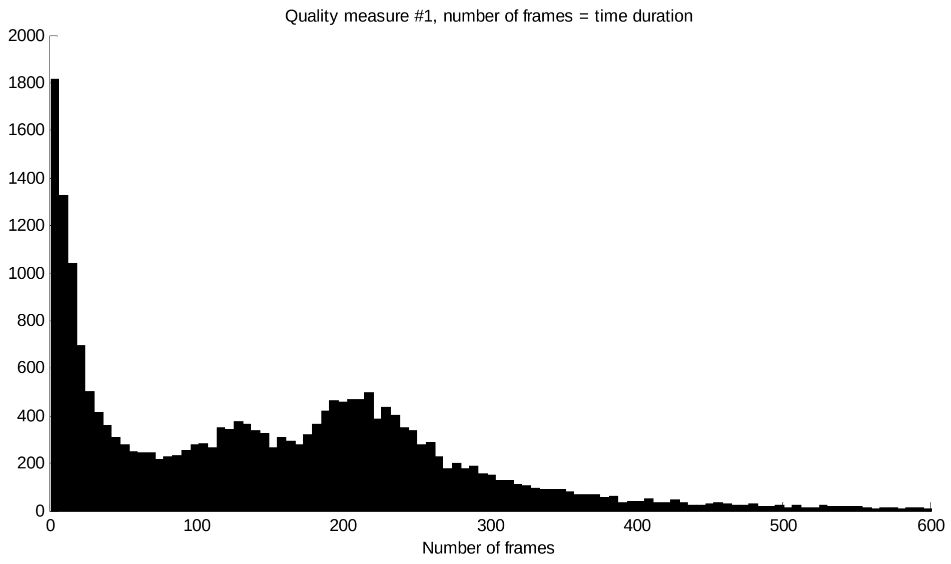
Partial quality measure statistics

The nine partial quality measures were evaluated in detail. After finding the distribution of a measure's typical values, trajectories with different quality were considered, starting from the end where low quality is claimed to reside. The trajectories within a certain interval of quality were plotted and manually evaluated. As the interval gradually was shifted towards higher and higher quality the trajectory behaviours became more and more vehicle-like. A limit was manually entered where the first trajectories that seemed valid appeared. At the same time, a manual grading was made for lowest and highest quality limits, i.e. where the trajectory quality did not decline or improve any further. The three values, transition limit, minimum and maximum are summarised in table B1 where also some statistical measures are included. Distributions for each partial quality measure are given in figure B1. Further details on the computation of the quality measures are provided in the script code. To recapitulate, the partial quality measures considered are:

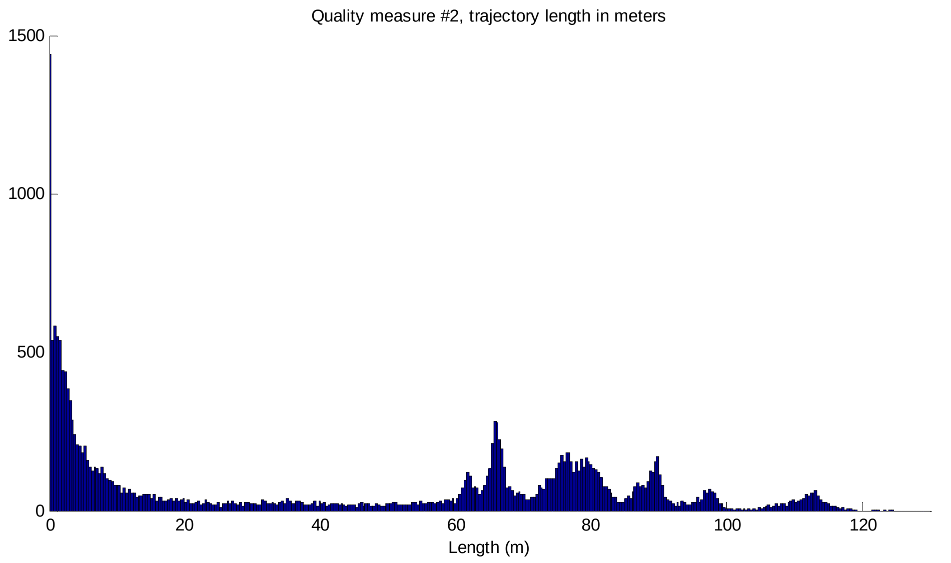
1. Number of frames (points) (proportional to duration in time).
2. Length in meters [m].
3. Largest span in space ($||(\text{span}(x), \text{span}(y))||_2$ where $\text{span}(x) = |\max(x) – \min(x)|$) [m].
4. Rotation quality measured as max total rotation/length in meters [m^{-1}].
5. Largest jump between two subsequent points (max dr = max $||(dx, dy)||_2$) [m].
6. Minimum absolute distance to intersection centre [m].
7. Average distance in space to the best matching principal trajectory [m].
8. Confidence for classification according to the best matching principal.
9. Length in parallel to the best matching principal, divided by the principal's length.

Quality measure	#1	#2	#3	#4	#5	#6	#7	#8	#9
Relevant transition	56	24	20	5,50	1,25	10	9,50	0,005	0,54
Percentage below	31 %	43 %	44 %	7 %	1 %	43 %	12 %	23 %	49 %
Growing with quality	Yes	Yes	Yes	No	No	No	No	Yes	Yes
Relevant minimum	0	0	0	10	3	50	14	0	0
Relevant maximum	200	60	50	1	0	0	2	0,70	1
Min	1,00	0,00	0,00	0,00	0,00	0,01	0,02	0,00	0,01
Max	5948	182	98,43	125,67	3,64	73,67	46,90	0,99	1,00
Std	185,46	37,28	32,19	3,66	0,26	14,18	5,14	0,40	0,43
Mean	164,09	43,68	37,50	1,94	0,41	14,24	3,30	0,49	0,53
Median	145,00	47,41	42,71	1,16	0,46	7,93	0,95	0,60	0,59

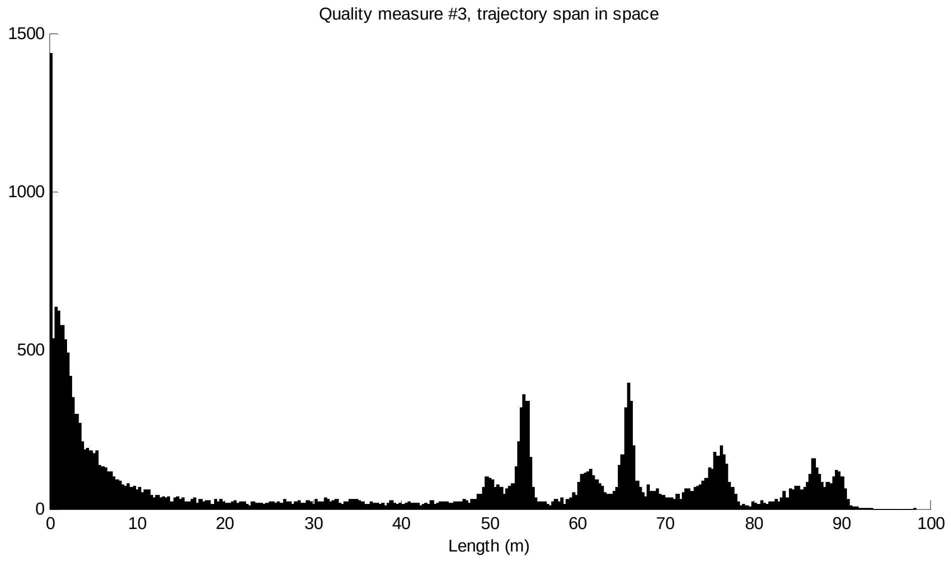
Table B1: Results from partial quality measure evaluation. The relevant values are manually entered as input for the aggregation to one single quality measure.



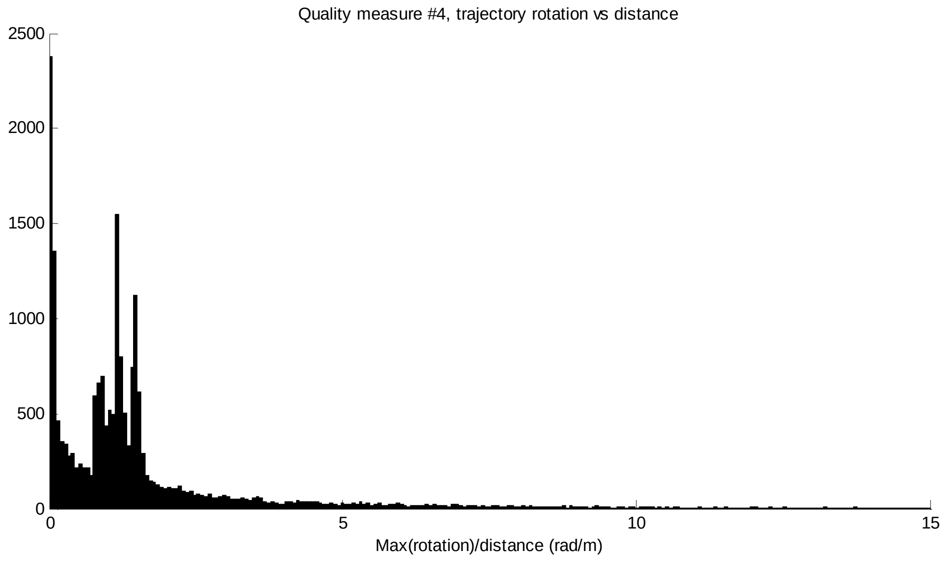
Quality measure #1, number of frames = time duration
2000
1800
1600
1400
1200
1000
800
600
400
200
0
0
100
200
300
400
500
600
Number of frames



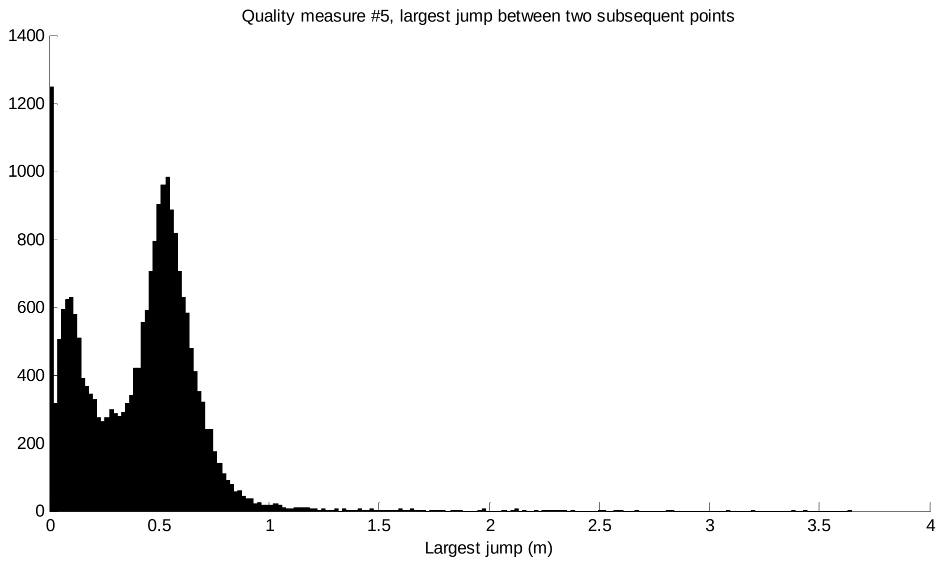
Quality measure #2, trajectory length in meters
1500
1000
500
0
0
20
40
60
80
100
120
Length (m)

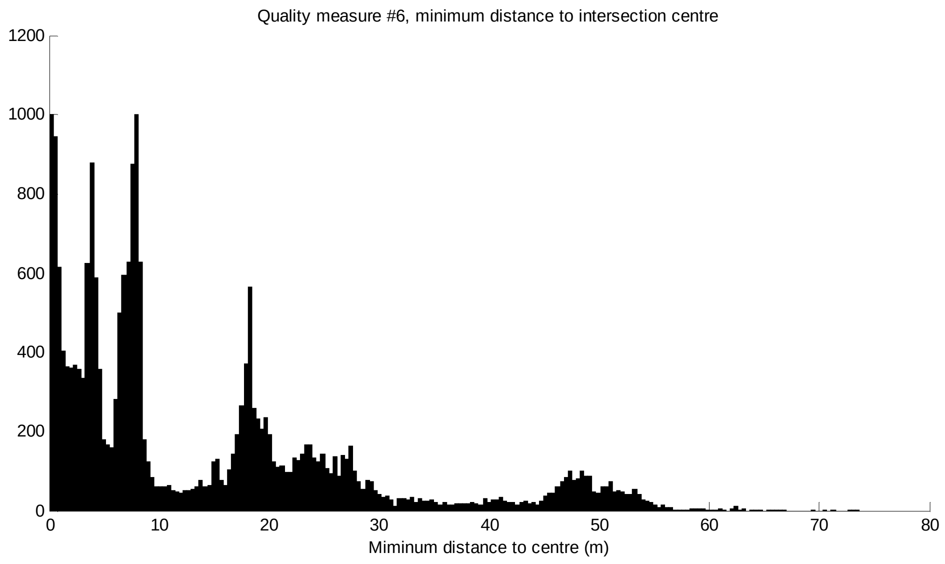


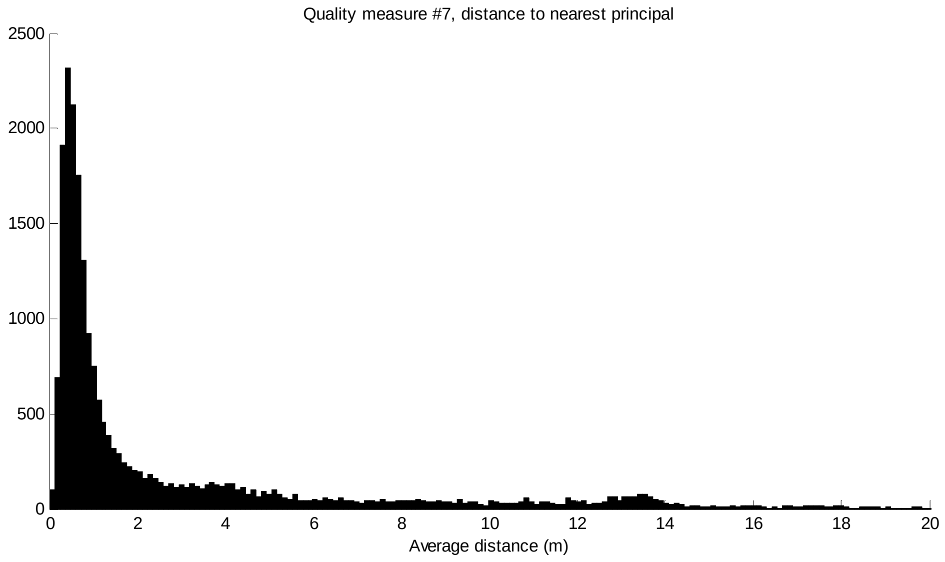
Quality measure #3, trajectory span in space
1500
1000
500
0
0
10
20
30
40
50
60
70
80
90
100
Length (m)



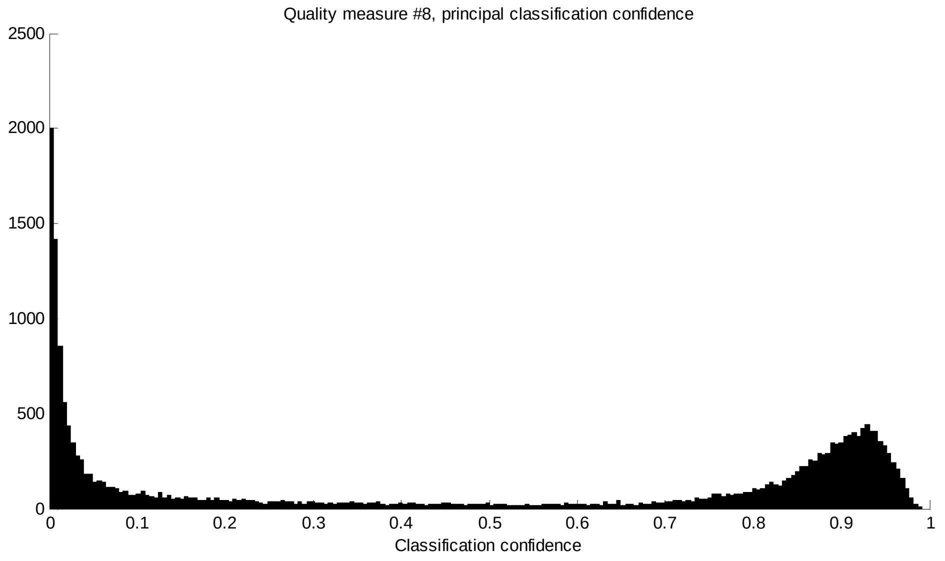
Quality measure #4, trajectory rotation vs distance
2500
2000
1500
1000
500
0
0
5
10
15
Max(rotation)/distance (rad/m)


Quality measure #5, largest jump between two subsequent points
1400
1200
1000
800
600
400
200
0
0
0.5
1
1.5
2
2.5
3
3.5
4
Largest jump (m)


Quality measure #6, minimum distance to intersection centre
1200
1000
800
600
400
200
0
0
10
20
30
40
50
60
70
80
Miminum distance to centre (m)



Quality measure #7, distance to nearest principal
2500
2000
1500
1000
500
0
0
2
4
6
8
10
12
14
16
18
20
Average distance (m)



Quality measure #8, principal classification confidence
2500
2000
1500
1000
500
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Classification confidence

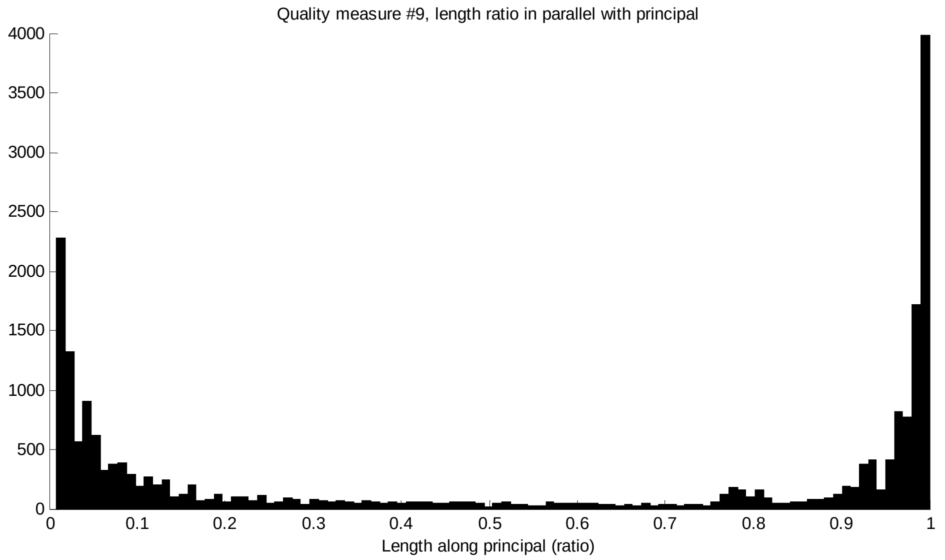


Figure B1: Distribution of the partial quality measures evaluated.

Aggregation to a single quality measure

To generate one single quality measure all partial measure were involved without individual weighting. The manually entered relevant values for min, transition and max were used to linearly transform the quality measures to a scale from 0 to 1, where 0 is bad quality and 1 good quality. Before transformation, values above the relevant maximum were set to the maximum limit and equivalently values below minimum where set to minimum limit. The linear transform was configured so that the manually entered transition from bad to possibly good data occurred at 0.5. After transforming each partial measure individually the aggregation was made through computing the average. It was found that, at least in the about 22 000 objects considered, no trajectory that appeared valid in shape had a quality below 0.5.

Appendix C – Algorithm behind automatic trajectory split

The script auto_split.m will find trajectories that need to be split and return unaffected trajectories and the parts remaining after split. Notice that U-turns are split up unless u-turns are represented in the principals. When examining the trajectory, its deviation is measured from the principal trajectory that lies closest in space. Typical behaviour for trajectories that need to be split is a smooth following with a next to constant deviation followed by a step where the deviation increases. After the step the deviation is moved to a different level where it often remains constant. Thus the deviation is moved to a different level. For detecting this, the derivative of the deviation is used.

The following criteria are used to detect if a trajectory needs to be split:

1. A trajectory needs to have a certain amount of points (typically at least 6 meters) in parallel with any principal trajectory. This is required for attaining enough of points for detecting a step in the deviation.
2. The largest absolute value of the deviation derivative must be larger than a certain value (typically ≥ 1.0). This is needed to ensure that the step is significant.
3. The median value of the absolute deviation derivative needs to be small (typically ≤ 0.2) to ensure that the trajectory is mostly travelling in the same direction as a (its) principal trajectory. This is equivalent with requiring the deviation to be fairly constant most of the time.

Further comments on the split algorithm are available as comments associated to the script.

Appendix D – Algorithm behind automatic trajectory merge

The merge algorithm will run through all trajectories identified as cut and try to merge them with other trajectories. The procedure follows these steps:

1. Find the principal paths that the cut trajectory might belong to. The criterion is that the distance must not be greater than cMaxValidDistance meters. Make sure it has not previously been merged with another trajectory.
2. Search for pieces to connect:
    a. Look at objects with first appearance later in time than for the cut trajectory considered but no later than last time stamp of the cut trajectory + cMaxTimeGap seconds.
    b. Only consider objects not previously merged with another cut trajectory (to prevent merging two vehicles to one).
    c. Only consider objects that possibly travel along the same principal path. Requirement: the distance must not be greater than cMaxValidDistance (meters).
    d. Only consider objects longer than cMinTimeStamps.
    e. Only consider objects that start further away on the principal trajectory than the first 80of the cut trajectory.
    f. Measure the distance between the cut trajectory and the found piece using the Euclidian distance r = sqrt(dx^2 + dy^2) (meters) as well as the time distance dt (seconds).
    g. Translate the time distance to meters assuming the average speed of cAssumedSpeed (km/h) and calculate the total distance as delta = sqrt(r.^2 + (dt*cAssumedSpeed/3.6).^2) (meters).
    h. Select the trajectory piece that has the lowest delta value as the most appropriate. If the distance is too far no piece should be selected. If the second best trajectory piece has a very close delta value the pieces are reported to the user (based upon cMinConfidence).
3. If a piece was selected, merge the two trajectories using interpolation.
4. Check that the merged trajectory appears valid. Otherwise, reject the merge and skip to next trajectory.
5. If it was valid, mark the attached piece as "used" (see 2.2).
6. If the merged trajectory is shorter than cLengthRatioLimit times its closest principal trajectory, use the merged trajectory as initial part and jump to step 1. Otherwise, continue with the next.

Specific values for the constants are available in the script together with further documentation and details.

IVSS Intersection Accidents: Project Report Image Processing on Video Sequences System Description

Björn Johansson, Johan Wiklund, Gösta Granlund

July 7, 2008

Contents

1 Introduction

This report describes the tracking system that is used in the project *IVSS Intersection Accidents*.

The report is outlined as follows: Section 2 gives a quick overview of the system. Further overview on each component can be found in sections 6-13. Many of the technical and theoretical details are moved to appendices, to make the presentation more clear. Sections 4-5, 14, and appendix A describe a few surrounding issues such as hardware related issues, camera calibration, video data management routines, and computational issues. Limitations and ideas for improvements are discussed in connection to each component.

1.1 Prerequisites

It was desired that as much video data as possible could be processed, which means data in many varying conditions. For example, figures 1-4 shows how the light can vary during a day, due to moving clouds, and due to seasonal variations. Some of these situations had to be disregarded, e.g. wet asphalt.

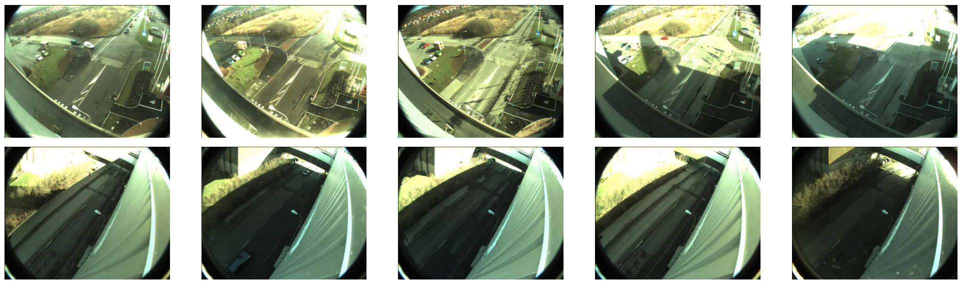


Figure 1: Example of light variations during a sunny day for two cameras mounted viewing different parts of a road.

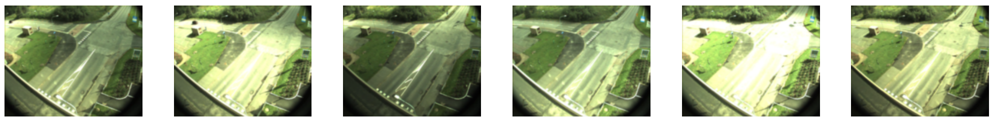


Figure 2: Example of light variations due to moving clouds.

It was also desired, from a behavioral point of view, that data should be collected from vehicles at least 3 seconds before they enter the intersection. This corresponds to about 75m if the speed limit is 90km/h, which is a rather large coverage of an intersection. The coverage was later reduced to about 40-50m due to technical limitations.

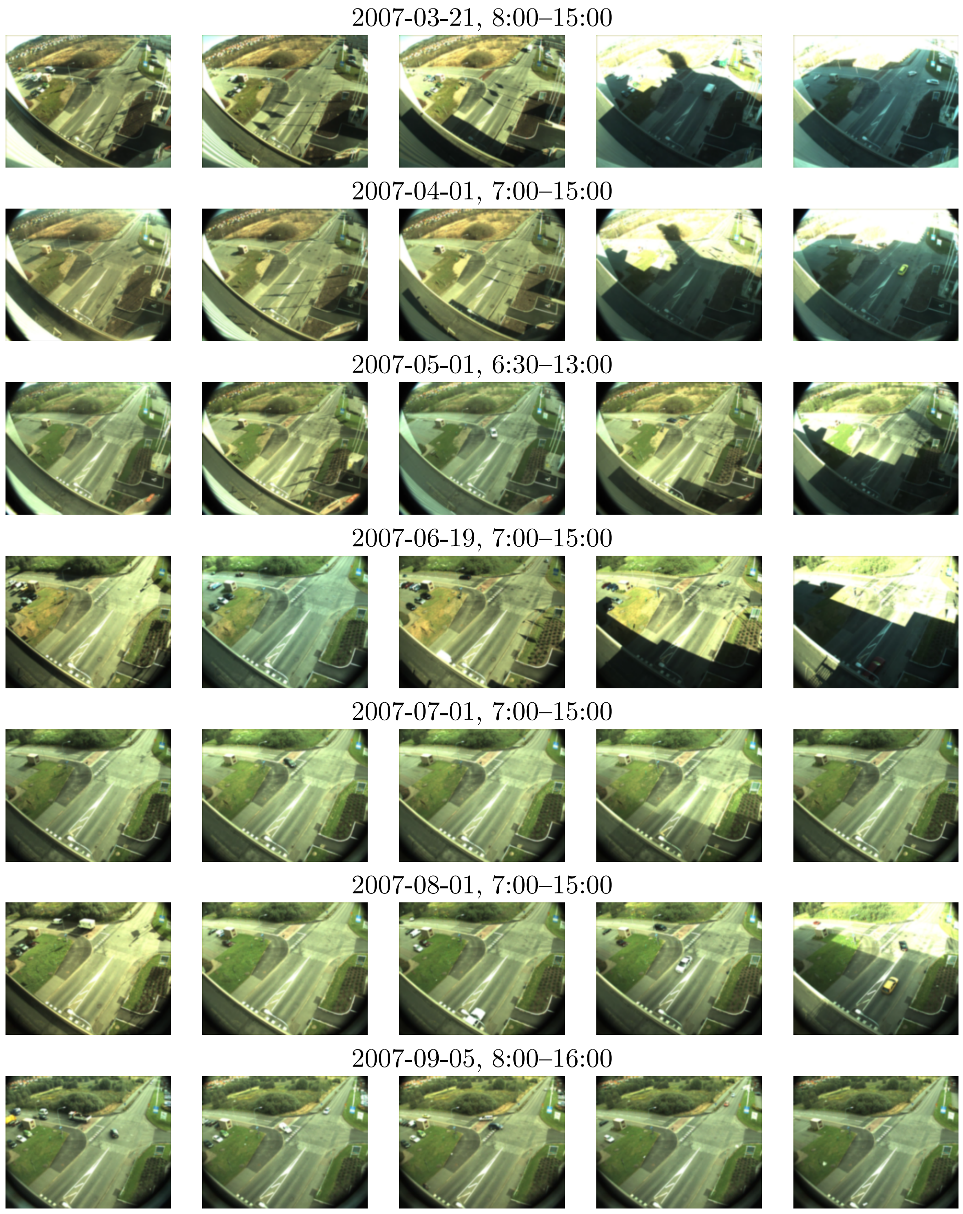


Figure 3: Video examples during an extended time period.

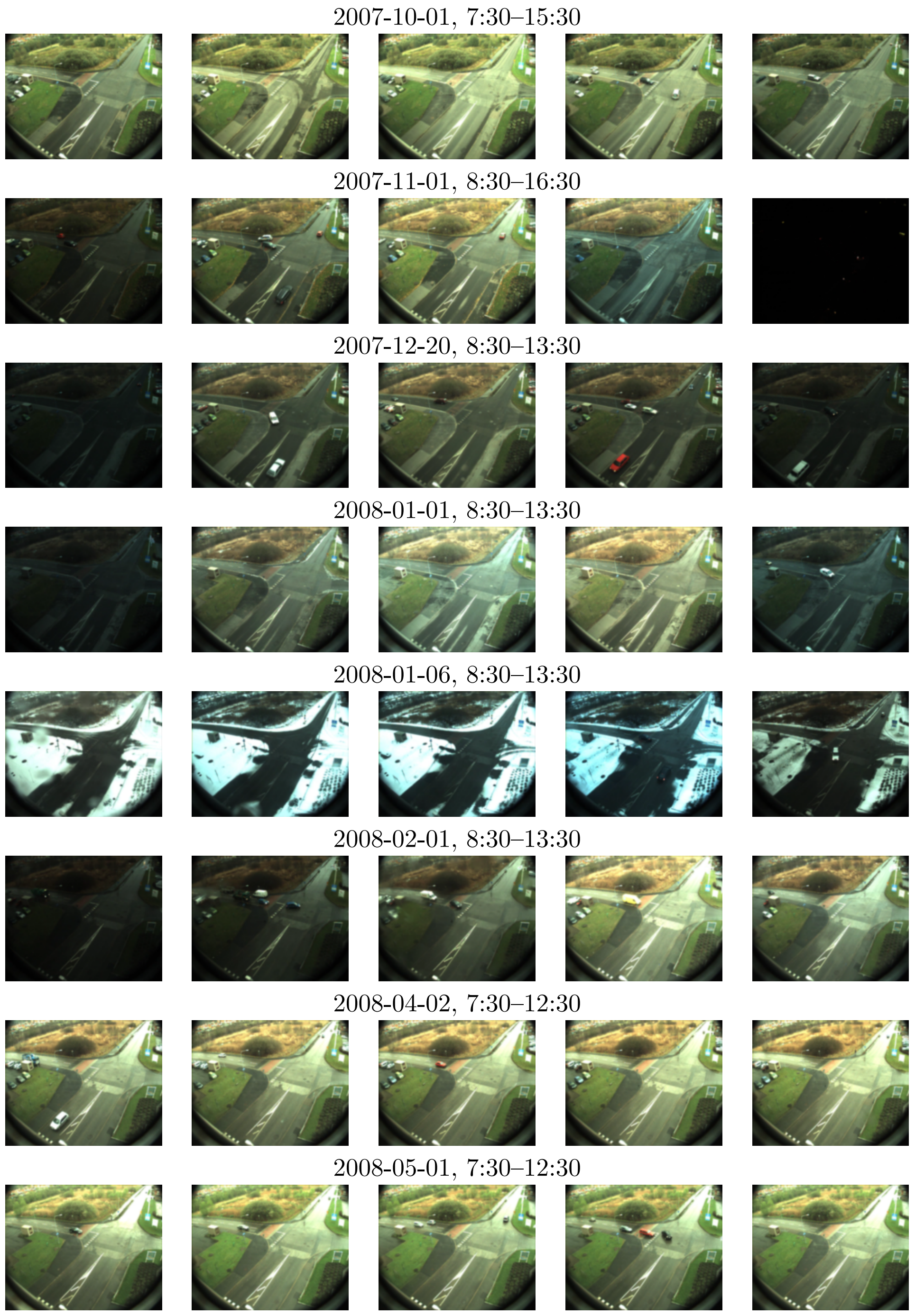


Figure 4: Video examples during an extended time period.

2 System overview

Figure 5 illustrates the process from an image sequence to trajectory data. We will here give a quick overview of the tracking system, details on each component are described further in other sections. The components are the following:

1. **Image rectification and ego-motion compensation:** (Section 6)

   The sequence is first rectified to remove lens distortion. The sequence is also compensated for camera ego-motion when required.

2. **Statistical background subtraction and foreground classification:** (Section 7)

   A statistical background model is then estimated, and used for classification of pixels into background/foreground. The foreground is then further classified into foreground/shadow/highlight.

3. **Coarse (first-round) tracking:** (Section 11, with sections 8, 9, and 10 as preparations)

   The classified image sequence is used as input to an image region tracker. The tracker is based on 3D boxes, which in each frame are predicted from previous frames (e.g. using a Kalman filter), and optimized to the classified image. The 3D boxes simultaneously segments regions (which can be sparse) in the classified image into objects and gives an estimate of position and size on the 3D world ground plane. The coarse tracking is computed in low resolution data, with a causal box prediction, and testing different box sizes in each frame.

4. **Post-filtering on trajectory data:** (Section 12)

   The estimated box position trajectories are after the coarse tracking processed with a non-causal filter to improve the estimates.

5. **Automatic and manual repair:** (See [24])

   Some trajectories will be broken due to occlusions, or trajectories from different vehicles may have been fused due to limitations in the tracker etc. Automatic and manual reparations, such as splitting and merging of trajectories, are applied to improve the data. The automatic repair is based on principal trajectories (see [24] for details). The manual repair is optional.

6. **Refined (second-round) tracking:** (Section 13)

   The trajectories are further refined by using the image data a second time. The refined tracking utilizes 3D boxes, similar to the coarse tracking, but now in high resolution, with a non-causal box prediction, and with a fixed box size.

7. **Post-filtering on trajectory data:** Same as step 4.

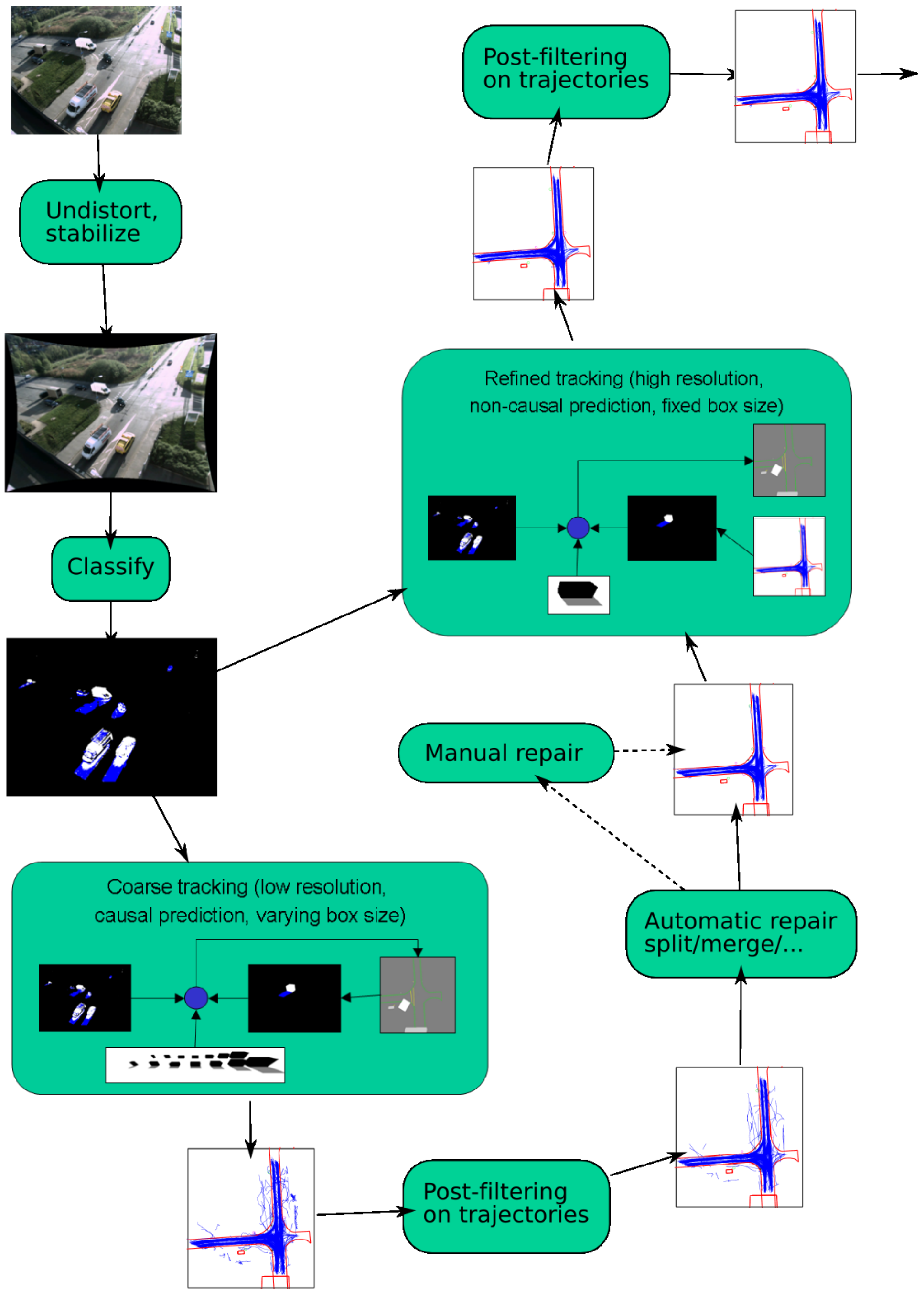

Undistort, stabilize
Classify
Coarse tracking (low resolution, causal prediction, varying box size)
Post-filtering on trajectories
Automatic repair split/merge/...
Manual repair
Refined tracking (high resolution, non-causal prediction, fixed box size)
Post-filtering on trajectories

Figure 5: Overview of the image processing tracking system.

2.1 System output

The output from the system is a .mat-file that contains the tracking data. There are essentially two output variables from the system, ImLabels that contains frame and object label information and Objects that contains vehicle data. Besides these variables there are also a few variables with parameters for the world ground plane, camera calibration, time interval etc.

ImLabels: Struct array with active trajectories and time stamps (msec since 1970) for each frame.

Field name:	Description:
ImLabels(n).ObjectsLabels = [O1 O2 ...]	Visible objects in 'frame n'
ImLabels(n).time = t	Time stamp for 'frame n' [msec since 1970]
ImLabels(n).syncind = [ind1 ind2 ...]	Indices to corresponding frames in the video files (The videos are not in sync, therefore using 'frame n' for both cameras does not work)

Objects: Struct array with objects. Time is measured in msec since 1970 and geometric data in GPS coordinate system.

Field name:	Description:
Objects(k).time = [t1 t2 t3 ...]	Time stamps [msec since 1970]
Objects(k).size = [L W H]'	Size [meters]
Objects(k).pos = [X1 X2 ...;Y1 Y2 ...]	Positions [meters]
Objects(k).orient = [fi1 fi2 ...]	Orientations [degrees]
Objects(k).vel = [v1 v2 ...]	Velocities (or rather speed) [meters/seconds]
Objects(k).acc = [a1 a2 ...]	Acceleration list [meters/seconds]

There are also a few other fields in Objects with quality measures from the automatic repair procedure, see [24] for further details.

2.2 Related work

A vehicle tracking system intended for processing of a large amount of data will have to deal with many different types of weather conditions and traffic situations. This is a difficult task and is still not sufficiently solved, hence the vast amount of continuing publications in this field. Nagel and co-workers [25] have for example written an interesting exposé over the progress of object tracking during 30 years, with frame differentiation, feature based optical flow, dense optical flow, wire-frame models, and inclusion of more a priori knowledge of e.g. lane locations.

One common approach in tracking systems is to first classify the image pixels into background and foreground regions, as in the Stauffer-Grimson statistical background subtraction method [30]. One problem in this approach is shadows, which are often included in the foreground and cause errors in segmentation and tracking. The Swedish conditions may be especially difficult in that respect, since the sun elevation angle in the middle of Sweden is at best about $55°$, thus casting long shadows even in summertime. There exist a number methods for detection of shadows, [26] evaluates a few early methods. One of the methods use the color model introduced in [16], which in [35] is combined with the Stauffer-Grimson background subtraction method to get a statistical background/foreground/shadow classification. This is the method used here, and we also describe a generalization for dealing with highlights as well (defined in section 7).

An alternative to detecting and removing shadows is to simulate them. [9] showed how a shadow added to their wire-frame model could improve the accuracy when fitting the model to edge features. [29] give examples of how simulating shadows on a 3D box (with fixed size) can improve detection. The detection is however still useful when the simulation is insufficient, e.g. when the vehicle model is inaccurate, or when other indirect shadows appear. We will combine both shadow detection and shadow simulation, and show that the combination can improve the estimation on average.

We use 3D boxes as vehicle appearance model, to aid the segmentation but also to simulate shadows. As said in [9, 8], switching from 2D tracking in the image plane to 3D tracking in the scene domain often results in a substantially reduced failure rate, because more prior knowledge about the objects can be utilized. For example, foreground regions are not always homogeneous, and we need some criteria to merge regions that are likely to belong to the same object. It can be difficult to choose an image based scale threshold for merging regions, especially if the objects vary in size e.g. due to varying distance. The use of 3D boxes is a way to choose this scale threshold based on geometric information.

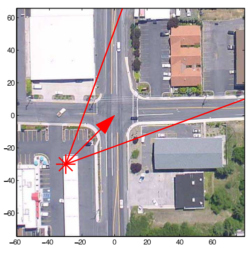
Many other models of different complexities have been proposed, ranging from a few 3D line features [18], rectangular 2D boxes [23], and more explicit models like wire-frame (polyhedric) models [25, 9, 8, 21]. In our system the 3D box appears to be a sufficiently complex model, and it is simple enough to cover most types of vehicles. It is difficult to have wire-frame models for all sorts of vehicles, especially in industrial areas where many uncommon types of vehicles occur (tractors, trucks, vehicles with different types of trailers, etc.). Furthermore, full adaptation of the wire-frame model to image data can be unstable [4].

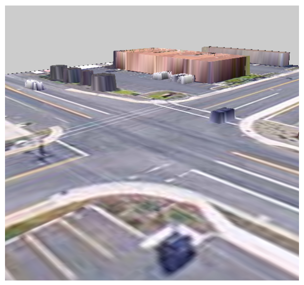
3 Camera setup issues

The choice of camera setup was simulated in Matlab (see e.g. 'help graph3d') before the actual setup and collection. Figure 3 shows some examples of simulated camera pictures in Matlab (the image was found on the Internet, and a depth-map for some of the largest objects has been estimated and designed by hand).

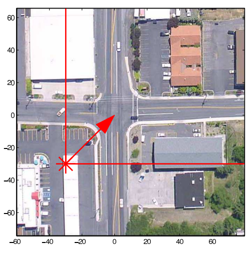
There are limitations to this approach. Lens distortions are not included in the present Matlab toolbox for simulation, which means that this method cannot be used to simulate camera images with wide angle lenses (unless we make special software). But the simulation at least gives an idea of the resulting view.

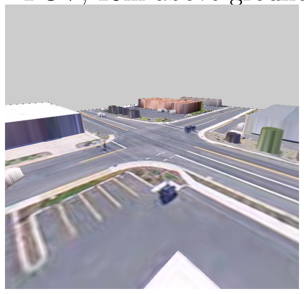
Simulation example. Camera with 50° FOV, 15m above ground.



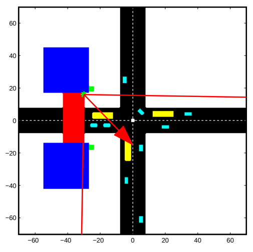


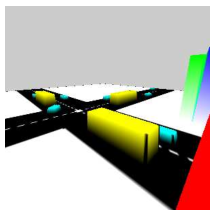
Simulation example. Camera with 90° FOV, 15m above ground.



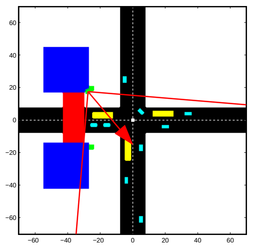


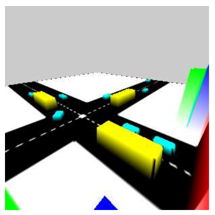
Simulation example, Renova in Göteborg. Camera with 90° FOV, 11m above ground.





Simulation example, Renova in Göteborg. Camera with 90° FOV, 18m above ground.





4 Hardware related issues

This section discusses software and other issues that are closely related to the hardware.

4.1 Adaptive shutter control

The camera need to have some type of adaptive control for compensation of illumination changes. The cameras used in this project did not have any build in automation for this control. A PID controller were implemented for selecting the shutter time to keep a constant average image intensity, the details are described in appendix B.

The shutter time cannot alone keep a constant intensity in all conditions, figures 3 and 4 shows some limitations. The ability to control the light sensitivity would be improved if we also included the aperture in the control, but this requires motorized lenses.

4.1.1 Post-processing due to instability in the controller

In one of the intersections (Jung), the parameters were set such that the camera shutter started to toggle, and the image intensity displayed the same phenomena. In order for the background subtraction to work, the images should have a constant intensity or at least change smoothly. A post-processing procedure to repair this data was therefore implemented and included in the GPU implementation for simultaneous ego-motion compensation and rectification (section 6). The images were adjusted such that the mean intensity in certain regions (manually marked, to avoid areas with moving objects like the road) was held constant. Some camera views cover mostly the road, so the marked regions were in some cases quite small but appeared to be sufficient.

4.2 Bayer conversion

The conversion from the Bayer format in the camera sensor to RGB images may cause color distortion effects. They are most obvious in high contrast regions, see e.g. figure 6, but may occur in any region. The shadow detection described in section 7 relies on the assumption that only the intensity of a pixel changes when the region becomes shadowed. The color distortion in the Bayer conversion may then cause some shadow misclassification. However, there is probably other more important sources of errors. But care should be taken not to use a too crude method in the Bayer conversion if the following processing is relying on color information.

4.3 Polarization filters

Polarization filters were later installed, to reduce light reflexes from wind-shields etc. However, the effect was minor. Instead, an image processing procedure to

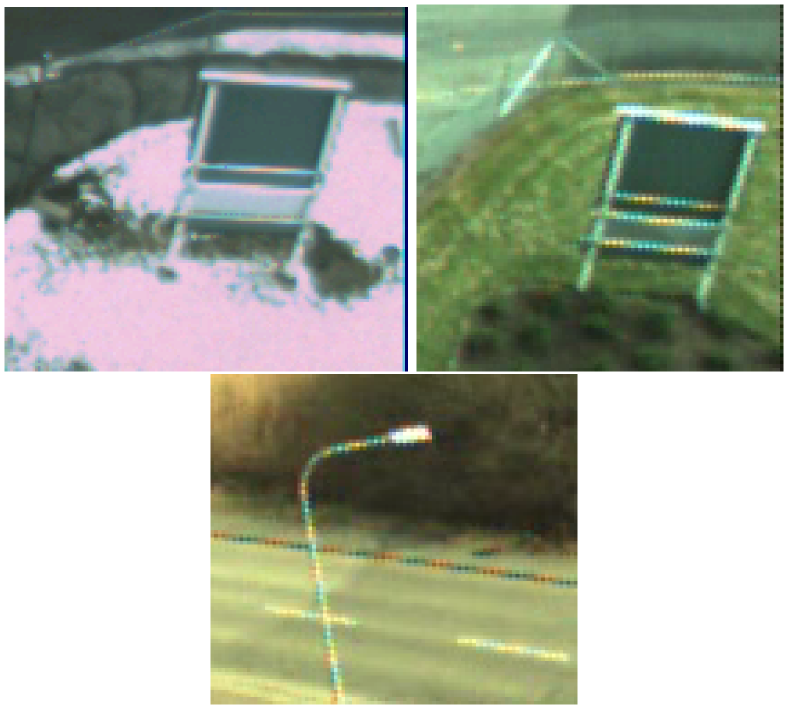


Figure 6: Examples of Bayer conversion effects.

detect highlight pixels were developed, which gave a better performance, see section 7. Reflexes due to wet asphalt is however still a problem.

4.4 Frame and time drops

There are occasional drops in frames and time stamps in the collected video data. The frame drops and the time drops does not always come at the same time. These drops may reduce the performance in the case of multiple cameras, where the tracking algorithm assumes that the cameras are synchronized. Initially, the synchronization relied on a constant framerate, but were later changed to rely on the time stamps.

5 Camera calibration

Figure 7 shows an overview of the camera calibration. The calibration is performed in two steps: First, parameters for the lens distortion is estimated. Second, after rectification of the lens distortion, only the pinhole camera model remains, and the mapping between 3D and 2D (the projection matrix) is estimated. These two steps are described in further detail below.

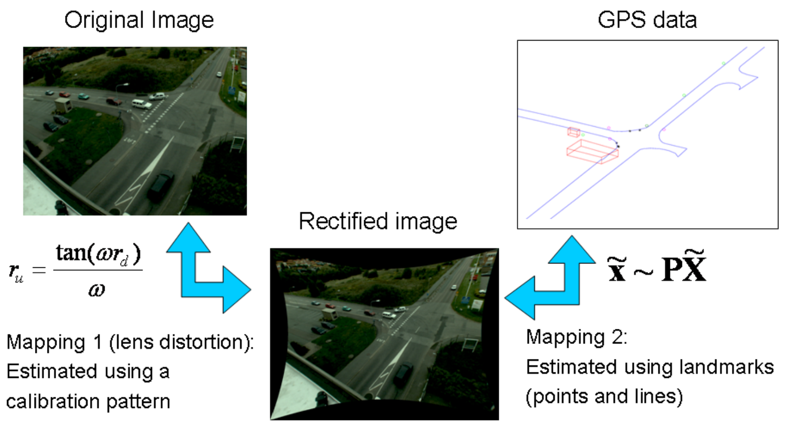

Original Image
GPS data
Rectified image
$r_u = \frac{\tan(\omega r_d)}{\omega}$
$\tilde{\mathbf{x}} \sim \mathbf{P}\tilde{\mathbf{X}}$
Mapping 1 (lens distortion):
Estimated using a
calibration pattern
Mapping 2:
Estimated using landmarks
(points and lines)

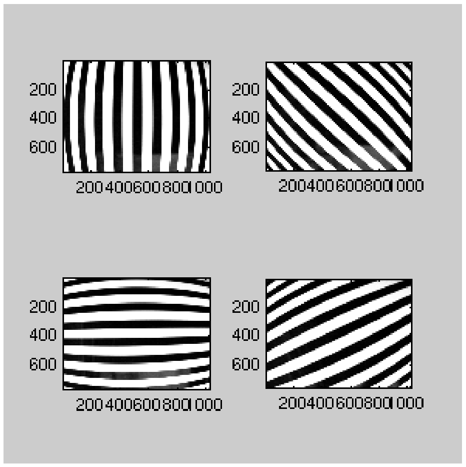
Figure 7: Overview of the camera calibration.

5.1 Lens distortion

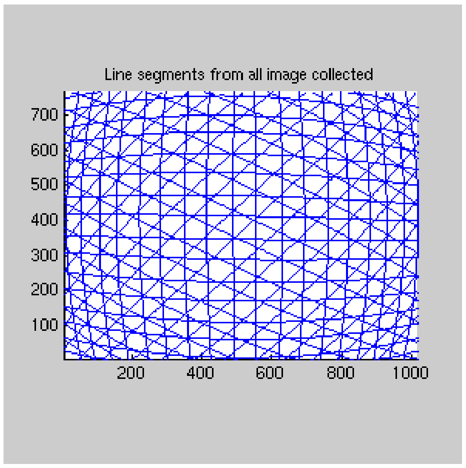
It was eventually decided to use cameras with wide angle lenses, due to the large coverage requirements. The techniques for calibration of such lenses are somewhat less standardized, and required some implementations and testing. Appendix C describes the radial tangent lens distortion model for fisheye lenses, that is used in our system.

The procedure to find the lens distortion parameters is as follows:

1. Make a striped calibration image.
2. Take a number of images of the calibration pattern in different orientation.

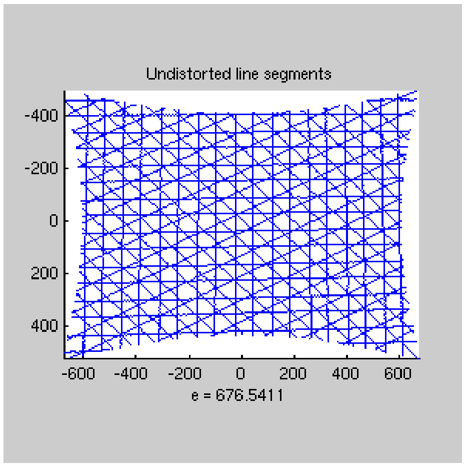


3. Find and collect line segments from all images. The chosen calibration pattern simplifies this procedure, compared to taking just one image of a pattern containing lines in different directions.

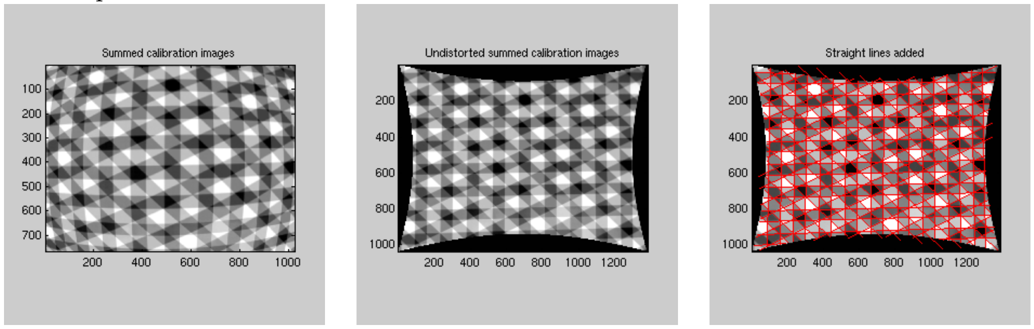


4. Optimize the lens distortion parameters. Choose the parameters that gives undistorted (straight) lines. The error function is computed as follows
   (a) Map the points on the line segments to the undistorted domain.
   (b) Estimate a straight line to each segment.
   (c) For each line segment, compute the closest distance from each point on the line to the fitted line.
   (d) Map back to the distorted domain before computing the sum of errors.

It may seem easier to compute the error in the undistorted domain. But the lens-parameters may rescale the undistorted domain, which can affect the error (the optimization may decide to rescale, i.e. shrink to nothing, instead of improving the straightness).



Example result:



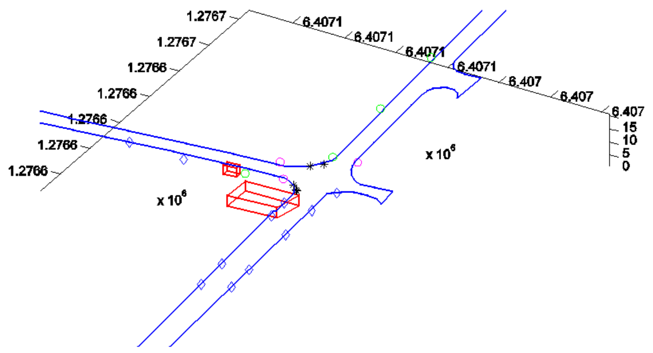
5.1.1 Comments

Some manual work had to be made to remove faulty line segments due to reflexes in the calibration images.

We also tried the radial polynomial model $r_d = r_u - \gamma r_u^3$ from [11] instead of the tangent model. This polynomial model gave twice the error in the example above.

5.2 GPS measurements

In order to estimate the mapping between the 3D and 2D domain, a set of measured 3D GPS points were used. Figures 8 and 9 shows some examples (see also figures 10-12).



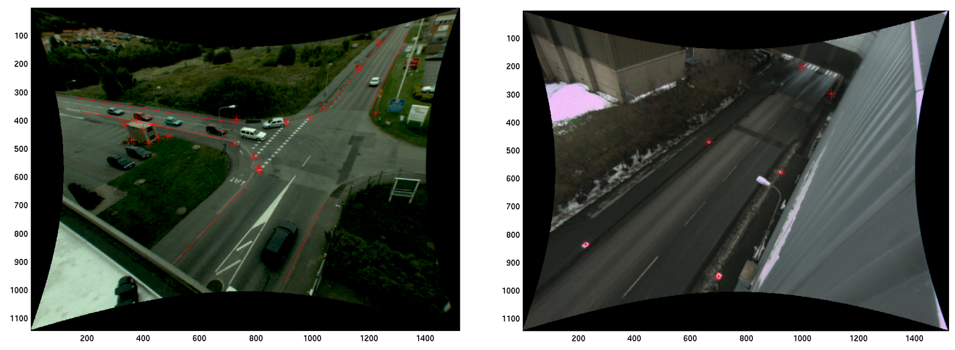


Figure 8: GPS measurements in Renova.

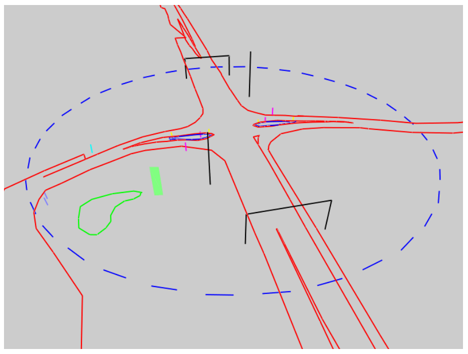


Figure 9: GPS measurements in Jung.

5.3 Ground plane

The ground was approximated with a plane. This approximation seemed sufficient for the two intersections in focus. For example, the difference between the measured GPS points and the estimated ground plane in Renova was at most ~ 20 cm in the selected region (a circle with 40m radius).

5.4 Projection Matrix

The theory regarding estimation of the ground plane and the mapping from the 3D world to the 2D image is described in appendix D. We discuss some general details here.

There are different ways to estimate the 3D-2D mapping (as described in the appendix). Initially, the idea was to find the mapping directly, but this method requires that the GPS points are sufficiently spread out in the 3D space. However, most of the points is located on the ground, i.e. in a 2D subspace, and some of the points were noisy. This limitation made the approach somewhat insufficient, especially since the focal length had to estimated simultaneously (the focal length is not part of the lens distortion parameters).

Another, indirect method, is to first estimate the mapping between the 2D ground plane and the 2D image (after first estimating the ground plane). Then, the full mapping can be estimated using some geometrical constraints (see the appendix). This method also gave a somewhat insufficient result, so a manually tuning of the focal length was performed instead of automatically estimation (an incorrect focal length will give a skewed result or an incorrect height scaling).

Furthermore, in addition to point correspondences, line correspondences was also included for the estimation to improve the result.

The procedure is as follows:

1. Collect 3D points and lines (e.g. using GPS).
2. Estimate the ground plane.
3. Get point and line correspondences. Manually mark the corresponding points and lines in the image.
4. Choose (guess) the focal length. Estimate the 3D-2D mapping (the projection matrix $\mathbf{P}$), use either the direct or the indirect method. Look at the result, by mapping the 3D points and lines into the image, and adjust the focal length until the result is satisfactory.

5.4.1 Renova and Jung result

Figure 10 shows an example result in the Renova intersection with two cameras. Figures 11-14 shows results from the Jung intersection with four cameras.

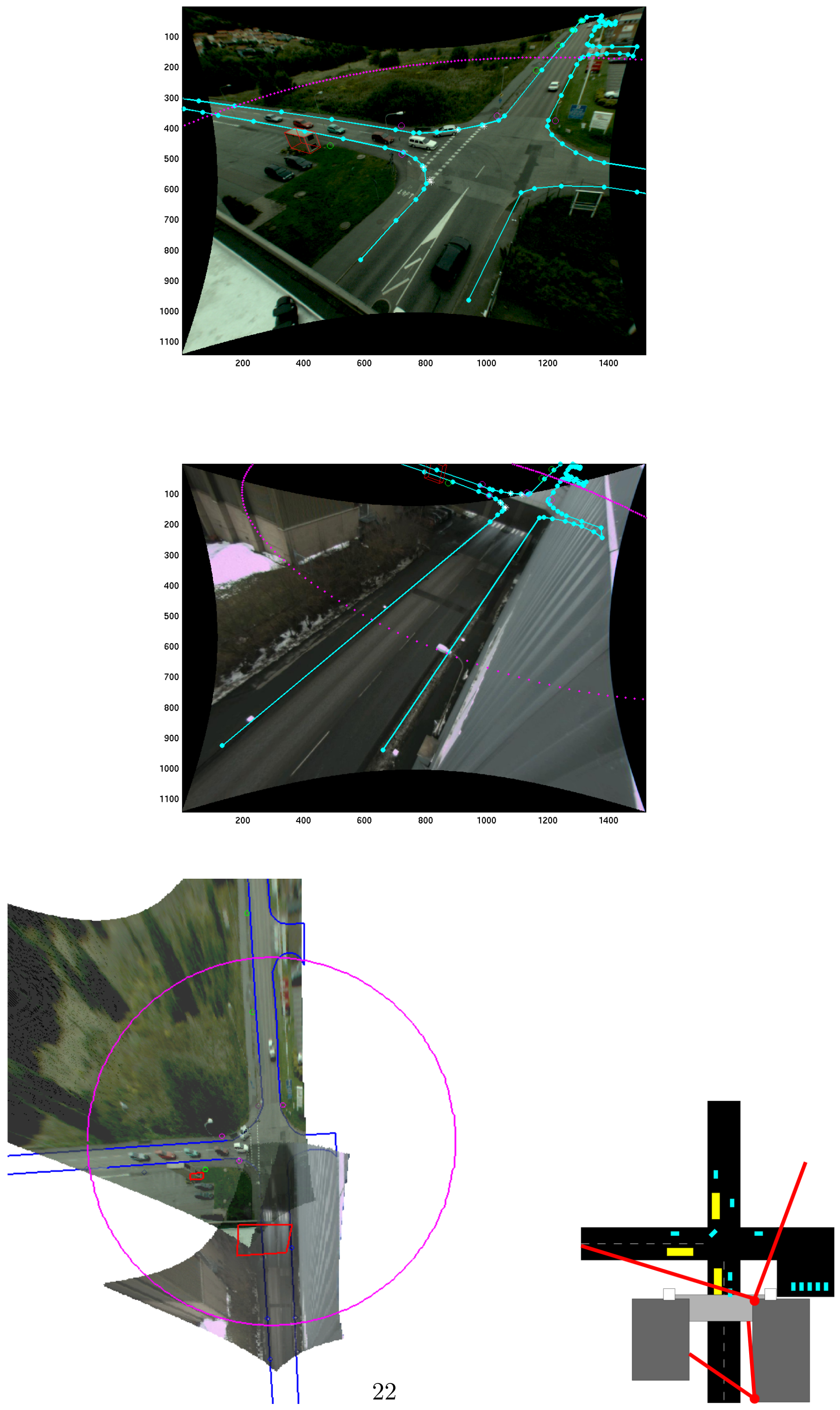
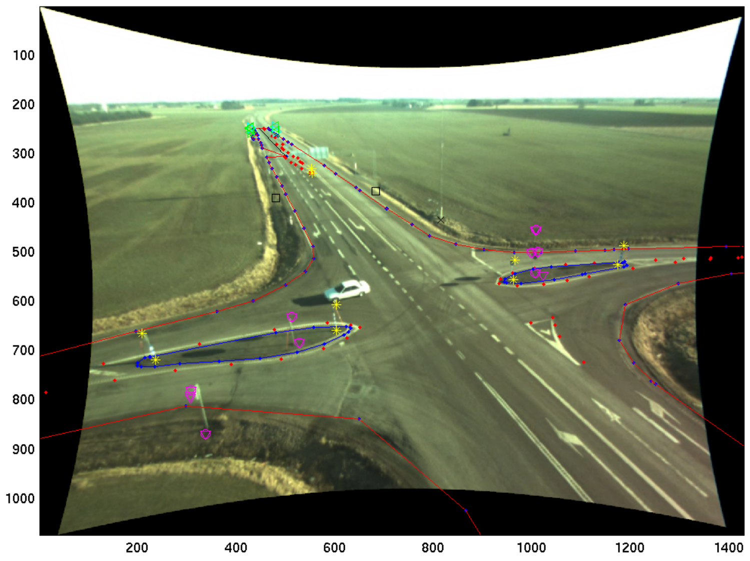


Figure 10: Resulting calibration in Renova.



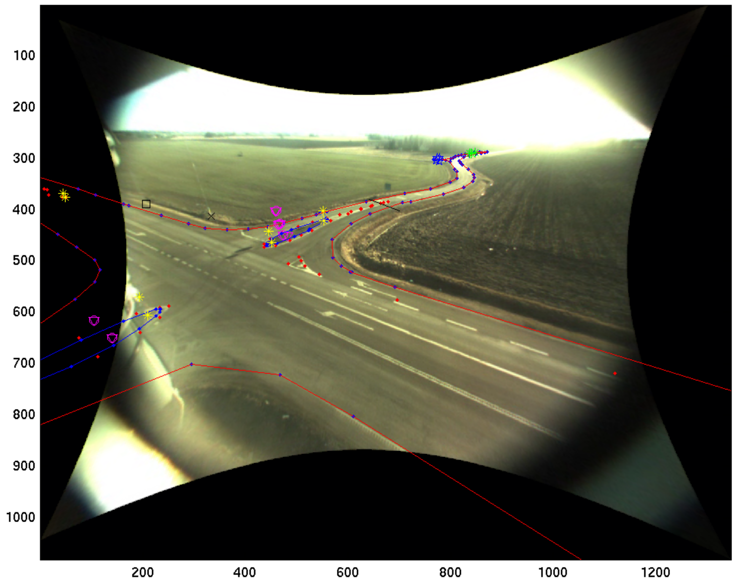
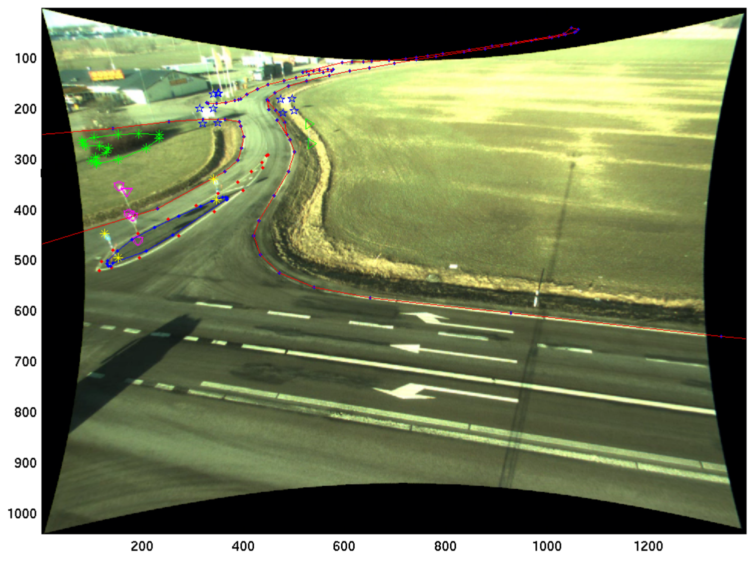


Figure 11: Resulting calibration in Jung for pole #1. GPS data mapped into the camera image.



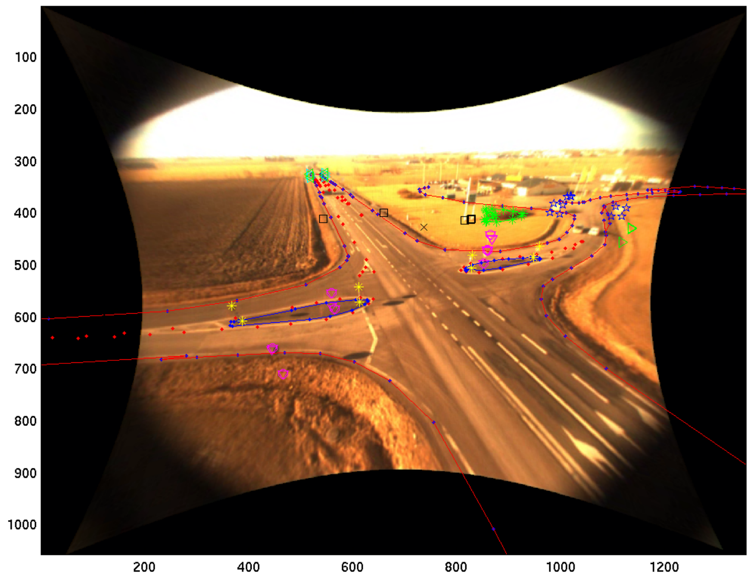


Figure 12: Resulting calibration in Jung for pole #2. GPS data mapped into the camera image.

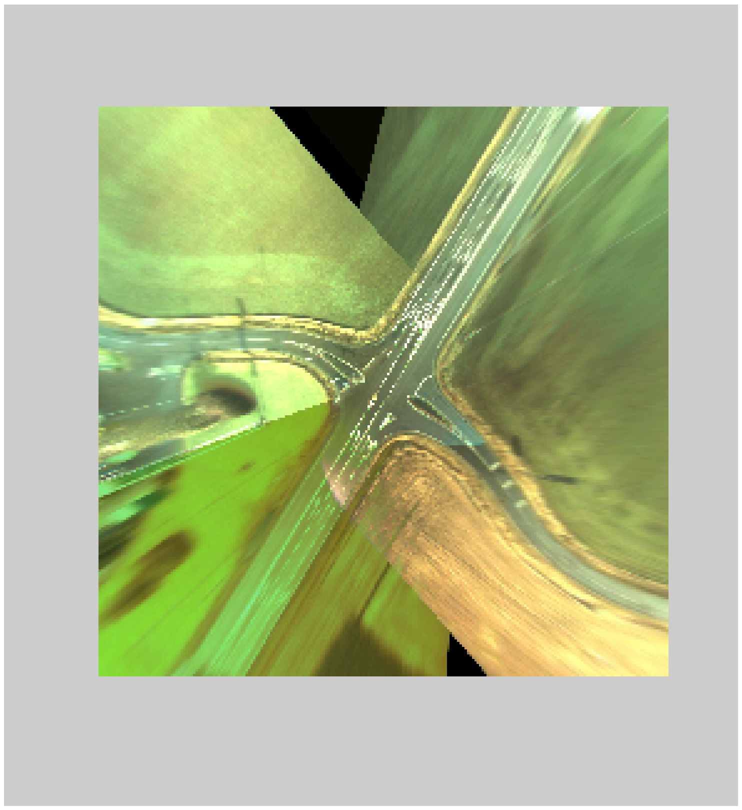


Figure 13: Overview of Jung.

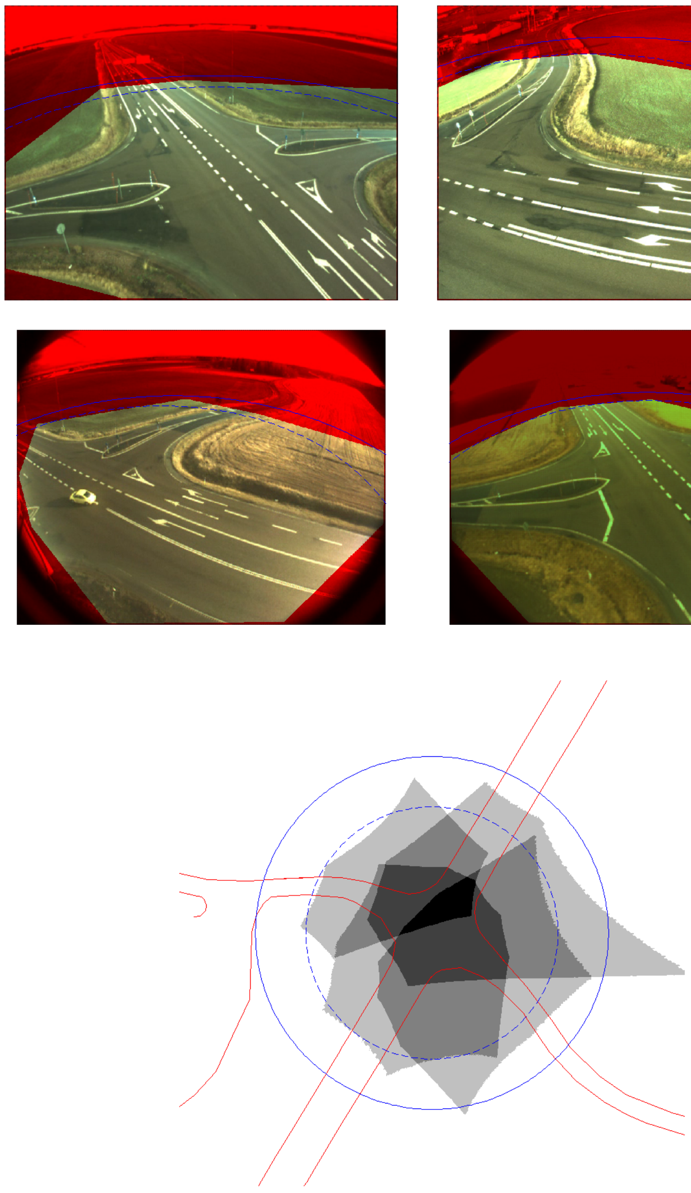


Figure 14: Selected coverage in Jung.

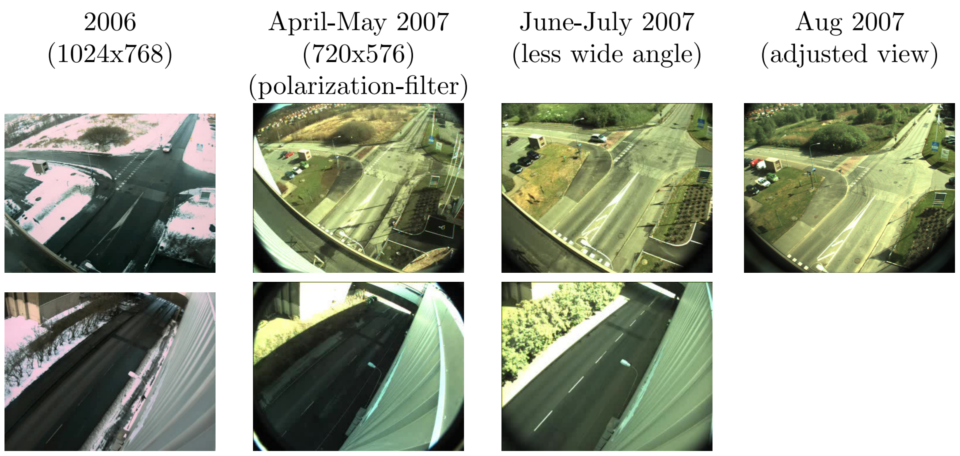


Figure 15: Different calibrations required in Renova.

5.4.2 Re-calibrations in Renova

The calibration had to be remade a few times in the Renova intersection, to improve the coverage and to make full use of the image pixels, see figure 15. Some of the initially measured GPS points is missing in the later versions due to repairs of the intersection, which affected the calibration somewhat. Care should be taken when selecting the points, and a more automatic procedure should be implemented if calibration is frequently required.

5.4.3 Re-calibrations in Jung

The calibration in Jung also had to be remade a few times, due to installation of polarization filters, to improve the accuracy of the vehicle position estimation, and due to a slight change in mounting position (cause unknown).

6 Camera ego-motion

Many intersections do not have buildings nearby, and cameras then have to be mounted on poles that sway due to the wind. Image analysis of video data, such as vehicle tracking, is easier if all motions in the video are caused by moving objects. The image sequence then has to be compensated for camera ego-motion. Moreover, the ego-motion compensation will also improve the accuracy of the camera calibration, i.e. the mapping between the 3D world and the 2D image plane. This will in turn improve the accuracy of image based 3D position estimation.

A method for simultaneous compensation of camera ego-motion and image rectification (due to lens distortion) has been implemented in GPU. Appendix E describes the theoretical details.

However, the video data is still encoded and decoded in software. The encoding is especially heavy when using mpeg4, so the algorithm is still not processing in real time. Currently, about 10fps is achieved, but there are four cameras in Jung and we have two dedicated computers for this processing (requires NVIDIA graphics cards) so the effective speed is about 5fps.

The ego-motion compensation method seems to work well in most cases, especially when combined with a statistical background model that reduce the remaining ego-motion that could not be compensated for (however, the background segmentation performance will still decrease somewhat due to the ego-motion). There are still some situations when the ego-motion is severe where the compensation fails, even if they are much more rare after the compensation.

Figure 16 shows two examples of good and bad results. Note though that even if there are some faulty detection in the second case, most of it is classified either as shadow or as highlight (see section 7). Also note that even if this classification sometimes also works without ego-motion compensation (as in the 'good' case in figure 16), the following position estimation will be less accurate.

We use a time window for the ego-motion compensation of about 10 seconds. This is a much shorter window than used for the statistical background model estimation, which may be one reason to the bad case in figure 16. The ego-motion compensation is based on estimating the average position over a time window, and the wind may temporarily give a constant offset on the mounting pole position which cannot be detected by the algorithm.

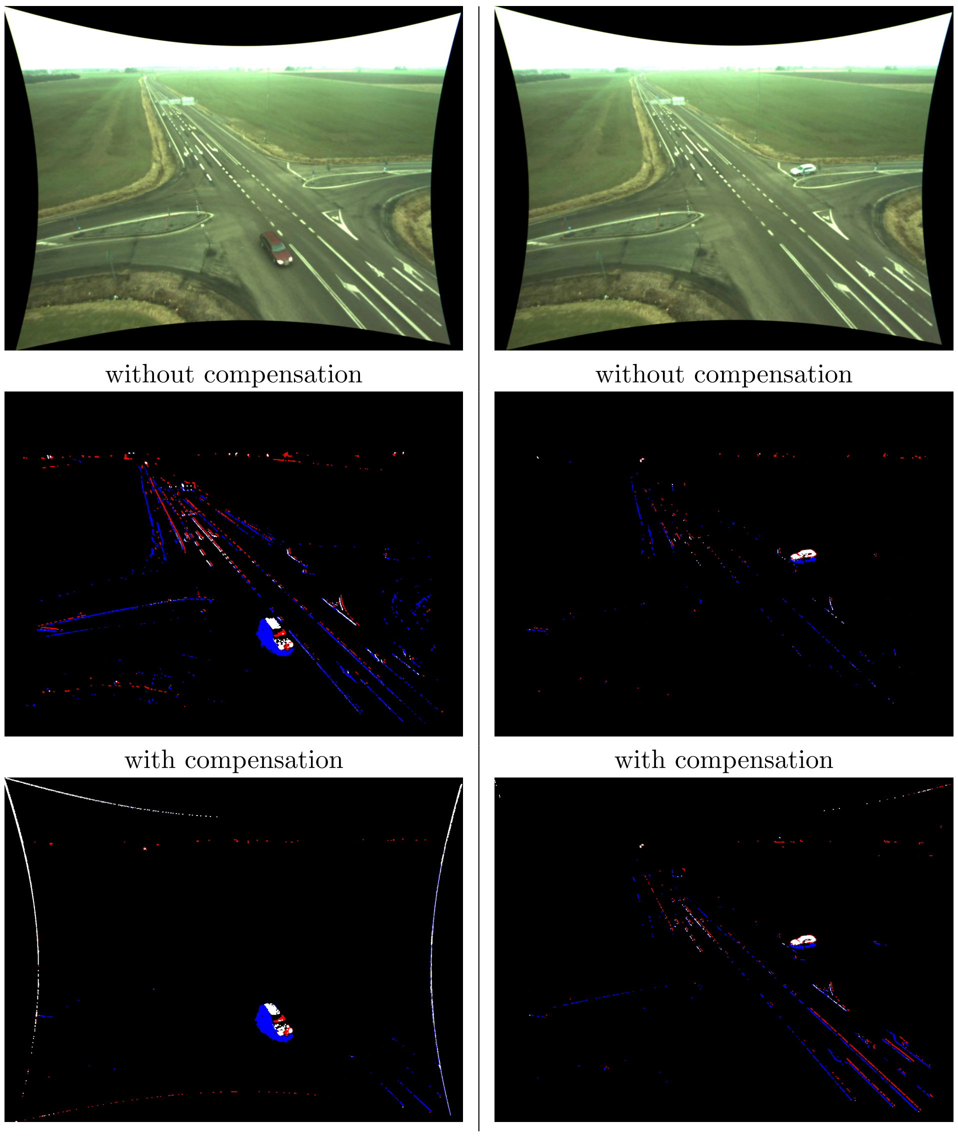


Figure 16: First column: Example of good results. Second column: Example of poor result (although most of the misclassification is detected as shadow or highlight). Top: Rectified image. Middle: Statistical segmentation without ego-motion compensation. Bottom: Statistical segmentation with ego-motion compensation. The colors denotes different foreground classes, i.e. white=foreground, blue=shadow, red=highlight.

7 Statistical background subtraction and foreground classification

The classification into background/foreground/shadow/highlight is a very important step in the tracking algorithm. However, it is also very difficult to choose parameters that will give a stable result in all weather and traffic conditions. We will first describe the method, and then show some results and discuss limitations.

7.1 Method

We use the statistical classification method in [35] for classifying each pixel into background or foreground. The method is a modification of the well known Stauffer-Grimson background subtraction method [30], with a somewhat different update rule and a lower regularization limit for the Gaussian standard deviations. Basically, a Gaussian mixture model is used in each pixel to estimate the color distribution over time and foreground pixels are detected if the color is unlikely to belong to the distribution.

The foreground pixels are further classified into foreground/shadow/highlight. In [35] the Stauffer-Grimson background subtraction is combined with the shadow detection method in [16] to get a foreground/shadow classification. In short, a pixel is classified as a shadow pixel if the color lies in a cylinder region between the black (the origin) and any of the center colors of the background Gaussians. We have also applied the same idea for highlight, so that a pixel is classified as a highlight pixel if the color lies in a cylinder located on the opposite side of any of the Gaussian center colors. Figure 17 illustrates the class regions.

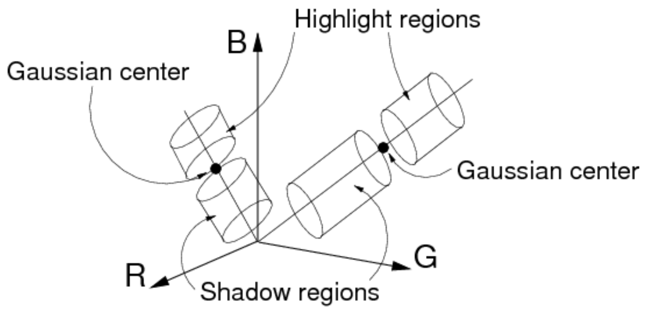


Figure 17: Illustration of the shadow/highlight classification in the RGB color space.

We define highlight as pixels that do not belong to the foreground objects, and that are brighter than the average background color (as in figure 17). Highlight can occur from many different phenomena, e.g. reflection on a bright vehicle onto the ground, bloom and streak effects, and rapid light changes caused by moving clouds, see figure 18.

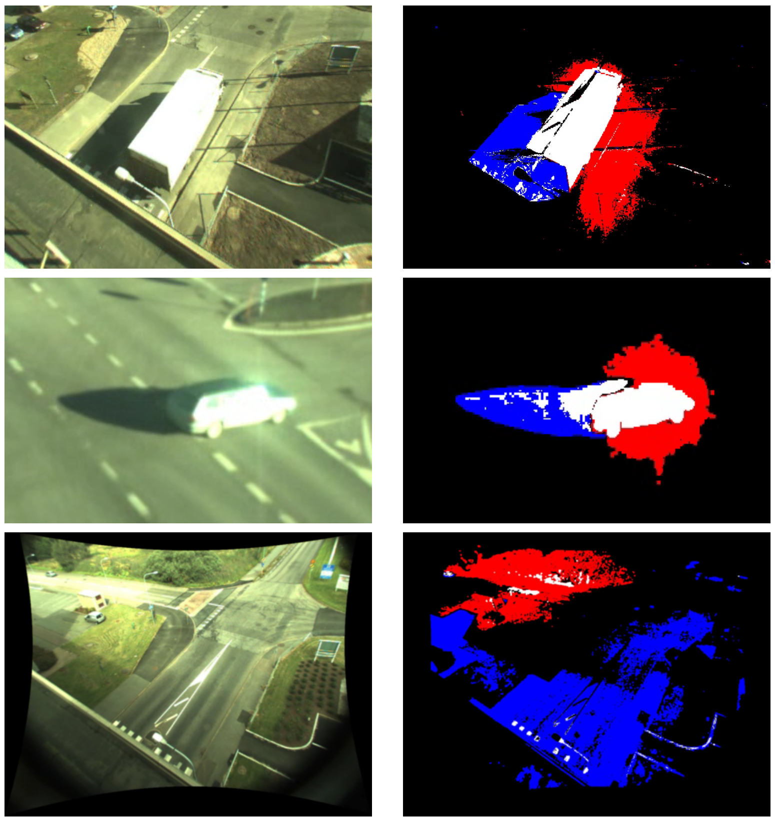


Figure 18: Examples of highlight and shadow detection. Top: Reflection of a bright vehicle onto the ground. Second: Bloom and streak effects. Third: Sun reappearing from behind moving clouds. The colors denote the following: black=background, white=foreground, blue/darkgray=shadows, red/lightgray=highlight.

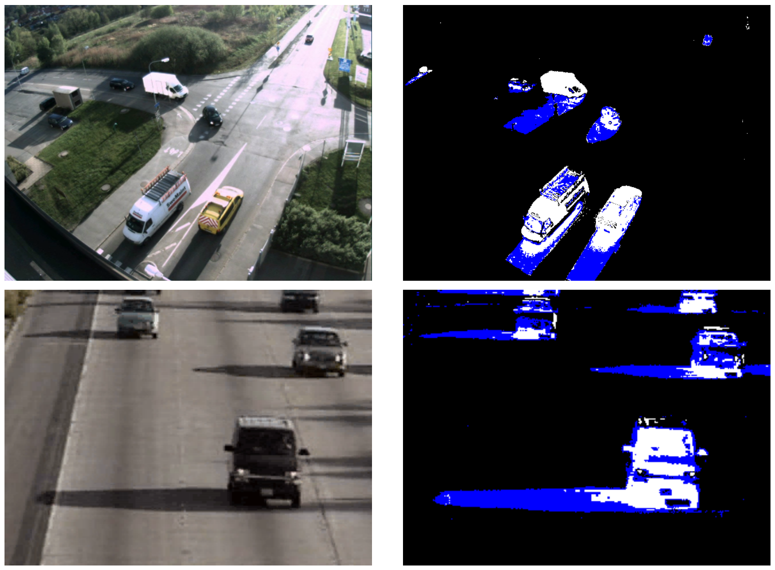


Figure 19: Examples of difficult classification in the umbra region.

Figure 18 also shows the result of the foreground/shadow/highlight detection. Much of the highlight can be detected in these cases and removed by the algorithm. The entire algorithm for background modeling and classification of pixels into background/foreground/shadow/highlight is summarized in appendix F.

7.2 Limitations

This sections describe some limitations of the background subtraction and foreground classification method. It is very difficult to achieve a classification performance for general conditions that works as well as tuning the parameters (learning rate, classification thresholds, etc.) for certain weather conditions and traffic density. These compromises will give rise to artifacts as discussed below.

7.2.1 Shadow misclassification

There are many situations where shadow and foreground objects are misclassified, we will here give some examples. Note that the 3D box modeling of vehicles that is described later in this report can deal with some misclassification, but the performance would be improved if the errors could be reduced.

Figure 19 shows two examples of the shadow detection. Note that there are some misclassification, especially in the shadow near the vehicle, i.e. the umbra region which is the dark core of the shadow.

Figures 20 and 21 shows other examples of good and poor results. Most of the misclassified cases have in common that the shadows are much darker than in the correctly classified cases. This happens more often for larger vehicles, where the umbra region is larger. There is also one case where the black car is brighter than its shadow. As mentioned above, the shadow theory in [16] that is used here (and in many other models) models shadows as being darker than the background, but not too dark. Extending the shadow class to more darker regions will increase the misclassification for dark vehicles. Still, it may perhaps be better to just set all dark pixels to background, including dark vehicles, in the shadow classification and deal with dark vehicles in a separate way.

7.2.2 The part sun/cloud problem

Changes in light due to the sun appearing and disappearing from behind moving clouds are quite common. Some of these changes can be classified into shadow or highlight, but light changes can also be classified as foreground (the white regions in the figures). These regions will initiate tracking filters and cause illusionary vehicles. Figures 22-24 shows examples with different results.

Furthermore, the light variations will decrease the performance of the statistical background model estimation, since they are detected as foreground and therefore prevents the background model to be updated. For example, the convergence of the background model will be very slow if the variations continue for a longer period of time (e.g. leaving ghost artifacts as in figures 23-24).

7.2.3 Choosing time window

One fundamental problem is the time window for learning of the statistical background model. It is difficult to tune the time window to suite all varieties of traffic density and light conditions. For example, [17] mention that the learning rate of the background update is tuned depending on the type of sequence and the expected speed of motion, which is not desirable when designing an automatic system. Heavy traffic, or if the light is varying too much due to moving clouds, will cause slower convergence of the background model since the model is not updated in the foreground/shadow/highlight regions. As we have seen, some of the light variations can be classified as shadows or highlight, but there will still be misclassification and it is desirable that the background model adapts to new conditions.

Figures 25-26 shows examples of behavior during the initial adaptation period (using the same learning rate in both cases). The time to learn the background depends on the time that the background has been visible. Hence, the length of the adaptation period depends largely on the amount of heavy traffic.

Furthermore, vehicles that are temporarily stationary for a longer period of time will eventually disappear and become part of the background model, figure 27 shows an example. The statistical background model can remember the previous background for some time, so that when the vehicle assumes its coarse it does not leave a ghost vehicle. However, there will be ghost effects for vehicles that are temporarily stationary for a sufficiently long period of time (depending on the learning rate of the background model), see figure 28.

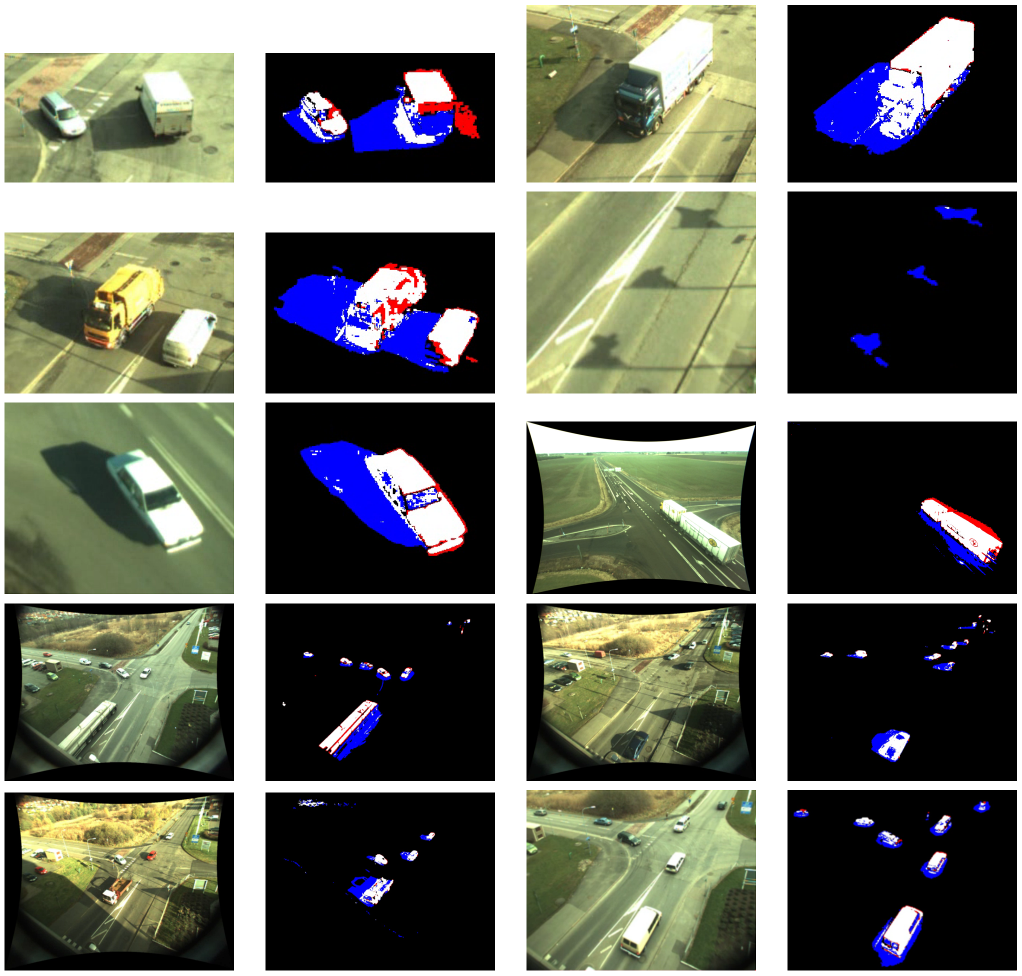


Figure 20: Examples of predominantly successful classifications.

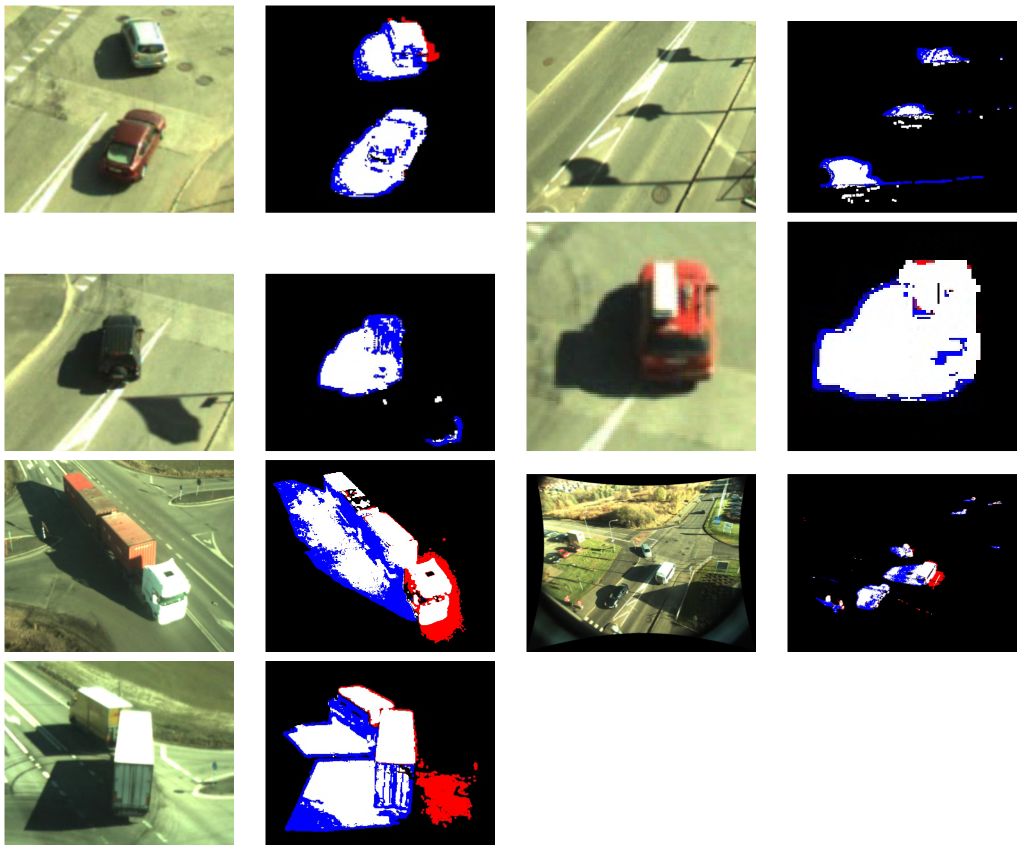


Figure 21: Examples of misclassifications.

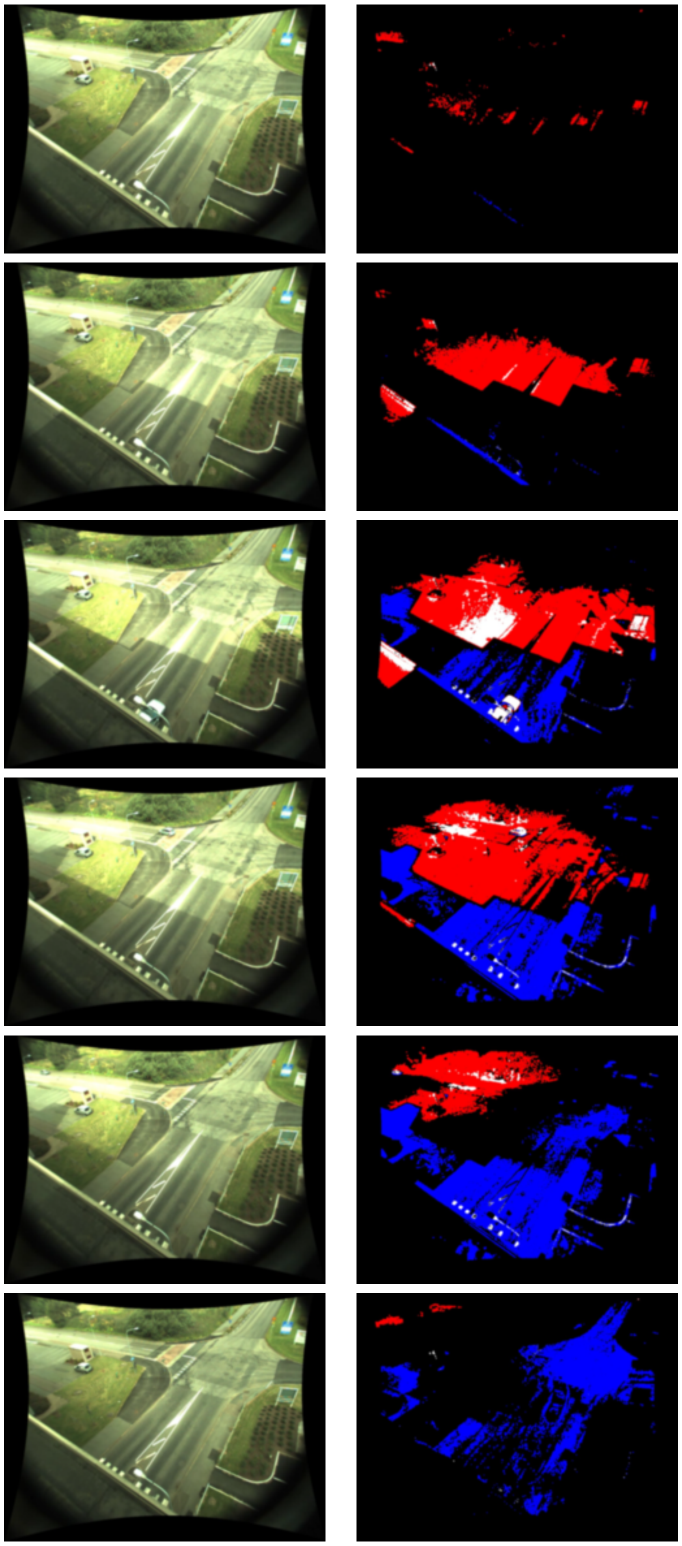


Figure 22: Example sequence of sun reappearing from behind moving clouds.

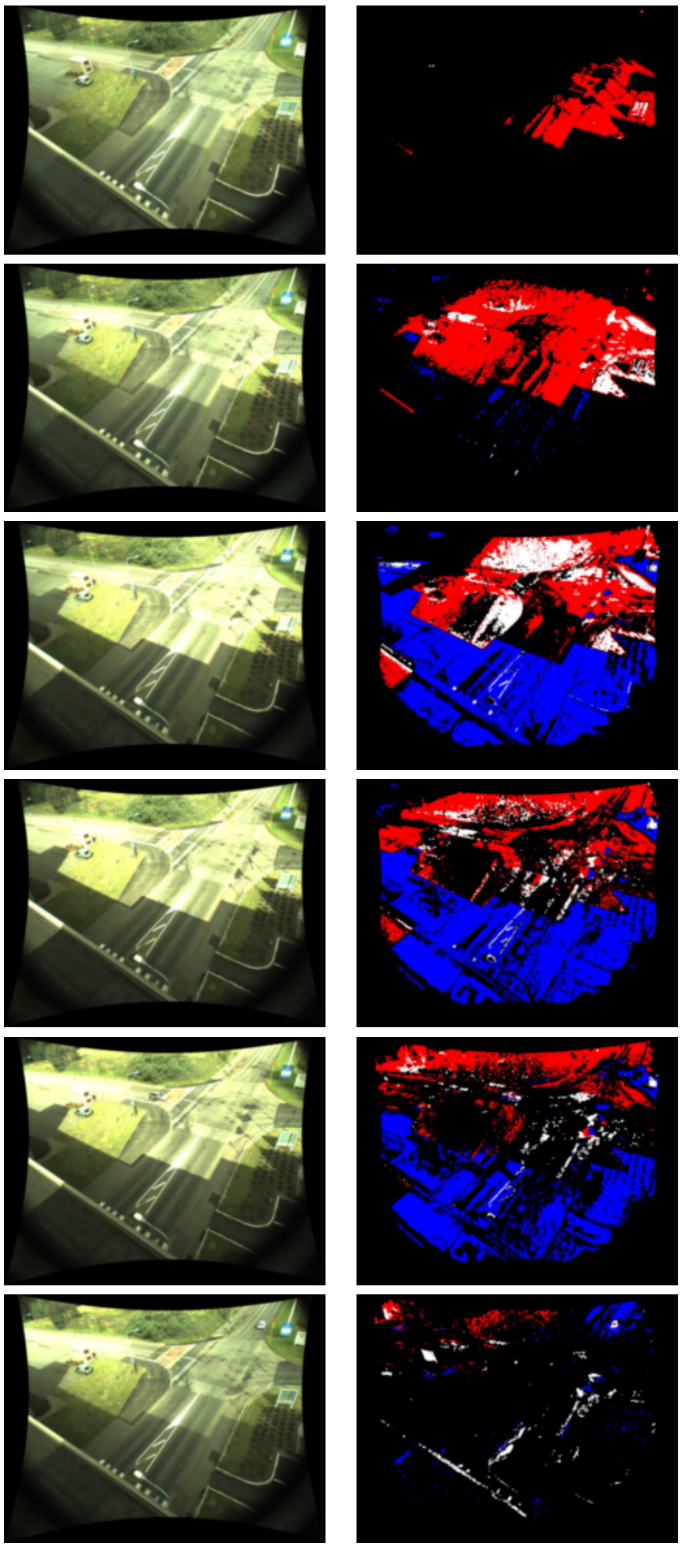


Figure 23: Example sequence of sun reappearing from behind moving clouds.

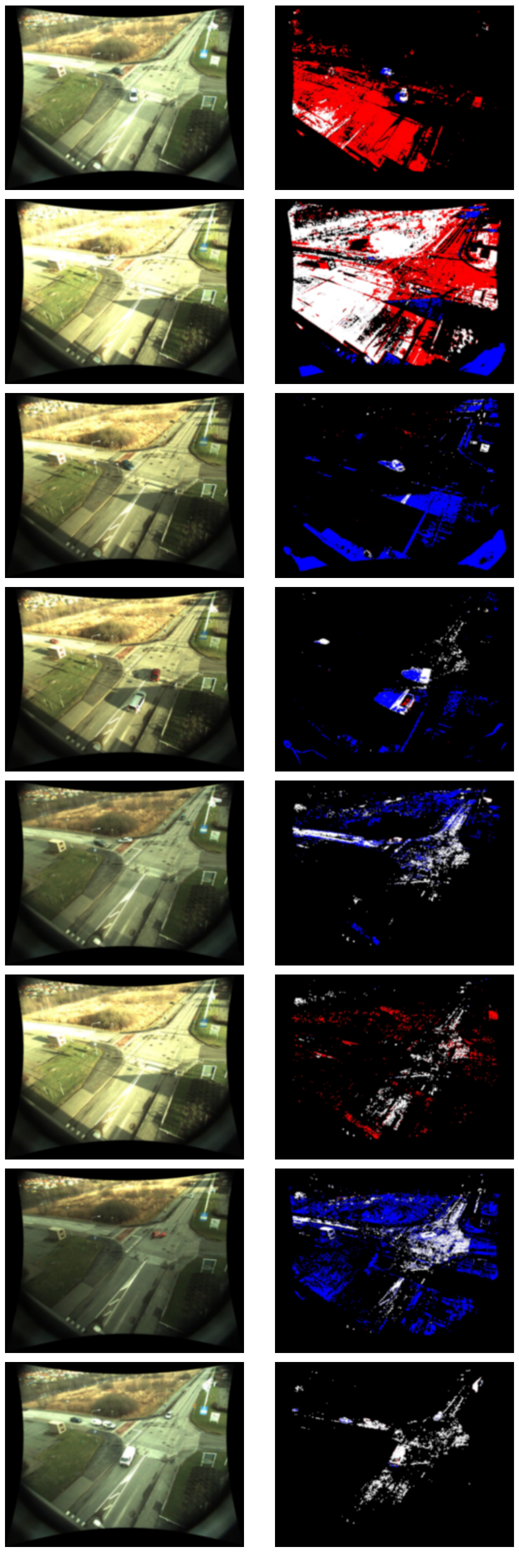


Figure 24: Example sequence of sun reappearing from behind moving clouds. The result in this last case is worse than the previous ones, partly because the light variations have been quite extended, so that the background model has not converged.

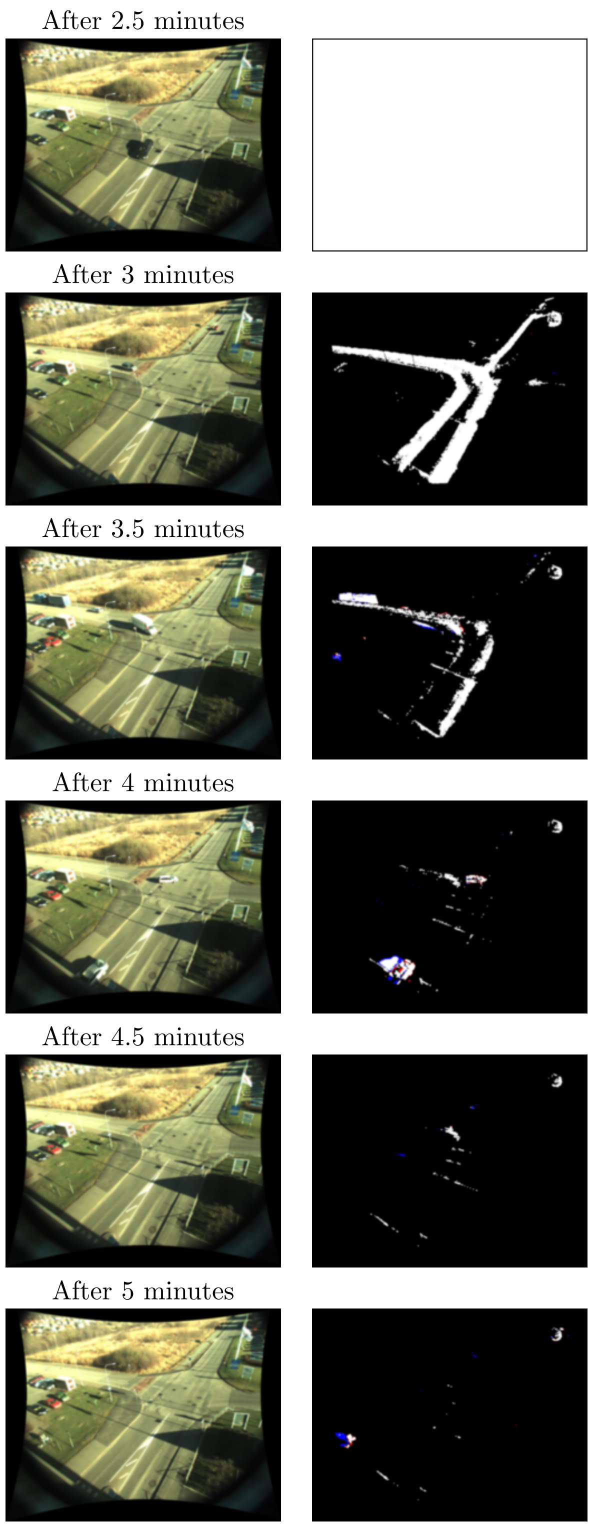


Figure 25: Example of classification behavior during the initial adaptation period. It takes longer time to learn the background in regions where moving objects are present.

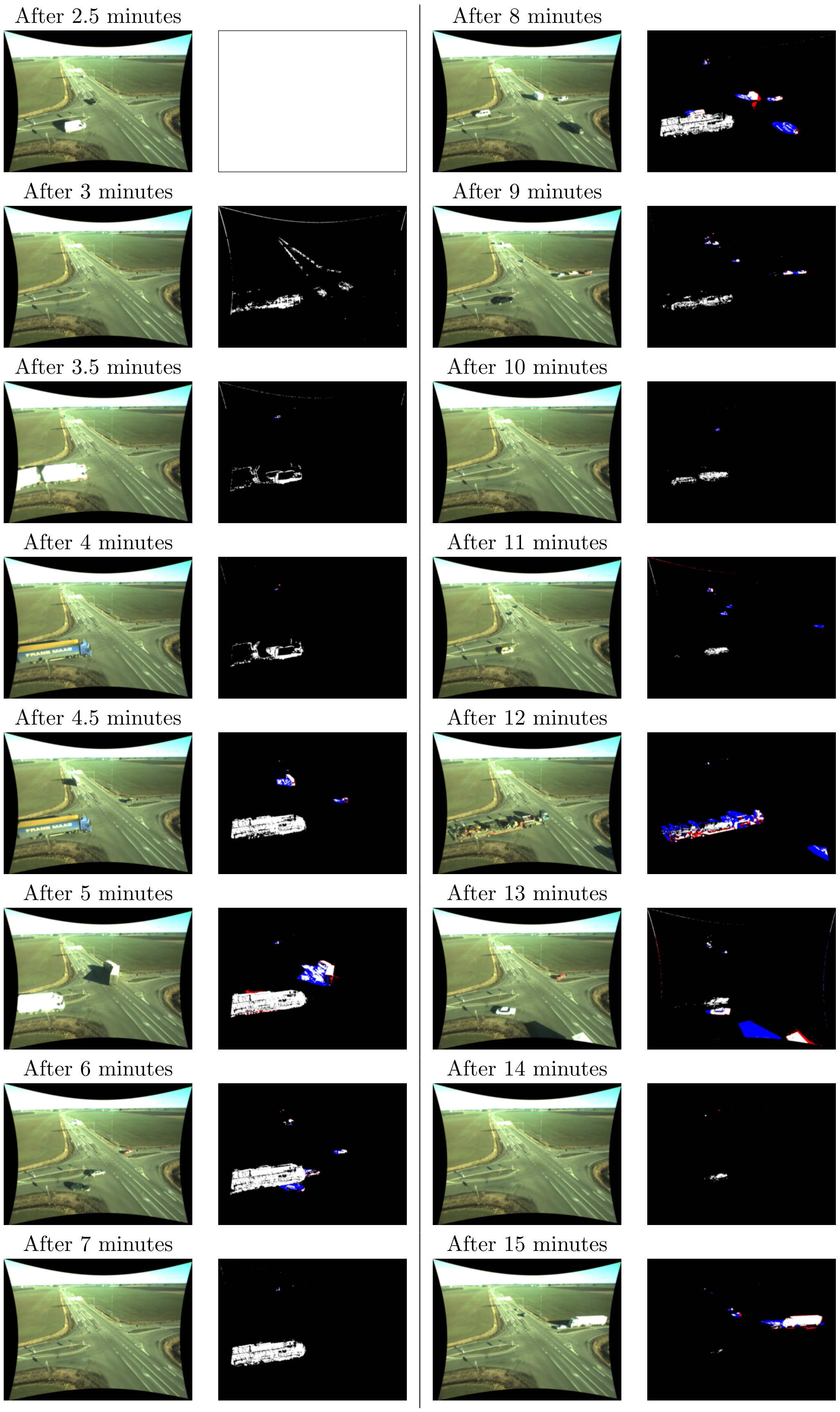


Figure 26: Example of classification behavior during the initial adaptation period (same learning rate as in figure 25. The adaptation period on the left exit road is particularly long, due to heavy traffic (several trucks have been temporarily static for a longer period of time).

We use a non-causal version of the subtraction method which helps to reduce the ghost vehicle effects somewhat, but the improvement is limited.

7.2.4 Other difficult weather conditions

There are some weather conditions that this type of classification method is not suitable for. Figure 29 shows some examples. Days which contain wet asphalt are manually found and discarded.

7.3 Discussion and related work

In a recent approach, [17] use a method similar to ours to detect shadows, but they also added edge cues to improve the shadow detection. Another recent similar method is [6] that classifies pixels into even more categories such as reflections (similar to our highlights), ghosts, and fluctuations. They use truncated cone regions instead of our cylinder regions to model the different class regions in the color space (however, they use truncated cones which is similar to cylinders). It is also possible to use more light invariant features in the Stauffer-Grimson method, as in e.g. [1]. It is likely that some of these new methods can improve the classification.

Still, method parameters are often tuned to certain sequences and light conditions, and it may be difficult to have perfect detection under general and varying conditions (the many publications in this field also indicate this challenge). Another problem is that light from a vehicle reflects onto the ground and changes the color on the ground depending on the color of the vehicle, this will affect the shadow classification if one vehicle reflects light into another vehicles shadow.

It may be better to either use a longer adaptation period than done here, and deal with change in light differently, or use a shorter period and modify the segmentation criteria. An alternative (or a special case) to background subtraction could be to use a very short time window, for example frame difference, but that requires a different way for segmenting objects. Using color for segmentation is in some cases helpful, as in e.g. [23] (where 2D rectangles are used in combination with color to segment objects). However, many objects are multi-colored, and parts of objects often resembles the background color. Using motion features can also be helpful, but an issue then is how to deal with temporarily static vehicles, which is not detected by frame difference or motion features. A combination of features for segmentation would probably give the best result, if the multi-feature fusion can be designed properly.

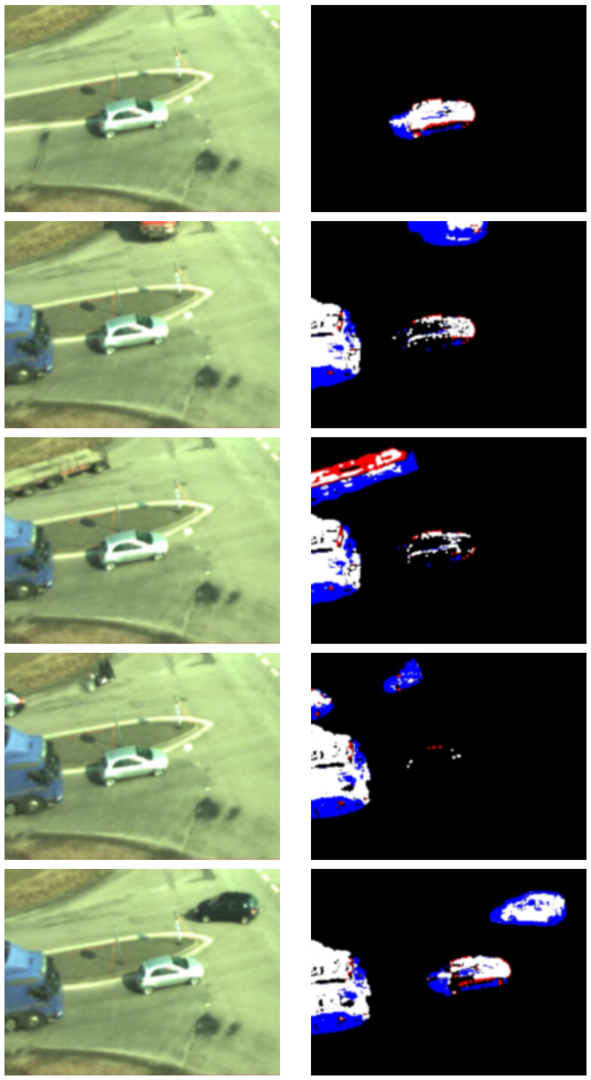


Figure 27: Example of the background model adapting to a vehicle.

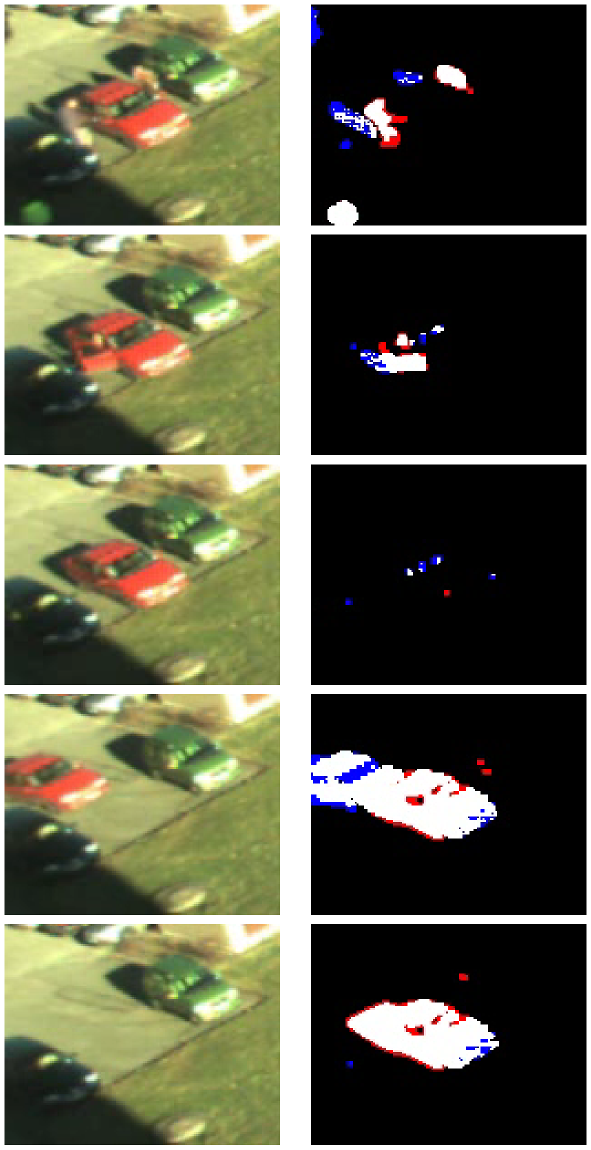


Figure 28: Example of ghost effect.

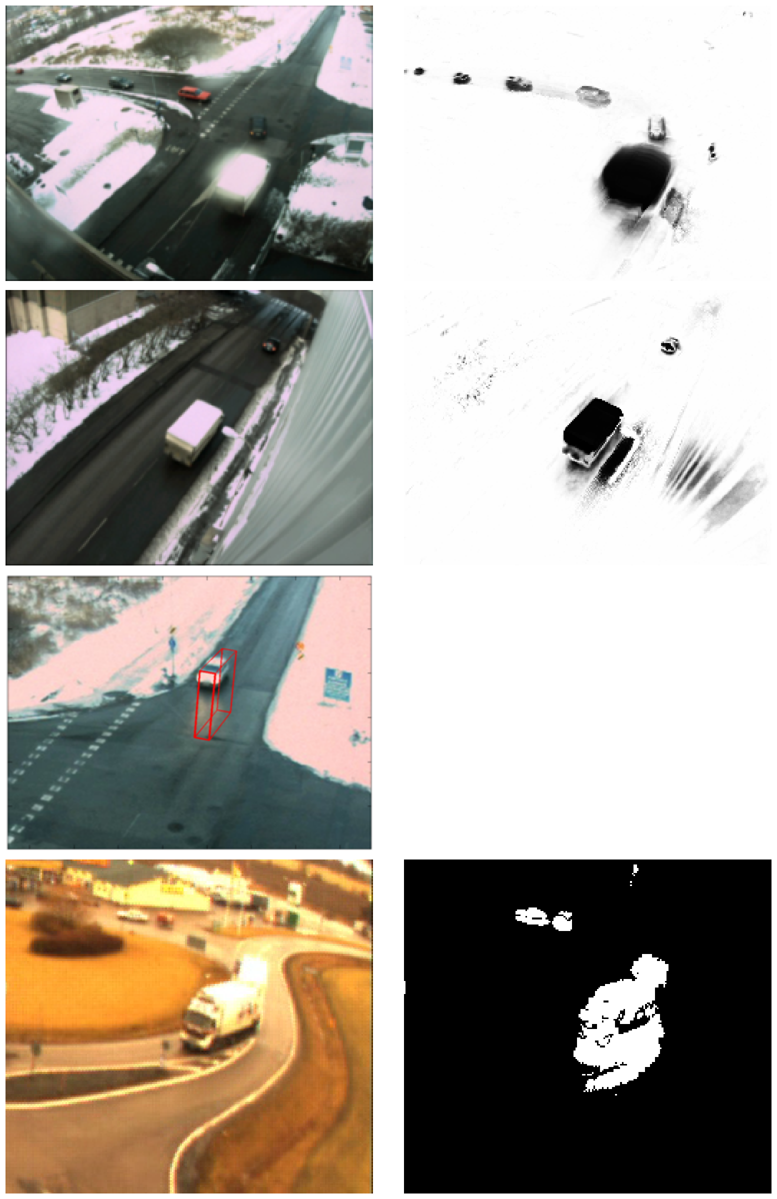


Figure 29: Examples of difficult data due to rain/snow on the lens, building reflection (which in this particular case be solved by ignoring the image region containing the wall), head light reflection, and wet asphalt reflection (the figures have been created at different points in time and are therefore visualized in different ways).

8 3D box shadow simulation

Let $\mathcal{B} = \{L, W, H, x, y, \varphi\}$ denote the state of a 3D box on a ground plane in 3D, where $(L, W, H) = (Length, Width, Height)$, (x, y) is the position on the ground plane, and φ is the orientation.

To simulate an image of a 3D box we need knowledge of the ground plane and the camera calibration parameters, i.e. lens distortion parameters and the projection matrix that defines the mapping of 3D world points to 2D image points. The calibration details are explained in section 5, we will not go into the calibration details here, but just mention that all the parameters are estimated using measured GPS points and general calibration techniques.

Moreover, to simulate box shadows we use the software package [5] to compute the solar position from date, time, and location on Earth. Figure 30 shows an example of the shadow simulation.

As an interesting footnote, according to the simulation, the sun elevation angle in the middle of Sweden is at best about 55°, thus casting long shadows even in summertime.

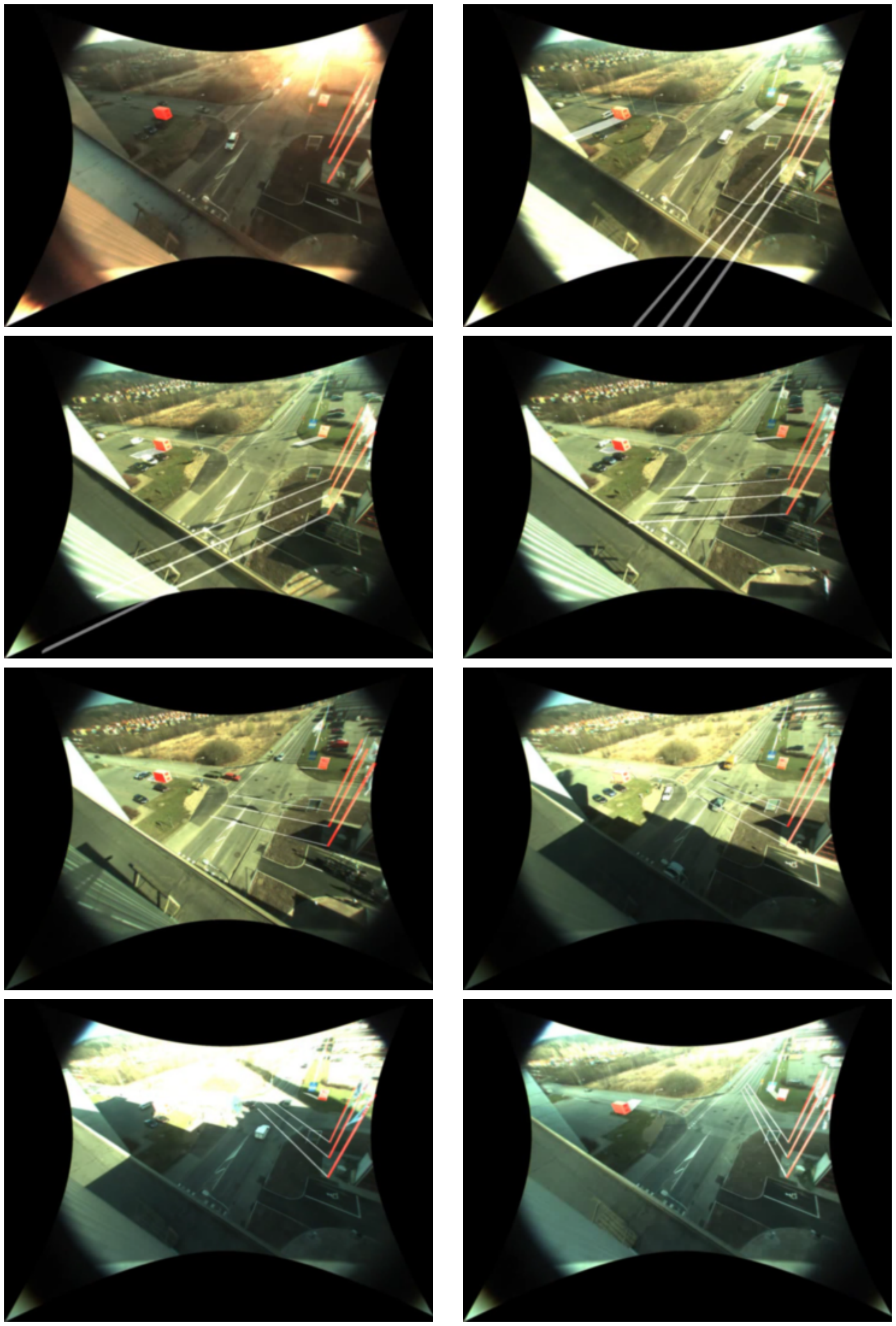


Figure 30: Example on light variations during a sunny day. The small house, three flag poles, and a sign are modeled and their simulated shadows can be compared to the true shadows. Note that some of the error between the true shadows and the simulated shadows on the flag poles may be due to the poles not being absolutely vertical.

9 3D box optimization

Figure 31 illustrates the problem of modeling region segments in an image with a 3D box. The figure is only intended to give an idea of the method. We will in this section go through the procedure in more detail, including how to incorporate shadow detection and simulation into the optimization (which is not illustrated in the figure).

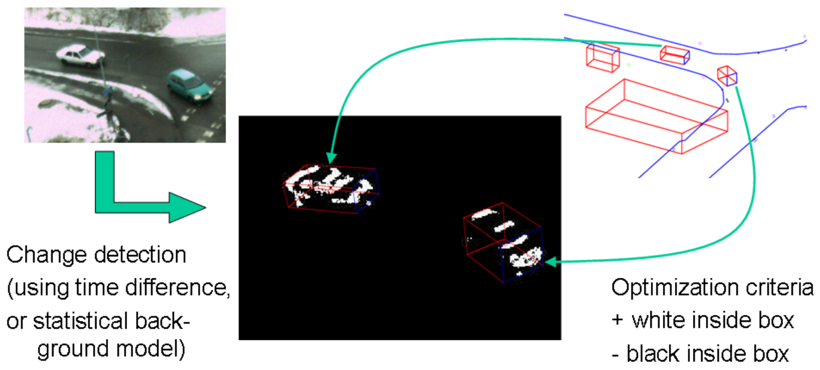


Figure 31: Illustration of the 3D box modeling of image region segments (without the shadow detection/simulation).

We will first define the similarity measure between a classified image which contains the classes background/foreground/shadow and a simulated image of a 3D box including the shadow of the box. Second, we will describe the optimization procedure to find the 3D box that maximizes the similarity measure.

9.1 Similarity function

Let $I(\mathbf{x})$ be the classified image and $S(\mathbf{x})$ be the simulated image, and represent the classes as follows

$$I(\mathbf{x}), S(\mathbf{x}) = \begin{cases} 0 & \text{if background} \\ 128 & \text{if shadow} \\ 255 & \text{if foreground} \end{cases} \tag{1}$$

Furthermore, let $\mathcal{V}$ denote the valid pixels, i.e. pixels that are not occluded by other objects (buildings, vehicles, image rectification borders, etc.). Define the similarity between the classified image and the simulated image as

$$s(I, S) = \sum_{\mathbf{x} \in \mathcal{V}} p(I(\mathbf{x})) S(\mathbf{x}) \,, \tag{2}$$

where

$$p(I) = \begin{cases} p_b < 0 \text{ if I=0 (background)} \\ p_s > 0 \text{ if I=128 (shadow)} \\ p_f > 0 \text{ if I=255 (foreground)} \end{cases} \quad (3)$$

For example $p_b = -0.2, p_s = 0.1, p_f = 1$. This function is quite ad-hoc, but it has some nice properties. A simulated box is rewarded more than its simulated shadow in a detected shadow region. For example, black cars are often misclassified as shadows, in which case the box should be covering the region instead of its shadow.

Initially, this similarity measure was intended to improve the box fitting in sunny situations when the shadow was misclassified as foreground. But the improvement compared to just using one class (foreground) and simulating shadows was minor. This is because knowledge of the solar position (and box orientation) seems sufficient to reduce the ambiguities. Instead, the improvement was most evident on days with diffuse light when the light causes shadows to appear around the vehicles. Figure 32 shows some examples of box optimization (explained in the next section) using this similarity function.

We also tried to use correlation as a similarity measure, but the results were worse. Correlation often causes the optimization to terminate at some local optimum if the classified region is sparse. Our measure favors large boxes, which helps to glue sparse segments together, unless there is too much background in-between, or if the other segments are invalid (e.g. covered by another box).

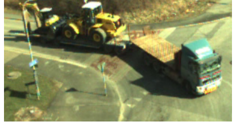
9.2 Optimization procedure

We employ a fairly simple optimization method. Basically, the algorithm is initiated with a predicted box state (from e.g. a Kalman filter) and tries different changes of size and position iteratively to find the box state that gives the highest similarity measure. The changes in size are not arbitrary, but chosen from a list of common sizes, see table 1 for examples[1]. The iteration loops over box sizes and position transformations, with gradually decreasing step size, see algorithms 3 and 4 in appendix G for details.

Figure 32 shows some example results. For comparison, the same figure also shows the result when simulating, but not detecting shadows, as well as when detecting and removing shadow pixels (and without simulating shadows).

This list does not cover all types of vehicles, especially not in industrial areas where uncommon types of vehicles occur more often than in an inner city, for example (the last one is not that common...):

[1]



Vehicle type	(Length, Width, Height)
Pedestrian	(1,1,2)
Motorcycle	(3.00,1.00,1.50)
Small car, VW Golf	(4.20,1.73,1.48)
V70	(4.73,1.86,1.56)
Van, VW Multivan	(4.89,1.90,1.94)
Van, slightly bigger	(6.00,2.00,3.00)
V70 with trailer	(7.73,1.86,1.56)
Truck, Volvo FL/FE	(9.00,2.50,4.00)
Extra ad-hoc size	(7.50,2.25,3.50)
Extra ad-hoc size	(10.00,2.50,3.50)
Extra ad-hoc size	(12.00,2.50,4.00)

Table 1: Examples of vehicle sizes.

The other methods are in some cases somewhat better, and a topic for future research may be to improve the similarity measure even further, but the new proposed method is better on average.

We give some observations from experiments here. Initially, we tried to use free sizes of boxes in algorithm 3, i.e. using the transformations $\{\mathcal{T}_n\} = \{$ Increase/decrease front position, Increase/decrease rear position, Increase/decrease height, Increase/decrease x, Increase/decrease y $\}$. However this seemed too unstable if the foreground/shadow-classification fails too much, and using a fixed set of sizes to reduce the degrees of freedom gave a more stable performance.

We have also tried to optimize with the shadow simulation as a degree of freedom (testing both turned on and off), but this also gave a more unstable result.

Moreover, we also tried to add change in orientation to the list of transformations, but it is too unstable (at least for some object views) to include a free change in orientation in the optimization when using only a single camera view. One solution might be to add a punishing term for the orientation in the similarity measure, i.e.

$$s' = s \cos(\varphi_0 - \varphi)^{p_\varphi sign(s)} \tag{4}$$

where φ_0 is the initial estimated orientation. However, the utility of this rule is still unclear.

We have also made experiments with merging of estimates from up to four cameras with partly overlapping views of an intersection (see e.g. figure 14 on page 26). Initially we tried to optimize the 3D box to all views simultaneously, by adding the similarity measures from each camera. However, this requires a very accurate time synchronization between the cameras. Otherwise the optimized box usually becomes larger than the actual vehicle since the box tries to cover all camera blobs (since foreground is rewarded more than background is punished). If the cameras are not fully in sync, it appears to be better to optimize to each camera separately and then merge the resulting 3D boxes by e.g. averaging the

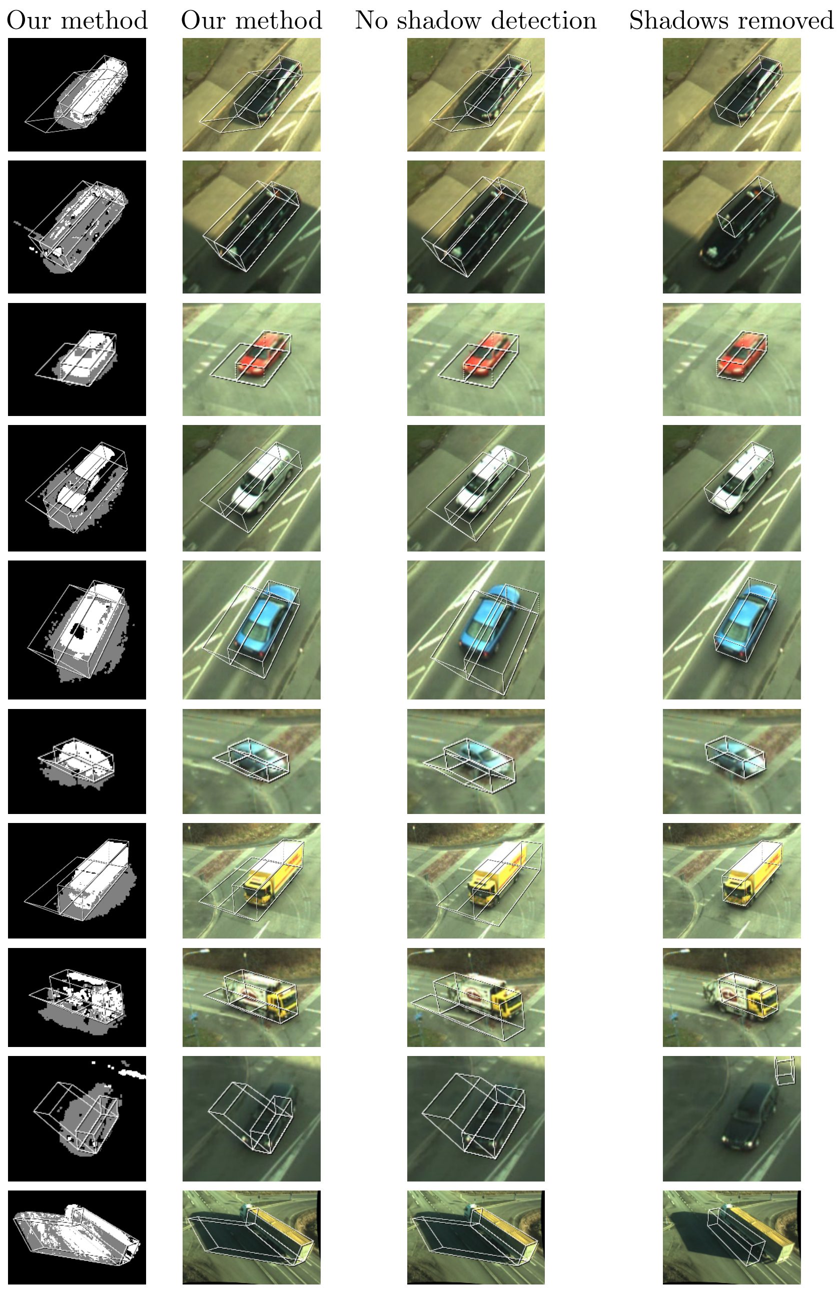


Figure 32: Examples of results from algorithm 4, using fixed parameters in (3). First and second column: Optimal box using our method, overlaid on the classified and original image respectively. Third column: Optimal box when not detecting shadows (i.e. setting all shadow pixels in the left column to foreground). Fourth column: Optimal box when removing all detected shadow pixels and without simulated shadows. The orientation is manually selected here, but could for example be predicted from previous positions or in some cases from known lane orientation. The initial position is roughly estimated from the image blob.

state parameters.

The optimization algorithm is very crude and may sometimes end up in a local optima if the initial position is too far from the optimum. It can then be useful to add a few more random transformations to the list in algorithm 3, or to use more elaborate optimization schemes. However, the improvement has not been significant when using prediction from previous frames (e.g. using a Kalman filter), and the estimation of size can also be improved by averaging estimates in time (as described below).

10 Kalman filters for prediction and smoothing

During tracking, the box state is predicted in the next frame. This prediction is used for initialization of the box optimization in section 9. A causal non-linear Kalman filter is used for the prediction. A non-causal Kalman filter is also used after the tracking, to smooth the trajectory measurements. We will here describe the various models and Kalman filters that is applied in different parts of the process.

10.1 Models

We have explored the linear models 'constant speed' and 'constant acceleration'. Details on these linear models can be found in e.g. [34], and is also summarized shortly in appendix H (since the reference [34] may be difficult to get hold of).

We have also explored non-linear models. Details on the bicycle model can be found in appendix I, and details on the simplified bicycle model can be found in appendix J.

It is still unclear which model that is most suited in the initial tracking step. A complex model may become unstable since we only have position (and possibly orientation) as measurement, and it takes longer time for these models to recover from noisy measurements (e.g. a bicycle model have to turn in a physically realistic manner to get back to the true position, while a linear model can simply 'slide back' to the position). Still, we currently use a bicycle model in the coarse (first round) tracking, but some of the errors (e.g. the performance in occlusion situations and instability in the beginning of a track) may be traced back to the use of this model. Initially, a simpler model was used in the coarse (first-round) tracking, but in order to get more realistic vehicle trajectories before the following steps had been implemented this model was chosen and has been kept.

More complex models can (and should) however be applied as a post-processing step to further refine the trajectories. Section 12 describes this step.

Note that in reality the direction of the speed does not have to be the same as the orientation of the vehicle. The difference is neglected in the bicycle models, and may be a topic for future research.

10.2 Causal and non-causal Kalman filters

In the coarse (first round) tracking, we only have access to measurements from previous points in time. In this case we use a causal Kalman filter for prediction of the vehicle state in the next frame (which is used as initialization to the box optimization). For the non-linear models, an extended Kalman filter (EKF) have to be used, see e.g. [33] for more details.

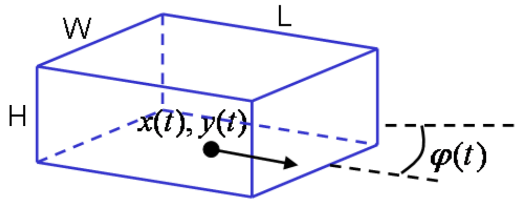
When the coarse tracking is finished, we have access to measurements from all times. It is then possible to apply a non-causal Kalman filter ('glättning' in swedish). This version is less common, and we have only implemented the linear models, see e.g. [20] for details. Hence, to utilize the benefits from both the non-linear models and the non-causal filters, a combination of filters have been used in the post-processing (see section 12 for details).

11 Coarse (first-round, causal) tracking

This section contains a description of the coarse tracking algorithm. By 'coarse' we mean that the tracking is applied in low resolution, and that the prediction is causal, i.e. only based on previous frames. Many of the components have been described in the other sections, and we will refer to them for more details when needed.

In essence, the tracking system computes 3D box state estimates in the current frame, which are used as measurements to an extended Kalman filter (EKF) in the 3D ground plane domain. The prediction from the Kalman filter and previous trajectory data is then used as initialization to estimate the 3D box state in the next frame. Hence, the tracking is performed in the 3D domain, and the measurements are also in 3D, even though they are computed using 2D images. Figures 33-34 may give a rough idea.

The details can be worked out in many different ways, we propose some solutions here. Let $\mathcal{B} = \{L, W, H, x, y, \varphi\}$ denote the state of a 3D box on a ground plane in 3D, where $(L, W, H) = (Length, Width, Height)$, (x, y) is the position on the ground plane, and φ is the orientation.



We will describe the system in terms of a box 'life cycle':

1. **Initiation:** Find foreground regions that is not yet covered by a box (after updating current boxes). Initiate a Kalman filter for each new box. Section 11.1.

2. For each new frame:

    (a) **Predict the box position and orientation** from previous estimates. Section 11.2.

    (b) **Get a measurement of box position and size** using the box optimization procedure and the prediction as initialization. The core procedure is described in section 9, but section 11.3 describes the surrounding details.

    (c) **Update the Kalman filter** using the box position and orientation estimate if the measurement is considered valid, otherwise use Kalman 'blind prediction'.

    (d) **Update the size estimate** using all measurements in time. Section 11.4.

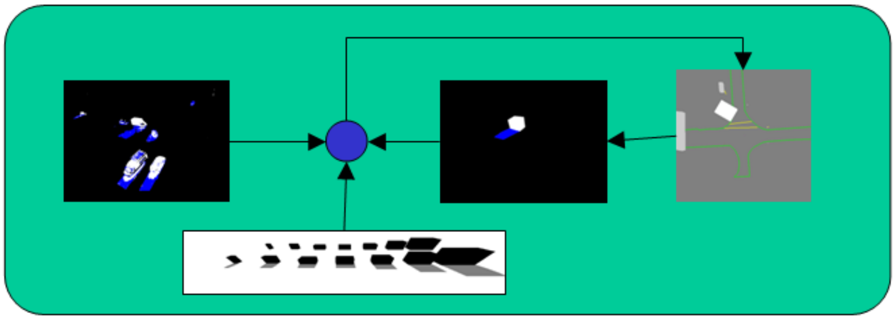


Figure 33: Illustration of the coarse tracking system.

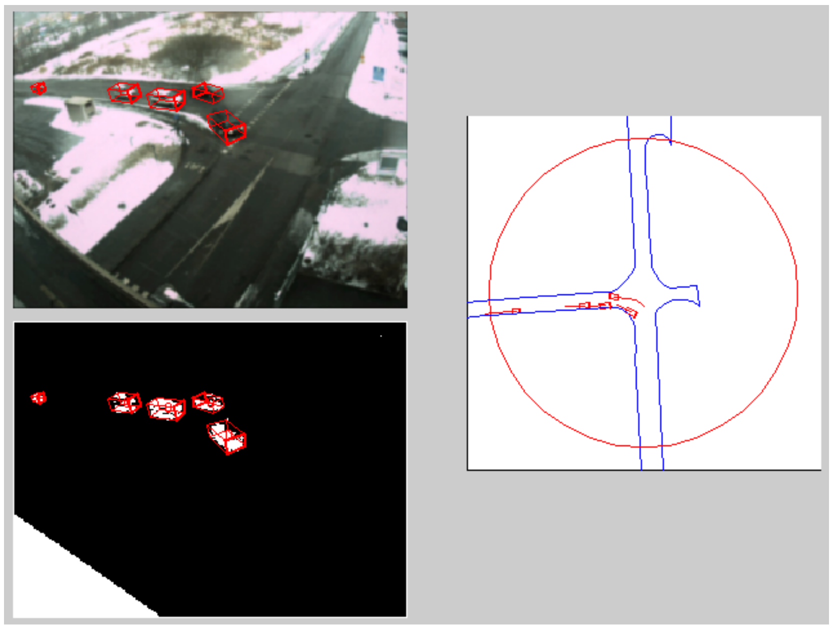


Figure 34: Example of the output from the coarse tracking system.

3. **Terminate** if the box is no longer in the image, or if the box is no longer covering a foreground region.

The output from each frame is:

- A 'raw' measurement of from the box optimization, which will be used in the post-processing (section 12).
- The Kalman state that will be used for prediction in the next frame. (The Kalman state was also used as final output before the post-processing and the refined tracking was implemented.)

After termination of the tracking we will also have a size estimate based on the estimates from all frames.

11.1 Initiating new boxes

Deciding when to initiate new boxes can be tricky, especially in heavy traffic. In our system, new boxes are currently only allowed to be initiated if the segment does not overlap much with existing boxes. This means that new vehicles must be almost fully visible to be initiated.

1. Find regions in the image that is not overlapping with existing boxes, and that is sufficiently large (using some ad-hoc measure that takes into account the 3D geometry which gives different size thresholds in different parts of the image).
2. Optimize the 3D box position and size to the region (see section 9). The optimization is made without using an occlusion mask for the existing boxes (which makes it easier to detect if the new box belongs to an old one). The lane orientation is used as initial orientation.
3. Keep the new box if considered valid. The validation criteria can for example be:
    - Sufficiently large in the image
    - Enough foreground pixels inside the box
    - The initial seed should still be inside the box
    - Not too much overlap with existing or other new boxes.

11.2 Prediction of the 3D box state

One component of the tracking involves predicting the state (position, orientation, size) of the 3D boxes to the next frame. The prediction is used as initialization to the box optimization, and is computed as follows:

- The box position is predicted from the Kalman filter. We use a bicycle model in an extended Kalman filter (see 10 and appendix I).

- The box orientation is predicted from the trajectory of previously collected estimates, by computing the tangent at the end of the trajectory in a certain spatial window (initially using the lane orientation). It has seemed more stable to use a spatially based window rather than a time based window like the orientation in the Kalman model. For vehicles that are temporarily stationary over extended periods the latter estimate may become unstable. In a recent paper [3] (which is an improved version of [2]), propose another model including curvature of the motion path which reduces the stop-and-go problem. But even if we use an exact vehicle model, noisy position measurements can still in theory make the car appear to move forward and backward and eventually turn around. It may also be possible to use optical flow as a hint for the orientation, as in e.g. [29], but this only works for moving vehicles.

11.3 Estimation of 3D box position and size

Once we have a prediction of the box state (section 11.2), we can initialize the box optimization. The procedure is as follows:

1. **Compute an occlusion mask** (the valid mask $\mathcal{V}$ in section 9). The mask marks out regions where the vehicle is likely to be occluded by other boxes and buildings. The mask is made from manually labeled building regions etc., and from image regions that overlap with predicted current boxes.

   Moreover, in cases where shadows are largely misclassified as foreground, it has appeared to be more robust to also include the regions that fall within simulated shadows of other boxes as invalid pixels. This means that the tracker will treat a vehicle that falls within (simulated) shadows from other vehicles as occluded. Unfortunately, since we do not know in which cases the shadow detection fails, this criteria is always used. The performance is especially reduced in situations where large vehicles and long shadows are present.

2. **Optimize a box** (section 9). When a vehicle becomes partly occluded (as can be detected by computing the overlap between the occlusion mask $\mathcal{V}$ and the image of the predicted box), it has appeared more stable to lock the size to the currently accumulated estimate (see section 11.4) during optimization. Hence, we use algorithm 4 (optimize both position and size) if the box is considered to be fully visible, and algorithm 3 (optimize position only) otherwise. The occlusion mask $\mathcal{V}$ is used in both cases.

3. **Decide whether the estimated box is valid or should be treated as an outlier**. The 3D box estimates are sometimes corrupted and should then be treated as outliers. Various outlier tests can be applied such as thresholds for

   - maximally allowed occlusion (both in 2D and 3D)

- lower limit for pixel area
- maximal foreground sparsity inside a box
- maximally allowed distance from the prediction

These tests may depend on the situation. For example, we have collected data from an intersection where flag poles on the side of the road cast moving shadows onto the road, which sometimes initiate new boxes. A 3D test for overlapping boxes would not allow these boxes to be 'run over' by true boxes.

11.4 Averaging size estimates

The vehicle size is computed by weighting together the estimates from each frame in time. Each estimate of L, W, and H is weighted depending on the view from which the vehicle was seen. For example, if the vehicle is viewed from above, the height estimate gets a low weight and the length and width get higher weights. The weight also depends on the size in pixels of each measurement. The exact formula for the weight is probably not critical.

11.5 Multiple cameras

As also mentioned in the box optimization section 9, we have also made experiments with merging of estimates from up to four cameras with partly overlapping views of an intersection (see e.g. figure 14 on page 26).

Initially we tried to optimize the 3D box to all views simultaneously, by adding the similarity measures from each camera. However, this requires a very accurate time synchronization between the cameras. Otherwise the optimized box usually becomes larger than the actual vehicle, since the box tries to cover all camera blobs (foreground is rewarded more than background is punished). If the cameras are not sufficiently synchronized, it appears to be better to optimize to each camera separately and then merge the resulting 3D boxes by e.g. averaging the state parameters (this may be true even if the cameras were synchronized).

The initial seeds for new boxes are also made separately from each camera, but the final selection of new boxes is made based on all cameras.

11.6 Discussion

The tracking performance varies depending on conditions. Figures 35 shows some results. Figure 36-36 shows some results where the tracking have failed. Better performance is still desired in situations with large occlusions and heavy traffic. This is a well known problem and currently unsolved for the general case.

However, the performance in moderate occlusions should, and probably could, be improved. Figure 37 shows an example where two vehicles meet, one of the vehicles (1173) are lost, and the other (1172) vehicle 'jumps' over to the lost one. The lost vehicle then have to be almost fully visible before a new box can be initiated. This behavior is due to some insufficient property of the Kalman filter and outlier testing (which was not investigated further due to lack of time). For example, initially the occlusion masks were created from images of the predicted boxes rather than the actual segments in the image, which gave some unpredictable behavior.

Some of these errors can be repaired afterwards by the automatic and manual repair. The automatic repair compares measured trajectories with a set of principal trajectories, see figure 38 for examples and [24] for more details.

There are also some problems with entering vehicles. The estimated box size increases as more of the vehicle becomes visible, which results in an incorrect estimate of the center position (which may even move backwards). Estimating the position of the front of the box may improve the behavior. It would be even better if the tracking could start once the entire vehicle is inside the image, but this will reduce the coverage considerably. The chosen bicycle model in the Kalman filter needs a few samples before the prediction can be reliable, and the same goes for the spatial window for prediction of the orientation. Therefore, the orientation is initially locked to the lane direction for the first few samples. Even if the orientation is incorrect, this approach seems to give a more robust behavior than to try to predict the true orientation.

Moreover, the coarse tracking is using low resolution images, mainly to increase the speed of the processing. This means that many small objects, such as pedestrians, will be too small to initiate a box. The large coverage demand also made it necessary to track vehicles that are only a few pixels large. This will reduce the size estimation accuracy, and even issues like how to downsample images becomes important (we found it better to exclude the usual lowpass filtering before downsampling, to avoid smearing of small regions).

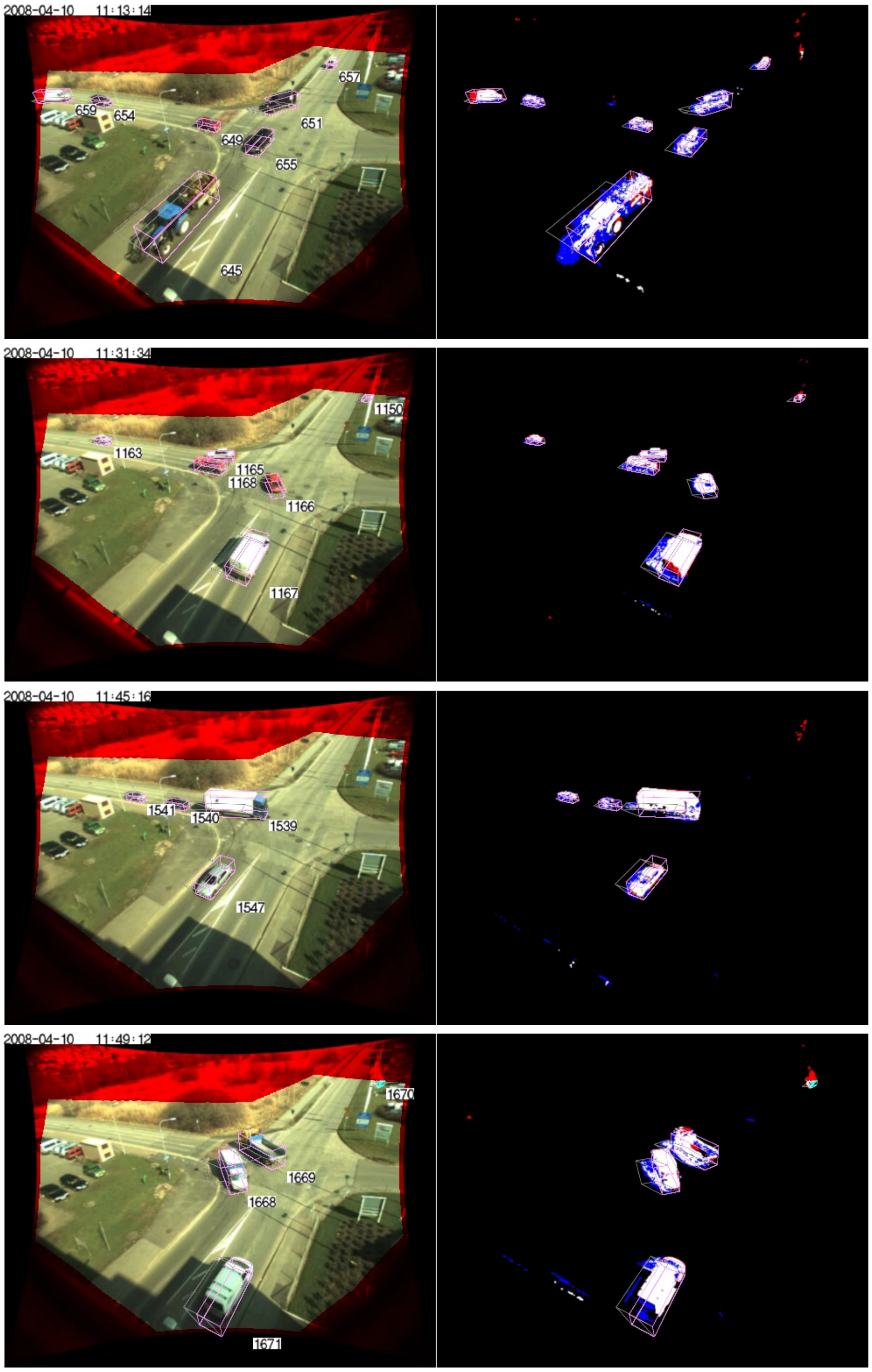


Figure 35: Examples of tracking results.

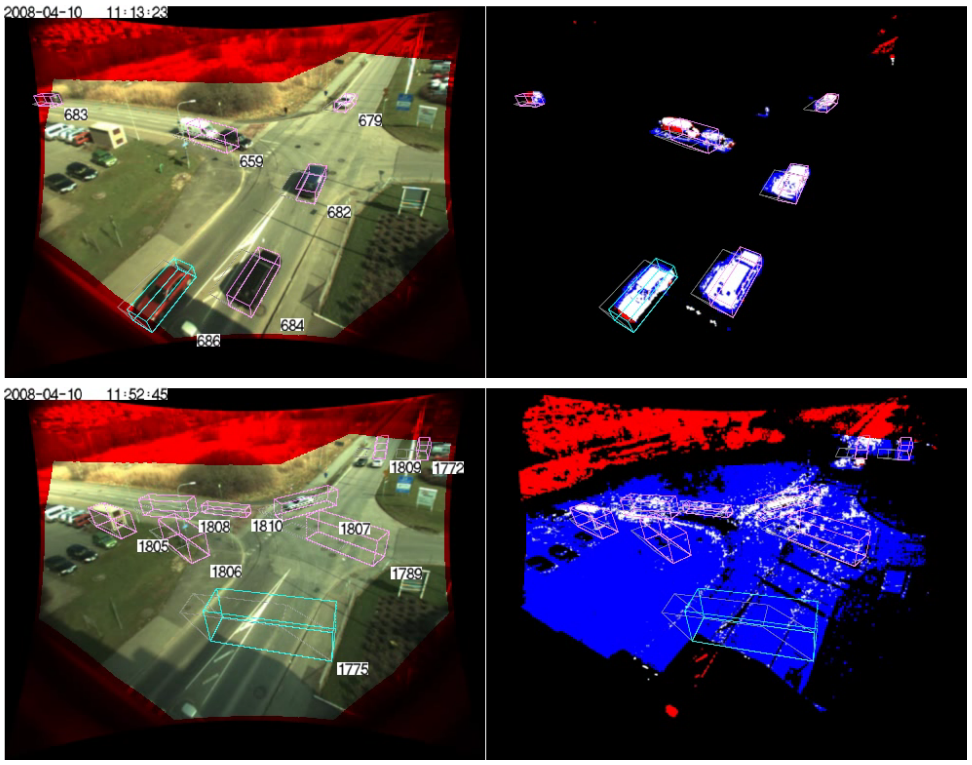


Figure 36: Examples of tracking errors. Top: Two vehicles are merged into a truck. Bottom: False vehicles are generated due to moving clouds.

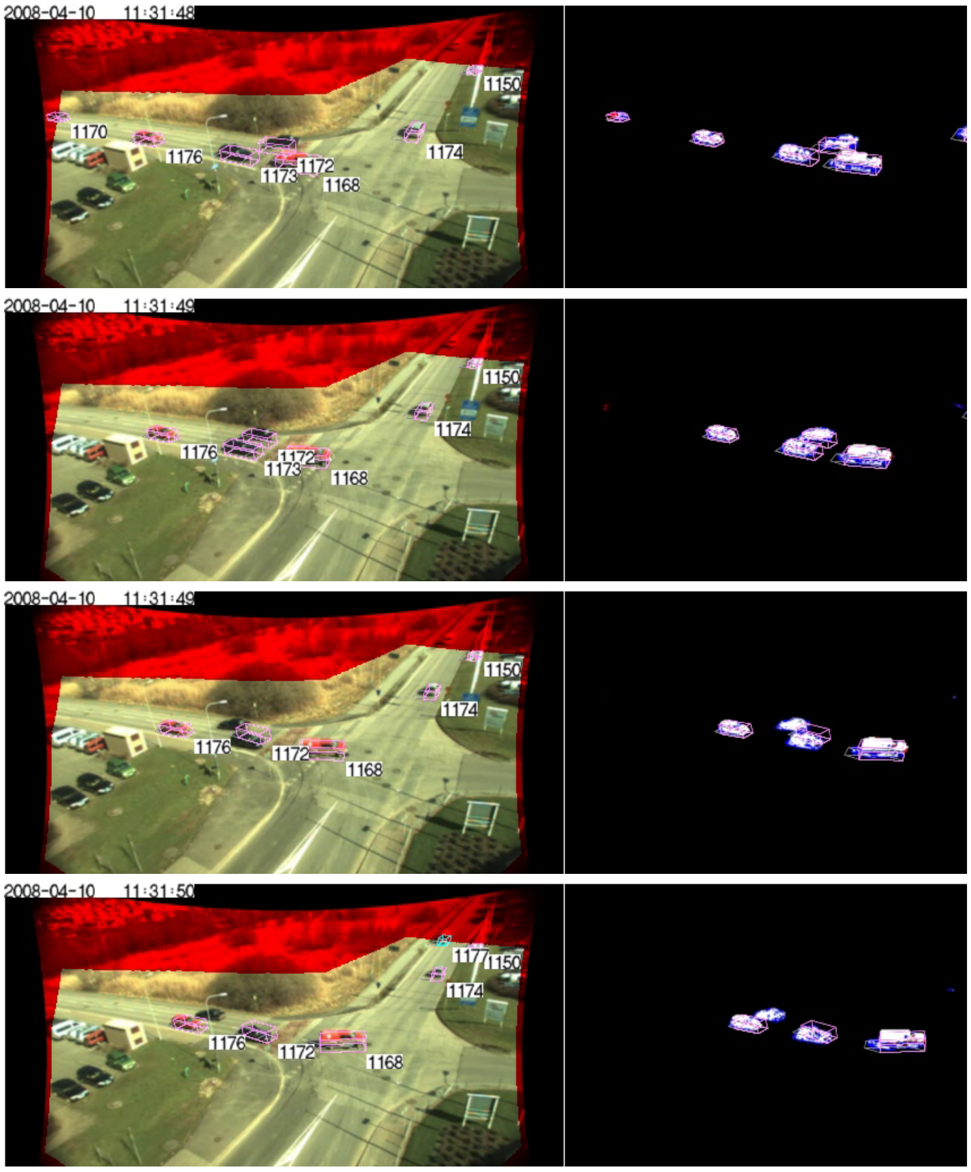
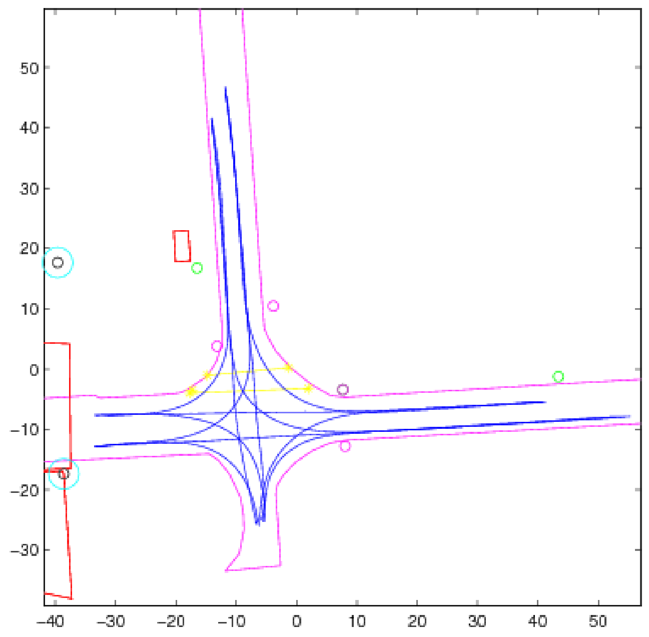


Figure 37: Examples of 'jump' error.

Renova intersection



Jung intersection

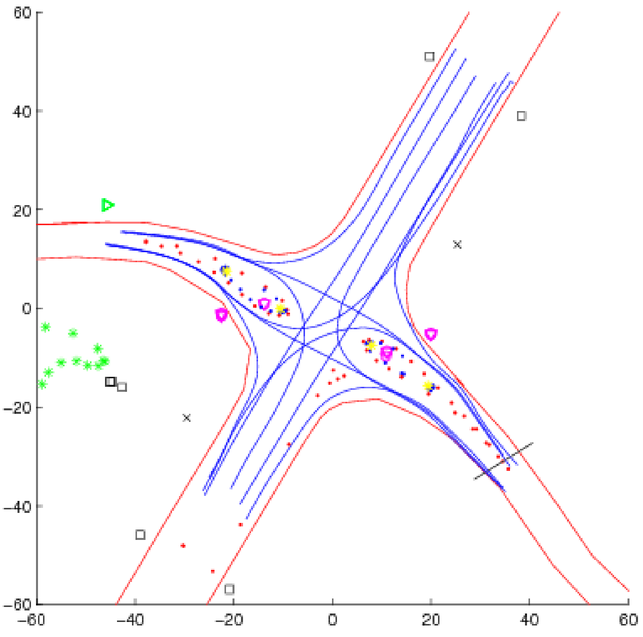


Figure 38: Examples of principal trajectories in the automatic repair.

12 Post-processing

The coarse tracking step and the refine tracking step are both followed by some post-processing on the trajectory data, to further improve the estimates of the position, orientation, speed, etc.

After the coarse tracking, we have measurements from all times and it is suitable to have a non-causal post-filtering to make use of both past and future measurements. As discussed in the Kalman filter section 10, we have only implemented the non-causal Kalman filter for the linear case. Therefore, to utilize the benefits from both the non-linear models and the non-causal filters, a combination of filters have been used in the post-processing. For some more details on the models and Kalman filters, see section 10.

1. First, a linear constant speed model is applied in a non-causal Kalman filter to improve the position measurements. The measurements are also weighted depending on the confidence, i.e. depending on occlusions, outlier tests etc. Hence, in regions where the vehicle has been occluded the positions will be interpolated from neighboring, visible, regions.

2. Second, the orientation in each position is refined by estimating the orientation of the position trajectory in a symmetric (non-causal) Gaussian weighted spatial window (e.g. a few meters wide). Basically, a local covariance matrix is estimated, and the orientation is computed from the dominant eigenvector.

   It has appeared more robust to estimate the orientation from a spatial window instead of a time based window (for example from a non-linear Kalman filter), especially when the vehicles are temporarily stationary for a longer period of time.

3. Finally (or in parallel to the second step), a nonlinear simplified bicycle model is applied in a causal Kalman filter, to further refine the position and speed estimates (e.g. preventing vehicles from 'sliding' laterally). The acceleration is computed by just simply differentiate the speed.

It should be noted, that it may not always be better to use these refined estimates for analysis. The original estimates may appear more noisy when plotted or visualized in a video sequence, but could be closer to the truth if the post-processing models are not physically correct.

13 Refined (second-round) tracking

Basically, the refined (second-round) tracking improves the estimated positions of existing boxes using the high resolution images. The differences to the coarse tracking are the following:

- The tracking is performed in the highest resolution.
- The box size is locked at all times, using the accumulated size estimate from the coarse tracking (section 11.4).
- The predictions are not made from previous frames, i.e. is not causal, as in the coarse tracker. Instead, the box states from the post-processing in section 12 is used, which smooths the coarse tracking measurements (possibly followed by automatic and manual repairs) in a non-causal manner.
- There is no initiation of new boxes, and no extrapolation of existing trajectories.

The refined tracking is followed by the same post-processing as the coarse tracking (section 11.4).

The refined tracking described here was initially planned to include refinement of the box size (but was not implemented due to lack of time). An improved size estimate would in turn would improve the position estimates even further. Currently, the box size is estimated in low resolution images in the coarse tracking step. The estimate is based on a number of estimates from different views, and a higher weight is given to estimates where the vehicle is close to the camera (is large in the image), but the estimate would still be improved using high resolution images and for example using a few well chosen views.

14 Computational issues

Most of the scripts surrounding the image processing have been written in Python (and some in bash). This includes video management scripts for putting new data at right places, making web-copies, gathering of data for processing, and automatization of image processing.

Section 2 and figure 5 on page 9 described the image processing procedures. Most of the image processing is computed on a Linux cluster at the Swedish National Supercomputer Center (NSC), except the camera ego-motion compensation on GPU which is processed on two dedicated windows-machines. The rectification, background subtraction and foreground classification, and the box optimization is written in C/C++. The remaining parts are written in Matlab and Mex. Each component in the overview figure 5 results in either a video .avi-file or a Matlab .mat-file, which will be input to the subsequent step. It would be more computationally effective to make a combined implementation of the components, but it would also be more difficult to develop and test individual components.

The main bottlenecks in the implementation are the coarse tracking and the refine tracking, due to the box optimization, Matlab functions, and large images. The speed depends on the traffic density. For example, the coarse tracking with two vehicles takes about 1.5 seconds/frame (on a standard machine), with the following partitions:

20%	Box optimization (C-code)
20%	Generate images of 3D boxes (C-code) to create image masks for use in various steps
20%	Visualization (plotting not included) and debugging variables, i.e. reading rectified images, downsampling, making vehicle templates (generating 3D box images)
20%	Matlab operations on large images (e.g. find, nnz, bwmorph, ...)
15%	Read classified images and convert to grayscale representation
5%	Remaining Matlab operations

A few more simple optimizations can be made, e.g. removing creation of debugging variables. Further improvements require more work, e.g. modifying the algorithm or code for the box optimization, and changing entirely from Matlab to C. For example, just converting the classified images from color representation to grayvalued representation takes about 10% of the time (which only involves a few operations for thresholding, multiplication, and conversion to uint8).

A Video management and processing routines

The management of collected video data has required some resources. This includes setting up the file structure, making low resolution copies for the web, gather selected data for processing, and scripts to make the image processing more automatic. Below follows an overview of the process from taking care of incoming data, to putting the resulting vehicle trajectories from the image processing on the web.

Most of the scripts for managing the video data have been written in Python.

IVSS Video data processing instructions

To access the scripts, run `setenv PATH $PATH':/proj/bb/ivss0/video_management'`
The scripts are preferably executed on beauty.

Incoming data

- Move new video data to (X is the next available storage)
  `/proj/bb/ivssX/data/DUMP`
- Text files for manually repaired data may have been uploaded to
  `/proj/bb/ivss0/web/upload/files` . Move these to
  `/proj/bb/ivss0/web/data/ProcessedData/data/manual_repair_data/`
  (Check that the file names looks consistent with the existing files...)

Putting new video at the right places

- There may be some days/libraries that contain incomplete (broken or missing) data. Move these to
  `/proj/bb/ivssX/data/DUMP/Something_is_wrong`
  The incomplete data are found semi-manually as:
  - Days that contain very few video files
  - Data from Jung should contain four cameras (NIC0_CAM0, NIC0_CAM1, NIC1_CAM0, NIC1_CAM1),
    data from Renova should at least contain the camera NIC1_CAM0 (this is the only camera that is currently processed). Run e.g.
    ```
    ivss_DUMP_find_incomplete.py -k Keyname /proj/bb/ivssX/data/DUMP/200611*
    ```
  - If `tcprobe` cannot read a file, typically the last file of the last day. Run e.g.
    ```
    tcprobe_multifile.py /proj/bb/ivssX/data/DUMP/200611*
    ```
- Run `ivss_DUMP2video.sh` that moves and renames the files to the video file tree, e.g.
  ```
  ivss_DUMP2video.sh -k Keyname /proj/bb/ivss1/data/DUMP/200611*
  ```
  Be sure to select the correct Keyname (Renova or Jung)!
- Once in a while, tell the TUS group to make backup of the new data.

Making web data

- Run `ivss_video2webvideo_loop.py` to compute web video and some other stuff from the original data, e.g.
  ```
  ivss_video2webvideo_loop.py -k Jung /proj/bb/ivss1/data/DUMP/Jung/200803*
  ```
- For a time graph overview of all available data, run the following script (at /proj/bb/ivss0/video_management/) in Matlab to generate overview images:
  ```
  >> mk_overview.m
  ```
  The result is two images, `Renova_videos.png` and `Jung_videos.png`.
- Remove the empty directories under `/proj/bb/ivssX/data/DUMP` , to indicate that the management of this new data now is complete.

Outgoing data

- Send the data disks back to
  `Erik Petre', Autoliv Development AB, 447 93 Vårgårda`

Image processing on video data

- Manually look through the data on the web and make a file with selected good days and times:
  - Avoid rainy days and wet asphalt (no 'mirror cars')
  - Avoid days with too much change in sun/clouds

  You also have to choose the desired types of processing. See NSC_template.txt (and other files in the same directory) for an example. We will below denote this file `NSC_YYYYMMDD.txt`. (The files Renova_notes.txt and Jung_notes.txt contain summaries of the weather for all collected data, it may be useful to update these files, to have an overview. These files also contain info on which days the test-vehicle has been driving.)
- Run `ivss_video2limitedvideo_loop.py` to extract data for selected times and create softlinks, e.g.
  ```
  ivss_video2limitedvideo_loop.py -s logfiles/NSC_YYYYMMDD.txt -o
  /proj/bb/ivss3/partdata
  ```
  You have to know in which `ivssX` the original data is stored.
- In case the data comes from Jung: On the dedicated Windows-machines, run `ivss_CamegoScript.py` to compensate for camera egomotion, camera flicker, and to rectify the images:
  - To make use of both machines, split the file `NSC_YYYYMMDD.txt` into two parts, e.g. `NSC_YYYYMMDD_a.txt` and `NSC_YYYYMMDD_b.txt`. Place one file on each machine under
    `C:\Bjorn\Subversion\RectEgomotion`
  - Open a cmd window, go to `C:\Bjorn\Subversion\RectEgomotion`
  - Run
    ```
    ivss_CamegoScript.py -s NSC_YYYYMMDD_x.txt -i \\discworld\bb\ivss3\partdata -e
    SteadCamWinTest.exe
    ```
- Now all data should be prepared for the NSC run. Run `ivss_makeNSCsoftlinktree.py` to make a softlink tree of data that is to be copied to NSC, e.g.
  ```
  ivss_makeNSCsoftlinktree.py -s logfiles/NSC_YYYYMMDD.txt -o
  /proj/bb/ivss3/softlink_NSCYYYYMMDD
  ```
- Sync the data to NSC and make the processing.
  - Run the rsync script: `sync2NSC softlink_NSCYYYYMMDD`
  - Create batch job scripts at NSC server:
    `generate_batch_jobs.py -s softlink_NSCYYYYMMDD -i /lcgstorage2/x_johwi/partdata`
  - Put the start jobs on the batch queue: `./startup.sh`
  - Wait until all jobs are finished. There will be a lot of emails sent containing job status from the batch queue system.
  - Run the rsync script: `sync2ISY softlink_NSCYYYYMMDD`
- Sync back the video files to ISY to

`/proj/bb/ivss3/partdata`

Sync back the mat-files to ISY to

`/proj/bb/ivss3/softlink_NSCYYYYMMDD`

- Move the resulting mat-files to the web at

  `/proj/bb/ivss0/web/data/ProcessedData/data/`

  There is not yet any automatic scripts for this...

B PID controller for adaptive shutter control

This section describe how to use a PID controller from the mean image intensity to the shutter time. Figure 39 shows a general feedback system.

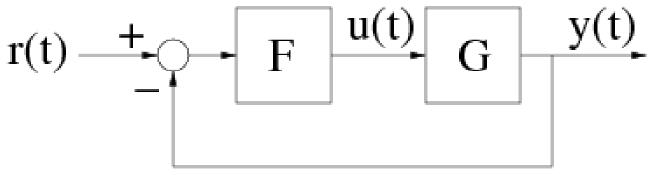


Figure 39: Sketch of a controller.

Variable definitions:

Variable	General description	In our case
$r(t)$	Reference signal	Desired mean intensity (constant in time)
$u(t)$	Control signal	Shutter time
$y(t)$	Measured signal	Current mean intensity
$F(s)$	Controller transfer function	PID controller
$G(s)$	System transfer function	Camera model for the mapping between shutter time and mean image intensity

The basic PID controller is in continuous time written as

$$u(t) = K\left(e(t) + \frac{1}{T_I}\int_{t_0}^{t} e(\tau)d\tau + T_D\frac{de(t)}{dt}\right), \tag{5}$$

where $e(t) = r - y(t)$, and K, T_I, and T_D is the user parameters. The user parameters can for example be chosen using the Ziegler-Nichols ad-hoc rule described in [12]. In practice (discrete time) we compute the control signal according to [12]:

$$\tilde{u}_n = K\left(e_n + S_n + D_n\right) \quad , \quad \begin{cases} e_n = r - y_n \\ S_n = S_{n-1} + \frac{T}{T_I}e_n \\ D_n = T_D\frac{e_n - e_{n-1}}{T} \end{cases} \tag{6}$$

$$\begin{array}{rl} \tilde{u}_n < u_{min}: & u_n = u_{min},\ S_n = u_{min}/K \\ u_{max} < \tilde{u}_n < u_{max}: & u_n = \tilde{u}_n \\ u_{max} < \tilde{u}_n: & u_n = u_{max},\ S_n = u_{max}/K \end{array}$$

where T is the sampling time.

Note that the integral variable S_n is limited in the case of the control signal being too high or too low. This gives a better behavior of the controller.

The PID controller was simulated in the test phase using the very simple system transfer function $G(s) = \frac{1}{1+s}$. This function may not be physically correct in our case, but used only for implementation and bug control purposes. We can convert G to discrete form using e.g. Tustins formula $s \approx \frac{2}{T}\frac{1-q^{-1}}{1+q^{-1}}$, where q is the shift operator, e.g. $qu_n = un + 1$. The system transfer function becomes $G(q) = \frac{T(1+q^{-1})}{(T+2)+(T-2)*q^{-1}}$, i.e. $y_n = -\frac{T-2}{T+2}y_{n-1} + \frac{T}{T+2}(u_n + u_{n+1})$.

Another useful tool for implementation and bug checking is `tf.m`, `step.m`, and `feedback.m` in MATLAB. These functions can be used to simulate a continuous transfer function, which can be compared with your own discrete implementation to make sure that your implementation is correct. For example

```
>> sys = tf(1,[1 1]);
>> step(sys);
```

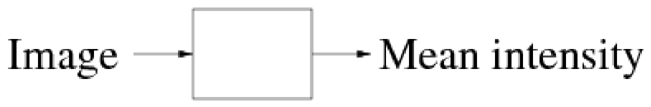
computed a step response for the function $G(s)$ above. This step response can be compared with the Y vector from the following script:

```
>> T = 0.01;
>> Y = 0;
>> for t=0:0.01:6
>>   if t==0
>>     y = (-(T-2)*Y(end) + T*1)/(T+2);
>>   else
>>     y = (-(T-2)*Y(end) + T*2)/(T+2);
>>   end
>>   Y = [Y y];
>> end
```

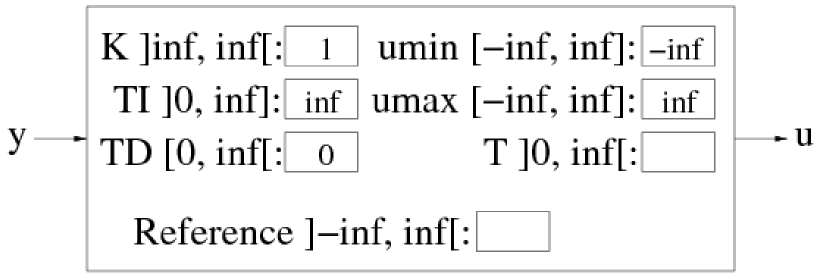
The closed system transfer function $y = \frac{GF}{1+GF}r$ can be generated using the function `feedback.m`, and simulated in the same manner as above.

B.1 DirectShow software specification

Two filters for adaptive control of the camera shutter time were implemented in DirectShow, as specified below.

Mean image intensity	
Input:	Color RGB image, gray-valued image, or Bayer image.
Output:	Mean (average) image intensity (real valued).
User parameters:	None
Description:	The mean intensity is for all image formats defined as the sum of all values divided by the number of values. Examples: • RGB image, stored in a $M \times N \times 3$ array: Mean $= \frac{1}{N*M*3} \sum_{m=1}^{M} \sum_{n=1}^{N} \sum_{i=1}^{3} I(m,n,i)$ • Gray-valued image or Bayer image, stored in a $M \times N$ array: Mean $= \frac{1}{N*M} \sum_{m=1}^{M} \sum_{n=1}^{N} I(m,n)$

PID controller



K]inf, inf[: 1   umin [–inf, inf]: –inf
TI]0, inf]: inf   umax [–inf, inf]: inf
TD [0, inf[: 0   T]0, inf[:
Reference]–inf, inf[:

Input: Measured value, real valued `y` (e.g. mean image intensity)

Output: Control value, real valued `u` (e.g. shutter time)

User parameters:

Symbol	Name	Range	Default
`K`	Gain	$]\infty,\infty[$	1
`TI`	Integral coeff	$]0,\infty]$	∞
`TD`	Derivative coeff	$[0,\infty[$	0
`umin`	Min output value	$[-\infty,\infty]$	$-\infty$
`umax`	Max output value	$[-\infty,\infty]$	∞
`T`	Sampling time ($= \frac{1}{\text{sampling rate}}$)	$]0,\infty[$	—
`r`	Reference value	$]-\infty,\infty[$	—

Description: Implementation of the function

$$u(t) = K\left(e(t) + \frac{1}{T_I}\int_{t_0}^{t} e(\tau)d\tau + T_D\frac{de(t)}{dt}\right)\ , \quad e(t) = r - y(t)$$

All variables can be real valued (i.e. not only integers).

Pseudo code:

```
% Initialisation
e0 = 0
S = 0
D = 0

% Function call, new measurement y
%-----------------------------------
e = r-y
S = S+T/TI*e
D = TD/T*(e-e0)
u = K*(e+S+D)
if u<umin
  u = umin
  S = umin/K
elseif u>umax
  u = umax
  S = umax/K
end

% Update
e0 = e;
%-----------------------------------
```

Note that `e0` and `S` need to be stored for the next call.

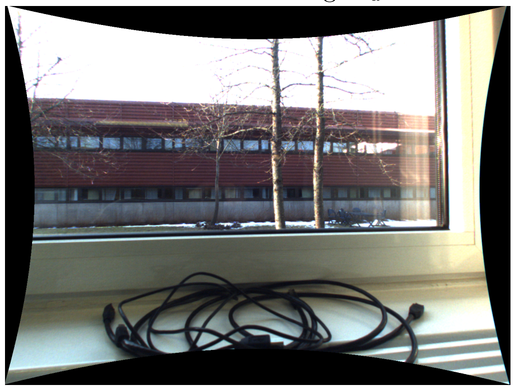
C Short description of the radial tangent lens distortion model

C.1 Definitions and notations

Distorted image I_d

$\mathbf{x}_d$ = point in distorted image

Undistorted image I_u



$\mathbf{x}_u$ = point in undistorted image

Lens distortion deals with the mapping between distorted images and undistorted images.

Fisheye lens (wide angle lens): Lens with extreme distortion, FOV (Field Of View) $>\approx 100°$.

C.2 Radial tangent lens distortion model

From [10]:

$$\begin{aligned} r_d &= \frac{1}{\omega} \arctan\left(2r_u \tan\frac{\omega}{2}\right) , \\ r_u &= \frac{\tan(r_d\omega)}{2\tan\frac{\omega}{2}} . \end{aligned} \tag{7}$$

This model corresponds to an ideal fisheye lens (but also seems to work well for more regular lenses, according to [7]?). The parameter ω corresponds to the field of view (FOV) angle.

However, this model does not fulfill the requirement that the metric in the center should be preserved, i.e.

$$\left.\frac{dr_u}{dr_d}\right|_{r_d=0} = 1 , \tag{8}$$

Of course we can always rescale r_u in (7) after the mapping from r_d, to accommodate to the metric-preserving demand (8). But this is equivalent to using the simpler model

$$\begin{aligned} r_d &= \frac{\arctan(r_u \omega)}{\omega}\,, \qquad (9) \\ r_u &= \frac{\tan(r_d \omega)}{\omega}\,. \end{aligned}$$

This simpler model also has the advantage that we do not have to normalize r_d such that $0 < r_d < 1/2$ which is necessary in (7). But note that ω now has the unit radians/pixel.

C.2.1 Origin and aspect ratio

Same as [10], we also include parameters for the origin and aspect ratio according to

$$x_d = s_x x'_d + c_x \qquad (10)$$

$$y_d = y'_d + c_y\,. \qquad (11)$$

where $r_d = \sqrt{x'^2_d + y'^2_d}$.

Furthermore, it is sometimes necessary to include a homography if the calibration picture was not taken orthogonal to the camera. This is not required in the estimation method used here.

C.2.2 Comments

- The model is estimated using calibration images with straight line segments. The line segments should be straight in the undistorted image. In this way we do not have to deal with homographies.
- It may seem simpler to compute the error in the undistorted domain. But the lens-parameters may rescale the undistorted domain, which can affect the error (the optimization may decide to rescale instead of improving the straightness).

D Theory for the projective geometry

D.1 Definitions and notations

It is assumed that we have a camera, a 3D world, and a 3D world ground plane, see figure 40.

$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$	2D image coordinates (we choose this notation instead of (x, y) to conform with Matlabs matrix notation)
$\mathbf{X} = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$	3D coordinates
$\mathbf{Y} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}$	Point $\mathbf{X}$ described in the 3D world ground plane coordinate system
$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}$	2D coordinates in the 3D world ground plane
$\tilde{\mathbf{x}}, \tilde{\mathbf{y}}, \tilde{\mathbf{X}}$	Homogeneous coordinates, i.e. $\tilde{\mathbf{x}} = \begin{pmatrix} \mathbf{x} \\ 1 \end{pmatrix}$, $\tilde{\mathbf{y}} = \begin{pmatrix} \mathbf{y} \\ 1 \end{pmatrix}$, $\tilde{\mathbf{X}} = \begin{pmatrix} \mathbf{X} \\ 1 \end{pmatrix}$
Projection matrix, $\mathbf{P}$	Also known as camera matrix. Mapping between 2D image coordinates, $\mathbf{x}$, and 3D world coordinates, $\mathbf{X}$, $\tilde{\mathbf{x}} = \begin{pmatrix} \mathbf{x} \\ 1 \end{pmatrix} \sim \mathbf{P} \begin{pmatrix} \mathbf{X} \\ 1 \end{pmatrix} = \mathbf{P}\tilde{\mathbf{X}}$
Homography, $\mathbf{H}$	Mapping between points in two planes, $\tilde{x} = \begin{pmatrix} \mathbf{x} \\ 1 \end{pmatrix} \sim \mathbf{H} \begin{pmatrix} \mathbf{y} \\ 1 \end{pmatrix} = \mathbf{H}\tilde{\mathbf{y}}$
Camera matrix, $\mathbf{K}$	Contains camera intrinsic parameters, generally $$\mathbf{K} = \begin{pmatrix} \gamma f & sf & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{pmatrix} \qquad (12)$$ where f = focal length, $\mathbf{c} = (c_1, c_2)$ = principal point (origin), s = skew, γ = aspect ratio. We will assume $s = 0$ and $\gamma = 1$ here.
Plane parameters, $\mathbf{V}$	Representation of the 3D ground plane, $V_1X + V_2Y + V_3Z + V_4 = 0$
Transformation, $\mathbf{T}$	Transformation between original 3D coordinate system and ground plane coordinate system, $\tilde{\mathbf{Y}} = \mathbf{T}\tilde{\mathbf{X}}$.

We assume here that the lens distortion has been removed, i.e. that all image coordinates belong to the undistorted domain ($\mathbf{x} = \mathbf{x}_u$).

This section is based on the references [14, 11, 32].

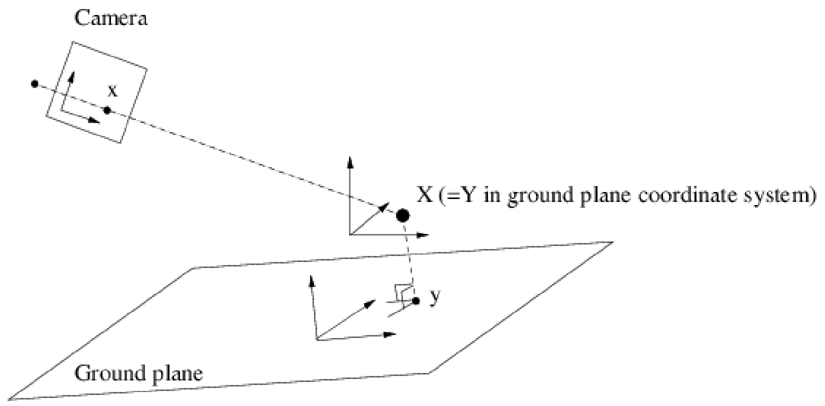


Figure 40: The camera, the 3D world, and the ground plane. Each of these three have a coordinate systems.

D.2 3D ground plane

D.3 Plane estimation

The 3D plane is here assumed to be fairly parallel to the X, Y-plane. We can therefore estimate the plane from the least squares problem

$$\min \sum_k (Z_k - a - bX_k - cY_k)^2 \,. \tag{13}$$

The plane vector becomes $\mathbf{V} = (b\ , c\ , -1\ , a)$.

D.4 3D $\leftrightarrow$ plane coordinate transformation

It is useful to have a mapping between 3D points $\mathbf{X}$ to plane coordinates (y_1, y_2). For this we define the plane coordinate system as

$$\{\hat{\mathbf{v}}_1, \hat{\mathbf{v}}_2, \hat{\mathbf{n}}\} \quad \text{where} \quad \begin{array}{l} \hat{\mathbf{v}}_1 = \begin{pmatrix} 0 & V_3 & -V_2 \end{pmatrix}^T / \sqrt{V_2^2 + v_3^2} \\ \hat{\mathbf{v}}_2 = \hat{\mathbf{n}} \times \hat{\mathbf{v}}_1 \\ \hat{\mathbf{n}} = \begin{pmatrix} V_1 & V_2 & V_3 \end{pmatrix}^T / \sqrt{V_1^2 + V_2^2 + v_3^2} \end{array} \tag{14}$$

Let

$$\mathbf{R}_p = \begin{pmatrix} \hat{\mathbf{v}}_1 & \hat{\mathbf{v}}_2 & \hat{\mathbf{n}} \end{pmatrix}^T \tag{15}$$

and define the translation vector

$$\mathbf{t}_p = \begin{pmatrix} 0 \\ 0 \\ V_4 / \sqrt{V_1^2 + V_2^2 + v_3^2} \end{pmatrix} \,. \tag{16}$$

Then $\{\mathbf{R}_p, \mathbf{t}_p\}$ defines the mapping between 3D points coordinate system and plane coordinate system. For example, a 3D point $\mathbf{X}$ is transformed to the plane coordinate system as

$$\mathbf{Y} = \mathbf{R}_p \mathbf{X} + \mathbf{t}_p \,. \tag{17}$$

The first two elements in $\mathbf{Y}$ contains the coordinates (y_1, y_2) within the plane, and the third element y_3 is the distance between the point and the plane. And the other way around, a point (y_1, y_2) in the plane can be transformed to the corresponding 3D point as

$$\mathbf{X} = \mathbf{R}_p^T \left(\begin{pmatrix} y_1 \\ y_2 \\ 0 \end{pmatrix} - \mathbf{t}_p \right) \,. \tag{18}$$

For later use we also define the transformation matrix

$$\mathbf{T} = \begin{pmatrix} \mathbf{R}_p & \mathbf{t}_p \\ \mathbf{0} & 1 \end{pmatrix} \,, \tag{19}$$

and we have the relation

$$\tilde{\mathbf{Y}} = \mathbf{T}\tilde{\mathbf{X}} \,. \tag{20}$$

D.5 Some theory and relations

Assume that we have a point $\mathbf{x}$ in the image, a corresponding point $\mathbf{X}$ in the 3D world, and that the same point is described by $\mathbf{y}$ in the 3D world ground plane. The mapping between $\mathbf{x}$ and $\mathbf{y}$ is defined by the homography as

$$\tilde{\mathbf{x}} \sim \mathbf{H}\tilde{\mathbf{y}} \,. \tag{21}$$

The mapping between $\mathbf{x}$ and $\mathbf{X}$ is defined by the projection matrix as

$$\tilde{\mathbf{x}} \sim \mathbf{P}\tilde{\mathbf{X}} \,. \tag{22}$$

D.6 Relation between H and P

It is easier to relate the homography for the 3D (ground) plane and the projection matrix if we describe the 3D point $\mathbf{X}$ in the 3D plane coordinate system $\mathbf{Y}$. Putting the relation (20) into (22) gives

$$\tilde{\mathbf{x}} \sim \mathbf{P}\mathbf{T}^{-1}\tilde{\mathbf{Y}} \,. \tag{23}$$

Let $\mathbf{P}' = \mathbf{P}\mathbf{T}^{-1}$ denote the projection matrix relative to the plane coordinate system. It is fairly easy to see that columns 1, 2, and 4 in $\mathbf{P}'$ equals $\mathbf{H}$ (the homography describes mapping of points in the plane $y_3 = 0$), i.e.

$$\mathbf{P}' \sim \begin{pmatrix} \mathbf{H}_1 & \mathbf{H}_2 & \mathbf{u} & \mathbf{H}_3 \end{pmatrix} \,, \tag{24}$$

where $\mathbf{H}_k$ denotes column k in $\mathbf{H}$, and $\mathbf{u}$ denotes the extra 'missing' column.

D.7 Focal length from H

In this section we assume that the principal point **c** is known (e.g. from the lens distortion calibration), and that we want to estimate the focal length from the homography. The projection matrix $\mathbf{P}'$ can be written as

$$\mathbf{P}' = \mathbf{K}\begin{pmatrix}\mathbf{R} & \mathbf{t}\end{pmatrix}, \tag{25}$$

where **K** is the camera matrix

$$\mathbf{K} = \begin{pmatrix} f & 0 & c_1 \\ 0 & f & c_2 \\ 0 & 0 & 1 \end{pmatrix} \tag{26}$$

and **R** and **t** are the rotation and translation that transforms the image coordinate system to the 3D world plane coordinate system. Let $\mathbf{R} = (\mathbf{r}_1\ \mathbf{r}_2\ \mathbf{r}_3)$. Then, from (24) we have that

$$\mathbf{H} = \alpha\mathbf{K}\begin{pmatrix}\mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t}\end{pmatrix}, \tag{27}$$

where α is an unknown proportionality factor. We thus have that $\alpha\mathbf{r}_1 = \mathbf{K}^{-1}\mathbf{H}_1$ and $\alpha\mathbf{r}_2 = \mathbf{K}^{-1}\mathbf{H}_2$. Since $\mathbf{r}_1$ and $\mathbf{r}_2$ are columns in a rotation matrix we know that

$$\|\mathbf{r}_1\| = \|\mathbf{r}_2\| = 1 \tag{28}$$

$$\mathbf{r}_1^T\mathbf{r}_2 = 0 \tag{29}$$

This knowledge can be used to estimate the focal length. Constraint (28) leads to

$$\begin{array}{c} (H_{11} - c_1H_{31})^2 + (H_{21} - c_2H_{31})^2 + f^2H_{31}^2 = f^2\alpha^2 \\ (H_{12} - c_1H_{32})^2 + (H_{22} - c_2H_{32})^2 + f^2H_{32}^2 = f^2\alpha^2 \end{array}, \tag{30}$$

which has the solution

$$f = \sqrt{\frac{(H_{11} - c_1H_{31})^2 + (H_{21} - c_2H_{31})^2 - (H_{12} - c_1H_{32})^2 - (H_{22} - c_2H_{32})^2}{H_{32}^2 - H_{31}^2}} \tag{31}$$

$$\alpha = \sqrt{\frac{(H_{11} - c_1H_{31})^2 + (H_{21} - c_2H_{31})^2 + f^2H_{31}^2}{f^2}}. \tag{32}$$

We may also use the constraint (29) to derive an estimate of f. Preliminary experiments indicate however that this latter solution is less reliable.

D.8 Camera position from P

The focal point **f** is the right null space of **P**, i.e.

$$\mathbf{P}\mathbf{f} = \mathbf{0}. \tag{33}$$

The focal point defines the camera position in the 3D space. It is easy to verify that the focal point can be written as

$$\mathbf{f} = \begin{pmatrix} -\mathbf{P}_{1:3}^{-1}\mathbf{P}_4 \\ 1 \end{pmatrix}, \tag{34}$$

where $\mathbf{P}_{1:3}$ denotes the three first columns in $\mathbf{P}$ and $\mathbf{P}_4$ denotes the last fourth column.

D.9 3D position X from image position x

The mapping from a 2D image point $\mathbf{x}$ to a 3D point $\mathbf{X}$ is not unique. In this section we assume that the height above the ground plane (i.e. y_3) is known.

It can be verified that a point $\mathbf{x}$ maps to the 3D ray

$$\mathbf{X} = \mathbf{X}_0 + s\mathbf{U} \quad \text{where} \begin{cases} \mathbf{X}_0 = -\mathbf{P}_{1:3}^{-1}\mathbf{P}_4 & \text{(the focal point)} \\ \mathbf{U} = \mathbf{P}_{1:3}^{-1}\tilde{\mathbf{x}} \end{cases} \tag{35}$$

This ray transformed into plane coordinates becomes

$$\mathbf{Y} = \mathbf{R}_p\mathbf{X} + \mathbf{t}_p = \mathbf{R}_p\mathbf{X}_0 + \mathbf{t}_p + s\mathbf{R}_p\mathbf{U}\,. \tag{36}$$

The line parameter s can be determined if y_3 is known. Then the point $\mathbf{Y}$ can be computed, and finally we can go back to the original 3D coordinate system and get the desired 3D point $\mathbf{X}$.

D.10 Estimation of homography, H

The goal here is to estimate $\mathbf{H}$ in (21). Let $\mathbf{h}^{kT}$ denote row k in the matrix $\mathbf{H}$ and let

$$\mathbf{h} = \begin{pmatrix} \mathbf{h}^1 \\ \mathbf{h}^2 \\ \mathbf{h}^3 \end{pmatrix} \tag{37}$$

denote the matrix $\mathbf{H}$ reshaped into a vector.

D.11 Estimation of H from point correspondences

Equation (21) can be formulated

$$\tilde{\mathbf{x}} \times \mathbf{H}\tilde{\mathbf{y}}\,, \tag{38}$$

which can be rewritten as

$$\begin{pmatrix} \mathbf{0}^T & -\tilde{\mathbf{y}}^T & x_2\tilde{\mathbf{y}}_1^T \\ \tilde{\mathbf{y}}^T & \mathbf{0}^T & -x_1\tilde{\mathbf{y}}^T \\ -x_2\tilde{\mathbf{y}}^T & x_1\tilde{\mathbf{y}}^T & \mathbf{0}^T \end{pmatrix} \mathbf{h} = \mathbf{0}\,. \tag{39}$$

Note that the three rows in the equation system are linearly dependent and the third row can therefore be removed, thus leaving the system

$$\begin{pmatrix} \mathbf{0}^T & -\tilde{\mathbf{y}}^T & x_2\tilde{\mathbf{y}}_1^T \\ \tilde{\mathbf{y}}^T & \mathbf{0}^T & -x_1\tilde{\mathbf{y}}^T \end{pmatrix} \mathbf{h} = \mathbf{0} \,. \tag{40}$$

This is a linear equation system which can be solved for $\mathbf{H}$ using SVD, after collecting equations from a number of points.

D.12 Estimation of H from line correspondences

A line in the 3D ground plane may be represented by two points $\mathbf{y}_0$ and $\mathbf{y}_1$ through which the line passes. The condition that these points lies on a line $\mathbf{l} = (l_1, l_2, l_3)$ in the image is

$$\mathbf{l}^T \mathbf{H} \tilde{\mathbf{y}}_i \quad , \; i = 0, 1 \,. \tag{41}$$

which can be rewritten using $\mathbf{h}$ in (45) as

$$\begin{pmatrix} l_1\tilde{\mathbf{y}}_i^T & l_2\tilde{\mathbf{y}}_i^T & l_3\tilde{\mathbf{y}}_i^T \end{pmatrix} \mathbf{h} = 0 \quad , \; i = 0, 1 \,. \tag{42}$$

This is a linear equation system which can be solved for $\mathbf{H}$ using SVD, after collecting equations from a number of lines.

Note that we can use both point correspondences and line correspondences simultaneously by combining (40) and (42).

D.13 Pre-processing and post-processing

In order to get a more stable solution, it is sometimes useful to pre-process the data. In this case we compute

$$\begin{aligned} \mathbf{x}' &= \mathbf{W}_x(\mathbf{x} - \mathbf{m}_x) \,, \\ \mathbf{y}' &= \mathbf{W}_y(\mathbf{y} - \mathbf{m}_y) \,. \end{aligned} \tag{43}$$

The transformation may for example be based on the mean and standard deviation of the data, to get values somewhere in the range $[-1, 1]$. The homography $\mathbf{H}'$ is then estimated using $\mathbf{x}'$ and $\mathbf{y}'$ and one of the methods above. The final homography is then computed as

$$\mathbf{H} = \begin{pmatrix} \mathbf{W}_x & -\mathbf{W}_x\mathbf{m}_x \\ \mathbf{0} & 1 \end{pmatrix}^{-1} \mathbf{H}' \begin{pmatrix} \mathbf{W}_y & -\mathbf{W}_y\mathbf{m}_y \\ \mathbf{0} & 1 \end{pmatrix} \,. \tag{44}$$

D.14 Estimation of projection matrix, P

The goal here is to estimate $\mathbf{P}$ in (22). Let $\mathbf{p}^{kT}$ denote row k in the matrix $\mathbf{P}$ and let

$$\mathbf{p} = \begin{pmatrix} \mathbf{p}^1 \\ \mathbf{p}^2 \\ \mathbf{p}^3 \end{pmatrix} \tag{45}$$

denote the matrix $\mathbf{P}$ reshaped into a vector.

D.15 Direct estimation of P from point correspondences

This method is useful when the 3D data points does not lie only in a plane. The method may still be useful if all points lie within a plane, but may then be unstable. The direct method to estimate $\mathbf{P}$ is very similar to the method of estimating $\mathbf{H}$ in section D.10. Reformulate 22 as

$$\tilde{\mathbf{x}} \times \mathbf{P}\tilde{\mathbf{X}} = \mathbf{0}\,. \tag{46}$$

which can be rewritten as

$$\begin{pmatrix} \mathbf{0}^T & -\tilde{\mathbf{X}}^T & x_2\tilde{\mathbf{X}}_1^T \\ \tilde{\mathbf{X}}^T & \mathbf{0}^T & -x_1\tilde{\mathbf{X}}^T \\ -x_2\tilde{\mathbf{X}}^T & x_1\tilde{\mathbf{X}}^T & \mathbf{0}^T \end{pmatrix} \mathbf{p} = \mathbf{0}\,. \tag{47}$$

The three rows in the equation system are linearly dependent and the third row can therefore be removed, thus leaving the system

$$\begin{pmatrix} \mathbf{0}^T & -\tilde{\mathbf{X}}^T & x_2\tilde{\mathbf{X}}_1^T \\ \tilde{\mathbf{X}}^T & \mathbf{0}^T & -x_1\tilde{\mathbf{X}}^T \end{pmatrix} \mathbf{p} = \mathbf{0}\,. \tag{48}$$

This is a linear equation system which can be solved for $\mathbf{P}$ using SVD, after collecting equations from a number of points.

Similar pre-processing and post-processing as in section D.13 may be used here as well in order to achieve a more reliable solution.

D.16 Direct estimation of P from line correspondences

A line in 3D may be represented by two points $\mathbf{X}_0$ and $\mathbf{X}_1$ through which the line passes. The condition that these points lies on a line $\mathbf{l} = (l_1, l_2, l_3)$ in the image is

$$\mathbf{l}^T\mathbf{P}\tilde{\mathbf{X}}_i \quad , \; i = 0, 1\,. \tag{49}$$

which can be rewritten using $\mathbf{p}$ in (45) as

$$\begin{pmatrix} l_1\tilde{\mathbf{X}}_i^T & l_2\tilde{\mathbf{X}}_i^T & l_3\tilde{\mathbf{X}}_i^T \end{pmatrix} \mathbf{p} = 0 \quad , \; i = 0, 1\,. \tag{50}$$

This is a linear equation system which can be solved for $\mathbf{P}$ using SVD, after collecting equations from a number of lines.

Note that we can use both point correspondences and line correspondences simultaneously by combining (48) and (50).

D.17 Estimation of P from ground plane homography H

This method is useful when the 3D points lie in a (ground) plane. The method is as follows:

1. First, estimate the ground plane, $\mathbf{V}$ (section D.3). Transform the points $\mathbf{X}$ to ground plane coordinates $\mathbf{y}$ (section D.4).

2. Estimate the homography $\mathbf{H}$ between image points $\mathbf{x}$ and points in the ground plane $\mathbf{y}$ (section D.10).

3. Estimate the focal length f (section D.7 if we not already know the focal length from some other method).

4. Use the formulas section D.7 to compute α, $\mathbf{K}$, $\mathbf{r}_1$, and $\mathbf{r}_2$. Then compute $\mathbf{r}_3 = \mathbf{r}_1 \times \mathbf{r}_2$.

5. Compute the image-to-groundplane projection matrix $\mathbf{P}'$ as

$$\mathbf{P}' = \begin{pmatrix} \mathbf{H}_1 & \mathbf{H}_2 & \alpha\mathbf{K}\mathbf{r}_3 & \mathbf{H}_3 \end{pmatrix} . \tag{51}$$

6. Finally, compute the image-to-3D projection matrix $\mathbf{P} = \mathbf{P}'\mathbf{T}$, where $\mathbf{T}$ is the 3D-to-plane transformation matrix (19).

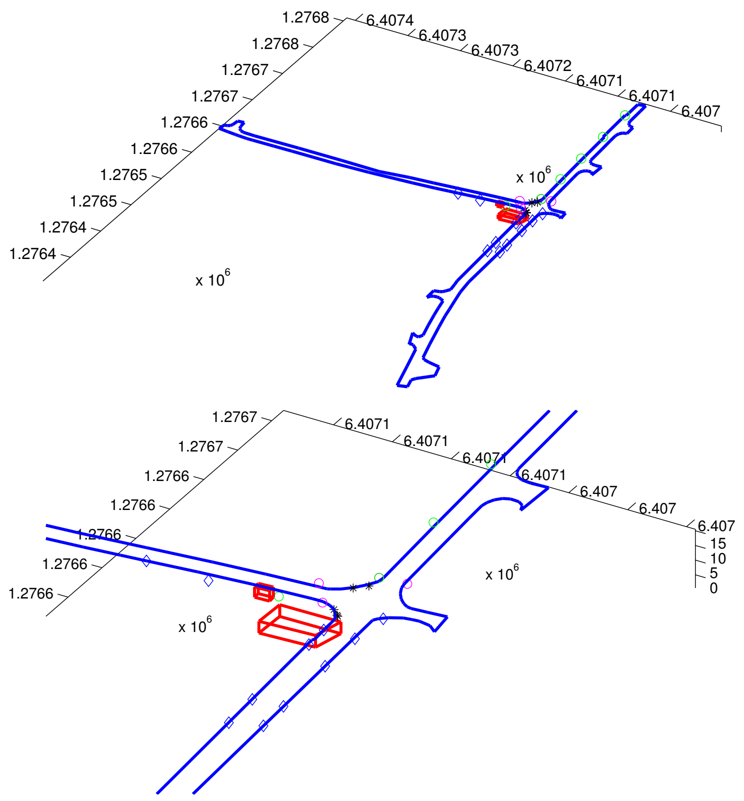


Figure 41: 3D data.

D.18 Examples

This section illustrates the theory in previous sections.

D.18.1 The data

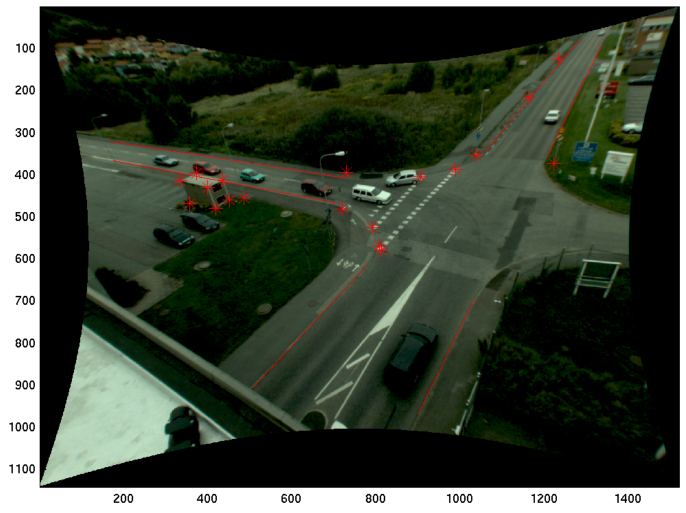
The GPS 3D data is shown in figure 41. There are two cameras at different positions in the intersection. The camera images are shown in figure 42.

D.18.2 Camera 1: Estimation of P directly

The result for camera 1 using the direct method is shown in the top image in figure 43.

D.18.3 Camera 2: Estimation of P from H

The result for camera 2 using the H method is shown in the bottom image in figure 43.



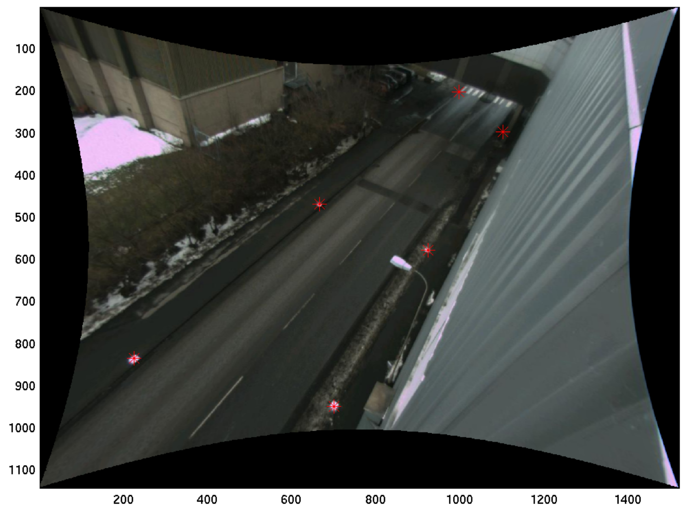
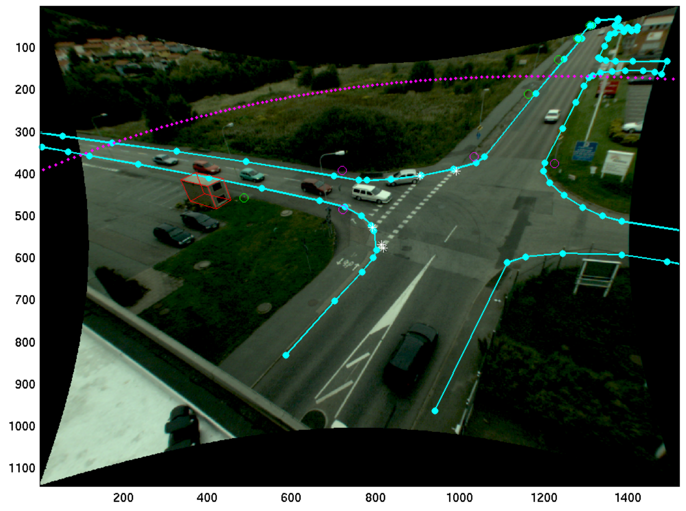


Figure 42: The two camera images and the landmarks.



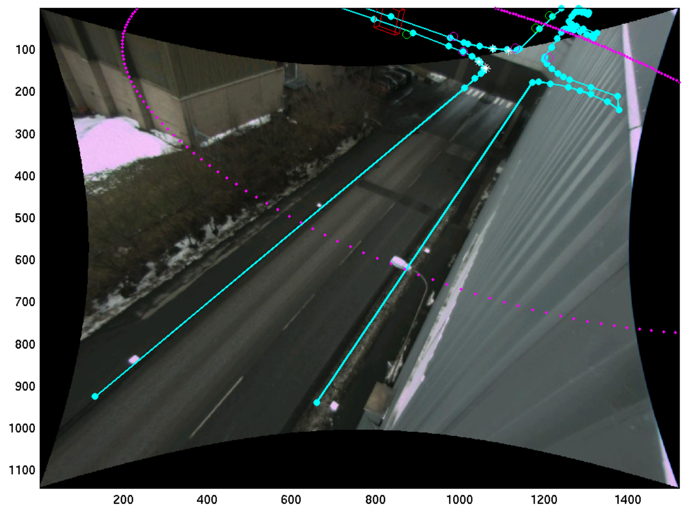


Figure 43: Result for camera 1 and 2. The circle marks out a region with radius 70m.

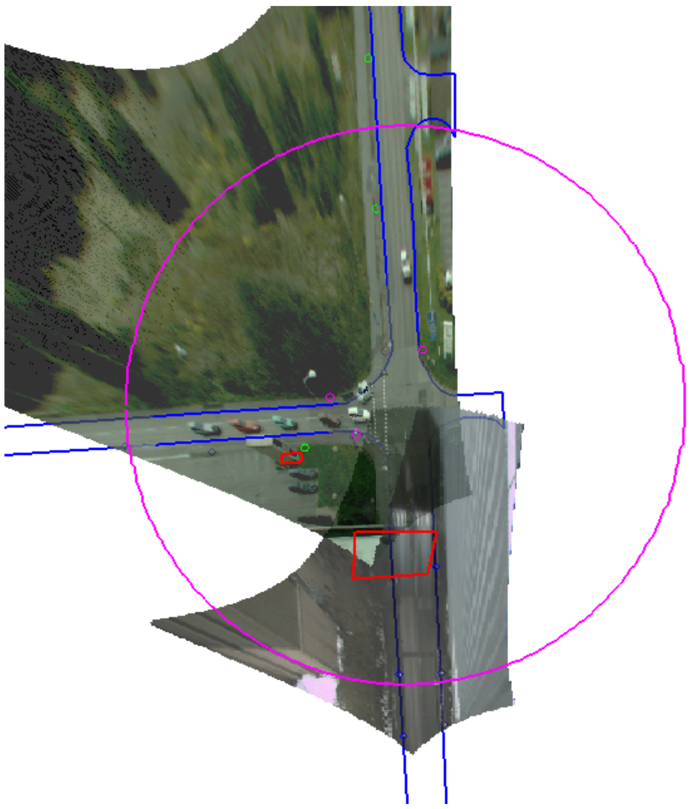


Figure 44: Result for camera 1 and 2 mapped to the 3D domain. The circle marks out a region with radius 70m.

E Simultaneous camera ego-motion compensation and rectification

We will here describe a theory for simultaneous compensation of camera ego-motion and image rectification. See figure 45 for an overview.

This section is outlined as follows: Sections E.1-E describe the theories for image point tracking, lens distortion rectification, and camera ego-motion estimation. Section E.4 gives the implementational details. Section E.5 gives a short description of a statistical background model estimator, which is used as a part of the performance evaluation experiments in section E.6.

E.1 Image point tracking

Our camera ego-motion estimation needs a number of traced points to compute the 'mean' image (point) position. For this we use the KLT (Kanade-Lucas) tracker, see e.g. [27, 22, 31], which tracks a local region template with a subpixel translation model. The KLT tracker is based on the early work of Lucas and Kanade [22], and was fully developed by Tomasi and Kanade [31].

At first we tried using the Harris interest point detector, see [13], to find suitable regions to track (c.f. [27]). But we finally decided to use a regularly sampled grid, in order to have the points spread out in the entire image in a simple manner, see figure 46.

In practice, the point grid has to be reset once in a while due to changes in light and in the static environment. We therefore use two grids that overlap in time and are alternately reset. A first per-point bounding box outlier test is applied to eliminate points that are too far from their original position, i.e. are stuck on some moving vehicle. This step is necessary for getting a good initial guess for the ego-motion solver. The active grid points are later fed in the ego-motion solver to estimate the motion of the camera.

E.2 Lens distortion model

Lens distortion rectification deals with the mapping between the captured, radially distorted, image and a rectified image where the pinhole camera model can be applied, see figure 47 for an example. For our application we used the FOV model from [10], but the choice of model is not critical for the implementation performance due to the highly programmable nature and computational performance of a modern graphics card. The model is briefly described below.

Let $\mathbf{x}_d$ denote a point in the distorted image, $\mathbf{x}_u$ the corresponding point in the undistorted image, and $r_d = \|\mathbf{x}_d\|$, $r_u = \|\mathbf{x}_u\|$. The FOV model is defined as

$$r_d = \frac{1}{\omega}\arctan\left(2r_u \tan\frac{\omega}{2}\right) \;\Leftrightarrow\; r_u = \frac{\tan(r_d\omega)}{2\tan\frac{\omega}{2}}\,. \tag{52}$$

This model corresponds to an ideal fish-eye lens (but also seems to work well for more regular lenses, according to [7]). The parameter ω corresponds to the

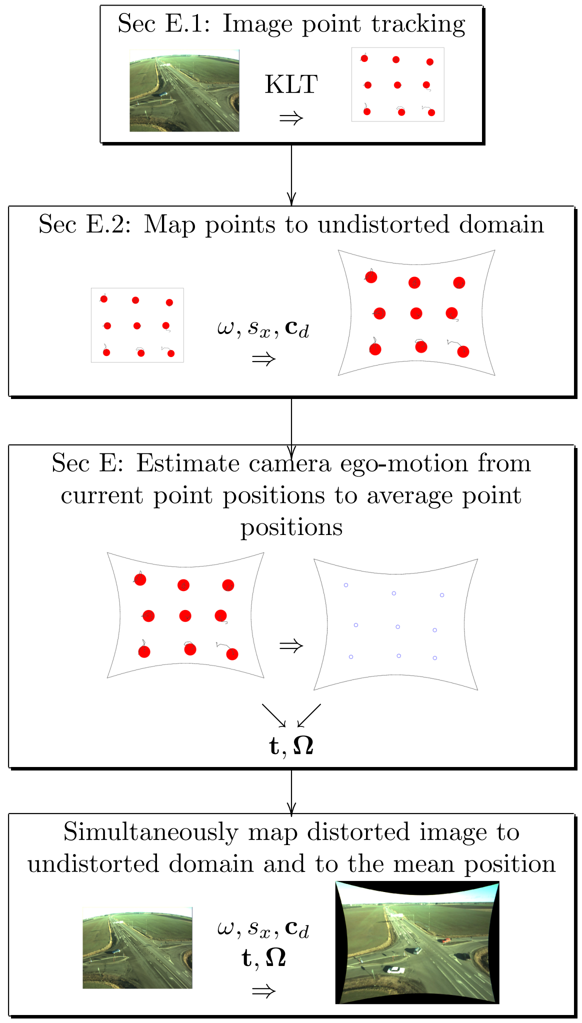

Sec E.1: Image point tracking
KLT
$\Rightarrow$
Sec E.2: Map points to undistorted domain
$\omega, s_x, \mathbf{c}_d$
$\Rightarrow$
Sec E: Estimate camera ego-motion from current point positions to average point positions
$\Rightarrow$
$\mathbf{t}, \mathbf{\Omega}$
Simultaneously map distorted image to undistorted domain and to the mean position
$\omega, s_x, \mathbf{c}_d$
$\mathbf{t}, \mathbf{\Omega}$
$\Rightarrow$

Figure 45: Process overview.

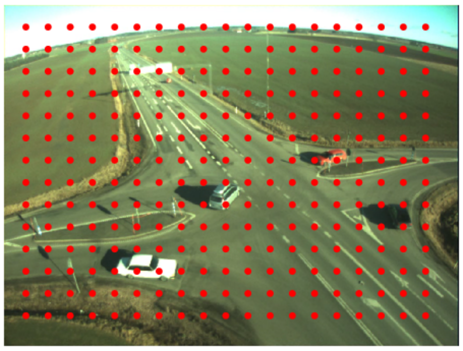


Figure 46: Example of point grid.

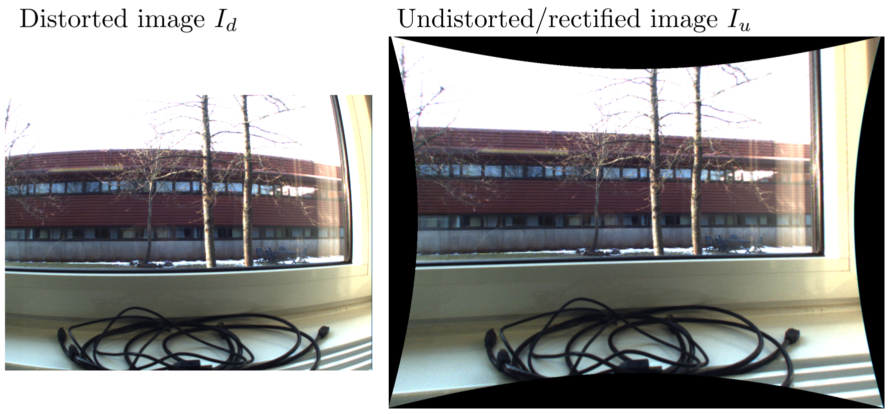


Figure 47: Example of lens distortion and image rectification.

field of view (FOV) angle.

We have observed that this model does not preserve the metric in the image center (i.e. $\frac{dr_u}{dr_d} = 1$ at $r_d = 0$), which sometimes is desirable if we do not want to loose resolution anywhere in the image. Of course one could always rescale r_u in (52) after the mapping from r_d, to fulfill to the preservation requirement. But this is equivalent to using the simpler expression

$$r_d = \frac{\arctan(r_u\omega)}{\omega} \;\Leftrightarrow\; r_u = \frac{\tan(r_d\omega)}{\omega}\,. \tag{53}$$

Note that this new ω (the same symbol is kept for simplicity) has the unit [radians/pixel] if r_d, r_u are measured in pixels.

In line with [10], we also include parameters for the origin and aspect ratio according to

$$\mathbf{x}_d = \begin{pmatrix} s_x & 0 \\ 0 & 1 \end{pmatrix} \mathbf{x}'_d + \mathbf{c}_d \tag{54}$$

(hence we use $r_d = \|\mathbf{x}'_d\|$ in (53) instead). For the estimation of the lens parameters $(\omega, s_x, \mathbf{c}_d)$ see appendix C.

Undistorted points, $\mathbf{x}_u$, will from now on be denoted $\mathbf{x} = (x, y)$ for simplicity.

E.3 Camera ego-motion compensation

E.3.1 Ego-Motion model

There exist several different camera ego-motion models, depending on the type of camera motion (e.g. only rotation) and the geometry of the 3D world (e.g. flat world). In our case we have camera rotation as well as translation, and we did not want to restrict ourselves to a flat world since many intersections are close to buildings and trees.

Let $\mathbf{T} = (T_X, T_Y, T_Z)^T$ denote the instantaneous translation and $\mathbf{\Omega} = (\Omega_X, \Omega_Y, \Omega_Z)^T$ the instantaneous rotation. Moreover, let f denote the camera focal length and (X, Y, Z) denote the 3D point that corresponds to the point $\mathbf{x} = (x, y)^T$ in the image in a common coordinate system, see figure 48.

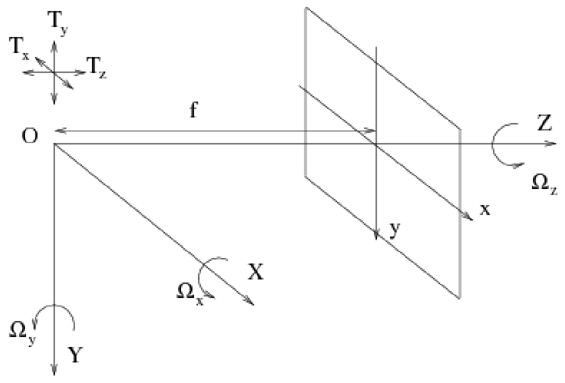


Figure 48: Camera geometry and motion parameters

It is well known that the instantaneous motion $\mathbf{v}$ at point $\mathbf{x}$, assuming a pinhole camera, in the general case can be expressed as

$$\begin{aligned}
\mathbf{v}(\mathbf{x}) &= \begin{pmatrix} f & 0 & -x \\ 0 & f & -y \end{pmatrix} \left(\frac{1}{Z(\mathbf{x})}\mathbf{T} + \frac{1}{f}\mathbf{\Omega} \otimes \begin{pmatrix} \mathbf{x} \\ f \end{pmatrix} \right) \qquad (55) \\
&= \frac{1}{Z(\mathbf{x})} \begin{pmatrix} f & 0 & -x \\ 0 & f & -y \end{pmatrix} \mathbf{T} + \begin{pmatrix} -\frac{xy}{f} & (f + \frac{x^2}{f}) & -y \\ -(f + \frac{y^2}{f}) & \frac{xy}{f} & x \end{pmatrix} \mathbf{\Omega}
\end{aligned}$$

Unfortunately (and naturally) the translation term depends on the distance to the observed 3D point, Z, which is unknown in our case. Various approximations to the first (the translation) term have been tested, e.g.

$$\mathbf{0}, \quad \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \mathbf{t}, \quad \begin{pmatrix} x & 0 \\ 0 & y \end{pmatrix} \mathbf{t}, \quad \begin{pmatrix} 1 & 0 & x & 0 \\ 0 & 1 & 0 & y \end{pmatrix} \mathbf{t}. \qquad (56)$$

The best choice depends (by empirical studies) on the camera setup relative to the 3D ground, existing static objects in the view, etc. We currently use the third one from the left.

E.3.2 Model estimation

The main idea in the ego-motion compensation is to warp the image so that tracked points in the image (assumed to follow from the ego-motion) map to their average position in time.

Let $\{\mathbf{x}(t)_k\}$ denote a set of K points that have been tracked for a period of time, and mapped to the undistorted domain (the tracking is done in the original distorted image). Before we use these points to estimate the camera ego-motion, we first remove outliers by the following criteria: A point that has moved too far from its initial position (the camera pole motion is bounded) will be classified as an outlier for a period of time. If the point after that time is inside the box again, then it is reinstated as an inlier, otherwise it is removed for another period of time. Typically, this handles points that are temporarily occluded by passing objects.

The average position in time is then computed for the remaining points, denoted $\{\bar{\mathbf{x}}_k\}$. After that we compute the motion of point k at time t as $\mathbf{v}_k(t) = \mathbf{x}_k(t) - \bar{\mathbf{x}}_k$.

The camera ego-motion parameters $\mathbf{t}, \mathbf{\Omega}$ in (55-56) can then be computed as the solution to a least squares problem by collecting (55-56) from the set $\{\mathbf{v}_k(t), \mathbf{x}_k(t)\}$ (excluding the outliers).

As a way to make the estimation more robust, new outliers are found as points that do not fulfill the estimated ego-motion (55-56) very well, i.e. have a large residual. The least squares problem is then solved again to give the final estimate.

Once the ego-motion parameters are estimated, we can warp the image accordingly to the 'average' position. The image rectification is done at the same time. In this way we avoid the additional blur caused by resampling the image twice.

E.4 GPU implementation

E.4.1 Programing Interface

There exist several possible ways to implement algorithms on a GPU. One way is to use a Graphics API (Application Programing Interface) such as DirectX or OpenGL. These APIs have a large support from different software and hardware platforms, especially OpenGL which lets you develop applications on every thing from small mobile devices and PlayStation 3 to any PC. There are however some limitations on flexibility. It can be very difficult if not impossible to make some algorithms efficient, even if they are parallel in their nature. This is the main reason why two of the biggest graphics hardware vendors have come up with more versatile APIs. NVIDIA has released CUDA, and AMD has developed the Stream SDK. These are more flexible and can make use of special hardware functionality, such as shared memory between processors and random address writes. These APIs are however vendor specific, and demand newer generations of their Graphics boards. We have chosen to implement our algorithm in a Graphics API (DirectX), mainly for two reasons. It uses functionality specific to Graphics APIs, and as far as we can see the KLT algorithm would not benefit from the extra functionalities of the new APIs. Because of the similarities between the Graphics APIs, techniques used here can easily be transfered to OpenGL.

E.4.2 The KLT Tracker

The KLT tracker is based on the early work of Lucas and Kanade [22], and was fully developed by Tomasi and Kanade [31]. They define the dissimilarity $\epsilon(d)$ between two local regions in two images (I and J) as:

$$\epsilon(d) = \iint_W \left(I(x + \frac{d}{2}) - J(x - \frac{d}{2}) \right)^2 dx. \tag{57}$$

Here $d = (d_x, d_y)$ is the displacement between the two regions and W defines the spatial window (patch size). To find the displacement, the dissimilarity (57) is approximated by its first order Taylor expansion and the minimum is found by differentiation. For the case of discrete images the solution is computed from the 2×2 equation system:

$$\begin{pmatrix} \sum\sum_w g_x^2 & \sum\sum_w g_x \cdot g_y \\ \sum\sum_w g_y \cdot g_x & \sum\sum_w g_y^2 \end{pmatrix} \begin{pmatrix} d_x \\ d_y \end{pmatrix} = 2 \begin{pmatrix} \sum\sum_w (I - J) g_x \\ \sum\sum_w (I - J) g_y \end{pmatrix}, \tag{58}$$

where $g_x = I'_x + J'_x$ and $g_y = I'_y + J'_y$. A more detailed derivation is found in [15].

Calculation Scheme The KLT algorithm is an iterative process, where each iteration can be divided into 5 steps, see Figure 49. Here follows a short explanation of each step. We first extract the two patches (linearly interpolated) that are going to be matched, for the following steps only the elementwise sums and differences are needed so these are also calculated here. Next we apply an neighboring pixel difference filter to $I_w + J_w$. The gradients are multiplied elementwise, as shown in the figure. Then we sum over the 5 unique elements. The last step of the iteration is to estimate the disparity by solving the 2x2 equation system. The J patch position is adjusted according to the current disparity estimate and then the iteration process is started over again. Several different stop criteria exist, this implementation uses a fixed number of iterations.

The KLT has been implemented on the GPU before [28]. There is however a significant difference between their implementation and ours. They do all the calculations for an entire patch and all its iterations in one shader thread

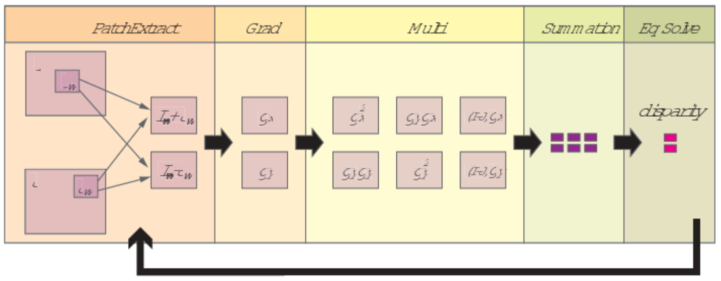


Figure 49: Illustration of the different steps in a GPU implementation of a KLT tracker.

(corresponds to one pixel calculation when doing graphics on the GPU). This is a good solution for the problem if there are enough points to track, however if there are a smaller amount of points (as in our implementation) the GPU will not be fully utilized. The main reason for this is that the GPU uses a large amount of threads to hide memory load latency and thread synchronization. Our method divides the patch calculations to a larger extent between different processors, and therefore can handle smaller amounts of patches more efficient.

Performance The hardware which was used for measuring the performance of the implementation has an Intel Core 2 Duo E6600 CPU, and the graphics board used is based on the Geforce 8800 GTX GPU from NVIDIA. The performance measure of the KLT is based on the following numbers. The two tracking grids were individually made up of 16x16 tracking points, giving us a total of 512 patches to track. To have enough structure within each patch we chose 32x32 as their size. The number of of iterations was set to 6, and the total computation time was 3 ms/frame.

When solving the 5 parameter least square problem for the ego-motion model in section E.3.2, a CPU version of Lapack is used. There are two reasons for this. It creates a better GPU-CPU load balance, and it is easer and almost as fast to use an already finished CPU implementation of an LSQ-Solver.

E.5 Statistical background segmentation

We will in the experiments use time difference, $\|\mathbf{I}_u(t) - \mathbf{I}_u(t-1)\|$, to evaluate the stabilization performance. However, the subsequent step in an image processing based object tracking system is usually some background/foreground classification. One method for background subtraction that has become popular in recent years is [30] statistical background modeling, This method collects color statistics in each pixel individually during time and detects foreground pixels as pixels that have an uncommon color. This method has also been combined with shadow detection in [35], and this combination is currently used in our system. The statistical background model has some ability to handle ego-motion due to the statistical model, but we will show in the experiments that the compensation still improves the result.

We use a CPU implementation of this method, but it can also be implemented in GPU, see e.g. [19].

E.6 Experiments

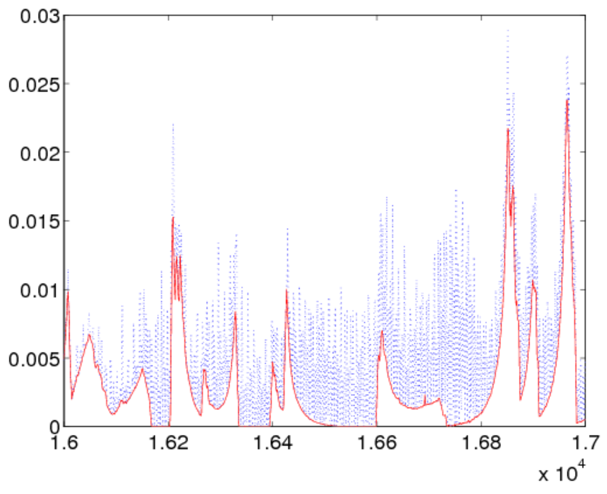
Figures 50-55 show some results on a windy day. The used video was 1h long at 20 fps, but the plot shows only a portion. The plot in figure 50 shows the average time difference, i.e. $mean(\|\mathbf{I}_u(t) - \mathbf{I}_u(t-1)\| > threshold)$, as function of time. We choose to compute the average after thresholding, to ignore the effects from sensor and compression noise that otherwise gives a large total contribution even though it is low in each pixel. The plot shows the average both without and with ego-motion compensation. We see that the average without the compensation contains a lot of spikes, which are to a great extent removed by the compensation. The remaining fluctuation mainly comes from moving objects in the image.

The plot in figure 51 shows the corresponding average for the statistical background subtraction with shadows detected and removed. This method can deal with some ego-motion even without the compensation, but the result is still better after the compensation.

Figures 52-55 shows the frames where the averages with and without compensation differ the most, i.e. in a sense the 'best' and 'worst' cases. However, the 'worst' case only shows that the performance is not decreased when using the compensation. There are still some cases in the sequence where the compensation has been insufficient, due to some extraordinary motion, figure 56 shows an example. These cases however appear to be quite rare.

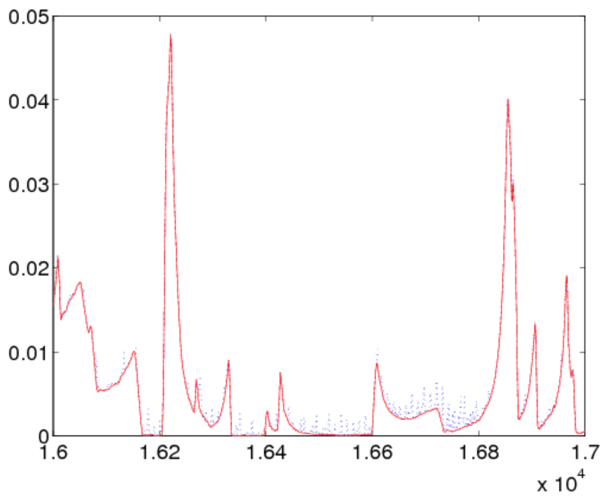
E.7 Performance

When measuring the overall performance of the implementation, we include all steps starting when the image is in main memory (CPU memory) and ending when the result has been downloaded to main memory. More precisely, the implementation uploads the 1024x768 image into the graphics memory, and computes the tracking. Then the least squares ego-motion problem is solved and the image is transformed accordingly to the ego-motion solution and the rectification parameters. The resulting image is then downloaded from the graphics memory (larger due to rectification, approximately 1440x1000). The measured time for these operations is 33 ms/frame, well within the bounds of a real-time system.



Frame number

Figure 50: Example of mean value of (the thresholded) time difference as function of time. Both with and without ego-motion compensation.



Frame number

Figure 51: Example of mean value of the statistical subtraction as function of time. Both with and without ego-motion compensation.

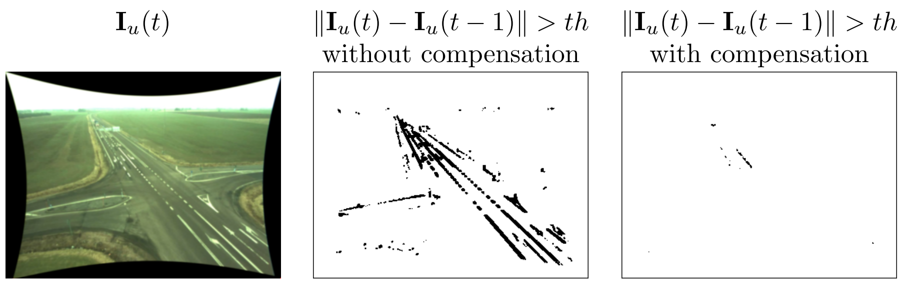

$\mathbf{I}_u(t)$
$\|\mathbf{I}_u(t) - \mathbf{I}_u(t-1)\| > th$ without compensation
$\|\mathbf{I}_u(t) - \mathbf{I}_u(t-1)\| > th$ with compensation

Figure 52: The frame where the averages with and without compensation has the largest negative difference. Col 1: Rectified image. Col 2: Frame difference without ego-motion compensation (using th=50, the original had 255 as max value on each color channel). Col 3: Frame difference with ego-motion compensation.

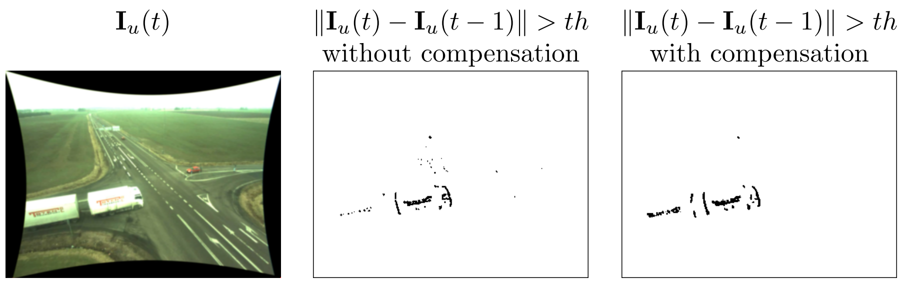

$\mathbf{I}_u(t)$
$\|\mathbf{I}_u(t) - \mathbf{I}_u(t-1)\| > th$ without compensation
$\|\mathbf{I}_u(t) - \mathbf{I}_u(t-1)\| > th$ with compensation

Figure 53: The frame where the averages with and without compensation has the largest positive difference.

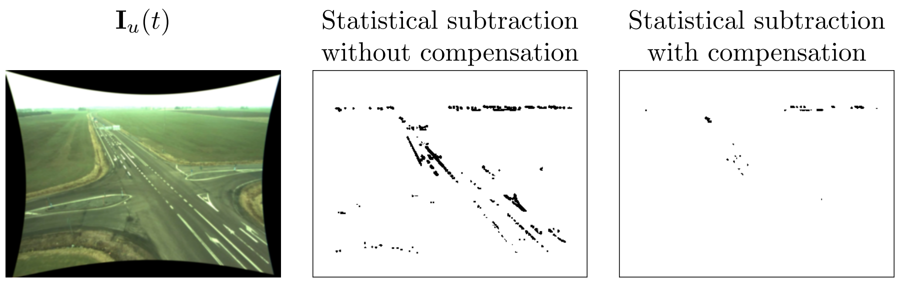

$\mathbf{I}_u(t)$
Statistical subtraction without compensation
Statistical subtraction with compensation

Figure 54: The frame where the averages with and without compensation has the largest negative difference. Col 1: Rectified image. Col 2: Statistical subtraction without ego-motion compensation. Col 3: Statistical subtraction with ego-motion compensation.

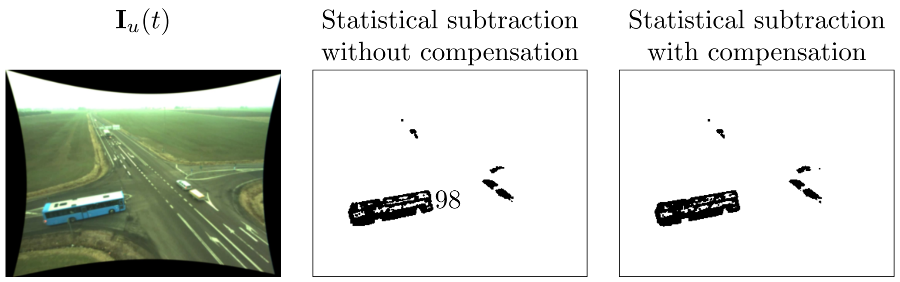

$\mathbf{I}_u(t)$
Statistical subtraction without compensation
Statistical subtraction with compensation

Figure 55: The frame where the averages with and without compensation has the largest positive difference. (Note that the subtraction indicates a false vehicle near the center of the image. This is because a vehicle earlier in time has been temporarily stationary for a longer period of time and has therefore become

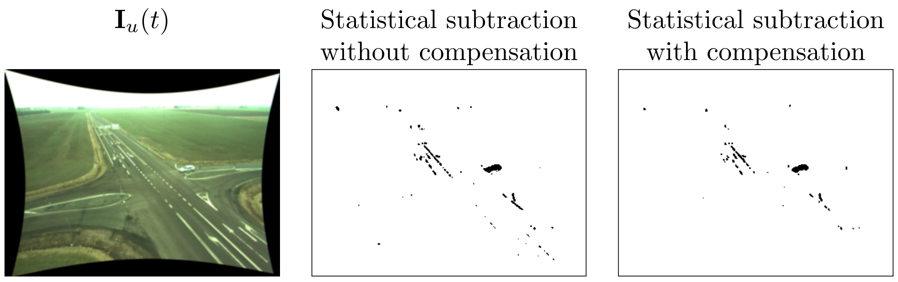

$\mathbf{I}_u(t)$
Statistical subtraction without compensation
Statistical subtraction with compensation

Figure 56: Col 1: Rectified image. Col 2: Frame difference without ego-motion compensation. Col 3: Frame difference with ego-motion compensation, where the compensation was insufficient.

F Algorithm for the statistical background model and foreground-shadow-highlight classification

The algorithm below is a generalization of [35] to also include highlight detection. Algorithm 1 describes the updating of the background model parameters (the same as in [35]). Algorithm 2 use the background model to classify pixels into foreground/shadow/highlight. The notation used in both algorithms can be found in table 2.

Table 2: Notation for algorithm 1 and 2.

x_t	pixel value at time t
f_t	frame at time
w	mixing density
μ	mean
σ^2	variance
$x_{rgb,t}$, $\mu_{rgb,k}$	RGB-components of x_t and μ_k
K	number of mixture components
D	dimension of mixture components
α	learning rate
λ	threshold
σ^2_{init}	initial covariance vector
σ^2_{min}	minimum covariance component allowed
T	threshold
k	integer in the range $[1, ..., K]$
m	integer in the range $[1, ..., K]$
$match$	boolean
B	number of mixture components belonging to the background
$\hat{B}$	class label (background>0, foreground=-1, shadow=-2, highlight=-3)
β_1, β_2	shadow thresholds $0 \leq \beta_1 < \beta_2 < 1$
τ_s	chrominance threshold for the shadow
β_3, β_4	highlight thresholds $1 < \beta_3 < \beta_4$
τ_h	chrominance threshold for the highlight
$\circ$	element-wise multiplication

Algorithm 1 Background modeling update using a mixture of D-dimensional Gaussians

```
for all $x_t$ in $f_t$ do
  $match = 0$
  # Check if the new color is close to one of the components
  for all $k \in [1, ..., K]$ do
    $d_k^2 = \sum_{d=1}^{D} \frac{(x_{t,d} - \mu_{k,d})^2}{\sigma_{k,d}^2}$
    if $d_k < \lambda$ then
      if $match = 0$ then
        $m = k$
      else if $\frac{w_k}{\sqrt{\|\sigma_k^2\|}} > \frac{w_m}{\sqrt{\|\sigma_m^2\|}}$ then
        $m = k$
      end if
      $match = 1$
    end if
  end for

  if $match = 0$ then
    # The new color is not close to a component, initiate a new component
    # (by reusing the least useful component)
    $m = K$
    $w_m = \alpha$
    $\mu_m = x_t$
    $\sigma_m^2 = \sigma_{init}^2$
  else
    # The new color is close to a component, update that component
    $w_m = (1 - \alpha)w_m + \alpha$
    $\rho_m = \frac{\alpha}{w_m}$
    $\mu_m = (1 - \rho_m)\mu_m + \rho_m \cdot x_t$
    $\sigma_m^2 = (1 - \rho_m)\sigma_m^2 + \rho_m(x_t - \mu_m) \circ (x_t - \mu_m)$

    Make sure no components of $\sigma_m^2$ are less than $\sigma_{min}^2$.
  end if

  # Decrease weights for non-matching components
  for all $k \in [1, ..., K] \neq m$ do
    $w_k = (1 - \alpha)w_k$
  end for

  # Sort components according to importance
  if $match \neq 0$ then
    Sort $w, \mu, \sigma$ with respect to $(\frac{w_1}{\sqrt{\|\sigma_1^2\|}}, ..., \frac{w_K}{\sqrt{\|\sigma_K^2\|}})$.
  end if

  # Find the (number of) components that models the background
  $B = argmin_b \left( \sum_{k=1}^{b} w_k > T \right)$
end for
```

Algorithm 2 Background subtraction for algorithm 1 incorporating shadow and highlight detection.

```
for all $x_t$ in $f_t$ do
    $\hat{B} = -1$
    # Check if the new color belongs to the background
    for all $k \in [1, ..., B]$ do
        $d_k^2 = \sum_{d=1}^{D} \frac{(x_{t,d} - \mu_{k,d})^2}{\sigma_{k,d}^2}$
        if $d_k < \lambda$ then
            $\hat{B} = k$
        end if
    end for

    # Classify foreground pixels
    if $\hat{B} = -1$ then
        for all $k \in [1, .., B]$ do
            $Dv = \frac{x_{rgb,t}^T \hat{\mu}_{rgb,k}}{\|\mu_{rgb,k}\|}$
            $Dc = \|x_{rgb,t} - (x_{rgb,t}^T \hat{\mu}_{rgb,k})\hat{\mu}_{rgb,k})\|$
            if $\beta_1 \leq Dv \leq \beta_2$ and $Dc \leq \tau_s$ then
                # Shadow
                $\hat{B} = -2$
                break
            else if $\beta_3 \leq Dv \leq \beta_4$ and $Dc \leq \tau_h$ then
                # Highlight
                $\hat{B} = -3$
                break
            end if
        end for
    end if
end for
```

G 3D box optimization procedure

This section describes the optimization procedure to find the optimal 3D box model to an image of a segmented object, see section 9 for more details. We repeat the similarity measure for reference:

Let $I(\mathbf{x})$ be the classified image and $S(\mathbf{x})$ be the simulated image, and represent the classes as follows

$$I(\mathbf{x}), S(\mathbf{x}) = \begin{cases} 0 & \text{if background} \\ 128 & \text{if shadow} \\ 255 & \text{if foreground} \end{cases} \tag{59}$$

Furthermore, let $\mathcal{V}$ denote the valid pixels, i.e. pixels that are not occluded by other objects (buildings, vehicles, image rectification borders, etc.). Define the similarity between the classified image and the simulated image as

$$s(I, S) = \sum_{\mathbf{x} \in \mathcal{V}} p(I(\mathbf{x}))S(\mathbf{x})\,, \tag{60}$$

where

$$p(I) = \begin{cases} p_b < 0 \text{ if I=0 (background)} \\ p_s > 0 \text{ if I=128 (shadow)} \\ p_f > 0 \text{ if I=255 (foreground)} \end{cases} \tag{61}$$

Algorithm 3 Optimization of box position (fixed size and orientation). Basically, the algorithm tries different positions, with decreasing step size, around the currently best value/state until the similarity measure is no longer increasing.

```
Let $\{\mathcal{T}_n\} = \{$ Increase $x$, Decrease $x$, Increase $y$, Decrease $y$ $\}$, with initially
1 meter step size (then divided by two a few times during the iterations).

# Given size, orientation, and initial position
$(L, W, H, x_0, y_0, \varphi) = ...$
# Initial optimum
$s_{opt} = -\infty$
$\mathcal{B}_{opt} = \{L, W, H, x_0, y_0, \varphi\}$
# Loop over transformation scales
for $m \in [0, ..., M]$ do
  # Loop over list of transformations as long as optimum is increasing
  iter = 0
  $n = 0$
  while iter $< |\{\mathcal{T}_k\}|$ do
    iter = iter + 1
    $n = \mathrm{mod}(n + 1, |\{\mathcal{T}_k\}|)$ # Get next transformation
    $\mathcal{B}_{new} = \mathrm{transform}(\mathcal{B}_{opt}, \mathcal{T}_n/2^m)$ # Transform box
    $S_{new} = \mathrm{simulate\_box\_image}(\mathcal{B}_{new})$ # Simulate a box image
    $s_{new} = s(I, S_{new})$ # Compute similarity
    # Store new optimum if new box is better
    if $s_{new} > s_{opt}$ then
      $s_{opt} = s_{new}$
      $\mathcal{B}_{opt} = \mathcal{B}_{new}$
      iter = 0
    end if
  end while
end for
```

Algorithm 4 Optimization of box position and size (fixed orientation). Basically, the algorithm tries different sizes in algorithm 3.

```
# Given orientation, and initial position
$(x_0, y_0, \varphi) = ...$
# Loop over list of box sizes
$s_{opt} = -\infty$
**for all** $(L_k, W_k, H_K)$ in $\{(L_k, W_k, H_k)\}$ **do**
  # Optimize position using algorithm 3
  $(s_k, \mathcal{B}_k)$ = algorithm_3$(L_k, W_k, H_k, x_0, y_0, \varphi)$
  # Store new optimum if new box is better
  **if** $s_k > s_{opt}$ **then**
    $s_{opt} = s_k$
    $\mathcal{B}_{opt} = \mathcal{B}_k$
  **end if**
**end for**
```

H Linear models

This section shortly summarize the linear models 'constant speed' and 'constant acceleration' that have been explored in the causal and non-causal Kalman filters in the tracking system. Details on these linear models can be found in e.g. [34]. The formulas describe the 1D version, but can easily be generalized to the 2D case.

We will for the state and measurement equations in the Kalman filter use the notation (same as in [33])

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{w}_k \tag{62}$$

$$\mathbf{z}_{k+1} = \mathbf{H}\mathbf{x}_k + \mathbf{v}_k \tag{63}$$

where $\mathbf{x}$ is the state vector, $\mathbf{z}$ is the measurement vector, and $\mathbf{w}$,$\mathbf{v}$ are the state noise and measurement noise respectively. Furthermore, let $\mathbf{Q}$ denote the process covariance matrix and $\mathbf{R}$ the measurement noise covariance matrix.

H.1 Constant speed

Let the constant speed model denote the constraint $\dot{v}(t) = b\dot{w}(t)$, where w is white unit noise. This model corresponds in the discrete case (assuming piecewise constant noise during the sample period T) to

$$\mathbf{x} = \begin{pmatrix} x \\ v \end{pmatrix} \quad , \quad \mathbf{A} = \begin{pmatrix} 1 & T \\ 0 & 1 \end{pmatrix} \quad , \quad \mathbf{Q} = b^2 \begin{pmatrix} T^3/3 & T^2/2 \\ T^2/2 & T \end{pmatrix} \tag{64}$$

H.2 Constant acceleration

Let the constant acceleration model denote the constraint $\dot{a}(t) = b\dot{w}(t)$, where w is white unit noise. This model corresponds in the discrete case (assuming piecewise constant noise during the sample period T) to

$$\mathbf{x} = \begin{pmatrix} x \\ v \\ a \end{pmatrix} \quad , \quad \mathbf{A} = \begin{pmatrix} 1 & T & T^2/2 \\ 0 & 1 & T \\ 0 & 0 & 1 \end{pmatrix} \quad , \quad \mathbf{Q} = b^2 \begin{pmatrix} T^5/20 & T^4/8 & T^3/6 \\ T^4/8 & T^3/3 & T^2/2 \\ T^3/6 & T^2/2 & T \end{pmatrix} \tag{65}$$

I Bicycle Model

This section describes how to derive the differential equations which govern the bicycle model's dynamic motion and how to write them in a time discrete form. A good introduction to the Kalman filter and EKF can be found in [33]. The bicycle model is illustrated in figure 57.

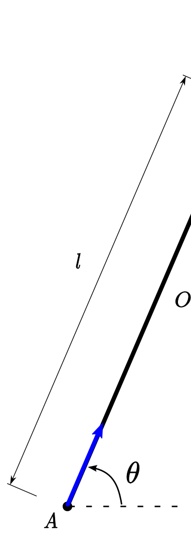


Figure 57: The bicycle model.

From rigid body mechanics for planar motion we know that the speed in point B is

$$\mathbf{v}_B = \mathbf{v}_A + \boldsymbol{\omega} \times \mathbf{r}_{B/A} \ , \tag{66}$$

where $\mathbf{v}_A$ is the velocity along the bicycle, $\omega = \dot{\theta}$, and $\mathbf{r}_{B/A}$ is read the distance vector from point A to B. With the notation introduced in figure 57, (66) gives the following equations

$$\dot{x} = v\cos\theta + \frac{v}{2}\tan\psi\sin\theta \tag{67}$$

$$\dot{y} = v\sin\theta + \frac{v}{2}\tan\psi\cos\theta \tag{68}$$

$$\dot{\theta} = \frac{v\tan\psi}{l} \tag{69}$$

$$\dot{\psi} = b \tag{70}$$

$$\dot{v} = a\,, \tag{71}$$

where a and b are constant, and $v = v_A$.

(Hint: Use $v_B^2 = v_A^2 + (\dot{\theta} l)^2$ and $\dot{\theta} l = v_B \sin\psi$ to get $\dot{\theta} l = v_A \tan\psi = v \tan\psi$. Then put this result into $\dot{\mathbf{x}} = \mathbf{v}_O = \mathbf{v}_A + \frac{1}{2}\boldsymbol{\omega} \times \mathbf{r}_{B/A} = ...$)

Time Discretization

In order to use (67)-(71) for extended Kalman filtering (EKF) the equations needs to be written in a discrete form. A simple way to achieve discrete form this is to utilize the Euler method for solving the differential equations, which gives the following equations

$$x_k = x_{k-1} + T(v_{k-1}\cos\theta_{k-1} + \frac{v_{k-1}}{2}\tan\psi_{k-1}\sin\theta_{k-1}) \tag{72}$$

$$y_k = y_{k-1} + T(v_{k-1}\sin\theta_{k-1} + \frac{v_{k-1}}{2}\tan\psi_{k-1}\cos\theta_{k-1}) \tag{73}$$

$$\theta_k = \theta_{k-1} + T\frac{v_{k-1}\tan\psi_{k-1}}{l} \tag{74}$$

$$\psi_k = \psi_{k-1} + Tb \tag{75}$$

$$v_k = v_{k-1} + Ta\,, \tag{76}$$

where T is the time step. Equations (72)-(76) can then be used in an extended Kalman filtering scheme.

J Simplified Bicycle Model

This section describes how to derive the differential equations which govern the bicycle model's dynamic motion and how to write them in a time discrete form. A good introduction to the Kalman filter and EKF can be found in [33]. The simplified bicycle model is illustrated in figure 58.

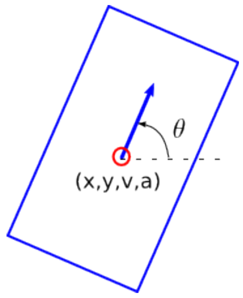


Figure 58: The simplified bicycle model.

J.1 Constant speed model

With the notation introduced in figure 57 define a state vector and its derivative as

$$\mathbf{x}(t) = \begin{pmatrix} x(t) \\ y(t) \\ \theta(t) \\ v(t) \end{pmatrix} \quad , \quad \dot{\mathbf{x}}(t) = \begin{pmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{\theta}(t) \\ \dot{v}(t) \end{pmatrix} = \begin{pmatrix} v(t)\cos\theta(t) \\ v(t)\sin\theta(t) \\ 0 \\ 0 \end{pmatrix} . \tag{77}$$

Time Discretization

In order to use (77) for extended Kalman filtering (EKF) the equations needs to be written in a discrete form. A simple way to achieve discrete form this is to utilize the Euler method for solving the differential equations, which gives the following equations

$$\mathbf{x}_k = \begin{pmatrix} x_k \\ y_k \\ \theta_k \\ v_k \end{pmatrix} = \begin{pmatrix} x_{k-1} + Tv_{k-1}\cos\theta_{k-1} \\ y_{k-1} + Tv_{k-1}\sin\theta_{k-1} \\ \theta_{k-1} \\ v_{k-1} \end{pmatrix} . \tag{78}$$

where T is the time step. Equations (78) can then be used in an extended Kalman filtering scheme.

J.2 Constant acceleration model

With the notation introduced in figure 57 define a state vector and its derivative as

$$\mathbf{x}(t) = \begin{pmatrix} x(t) \\ y(t) \\ \theta(t) \\ v(t) \\ \dot{\theta}(t) \\ a(t) \end{pmatrix} \quad , \quad \dot{\mathbf{x}}(t) = \begin{pmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{\theta}(t) \\ \dot{v}(t) \\ \ddot{\theta}(t) \\ \dot{a}(t) \end{pmatrix} = \begin{pmatrix} v(t)\cos\theta(t) \\ v(t)\sin\theta(t) \\ \dot{\theta}(t) \\ a(t) \\ 0 \\ 0 \end{pmatrix} . \qquad (79)$$

Time Discretization

In order to use (79) for extended Kalman filtering (EKF) the equations needs to be written in a discrete form. A simple way to achieve discrete form this is to utilize the Euler method for solving the differential equations, which gives the following equations

$$\mathbf{x}_k = \begin{pmatrix} x_k \\ y_k \\ \theta_k \\ v_k \\ \dot{\theta}_k \\ a_k \end{pmatrix} = \begin{pmatrix} x_{k-1} + Tv_{k-1}\cos\theta_{k-1} \\ y_{k-1} + Tv_{k-1}\sin\theta_{k-1} \\ \theta_{k-1} + T\dot{\theta}_{k-1} \\ v_{k-1} + Ta_{k-1} \\ \dot{\theta}_{k-1} \\ a_{k-1} \end{pmatrix} . \qquad (80)$$

where T is the time step. Equations (80) can then be used in an extended Kalman filtering scheme.

References

[1] Håkan Ardö and Rikard Berthilsson. Adaptive background estimation using intensity independent features. In *Proceedings of the 2006 British Machine and Vision Conference BMVC'06*, page III:1069, Edinburgh, Scotland, September 2006.

[2] Stefan Atev, Hemanth Arumugam, Osama Masoud, Ravi Janardan, and Nikolaos Papanikolopoulos. A vision-based approach to collision prediction at trafic intersections. *IEEE Transactions on Intelligent Transportation Systems*, 6(4):416–423, December 2005.

[3] Stefan Atev and Nikolaos Papanikolopoulos. Multi-view 3D vehicle tracking with a constrained filter. In *IEEE International Conference on Robotics and Automation*, pages 2277–2282, Pasadena, CA, USA, May 2008.

[4] Andreas Böckert. Vehicle detection and classification in video sequences. Master's thesis, Linköping University, SE-581 83 Linköping, Sweden, 2002. LiTH-ISY-EX-3270.

[5] Distributed by the National Renewable Energy Laboratory. Center for Renewable Energy Resources. Renewable Resource Data Center. NREL, SOLPOS 2.0, February 2000. URL: `http://rredc.nrel.gov/solar/codes_algs/solpos`.

[6] Enrique J. Carmona, Javíer Martinez-Cantos, and José Mira. A new video segmentation method of moving objects based on blob-level knowledge. *Pattern Recognition Letters*, 29:272–285, 2008.

[7] David Claus and Andrew W. Fitzgibbon. A rational function lens distortion model for general cameras. In *CVPR'05*, 2005.

[8] Hendrik Dahlkamp, Hans-Hellmut Nagel, Artur Ottlik, and Paul Reuter. A framework for model-based tracking experiments in image sequences. *International Journal of Computer Vision*, 73(2):139–157, 2007.

[9] Hendrik Dahlkamp, Arthur E.C. Pece, Artur Ottlik, and Hans-Hellmut Nagel. Differential analysis of two model-based vehicle tracking approaches. In *DAGM'04*, pages 71–78, 2004.

[10] Frederic Devernay and Olivier Faugeras. Straight lines have to be straight: Automatic calibration and removal of distortion from scenes of structured environments. *Machine Vision and Applications*, 13:14–24, 2001.

[11] Per-Erik Forssén. Updating camera location and heading using a sparse displacement field. Technical Report LiTH-ISY-R-2318, Dept. EE, Linköping University, SE-581 83 Linköping, Sweden, November 2000.

[12] Torkel Glad and Lennart Ljung. *Reglerteknik; Grundläggande teori*. Studentlitteratur, 1989.

[13] C. G. Harris and M. Stephens. A combined corner and edge detector. In *4th Alvey Vision Conference*, pages 147–151, September 1988.

[14] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision.* Cambridge University Press, 2000.

[15] Johan Hedborg, Johan Skoglund, and Michael Felsberg. KLT tracking implementation on the GPU. In *Proceedings SSBA 2007*, Linkoping, Sweden, Mars 2007.

[16] T. Horprasert, D. Harwood, and L.S. Davis. A statistical approach for real-time robust background subtraction and shadow detection. In *Proceedings of IEEE ICCV'99 FRAME-RATE Workshop*, 1999.

[17] Ajay J. Joshi, Stefan Atev, Osama Masoud, and Nikolaos Papanikolopoulos. Moving shadow detection with low- and mid-level reasoning. In *IEEE International Conference on Robotics and Automation*, April 2007.

[18] Zu Whan Kim and Jitendra Malik. Fast vehicle detection with probabilistic feature grouping and its application to vehicle tracking. In *Proceedings of ICCV'03*, pages 524–531, 2003.

[19] Shu-Ju Lee and Chang-Sung Jeong. Real-time object segmentation based on gpu. In *Internation Conference on Computational Intelligence and Security*, volume 1, pages 739–742, 2006.

[20] Lennart Ljung and Mille Millnert. *Digital signalbehandling.* Studentlitteratur, 1995.

[21] Jianguang Lou, Tieniu Tan, Weiming Hu, Hao Yang, and Steven J. Maybank. 3-D model-based vehicle tracking. *IEEE Transactions on image processing*, 14(10):1561–1569, October 2005.

[22] B. Lucas and T. Kanade. An Iterative Image Registration Technique with Applications to Stereo Vision. In *Proc. Darpa IU Workshop*, pages 121–130, 1981.

[23] Derek R. Magee. Tracking multiple vehicles using foreground, background and motion models. *Image and Vision Computing*, 22:143–155, 2004.

[24] Fredrik Moeschlin. Ivss intersection accidents: Automatic repair and quality measuring of vehicle trajectries. Technical Report 91413, Volvo Car Corporation, March 2008.

[25] Hans-Helmut Nagel. Steps towards a cognitive vision system. *AI Magazine*, 25(2):31–50, Summer 2004.

[26] Andrea Prati, Ivana Mikić, Mohan M. Triveldi, and Rita Cucchiara. Detecting moving shadows: Algorithms and evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(7):918–923, 2003.

[27] J. Shi and C. Tomasi. Good features to track. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994.

[28] Sudipta N Sinha, Jan-Michael Frahm, Marc Pollefeys, and Yakup Genc. Gpu-based video feature tracking and matching. Technical report, Department of Computer Science, UNC Chapel Hill, May 2006.

[29] Xuefeng Song and Ram Nevatia. Detection and tracking of moving vehicles in crowded scenes. In *IEEE Workshop on Motion and Video Computing, WMVC'07*, pages 4–4, 2007.

[30] C. Stauffer and W. Grimson. Adaptive background mixture models for real-time tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 246–252, 1999.

[31] C. Tomasi and L. Kanade. Detection and tracking of point features. Technical Report CMU-CS-91-132, Carnegie Mellon University, April 1991.

[32] Yuko Uematsu and Hideo Saito. Vision-based registration for augmented reality with integration of arbitrary multiple planes. In *13th International Conference on Image Analysis and Processing, ICIAP 2005*, pages 155–162, Italy, September 2005.

[33] Greg Welch and Gary Bishop. An introduction to the kalman filter. Technical report, Chapel Hill, NC, USA, 1995.

[34] Åke Wernersson. 'tidsdiskreta rörelsemodeller', utdrag ur kursmaterial om sensorstyrda autonoma robotar, 1993.

[35] John Wood. Statistical background models with shadow detection for video based tracking. Master's thesis, Linköping University, SE-581 83 Linköping, Sweden, March 2007. LiTH-ISY-EX–07/3921–SE.

IVSS

Intelligent Vehicle Safety Systems

Vägverket

LINKÖPINGS UNIVERSITET

Linköping University

CHALMERS

VOLVO

SAAB

Autoliv