EFFICIENT MIN-COST FLOW TRACKING WITH BOUNDED MEMORY AND COMPUTATION

PHILIP LENZ

DOI: 10.5445/IR/1000049448

EFFICIENT MIN-COST FLOW TRACKING WITH BOUNDED MEMORY AND COMPUTATION

Zur Erlangung des akademischen Grades eines

DOKTORS DER INGENIEURWISSENSCHAFTEN

von der Fakultät für Maschinenbau
des Karlsruher Instituts für Technologie
genehmigte

DISSERTATION

von

DIPL.-ING. PHILIP LENZ

aus Pfinztal

Tag der mündlichen Prüfung:	23. Februar 2015
Hauptreferent:	Prof. Dr.-Ing. Christoph Stiller
Korreferent:	Prof. Dr. Raquel Urtasun

For Sybille.

PREFACE

This thesis was written during my time as a staff researcher at the *Institut für Mess- und Regelungstechnik* at the *Karlsruhe Institute für Technologie* (KIT). It was supervised by Prof. Dr.-Ing. Christoph Stiller. I want to thank him for offering me the opportunity to investigate multiple aspects of computer vision in the context of autonomous driving and providing a reliable research environment.

In particular, I want to sincerely thank Prof. Dr. Raquel Urtasun for agreeing to be the second reviewer of this thesis and providing me the opportunity to visit her at the Toyota Technological Institute at Chicago as a guest researcher. I have learned a lot about research in general and optimization in particular in innumerable discussions with her and her team. I owe her my gratitude.

Moreover, I want to thank all my colleagues in Karlsruhe, in particular my scrum partners, Dr.-Ing. Miriam Schönbein, Dr.-Ing. Henning Lategahn, and Dr.-Ing. Holger Rapp for many open-minded discussions, bearing the goal in mind, and always having time for a good coffee. I also want to express my deep gratitude to Dr.-Ing. Andreas Geiger for initiating and jointly working on the great KITTI project as well as supporting my work on multi-target tracking and always having time for a tremendously helpful discussion and a tea at MPI in Tübingen. And I want to thank all proofreaders for their invaluable work – all remaining errors are of course my own.

For financial support of my work and projects, the possibility to visit leading computer vision conferences, and the possibility to stay abroad, I want to thank the Karlsruhe School of Optics and Photonics (KSOP) and the Karlsruhe House of Young Scientists (KHYS).

Furthermore I want to thank the secretaries Sieglinde Klimesch and Erna Nagler for solving any administrative problem and keeping me from German bureaucracy, and also Werner Paal for providing an outstanding IT-infrastructure fixing problems at almost every time, and the men of our workshop for solving all hardware related problems.

I want to thank my mother and my late father for their never ending support, the possibility to study in a carefree manner, and always providing a home. And I thank my wife Sybille – I love you like the stars above.

Karlsruhe, in February 2015 *Philip Lenz*

ABSTRACT

Multi-target tracking is an important task in computer vision and its applications such as autonomous driving or sport statistics. Intelligent vehicles rely on information of surrounding traffic participants for path planning and collision avoidance. Increasingly, video cameras are employed, since they are cheap and give rich information on the surrounding. In particular in computer vision, multi-target tracking is still a very active research area and a challenging task. Nowadays, one of the most popular approaches to multi-target tracking is tracking-by-detection. Current solutions focus on solving the data association problem in a (near) optimal fashion and therefore typically in a batch setting [180, 22, 7]. Promising min-cost flow algorithms which solve the data association problem optimally have three main drawbacks: they are computationally expensive, they assume that the whole video is given as a batch and they scale badly in memory and computation with the length of the video sequence. For real-time demanding applications as autonomous driving, camera-based multi-target tracking still remains an unsolved problem. This thesis addresses each of these issues. Our first contribution is a dynamic version of the successive shortest-path algorithm, which solves the data association problem optimally while re-using computation and results in significantly faster inference than standard solvers. With our second contribution, we address the optimal solution to the data association problem when dealing with an incoming stream of data (i.e., online setting). Our last contribution to multi-target tracking is an approximate online solution with bounded memory and computation, which can handle videos of arbitrary length while performing tracking in real-time. Moreover, we introduce a scene flow-based 3D object detector providing another source of input data for our tracking algorithms. Furthermore, we introduce a challenging benchmark, recorded with our autonomous driving platform AnnieWAY to demonstrate the effectiveness of our algorithms showing state-of-the-art performance, while being significantly faster than existing solvers. Finally, we give an extensive evaluation of the proposed dataset and features for data association.

ZUSAMMENFASSUNG

Objektverfolgung für komplexe, reale Szenarien ist nach wie vor eine herausfordernde Aufgabe im Bereich der Bildverarbeitung und für darauf aufbauende Anwendungen wie autonomes Fahren oder die Erzeugung von Sportstatistiken. Autonom fahrende Fahrzeuge benötigen verlässliche Schätzung von Objekttrajektorien für sich daran anschließende Aufgaben wie Pfadplanung oder Kollisionsvermeidung. In den letzten Jahren wurde für diese Aufgabe zunehmend kostengünstige, video-basierte Sensorik verwendet. Objektverfolgung ist seit Jahrzehnten ein aktives Forschungsfeld und insbesondere im Bereich der Bildverarbeitung nach wie vor ein aktuelles und herausforderndes Problem. Moderne Methoden untersuchen den Datenassoziationsaspekt bei gegebenen Objektdetektionen und schätzen (approximierte) global-optimale Lösungen, für die komplette Sequenzen zur Optimierungszeit vorliegen [180, 22, 7]. Aktuelle Algorithmen zur Lösung des Minimum-Cost Flow Problems lösen das Assoziationsproblem optimal. Allerdings ergeben sich dadurch bei Anwendung für Objektverfolgungsszenarien mit Echtzeitanforderung folgende drei Nachteile: Die Komplexität ist unnötig groß, Sequenzen müssen zur Lösung komplett vorliegen und der Ressourcenverbrauch steigt unbegrenzt für längere Eingangssequenzen. Insbesondere für autonomes Fahren ist die kamerabasierte Objektverfolgung daher weiterhin ein aktiver, ungelöster Forschungsbereich. Diese Arbeit stellt Lösungen für alle drei genannten Problembereiche vor: Der erste Beitrag ist eine dynamische Erweiterung des *Successive Shortest Paths* Algorithmus zur optimalen Lösung des Assoziationsproblems. Der vorgeschlagene Algorithmus führt nur notwendige Berechnungen erneut aus und erzielt deutlich schneller Inferenzergebnisse im Vergleich zu Standardlösern. Dieser dynamisch Ansatz wird für ein Online-Szenario mit kontinuierlich abzuarbeitenden Eingangsdaten entsprechend erweitert. Im letzten Schritt kann darauf aufbauend eine approximative Lösung abgeleitet werden, die für das Assoziationsproblem unabhängig von der Sequenzlänge ist und damit für Echzeit-Anwendungen im Bereich des autonomen Fahrens nutzbar ist. Darüber hinaus stellt diese Arbeit einen generischen Objektdetektor auf Basis von Szenen-

flussmerkmalen vor, die als Eingang für die Objektverfolgung geeignet sind. Zur Evaluation der untersuchten Methodiken und Merkmale wurde ein anspruchsvoller Benchmark mit Hilfe des Versuchsträgers AnnieWAY erstellt. Die durchgeführten Experimente zeigen die Eignung der vorgestellten Methodik zum robusten Lösen des Assoziationsproblems für komplexe Objektverfolgungsszenarien. Dabei ist die Laufzeit geringer als bei vergleichbaren, aktuellen Ansätzen und im Falle des approximativen Algorithmus unabhängig von der Sequenzlänge. Abschließend werden die verwendeten Merkmale bewertet und die Eigenschaften des erstellten Datensatzes detailliert analysiert.

CONTENTS

NOTATION AND SYMBOLS

AKRONYMS

BF	Bellman-Ford
CHIL	Computers in the Human Interaction Loop
CLEAR	CLassification of Events, Activities and Relationships
DAG	Directed Acyclic Graph
DFS	Depth-First Search
DP	Dynamic Programming
FAR	False Alarm Rate per Frame
HOG	Histograms of Oriented Gradients
ICP	Iterative Closest Point
KSP	k-Shortest Paths
mbod	memory-bounded online dynamic (SSP)
MHG	Multiple Hypothesis Tracking
MOT	Multiple Object Tracking
MODA	Multiple Object Detection Accuracy
MOTA	Multiple Object Tracking Accuracy
MODP	Multiple Object Detection Precision
MOTP	Multiple Object Tracking Precision
ML	Mostly Lost
MT	Mostly Tracked
(J)PDA	(Joint) Probabilistic Data Association
PT	Partly Tracked
SSP	Successive Shortest Paths

RADAR	Radio Detection and Ranging
VACE	Video Analysis Content extraction
v.m.t.	vehicle miles traveled
UN	United Nations

GENERAL NOTATION

Scalars	Regular (greek) lower case	a, b, γ, δ
Image Coordinates	Sans serif lower case	u, v
Vectors	Bold (greek) lower case	$\mathbf{a}$, $\mathbf{b}$, $\boldsymbol{\gamma}$, $\boldsymbol{\delta}$
Matrices	Bold (greek) upper case	$\mathbf{A}$, $\mathbf{B}$, $\boldsymbol{\Gamma}$, $\boldsymbol{\Delta}$
Sets	Calligraphic upper case	$\mathcal{A}$, $\mathcal{B}$
Numbers	Regular blackboard	$\mathbb{N}$, $\mathbb{R}$
Functions	Regular lower case	$f(\cdot)$

INDEXING

i	First-order index $i \in \{1, \ldots, N\}$
j	Second-order index $j \in \{1, \ldots, N\}$
a_i	i-th element of vector $\mathbf{a}$
k	First-order trajectory index $k \in \{1, \ldots, N\}$
l	Second-order trajectory index $l \in \{1, \ldots, N\}$
en	Entry (edge)
ex	Exit (edge)
li	Link (edge)
det	Detection (edge)
L	Left image
R	Right image
$l_{\text{<entity>}}$	Length of entity
$n_{\text{<entity>}}$	Number of entities

NUMBERS

$\mathbb{R}$	Real numbers
$\mathbb{R}_0^+$	Positive real numbers (including zero)

NETWORK OPTIMIZATION

n_K	Number of trajectories
n_f	Number of time steps/frames
$\mathcal{O}(\cdot)$	Big O Notation
$\Psi(\cdot)$	Potential
$\emptyset$	Empty set
C	Cost for a graph edge with $C \in \mathbb{R}$
C'	Cost for a graph edge with $C \in \mathbb{R}_0^+$
$\mathcal{G}(\mathcal{X})$	Min-cost flow network
$\mathcal{G}_r(\mathcal{X})$	Residual min-cost flow network
Γ	Arborescence of the shortest path in $\mathcal{G}$
$c(\cdot,\cdot)$	Cost function
f	Binary flow for a graph edge
E	Edges in the graph
V	Vertices in the graph
s	Virtual source node
t	Virtual target node
u	First detection node
v	Second detection node
$\mathcal{X}$	Set of detections
T	Trajectory
$\mathcal{T}$	Set of trajectories
γ	Shortest path in a min-cost flow network
π	Predecessor map

ρ	Distance map
$\mathcal{C}$	Optimization cache
δ	Cache time step
τ	Optimization window length

IMAGE EVIDENCE

$\mathbf{x}$	Single detection
$\mathbf{u}$	2D position
t	Time step (frame index)
$\mathbf{s}$	2D object dimensions
α	Bounding box orientation
a	Bounding box appearance
d	Detector score
o	Object class label
$(w, h)^\intercal$	Object dimensions in image coordinates
$u, v^\intercal$	Image coordinates
g	Pixel value
$\mathbf{X}$	3D position
$\mathbf{S}$	3D object dimensions
$\mathbf{f}$	Scene flow
$[X, Y, Z]^\top$	World coordinates
$\mathbf{J}$	Jacobian matrix
$\boldsymbol{\Sigma}$	Covariance matrix
D	Mahalanobis Distance

1

INTRODUCTION

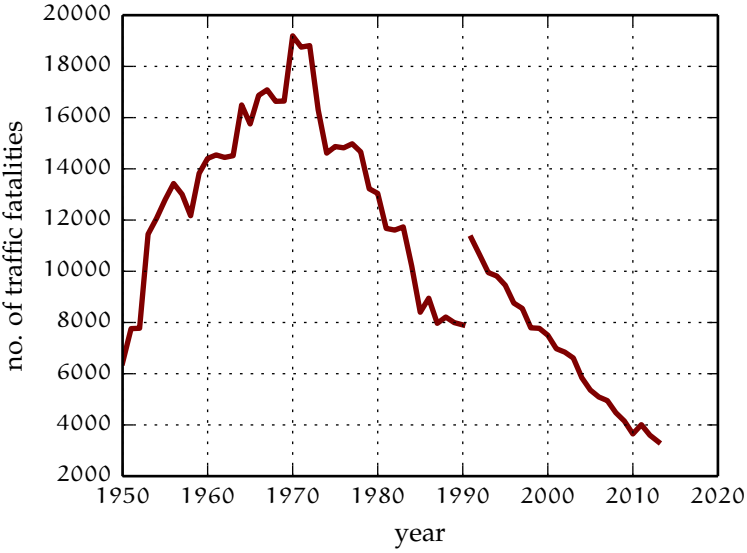
> *Everything in life is somewhere else, and you get there in a car.*
>
> ELWYN B. WHITE

Nowadays mobility for people and commodities is a key issue for the prosperity of our society. In consequence, by today there are more than 50 million vehicles manufactured worldwide every year in this millennium [92, 164].

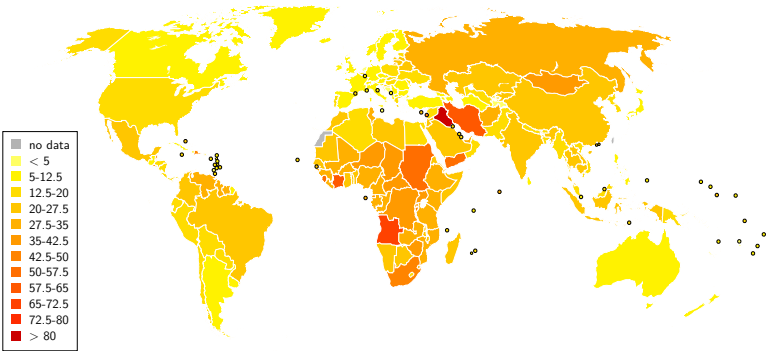
For this reason, there are various open tasks for future mobility: The number of road casualties has been decreasing for the last decades in countries with a high level of income only (Fig. 1.1(a), [168]), but is still a major cause of death and injury even there (Fig. 1.1(b)). Due to the increasing number of vehicles, people are wasting more and more time in traffic jams [91]. Urban planning must consider sufficient parking space and the increasing amount of exhaust gases [9].

The current *UN Decade of Action for Road Safety*[1] (2011-2020) is trying to reduce the expected number of road casualties at the end of this decade by 50% [93, 168]. The five pillars of the *Decade of Action* include improvement on road safety management, safer roads and road users, as well as post-crash response, and in particular *safer vehicles*.

1 http://www.un.org/en/roadsafety/



(a) Traffic Fatalities in Germany. [152]



(b) Number of Traffic Fatalities per 100,000 citizens. [169]

Figure 1.1: Road Safety and Traffic Fatalities. After the German government started pushing road safety in the 1970s, the total number of traffic fatalities decreased continuously (a). However, traffic fatalities are still one of the major causes of death worldwide (b).

Self-driving and interacting vehicles in their different manifestations will play a key role not only to further decrease the number of road casualties but also to exploit energy more efficiently and to avoid unnecessary traffic jams [172]. Furthermore, driverless vehicles can fulfill time-independent tasks at off-peak hours to unify the road network usage over the whole day further reducing traffic jams.

To fulfill these tasks fully autonomously, self-driving cars must perform trajectory planning and environmental perception in complex traffic scenarios, including non-autonomous traffic participants such as pedestrians, cyclists, or driver-operated cars. Consequently, robust environmental perception is still one of the fundamental open challenges in both autonomous driving and computer vision. In contrast, downstream applications such as trajectory planning already proved suitability given a decent perception input [98, 184].

This thesis presents an efficient method to track traffic participants over time for complex and cluttered real-world traffic scenarios. As input data, the proposed system does not rely on any specific sensor, although monoscopic and stereo camera input data is used for validation.

1.1 PROBLEM STATEMENT

Given a variable and unknown number of (dynamic) objects over time captured from an arbitrary (moving or static) sensor, multi-target tracking is the problem of associating these detections to form tracklets. Building on this, smoothing can be applied, but is not considered explicitly in this thesis. In particular, we are interested in video-based multi-target tracking for complex traffic scenarios as they are present in inner cities. Such scenarios are challenging due to interactions between objects, frequent occlusions, and an arbitrary scene geometry constituting an interesting problem for tracking. These challenges stem mainly from the problem constitution, having a moving observer and a flat observation angle.

For a successful multi-target tracking approach, the following questions should be answered:

- How many unique objects are present in the scene?
- Where do tracklets start and end?
- Which detections are outliers?
- How can occlusions be tackled?
- What are good features for data association?

The proposed approach solves the problems arising from these questions in a globally optimal way and is extended to achieve a solution for bounded memory and computational resources. We show, that using a cheap sensor set-up with a close-to-production monoscopic camera system and a state-of-the-art object detector already allows for good results. Furthermore, combining different kinds of object detections by extending the camera set-up to acquiring stereo information can be easily incorporated in the proposed system.

An overview of the processing stages is given in Fig. 1.2 showing the original scene, the detector output for the 2D case including false positives, and the finally associated detections.

1.2 APPLICATIONS

We emphasize three different areas of application, that rely on robust multi-target tracking:

AUTONOMOUS DRIVING: By now, autonomous driving is commonly divided into three categories [27, 70]: While *partially automated driving*, still requiring the driver to constantly monitor the traffic scenario, already starts to enter the end user market (e.g., traffic jam assistance), *highly* and *fully automated driving* (the former still requires a backup driver, the latter can be a driverless system) is still an active field of research [74, 98, 184].

Figure 1.2: MULTI-TARGET TRACKING STAGES: For the given input data (top), an object detector returns different types of traffic participants (middle), and object tracks are established for the input sequence discarding outliers (bottom).

Such systems arouse the expectation to enhance private transport in terms of safety, efficiency, as well as energy and resource consumption while making traveling and commuting more comfortable. Furthermore, car sharing can be more efficient since cars can be autonomously distributed on demand. Considerations merging private and public transportation in a similar manner already existed decades ago in the 1970s [17].

Considering benefits in terms of safety aspects, end users must be able to bear the costs of such systems. Taking into account that the number of road casualties is especially high and increasing for emerging countries [168], utilizing low-cost sensors is desirable. Current autonomous systems mainly rely on particular and pricey sensor systems [165]. However, developers start to utilize more close-to production sensors for highly automated driving [184].

SPORTS: Especially for team sports, organizers and television broadcasters provide wide statistics for the audience. Nowadays, these statistics (e.g., ball contacts, running stats, duels) are largely computed automatically. Therefore, individual players must be tracked during a game. This is a challenging task especially due to highly dynamic and holonomic motion. Furthermore, players have a uniform appearance and usually converge to a single target. Commonly, TV cameras with a relatively flat observation angle are used and therefore occlusions must be considered as well [22].

BIOLOGY: Since data in biology has grown tremendously in recent years, manual analysis cannot satisfy current research demands and automated approaches are essential. In particular, live-cell imaging experiments heavily rely on tracking algorithms to take advantage of time-lapse microscopic images [119].

1.3 CONTRIBUTIONS

This thesis addresses several aspects of two fundamental challenges of multi-target tracking:

DATA ASSOCIATION:

- We develop an efficient dynamic method to solve multi-target tracking formulated as a minimum-cost flow problem for batch processing in a globally optimal manner.
- Considering online scenarios (i.e., continuously arriving measurements), the proposed solver is extended in a way, which allows integration of novel frames in a previously computed solution instead of performing a complete re-computation.
- For handling input sequences of an arbitrary length, we derive an approximative solver that overcomes the limitation of unbounded growth of existing min-cost flow approaches in memory and computation without temporal delay.

EVALUATION:

- The average-case complexity of min-cost flow tracking is compared for the proposed methods against existing state-of-the-art approaches.
- An extensive evaluation is performed on 50 challenging real-world sequences to demonstrate the tracking performance of the proposed methods. Despite evaluation on this introduced dataset, we present results on another state-of-the-art benchmark which is used in the long-term.
- Performance of association features and metrics is evaluated on the proposed benchmark.

1.4 OUTLINE

The following chapters are structured as follows: Chapter 2 reviews the state-of-the-art while delimiting the contributions of this thesis with respect to previous work. Chapter 3 states the problem formulation of multi-target tracking formulated as a min-cost flow optimization problem. Chapter 4 outlines the proposed methods for solving this problem in an efficient, dynamic manner and how the solution can be extended to tackle online

scenarios. Furthermore, an approximative solution to handle sequences of an arbitrary length is presented. Image evidence, describing the detections used as input data as well as association features are discussed in Chapter 5. Additionally, this chapter introduces our scene flow-based 3D object detector to generically detect moving traffic participants. Details on the experimental platform, the collected data, the proposed method, and experimental results are given in Chapter 6. Finally, conclusions are drawn in Chapter 7. An overview of related network optimization algorithms is given in Appendix A.

2

RELATED WORK

> *To study and not think is a waste. To think and not study is dangerous.*
>
> CONFUCIUS

In this section, we review related work for multi-target tracking. First, we discuss the state-of-the-art in computer vision and position the contributions of this thesis. Secondly, we discuss the currently used benchmarks and metrics for vision-based multi-target tracking. Thirdly, a summary of the current challenges for different applications requiring to solve a multi-target tracking problem is given.

2.1 MULTI-TARGET TRACKING IN COMPUTER VISION

Despite decades of research, robust multi-target tracking remains one of the fundamental challenges in computer vision and for its applications such as autonomous driving. This chapter starts by reviewing the development of object tracking in general. Tracking-by-detection has proven as one of the most successful strategies in recent years, hence the state-of-the-art in object detection is summarized before discussing recent methods for data association. Since there is typically higher level knowledge for particular problem scenarios, we give an additional overview on how this can be exploited to increase tracking performance.

2.1.1 *Target Tracking*

Tracking of arbitrary targets of an unknown type has been persistently pursued since the 1930s for different sensors and applications [3, 29, 94, 97, 102, 148]. In computer vision, it remains one of the fundamental open challenges [7, 8, 43, 60, 108, 180]. We give an overview of the history of tracking in Fig. 2.1, outline the history shortly in this chapter, finally giving an extensive overview of the state-of-the-art.

The pioneer work of Heinrich Hertz in the 1880s of generating and detecting radio waves led to the first RADAR applications in the 1930s. Until 1950, tracking algorithms to detect moving targets within large clutter reflected from land and sea were built into custom hardware for analogous signal processing. Digital signal processing for such systems was developed starting in the 1950s while most effort was made for military purposes [29, 148]. Inevitably, multi-target tracking (even in a multi-sensor environment) became an active topic of research in the 1970s, already considering non-linearities, probabilistic data association, and exploiting parallel computational structures. The resulting algorithms can be generally classified as filtering-based approaches. Exploiting the Markov assumption, i.e., the current state depends only on the previous state, these algorithms were already fast and real-time applicable for RADAR scenarios [3, 16, 97, 139, 150].

In the early days of computer vision, low-level tasks, e.g., edge detection, optical flow estimation, or image segmentation were the primary focus of research. In the 1990s, subsequent tasks based on geometric image interpretation started to be of particular interest, including visual object tracking. Hence, existing filtering-based approaches were applied to computer vision. However, they typically suffer from their inability to recover from early errors, which is particularly problematic in complex scenarios while applied to noisy sensor data. Primary, approaches for arbitrary scenes typically used motion and gray level information for tracking such as Burt et al. [40] in surveillance-based scenarios to establish and validate object tracks. Koller et al. [101] already considered fundamental multi-target tracking issues such as occlusions. Furthermore, a moving observer was discussed, e.g.,

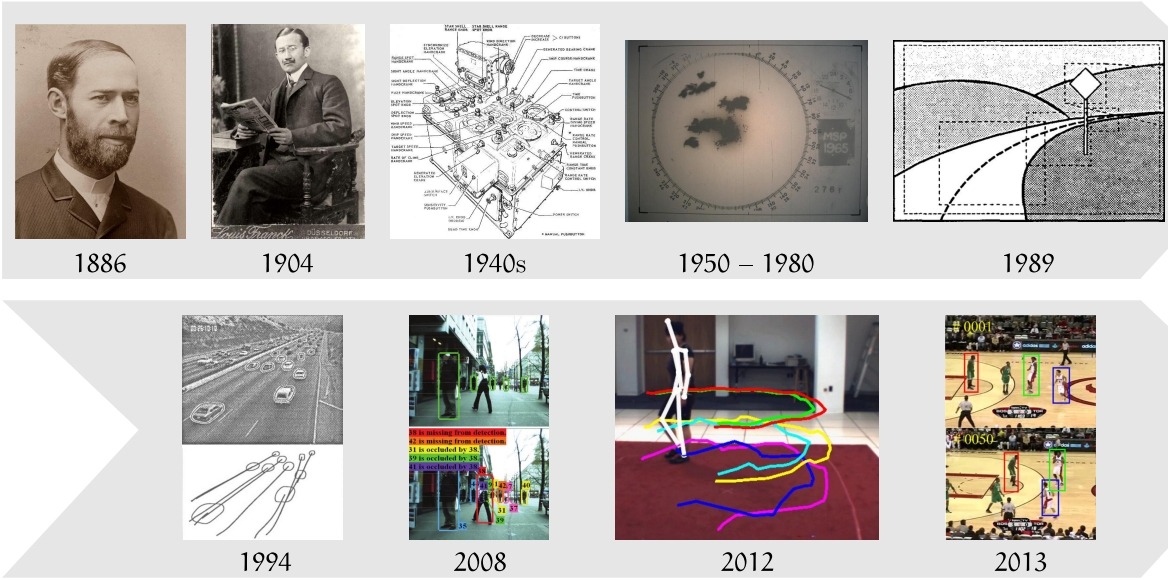


Figure 2.1: History of Tracking. Starting in 1886, Heinrich Hertz showed the reflectance of radio waves and Christian Hülsmeyer successfully exploited this principle to detect metallic objects. Up to the 1940s, customized, analogous hardware was used for RADAR systems, also performing target tracking. Starting in the 1950s, digital systems were built, broadening the scope of RADAR beyond military systems, exploiting weather and civil air traffic applications. Early tracking approaches in computer vision used optical flow or background subtraction to establish object tracks. Current research focuses on tracking-by-detection, explicitly considering occlusions, target interaction, and group behavior. Pictures not under public domain are taken from [40, 102, 180, 108, 181].

by Dickmanns and Gräfe [51] and Papanikolopoulos et al. [136]. Contour tracking [94, 102] was a common approach for object tracking and was applied to cluttered sequences. Isard and Blake [94] extended unimodal filtering, e.g., [51], to represent simultaneous alternative tracking hypotheses while using learned dynamic models for increasing robustness in cluttered environments.

Early approaches in computer vision typically needed to tackle a low signal-to-noise ratio. Therefore, tracking was performed on raw sensor data (e.g., gray-values, optical flow information) to reduce information loss caused by e.g., an object detection step. Depending on the application, detection is performed subsequently of the tracking step. Avoiding a detection step and executing tracking on the raw sensor data directly is known as *track-before-detect* in the literature. [143] Typically, Bayesian tracking is used for this approach [30, 143] as shortly outlined in the following section or dynamic programming is used as an efficient solution to this problem [143, 158].

Current research still employs filtering-based methods, where, in contrast to work in the 1990s, object hypotheses are generated by a class-specific object detector providing a higher signal-to-noise ratio [35, 60, 122]. This approach, known as *tracking-by-detection*, has become the primarily used approach for object tracking in computer vision [5, 34, 72, 109, 170, 180], taking the place of background subtraction as summarized in [178]. We shortly review the state-of-the-art for object detection in Section 2.1.3.

Nevertheless, recent efforts focus primarily on *batch methods*, generating object hypotheses by a tracking-by-detection approach and formulating tracking as an optimization problem for a completely acquired sequence [7, 44, 180]. This mitigates the aforementioned problems, since all measurements are optimized jointly, rejecting outliers more robustly and fixing occlusions by incorporating higher-order knowledge.

2.1.2 *Data Association for Multi-Target Tracking*

Filtering-based approaches for tracking typically handle single targets [14, 94, 97]. Starting with the development of RADAR,

associating current measurements (so-called plot association) to existing tracks was already a key issue for such systems, even in the presence of only a single (wanted) target but in an environment with high clutter [3, 16]. In this thesis, we refer to this task as *data association* for multi-target tracking.

The simplest approach to fulfill this task is a nearest-neighbor association, which links the closest observation to a track in a greedy manner. This results obviously in a non-optimal data association, but was used in productive systems for a long period of time [28]. Solving the given problem, associating n tracks to m observations in the current frame, can be solved optimally by the Hungarian algorithm [2, 106, 125]. All the aforementioned approaches only perform a frame-to-frame association, taking further information on the tracks only indirectly into account by associating detections to tracks predicted by the applied filter. Furthermore, they only allow to form single data association hypotheses.

For complex scenarios, where interacting targets are present and violating underlying motion models, such a simple data association strategy may cause the used filter to fail. Already in the 1970s, probabilistic approaches were developed to tackle this problem [14, 16, 140]. An in-depth explanation is given in [15] and summarized in the following. *Probabilistic data association* (*PDA*) is a Bayesian approach for single target data association. PDA solves the problem of selecting measurements for a regular tracking filter, such as an (extended) Kalman filter [97], for updating the filter state. PDA considers a single, already initialized target assuming a given detection and false alarm probability. The state of the target $\mathbf{x} \in \mathbb{R}^n$ is evolved from the previous time step $k-1$ by

$$\mathbf{x}_k = \mathbf{F}_{k-1}\mathbf{x}_{k-1} + \mathbf{w}_{k-1} \tag{2.1}$$

exploiting the Markov assumption

$$p(\mathbf{x}_k \mid \mathbf{x}_0, \ldots, \mathbf{x}_{k-1}) = p(\mathbf{x}_k \mid \mathbf{x}_{k-1}) \tag{2.2}$$

for which the current state $\mathbf{x}_k$ is conditionally independent of all earlier states which are contained in the given previous state $\mathbf{x}_{k-1}$.

For the target, a true measurement $\hat{\mathbf{z}}_k \in \mathbb{R}^m$ is given by

$$\hat{\mathbf{z}}_{k|k-1} = \mathbf{H}_k \hat{\mathbf{x}}_{k|k-1} + \mathbf{v}_k \tag{2.3}$$

Both noise terms $\mathbf{w}_{k-1}$ and $\mathbf{v}_k$ are assumed to be zero-mean mutually independent white Gaussian noise. The covariances are given by $\mathbf{w}_{k-1} \sim N(0, \mathbf{Q}_{k-1})$ and $\mathbf{v}_k \sim N(0, \mathbf{R}_k)$ respectively.

The state vector and its covariance at time $k-1$ is predicted for the most recent time step k by

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{F}_{k-1} \hat{\mathbf{x}}_{k-1|k-1} \tag{2.4}$$

$$\mathbf{P}_{k|k-1} = \mathbf{F}_{k-1} \mathbf{P}_{k-1|k-1} \mathbf{F}_{k-1}^T + \mathbf{Q}_{k-1}. \tag{2.5}$$

With Eq. 2.3 an elliptical validation region $\mathcal{V}$ can be defined. Within this validation region, measurements made by the sensor can be associated with the target of interest. This region can be set up to guarantee that the correct measurement is within this region with high probability. Ultimately, the PDA results in association probabilities depending on the location of the measurement as outlined below. The validation region is given by

$$\mathcal{V} = \{\mathbf{z} : [\mathbf{z} - \hat{\mathbf{z}}_{k|k-1}]^\top \mathbf{S}_k^{-1} [\mathbf{z} - \hat{\mathbf{z}}_{k|k-1}] \leqslant \gamma\} \tag{2.6}$$

with residual covariance

$$\mathbf{S}_k = \mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k^\top + \mathbf{R}_k \tag{2.7}$$

where γ is the threshold corresponding to the probability that the region contains the true measurement (if it was detected).

Using a Poisson clutter model for the parametric case, the association probability for $\mathbf{z}_k^{(i)}$ being the correct measurement ($i > 0$) is given by

$$\beta_k^{(i)} = \begin{cases} \frac{\mathcal{L}_k^{(i)}}{1 - P_D P_G + \sum_{j=1}^{m_k} \mathcal{L}_k^{(j)}}, & i = 1, \ldots, m_k \\ \frac{1 - P_D P_G}{1 - P_D P_G + \sum_{j=1}^{m_k} \mathcal{L}_k^{(j)}}, & i = 0 \end{cases} \tag{2.8}$$

for m_k validated measurements and with the likelihood ratio $\mathcal{L}_k^{(i)}$ of the measurement $\mathbf{z}_k^{(i)}$

$$\mathcal{L}_k^{(i)} \triangleq \frac{\mathcal{N}(\mathbf{z}_k^{(i)}; \hat{\mathbf{z}}_{k|k-1}, \mathbf{S}_k)\ P_D}{\lambda} \tag{2.9}$$

being the true target instead of clutter, P_D the probability for detecting a target, and P_G the probability for the validation region containing the true measurement.

After predicting the state $\mathbf{x}_{k|k-1}$, validating and associating the measurement, the final state is estimated by

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{W}_k \tilde{\mathbf{y}}_k \tag{2.10}$$

with the combined innovation

$$\nu_k = \sum_{i=1}^{m_k} \beta_k^{(i)} \nu_k^{(i)} \tag{2.11}$$

and the gain

$$\mathbf{W}_k = \mathbf{P}_{k|k-1} \mathbf{H}_k^\top \mathbf{S}_k^{-1}. \tag{2.12}$$

The estimated covariance is obtained by

$$\mathbf{P}_{k|k} = \beta_k^{(0)} \mathbf{P}_{k|k-1} + [1 - \beta_k^{(0)}] \mathbf{P}_{k|k}^c + \tilde{\mathbf{P}}_k \tag{2.13}$$

with

$$\mathbf{P}_{k|k}^c = \mathbf{P}_{k|k-1} - \mathbf{W}_k \mathbf{S}_k \mathbf{W}_k^\top \tag{2.14}$$

and

$$\tilde{\mathbf{P}}_k = \mathbf{W}_k \left[\sum_{i=1}^{m_k} \beta_k^{(i)} \nu_k^{(i)} \nu_k^{(i)\top} - \nu_k \nu_k^\top \right] \mathbf{W}_k^\top. \tag{2.15}$$

PDA can be extended to handle an already known number of multiple targets. This approach is known as *joint probabilistic data association* (*JPDA*). Each target has an (individual) dynamic and measurement model and states are assumed to be Gaussian distributed. However, target measurements may be overlapping and cannot be unambiguously associated. Association probabilities are computed jointly across the targets. JPDA evaluates the conditional probabilities joint association

$$\mathbf{A}_k = \bigcap_{j=1}^{n_m} A_k^{(jt_j)} \tag{2.16}$$

at the current time step k with $A_k^{(jt_j)}$ indicating that measurement j is originated from target t for a total of n_t targets and n_m measurements. Modeling the number of false alarms ϕ as a Poisson distribution in the parametric case, the probability mass function is given by

$$\mu_F(\phi) = \exp^{-\lambda V} \frac{(\lambda V)^\phi}{\phi!} \tag{2.17}$$

with the volume V_k of the validation region for an n_z dimensional unit hypersphere

$$V_k = c_{n_z} |\gamma \mathbf{S}_k|^{\frac{1}{2}} = c_{n_z} \gamma^{\frac{n_z}{2}} |\mathbf{S}_k|^{\frac{1}{2}} \tag{2.18}$$

resulting in the joint association probabilities

$$P\{A_k \mid Z^k\} = \frac{1}{c_2} \prod_j \{\lambda^{-1} f_{t_j}[z_k^{(j)}]\}^{\tau_j} \prod_t (P_D^t)^{\delta_t} (1 - P_d)^{1-\delta_t} \tag{2.19}$$

where

$$f_{t_j}(z_k^{(j)}) = \mathcal{N}(z_k^{(j)}; \hat{z}_{k|k-1}^{(t_j)}, \mathbf{S}_k^{(t_j)}) \tag{2.20}$$

and $\hat{z}_{k|k-1}^{(t_j)}$ is the predicted measurement for target t_j with its covariance $\mathbf{P}_k^{(t_j)}$ after the innovation. P_D^t is the probability of detecting target t, whereas τ_j and δ_t are the detection and measurement indicators.

For the decoupled case with mutually independent past observations, the marginal probabilities of the target states are obtained by summing over all joint events resulting in

$$\beta_k^{jt} \triangleq P\{A_k^{(jt)} \mid Z^k\} = \sum_{A : A^{(jt)} \in A} P\{A_k \mid Z^k\}. \tag{2.21}$$

Consequently, the state estimation is done as for the regular PDA.

Both approaches approximate the conditional probability density function of the state using a single Gaussian and past associations must not be re-computed since the information state is (approximately) given by the most recent time step. In particular,

these approaches showed good results for low-noise sensor data (e.g., RADAR) and relatively few targets.

Generally, the optimal Bayesian solution evaluates the probability of all association hypotheses. This approach is known as the *multi hypotheses tracker* (*MHT*). Assuming white noise for mutually independent process and measurement sequences as well as white noise for false measurements, the probability density function (pdf) is given by

$$p_k = p(\mathbf{x}_k \mid I^k) \tag{2.22}$$

with the information set $I^k = \{\{z(j)\}_{j-1}^k, U^{k-1}\}$ for all observations through time k (subsumming the initial information Z_0) and all known inputs U^{k-1} prior to time k. These assumptions are sufficient for p_k being an information state. This allows to estimate the trajectories of multiple objects in the presence of clutter as sets of associated measurements.

The state p_k can be recursively updated by

$$p_{k+1} = \psi_{k+1}(p_k, \mathbf{z}_{k+1}, u_k) \tag{2.23}$$

where

$$\begin{aligned}
&\psi_{k+1}(p_k, \mathbf{z}_{k+1}, u_k) = \\
&\frac{1}{c} \sum_{i=1}^{n_A(k+1)} p(\mathbf{z}_{k+1} \mid \mathbf{x}_{k+1}, A_{k+1}^{(i)}) \\
&\times \int p(\mathbf{x}_{k+1} \mid \mathbf{x}_k, u(k)) p_k \, d\mathbf{x}_k \\
&\times P\{A_{k+1}^{(i)}\}
\end{aligned} \tag{2.24}$$

with $n_A(k+1)$ mutually exclusive and exhaustive association events $A_{k+1}^{(i)}$ at time $k+1$ and the normalization constant c. This optimal estimator results in an exponentially growing number of terms over time for the sum in Eq. 2.24 introduced by the association events, represented in the MHT hypothesis tree, encoding all possible trajectories. However, this results in increased com-

putational resources in orders of magnitude for solving the association problem and a far more complex implementation. Consequently, real-world multi hypotheses tracking approaches apply pruning and support typically only the k-best solutions.

To further facilitate computations, branching is applied [140] or the probability density is approximated [16]. These approaches were extended in several ways in recent years [13, 68, 127, 128], handling multi-modal track hypotheses, but still resulting in suboptimal solutions using a heuristical track management. An overview and evaluation of MHT and PDA approaches is given in [28, 178].

Considering multi-target tracking in a batch setting, data association should consequently be formulated as a multi-frame problem, e.g., in [180]. Recent approaches perform this *multi-frame data association* following different approaches. Markov chain Monte Carlo data association is employed in [21, 33, 43]. For a statically observing camera the problem can be casted into a relatively complex linear program, while optimization is performed for detections represented on a discrete grid [19, 22, 89].

The problem of tracking through occlusions has been tackled in [90, 112, 174, 173] by using context from outside the object region and by building strong statistical motion models. Associating initial tracklets on a tracklet level using online-learned motion cues was also investigated by [176] and extended to online learned appearance models in [177]. Further online learning approaches have been suggested in order to increase discriminative power [108, 111, 151, 175, 179, 181]. Higher-level cues as group behavior for pedestrian detection to tackle highly complex, crowded scenarios were recently investigated by [44, 107].

While all of the aforementioned formulations resort to approximate optimization without optimality guarantees, Zhang et al. [180] showed in their seminal work how multi-frame data association can be cast as a network flow problem. This formulation solves for the globally optimal trajectories applying a min-cost flow algorithm, inherently solving the model selection problem, i.e., resulting in the number of trajectories and their respective start and end in time. Therefore a cost-function is created, which incorporates the likelihood of a detection as well as transition

probability between detections in two consecutive frames. Since this thesis is based upon this problem formulation, we give a detailed outline of min-cost flow data association and a general solution in Chapter 3.

2.1.3 *Object Detection*

In recent years, tracking-by-detection has proven as a successful strategy for multi-target tracking. Consequently, object detection is a key requirement for multi-target tracking. Beyond estimating tracklets, multi-target tracking results in improving object detection by exploiting knowledge over time. Therefore, "over-detection" is preferable in the initial detection stage. Nevertheless, object detection itself is one of the main challenges in computer vision. Since this thesis relies on detections as input data, we give a short outline of the current state-of-the-art.

In early days of computer vision, object hypotheses in the image plane for a subsequent classification were generated by exhaustive scanning [47, 135, 182, 183], using image cues such as color, intensity, gradients, symmetry [36, 95, 114, 157], or relied on stereo information for specialized applications, e.g., autonomous driving [69, 105, 122]. Although motion is an important cue for human perception, it is a challenging task to be applied to in computer vision [48, 55, 160]. More recent work used characteristic features for a specific object class such as shadow and symmetry features for cars. For associating resulting object hypotheses over time, approximative JPDA was used and the object state estimated using Kalman filtering for multiple motion models using an interacting multiple model filter method. [86, 87]

Primarily, state-of-the-art methods follow an appearance-based procedure for verifying generated hypotheses, where, in general, descriptors are evaluated by a (pre-trained) classifier. For this task, state-of-the-art approaches rely on adaptive boosting, support vector machine classification, or neural networks [1, 12, 32, 53, 54, 58, 65, 110, 116, 117, 123, 146, 161]. As descriptors, histograms of oriented gradients (HOG), Haar wavelets, edgelets, or shape context descriptors are commonly used [47, 71, 113, 135,

142, 171]. For a more detailed overview and an evaluation of the state-of-the-art we refer the reader to [55, 81].

Felzenszwalb et al. [65] proposed one of the best-performing algorithms for object detection in the PASCAL Visual Object Challenge [62]. Mixtures of multiscale deformable part models are used within the training procedure of a latent support vector machine. This allows for representing large variance within an object class.

By now, large datasets such as ImageNet [50] draw increasing interest for computer vision algorithms. Therefore, classification methods must handle an increasing amount of (partly unlabeled) training data and discriminate several hundred different classes. Krizhevsky et al. [104] showed on the ImageNet dataset, that for such problems, deep convolutional neural networks outperform current part-based approaches.

2.1.4 *Incorporating Domain Knowledge*

Multi-target tracking for complex and crowded scenes is still a challenging task. In such an environment, additional information such as motion or activity recognition can increase performance of a tracking system.

Filtering-based tracking methods typically employ motion models to increase robustness [5, 35, 60]. Similarly, rich motion models can be incorporated in the optimization problem formulation for batch processing, while a joint discrete-continuous optimization still allows for taking the whole sequence into account [7, 121], since pure continuous energy minimization is a difficult optimization task [6]. The discrete (data association) and continuous (trajectory estimation) optimization are performed iteratively until convergence to a (generally local) minimum. Extending the work of [180], higher-order terms for track smoothness were introduced by [41]. However, necessary constraints extend the problem, which cannot be solved in a min-cost flow sense anymore. To keep the problem tractable, the constraints are relaxed using Lagrangian relaxation while violating optimality.

Furthermore, problem specific knowledge, e.g., group behavior in the case of pedestrian tracking, can be incorporated in

the optimization problem by introducing higher-order relationships [44, 103, 107].

2.2 BENCHMARKING

Demanding benchmarks are a key requirement to close the gap between algorithm development and applicability for real-world scenarios. Diverse benchmarks are mandatory to avoid a bias to particular benchmarks. Furthermore, comparability requires a significant measure. In the past few years an increasing number of benchmarks have been developed to push forward the performance of visual recognitions systems, e.g., Caltech-101 [63], Middlebury for stereo [145] and optical flow [11] evaluation. However, most of these datasets are simplistic, e.g., are taken in a controlled environment. A notable exception is the PASCAL VOC challenge [62] for detection and segmentation. Summarizing the state-of-the-art, this section introduces existing benchmarks and currently used metrics for multi-target tracking.

2.2.1 *Benchmarks*

Several benchmarks to evaluate object detection were proposed mainly in the last decade [59, 62, 64, 124, 130, 133, 141]. An overview of the existing benchmarks is given in Table 2.1. The recent releases of the PASCAL Visual Object Challenge [62] are still the established state-of-the-art benchmarks for object detection and segmentation, although no future challenges will be released. Beyond the provided ground truth data, [62] established the commonly used *average precision* (AP) as a measure for evaluating and comparing object detection approaches. Summarizing, for computing AP, detections are iteratively assigned to ground truth labels starting with the largest overlap. The overlap criterion is measured by bounding box intersection over union. Typically, a minimum overlap of 50% is required for a true positive and multiple detections of the same object are considered as false positives. AP is defined as the area under the resulting curve for precision and recall.

Object Detection / 3D Estimation	#categories	avg. #labels/category	occlusion labels	3D labels	orientations
Caltech 101 [63]	101	40-800			
MIT StreetScenes [25]	9	3,000			
LabelMe [141]	3997	60			
ETHZ Pedestrian [59]	1	12,000			
PASCAL 2011 [62]	20	1,150	✓		
Daimler [100]	1	56,000	✓		
Caltech Pedestrian [55]	1	350,000	✓		
COIL-100 [130]	100	72		✓	72 bins
EPFL Multi-View Car [133]	20	90		✓	90 bins
Caltech 3D Objects [124]	100	144		✓	144 bins

Table 2.1: Object Detection: Comparison of current State-of-the-Art Benchmarks and Datasets. Most benchmarks provide multi-class labeling, partly for a very wide variety of object classes. Higher-level information such as occlusion, 3D, or orientation labels are only available for some of the datasets.

In contrast, there is no single ranking criterion for multi-target tracking (Section 2.2.2). However, a systematic evaluation of multi-target tracking approaches in computer vision has drawn increasing attention in recent years [42, 67, 131, 162, 185]. Several benchmarks were created, covering surveillance [56, 67, 149] and traffic scenarios [4, 61]. Additional ground truth for the primarily used benchmarks is provided by [7, 175]. A comprehensive comparison of the existing benchmarks is given in Table 2.2 and commonly used metrics are described in the following section.

2.2.2 *Tracking Metrics*

In computer vision, there are currently two common sets of metrics used for benchmarking. An illustrative overview of these metrics is given in Fig. 2.2. Bernardin and Stieflhagen [23] introduce the *CLEAR MOT* metrics by discussing necessary demands for evaluating multi-target tracking and consequently derive a set of metrics. CLEAR MOT unifies the previously existing metrics introduced by the VACE [42] and CHIL [162] benchmarks. On the one hand, CLEAR MOT takes the precision of the estimated tracklets with respect to a ground truth into account, normalized by number of objects (*Multiple Object Tracking Precision* (MOTP)) or number of frames (*Multiple Object Detection Precision* (MODP)). On the other hand, missed targets, false positives, and identity switches are evaluated with respect to the number of ground truth targets (*Multiple Object Tracking Accuracy* (MOTA)).

$$\text{MOTP} = \frac{\sum_{i,t} |d_t^i|}{\sum_t c_t} = \frac{\text{total position error}}{\text{no. of matches}} \tag{2.25}$$

$$\begin{aligned}\text{MOTA} &= 1 - \frac{\sum_t (m_t + fp_t + mme_t)}{\sum_t g_t} \\ &= 1 - \frac{\text{no. of misses} + \text{false positives} + \text{mismatches}}{\text{no. of objects in all frames}}\end{aligned} \tag{2.26}$$

Li et al. [111] proposed a set of metrics, clustering tracked trajectories according to the tracked length. Tracklet (hypotheses) and (ground truth) trajectories are associated frame-based

Object Tracking	#categories	avg. duration	avg. #labels/frames	setting	#sequences	moving camera
PETS 2009 [56, 7]	1	40s	37 151/2 258	outdoor	8	
TUD [4, 7]	1	30s	2 617/450	outdoor	3	
ETHMS [61, 7]	1	1′	12 928/2 289	outdoor	3	✓
TrecVid [149, 175]	1	3.5′	665,114/90 000	indoor	18	
ETISEO [131]	3	2′	50 5094/66 627	in-/outdoor	20	
CAVIAR [67]	1	1′	206 515/62 560	indoor	54	

Table 2.2: Object Tracking: Comparison of current State-of-the-Art Benchmarks and Datasets. All benchmarks provide 2D bounding boxes without occlusion labeling. Some datasets contain additional annotations describing the object, information on event recognition, or group behavior. Except the ETHMS benchmark, all scenarios focus on surveillance tasks, e.g., airport or shopping center setups.

by evaluating the bounding box overlap. Depending on the overlap of tracklets and trajectories in time, associations are counted as *mostly tracked* (MT), if the trajectory is covered for more than 80% by a tracklet. Trajectories tracked for less than 20% are considered as *mostly lost* (ML) and remaining trajectories as *partly tracked* (PT). Furthermore, id-switches and fragmentations are counted for all trajectories, according to the definition in Fig. 2.2. Beyond these measures on a trajectory level, precision, recall, and *false alarm rat*e (FAR) are evaluated on a frame-based level. An overview of the definitions is given in Table 2.3

$$\mathrm{MT} = \frac{\{n_t \mid l_t > 0.8 l_{gt}\}}{n_{gt}} \tag{2.27}$$

$$\mathrm{ML} = \frac{\{n_t \mid l_t < 0.2 l_{gt}\}}{n_{gt}} \tag{2.28}$$

$$\mathrm{PT} = 1.0 - \mathrm{MT} - \mathrm{ML} \tag{2.29}$$

$$\text{Fragments} = \sum \text{trajectory interruptions} \tag{2.30}$$

$$\text{Id-Switches} = \sum \text{trajectory identity changes} \tag{2.31}$$

Nghiem et al. [131] proposed three metrics on tracking level: *Tracking time* evaluates the normalized time a trajectory is tracked. In contrast, *object ID persistence* counts the number of tracker hypotheses associated to a trajectory. To penalize a low recall, *object ID confusion* computes the number of trajectories per tracker hypothesis.

Individually, all the aforementioned measures are not suitable as a ranking criterion. Therefore, comparing and judging tracking approaches quantitatively may lead to ambiguous assessments, also depending on the considered application.

2.3 MULTI-TARGET TRACKING APPLICATIONS

Nowadays, better and cheaper sensors as well as increasing computational power and memory lead to an unseen data explosion. To cope with and take advantage of such an amount of data, automated processing is essential for many different application

Name	Definition
MT [%]	MOSTLY TRACKED: Percentage of GT trajectories which are covered by tracker output for more than 80% in length.
ML [%]	MOSTLY LOST: Percentage of GT trajectories which are covered by tracker output for less than 20% in length. The smaller the better.
PT [%]	PARTIALLY TRACKED: 100% − MT − ML.
GT	No. of ground truth trajectories.
Frag	FRAGMENTS: The total of No. of times that a ground truth trajectory is interrupted in tracking result. The smaller the better.
IDS	ID SWITCHES: The total of No. of times that a tracked trajectory changes its matched GT identity. The smaller the better.
Recall	(Frame-based) correctly matched objects / total ground truth objects.
Precision	(Frame-based) correctly matched objects / total output objects.
FA/Frm	(Frame-based) No. of false alarms per frame. The smaller the better.

Table 2.3: Metrics proposed by and definitions taken from [111].

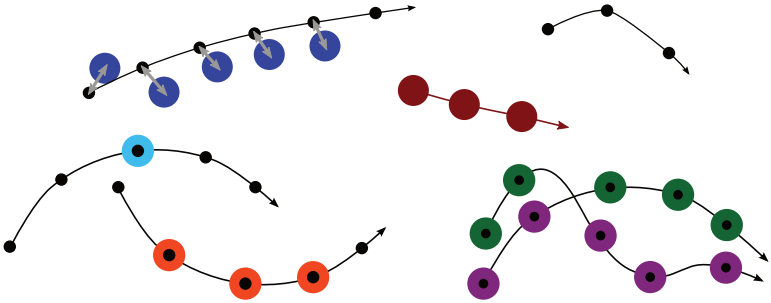


Figure 2.2: STATE-OF-THE-ART TRACKING METRICS. Ground truth trajectories (black) are associated to tracker hypotheses (larger, colored circles). MOTP evaluates the precision of the tracker (gray arrows). MOTA summarizes false positives (red hypothesis), false negatives (untracked trajectory, top right) and id-switches (green and purple hypotheses) in one measure. Furthermore, MT (blue), ML (cyan), and PT (orange) are evaluated. The tracker hypotheses at the bottom (right) result in two id-switches and two fragmentations. The examples are taken from the metric definitions in [23, 111].

areas. Typically, time series are of particular interest, and consequently multi-target tracking is a fundamental task to be solved. This chapter discusses applications on autonomous driving, sport statistics, and biology taking advantage of multi-target tracking and positions global methods within the particular areas.

2.3.1 *Autonomous Driving*

Autonomously driving vehicles have been targeted continuously in research for several decades, but fully automated driving is still an open challenge. First successful results were shown in the 1980s, already using camera-based environmental perception [51, 52]. Increasing levels of difficulty such as suburban scenarios, vehicle-to-vehicle communication, and challenging real-traffic scenarios were considered in the following years [39, 74, 98, 165, 184]. These approaches primarily rely on particular high-end sensor systems or accurate pre-built maps, whereas visual sensors are rarely exploited. However, recent systems increasingly integrate camera-based sensor systems to take advantage of a rich visual description of the scene and mass production for cheaper sensors.

Environmental perception for autonomously driving vehicles is mandatory for a robust overall system. By now, higher-level scene understanding draws an increasing interest for this application area [38, 75, 167]. From the outset, multi-target tracking is considered as a key task and was initially performed using visual sensors exploiting monocular and stereo information for object detection, while multi-target tracking was performed relying on Kalman filter-based approaches [52, 69]. These approaches did not consider a robust data association in a cluttered environment and widely relied on the quality of the given input data. In the meantime, fully autonomously driving systems and driver assistance systems widely rely on RADAR or laser sensor systems for car detection [98, 166], where vision based approaches increasingly draw attention especially for pedestrian detection [10, 20, 36, 37, 57, 81].

For multi-target tracking, autonomous driving requires real-time capable algorithms for frame rates of the sensor system,

which are typically 10fps and above. Therefore, existing approaches for multi-target tracking to the best of our knowledge solely rely on filtering-based approaches using local data association. Consequently, real-world traffic scenarios are typically tackled by applying multi-hypotheses tracking approaches and probabilistic data association [126, 153]. However, none of the existing approaches directly takes advantage of the measurement history to overcome heuristically determined track instantiation, target number estimation, and reduce approximation errors.

2.3.2 *Sports*

Team sports have always relied on evaluating games and single players or actions. Box scores are a common feature for several sports (e.g., basketball, football, and baseball) for which Henry Chadwick originally invented the modern box scores [129]. Today, box scores are much more informative and assistant coaches created additional statistics for their particular team for a long period of time. Such statistics are used as additional information in the stadium, for television broadcasting, and as a basis of decision-making for team coaches. Beyond that emerge decision theory-based algorithms, e.g., for football a system called *ZEUS*[1] to judge critical play choices.

Increasingly, automated vision based systems are replacing hand-compiled statistics across different team sports. The teams of the US basketball league widely use a commercial system called *SportVU*[2], which relies solely on setting up several cameras to monitor the playing field. In the meantime, the German *Bundesliga* introduced a commercial system called *VIS.TRACK*[3] for creating statistics, utilizing only two cameras. Basically, the derived statistics from the captured data can be divided into two categories: positional information and activity recognition. While positional information primarily requires tracking of each player individually during the game, activity recognition additionally

1 http://vimassgroup.com
2 http://www.stats.com
3 http://www.bundesliga-datenbank.de

needs to track the ball and detect interactions between players of the same or opposing team.

Basketball (NBA) turned out to be a relatively easy task for vision based evaluation, since there are only 11 moving targets, typically moving constantly. Consequently, research projects focus on this task as the Swiss *ARAMIS* project. Recently, filtering-based and batch processing methods for online scenarios were proposed for sport applications. Since this application requires results promptly (but not in real-time), multi hypotheses filtering-based approaches focus on efficient multi-target tracking considering noisy input data [34]. Batch methods relax the global formulation to obtain a rapid solution by pruning unlikely associations [18] or processing sub-sequences and linking results [22].

2.3.3 *Biology*

Research in medicine and biology is still investigating living organisms. Time-lapse microscopy is a key approach to analyze anatomic and dynamic properties. Currently, live cell imaging is widely used, creating a large amount of data. Inevitably, automated tools take over the place of manual inspection. One key requirement of such systems is tracking a variable and large number of cells and estimating their dynamic behavior. However, taking advantage of sophisticated tracking algorithms was primarily initiated in the past decade [118].

Generally, the focus of biology applications is tracking both cells and particles. Where cell tracking mainly focuses on a robust cell segmentation and solves data association with a simple nearest-neighbor approach, particle tracking is subject to cluttered and noisy input data, e.g., due to poor contrast, where data association is more challenging [119]. Solving the model selection problem, tackling occlusions, and particle splitting and merging cannot be handled by a simple nearest-neighbor approach anymore. Since tracking-by-detection is commonly used, both filtering-based multi-target tracking approaches [45, 82, 99] and batch methods are investigated [31, 96, 144] including min-cost flow based data association for cell and particle tracking [26, 134].

3

TRACKING-BY-DETECTION

> *Elegance is not a dispensable luxury but a quality that decides between success and failure.*
>
> EDSGER W. DIJKSTRA

The contributions of this thesis to solve multi-target tracking are focused on linking individual detections to form tracklets. Reviewing existing approaches, we consider the seminal work of Zhang et al. [180] as most promising to tackle the challenges of multi-target tracking as discussed previously (Section 1.1).

One of the most popular approaches to multi-target tracking is tracking-by-detection, where a set of detections is computed for each frame and trajectories are formed by linking the detections as discussed in Section 2.1. This chapter reviews the formulation of multi-target tracking as a min-cost flow problem, the originally introduced solution [180], and a faster algorithmic approach proposed recently [22, 138].

3.1 PROBLEM FORMULATION

Min-cost flow multi-target tracking follows a tracking-by-detection strategy, representing an available set of detections as $\mathcal{X} = \{\mathbf{x}_i\}$. Let $\mathbf{x}_i = (\mathbf{u}_i, t_i, a_i, d_i)$ denote a detection, with $\mathbf{u}_i$

the position of the bounding box, t_i the time step (frame index), a_i the appearance, and d_i a score returned by the detector. We define a trajectory as a sequence of observations $T_k = (o_1, o_2, \ldots, o_{l_k})$ with $o \in \{1, \ldots, |\mathcal{X}|\}$ the detection index of temporally adjacent detections $t_{o_{i+1}} = t_{o_i} + 1$.

The full association hypothesis is then given by a set of trajectories $\mathcal{T} = \{T_k\}$, and the data association problem can be formulated as a Markov random field (MRF). More specifically, we aim at maximizing the posterior probability of trajectories:

$$p(\mathcal{T}|\mathcal{X}) = p(\mathcal{T}) \prod_i p(\mathbf{x}_i|\mathcal{T}) \tag{3.1}$$

The observation model is given by

$$p(\mathbf{x}_i|\mathcal{T}) = \begin{cases} P_i & \text{if } \exists T_k \in \mathcal{T} \wedge i \in T_k \\ 1 - P_i & \text{otherwise} \end{cases} \tag{3.2}$$

where P_i denotes the probability of $\mathbf{x}_i$ being a true detection. The prior over trajectories decomposes into a product of unary and pairwise factors

$$p(\mathcal{T}) \propto \prod_{T \in \mathcal{T}} \Psi(T) \prod_{T, T' \in \mathcal{T}} [T \cap T' = \emptyset] \tag{3.3}$$

where the pairwise term ensures that trajectories are disjoint. The unary factors are given by

$$\Psi(T) = \Psi_{en}(\mathbf{x}_{o_1}) \Psi_{ex}(\mathbf{x}_{o_l}) \prod_{i=1}^{l-1} \Psi_{li}(\mathbf{x}_{o_i}, \mathbf{x}_{o_{i+1}}) \tag{3.4}$$

where $\Psi_{en}(\mathbf{x}_{o_1})$, $\Psi_{ex}(\mathbf{x}_{o_l})$, and $\Psi_{li}(\mathbf{x}_{o_i}, \mathbf{x}_{o_{i+1}})$ model the likelihood of entering a trajectory, exiting a trajectory and linking temporally adjacent detections within a trajectory.

3.2 MINIMUM-COST FLOW SOLUTION

Taking the negative logarithm of (Eq. 3.1), the maximization can be transformed into an equivalent minimization problem over flow variables [180] as follows

$$\begin{aligned}\mathbf{f}^* = \underset{\mathbf{f}}{\operatorname{argmin}} & \sum_i C_i^{en} f_i^{en} + \sum_i C_i^{ex} f_i^{ex} \\ & + \sum_{i,j} C_{i,j}^{li} f_{i,j}^{li} + \sum_i C_i^{det} f_i^{det} \\ \text{s.t. } & f_i^{en} + \sum_j f_{j,i}^{li} = f_i^{det} = f_i^{ex} + \sum_j f_{i,j}^{li} \quad \forall i \end{aligned} \tag{3.5}$$

where $C_i^{en} = -\log \Psi_{en}(\mathbf{x}_i)$ is the cost of creating a new trajectory at $\mathbf{x}_i$ and $C_i^{ex} = -\log \Psi_{ex}(\mathbf{x}_i)$ is the cost of exiting a trajectory at $\mathbf{x}_i$. The cost of linking two consecutive detections $\mathbf{x}_i$ and $\mathbf{x}_j$ is denoted $C_{i,j}^{li} = -\log \Psi_{li}(\mathbf{x}_i, \mathbf{x}_j)$ and C_i^{det} encodes the cost of $\mathbf{x}_i$ being a true detection or a false positive (data term). Furthermore, the binary flow variables f_i^{en} or f_i^{ex} take value 1 if the solution contains a trajectory such that $\mathbf{x}_i$ is the first frame or last frame, respectively. $f_i^{det} = 1$ encodes the fact that $\mathbf{x}_i$ is part of a trajectory and $f_{i,j}^{li} = 1$ if a tracklet exists which contains both detections $\mathbf{x}_i$ and $\mathbf{x}_j$ in two consecutive frames.

Given the stated problem, an optimal solution can be obtained by mapping it into a min-cost flow network $G(V, E)$ being a directed graph with nodes $u_i \in V$, edges $(u_i, v_i) \in E$, a source $s \in V$, and a target node $t \in V$. Every edge has an assigned cost $c(u_i, v_i)$, a maximum flow capacity $f_{max}(u_i, v_i) \geqslant 0$, and an assigned flow $f(u_i, v_i) \geqslant 0$. After creating this network, the min-cost flow problem is the task to maximize a suitable flow resulting in a minimum of the total cost for paths from source to target minimizing

$$\sum_{(u_i, v_i) \in E} c(u_i, v_i)\, f(u_i, v_i). \tag{3.6}$$

Note that the solution may be that no suitable flow exists. A suitable solution must fulfill the capacity constraints $f(u_i, v_i) \leqslant f_{max}(u_i, v_i)$, skew symmetry $f(u_i, v_i) = -f(v_i, u_i)$, as well as flow

conservation $\sum_{w \in V} f(u, w) = 0 \; \forall \; u \notin \{s, t\}$ allowing flow only to be introduced or absorbed at source or target. [2]

Mapping the stated multi-target tracking problem into a min-cost flow problem is illustrated in Fig. 3.1, where two virtual nodes (source s and target t) are introduced. For each observation two nodes u_i and v_i are created (summarized as one node for clarity of illustration) with an edge between them with cost $c(u_i, v_i) = C_i^{det}$ and flow $f(u_i, v_i) = f_i^{det}$. For entering and exiting trajectories, edges between the source s and the first variable (with cost $c(s, u_i) = C_i^{en}$ and flow f_i^{en}), and the second variable and the sink t (with cost $c(v_i, t) = C_i^{ex}$ and flow f_i^{ex}) are introduced. Finally, edges between consecutive detections (v_i, u_j) encode pairwise association scores with cost $c(v_i, u_j) = C_{i,j}^{li}$ and flow $f_{i,j}^{li}$. Trajectories are established as paths between the virtual nodes and flow can only be created and absorbed at these locations. The maximum flow capacity for all edges of this network is $f_{max} = 1$ since a detection can only be part of a exactly one trajectory (and therefore on path from source to target). While we assume that only detections in consecutive frames can be linked, this can be easily generalized by allowing transitions between detections in non-consecutive frames resulting in additional edges.

To find the optimal solution, Zhang et al. [180] iteratively computed the min-cost flow solution for a given amount of flow. Starting with flow zero, the flow pushed from the source into the network is augmented one unit at a time before applying the push relabeling algorithm of [84] (with a time complexity of $\mathcal{O}(|E||V|^2 log|V|)$) to retrieve the shortest paths at each iteration. The algorithm terminates if the cost of the currently retrieved shortest path is greater or equal to zero and rejects the trajectory introduced by this shortest path. Note that this results in the computation of at least one shortest path. For efficiency, the bisection method can be applied on the number of flow units, reducing time complexity from linear to logarithmic with respect to the number of trajectories n_K (which is upper bounded by the number of nodes $|V|$). The total complexity is then $\mathcal{O}(|E||V|^2 \log^2 |V|)$, with $|V|$ number of nodes and $|E|$ number of edges. The obvious drawback of this algorithm is that the min-cost flow problem is solved $\log |V|$ times. Computations from previous iterations are

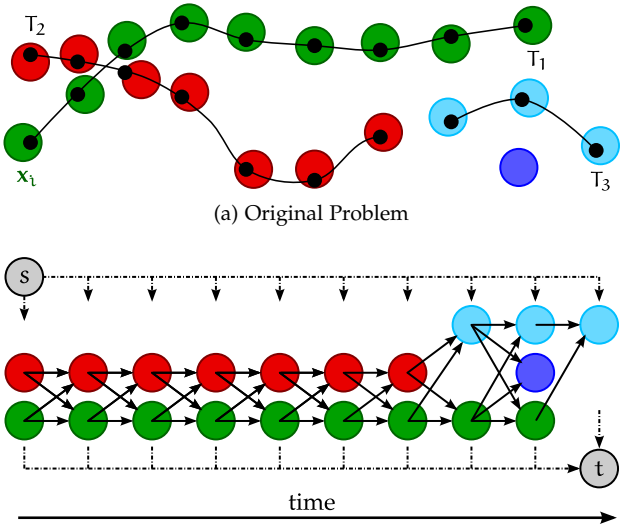


Figure 3.1: PROBLEM MAPPING. The original problem (a) is mapped into a min-cost flow network (b). Ground truth trajectories are shown in black. Colored nodes encode detections and correspond to two nodes in the min-cost flow network $G(\mathcal{X})$. For clarity of illustration, edges from the source and to the sink have been omitted.

discarded, since the coupled problems of each iteration are solved independently. Assuming a constant number of objects per frame on average and therefore $|V|$ and $|E|$ roughly linear with the number of frames n_f, the total complexity is $\mathcal{O}(n_f{}^3 \log^2 n_f)$ which is inapplicable for any (near) real-time demands. The resulting run time takes up to several minutes per frame for sequences with less than 1000 images. In the following section, we review a better optimization scheme. However, to the best of our knowledge, there is no existing approach tackling this problem in a real online fashion independent of the sequence length. We derive such a solution (which also exploits the special structure of this problem for a batch setting) in Chapter 4 in this thesis.

3.3 SUCCESSIVE SHORTEST PATHS (SSP) FOR MIN-COST FLOW

Recently, Berclaz et al. [22] proposed to use the *k-shortest paths* algorithm (*KSP*, [154]) and Pirsiavash et al. [138] adapted the *successive shortest paths* algorithm (*SSP*, Algorithm 3.1 as stated in [2, p. 106]) resulting in the same algorithm as KSP, computing the optimal set of trajectories. Both approaches exploit *Suurballe's* algorithm [154, 155] to find n_K node-disjoint paths in a network without negative costs.

The proposed algorithm computes shortest paths in residual graphs iteratively, without discarding information on trajectories found in previous iterations. Beyond that, converting the original problem (containing edges with negative costs) in a network with positive costs only further reduces the time complexity of the algorithm.

Finding a shortest path in each iteration is elementary. Generally, this problem can be solved by applying the *Bellman-Ford* algorithm (*BF*, Algorithm A.2) within the SSP framework for a graph without negative cycles and arbitrary costs [46, p. 651ff.]. Advantageously, this reduces computational complexity to $\mathcal{O}(n_K {n_f}^2)$ where n_K is the number of trajectories and n_f denotes the number of frames. Beyond that, initially converting the graph to have positive costs only [22, 138], *Dijkstra's* algorithm (Algorithm 3.2) can be applied to find shortest paths during optimization. For an initial graph without negative costs, the SSP algorithm guarantees edge weights of the residual graphs which are positive only as well. Note that we will distinguish both cases (positive and arbitrary costs) in the following chapters by mentioning the respective shortest path solver. For graphs with positive costs *SSP – Dijkstra* is used and *SSP – Bellman-Ford* is applied in the case of arbitrary costs.

GRAPHS WITH ARBITRARY COSTS We start our discussion by describing the Bellman-Ford algorithm, which is typically called at each SSP-iteration for computing the shortest path on the residual graph G_r. BF is based on the principle of *relaxation* (Algorithm A.3), in which an upper bound of the correct distance to the source for each node is gradually replaced by tighter bounds

Algorithmus 3.1 : Successive Shortest Paths (SSP) Algorithm [2]

Input : Detections $\mathcal{X} = \{\mathbf{x_i}\}$, Source s, Target t
Output : Set of trajectory hypotheses $\mathcal{T} = \{T_k\}$

```
// construct graph from observations
G(V, E, C, f) ← X
// initialize flow to 0
f(G) ← 0
// initialize G_r(f) as a DAG
G_r(f) ← G(f)
// find k-th shortest path
while C(γ_k) < 0 do
    // using any SP solver
    γ_k ← FindShortestPath(G_r(f), s, t)
    // Revert edges for G_r(f) along γ_k
    G_r(f) ← RevertEdges(G_r(f),γ_k)
return T
```

(by computing the predecessor and its distance) until the optimal solution is reached. We refer the reader to the appendix for a more detailed review (Section A.1.2).

While the BF algorithm is able to compute a single trajectory with the lowest cost, in multi-target tracking we are interested in recovering the optimal set of trajectories. In the following, we review how to compute the optimal set of trajectories by means of SSP (see Algorithm 3.1) for a given example. Let us consider the *directed acyclic graph* (*DAG*) in Fig. 3.2a which specifies an instance of the network flow problem formulated in Eq. 3.5 for four frames and three detections per frame. Note that in general we do not require the number of detections to be the same for each time step. For clarity, we only sketch edges coming from the source and going to the target. First, we relax all edges by traversing the graph from left-to-right, which is illustrated in Fig. 3.2 (b)+(c) for the first and second time step. This allows to recover the first optimal trajectory depicted in green in Fig. 3.2d. We refer to this algorithm as *DAG-SP* as stated in [46, p. 655]. An alternative at this point is to remove all edges from this trajectory and run the

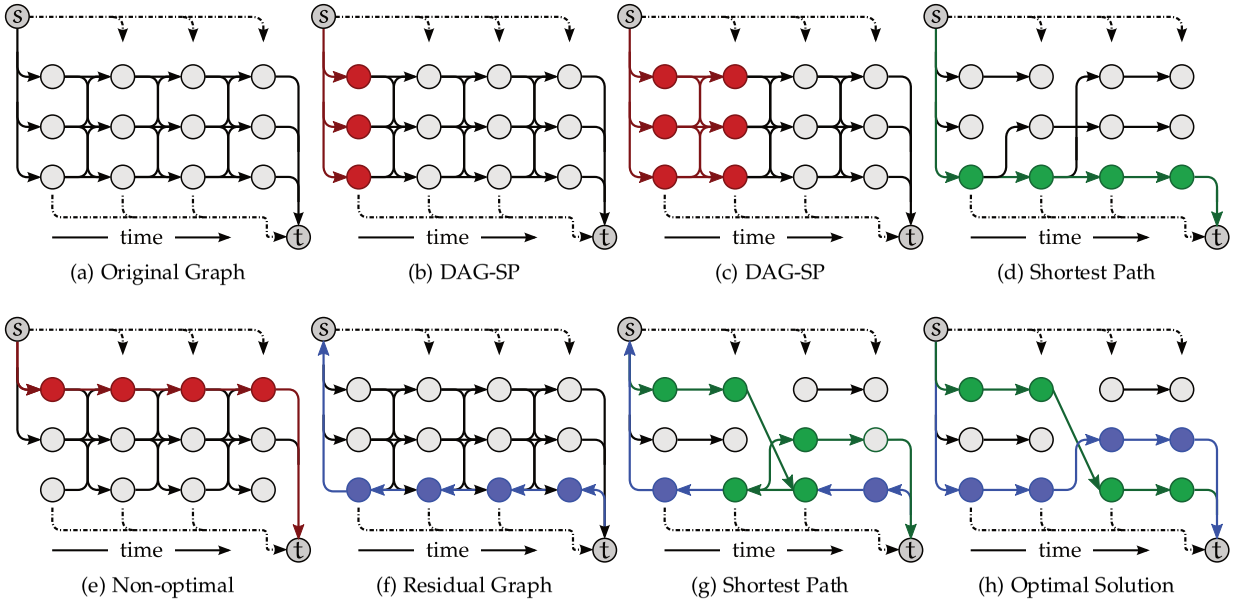


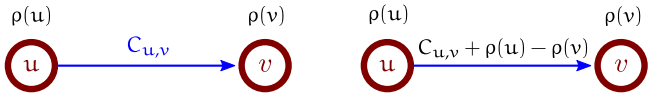
Figure 3.2: Illustration of the SSP Algorithm. The original problem is depicted in (a). The BF algorithm relaxes the red edges in (b) and (c) yielding the shortest path in the DAG, depicted in green in (d). Simply removing this path and re-running BF violates optimality as shown in (e). Instead, a SSP strategy on the residual graph (f) allows for finding the optimal solution (g)+(h).

algorithm again [138]. However, this greedy version of the algorithm does not guarantee optimality, as shown in Fig. 3.2e where some of the edges are absent in the graph and therefor not all possible trajectories are encoded anymore. Introducing a residual graph $G_r^{(k)}$ (Fig. 3.2f) by inverting the edges of the previously found optimal trajectory (by changing the direction of the edge and assigning the original cost with the opposite sign), the resulting graph is not a DAG anymore. A slight improvement of the approximative greedy solution can be achieved by considering backward-pointing edges once by traversing through the graph against the direction of the initial DAG [138]. In the following, we refer to this heuristic approximation as *dynamic programming* (DP). Instead, the SSP algorithm applies BF again on the residual graph (Fig. 3.2g) yielding the second shortest path and preserving optimality. The process finishes when a newly found shortest path has positive costs, i.e., it can not further reduce the total cost specified in Eq. 3.5. The final trajectories shown in Fig. 3.2h are recovered by extracting all backward edges from the most recent residual graph, starting the back-tracking procedure at the target node.

GRAPHS WITH POSITIVE COSTS After stating the SSP algorithm for graphs with arbitrary edge weights, we continue by reviewing the KSP algorithm [22, 154, 155] (SSP-Dijkstra) for a graph with positive costs only. Since the original tracking problem contains negative edge weights, an initial conversion of the graph is required. This can be done after initially running the DAG-SP algorithm for the topologically ordered graph, resulting in a predecessor map containing the shortest path to every node of the graph. Exploiting this predecessor map, arbitrary edge weights $C_{i,j} \in \mathbb{R}$ can be converted to weights $C'_{i,j} \in \mathbb{R}_0^+$ by

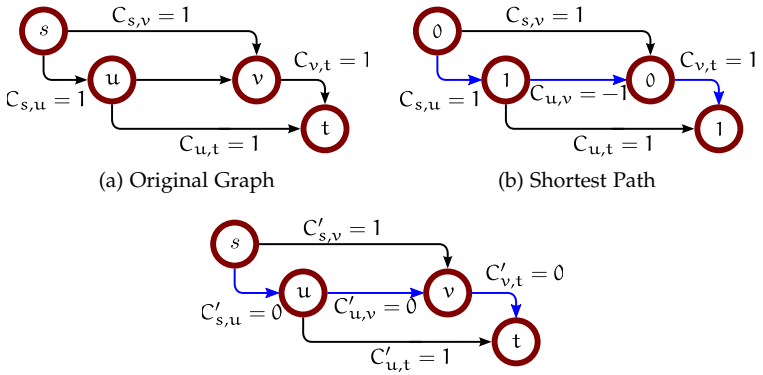
$$C'_{i,j} = C_{i,j} + \rho(i) - \rho(j) \quad \forall e_{i,j} \in E \tag{3.7}$$

resulting in a graph without negative costs (Fig. 3.3). Edges on a shortest path of this graph have a weight $C'_{i,j} = 0$ describing an arborescence Γ [159, p. 126ff.]. An example of this conversion is depicted in Fig. 3.4.



(a) Original Costs (b) Converted Costs

Figure 3.3: Cost Conversion. For an edge between the nodes u and v, costs $C_{u,v} \in \mathbb{R}$ can be converted into $C' \in \mathbb{R}_0^+$. Therefore, the distance ρ on the shortest path to both nodes u and v must be computed previously.



(a) Original Graph (b) Shortest Path

(c) Converted Graph

Figure 3.4: Cost Conversion Example. For the graph in (a) with negative costs, the shortest s-t path is computed (blue edges in (b)). Numbers in the nodes state the distance on the shortest path for this particular node. The graph is converted into a graph without negative costs according to Eq. 3.7. The resulting graph in (c) contains the arborescence Γ in blue which is identical with the shortest s-t path in this case.

We continue by reviewing KSP and start with Dijkstra's algorithm to find shortest paths as stated in Algorithm 3.2. In each KSP iteration, Dijkstra is initially called on the residual graph $G_r^{(k)}$. As BF, Dijkstra is based on the principle of relaxation. Since the minimum cost of an edge is guaranteed to be $C'_{i,j} = 0$, the algorithm exploits this property as a lower bound for the best shortest path. Starting at the source node, a minimum-priority queue is maintained and in every iteration all edges of the most promising node u to its successors v are relaxed. For every node u taken from the queue, the final shortest path was determined. Consequently, while establishing a particular s-t path, the algorithm may terminate early after taking the target node from the queue. However, within the KSP algorithm the shortest path to every node must be computed as mentioned above, whereas an early termination may leave the predecessor map in an incomplete state.

Algorithmus 3.2 : Dijkstra's Algorithm [46, p. 658ff.]

```
   Input : Graph G, Source s, Costs c
   Output : Predecessor Map π, Distance Map ρ
   // set π(u) = Unkown, ρ(u) = ∞, u ∈ |V| \ s
 1 π, ρ ← InitializeSingleSource(G,s)
 2 S ← ∅                                          // invariant nodes
 3 Q ← edges(G)                                   // min-priority queue
 4 while ¬ empty(Q) do
 5 |   u ← NodeWithMinDistance(Q)
 6 |   S ← S ∪ {u}
 7 |   foreach node v ∈ Neighbors(G,u) do
   |   |   // Check if u is a better predecessor for v
 8 |   |   if ρ(v) > ρ(u) + c(u,v) then
 9 |   |   |   ρ(v) ← ρ(u) + c(u,v)
10 return π,ρ
```

After detecting a shortest path on $G_r^{(k)}$, an iteration of the KSP algorithm is continued by converting again the edge costs of $G_r^{(k)}$

according to Eq. 3.7 resulting in a graph containing the arborescence

$$\Gamma = G_r^{(k)} \setminus \{E \mid C'_{i,j} \neq 0\}. \tag{3.8}$$

Finally, the subsequent residual graph $G_r^{(k+1)}$ is computed according to [154] by reverting the direction of edges along the shortest path $\gamma^{(k)}$. While for the original graph the SSP algorithm terminated after finding a path with positive costs, the KSP algorithm for the converted problem finishes if the sum of the total costs exceeds the costs of first iteration:

$$\sum_{i=1}^{k} \text{cost}(\gamma^{(i)}) > |\,\text{cost}(\gamma^{(0)}|\,. \tag{3.9}$$

The KSP algorithm is summarized as Algorithm 3.3.

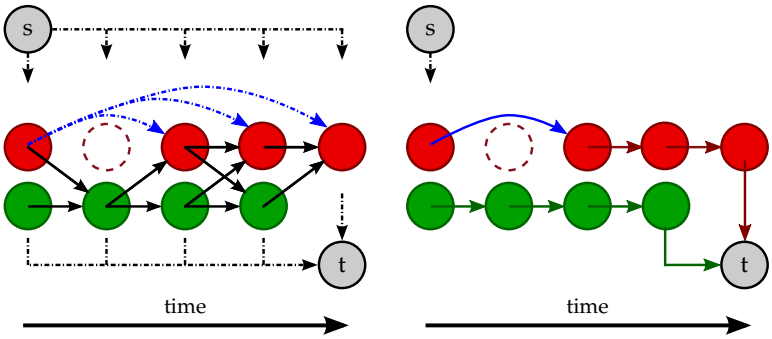
Comparing the time complexity of both SSP and KSP, differences result from the different time complexities of the shortest path solver, which must be repeated $n_K + 1$ times for n_K objects present in the scene. The total complexity of the BF algorithm is $\mathcal{O}(|V||E|)$, where $|V|$ is the number of vertices and $|E|$ is the number of edges. Basically, BF relaxes all edges as many times as vertices exist in the graph to guarantee the shortest path to every node. This is typically too expensive for real-world applications. In contrast, the lower bound and the minimum-priority queue used in Dijkstra's algorithm reduce the worst-case complexity, depending on the complexity for queue operations. Using an efficient Fibonacci heap for this task results in a time complexity of $\mathcal{O}(|E| + |V|\log(|V|))$ resulting in KSP ($\mathcal{O}(n_K(|E||V|\log(|V|))$) being more efficient than SSP ($\mathcal{O}(n_K|V||E|)$). The non-optimal DP algorithm [138] still saves $\log(|V|)$ operations over using KSP.

However, for all aforementioned algorithms run time and memory consumption still increases unbounded for larger problems and cannot be applied to an online scenario with arbitrary sequence length. Moreover, all algorithms must be ran uninitialized if new frames are arriving.

Algorithmus 3.3 : K-Shortest Paths algorithm (KSP) [22, 154]

Input : Detections $\mathcal{X} = \{\mathbf{x_i}\}$, Source s, Target t
Output : Set of trajectory hypotheses $\mathcal{T} = \{T_k\}$

$G(V, E, C, f) \leftarrow$ ConstructGraph($\mathcal{X}$) // G is a DAG
$f(G) \leftarrow 0$ // initialize flow to 0
$G_r^{(0)}(f) \leftarrow G(f)$
$\pi^{(0)} \leftarrow$ DAG-SP($G_r^{(0)}(f), s$) // find shortest path in DAG
// encode arborescence in residual graph
$G_r^{(0)}(f) \leftarrow$ ConvertEdgeCost($G_r^{(0)}(f), \pi^{(0)}$)
// Revert edges along shortest path
$G_r^{(1)}(f) \leftarrow$ ComputeResidualGraph($G_r^{(0)}(f), \pi^{(0)}$)
// find k-th shortest path
$k \leftarrow 0$
while 1 **do**
  $k \leftarrow k + 1$
  $\gamma^{(k)}, \pi^{(k)} \leftarrow$ Dijkstra($G_r^{(k)}(f), s$) // shortest path $\gamma^{(k)}$
  $G_r^{(k)}(f) \leftarrow$ ConvertEdgeCost($G_r^{(k)}(f), \pi^{(k)}$)
  $G_r^{(k+1)}(f) \leftarrow$ ComputeResidualGraph($G_r^{(k)}(f), \gamma^{(k)}$)
  // evaluate converted costs
  if $\sum_{i=1}^{k} \text{cost}(\gamma^{(i)}) > |\text{cost}(\gamma^{(0)})|$ **then**
    break
return $\mathcal{T}$



(a) Initial Network. (b) Solution.

Figure 3.5: The original problem in Fig. 3.1a is extended to explicitly model occlusions resulting in possible short-term track fragmentations. The dashed node indicates a missing detection. Blue arrows are extending the original formulation, bridging possible occlusions, leading to the optimal solution.

3.4 OCCLUSION MODELLING

So far, the introduced problem formulation and energy minimization only considers object detections in consecutive frames. However, explicitly modeling track fragmentation, e.g., due to occlusions, either on detection or on tracklet level [44, 75, 180] and considering motion models [41] is an important aspect for multi-target tracking. Therefore, this section shortly reviews how an extended problem formulation can be modeled within the energy minimization framework.

As already discussed, the min-cost flow formulation translates the multi-target tracking problem into the DAG $G(\mathcal{X})$. This formulation can easily be extended to model occlusions by introducing additional, forward-pointing edges between detections (v_i, u_j), which exist not necessarily in consecutive frames but are in chronological order. The extended problem formulation is depicted in Fig. 3.5 for a maximum occlusion length of 3 frames.

Explicitly modeling occlusions in the network structure still allows to solve the problem optimally. However, increasing the

connectivity of the network for a large number of occluded frames results in an exponentially growing solution space. To keep the problem tractable, approximations are made such as pruning [138], explicitly modeling occlusions on tracklet level [180], using branch and bound techniques, or introducing motion models and applying Lagrangian relaxation [41].

Another approach for slightly occluded objects is to also use low confident detections. This avoids fragmentations already on detection level and shifts outlier rejection to a later stage in the optimization process by exploiting image evidence.

4

EFFICIENT MULTI-CLASS DATA-ASSOCIATION

> *There is a way out of every box, a solution to every puzzle; it's just a matter of finding it.*
>
> JEAN-LUC PICARD

Chapter 3 introduced the formulation of multi-target tracking as an energy minimization problem, finding an optimal solution by exploiting min-cost flow optimization, and computing a faster approximation. Above and beyond, this chapter addresses three fundamental challenges of min-cost flow data association:

- First, the high computational complexity of existing optimal algorithms prohibits their application to large video sequences. To tackle this problem, we introduce a dynamic solver combining the advantages of Dijkstra's algorithm for finding shortest paths and the distributed Bellman-Ford algorithm [24, 163] for package routing. While the former allows for efficiently finding shortest paths in networks with positive edge weights, the latter leverages the special structure of tracking networks and reuses computation while computing shortest paths. Combining both algorithms and integrating this into the successive shortest paths framework results in a very efficient optimization algorithm for batch scenarios.

- Secondly, another limitation of current global data association algorithms is that they require a batch setting, i.e., they can only be applied to the full video sequence or handle a sequence in several batches which introduces another data association problem. This is a non-realistic assumption for many tracking scenarios. To address this, we propose an online algorithm, which integrates novel frames into the current solution in an optimal fashion.

- Thirdly, the last fundamental challenge we address is the unbounded growth of existing min-cost flow solvers in terms of memory and computational complexity. Although a globally optimal solution cannot be achieved without considering all nodes at all times, our efficient approximation allows a sufficient number of time steps during optimization and the solution can be transfered into the optimal solution continuously.

After introducing the distributed strategy for finding shortest paths in a SSP framework, in Section 4.2 we extend this dynamic algorithm to an online setting, i.e., when dealing with an incoming stream of frames instead of the common batch setup. This is an intermediate step for an unrestricted online setting. Consequently, we show how to estimate the trajectories in an online fashion while dealing with bounded memory and computational resources. This will result in an algorithm that while not being optimal can very efficiently estimate trajectories close to the optimal solution.

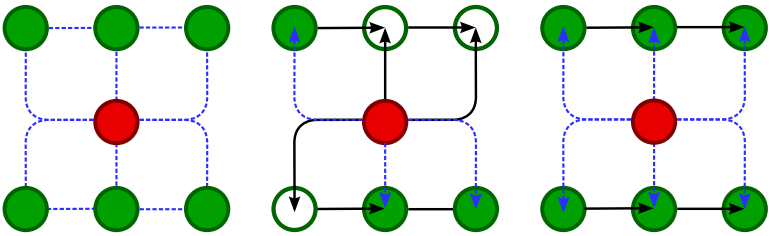
4.1 DYNAMIC MIN-COST FLOW FOR BATCH PROCESSING

The SSP algorithm requires in every iteration a predecessor map, containing the shortest path from the source to every node u including the target node (*s-t path*). Independent of the applied shortest path solver, the predecessor map must be completely known in the end of an SSP iteration. For a graph with arbitrary costs, this guarantees the shortest path to be the best solution while for a converted graph with positive costs only the resulting residual graph requires this information to encode shortest paths

γ_k to every node with $\text{cost}(\gamma_k) = 0$. Consequently, a completely known predecessor map is mandatory within every iteration of the successive shortest paths algorithm, which is also discussed in [155]. Computing this predecessor map can be achieved by starting from a empty, initialized predecessor map in each iteration using a suitable shortest path solver (as BF or Dijkstra depending on the graph formulation) or approximated as discussed in Chapter 3. Disregarding previously performed computations after finding a new trajectory is in the worst-case optimal but computationally expensive. The worst-case complexity for the shortest path computation is generally given by $\mathcal{O}(|V|^2)$ (BF) or $\mathcal{O}(|E| + |V| \log |V|)$ (Dijkstra) respectively. The shortest path solver must be called for $n_K + 1$ times within the SSP framework in order to find n_K trajectories. Considering a min-cost flow formulation for a multi-target tracking scenario, the residual graph G_r is altered only slightly within each iteration. Changes of the predecessor map are solely triggered by the previously found shortest path.

Since finding the shortest path is the key component in every iteration of the SSP algorithm, re-using previously obtained information on predecessors can reduce average-computational costs significantly. To exploit this property, we propose tailoring Dijkstra's algorithm to the multi-target tracking problem by re-using computations and only updating predecessors when needed. Combining both Dijkstra's queuing strategy and the distributed Bellman-Ford concept allows for finding shortest paths more efficiently. We will refer to this proposed algorithm as dynamic Dijkstra (*dDijkstra*) in the following.

In terms of classical network routing schemes, re-building the predecessor map from initialization is comparable to a simple flooding strategy, where every node sends information to all of its neighbors (in case of the SSP until convergence). This strategy guarantees that the information is received by every node, although multiple copies might be received (Fig. 4.1a). Consequently, flooding guarantees to utilize the shortest path, whereas broadcasting (Fig. 4.1b) ensures that each node receives the information only once [24, p.368]. In contrast, multicast communicates information only to specified group of receivers (Fig. 4.1c). Con-



(a) Flooding. (b) Broadcast. (c) Multicast.

Figure 4.1: ROUTING SCHEMES. While flooding (a) communicates a message from the red node using all edges in a network (dashed blue edges), a broadcast based strategy (b) uses a spanning tree to communicate the message to all nodes. Flooding results in multiple copies received by every node while broadcasting provides the information for every node exactly once. In contrast, multicast (c) communicates a message to a specific sub-group of nodes of a particular network. Blue edges in the network indicate a propagated message while black edges only show physical connections between nodes.

sidering the whole network, dDijkstra is closely related to a multicast strategy, since only nodes in the network receive a message, when a predecessor was updated. However, this group of nodes is initially not explicitly known. Instead of repeatedly computing and updating a spanning tree containing all nodes requiring an update, the proposed distributed algorithm solves this problem implicitly, updating nodes until no further messages must be emitted and the predecessor map converged.

The initial shortest path ($k = 0$) can be found by applying dynamic programming (DAG-SP) since all edges are forward pointing in time, whereas for subsequent iterations the residual graph also contains backward pointing edges. After finding each shortest path, the subsequent residual graph is created by reverting edges along this path and encoding the arborescence Γ (c.f. Section 3.3). Creating the most current residual graph results in all nodes along the shortest path having an invalid predecessor. Adding these nodes to an initial queue, new predecessors for all queue elements are obtained by traversing the graph in temporal

direction. This initial step updates outdated predecessors while all nodes remain in the queue. Using this as a starting point, the most promising node (with the lowest cost on its shortest path) is taken from the queue iteratively, exploiting the strategy of Dijkstra's algorithm. Thus, all successors receive the information on a new shortest path of the broadcasting node. Nodes having broadcasted a message to all neighbors are removed from the queue and nodes receiving an update (having a new predecessor) are added. In contrast to Dijkstra's algorithm, a node might be relaxed another time, since other nodes on its shortest path may receive an update before convergence.

Reviewing the algorithm, nodes can be in three different states (broadcasting, receiving, or idle). Thus, if no node is broadcasting (and therefore receiving information on a new predecessor) anymore, all nodes are in idle state. This is represented by an empty queue when the algorithm terminates and the optimal solution is obtained. Algorithm 4.1 summarizes the proposed dynamic solution. For arbitrary DAGs of different sizes, our experiments in Section 6.2.2 showed, that speed-ups can be obtained in particular for longer scenarios compared to a regular SSP implementation applying Dijkstra's algorithm. For real-world tracking problems, our experiments in Section 6.2 show that for such graphs speed-ups up to factor 2 can be obtained over a standard SSP implementation.

We illustrate our dynamic algorithm in Fig. 4.2, for the example from Fig. 3.2. Given the trajectory found in Fig. 3.2d, we revert its edges to form the residual graph in Fig. 3.2f. Note that the corresponding predecessor mappings have to be updated as the direction of these edges has changed. We start by updating all predecessors for nodes belonging to the most recent trajectory (blue path in Fig. 4.2a) in a forward sweep (relaxed edges are marked in red). Next, all nodes which received an update (a new predecessor) are added to the queue and propagate their cost to their successors (Fig. 4.2b). Nodes are taken from the queue until the predecessor map converged (Fig. 4.2c) i.e., all nodes are in idle state. The algorithm terminates and the optimal set of two shortest paths for flow $f = 2$ is found (Fig. 4.2d).

The proposed dynamic algorithm is suitable for offline scenarios when complete batches must be processed. For an online scenario when new frames arrive continuously, dDijkstra would be initialized and called again for every new time step. While this is possible for short sequences, longer scenarios create larger graphs causing every batch solver to loose real-time capability. A suitable extension to apply the dynamic algorithm using an extended caching strategy is described in the following section. Since any solver guaranteeing an optimal solution will suffer from a growing sequence length, this intermediate approach will be extended by introducing an optimization window of length τ. While introducing memory and computational bounds by setting $\tau \in \mathbb{R}^+$, optimality cannot be guaranteed anymore. However, our experiments (Section 6.2.3) show, that the tracking performance does not change significantly in our application.

4.2 DATA ASSOCIATION FOR ONLINE SCENARIOS

Typically, multi-target tracking scenarios require a problem solution promptly (allowing a short temporal delay) or even more demanding in real-time (computing results before a new frame arrives) as outlined in Section 2.3. The problem with current optimal tracking algorithms is that they scale badly with the size of the video. Tackling this problem by an approximation as the 2-pass dynamic approach in [138] merely shifts this problem, since the input sequence under consideration still cannot contain an arbitrarily large number of frames. A simple solution would be to just enforce temporal consistency between processed batches as proposed in [22], introducing a time delay between frame capturing and obtaining results. A more sophisticated approach would be to perform data association on a intermediate tracklet level, e.g., done in [44]. Both approaches, however, will not lead to the optimal solution and only the former considers a (near) online scenario at all.

Likewise, the previously introduced dynamic algorithm only exploits previously obtained computations in trajectory space (within every SSP iteration) obtaining the optimal solution for one particular time step. Consequently, a dynamic algorithm

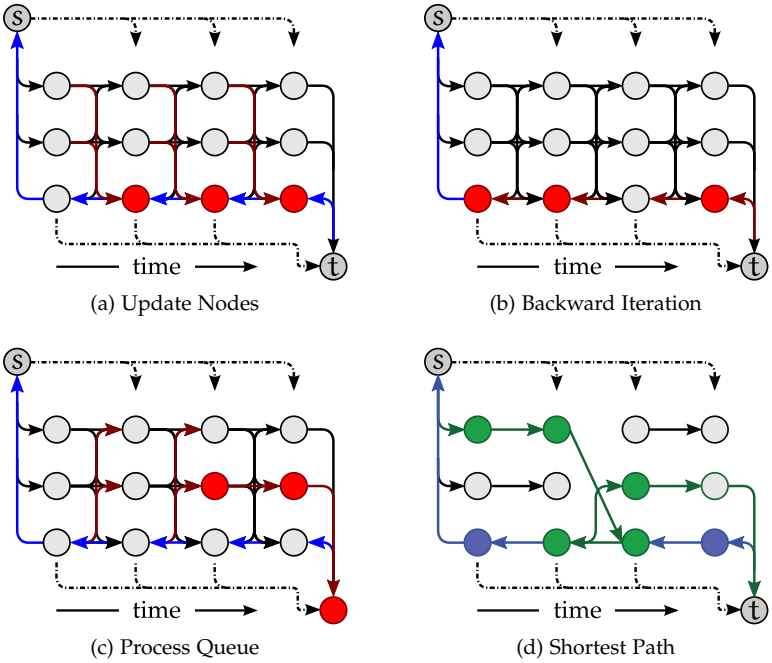


Figure 4.2: DYNAMIC MESSAGE BROADCASTING: (a) For a given residual graph (the shortest path marked in blue), nodes with invalid predecessors are detected (in red). For each node, the current best predecessor from the previous time step is selected among all candidates (connected by red edges). Nodes from (a) are put into a queue. (b) The queue is maintained, taking successively nodes from the queue and relax their outgoing edges. Nodes with updated predecessors (in red) are in turn put into the queue. (c) Exploiting Dijkstra's algorithm, the queue is maintained in a minimum-priority manner until no nodes are left, which leads to the final solution for this particular residual graph (d).

Algorithmus 4.1 : Data Association by SSP using dDijkstra

Input : Set of Detections $\mathcal{X} = \{\mathbf{x_i}\}$
Output : Set of trajectory hypotheses $\mathcal{T} = \{T_k\}$

$G(V, E, C, f) \leftarrow$ ConstructGraph$(\mathcal{X}, s, t)$
$f(G) \leftarrow 0$ `// initialize flow to 0`
$\gamma^{(0)}, \pi^{(0)} \leftarrow$ DAG-SP$(G(f))$ `// get shortest path in DAG`
$G_r^{(0)}(f) \leftarrow$ ConvertEdgeCosts$(G(f), \gamma^{(0)}, \pi^{(0)})$ `// initialize` $G_r(f)$
$G_r^{(1)}(f) \leftarrow$ ComputeResidualGraph$(G_r^{(0)}(f), \pi^{(0)})$
$q \leftarrow \emptyset$ `// q is maintained for every iteration k`
while 1 **do** `// find shortest paths for` $k \geqslant 1$
  $k \leftarrow k + 1$
  `// initial predecessor map` π_k `must be updated`
  $\pi^{(k)} \leftarrow \pi^{(k-1)}$
  `// predecessors on last shortest path are invalid`
  $q \leftarrow \gamma^{(k-1)}$
  `// process queue in time direction`
  foreach *node* $u \in q$ **do**
    `// check predecessor from past`
    $\pi^{(k)} \leftarrow$ Update$(\pi^{(k)}, u)$
    if *updated* **then**
      $q \leftarrow$ AddSuccessors$(q, u, G_r^{(k)}(f))$
  `// process queue`
  while $\neg$ *empty*(q) **do**
    $u \leftarrow$ NodeWithMinDistance(q) `// pop node`
    $q \leftarrow q \setminus u$
    foreach *node* $v \in$ *Neighbors*$(G_r^{(k)}(f), u)$ **do**
      `// Check if` u `is a better predecessor for` v
      $\pi^{(k)}(v) \leftarrow$ Relax(u, v, c)
      if $\rho(v) > \rho(u) + c(u, v)$ **then**
        $\rho(v) \leftarrow \rho(u) + c(u, v)$
        $q \leftarrow$ AddNode(q, v)
  $G_r^{(k)}(f) \leftarrow$ ConvertEdgeCost$(G_r^{(k)}(f), \pi^{(k)})$
  $G_r^{(k+1)}(f) \leftarrow$ ComputeResidualGraph$(G_r^{(k)}(f), \gamma^{(k)})$
  `// evaluate converted costs`
  if $\sum_{i=1}^{k} \text{cost}(\gamma^{(i)}) > |\text{cost}(\gamma^{(0)})|$ **then**
    break
return $\mathcal{T}$

for an online setting should additionally exploit previously computed but unchanged predecessors for previous time steps. Hence, we extend the dynamic algorithm (dDijkstra) presented in Section 4.1 to the online setting (*odDijkstra*, Section 4.2.1) allowing for further reductions in runtime while maintaining optimality. We then present a memory-bounded algorithm (*mbodDijkstra*, Section 4.2.2), which can estimate trajectories in a principled way in large graphs, where existing algorithms fail due to their unbounded memory and computation requirements. Our memory bounded algorithm computes locally optimal trajectories for a time window of τ frames in a dynamic fashion. Previous trajectories, predecessor maps, and residual graphs are held in a cache and updated at each iteration. By collapsing paths of nodes which leave the optimization window, our algorithm is able to remember previously found trajectories and thus maintains track identities. While optimality can not be guaranteed anymore, the performed experiments in Section 6.2 indicate little loss in performance with respect to batch processing on the full sequence. The time window size τ controls the tightness of the approximation and for $\tau = n_f \in \mathbb{R}^+$ our mbodDijkstra algorithm reduces to the odDijkstra algorithm which obtains an optimal solution again.

4.2.1 *Globally Optimal Online Solution (odDijkstra)*

Efficiently computing the optimal min-cost flow solution for an online setting (i.e., new frames are arriving continuously) must consider two main problems. First, a previously obtained predecessor map (in time and trajectory space) must be selected as an initialization. Secondly, for updating this initialization, an efficient broadcasting strategy must be applied to make the whole predecessor information current. Hence, an efficient caching and broadcasting strategy is mandatory and derived from the dDijkstra strategy in the following.

Fig. 4.3 illustrates the idea of our distributed online algorithm: Consider the network specified in (a) with the most recent time step n_f and its current solution in (b) for which the optimal set of trajectories and predecessor maps are available. As a new frame arrives (c), new nodes are added extending the graph to the next

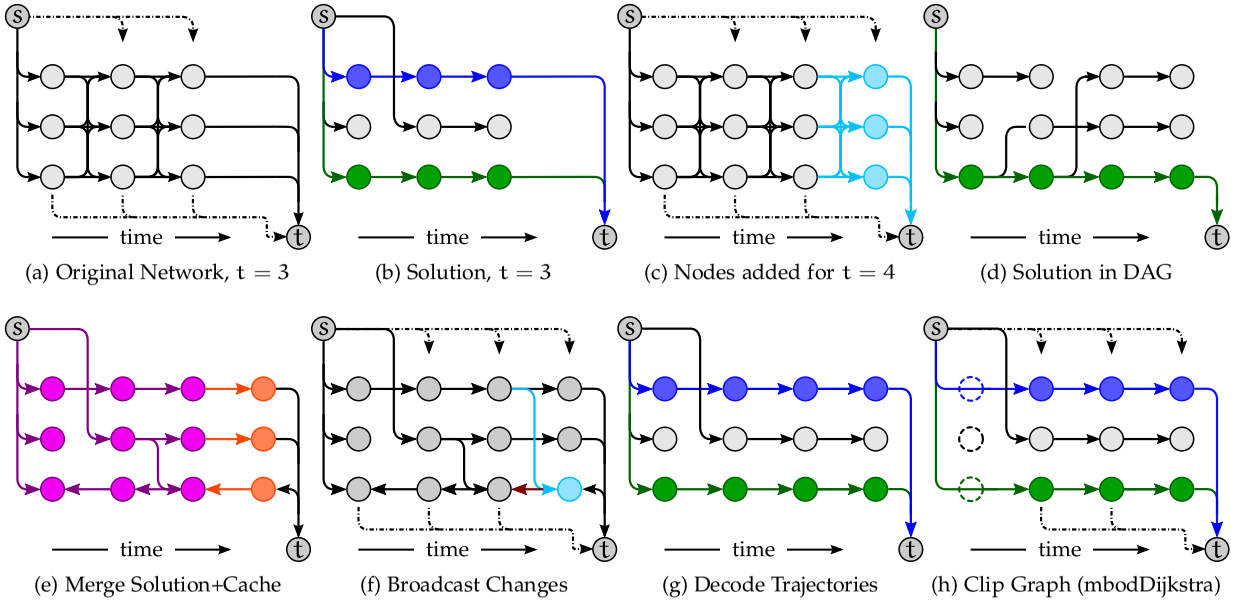


Figure 4.3: ONLINE (A)–(G) AND MEMORY BOUNDED (H) ALGORITHM: Assume that for the original network at t = 3 (a), the solution (b) is already known (shortest paths in blue and green). A new set of detections (cyan nodes) for t = 4 arrives and is connected to the graph (c) resulting in a DAG for which the shortest path can be found by applying one step of the DAG-SP algorithm (d). The predecessor map from the cache for the results up to t = 3 (in magenta) is re-used and the orange nodes are added to the queue (e). The queue is maintained and new predecessors are found (f). After one iteration, the algorithm terminates and the optimal solution is found (g). For the mbodDijkstra algorithm, paths which pass through nodes in the oldest time step are merged into the entry costs of the next time step (h).

time step. The first trajectory from the previous time step $n_f - 1$ is extended by running one dynamic programming step of the DAG-SP solver, since all edges point forward. For this iteration, the cached predecessor map from $n_f - 1$ can be extended at all times. Solely adding new nodes for n_f to the DAG does not introduce any changes for computed predecessors in previous time steps. For the current iterations of the successive shortest path algorithm, the predecessor maps from the previous time steps can be re-used if the paths under consideration are in an identical order. In this case, we use the predecessor map marked in magenta in Fig. 4.3e. Note that the second trajectory in blue has already been found up to the previous time step. The new nodes marked in orange are added to a minimum-priority queue, which is maintained until all nodes are in idle state. In this example, only the predecessor of the node marked in cyan in Fig. 4.3f is updated. This node broadcasts a message to its successors (red edges). As no changes take place, the algorithm terminates. The final result for frame $t = 4$ is illustrated in Fig. 4.3g.

Unfortunately, this simple caching strategy alone is often not sufficient in practice. In the worst case, a non-cached trajectory is found and the batch processing step must be performed since no previously computed information is available. For the simple caching strategy described above, this happens primarily in the presence of competing trajectories with similar costs, frequently changing their ordering within the SSP iterations from frame to frame. This happens primarily for groups of pedestrians and inner-city intersection scenarios and is therefore relevant for urban automated driving.

To tackle this problem, we extend our cache $\mathcal{C}$ in the following way. We keep all predecessor maps for a cache length of $|\mathcal{C}|$ frames in memory. If a trajectory from the previous time step is not extended, the cache is searched for a match containing identical shortest paths up to both frame $n_f - \delta_i$ and the current iteration k. Consequently, a valid cache has the same history in trajectory space as in the current SSP iteration. This procedure always uses the most recent cache, since fewer nodes must be updated in the following broadcasting procedure. If such a cache was found from an older time step δ_i, the predecessor map $\pi_k^{(n_f)}$ is ini-

tialized using this cache. Therefore, this initialization contains no information on predecessors for frames newer than $n_f - \delta_i$. Using this as a starting point, the current predecessor map is updated by the dynamic broadcasting strategy (Section 4.1). All outdated nodes are added to a minimum-priority queue which is processed until convergence. Every node taken from the queue broadcasts its updated shortest path information to its successors, which are added to the queue in case of receiving information on a better predecessor.

This caching strategy allows to re-use computations even in the presence of competing trajectories and to reduce the number of computations compared to applying a pure offline solution to this online problem. However, since finding a shortest path for a new frame typically results in changes of the total cost for this path from source to sink, the sink broadcasts this information backwards in time due to the reverted edges of the already established paths in the residual graph. Note that this is not the case for any shortest path terminating in a frame earlier than most recent time step.

Nonetheless, an arbitrary large number of arriving frames results in an arbitrary large graph possibly growing beyond memory or computational bounds. Since this cannot be restricted while ensuring global optimality at all times, we introduce a memory-bounded solution in the following section. Although global optimality is obviously violated, a trade-off between performance and optimality can continuously be chosen and at least for the optimized time window an optimal solution is computed.

4.2.2 *Memory-Bounded Solution (mbodDijkstra)*

While the odDijkstra strategy proposed in the previous section allows to reduce the number of relaxations compared to running a batch solver for every time step, it still scales poorly to very large problems since messages might be broadcasted back to the first frame in order to guarantee optimality. Furthermore, storing graphs of arbitrary length in memory is infeasible in practice. Thus, in this section we devise a memory and computation-

ally bounded approximation which we call *mbodDijkstra* (Algorithm 4.2, Fig. 4.3).

Unlike the batch processing proposed in [22], mbodDijkstra does not introduce any temporal delay. In contrast, the previously described optimal approach for online scenarios is extended in order to perform optimization within a specified time window length τ. Since the most recent time step considered within optimization is the newest available captured frame, the most available evidence is used for optimization and therefore the model selection problem is solved recurrently by incorporating new evidence arriving with every captured frame. To keep the problem tractable in terms of memory and computation, frames leaving the optimization window are deleted. Beyond temporal consistency, previously obtained results are remembered during optimization for the current time step. Results obtained within the clipped time window are still guaranteed to be optimal, although global optimality is sacrificed. However, erroneously remembered results can be detected, even if a correction is not possible anymore.

As this algorithm extends the previously described odDijkstra algorithm, we will now review necessary modifications in order to extended to the memory-bounded version of this dynamic online algorithm: Consider an optimized time window of $\tau = 4$ frames with the most recent time step n_f. The solution from the previous time step (i.e., up to frame $n_f - 1$) is given by the graph in Fig. 4.3g. Before adding new nodes to the graph for the most current time step n_f, the oldest time step $n_f - \tau$ is removed from the graph as illustrated in Fig. 4.3h to guarantee a memory/computational budget. Simply deleting edges from the graph will be very suboptimal as this completely discards computations from the previous time steps. Even temporal consistency cannot be enforced. In order to "remember" established paths, we map the predecessors for nodes at $n_f - \tau$ to the respective entry edge and delete all remaining edges:

$$C_{en,i}^{(n_f-\tau+1)} = C_{en,i}^{(n_f-\tau)} + C_i^{(n_f-\tau)} + C_{i,j}^{(n_f-\tau)} \quad \forall u_i \in \mathcal{T}^{(n_f-\tau)} \tag{4.1}$$

The caching strategy is similar to the one used for the optimal online approach. The main difference results from the clipped

graph used during optimization. Thus, the cache obtained in previous time steps contains nodes that do not exist in the current graph anymore. While extracting the first shortest path ($k = 0$) by running another dynamic programming step of the DAG-SP solver, the corresponding predecessor map of the previous time step $\pi_0^{(n_f-1)}$ can be extended without any modifications of the algorithm. No changes of the predecessor map must be propagated for older time steps while extending $\pi_0^{(n_f-1)}$ to $\pi_0^{(n_f)}$ for clipping the original DAG. For the subsequent shortest paths, the predecessor map $\pi_k^{(n_f)}$ of each iteration k is initialized by a cached equivalent from a previous time step as in odDijkstra. In comparison, the previously found shortest paths $k_i = 0, \dots, k-1$ for the current time step n_f are compared only within the optimization window $n_f - \tau$ to predecessor maps held in the cache. A matching predecessor map

$$\pi_{k_i}^{(n_f)} \equiv \pi_{k_i}^{(n_f-\delta)} \quad \forall\, k_i = 0, \dots, k-1 \tag{4.2}$$

is then clipped to the current optimization window. Again, no changes must be propagated at this stage within $\pi_{k_i}^{(n_f)}$ since the clipped DAG and consequently the intermediate graphs inherently contain the clipped trajectories. After initializing $\pi_{k_i}^{(n_f)}$, the updating strategies as described for odDijkstra can be applied to the predecessor map resulting in a new shortest path. In the rare event that a trajectory is found, which is not represented in the current cache, we resort to the batch solver on the full time window, guaranteeing a valid predecessor map for the current SSP iteration.

Clipping the DAG to the length of the optimized time window maintains trajectories which cannot be changed anymore. However, a shortest path containing a clipped edge indicates a (not possible) modification in the past and therefore deviations from the optimal solution.

The memory-bounded solution only returns results obtained within the optimization window. Thus, the optimized trajectories are only representing a very small part compared to the full sequence length. Typically, trajectories for a longer period or even for the full time window are required for downstream applica-

tions. Consequently, our approach allows for storing the trajectories over the whole sequence, even if the offline part cannot be considered during optimization anymore. Clipped edges are used to extend the offline trajectories and link them to the online part. Thus, the most available evidence is used and trajectories are always estimated up to the most recent time step. Furthermore, this allows remembering trajectories for an arbitrary history without considering them explicitly during optimization.

Algorithmus 4.2 : Data association by SSP using mbodDijkstra

Input : Current Detections $\mathcal{X}_{n_f} = \{\mathbf{x}_{n_f}\}$, $G(V, E, C, f, s, t)$, Cache $\mathcal{C}$
Output : Set of trajectory hypotheses $\mathcal{T} = \{T_k\}$

`// clip graph at the beginning (mbodDijkstra)`
foreach *node* $u \in [n_f - \tau]$ **do**
  `// remember predecessor by updating` C_i^{en}
  $G(f) \leftarrow \text{UpdateEntryEdge}(G(f), u)$
  $G(f) \leftarrow \text{RemoveObservation}(G(f), u)$
`// From here on, (mb-)odDijkstra are identical.`
$G(f) \leftarrow \text{AddObservations}(G(f), \mathcal{X}_{t_i})$ `// G is still a DAG`
`// initialize` $\pi^{(0)}$ `for the DAG from last time step`
$\pi^{(0)} \leftarrow \mathcal{C}^{(0)}(n_f - 1)$
`// run DAG-SP for edges` $(u, v) \in [n_f - 1, n_f]$
$\gamma^{(0)}, \pi^{(0)} \leftarrow \text{DAG-SP}(G_r(f), n_f - 1)$
$G_r^{(0)}(f) \leftarrow \text{ConvertEdgeCosts}(G(f), \gamma^{(0)}, \pi^{(0)}, n_f - 1)$
$G_r^{(1)}(f) \leftarrow \text{ComputeResidualGraph}(G_r^{(0)}(f), \pi^{(0)})$
$q \leftarrow \emptyset$, $k \leftarrow 0$
while 1 **do**
  $k \leftarrow k + 1$
  `//` $\gamma_{n_f - \delta_i}^{(k-1)} = \gamma_{n_f}^{(k-1)}$, $\delta_i \in \mathcal{D}$
  $\delta_i \leftarrow \text{FindMostRecentCache}(\mathcal{C}, \gamma^{(l)} \; \forall \, l = 0, \ldots, k-1)$
  $\pi^{(k)} \leftarrow \mathcal{C}(\delta_i, k)$ `// update predecessor map` π_k
  $q \leftarrow \{\gamma_{n_f}^{(k-1)}, (u, v) \in [n_f - \delta_i, n_f]\}$ `// invalid predecessors`
  `// Algorithm 4.1, line 11`
  $G_r^{(k+1)}(f), \gamma^{(k)} \leftarrow \text{ProcessQueue}(q, \pi^{(k)}, G_r^{(k)}(f))$
return $\mathcal{T}$

5

IMAGE EVIDENCE

> *You create a universe by perceiving it, so everything in the universe you perceive is specific to you.*
>
> DOUGLAS ADAMS

The previous chapters introduced optimization techniques resulting in an (approximated) optimal solution for multi-target tracking. Both, batch methods and our online and memory-bounded algorithm described in Section 4.2 rely on unary (i.e., per detection) and pair-wise (i.e., association) costs. Therefore, image cues to encode these terms as costs for a min-cost flow formulation are described in this chapter. For both terms, monocular and stereo knowledge can be incorporated to encode the observation model $p(\mathbf{x}_i|\mathcal{T})$ of an observation $\mathbf{x}_i$ and the prior over trajectories $p(\mathcal{T})$ with its unary terms $\Psi(T)$, $T \in \mathcal{T}$ and map it into the min-cost flow network $G(\mathcal{X})$. Costs related to detections and associations in 3-dimensional space can easily be integrated in the min-cost flow formulation by adding respective edges, but require a stereo camera setup. The proposed 3D object detections are computed using scene flow information, whereas the observer's ego-motion is implicitly obtained.

To create the final costs of the image cues used for association (i.e., bounding box overlap, color histogram similarity, cross-correlation, bounding box size similarity, optical flow overlap,

orientation similarity as well as positional similarity), the unary factors $\Psi(T)$ of the trajectory prior are obtained using ground truth for real data. Weights $\mathbf{w}$ and offsets $\mathbf{o}$ for the final costs $C_{i,j}^{li} = ((\mathbf{1} - \mathbf{s}) + \mathbf{o})^T \mathbf{w}$ are optimized similarly. While this is outlined in the following, the description of the used data and obtained results are presented in Chapter 6.

5.1 DETECTIONS

Chapter 3 introduced tracking-by-detection as one of the most promising approaches to multi-target tracking. Consequently, the problem statement introduced multi-target tracking as the problem of associating detections to form tracklets. This formulation defines a tracklet $T_k = \{\mathbf{x}_{k_1}, \mathbf{x}_{k_2}, \ldots, \mathbf{x}_{k_{l_k}}\}$ as a set of detections $\mathbf{x}_i$. We start by representing such a detection as a 2D bounding box as commonly used for tracking-by-detection approaches. Beyond that, this section presents a 3D detector for moving objects, providing additional information in terms of a 3D position relative to the observer and a continuous orientation estimation.

Classified object detections of *cars*, *pedestrians*, and *cyclists* are considered for qualitative evaluation, resulting in a comprehensive description of the traffic scene. While these classified detections are obtained as pure 2D results using a monocular camera system, for the proposed 3D detections of moving objects (which are typically active traffic participants) we introduce the generic class label *traffic participant*. Empirically, we found that the dominant object classes in our dataset are *cars* and (with some limitations in their total amount) *pedestrians* and *cyclists*. Therefore, we restrict our quantitative evaluation to cars while qualitative results are shown for all three object classes.

5.1.1 *2D Object Detections*

We start with the definition of a 2D object detection as $\mathbf{x}_i = (\mathbf{u}_i, t_i, \mathbf{s}_i, a_i, \alpha_i, d_i, o_i)$. Each detection occurs in a particular time step specified by its frame index $t_i \in \mathbb{N}$. A 2D detection is described by its bounding box $\mathbf{b}_i$ of size $\mathbf{s}_i = (w, h)^T \in \mathbb{R}^2$ de-

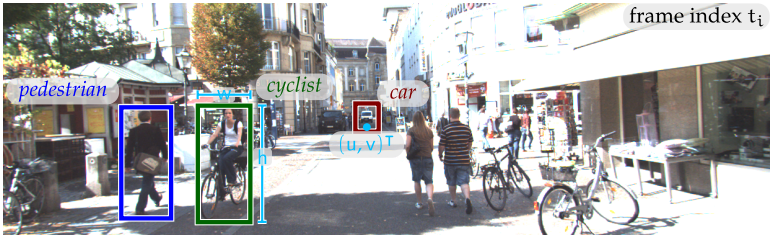


Figure 5.1: OBJECT DETECTIONS. Different object classes are detected.

scribing width and height at its position specified by the bottom center of the bounding box $\mathbf{u}_i = (u, v)^\intercal \in \mathbb{R}^2$. Additionally, the appearance $a_i = f(g|\mathbf{b}_i)$ described by pixel values $g \in \mathbb{R}^3$ for color images and for a given bounding box, the object orientation relative to the camera position $\alpha_i \in \mathbb{R}$ [79], and a detector score $d_i \in \mathbb{R}$ [65] are available as additional cues for all detections. Finally, each detection contains an assigned discrete class label $o_i \in \{car, pedestrian, cyclist\}$. A depicting example of the object definition is given in Fig. 5.1.

We used the publicly available part-based object detector[1] of [65] to obtain a set of 2D object detections $\mathcal{X}$ in each frame as input data for our algorithms. This object detector is capable of representing high intra-class variance for arbitrary object classes and showed state-of-the-art performance on challenging object detection benchmarks as e.g., the PASCAL visual object challenge [62]. The object representation is based on mixtures of multi-scale deformable part models and for training a latent support vector machine is used. This training procedure allows the position of individual parts to be unknown. These hidden variables and the model parameters are found by applying a stochastic gradient descent within an iterative training process. For training the object detector, we manually annotated the object classes car, pedestrian and cyclists for a training set. We provide a more detailed description of the dataset and properties of training and test data in Section 6.1.

1 Source Code: http://www.cs.berkeley.edu/~rbg/latent/

5.1.2 *3D Object Detections*

In this thesis, we propose a 3D object detector for moving traffic participants.[2] This detector is based on 3D scene flow information which is caused by moving traffic participants creating primarily non-background motion in the scene. Consequently, we extend the object detection $\mathbf{x}_i$ by a 3D positional information $\mathbf{X}_i$ and dimensions $\mathbf{S}_i$ respectively. Figure 5.2 outlines our system which is described in the following.

To retrieve scene flow information, feature matches in the current and previous stereo image pair must be extracted. We use the algorithm of [80] which extracts interest points by filtering the input images with 5×5 blob and corner masks. Feature candidates are obtained from the filtered images using non-maximum and non-minimum-suppression. Assuming smooth camera motion, a compact Sobel-based descriptor is computed and features are matched between left and right images in the current and previous image based on the sum-of-absolute-differences of the descriptor. Such a "circle match" at image position $\mathbf{u}$ for the most current time t is defined as $\mathbf{u}_{L,t-1} \leftrightarrow \mathbf{u}_{R,t-1} \leftrightarrow \mathbf{u}_{R,t} \leftrightarrow \mathbf{u}_{L,t} \leftrightarrow \mathbf{u}_{L,t-1}$. A candidate is accepted if the last feature in the "circle" coincides with the first one. All accepted feature matches are used to compute the egomotion of the camera system using the publicly available implementation of libVISO[3]. The result is obtained by minimizing the sum of re-projection errors for the 3-point algorithm [85, 132]. To increase robustness a RANSAC scheme is applied to compute the final result [73, 66]. The egomotion is then given for each time step by the estimated transformation parameters $(\hat{\mathbf{R}}, \hat{\mathbf{t}})$ for rotation and translation. Egomotion is compensated by representing all scene flow vectors in the coordinate system of the current image. Note that very small flow vectors are considered as background noise and removed for further processing.

Since calibrated and rectified images are used and the disparities for a match between left and right frame at a particular time step are estimated at sub-pixel accuracy, 3D points

2 Source Code: http://www.mrt.kit.edu/software/
3 Source Code:http://www.mrt.kit.edu/software

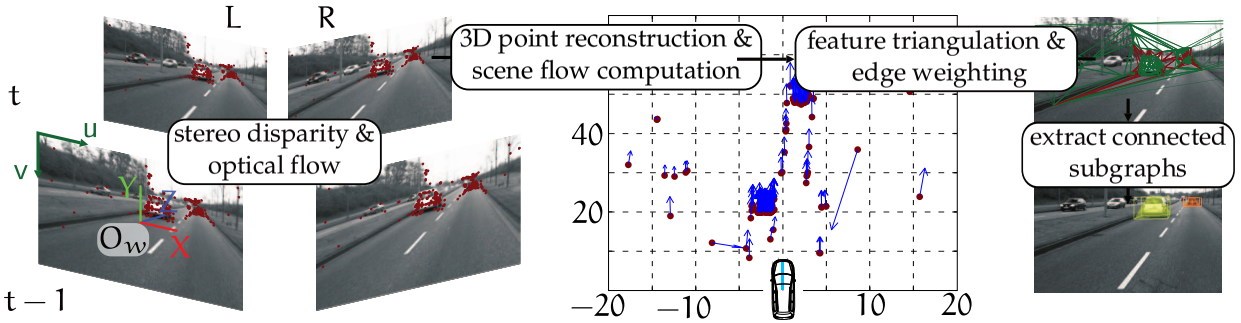


Figure 5.2: System Overview. (a) Interest points are detected in two consecutive stereo image pairs. (b) Stereo disparity and optical flow information lead to a 3D reconstruction of the detected points which are here projected on the detected ground plane. (c) Interest points are connected in a graph-like structure. The resulting edges are removed (marked in red) if the scene flow difference exceeds a certain threshold. The remaining connected components describe moving objects in the scene.

$\mathbf{X} = [X, Y, Z]^\top \in \mathbb{R}^3$ are obtained using a simple triangulation by

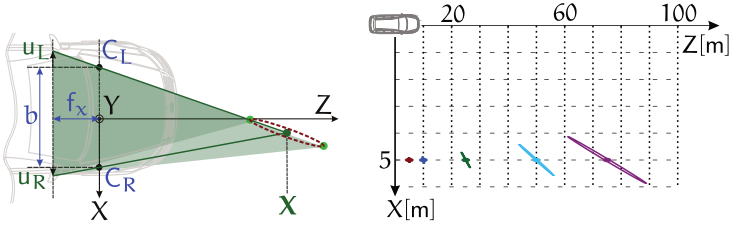
$$X = \frac{(u_L - c_{u,L}) \cdot b}{u_L - u_R}, \quad Y = \frac{(c_{v,L} - v_L) \cdot b}{u_L - u_R}, \quad Z = \frac{b \cdot F}{u_L - u_R} \tag{5.1}$$

where b denotes the baseline of the stereo system, c_u and c_v the principal point of the camera and F the focal length for the rectified images. The center of the left-handed world coordinate system is $\mathbf{O}_W$ with the X axis pointing to the right as depicted in Fig. 5.2.

The scene flow $\mathbf{f}$ for a world point is assumed to be constant for two consecutive frames. Thus, the egomotion-corrected scene flow is computed as the first order derivative which can be approximated by the difference of the world points $\mathbf{X}$ in two consecutive frames. Assuming a constant sampling rate of $1/\Delta t$, all points are represented in the most current coordinate system. Rigidly moving objects in the scene are assumed to have similar scene flow values. Thus, clustering regions of homogeneous scene flow describes moving traffic participants (Fig. 5.4).

$$\mathbf{f} = \frac{\Delta X}{\Delta t} \tag{5.2}$$

For such a clustering, our approach establishes a graph-like structure connecting all detected interest points in the image plane using a Delaunay triangulation [49] (Fig. 5.4a). Classifying edges of this graph according to scene flow differences $\|\mathbf{f}_i - \mathbf{f}_j\|_2$ of directly connected nodes (i, j) leads to a clustering of similarly moving objects. However, the uncertainty stemming from the stereo reconstruction error does not lead to a satisfying solution for such a simple thresholding as shown in Fig. 5.4b. Consequently, we consider the resulting error of the 3D position by linear error propagation. The quadratically growing reconstruction error is exemplarily shown in Fig. 5.3.



(a) Model of the 3D reconstruction error. (b) Quadratically growing reconstruction error.

Figure 5.3: RESULTING NOISE OF 3D RECONSTRUCTION. One pixel noise of the disparity $u_L - u_R$ leads to the 3D position $\mathbf{X}$ (and its error shown in red) (a). Error propagation leads to a quadratically growing error of the 3D position with the largest uncertainty perpendicular to the line of sight (b).

To compute the Mahalanobis distance of two connected nodes, we need to compute the Jacobian $\mathbf{J}$ for the scene flow $\mathbf{f}$ by

$$\mathbf{J} = \frac{\mathrm{d}\mathbf{f}}{\mathrm{d}(u_L, u_R, v)^\top} = \begin{bmatrix} \frac{\partial \mathbf{f}_X}{\partial u_L} & \frac{\partial \mathbf{f}_X}{\partial u_R} & \frac{\partial \mathbf{f}_X}{\partial v} \\ \frac{\partial \mathbf{f}_Y}{\partial u_L} & \frac{\partial \mathbf{f}_Y}{\partial u_R} & \frac{\partial \mathbf{f}_Y}{\partial v} \\ \frac{\partial \mathbf{f}_Z}{\partial u_L} & \frac{\partial \mathbf{f}_Z}{\partial u_R} & \frac{\partial \mathbf{f}_Z}{\partial v} \end{bmatrix} \tag{5.3}$$

The covariance $\boldsymbol{\Sigma}$ of the scene flow is then given by

$$\boldsymbol{\Sigma} = \mathbf{J}\sigma^2\mathbf{I}\mathbf{J}^\top \tag{5.4}$$

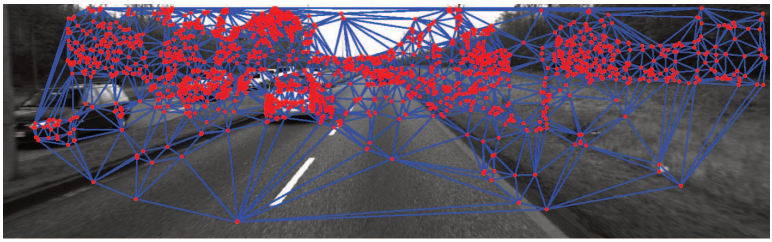
where $\sigma^2\mathbf{I}$ is the diagonal measurement noise matrix assuming a measurement noise of 0.5 pixel. For two adjacent nodes i and j of the graph, the Mahalanobis distance D equals to

$$D(\mathbf{f}_i, \mathbf{f}_j) = \sqrt{(\mathbf{f}_i - \mathbf{f}_j)^\top \boldsymbol{\Sigma}_{i,j}^{-1} (\mathbf{f}_i - \mathbf{f}_j)} \tag{5.5}$$

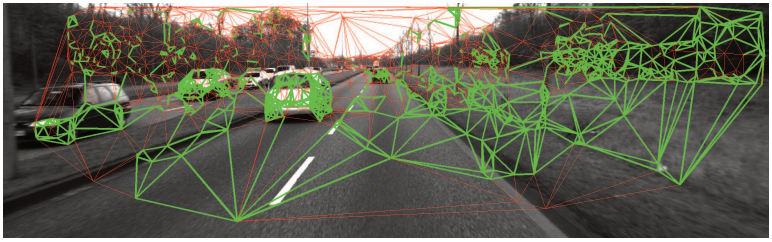
according to [115]. We compute D for all edges $(i, j) \in E$, where a single edge is weighted 0 (or removed) if $D(\mathbf{f}_i, \mathbf{f}_j) > D_{\text{min}}$ exceeding a certain threshold. Note that this threshold D_{min} can be

chosen as a fixed value already considering the stereo reconstruction error. Ultimately, moving objects in the scene are extracted as the remaining subgraphs of the initial graph. These subgraphs contain nodes with small scene flow differences and similar motion describing a rigid object as shown in Fig. 5.4c. The remaining connected components in Fig. 5.5a of the graph are detected by depth-first search (c.f. Section A.1.1) [46].

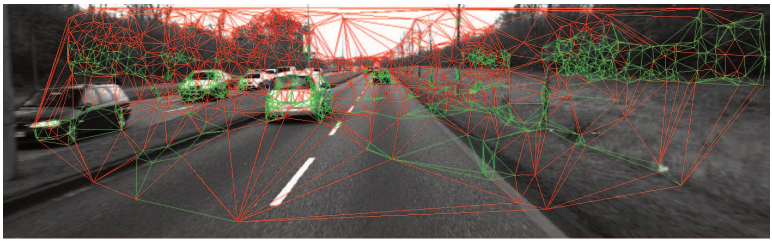
Although small flow vectors were removed while compensating the egomotion, noisy input data may still result in large background objects. Since this happens particularly for far feature points the resulting bounding boxes typically enclose an implausibly large volume and, likewise, distances between neighboring feature points are long. Therefore, we add a plausibility check to our algorithm that examines these geometric conditions for all segments and eventually eliminates false alarm detections. An example for geometrically rejected objects and the final hypotheses is given in Fig. 5.5a and Fig. 5.5b.



(a) Interest Point Triangulation

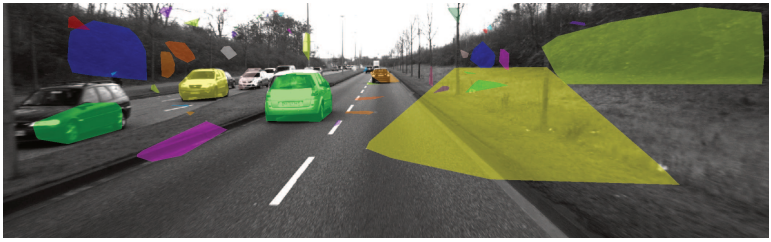


(b) Fixed Threshold

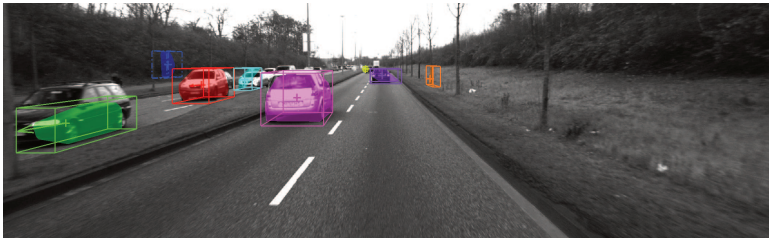


(c) Thresholding of Mahalanobis Distance

Figure 5.4: GRAPH BUILDING AND CLUSTERING. Detected interest points are connected using a Delaunay Triangulation (a). A fixed threshold to remove the edges considering the scene flow difference does not lead to a satisfying solution (b). Considering error propagation and applying a threshold on the resulting Mahalanobis distance groups similar objects in the scene (c).



(a) Remaining connected components



(b) Geometrically checked objects

Figure 5.5: Object detection. All remaining connected components of the computed graph are found by depth-first search (a). Geometric features are taken into account to remove static parts of the scene. Objects are covered by a bounding box. Center and velocity of each object are marked in their respective color (b).

5.2 ASSOCIATION FEATURES

Section 1.1 introduced multi-target tracking as the problem of associating detections to form tracklets, representing objects present in the scene. While object detections used for this thesis were introduced in the previous section, measures to compare similarity of two detections in different frames are presented in the following. While the detector score d_i translates to the costs C_i^{det} of the network formulation introduced in Section 3.2, the similarity measures translate to the costs for linkin detections $C_{i,j}^{li}$.

For this thesis, we make use of seven measures using pure monocular image information (*image features*, Fig. 5.6) and two additional measures exploiting 3D information for detections (*world features*). All measures return a similarity score $s_d \in [0, 1]$ for two detections, say, $\mathbf{x}_1$ and $\mathbf{x}_2$ in different frames. A score $s_d \to 1$ indicates a high similarity between both objects $\mathbf{x}_1$ and $\mathbf{x}_2$.

As image measures, we compute appearance-based cues namely color histogram similarity and cross-correlation. Furthermore, we compute positional-based cues as bounding box overlap, bounding box size similarity, positional similarity, optical flow overlap, and orientation similarity. These measures have proven useful for multi-target tracking applications [65, 75, 178, 180] or evaluation [77, 76, 62].

Appearance-based image cues can be used for comparing the similarity of objects in consecutive and non-adjacent frames and are explained in the following:

- COLOR HISTOGRAM SIMILARITY s_{color} computes a measure how well the histograms of both detections match. Therefore, the correlation based metric for every histogram bin i is computed by

$$s_{color}(H_1, H_2) = \frac{\sum_{(i)}(H_1(i) - \bar{H_1})(H_2(i) - \bar{H_2})}{\sqrt{\sum_{(i)}(H_1(i) - \bar{H_1})^2 \sum_{(i)}(H_2(i) - \bar{H_2})^2}} \tag{5.6}$$

where

$$\bar{H_k} = \frac{1}{n_p} \sum_{(j)} H_k(j)$$

and n_p the total number of histogram bins depending on the color depth of the image. Note that a normalization of the input data is performed by subtracting the average $\bar{H_k}$ in the numerator and computing the standard deviation σ_H in the denominator.

In case of color images with $n_{ch} = 3$, each channel is compared separately for both images and the average is computed as the final measure:

$$s_{color}(a_1, a_2) = \frac{\sum_{i=1}^{n_{ch}} d_h^{(i)}(H_1^{(i)}, H_2^{(i)})}{3} \tag{5.7}$$

Note that this formulation can be used for arbitrary color spaces such as *RGB*, *HSV*, *Lab*. We decided to use the Lab color space which was designed to approximate visual human perception and does not suffer as heavily as e.g., the RGB color space from illumination changes.

- Normalized Cross-Correlation $s_{x\text{-}corr}$ (template matching) is used as an appearance-based cue by correlating the bounding box region defined by detection $\mathbf{x}_1$ with the region specified by the detection $\mathbf{x}_2$ used as template for correlation:

$$s_{x\text{-}corr}(g_1, g_2) = \frac{1}{n} \sum_{u,v} \frac{(g_1(u,v) - \overline{g_1})(g_2(u,v) - \overline{g_2})}{\sigma_{g_1} \sigma_{g_2}} \tag{5.8}$$

where $g_i(\cdot,\cdot)$ returns the color value for a specified pixel (u,v), n the number of pixels, and $\bar{\cdot}$ being the respective mean. Changes in the scale of the object are considered by computing the correlation for a range of scales and taking the maximum value. To handle uncertainties in the bounding box position, the template is cropped to 80% of its original size.

Positional-based image cues are primarily suitable for comparing objects in consecutive frames. Bounding box overlap, size, and position in the image plane are subject to perspective changes for non-adjacent frames and moving objects, considerably changing for a longer time period. The five positional-based cues are defined as follows:

- BOUNDING BOX OVERLAP s_{over} is the intersection over union fraction for the (axis aligned) bounding boxes of both objects, requiring a minimum overlap

$$s_{over}(\mathbf{b}_1, \mathbf{b}_2) = \begin{cases} O & \text{if } O > 0.5 \\ 0 & \text{otherwise} \end{cases} \tag{5.9}$$

$$\text{with} \quad O = \frac{\mathbf{b}_1 \cap \mathbf{b}_2}{\mathbf{b}_1 \cup \mathbf{b}_2}$$

- OPTICAL FLOW OVERLAP s_{flow} is the number of flow vectors ζ pointing from bounding box $\mathbf{b}_1$ (tail) to $\mathbf{b}_2$ (head), normalized by the bounding box area of $\mathbf{b}_1$ ($w_{\mathbf{b}_1} \cdot h_{\mathbf{b}_1}$).

$$s_{flow} = \frac{|\{\mathbf{u}_i + \zeta_k \cap \mathbf{u}_j \mid \mathbf{u}_i \in \mathbf{b}_1, \mathbf{u}_j \in \mathbf{b}_2, \zeta_k \in \zeta\}|}{w_{\mathbf{b}_1} \cdot h_{\mathbf{b}_1}} \tag{5.10}$$

- BOUNDING BOX SIZE SIMILARITY s_{size} is the normalized sum of the absolute width and height difference

$$s_{size}(\mathbf{s}_1, \mathbf{s}_2) = 1 - \frac{1}{2}\left[\frac{|w_1 - w_2|}{\max(w_1, w_2)} + \frac{|h_1 - h_2|}{\max(h_1, h_2)}\right] \tag{5.11}$$

- POSITIONAL SIMILIARITY s_{loc} is the sum of absolute differences for the bounding box position normalized by the image dimensions.

$$s_{loc}(\mathbf{u}_1, \mathbf{u}_2) = 1 - \frac{1}{2}\left[\frac{|u_2 - u_1|}{w_{img}} + \frac{|v_2 - v_1|}{h_{img}}\right] \tag{5.12}$$

- ORIENTATION SIMILARITY s_α is the normalized cosine similarity of the absolute angular difference

$$s_\alpha(\alpha_1, \alpha_2) = \frac{\cos(|\alpha_1 - \alpha_2|) + 1}{2} \tag{5.13}$$

- 3D Overlap S_{over} is the bounding box overlap on the ground plane. Therefore, S_{over} is computed similarly to s_{over} in 2D.
- 3D Positional Similarity S_{loc} is the Euclidean distance of the 3D bounding box positions normalized by the maximum possible distance, which has an upper bound introduced by the maximum observation distance l_{obs}.

$$S_{loc} = \frac{\sqrt{X^2 + Y^2 + Z^2}}{\sqrt{(2l_{obs})^2 + l_{obs}^2}} \tag{5.14}$$

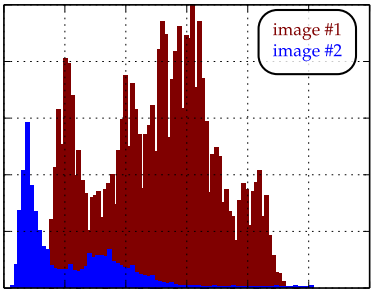
In the remaining part of this thesis, we use introduced abbreviations for the feature names summarized as follows:

color	color histogram similarity
x-corr	normalized cross-correlation (template matching)
over	bounding box overlap
flow	optical flow overlap
size	bounding box size similarity
loc	positional similarity
α	orientation similarity

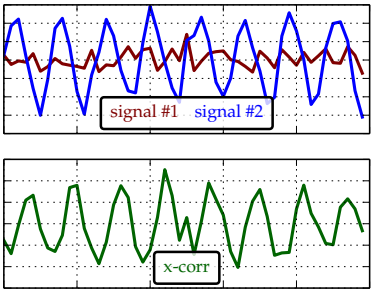
5.3 Learning Unary and Pair-Wise Costs

The previous sections introduced unary and pair-wise terms as (not necessarily normalized) scores where higher values indicate a more reliable detection or a higher similarity between detections. To use these scores within the optimization framework, a conversion into observation likelihood $\Psi(\mathbf{x}_i|\mathcal{T})$ and transition probabilities $\Psi_{link}(\mathbf{x}_{t_i+1}|\mathbf{x}_{t_i})$ is necessary.

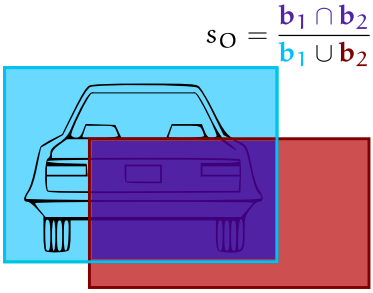
This is done by using logistic regression and manually labeled training data containing 8008 images with annotated bounding boxes for tracklets of 579 cars (28 253 labels), 167 pedestrians (11 628 labels), and 37 cyclists (1 972 labels).



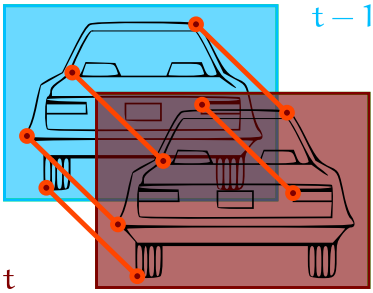
(a) Color Histogram Similarity



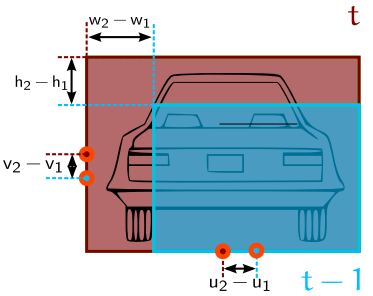
(b) Cross Correlation



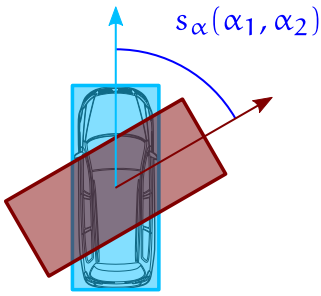
(c) Bounding Box Overlap



(d) Optical Flow Overlap



(e) Positional and Size Similarity



(f) Orientation Similarity

Figure 5.6: Association Features. Image-based ((a), (b)) and positional-based ((c), (d), (e), (f)) features used as input for association costs within the network flow formulation.

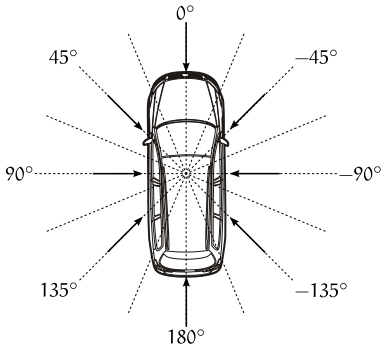


Figure 5.7: DISCRETE ORIENTATIONS FOR 2D DETECTIONS.

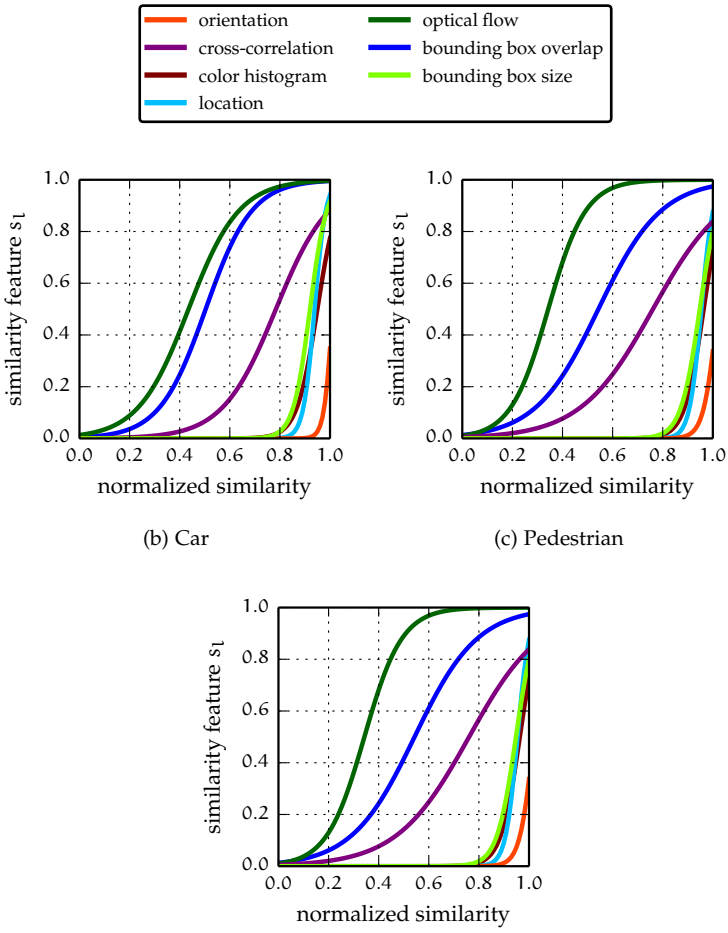
Logistic regression is used to predict the outcome of a dependent variable (e.g., transition probability) for given predictor variable s (score) of the logistic function

$$\bar{s} = \frac{1}{1 + e^{-\beta s}}. \tag{5.15}$$

A closed-form solution for this problem does not exist, hence regression coefficients are estimated as a maximum likelihood solution. The algorithm is exemplarily outlined as an iteratively reweighted least-squares approach (Appendix A.2), solving iteratively a linear problem by conjugate gradient approach. We used the similar and publicly available implementation of [137][4] to estimate the regression coefficients.

The learned logistic regression model for the object classes *car*, *pedestrian*, and *cyclist* are depicted as differently colored curves in Fig. 5.8 for the similarity features used in this thesis.

4 Source Code: http://scikit-learn.org



(b) Car

(c) Pedestrian

(d) Cyclist

Figure 5.8: LOGISTIC REGRESSION OUTPUT for quantitatively evaluated object classes. The normalized similarity measure (feature) is predicted by the learned logistic regression model resulting in the similarity feature s_l representing the unary term $\Psi_{li}(\mathbf{x}_{o_i}, \mathbf{x}_{o_{i+1}})$ for linking temporally adjacent detections (Eq. 3.4). The transformation for a similarity feature into costs is described in Section 6.2.1 for the costs defined as the negative logarithm of the initial maximization problem (Section 3.2). The orientation feature only allows for confirming an association while, e.g., the color or flow feature allows rejecting an association being therefore more discriminative.

6

EVALUATION

> *There are no facts, only interpretations.*
>
> FRIEDRICH NITZSCHE

For the experimental section of this thesis, we developed the KITTI vision dataset [76, 77] including object detection and tracking benchmarks. Sequences for this dataset were recorded from a VW Passat station wagon [98] using both color and gray-scale stereo camera images, laser scans, and a combined GPS/IMU system fusing high-precision GPS and acceleration/angular velocity information. Detailed information on the setup is given in Section 6.1.1. This dataset contains footage from inner-city, rural, and freeway traffic situations from the area of Karlsruhe, Germany (c.f. Fig. 6.1). Benchmarks for several tasks such as stereo and optical flow estimation, visual odometry, and in particular 3D object detection and tracking were extracted from this data and are available online.[1]

Our evaluation contains performance and run time comparisons for the different solvers for multi-target tracking as a min-cost flow problem as introduced in Chapters 3+4. The proposed methods are compared against other state-of-the-art multi-target tracking algorithms. Finally, qualitative results are shown.

1 The KITTI Vision Dataset http://www.mrt.kit.edu/software/datasets.html

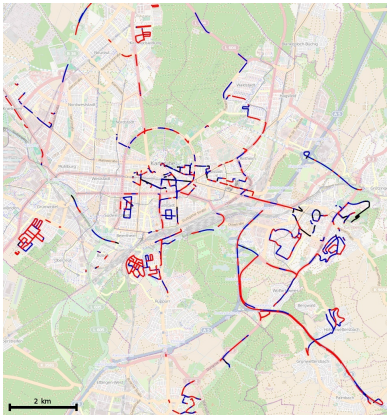


Figure 6.1: KITTI Recording Area. This figure shows the GPS tracks of our recordings in the metropolitan area of Karlsruhe in Germany. Colors encode the GPS signal quality: While green tracks have been recorded with highest precision using RTK corrections, blue denotes the absence of correction signals. The black runs have been excluded from our data set as no GPS signal was availble.

6.1 DATA COLLECTION

This section gives a short overview of the setup, recording, benchmark extraction, and data statistics of the KITTI vision dataset for the object detection and tracking part. More detailed information on calibration and other benchmarks is provided in [76, 77].

The object detection benchmark focuses on computer vision algorithms for object detection and 3D orientation estimation and provides accurate 3D bounding boxes for object classes such as cars, vans, trucks, pedestrians, cyclists and trams. This information is obtained manually by labeling objects in 3D point clouds produced by the laser scanner, and projecting them back into the image. This results in tracklets with accurate 3D poses, which can be used to asses the performance of algorithms for object detection and orientation estimation as well as tracking.

6.1.1 *Setup*

We equipped a standard station wagon with two color and two grayscale PointGrey Flea2 video cameras (10 Hz, resolution: 1392×512 pixels, opening: $90° \times 35°$), a Velodyne HDL-64E 3D laser scanner (10 Hz, 64 laser beams, range: 100 m), and a GPS/IMU localization unit with RTK correction signals (open sky localization errors < 5 cm).

The sensor setup is illustrated in Fig. 6.3 and the specifications of the sensors are provided as follows:

- $2 \times$ PointGrey Flea2 grayscale cameras (FL2-14S3M-C), 1.4 Megapixels, 1/2" Sony ICX267 CCD, global shutter
- $2 \times$ PointGrey Flea2 color cameras (FL2-14S3C-C), 1.4 Megapixels, 1/2" Sony ICX267 CCD, global shutter
- $4 \times$ Edmund Optics lenses, 4mm, opening angle $\sim 90°$, vertical opening angle of region of interest (ROI) $\sim 35°$
- $1 \times$ Velodyne HDL-64E rotating 3D laser scanner, 10 Hz, 64 beams, 0.09 degree angular resolution, 2 cm distance accuracy, collecting $\sim$ 1.3 million points/second, field of view: 360° horizontal, 26.8° vertical, range: 120 m
- $1 \times$ OXTS RT3003 inertial and GPS navigation system, 6 axis, 100 Hz, L1/L2 RTK, resolution: 0.02m / 0.1°

Note that the color cameras lack in terms of resolution due to the Bayer pattern interpolation process and are less sensitive to light. This is the reason why we use two stereo camera rigs, one for grayscale and one for color. The baseline of both stereo camera rigs is approximately 54 cm. The distance between color and grayscale cameras is minimized (6 cm). We believe this is a good setup since color images are very useful for tasks such as segmentation and object detection while their grayscale counterparts are more suitable for stereo matching and optical flow estimation. The trunk of our vehicle houses a PC with two six-core Intel XEON X5650 processors and a shock-absorbed RAID 5 hard disk storage with a capacity of 4 Terabytes. Our computer

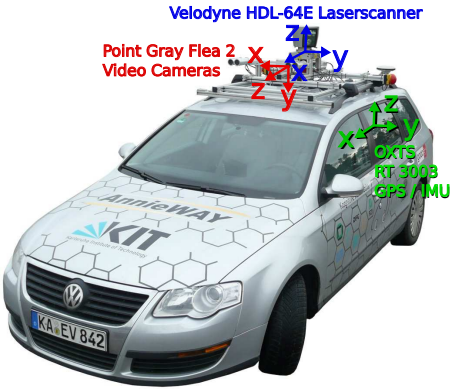


Figure 6.2: Recording Platform. Our VW Passat station wagon is equipped with four video cameras (two color and two grayscale cameras), a rotating 3D laser scanner and a combined GPS/IMU inertial navigation system.

runs Ubuntu Linux (64 bit) and a real-time database [83] to store the incoming data streams.

We use a Velodyne HDL-64E unit, as it is one of the few sensors available that can provide accurate 3D information from moving platforms. To compensate egomotion in the 3D laser measurements, we use the position information from our GPS/IMU system.

In order to synchronize the sensors, we use the timestamps of the Velodyne 3D laser scanner as a reference and consider each spin as a *frame*. We mounted a reed contact at the bottom of the continuously rotating scanner, triggering the cameras when facing forward. This minimizes the differences in the range and image observations caused by dynamic objects. Unfortunately, the GPS/IMU system cannot be synchronized that way. Instead, as it provides updates at 100 Hz, we collect the information with the closest timestamp to the laser scanner timestamp for a particular frame, resulting in a worst-case time difference of 5 ms between a GPS/IMU and a camera/Velodyne data package. Note that all timestamps are provided such that positioning information at

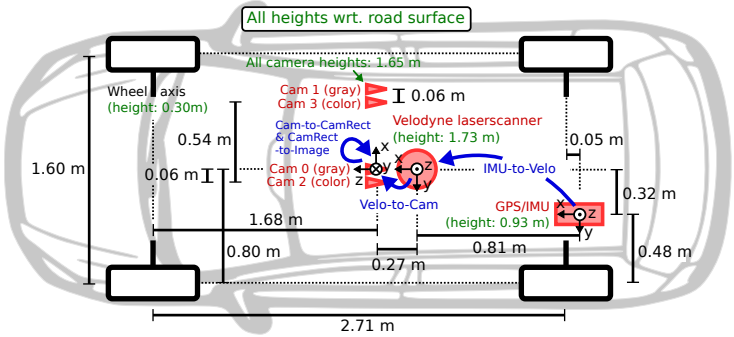


Figure 6.3: SENSOR SETUP. This figure illustrates the dimensions and mounting positions of the sensors (red) with respect to the vehicle body. Heights above ground are marked in green and measured with respect to the road surface. Transformations between sensors are shown in blue.

any time can be easily obtained via interpolation. All timestamps have been recorded on our host computer using the system clock.

For calibrating the cameras intrinsically and extrinsically, we use the approach proposed in [78]. Note that all camera centers are aligned, i.e., they lie on the same x/y-plane. This is important as it allows us to rectify all images jointly.

We have registered the Velodyne laser scanner with respect to the reference camera coordinate system (camera 0) by initializing the rigid body transformation using [78]. Next, we optimized an error criterion based on the Euclidean distance of 50 manually selected correspondences and a robust measure on the disparity error with respect to the 3 top performing stereo methods in the KITTI stereo benchmark [77]. The optimization was carried out using Metropolis-Hastings sampling.

For registering the IMU/GPS with respect to the Velodyne laser scanner, we first recorded a sequence with an '∞'-loop and registered the (untwisted) point clouds using the Point-to-Plane ICP algorithm. Given two trajectories this problem corresponds to the well-known hand-eye calibration problem which can be solved using standard tools [88].

6.1.2 *Object Annotation*

Ground truth for object detection, orientation estimation, and tracking was created manually by annotating the recorded image sequences. Towards this goal, we created a special purpose labeling tool, which displays 3D laser points as well as the camera images to increase the quality of the annotations. This tool enables annotators to place and orient 3D bounding boxes within the displayed point cloud, where an estimated ground plane is shown for reference. Bounding boxes are placed in key frames and motion between these frames can be interpolated either linearly (even restricted on the ground plane) or using an ICP-based tracker. To gain maximum precision of the annotations, perspective and orthogonal projections of the point cloud as well as the zoomed part of the image are available to the user for verification. The user interface is exemplarily shown in Fig. 6.4. Following [62], we asked the annotators to additionally mark each bounding box as either visible, semi-occluded, or fully occluded, while truncation is computed automatically. Benchmark users can use a set of tools for visualization and evaluation. A tracklet viewer is shown in Fig. 6.5

To fulfill this labeling task, we hired a set of annotators, and asked them to assign tracklets in the form of 3D bounding boxes to objects such as cars, vans, trucks, trams, pedestrians and cyclists. Unlike most existing benchmarks, we do not rely on online crowd-sourcing to perform the labeling. Consequently, our benchmark offers accurate 3D position and orientation information for each object. Statistics of our labeling effort are discussed in the following section.

6.1.3 *Data Statistics and Quality*

In total, we collected $\sim$ 3 TB of data from which we select a representative subset to evaluate each benchmarking task. Our 3D OBJECT DETECTION AND ORIENTATION estimation benchmark is chosen according to the number of non-occluded objects in the scene, as well as the entropy of the object orientation distribution. High entropy is desirable in order to ensure diversity. This allows us

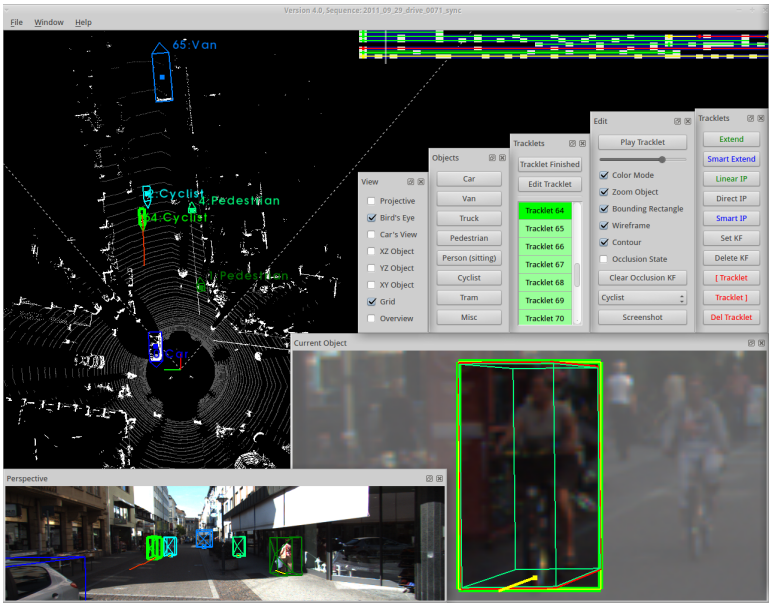


Figure 6.4: TRACKLET ANNOTATION UTILITY. The point cloud (here in an orthogonal bird's eye view) contains user placed labels for different classes. For the selected tracklet (here: #4), the already labeled trajectory for future time steps is displayed in red to verify the object orientation. The projection of the label in the image and a zoomed version of the selection are displayed as well. Occlusion states are color-coded in the tracklet bar (top) and the bounding box of the selected tracklet.

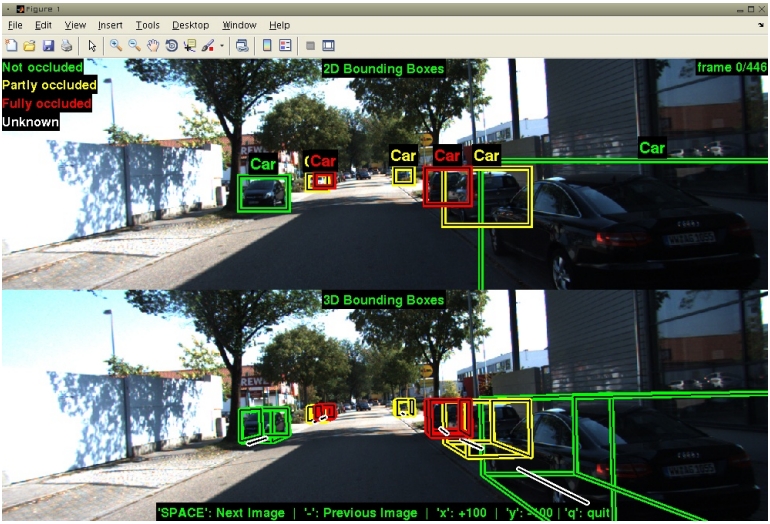


Figure 6.5: DEVELOPMENT KIT. The KITTI development kit contains besides other tools for visualization and evaluation a viewer for working with tracklets.

to automatically select a smaller subset of images with a large amount of non-occluded objects and a well distributed set of orientations. Towards this goal we utilize a greedy algorithm: We initialize our dataset $\mathcal{D}$ to the empty set $\emptyset$ and iteratively add images using the following rule

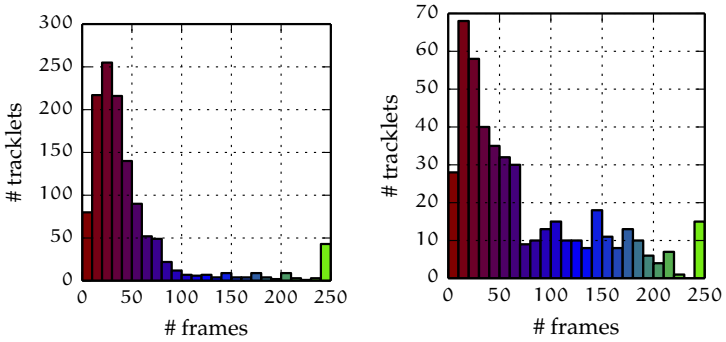
$$\mathcal{D} \leftarrow \mathcal{D} \cup \underset{\mathcal{I}}{\operatorname{argmax}} \left[\alpha \cdot \mathrm{noc}(\mathcal{I}) + \frac{1}{C} \sum_{c=1}^{C} H_c\left(\mathcal{D} \cup \mathcal{I}\right) \right] \tag{6.1}$$

where $\mathcal{D}$ is the current set, $\mathcal{I}$ is an image from our dataset, $\mathrm{noc}(\mathcal{I})$ stands for the number of non-occluded objects in image $\mathcal{I}$ and C denotes the number of object classes. H_c is the entropy of class c with respect to orientation (we use 8/16 orientation bins for pedestrians/cars). We further ensure that images from one sequence do not appear in both training and test set.

For the 3D TRACKING benchmark, sequences in training and test set are chosen in order to balance sequence types in both sets, number of frames, and number of objects. A qualitative

overview is given in Fig. 6.11. The number of labels, tracklets, and images for training and test sequences is summarized in Table 6.1. Note that a complete balanced test and training set is not possible in order to guarantee completely exclusive sequences in both sets. For the same reason, training sequences of other benchmarks cannot be added to the tracking test set. Considering these restrictions, we decided for a more challenging evaluation by selecting a slightly larger test set. Fig. 6.6 shows the tracklet length per class of all benchmark sequences as a histogram. For both of the two predominant classes car and pedestrian, most tracks are visible for a less than 5s (50 frames) such as oncoming or crossing traffic. There are only a few tracks visible for more than 25s (250 frames) such as preceding traffic.

To give a more comprehensive overview of the annotated sequences and extracted benchmarks, this section provides statistics on object occurence as well as geometry and orientation (Fig. 6.7). The total occurences of the individually labeled classes are shown in Fig. 6.8, justifying that the benchmarks focus on the predominant classes car and pedestrian for tracking and, additionally, cyclists for object detection and orientation estimation. The remaining classes do not contain enough labels for training procedures or even a quantitatively sound evaluation. Object labels per class and image are summarized in Fig. 6.8, whereas statistics on object motion are depicted in Fig. 6.9. A detailed evaluation of the different scene types such as inner-city, rural, or freeway traffic can be found in Fig. 6.10.



(a) Car (b) Pedestrian

Figure 6.6: TRACKLET LENGTH. This figure shows the tracklet length in frames for the most predominant classes (captured at 10fps). Only a few tracklets are visible for more than 250 frames such as preceding traffic.

	TRAINING	TESTING
# LABELS		
car	28 253	38 660
pedestrian	11 628	24 113
cyclist	1 972	4 919
# TRACKLETS		
car	579	670
pedestrian	167	292
cyclist	37	80
SEQUENCES		
# images	8 008	11 095
# sequences	21	29
images/sequence	381 ±262	382 ±265

Table 6.1: KITTI TRACKING BENCHMARK STATISTICS. For training and test set of this benchmark, the number of object labels (in images), tracklets (in sequences), and the average sequence length is shown.

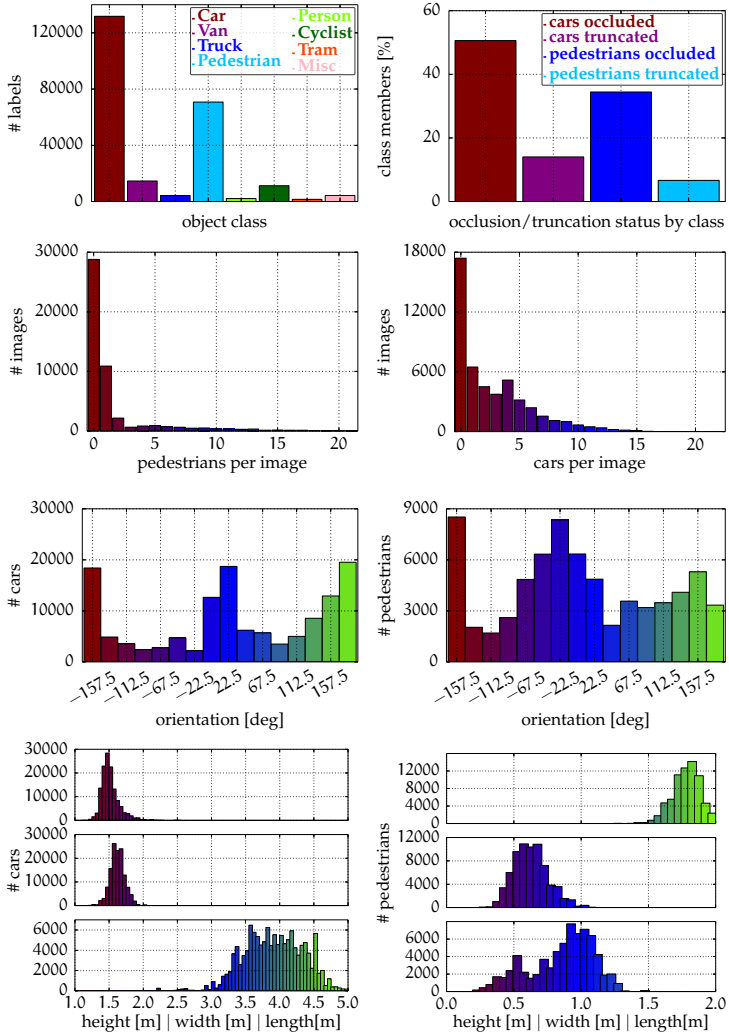


Figure 6.7: OBJECT OCCURRENCE AND OBJECT GEOMETRY STATISTICS. This figure shows (from left to right and top to bottom): The different types of objects occurring in all sequences (including non-released ones), the power-law shaped distribution of the number of instances within an image and the orientation histograms and object size distributions for the two most predominant categories CARS and PEDESTRIANS.

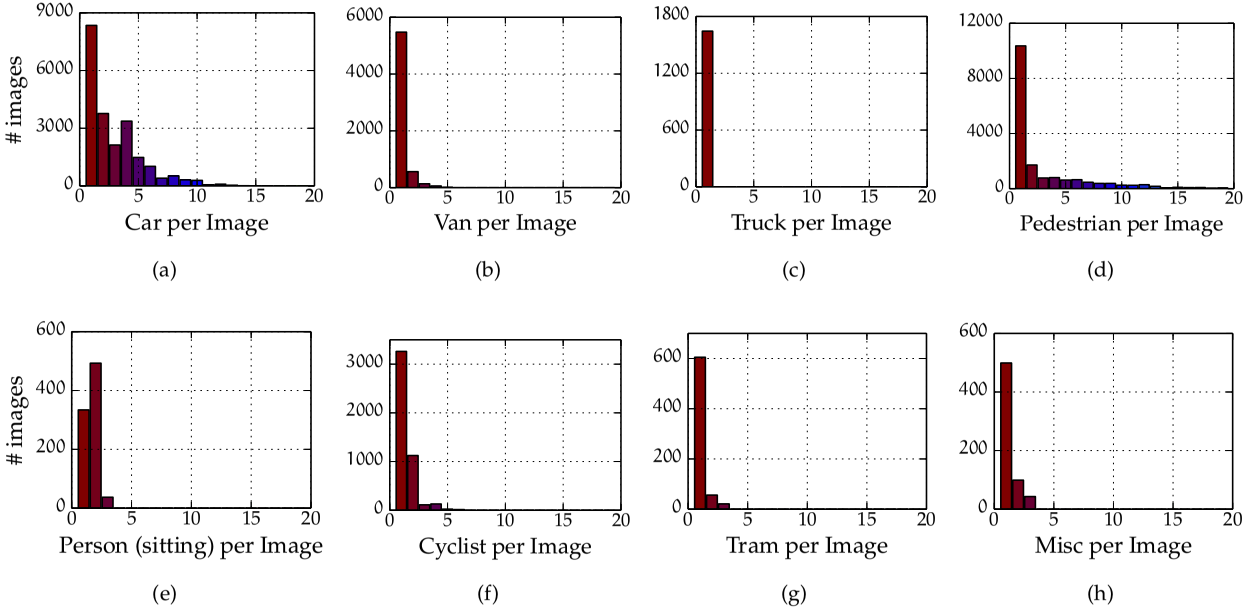


Figure 6.8: Number of Object Labels per Class and Image. This figure shows how often an object occurs in an image. Since our labeling efforts focused on cars and pedestrians, these are the most predominant classes here.

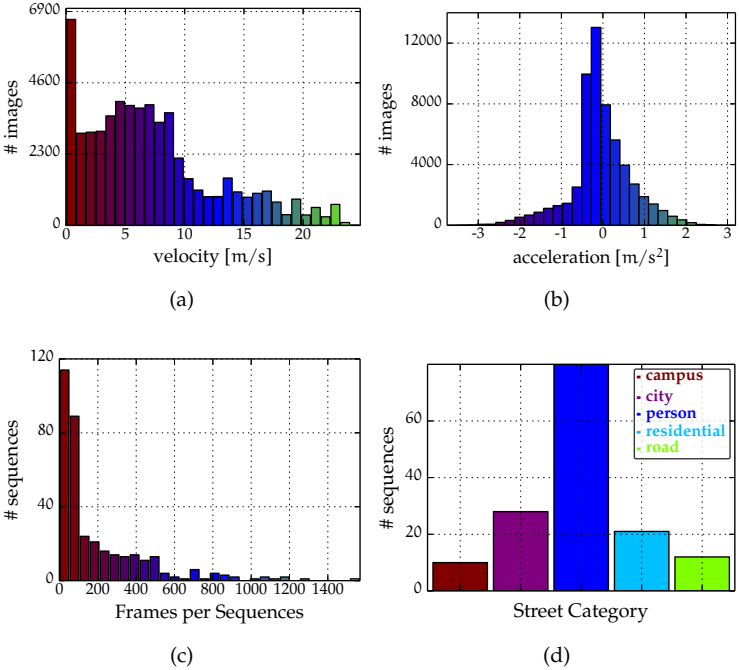


Figure 6.9: EGOMOTION, SEQUENCE COUNT AND LENGTH. This figure shows (from-left-to-right) the egomotion (velocity and acceleration) of our recording platform for the whole dataset. Note that we excluded sequences with a purely static observer from these statistics. The length of the available sequences is shown as a histogram counting the number of frames per sequence. The rightmost figure shows the number of frames/images per scene category.

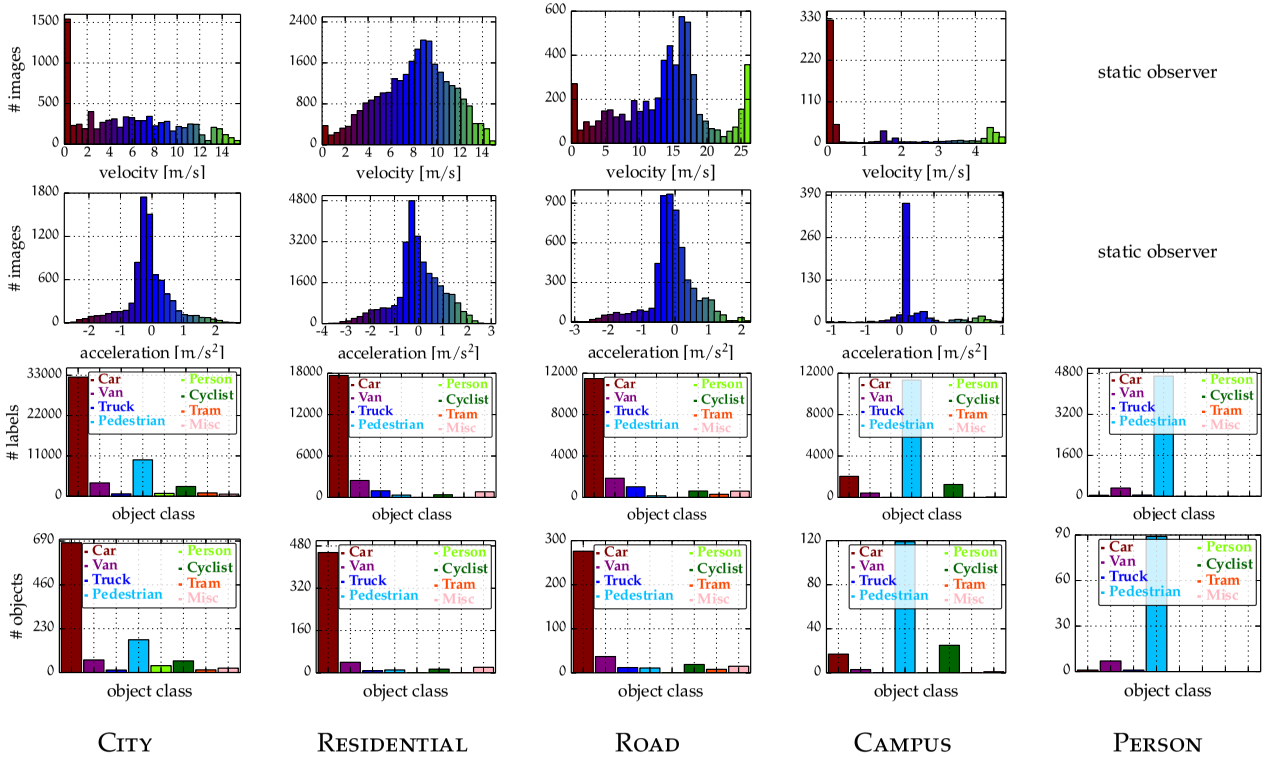


Figure 6.10: Velocities, Accelerations and Number of Objects per Object Class. For each scene category we show the acceleration and velocity of the mobile platform as well as the number of labels and objects per class. Note that sequences with a purely static observer have been excluded from the velocity and acceleration histograms as they do not represent natural driving behavior. The category 'Person' has been recorded from a static observer.

Figure 6.11: ILLUSTRATION OF THE DATASET. This figure exemplarily shows 18 images selected from the 50 sequences compiling the KITTI tracking benchmark. For all sequences grayscale and color stereo images, laser point clouds, GPS/IMU information, and calibration matrices are available. Note the wide variety and complexity of scene types, quantitatively stated in Fig. 6.9d.

6.2 EXPERIMENTAL RESULTS

In the following, we give a comprehensive evaluation of the runtime performance of the methods proposed in this thesis. A detailed performance analysis gives insights into the portion of the total runtime for the different solver elements and compares to regular as well as approximative batch methods and a standard baseline. Additionally, average-case complexity shows the effectiveness for all algorithms to demonstrate benefits for online scenarios. Runtime performance is evaluated using the training sequences compiled in the KITTI tracking benchmark. Moreover, tracking performance is evaluated against state-of-the-art methods on both the challenging test sequences of the KITTI benchmark (c.f. Section 6.1) and for a comparative evaluation on the PETS2009 [56] dataset. Furthermore, we gain insights into the applicability of the association features by evaluating the training results of the algorithms. Again, we used the KITTI training set to optimize the parameters for feature scaling and the subsequent evaluation.

We use the following abbreviations for the different shortest path solvers within the SSP framework:

Bellman-Ford	SSP – Bellman-Ford
Dijkstra	SSP
dynamic Dijkstra	dSSP
online-dynamic Dijkstra	odSSP
memory-bounded odDijkstra	mbodSSP

6.2.1 *Graph Building*

For object detection, we make use of the deformable part-based model (*DPM*) [65] in combination with the pre-trained object detection model provided with the KITTI benchmark and our moving objects detector as outlined in Section 5.1. We convert the detector score for each bounding box d_i into the unary cost C_i^{det} using the provided training data for obtaining β

$$C_i^{det} = \frac{1}{1 + e^{\beta d_i}} . \tag{6.2}$$

To encode association costs, we use seven different pairwise similarity features $\bar{\mathbf{s}} = \{\bar{s}_i\}$: bounding box overlap, orientation similarity, positional similarity, bounding box size similarity, flow overlap, color histogram similarity as well as cross-correlation as discussed in Section 5.2. Similar to detection, here, $\bar{s}_i$ denotes the output of a logistic function which has been learned via logistic regression from training data and ranges $[0, 1]$ using the association feature s_i as input data. The detection/association cost for each edge (u, v) is then defined as

$$C_{u,v} = ((\mathbf{1} - \bar{\mathbf{s}}) + \mathbf{o})^\top \mathbf{w}\,, \tag{6.3}$$

where $\mathbf{o}$ denotes an offset and $\mathbf{w}$ the scale. Note that the offset is required to allow for negative as well as positive costs. All parameters $\beta, \mathbf{o}, \mathbf{w}$ have been obtained using grid search on the training set and kept fix to the values in Table 6.2 during our experiments. The parameter for the logistic function in Eq. 6.2 was set to $\beta = 8.4$.

Learning the scaling values for unary and pair-wise similarity features allows insights into the performance of the features. Due to the huge feature space and the particular problem formulation which does not allow for well-known learning techniques such as structured prediction [147], we used a 1D grid search strategy fixing parameters for all but one feature. The resulting performance is shown in Fig. 6.12+6.13. As a ranking criterion r_g, we used a weighted sum of five different metrics

$$r_g = a_1 \mathrm{MT} + a_2 \mathrm{PT} + a_3 \mathrm{F1} + a_4\,\mathrm{precision} + a_5 \mathrm{MOTA} \tag{6.4}$$

where MT/PT *mostly tracked/partly tracked* are the metrics introduced in Section 2.2, F1 is the harmonic mean of precision and recall

$$\mathrm{F1} = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}\,, \tag{6.5}$$

and precision and recall are defined as follows

$$\mathrm{precision} = \frac{tp}{tp + fp} \qquad \mathrm{recall} = \frac{tp}{tp + fn}\,. \tag{6.6}$$

FEATURE	en	ex	det	α	over	size	color	x-corr	loc	flow
w_l	5.0	5.0	21.5	1.5	5.0	1.0	9.25	6.5	1.25	8.0
o_l	0.0	0.0	-0.5	1.0	-0.75	0.0	-0.8	-0.01	0.625	-0.4

Table 6.2: MODEL PARAMETERS. This table shows the weight w_l and offset o_l for each feature l which we kept fixed during all our experiments. The robustness of each feature l against variation of its weight value w_l is shown in Fig. 6.12+6.13 for different values of w_l.

We chose the weight coefficients a_i according to the observed weaknesses of the tracker output and used the following weights:

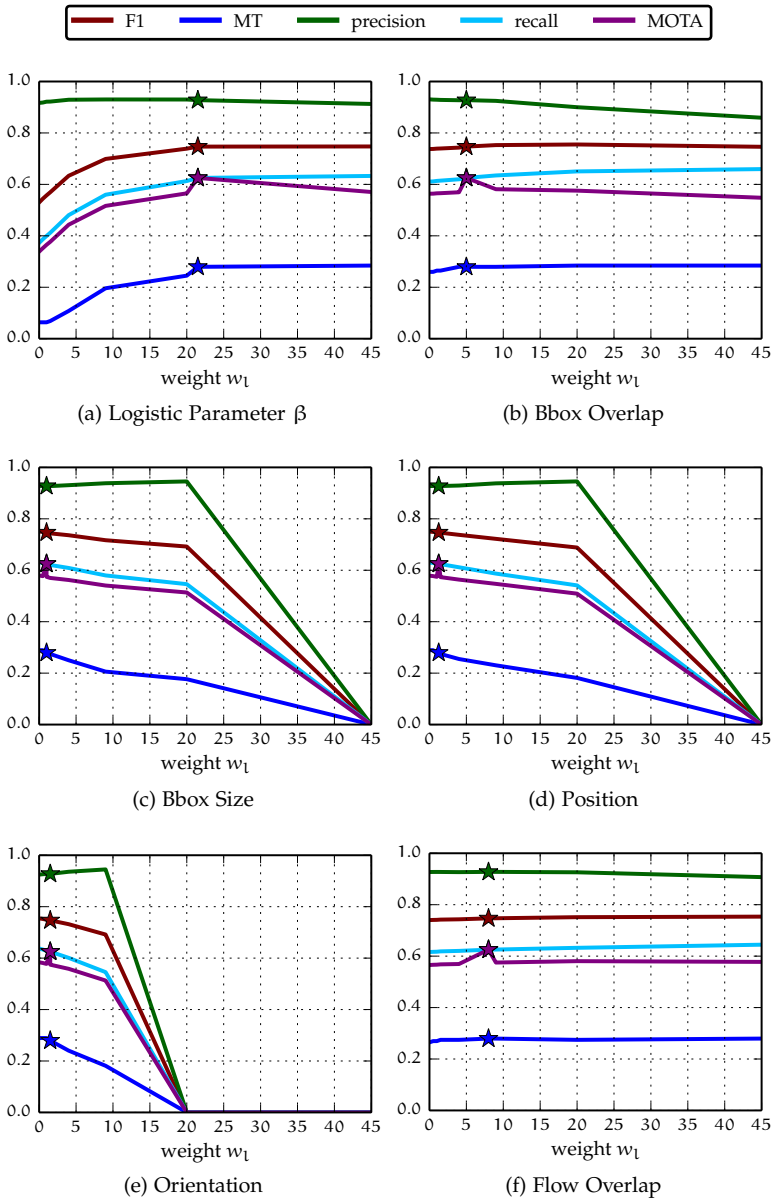
$$a_1 = 4/10, \quad a_2 = 1/10, \quad a_3 = 1/10, \quad a_4 = 2/10 \quad a_5 = 2/10\,.$$

Bounding box and flow overlap turned out to be the most robust positional features, almost not influenced by changing their weight. The same observation can be made for color histogram similarity and cross correlation. In contrast, bounding box size, positional and orientation similarity can only reject a wrong hypothesis (Consider two different but similarly oriented cars.). Therefore, changing the respective weights has significant influence on tracking performance.

6.2.2 *Average-case Complexity Analysis*

We discussed the response of the standard batch algorithms in their runtime to changes in the input size of the tracking problem, i.e., the sequence length, using big $\mathcal{O}$ notation in Chapter 3. While these theoretical considerations allow classifying algorithms, for the distributed algorithms proposed in this thesis achievable runtime reductions are not considered. Moreover, the particular graph structure (which mainly remains a DAG during optimization) prevents the batch solvers reaching their worst-case complexity.

Consequently, we study average-case complexity for the methods proposed in this thesis and the respective baselines. This allows judging the efficiency of different algorithms in practice.



(a) Logistic Parameter β

(b) Bbox Overlap

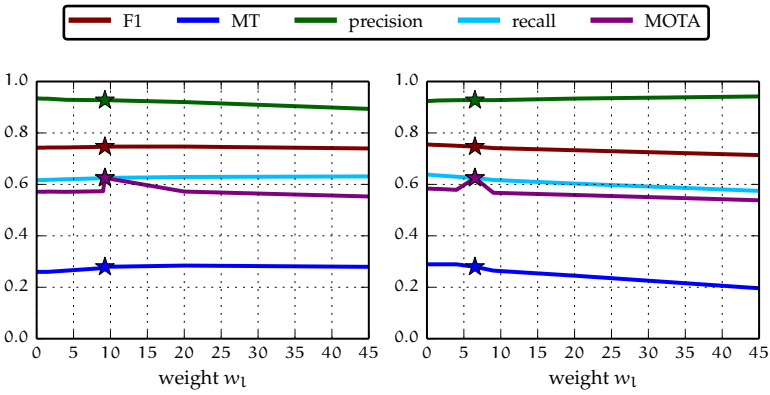
(c) Bbox Size

(d) Position

(e) Orientation

(f) Flow Overlap

Figure 6.12: ROBUSTNESS OF OBJECT DETECTION SCORE AND POSITIONAL FEATURES AGAINST VARIATION OF PARAMETERS. This figure shows the performance for the logistic parameter β and different weight values on the KITTI training set using dSSP. While varying one parameter, all other parameters are kept fixed (c.f. Table 6.2).



(b) Color Histogram (c) Cross-Correlation

Figure 6.13: Robustness of Appearance-based Features against Variation of Parameters. This figure shows the performance dSSP for different weight values on the KITTI training set. While varying one parameter, all other parameters are kept fixed (c.f. Table 6.2).

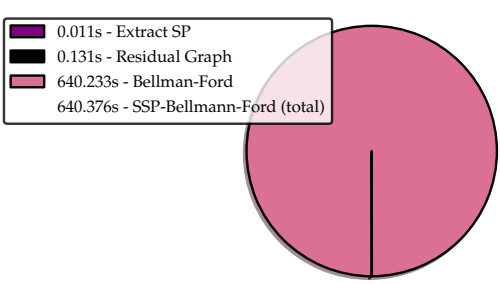
We used the KITTI training set sequences as samples for the evaluation, which provide a wide variety of different scene types, crowdedness, and object classes. For a fair comparison, all algorithms are implemented in python, using the same datastructures, and a single CPU core of an Intel Core i7-3740QM@2.7GHz processor. All standard algorithms are implemented using the textbook implementation described in [46]. For all our experiments, we used the previously discussed DPM detector for cars with a threshold $d_i = -0.3$.

Batch Methods. For comparing the total runtime of the batch solvers in Fig. 6.14, optimization is performed for all sequences and runtime is summed. In particular, the different methods for finding a shortest path within the SSP framework are evaluated. The quadratically growing response to the input sequence length of the Bellman-Ford algorithm (a) is reflected by a total runtime one order of magnitude larger than by applying Dijkstra's algorithm (b), even for the well-posed DAG problem avoiding the worst-case complexity considerably (Fig. 6.16). In the following,

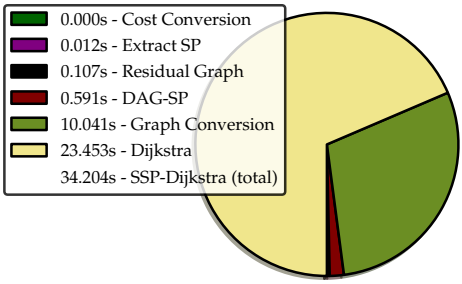
we only refer to the BF implementation exploiting an early termination if no changes take place anymore during optimization. Although applying Dijkstra's algorithm requires converting the graph after finding a shortest path (i.e., encoding the arborescence), the additional, simple computations do not affect the overall runtime adversarially.

Applying the proposed dynamic modifications to Dijkstra's algorithm (c) reduces the overall runtime almost by factor two on average. In particular, the dynamic algorithm reduces the runtime for finding shortest paths by factor three. Note that computing and updating the residual graph is merged into *dDijkstra Update* to initialize the minimum-priority queues for the dynamic case, introducing a marginal overhead. Additionally, we created DAGs with the typical structure of the introduced tracking problem but with arbitrary edge weights. This experiment (Fig. 6.15) poses a more complex problem since there are no predominant trajectories present. While for very short sequences the speed-ups achieved by dSSP are marginal, an increasing sequence length shows the advantages of our dynamic approach.

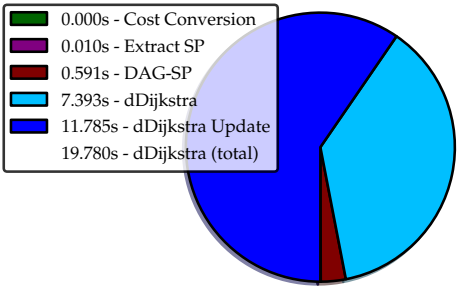
ONLINE AND APPROXIMATIVE ALGORITHMS. Considering an online scenario, we used all solvers to compute a solution for every single time step for all KITTI training sequences. We only evaluated frames which existed for at least three sequences, resulting in a maximum evaluation length of 803 frames (Fig. 6.17). While our proposed dynamic batch solver outperforms a regular SSP implementation, all optimal solvers respond with an increased runtime to larger input sequences as already theoretically discussed in Section 4.2. Extending dSSP to tackle an online scenario yields significant speed-ups up to one order of magnitude for well-posed problems. This is indicated by the shown variance over the test sequences in Fig. 6.17. However, the offline strategy using a minimum-priority queue is already very efficient. Therefore, the introduced overhead for maintaining such a queue in the online version on average leads to a slightly increased runtime. The main advantages of odSSP are achieved in the memory-bounded case. Most importantly, unlike any other of the discussed solvers, our memory-bounded algorithm's com-



(a) SSP – Bellman-Ford



(b) SSP – Dijkstra



(c) SSP – dDijkstra

Figure 6.14: Detailed Runtime Evaluation for Batch Scenarios. This figure gives a runtime comparison for the different components of the batch solvers using the KITTI training sequences as input data with a sequence length of 381 frames on average (Table 6.1). While the Bellman-Ford algorithm suffers from its complexity class (a), even the introduced overhead by applying Dijkstra's algorithm (cost conversion) does not affect the runtime reductions (b). Applying the dynamic modifications to Dijkstra's algorithm (c), the total runtime can be further reduced by factor two, although a marginal overhead is introduced by initializing the minimum-priority queues in the update step (computing and converting the residual graph).

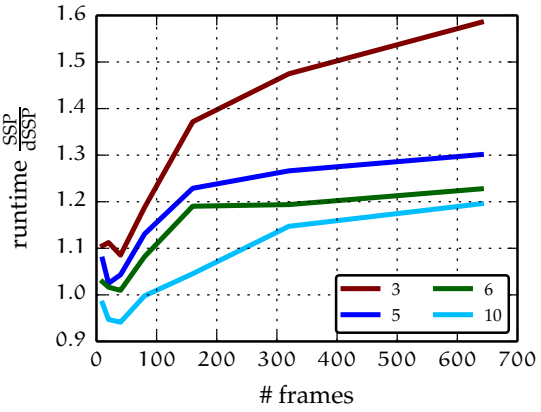


Figure 6.15: RUN TIME FOR RANDOM DAGS. This figure shows the speed-ups our dynamic algorithm (dSSP) achieves over a regular SSP implementation in a relative fashion. The experiment was carried out for different lengths of a random DAG and a different number of shortest paths (differently colored curves) and averaged over four runs.

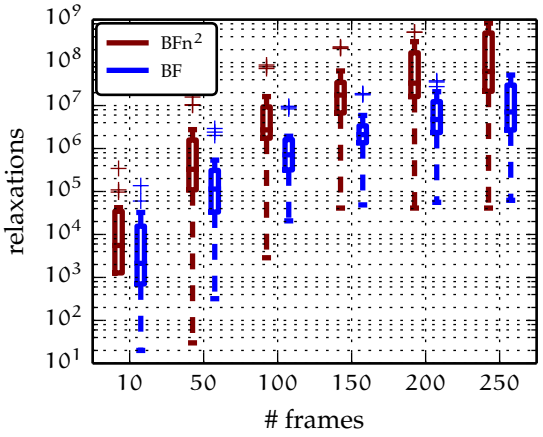


Figure 6.16: RELAXATIONS FOR BELLMAN-FORD ALGORITHM. As expected, the worst-case complexity for the Bellman-Ford algorithm (BFn^2) is never reached. Tracking graphs under consideration are not fully connected and forward pointing edges dominate the graph structure even for residual graphs. This results in one order of magnitude less relaxations for an early termination of the algorithm.

plexity is constant and not affected by the size of the sequence. Additionally, it can be applied to unlimited data streams as its memory requirements are also constant (and only dependent on the number of detected objects per frame) (Fig. 6.18).

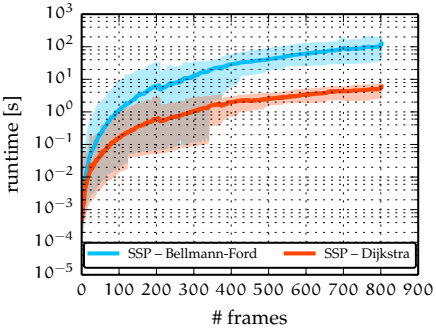
When comparing our optimal odSSP to the approximative DP solution of [138], we find that the proposed approach requires slightly more runtime on average while still being optimal. However, for sequences allowing to fully exploit the dynamic strategy, our optimal solver is on par with the approximative solution. In contrast, our memory-bounded solver mbodSSP outperforms DP by approximately two orders of magnitude while at the same time being more accurate (see Table 6.3).

To evaluate runtime and memory consumption for a long-term scenario, we used a sequence with almost 10^6 frames. Fig. 6.18 shows the results as a log-log plot. While all optimal solvers and the approximate DP solution grow infinite in both memory usage and computational time, our memory-bounded algorithm requires constant resources. Note that the memory consumption is computed idealized and based on the size of the used data types to avoid artifacts introduced by pythons garbage collection.

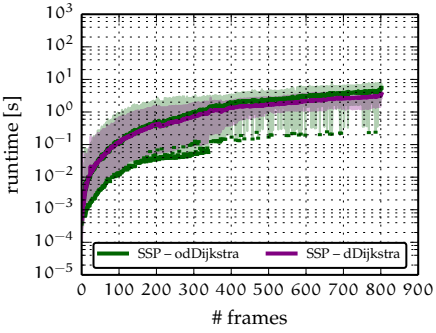
Memory-bounded Parametrization. Finally, we evaluate the runtime of mbodSSP for different values of τ (Fig. 6.19) on all frames of the KITTI training dataset. The runtime comparison shows that for a reasonable value of $\tau = 10$, our non-optimized Python implementation requires less than 10ms which is sufficient for many real-time applications. Furthermore, independent of the selected value of τ, the runtime always converges to a constant value, which is still real-time capable even for a longer history.

6.2.3 *Optimal Data Association Performance*

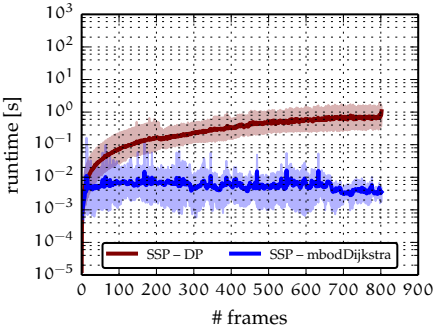
Tracking performance is evaluated using the metrics discussed in Section 2.2 and summarized in Table 2.3. Since our 3D moving object detector disregards static parts of the scene which are a considerable part of the benchmark, we only evaluate the 2D case quantitatively. We evaluate the proposed algorithms against state-



(a) Standard Algorithms



(b) Proposed Algorithms



(c) Approximative Algorithms

Figure 6.17: RUN TIME COMPARISON FOR AN ONLINE SCENARIO. This figure compares all solvers using the KITTI training set (showing the mean runtime over all sequences and the respective variance) and the object class car. We used a window size of $\tau = 10$ for mbodSSP and $|\mathcal{C}| = 4$ for (mb)odSSP and evaluated all methods on a fully connected network without pruning any edges or an increased detector threshold.

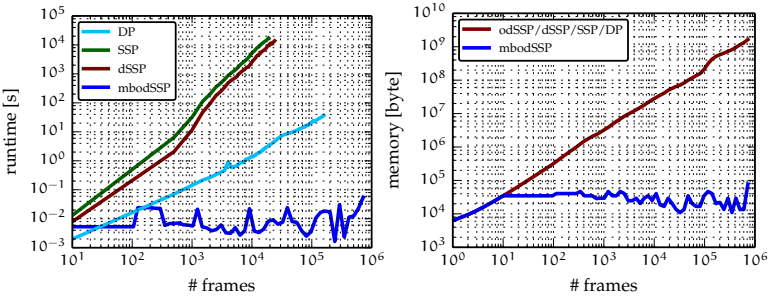


Figure 6.18: Run Time and Memory Comparison. Comparing all solvers using one long sequence, this figure shows the mean runtime and idealized memory consumption for every solver.

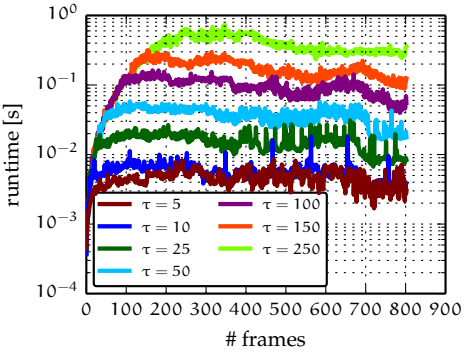


Figure 6.19: Run Time for mbodSSP Parameter Variation. We evaluated the complexity of mbodSSP for different values of its history length τ on the whole KITTI training set for at least three different scenarios for each frame.

of-the-art methods on the challenging test sequences of the KITTI tracking benchmark containing 670 car trajectories. Moreover, we evaluate the methods on the PETS2009 [56] tracking benchmark as a dataset used in the long term.

COMPARISON TO STATE-OF-THE-ART ON KITTI. First, we compare the proposed dSSP and mbodSSP algorithm against four baselines [7, 120, 138, 75] as well as the pairwise optimal Hungarian method (*HM*) on the challenging KITTI dataset (see Fig. 6.11 for an illustration). Note that all optimal solvers (SSP – Bellman-Ford, SSP, dSSP, odSSP) obtain identical results and are therefore represented by dSSP in the evaluation. As shown in Table 6.3 + 6.4, the optimal (batch) algorithms outperform current state-of-the-art, where the proposed dSSP algorithm solved the task most efficiently (Section 6.2.2). In our experiments, we made use of a relatively low threshold $d_i = -0.3$ for the DPM object detector to avoid early pruning. Moreover, this allows evaluating each method with respect to outlier rejection performance. Note that our method attains the best performance with respect to mostly tracked trajectories (*MT*) while only exhibiting a slightly higher false alarm rate (*FAR*) than the other methods. Also note the little loss in performance when running mbodSSP for a window length of $\tau = 10$. Compared to the non-optimal DP solution, mbodSSP achieves higher performance, especially in terms of identity switches and fragmentations. The low detection threshold used for this comparison results in outlier rejection as a major task during optimization. The method of [7] could not handle these outliers, so that a higher threshold of $d_i = 0.0$ was used in this case. Generally, the relatively low F1 score of all methods is mainly caused by the challenging scenario with many small and often partially occluded objects. Qualitative results are discussed in Section 6.2.5.

COMPARISON TO STATE-OF-THE-ART ON PETS2009. Additionally, we evaluate our method on the commonly used PETS2009 dataset for sequence *S2.L1*. We used the detections and ground truth provided by the authors of [7], since the annotations of the benchmark authors are not publicly available. The PETS benchmark

	HM	[7]	[120]	[138]	[75]	mbodSSP	dSSP
MOTA	0.42	0.35	0.48	0.44	0.52	0.52	**0.54**
MOTP	0.78	0.75	0.77	0.78	0.78	0.78	0.78
MODA	0.42	0.36	0.48	0.52	0.52	0.52	**0.54**
MODP	0.53	0.54	0.58	0.54	0.58	**0.59**	**0.59**
Recall	0.43	0.50	0.54	0.46	0.54	**0.56**	**0.58**
Prec.	**0.97**	0.77	0.90	0.96	0.95	0.93	0.94
F1	0.60	0.61	0.67	0.62	0.69	**0.70**	**0.71**
FAR	**0.048**	0.46	0.18	0.053	0.083	0.14	0.11
MT	0.077	0.11	0.14	0.11	0.14	**0.15**	**0.21**
ML	0.42	0.34	0.34	0.39	0.35	**0.30**	**0.27**
IDS	12	223	125	2738	33	**0**	**7**
Frag.	578	624	**401**	3241	540	708	717

Table 6.3: Comparison of our proposed methods to four state of the art methods and a HM baseline implementation on KITTI (car) [77]. Bold entries for the proposed methods indicate that this method performed better than the state-of-the-art.

	[6]	[7]	EKF [120]	[120]	mbodSSP	dSSP
MOTA	0.81	**0.96**	0.68	0.91	0.89	0.91
MOTP	0.76	0.79	0.77	0.80	**0.87**	**0.87**
MODA	n/a	n/a	n/a	n/a	0.89	0.91
MODP	n/a	n/a	n/a	n/a	0.87	0.87
Recall	n/a	n/a	0.70	**0.92**	0.90	**0.92**
Precision	n/a	n/a	0.98	0.98	**0.99**	**0.99**
F1	n/a	n/a	0.82	**0.95**	0.94	**0.95**
FAR	n/a	n/a	0.08	0.07	**0.057**	0.067
MT	0.83	**0.96**	0.39	0.91	0.89	0.89
ML	**0.0**	**0.0**	0.04	0.04	**0.0**	**0.0**
IDS	15	10	25	11	**7**	23
Frag.	21	8	30	**6**	100	100

Table 6.4: Comparison of our proposed method to three baselines on PETS 2009 [56]. Bold entries for the proposed methods indicate that this method performed better than the state-of-the-art.

compiles several surveillance scenarios containing pedestrian trajectories. Our methods perform particularly well for precision-based measures resulting e.g., in a low false alarm rate. However, the underlying motion models for the reference methods allow to reduce the number of fragmentations compared to our purely association-based approach.

COMPARING RESULTS ON BOTH DATASETS, the different performance is primarily caused by the difficulty of both scenarios, reflected by different object detection performance. The PETS2009 surveillance setup provides a relatively large zenith angle resulting in few occlusions. In comparison, for the KITTI setup, objects are observed from a drivers point of view resulting in frequent inter-object occlusions. Furthermore, the object detections provided for the PETS2009 scenario gain better performance since both camera and background are static while the camera for the KITTI scenarios is mounted on a moving vehicle.

PEFORMANCE FOR DIFFERENT SLIDING WINDOW SIZES. Finally, we evaluate the tracking performance of mbodSSP for different values of τ (Fig. 6.19) on all frames of the KITTI training dataset. (The test set ground truth is not publicly available and would therefore not allow to reproduce results.) The run time comparison shows that for a reasonable value of $\tau = 10$, our non-optimized Python implementation requires less than 10ms which is sufficient for many real-time online applications. To gain quantitatively good results, such a small value of τ is sufficient, considering the increasing run time. Therefore, we decided to use $\tau = 10$ as a compromise between tracking performance and speed for all experiments in this thesis.

6.2.4 *Detection-independent Performance*

Multi-target tracking relies on detections as its input. Using any object detector to provide this information already results in noisy input data, missing detections, and false positives. Therefore, we perform an additional experiment with artificially assembled data to evaluate these effects in an isolated fashion. We

	Car		Pedestrian	
	GT	GT + FP	GT	GT + FP
MOTA	0.98	0.94	0.96	0.77
MOTP	0.92	0.92	0.93	0.93
MODA	0.98	0.94	0.97	0.78
MODP	0.80	0.79	0.28	0.27
recall	0.98	0.96	0.96	0.90
precision	1.0	0.98	1.0	0.88
F1	0.99	0.97	0.98	0.89
FAR	0.00012	0.054	0.0	0.17
MT	0.80	0.72	0.81	0.57
ML	0.030	0.047	0.090	0.16
Id-Switches	4	2	102	54
Fragmentations	6	35	134	111

Table 6.5: Detector-independent Performance. We used groundtruth (GT) labels with a sampled "detection" score and additional false positive (FP) detections to evaluate tracking performance independent of the input data and to verify outlier detection.

computed the distribution of the detector score for true positives of a regular experiment. In the new dataset, true positives are represented by ground truth bounding boxes with a score sampled from this distribution. Additionally, we added remaining detections (being a false positive in the regular case) to introduce outliers into the data. We were running the globally optimal algorithm dSSP to obtain tracklets for cars and pedestrians. The results in Table 6.5 demonstrate that multi-target tracking for challenging traffic scenarios is solved in the presence of a "perfect" object detector by the proposed algorithms. Adding false positives only leads to a slight drop in performance for most measures. However, this indicates that a robust outlier rejection is a key task for multi-target tracking which is realized by the proposed algorithms. Moreover, the results indicate that a relatively low detector threshold is preferable to reduce fragmentations or identity switches in an early stage based on image evidence.

6.2.5 *Qualitative Evaluation*

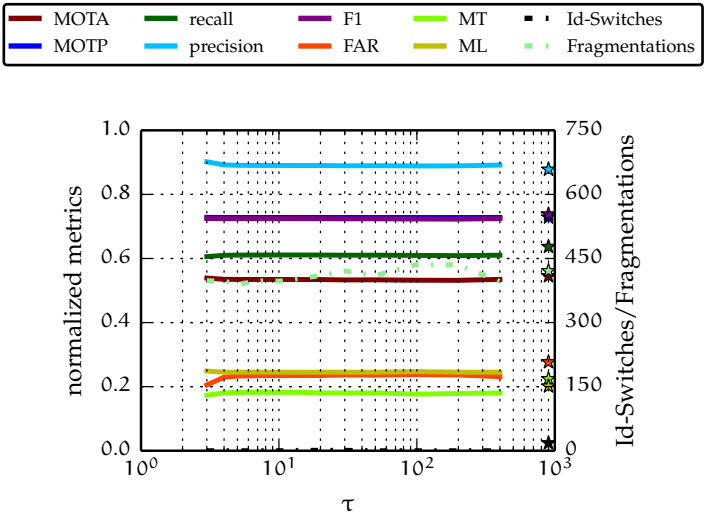
Fig. 6.21 – 6.26 illustrate qualitative results for the optimal data association and our proposed approximative algorithm. First, we discuss performance for the 2D case for cars in general and review selected differences between the two solutions. Secondly, we state solely qualitative results for a challenging inner-city scenario. For a more complete scene description, in this case we used additional pedestrian and cyclist detections during optimization. Finally, we show qualitative results for the proposed 3D object detector (Section 5.1.2) for two challenging scenarios. For the optimal solution, results were obtained by applying our dSSP implementation. For the approximative solution, we used a cache length $|\mathcal{C}| = 4$ and an optimization window of $\tau = 10$. For both methods and all object classes, the parameters were set according to Table 6.2.

For most tracklets data association has been performed correctly. Far away objects are tracked from a very early point and followed persistently (Fig. 6.21). Note that a quantitative evaluation for such cases was not possible. Objects posing an interesting problem for this experiment are typically more than 80m away from the egovehicle and cannot be labeled reliably in the laser point cloud. In contrast to several existing approaches, our method is capable of starting tracks reliably from the first association onwards. Turning cars with a significant change in appearance and bounding box size are tracked continuously during such a scenario (Fig. 6.22). Even for crowded inner-city scenarios, preceding objects are followed over their complete presence (Fig. 6.23). Low-confident object detections (e.g., due to illumination changes) may lead for the approximative solution to terminate a track early, while the optimal solution can exploit the full evidence to persistently follow this track (Fig. 6.24). Parts of the sequence which went out of the optimization window for mdodSSP may also avoid fragmentations (Fig. 6.25). Low confident evidence (e.g., due to occlusions) in the beginning of a trajectory may lead to a delayed initialization for mbodSSP (Fig. 6.26). For a challenging, crowded inner-city scenario with additional object classes the proposed framework obtains good

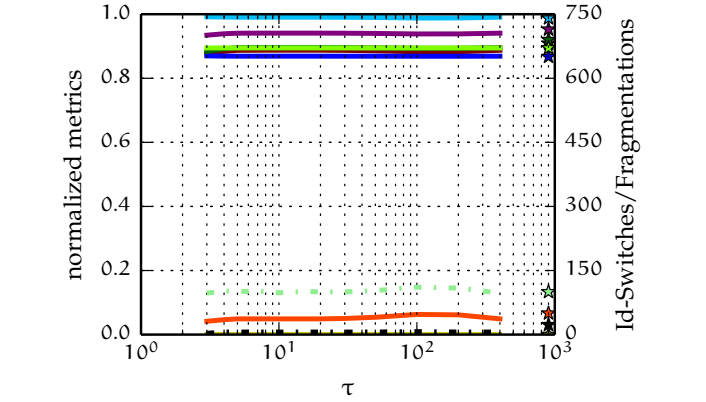
results (Fig. 6.27 + 6.28). However, high, frequent occlusions for several frames lead to fragmentations.

Our scene flow-based 3D object detector can be used to reliably estimate trajectories (Fig. 6.29). We used our mbodSSP algorithm for data association in this experiment to review results in an online setting. The 3D information additionally allows to associate tracks in a world coordinate system which supports rejecting implausible associations. The varying number of scene flow vectors on an object during its presence in the scene leads to significant changes in the estimated 3D dimension. Nevertheless, data association performs well in this case. However, the scene flow-based clustering creates false positive detections more consistently at one place than the 2D detector. This results in false positive tracks even after the data association since both detector score and association features indicate a correct tracklet.

In summary, the proposed methods for globally-optimal multi-target tracking and the memory-bounded approximation work well, fast, and robustly even in the presence of outliers. While the optimal algorithms cannot be used for an online scenario such as autonomous driving with an arbitrarily large input sequence and real-time demands, the memory-bounded approximation is already as a python implementation capable of handling such scenarios. Consequently, input data (object detections and association features) must be processed real-time capable as well.



(b) KITTI (car)



(c) PETS (pedestrian)

Figure 6.20: PERFORMANCE FOR DIFFERENT OPTIMIZATION WINDOWS. We evaluated the performance of mbodSSP for different values of its optimization window τ on the whole KITTI training set and PETS S2.L1. Results for $\tau \to \infty$ are given on the very right of the plot.

Figure 6.21: Results for the Optimal (left) and Approximative Solution (right). Far away objects are tracked continuously. The oncoming car is followed for 3.7s and not confused with several parking cars.

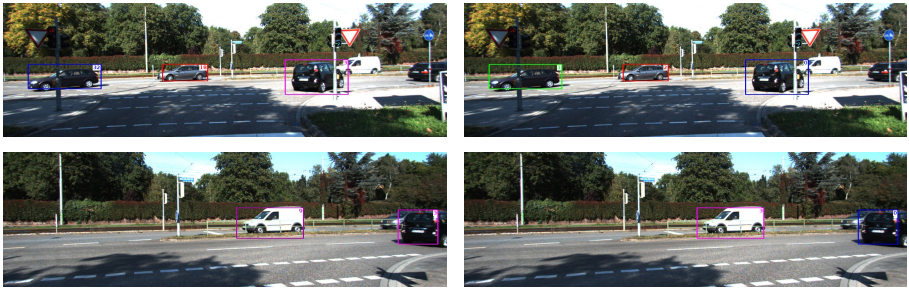


Figure 6.22: Results for the Optimal (left) and Approximative Solution (right). Objects are tracked continuously during sharp turns. Significantly changing bounding box sizes and appearance do not lead to track interruption for the preceding car.

Figure 6.23: RESULTS FOR THE OPTIMAL (LEFT) AND APPROXIMATIVE SOLUTION (RIGHT). Preceding objects in crowded scenes are tracked continuously. The preceding (black) car is followed for 9.5s

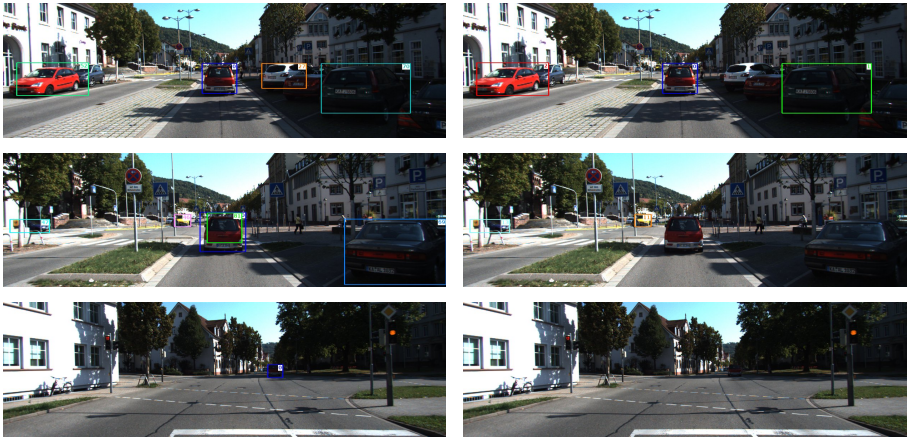


Figure 6.24: RESULTS FOR THE OPTIMAL (LEFT) AND APPROXIMATIVE SOLUTION (RIGHT). Changes in illumination and resulting ambiguous detections (middle) may lead the approximative solution to terminate a track due to missing future evidence. In contrast, the optimal solution tracks the preceding car for 37.2s.

Figure 6.25: Results for the Optimal (left) and Approximative Solution (right). Challenging intersection scenarios lead to qualitatively good results. The limited optimization window for the approximative solution may also avoid fragmentations of the trajectory outside of this window.

Figure 6.26: Results for the Optimal (left) and Approximative Solution (right). Missing future evidence for the approximative solution may lead to tracks starting later compared to the optimal solution (second car on the right). This happens primarily for detections with a low score for the early part of the trajectory.

Figure 6.27: RESULTS FOR THE OPTIMAL (LEFT) AND APPROXIMATIVE SOLUTION (RIGHT). For crowded inner-city scenes, objects of different classes (cars, pedestrians, cyclists) are tracked robustly over the main part of their presence in the image.

Figure 6.28: RESULTS FOR THE OPTIMAL (LEFT) AND APPROXIMATIVE SOLUTION (RIGHT). For groups of interacting pedestrians, tracks are followed continuously even for partly occluded objects. Very high occlusions for several frames may lead to interrupted tracks.

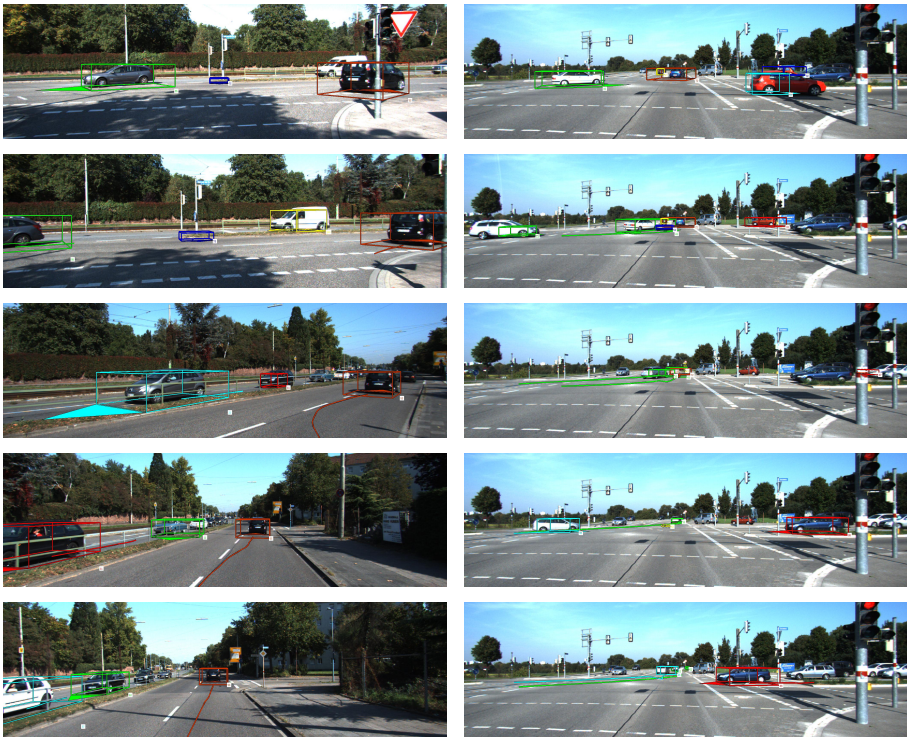


Figure 6.29: Qualitative Results for our 3D Object Detector. This figure shows two different sequences (left, right) for the 3D object detector (Section 5.1.2). Tracking was performed using mbodSSP. Moving objects are tracked persistently although the bounding box size is varying significantly depending on the number of scene flow vectors for the particular object in a frame. However, false positives cannot be rejected as good as in the 2D case due to their constant appearance (left: blue track, right: yellow track).

7

CONCLUSION AND FUTURE DIRECTIONS

> *I never think of the future – it comes soon enough.*
>
> ALBERT EINSTEIN

This thesis has proposed an efficient approach for optimal data association to solve multi-target tracking in a tracking-by-detection framework. The application focused on is autonomous driving, which poses an interesting and currently unsolved problem due to crowded scenes and real-time demands. To solve the data association problem, the actual maximization task has been casted into a min-cost flow formulation which allows to find the optimal data association for given detections in a batch setting. This formulation inherently solves the model selection problem, i.e., estimating the number of tracks present in the scene including their respective start and end point in time as well as rejecting detection outliers.

Generally, the successive shortest paths algorithm solves this network optimization problem in an efficient fashion. This thesis has proposed a dynamic approach adopting Dijkstra's algorithm to the special structure of the tracking problem achieving further runtime reductions. In contrast to existing approaches, the dynamic algorithm has been extended to be capable solving the stated problem in an online setting. However, all previously mentioned approaches scale badly with a growing sequence length.

To tackle this problem, this thesis has finally proposed an approximation of the online algorithm with bounded memory and computational resources solving multi-target tracking in real-time with only marginally reduced tracking performance.

A challenging dataset providing a large number of traffic scenes with a comprehensive ground truth has been proposed to evaluate object detection and tracking algorithms. An extensive evaluation of the proposed algorithms has shown, that for the challenging task of multi-target tracking for autonomous driving scenarios the proposed approach has significantly outperformed the state-of-the-art. Moreover, parameters to weight different association features have been learned using ground truth training data and evaluated with regard to their influence on tracking performance.

An interesting direction of future research is given by exploiting a joint optimization for different object detectors and incorporating motion models. This may lead to higher-order terms and/or quadratic pair-wise costs. A possible solution must either formulate the resulting network (which poses a NP-hard optimization problem in general) and apply Lagrangian or heuristic relaxation approaches or iteratively solve the data association for initially given constraints in an EM-like scheme. Extending the cost functions in such a manner introduces additional parameters and requires more sophisticated parameter optimization. Consequently, using the introduced ground truth data for parameter learning presents an interesting research aspect.

Furthermore, non-imaged based cost terms will increase robustness in complex environments. Using maps as prior knowledge, object hypotheses, motion models, or group behavior estimation can benefit from this information. In the case of outdated maps, tracking results provide valuable information on changed infrastructure and can be used for subsequent algorithms to keep such maps up-to-date.

Moreover, current tracking metrics primarily focus on evaluation on a detection level. Designing a metric for an evaluation on tracklet level can facilitate learning and make a comparison between existing tracking approaches more explicit.

A

NETWORK AND OPTIMIZATION ALGORITHMS

This appendix states related network optimization algorithms, which are used to implement the necessary functionality for the proposed algorithms. Moreover, a generic approach for minimum-cost maximum-flow computation in a network is reviewed, which can solve the multi-target tracking problem discussed in this thesis in a general and hence more complex fashion. Additionally, we state the optimization scheme used for logistic regression in Section 5.3.

A.1 NETWORK ALGORITHMS

A.1.1 *Depth-first Search*

All proposed algorithms implicitly require finding connected components (nodes with outdated predecessors and their successors), typically during initialization. For this task, depth-first search (Algorithm A.1) is an efficient strategy to collect all relevant nodes. This algorithm allows traversing a graph starting at arbitrary root nodes (in this case nodes on the most recent shortest path) and continues by exploring neighboring nodes affected by the predecessor change. Affected nodes have the root node (or one of it successors) as their predecessor. The time complexity for DFS is linear for the graph size with $\mathcal{O}(|V| + |E|$.

Algorithmus A.1 : DEPTH-FIRST SEARCH

Input : Graph G, Starting Node $v \in G$
Output : Connected Subgraph S

```
1  D ← emptyStack()
2  S ← emtpyStack()
3  S ← pushToStack(v)
4  while S ≠ empty do
5  |  v ← popFromStack(S)
6  |  if ¬ D(v) then
7  |  |  D ← pushToStack(v)
8  |  |  foreach edge (v, w) ∈ adjacentEdges(G, v) do
9  |  |  |  S ← push(w)
10 return S
```

A.1.2 *Bellman-Ford Algorithm*

The Bellman-Ford (*BF*) algorithm (Algorithm A.2) is based on the principle of *relaxation* (Algorithm A.3). All nodes are initialized with an unknown predecessor and infinite costs (as an upper bound) on their shortest path except for the source with a cost of 0. The algorithm iteratively replaces this upper bound of the correct distance to the source for each node gradually by tighter bounds (by computing the predecessor and its distance) until the optimal solution is reached. To achieve optimality the BF algorithm relaxes all edges in the graph for $|V| - 1$ iterations, where $|V|$ denotes the number of vertices in the graph. In each of these repetitions, the number of vertices with correctly calculated distances grows, from which it follows that eventually all vertices will have their correct distances, even before the $|V| - 1$ repetition. In contrast to the originally proposed BF algorithm, current implementations consider this behavior by terminating the algorithm early when no changes between two iterations are detected anymore.

Algorithmus A.2 : Bellman-Ford [46]

Input : Graph G, Costs C, Source s
Output : Predecessor Map π, Distance Map ρ
`// set` $\pi(u) =$ `Unkown,` $\rho(u) = \infty$, $u \in |V| \setminus s$
$\pi, \rho \leftarrow$ InitializeSingleSource(G,s)
for $i \leftarrow 1, \ldots, |V| - 1$ **do**
  foreach *edge* $(u, v) \in G$ **do**
    $\pi(v) \leftarrow$ Relax($\pi(v)$, (u, v), $c(u, v)$)
`// check for negative cycles`
foreach *edge* $(u, v) \in G$ **do**
  if $\rho(v) > \rho(u) + c(u, v)$ **then**
    `// a negative cycle was detected`
    RaiseError("negative cycle")
return π, d

Algorithmus A.3 : Relaxing Edges.

Input : Predecessor Map $\pi(v)$, Distance Map ρ, Edge (u, v)
Output : Predecessor Map π, Distance Map ρ
`// relax edge` (u, v) `with cost` $c(u, v)$
if $\rho(v) > \rho(u) + c_{uv}$ **then**
  `// update cost d for shortest path to` v
  $\rho(v) \leftarrow \rho(u) + c(u, v)$
  `// update predecessor for edge` (u, v)
  $\pi(v) \leftarrow u$
return $\pi(v)$, $d(v)$

A.1.3 *Minimum-Cost Maximum-Flow Algorithms*

In contrast to the algorithms discussed in the main part of this thesis, many network optimization problems need to solve a min-cost max-flow problem. Such a generic algorithm can also be used to solve the problems discussed in this thesis, resulting in a high complexity as discussed in Section 3.2. A generic algorithm for solving min-cost max-flow problems is the push-relabel algorithm (Algorithm A.4, [84]). The algorithm starts by pushing maximum flow from the source and than repeatedly performing basic operations (*push* and *relabel*) maintaining a *preflow* and gradually obtaining the maximum flow (for given cost constraints). Therefore, flow is pushed between neighbors or relabeling is applied to create an admissible out edge. If no active node is existing anymore, the algorithm terminates. A variant of this algorithm computing min-cost max-flow for a network is available as an open source implementation.[1]

A.2 ITERATIVELY REWEIGHTED LEAST-SQUARES (IRLS)

An optimization problem given by an objective function

$$\arg\min_{\beta} \sum_{i=1}^{n} w_i(\beta)\left|y_i - f_i(\beta)\right|^2, \tag{A.1}$$

cannot be solved in a closed form anymore. A maximum likelihood estimation can be achieved by iteratively solving a weighted least-square problem given by

$$\beta^{(t+1)} = \arg\min_{\beta} \sum_{i=1}^{n} w_i(\beta^{(t)})\left|y_i - f_i(\beta)\right|^2. \tag{A.2}$$

We refer the reader to [156] for a more complete introduction.

1 Source Code: http://lemon.cs.elte.hu/

Algorithmus A.4 : PUSH-RELABEL ALGORITHM

Input : Graph G, Source s, Sink t, Capacity Function c

Output : Flow f containing s-t paths

foreach *edge* $(u, v) \in E$ **do**
  $f(u, v) \leftarrow 0$ // flow
  if $u = s$ **then**
    $f(u, v) \leftarrow c(u, v)$
  if $v = s$ **then**
    $f(u, v) \leftarrow -f(v, u)$
foreach $v \in V$ **do**
  $e_f \leftarrow \sum_{w:(w,v)\in E} f(w, v)$
  if v=s **then**
    $d(v) \leftarrow n$
  else
    $d(v) \leftarrow 0$
while $\exists$ *active node* **do**
  if v *active and* $(v, w) \in A_f$ *and* $d(v) = d(w) + 1$ **then**
    PushFlow((v, w), $\min(e_f(v), c_f(v, w))$)
  if v *active and* $\forall w \in \Gamma^+(v)$ *with* $(v, w) \notin E_f$ *or* $d(W) \geqslant d(v)$ **then**
    $d(v) \leftarrow \min(d(w) + 1 | w \in \Gamma^+(v)$ with $(v, w) \in E_f)$
return f

Algorithmus A.5 : Iteratively Reweighted Least-Squares

Input : Feature Matrix $\mathbf{X} \in \mathbb{R}^m \times n$
Vector of Class Variables $\mathbf{y} \in \mathbb{R}^m$
Ridge Regression Constant λ
Output : Model Parameter β

$\beta \leftarrow 0$
while $\neg$ *terminated* **do**
  $u_i \leftarrow \frac{1}{1+e^{-X_i\beta}}$
  $W_{ii} \leftarrow \mu_i(1-\mu_i)$
  $U_i \leftarrow X_i\beta + \frac{y_i-\mu_i}{W_{ii}}$
  $A \leftarrow X^\top WX + \lambda \mathbf{I}$
  $b \leftarrow X^\top WU$
  $\beta \leftarrow$ conjugateGradient(A, b, β)
return β

Algorithmus A.6 : Conjugate Gradient

Input : A, b, β
Output : β satisfying $A\beta = b$

$r \leftarrow b - A\beta$
$d \leftarrow r$
$\delta_{new} \leftarrow r^t opr$
$\delta_0 \leftarrow \delta_{new}$
while $\delta_{new} > \delta_{min}$ **do**
  $q \leftarrow Ad$
  $\alpha \leftarrow \frac{\delta_{new}}{d^\top q}$
  $\beta \leftarrow \beta + \alpha d$
  $r \leftarrow r - \alpha q$
  $\delta_{old} \leftarrow \delta_{new}$
  $\delta_{new} \leftarrow r^\top r$
  $\gamma \leftarrow \frac{\delta_{new}}{\delta_{old}}$
  $d \leftarrow r + \gamma d$
return β

BIBLIOGRAPHY

[1] S. Agarwal, A. Awan, and D. Roth, "Learning to detect objects in images via a sparse, part-based representation," *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 26, no. 11, pp. 1475–1490, 2004. 19

[2] R. Ahuja, T. Magnanti, and J. Orlin, *Network Flows*. Prentice-Hall, 1993. 13, 34, 36, and 37

[3] D. L. Alspach, "A gaussian sum approach to the multi-target identification-tracking problem," *Automatica*, vol. 11, no. 3, pp. 285–296, 1975. 10 and 13

[4] M. Andriluka, S. Roth, and B. Schiele, "People-tracking-by-detection and people-detection-by-tracking," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2008. 23 and 24

[5] M. Andriluka, S. Roth, and B. Schiele, "Monocular 3d pose estimation and tracking by detection," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2010. 12 and 20

[6] A. Andriyenko and K. Schindler, "Multi-target tracking by continuous energy minimization," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2011. 20 and 108

[7] A. Andriyenko, K. Schindler, and S. Roth, "Discrete-continuous optimization for multi-target tracking," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2012. ix, xi, 10, 12, 20, 23, 24, 107, and 108

[8] C. Arora and A. Globerson, "Higher order matching for consistent multiple target tracking," in *Proc. IEEE International Conf. on Computer Vision (ICCV)*, 2013. 10

[9] W. H. Bachman, "A gis-based modal model of automobile exhaust emissions," School of Civil and Environmental Engineering, Tech. Rep., 1998. 1

[10] M. Bajracharya, B. Moghaddam, A. Howard, S. Brennan, and L. H. Matthies, "A fast stereo-based system for detecting and tracking pedestrians from a moving vehicle," *International Journal of Robotics Research (IJRR)*, vol. 28, pp. 1466–1485, 2009. 27

[11] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. Black, and R. Szeliski, "A database and evaluation methodology for optical flow," *International Journal of Computer Vision (IJCV)*, vol. 92, pp. 1–31, 2011. 21

[12] A. Bar-Hillel, D. Levi, E. Krupka, and C. Goldberg, "Part-based feature synthesis for human detection," in *Proc. of the European Conf. on Computer Vision (ECCV)*. Springer, 2010, pp. 127–142. 19

[13] Y. Bar-Shalom, K. Chang, and H. A. P. Blom, "Automatic track formation in clutter with a recursive algorithm," in *IEEE Conference on Decision and Control (DC)*, Dec 1989, pp. 1402–1408 vol.2. 18

[14] Y. Bar-Shalom and T. E. Fortmann, *Tracking and Data Association*. Academic Press, 1988. 12 and 13

[15] Y. Bar-Shalom, F. Daum, and J. Huang, "The probabilistic data association filter," *Control Systems*, vol. 29, no. 6, pp. 82–100, 2009. 13

[16] Y. Bar-Shalom and E. Tse, "Tracking in a cluttered environment with probabilistic data association," *Automatica*, vol. 11, no. 5, pp. 451–460, 1975. 10, 13, and 18

[17] K. D. Becker, *Träumen, erfinden, die Welt verbessern*. Haag + Härchen, 2002. 6

[18] H. Ben Shitrit, J. Berclaz, F. Fleuret, and P. Fua, "Tracking multiple people under global appearance constraints,"

in *Proc. IEEE International Conf. on Computer Vision (ICCV)*, Nov 2011, pp. 137–144. 29

[19] H. Ben Shitrit, J. Berclaz, F. Fleuret, and P. Fua, "Multi-Commodity Network Flow for Tracking Multiple People," *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 2013. 18

[20] R. Benenson, M. Mathias, R. Timofte, and L. J. V. Gool, "Pedestrian detection at 100 frames per second," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2012. 27

[21] B. Benfold and I. Reid, "Stable multi-target tracking in real-time surveillance video," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2011. 18

[22] J. Berclaz, F. Fleuret, E. Turetken, and P. Fua, "Multiple object tracking using k-shortest paths optimization," *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 2011. ix, xi, 6, 18, 29, 31, 36, 39, 43, 52, and 59

[23] K. Bernardin and R. Stiefelhagen, "Evaluating multiple object tracking performance: The clear mot metrics," *Journal on Image and Video Processing (JIVP)*, vol. 1, pp. 1–10, 2008. 23 and 26

[24] D. Bertsekas and R. Gallager, *Data networks*. Prentice-Hall, Inc., 1992. 47 and 49

[25] S. M. Bileschi, "Streetscenes: Towards scene understanding in still images," Ph.D. dissertation, Massachusetts Institute of Technology, 2006. 22

[26] R. Bise, Z. Yin, and T. Kanade, "Reliable cell tracking by global data association," in *International Symposium on Biomedical Imaging: From Nano to Macro*, March 2011, pp. 1004–1010. 29

[27] R. Bishop, "Driverless car summit 2012." Association of Unmanned Vehicle Systems International (AUVSI), 2012. 4

[28] S. Blackman, "Multiple hypothesis tracking for multiple target tracking," *AESM*, vol. 19, no. 1, pp. 5–18, Jan 2004. 13 and 18

[29] O. Blumtritt, H. Petzold, and W. Aspray, *Tracking the history of radar*. IEEE-Rutgers Center for the History of Electrical Engineering, 1994. 10

[30] Y. Boers and J. Driessen, "Multitarget particle filter track before detect application," *IEE Proceedings-Radar, Sonar and Navigation*, vol. 151, no. 6, pp. 351–357, 2004. 12

[31] S. Bonneau, M. Dahan, and L. Cohen, "Single quantum dot tracking based on perceptual grouping using minimal paths in a spatiotemporal volume," *Image Processing, IEEE Transactions on*, vol. 14, no. 9, pp. 1384–1395, Sept 2005. 29

[32] L. Bourdev and J. Malik, "Poselets: Body part detectors trained using 3d human pose annotations," in *Proc. IEEE International Conf. on Computer Vision (ICCV)*, Sept 2009, pp. 1365–1372. 19

[33] E. Brau, J. Guan, K. Simek, L. D. Pero, C. R. Dawson, and K. Barnard, "Bayesian 3d tracking from monocular video," in *Proc. IEEE International Conf. on Computer Vision (ICCV)*, 2013. 18

[34] M. D. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier, and L. J. V. Gool, "Online multiperson tracking-by-detection from a single, uncalibrated camera," *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 33, no. 9, pp. 1820–1833, 2011. 12 and 29

[35] M. D. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier, and L. Van Gool, "Robust tracking-by-detection using a detector confidence particle filter," in *Proc. IEEE International Conf. on Computer Vision (ICCV)*, 2009. 12 and 20

[36] A. Broggi, M. Bertozzi, A. Fascioli, and M. Sechi, "Shape-based pedestrian detection," in *Proc. IEEE Intelligent Vehicles Symposium (IV)*, 2000. 19 and 27

[37] A. Broggi, P. Medici, P. Zani, A. Coati, and M. Panciroli, "Autonomous vehicles control in the VisLab Intercontinental Autonomous Challenge," *Annual Reviews in Control*, vol. 36, no. 1, pp. 161–171, 2012. 27

[38] M. A. Brubaker, A. Geiger, and R. Urtasun, "Lost! leveraging the crowd for probabilistic visual self-localization," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2013. 27

[39] M. Buehler, K. Iagnemma, and S. Singh, *The 2005 darpa grand challenge: The great robot race*. Springer, 2007, vol. 36. 27

[40] P. Burt, J. Bergen, R. Hingorani, R. Kolczynski, W. A. Lee, A. Leung, J. Lubin, and H. Shvayster, "Object tracking with a moving camera," in *Proc. Workshop on Visual Motion*, Mar 1989, pp. 2–12. 10 and 11

[41] A. Butt and R. Collins, "Multi-target tracking by lagrangian relaxation to min-cost network flow," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2013. 20, 44, and 45

[42] L. H. Carter, A. Peine, and G. Golden, "Proposer information pamphlet (pip) – video analysis and content extraction (vace)," Disruptive Technology Office, Tech. Rep., 2006. 23

[43] W. Choi, C. Pantofaru, and S. Savarese, "A general framework for tracking multiple people from a moving camera," *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 2013. 10 and 18

[44] W. Choi and S. Savarese, "A unified framework for multi-target tracking and collective activity recognition," in *Proc. of the European Conf. on Computer Vision (ECCV)*, 2012. 12, 18, 21, 44, and 52

[45] A. Chowdhury, R. Chatterjee, M. Ghosh, and N. Ray, "Cell tracking in video microscopy using bipartite graph matching," in *Proc. of the International Conf. on Pattern Recognition (ICPR)*, Aug 2010, pp. 2456–2459. 29

[46] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson, *Introduction to Algorithms*. McGraw-Hill Higher Education, 2001. 36, 37, 41, 70, 100, and 125

[47] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2005. 19

[48] N. Dalal, B. Triggs, and C. Schmid, "Human detection using oriented histograms of flow and appearance," in *Proc. of the European Conf. on Computer Vision (ECCV)*, 2006. 19

[49] M. De Berg, M. Van Kreveld, M. Overmars, and O. C. Schwarzkopf, *Computational geometry*. Springer, 2000. 68

[50] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2009. 20

[51] E. D. Dickmanns and V. Graefe, "Dynamic monocular machine vision," *Machine Vision and Applications (MVA)*, vol. 1, no. 4, pp. 223–240, 1988. 12 and 27

[52] E. Dickmanns and V. Graefe, "Applications of dynamic monocular machine vision," *Machine Vision and Applications (MVA)*, vol. 1, no. 4, pp. 241–261, 1988. 27

[53] P. Dollar, B. Babenko, S. Belongie, P. Perona, and Z. Tu, "Multiple component learning for object detection," in *Proc. of the European Conf. on Computer Vision (ECCV)*, 2008. 19

[54] P. Dollar, Z. Tu, H. Tao, and S. Belongie, "Feature mining for image classification," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2007. 19

[55] P. Dollar, C. Wojek, B. Schiele, and P. Perona, "Pedestrian detection: An evaluation of the state of the art," in *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 99, 2012. 19, 20, and 22

[56] A. Ellis and J. M. Ferryman, "PETS 2010 and PETS 2009 evaluation of results using individual ground truthed single views." in *Proc. IEEE International Conf. on Advanced Video and Signal-Based Surveillance (AVSS)*, 2010, pp. 135–142. 23, 24, 96, 107, and 108

[57] M. Enzweiler and D. M. Gavrila, "Monocular pedestrian detection: Survey and experiments," *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 31, pp. 2179–2195, 2009. 27

[58] M. Enzweiler and D. M. Gavrila, "Integrated pedestrian classification and orientation estimation," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2010. 19

[59] A. Ess, B. Leibe, and L. V. Gool, "Depth and appearance for mobile scene analysis," in *Proc. IEEE International Conf. on Computer Vision (ICCV)*, 2007. 21 and 22

[60] A. Ess, B. Leibe, K. Schindler, and L. V. Gool, "Robust multi-person tracking from a mobile platform," *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 31, pp. 1831–1846, 2009. 10, 12, and 20

[61] A. Ess, B. Leibe, K. Schindler, and L. Van Gool, "A mobile vision system for robust multi-person tracking," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2008. 23 and 24

[62] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2011 (VOC2011) Results." 20, 21, 22, 65, 73, and 86

[63] L. Fei-Fei, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categories," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2004. 21 and 22

[64] L. Fei-Fei, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: an incremental bayesian approach tested on 101 object categories," in *Workshop on Generative-Model Based Vision (GMBV)*, 2004. 21

[65] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part based models," *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 32, pp. 1627–1645, 2010. 19, 20, 65, 73, and 96

[66] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, pp. 381–395, 1981. 66

[67] R. Fisher. Caviar: Context aware vision using image-based active recognition. http://homepages.inf.ed.ac.uk/rbf/CAVIARDATA1/ [Online. Accessed: 14/04/2014] 23 and 24

[68] T. E. Fortmann, Y. Bar-Shalom, and M. Scheffe, "Sonar tracking of multiple targets using joint probabilistic data association," *IEEE Journal of Oceanic Engineering (JOE)*, vol. 8, no. 3, pp. 173–184, Jul 1983. 18

[69] U. Franke and I. Kutzbach, "Fast stereo based object detection for stop & go traffic," in *Proc. IEEE Intelligent Vehicles Symposium (IV)*, Sep 1996, pp. 339–344. 19 and 27

[70] T. M. Gasser, "Ergebnisse der projektgruppe automatisierung: Rechtsfolgen zunehmender fahrzeugautomatisierung," Bundesanstalt für Straßenwesen, Bergisch Gladbach, Tech. Rep., 2010. 4

[71] D. M. Gavrila, "A bayesian, exemplar-based approach to hierarchical shape matching," *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 29, pp. 1408–1421, 2007. 19

[72] D. M. Gavrila and S. Munder, "Multi-cue pedestrian detection and tracking from a moving vehicle," *International Journal of Computer Vision (IJCV)*, vol. 73, pp. 41–59, 2007. 12

[73] A. Geiger, "Probabilistic models for 3d urban scene understanding from movable platforms," Ph.D. dissertation, KIT, 2013. 66

[74] A. Geiger, M. Lauer, F. Moosmann, B. Ranft, H. Rapp, C. Stiller, and J. Ziegler, "Team annieway's entry to the grand cooperative driving challenge 2011," *IEEE Trans. on Intelligent Transportation Systems (TITS)*, 2012. 4 and 27

[75] A. Geiger, M. Lauer, C. Wojek, C. Stiller, and R. Urtasun, "3d traffic scene understanding from movable platforms," *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 2014. 27, 44, 73, 107, and 108

[76] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *International Journal of Robotics Research (IJRR)*, vol. 32, pp. 1229 – 1235, 2013. 73, 81, and 82

[77] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2012. 73, 81, 82, 85, and 108

[78] A. Geiger, F. Moosmann, O. Car, and B. Schuster, "Automatic calibration of range and camera sensors using a single shot," in *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, 2012. 85

[79] A. Geiger, C. Wojek, and R. Urtasun, "Joint 3d estimation of objects and scene layout," in *Advances in Neural Information Processing Systems (NIPS)*, 2011. 65

[80] A. Geiger, J. Ziegler, and C. Stiller, "StereoScan: Dense 3d reconstruction in real-time," in *Proc. IEEE Intelligent Vehicles Symposium (IV)*, 2011. 66

[81] D. Geronimo, A. M. Lopez, A. D. Sappa, and T. Graf, "Survey on pedestrian detection for advanced driver assistance

systems," *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 32, no. 7, pp. 1239–1258, 2010. 20 and 27

[82] W. J. Godinez, M. Lampe, S. Wörz, B. Müller, R. Eils, and K. Rohr, "Deterministic and probabilistic approaches for tracking virus particles in time-lapse fluorescence microscopy image sequences," *Medical Image Analysis*, vol. 13, no. 2, pp. 325–342, 2009. 29

[83] M. Goebl and G. Faerber, "A real-time-capable hard- and software architecture for joint image and knowledge processing in cognitive automobiles," in *Proc. IEEE Intelligent Vehicles Symposium (IV)*, 2007. 84

[84] A. V. Goldberg, "An efficient implementation of a scaling minimum-cost flow algorithm," *Journal of Algorithms*, vol. 22, no. 1, pp. 1–29, Jan. 1997. 34 and 126

[85] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, 2004. 66

[86] C. Hoffmann and T. Dang, "Cheap joint probabilistic data association filters in an interacting multiple model design," in *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, 2006. 19

[87] C. Hoffmann, T. Dang, and C. Stiller, "Vehicle detection fusing 2d visual features," in *Proc. IEEE Intelligent Vehicles Symposium (IV)*, 2004, pp. 280–285. 19

[88] R. Horaud and F. Dornaika, "Hand-eye calibration," *International Journal of Robotics Research (IJRR)*, vol. 14, no. 3, pp. 195–210, 1995. 85

[89] B. S. Horesh, J. Berclaz, F. Fleuret, and P. Fua, "Tracking multiple people under global appearance constraints," in *Proc. IEEE International Conf. on Computer Vision (ICCV)*, 2011. 18

[90] H. Idrees, I. Saleemi, and M. Shah, "Statistical inference of motion in the invisible," in *Proc. of the European Conf. on Computer Vision (ECCV)*, 2012. 18

[91] Inrix, "Inrix scorecard." http://scorecard.inrix.com/scorecard/ [Online. Accessed: 04/14/2014] 1

[92] International Organization of Motor Vehicle Manufacturers. Production statistics. http://www.oica.net/category/production-statistics/ [Online. Accessed: 04/04/2014] 1

[93] International Traffic Safety Data and Analysis Group, "Road safety annual report 2013." 1

[94] M. Isard and A. Blake, "Condensation - conditional density propagation for visual tracking," *International Journal of Computer Vision (IJCV)*, vol. 29, pp. 5–28, 1998. 10 and 12

[95] L. Itti, C. Koch, E. Niebur *et al.*, "A model of saliency-based visual attention for rapid scene analysis," *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 20, no. 11, pp. 1254–1259, 1998. 19

[96] K. Jaqaman, D. Loerke, M. Mettlen, H. Kuwata, S. Grinstein, S. L. Schmid, and G. Danuser, "Robust single-particle tracking in live-cell time-lapse sequences," *Nature Methods*, vol. 5, no. 8, pp. 695–702, 2008. 29

[97] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Basic Engineering (JBE)*, vol. 1, no. 82, pp. 35–45, 1960. 10, 12, and 13

[98] S. Kammel, J. Ziegler, and C. Stiller, "Team annieway's autonomous system for the 2007 darpa urban challenge," *Journal of Field Robotics (JFR)*, vol. 25, no. 9, pp. 615–639, September 2008. 3, 4, 27, and 81

[99] T. Kanade, Z. Yin, R. Bise, S. Huh, S. Eom, M. Sandbothe, and M. Chen, "Cell image analysis: Algorithms, system and applications," in *IEEE Workshop on Applications of Computer Vision (WACV)*, Jan 2011, pp. 374–381. 29

[100] C. G. Keller, M. Enzweiler, and D. M. Gavrila, "A new benchmark for stereo-based pedestrian detection," in *Proc. IEEE Intelligent Vehicles Symposium (IV)*, 2011. 22

[101] D. Koller, J. Weber, and J. Malik, "Towards realtime visual based tracking in cluttered traffic scenes," in *Proc. IEEE Intelligent Vehicles Symposium (IV)*, 1994. 10

[102] D. Koller, J. Weber, and J. Malik, "Robust multiple car tracking with occlusion reasoning," in *Proc. of the European Conf. on Computer Vision (ECCV)*. Springer, 1994. 10, 11, and 12

[103] J. F. P. Kooij, G. Englebienne, and D. M. Gavrila, "A non-parametric hierarchical model to discover behavior dynamics from tracks," in *Proc. of the European Conf. on Computer Vision (ECCV)*, 2012. 21

[104] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks." in *Advances in Neural Information Processing Systems (NIPS)*, vol. 1, no. 2, 2012, p. 4. 20

[105] S. Krotosky and M. Trivedi, "On color-, infrared-, and multimodal-stereo approaches to pedestrian detection," *IEEE Trans. on Intelligent Transportation Systems (TITS)*, vol. 8, no. 4, pp. 619–629, Dec 2007. 19

[106] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval research logistics quarterly (NRLQ)*, vol. 2, pp. 83–97, 1955. 13

[107] L. Leal-Taixé, G. Pons-Moll, and B. Rosenhahn, "Everybody needs somebody: modeling social and grouping behavior on a linear programming multiple people tracker," in *Proc. IEEE International Conf. on Computer Vision (ICCV) Workshops*, 2011. 18 and 21

[108] L. Leal-Taixé, G. Pons-Moll, and B. Rosenhahn, "Branch-and-price global optimization for multi-view multi-target tracking," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2012. 10, 11, and 18

[109] B. Leibe, N. Cornelis, K. Cornelis, and L. Van Gool, "Dynamic 3d scene analysis from a moving vehicle," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2007. 12

[110] B. Leibe, E. Seemann, and B. Schiele, "Pedestrian detection in crowded scenes," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2005. 19

[111] Y. Li, C. Huang, and R. Nevatia, "Learning to associate: HybridBoosted multi-target tracker for crowded scene," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2009. 18, 23, and 26

[112] J. Liu, P. Carr, R. T. Collins, and Y. Liu, "Tracking sports players with context-conditioned motion models," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2013. 18

[113] Y. Liu, S. Shan, W. Zhang, X. Chen, and W. Gao, "Granularity-tunable gradients partition (ggp) descriptors for human detection," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, June 2009, pp. 1255–1262. 19

[114] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision (IJCV)*, vol. 60, no. 2, pp. 91–110, 2004. 19

[115] P. C. Mahalanobis, "On the generalized distance in statistics," *Proceedings of the National Institute of Sciences of India*, vol. 2, pp. 49–55, 1936. 69

[116] S. Maji, A. Berg, and J. Malik, "Classification using intersection kernel SVMs is efficient," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2008. 19

[117] S. Maji and J. Malik, "Object detection using a max-margin hough transform," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2009. 19

[118] E. Meijering, O. Dzyubachyk, I. Smal *et al.*, "Methods for cell and particle tracking," *Methods Enzymol*, vol. 504, no. 9, pp. 183–200, 2012. 29

[119] E. Meijering, O. Dzyubachyk, I. Smal, and W. A. van Cappellen, "Tracking in cell and developmental biology," *Seminars in Cell & Developmental Biology (SemCDB)*, vol. 20, no. 8, pp. 894 – 902, 2009. 6 and 29

[120] A. Milan, S. Roth, and K. Schindler, "Continuous energy minimization for multitarget tracking," *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 36, no. 1, pp. 58–72, 2014. 107 and 108

[121] A. Milan, K. Schindler, and S. Roth, "Detection- and trajectory-level exclusion in multiple object tracking," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2013. 20

[122] D. Mitzel and B. Leibe, "Taking mobile multi-object tracking to the next level: People, unknown objects, and carried items," in *Proc. of the European Conf. on Computer Vision (ECCV)*, 2012. 12 and 19

[123] A. Mohan, C. Papageorgiou, and T. Poggio, "Example-based object detection in images by components," *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 23, no. 4, pp. 349–361, Apr 2001. 19

[124] P. Moreels and P. Perona, "Evaluation of features, detectors and descriptors based on 3d objects," *International Journal of Computer Vision (IJCV)*, vol. 73, pp. 263–284, 2007. 21 and 22

[125] J. Munkres, "Algorithms for the assignment and transportation problems," *Journal of the Society for Industrial and Applied Mathematics*, vol. 5, no. 1, pp. 32–38, 1957. 13

[126] M. Munz, K. Dietmayer, and M. Mählisch, "Generalized fusion of heterogeneous sensor measurements for multi target tracking," in *Conference on Information Fusion (FUSION)*, July 2010, pp. 1–8. 28

[127] D. Musicki and R. Evans, "Joint integrated probabilistic data association: Jipda," *IEEE Trans. on Aerospace and Electronic Systems (AES)*, vol. 40, no. 3, pp. 1093–1099, July 2004. 18

[128] D. Musicki, R. Evans, and S. Stankovic, "Integrated probabilistic data association," *IEEE Trans. on Automatic Control (TAC)*, vol. 39, no. 6, pp. 1237–1241, Jun 1994. 18

[129] National Baseball Hall of Fame and Museum. Biography: Chadwick, Henry. http://baseballhall.org/hof/chadwick-henry [Online. Accessed: 21/04/2014] 28

[130] Nayar and H. Murase, "Columbia Object Image Library: COIL-100," Department of Computer Science, Columbia University, Tech. Rep., 1996. 21 and 22

[131] A. T. Nghiem, F. Bremond, M. Thonnat, and V. Valentin, "ETISEO, performance evaluation for video surveillance systems," in *Proc. IEEE International Conf. on Advanced Video and Signal-Based Surveillance (AVSS)*, Sept 2007, pp. 476–481. 23, 24, and 25

[132] D. Nister, O. Naroditsky, and J. R. Bergen, "Visual odometry," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2004. 66

[133] M. Ozuysal, V. Lepetit, and P.Fua, "Pose estimation for category specific multiview object localization," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2009. 21 and 22

[134] D. Padfield, J. Rittscher, and B. Roysam, "Coupled minimum-cost flow cell tracking," in *Information Processing in Medical Imaging (IPMI)*. Springer, 2009, pp. 374–385. 29

[135] C. Papageorgiou and T. Poggio, "A trainable system for object detection," *International Journal of Computer Vision (IJCV)*, vol. 38, no. 1, pp. 15–33, Jun. 2000. 19

[136] N. Papanikolopoulos, P. Khosla, and T. Kanade, "Visual tracking of a moving target by a camera mounted on a

robot: a combination of control and vision," *IEEE Trans. on Robotics and Automation (TRA)*, vol. 9, no. 1, pp. 14–35, Feb 1993. 12

[137] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research (JMLR)*, vol. 12, pp. 2825–2830, 2011. 78

[138] H. Pirsiavash, D. Ramanan, and C. C. Fowlkes, "Globally-optimal greedy algorithms for tracking a variable number of objects," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2011. 31, 36, 39, 42, 45, 52, 104, 107, and 108

[139] H. Rauch, "Combining the functions of signal detection and trajectory estimation(combined signal detection and trajectory estimation functions optimization application to monte carlo simulation for trajectory moving across two dimensional grid)," in *Symposium on Nonlinear Estimation Theory and Its Applications, 2 nd, San Diego, Calif*, 1972, pp. 234–240. 10

[140] D. Reid, "An algorithm for tracking multiple targets," *IEEE Trans. on Automatic Control (TAC)*, vol. 24, no. 6, pp. 843–854, 1979. 13 and 18

[141] B. Russell, A. Torralba, K. Murphy, and W. Freeman, "Labelme: A database and web-based tool for image annotation," *International Journal of Computer Vision (IJCV)*, vol. 77, pp. 157–173, 2008. 21 and 22

[142] P. Sabzmeydani and G. Mori, "Detecting pedestrians by learning shapelet features," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2007. 20

[143] D. Salmond and H. Birch, "A particle filter for track-before-detect," in *Proceedings of the American control conference*, vol. 5, 2001, pp. 3755–3760. 12

[144] I. F. Sbalzarini and P. Koumoutsakos, "Feature point tracking and trajectory analysis for video imaging in cell biology," *Journal of Structural Biology (JSB)*, vol. 151, no. 2, pp. 182–195, 2005. 29

[145] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *International Journal of Computer Vision (IJCV)*, vol. 47, pp. 7–42, 2002. 21

[146] W. Schwartz, A. Kembhavi, D. Harwood, and L. Davis, "Human detection using partial least squares analysis," in *Proc. IEEE International Conf. on Computer Vision (ICCV)*, Sept 2009, pp. 24–31. 19

[147] A. Schwing and R. Urtasun, "Efficient exact inference for 3d indoor scene understanding," in *Proc. of the European Conf. on Computer Vision (ECCV)*, 2012. 97

[148] M. Skolnik, "Fifty years of radar," *Proceedings of the IEEE*, vol. 73, no. 2, pp. 182–197, Feb 1985. 10

[149] A. F. Smeaton, P. Over, and W. Kraaij, "Evaluation campaigns and TRECVid," in *Proc. of the International Workshop on Multimedia Information Retrieval (MIR)*. New York, NY, USA: ACM Press, 2006, pp. 321–330. 23 and 24

[150] P. Smith and G. Buechler, "A branching algorithm for discriminating and tracking multiple objects," *IEEE Trans. on Automatic Control (TAC)*, vol. 20, no. 1, pp. 101–104, Feb 1975. 10

[151] B. Song, T.-Y. Jeng, E. Staudt, and A. K. Roy-Chowdhury, "A stochastic graph evolution framework for robust multi-target tracking," in *Proc. of the European Conf. on Computer Vision (ECCV)*, 2010. 18

[152] Statistisches Bundesamt, "Polizeilich erfasste unfÃ¤lle," Tech. Rep., 2013. 2

[153] C. Stiller, F. Puente León, and M. Kruse, "Information fusion for automotive applications–an overview," *Information Fusion*, vol. 12, no. 4, pp. 244–252, 2011. 28

[154] J. W. Suurballe, "Disjoint paths in a network," *Networks*, vol. 4, no. 2, pp. 125–145, 1974. 36, 39, 42, and 43

[155] J. W. Suurballe and R. E. Tarjan, "A quick method for finding shortest pairs of disjoint paths," *Networks*, vol. 14, no. 2, pp. 325–336, 1984. 36, 39, and 49

[156] R. Szeliski, *Computer Vision: Algorithms and Applications*, ser. Texts in Computer Science. Springer, 2010. http://books.google.de/books?id=bXzAlkODwa8C 126

[157] Q. Tian, H. Sun, Y. Luo, and D. Hu, "Nighttime pedestrian detection with a normal camera using svm classifier," in *Advances in Neural Networks (ANN)*, J. Wang, X.-F. Liao, and Z. Yi, Eds. Springer, 2005, vol. 3497, pp. 189–194. 19

[158] S. M. Tonissen and R. J. Evans, "Peformance of dynamic programming techniques for track-before-detect," *Aerospace and Electronic Systems, IEEE Transactions on*, vol. 32, no. 4, pp. 1440–1451, 1996. 12

[159] W. Tutte, *Graph Theory*, ser. Cambridge Mathematical Library. Cambridge University Press, 2001. 39

[160] P. A. Viola, M. J. Jones, and D. Snow, "Detecting pedestrians using patterns of motion and appearance," *International Journal of Computer Vision (IJCV)*, vol. 63(2), pp. 153–161, 2005. 19

[161] P. A. Viola and M. J. Jones, "Robust real-time face detection," *International Journal of Computer Vision (IJCV)*, vol. 57, no. 2, pp. 137–154, 2004. 19

[162] A. Waibel, H. Steusloff, and R. Stiefelhagen, "CHIL: Computers in the Human Interaction Loop," Tech. Rep. 23

[163] D. Walden, "The bellman-ford algorithm and "distributed bellman-ford"," Tech. Rep., 2003. 47

[164] Ward's Automotive Group, *Ward's: World Motor Vehicle Data 2007*, 2007. 1

[165] Wikipedia. Google driverless car. Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/Google_driverless_car [Online. Accessed: 04/04/2014] 6 and 27

[166] H. Winner, S. Hakuli, and G. Wolf, *Handbuch Fahrerassistenzsysteme*. Vieweg + Teubner, 2009. 27

[167] C. Wojek, S. Walk, S. Roth, K. Schindler, and B. Schiele, "Monocular visual scene understanding: Understanding multi-object traffic scenes," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 35, no. 4, pp. 882–897, April 2013. 27

[168] World Health Organization, "Global status report on road safety 2013." 1 and 6

[169] World Health Organization, "Disease and injury country estimates," Tech. Rep., 2004. 2

[170] B. Wu and R. Nevatia, "Detection and tracking of multiple, partially occluded humans by bayesian combination of edgelet part detectors," *International Journal of Computer Vision (IJCV)*, vol. 75, no. 2, pp. 247–266, 2007. 12

[171] B. Wu and R. Nevatia, "Detection of multiple, partially occluded humans in a single image by bayesian combination of edgelet part detectors," in *Proc. IEEE International Conf. on Computer Vision (ICCV)*, 2005. 20

[172] C. Wu, G. Zhao, and B. Ou, "A fuel economy optimization system with applications in vehicles with human drivers and autonomous vehicles," *Transportation Research Part D: Transport and Environment (TRD)*, vol. 16, no. 7, pp. 515 – 524, 2011. 3

[173] J. Xing, H. Ai, and S. Lao, "Multi-object tracking through occlusions by local tracklets filtering and global tracklets association with detection responses," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2009. 18

[174] F. Xiong, O. I. Camps, and M. Sznaier, "Dynamic context for tracking behind occlusions," in *Proc. of the European Conf. on Computer Vision (ECCV)*, 2012. 18

[175] B. Yang, C. Huang, and R. Nevatia, "Learning affinities and dependencies for multi-target tracking using a crf model," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2011. 18, 23, and 24

[176] B. Yang and R. Nevatia, "An online learned crf model for multi-target tracking," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2012. 18

[177] B. Yang and R. Nevatia, "Online learned discriminative part-based appearance models for multi-human tracking," in *Proc. of the European Conf. on Computer Vision (ECCV)*, 2012. 18

[178] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *ACM Computing Surveys (CSUR)*, vol. 38, no. 4, Dec. 2006. 12, 18, and 73

[179] S.-I. Yu, Y. Yang, and A. Hauptmann, "Harry potter's marauder's map: Localizing and tracking multiple persons-of-interest by nonnegative discretization," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2013. 18

[180] L. Zhang, Y. Li, and R. Nevatia, "Global data association for multi-object tracking using network flows," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2008. ix, xi, 10, 11, 12, 18, 20, 31, 33, 34, 44, 45, and 73

[181] L. Zhang and L. van der Maaten, "Structure preserving object tracking," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2013. 11 and 18

[182] W. Zhang, G. J. Zelinsky, and D. Samaras, "Real-time accurate object detection using multiple resolutions," in *Proc. IEEE International Conf. on Computer Vision (ICCV)*, 2007. 19

[183] Q. Zhu, S. Avidan, M. Yeh, and K. Cheng, "Fast human detection using a cascade of histograms of oriented gradients," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2006. 19

[184] J. Ziegler, P. Bender, M. Schreiber, H. Lategahn, T. Strauss, C. Stiller, T. Dang, U. Franke, N. Appenrodt, C. Keller *et al.*, "Making bertha drive? an autonomous journey on a historic route," *IEEE Intelligent Transportation Systems Magazine*, vol. 6, no. 2, pp. 8–20, 2014. 3, 4, 6, and 27

[185] F. Ziliani, S. Velastin, F. Porikli, L. Marcenaro, T. Kelliher, A. Cavallaro, and P. Bruneaut, "Performance evaluation of event detection solutions: the creds experience," in *Proc. IEEE International Conf. on Advanced Video and Signal-Based Surveillance (AVSS)*, Sept 2005, pp. 201–206. 23