Hochschule für Technik, Wirtschaft und Kultur Leipzig

Fakultät für Informatik, Mathematik und Naturwissenschaften

Masterarbeit

Echtzeitfähiges Objekt-Tracking in Frontkameraaufnahmen von Fahrzeugen

vorgelegt von: Tino Weidenmüller
Matrikelnummer: 66761

Betreuender Professor: Prof. Dr. rer. nat. Sibylle Schwarz
Betrieblicher Betreuer: M.Sc. Marcel Graef

Stand: 28. Februar 2018

Eidesstattliche Erklärung

Ich versichere, dass die Masterarbeit mit dem Titel „Echtzeitfähiges Objekt-Tracking in Frontkameraaufnahmen von Fahrzeugen“ nicht anderweitig als Prüfungsleistung verwendet wurde und diese Masterarbeit noch nicht veröffentlicht worden ist. Die hier vorgelegte Masterarbeit habe ich selbstständig und ohne fremde Hilfe abgefasst. Ich habe keine anderen Quellen und Hilfsmittel als die angegebenen benutzt. Diesen Werken wörtlich oder sinngemäß entnommene Stellen habe ich als solche gekennzeichnet.

Tino Weidenmüller

Ort, Datum

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Fahrerassistenzsysteme sind bereits weit verbreitet und erhöhen Komfort sowie Sicherheit im Straßenverkehr. Auch autonomes Fahren schreitet weiter voran und kommt der Marktreife immer näher. Für beide Systeme sind eine Vielzahl von Sensoren notwendig um andere Fahrzeuge, Passanten, Straßenschilder oder andere Objekte im Fahrbahnbereich zu erkennen. Weit verbreitet sind z. B. Laserscanner und Kameras. Kameras sind hierbei eine günstige und einfache Methode der Datengewinnung.

Die Verarbeitung von Bildern hat in den letzten Jahren mit dem Aufkommen konvoluter Neuronaler Netze eine erhebliche Qualitätsverbesserung erlebt. Zu diesem Thema hat Philipp Kleinhenz, bei Sedenius Engineering, bereits eine Bachelorarbeit zum Thema *„Detektion und Klassifikation von Objekten im Fahrbahnbereich mit Verfahren des maschinellen Lernens“* geschrieben[1]. In seiner Arbeit hat er einen Objektdetektor trainiert, welcher mehrere Objekte in Bildern erkennen und markieren kann. Auf der Grundlage dieser Daten ist es möglich Objekte zu tracken. Das bedeutet, jedem Objekt eine Identifikation zu geben und es über einen Zeitraum zu beobachten, auch wenn der Detektor es nicht mehr sieht.

Der Objekttracker soll dabei dazu fähig sein ein Objekt weiterzverfolgen, auch wenn es nicht vom Detektor gesehen wird.

1.2 Zielstellung

Das Ziel dieser Arbeit ist der Entwurf, sowie die Implementierung von sog. Multi Object Tracking (MOT) in Videoaufnahmen, welche durch Frontkameras von Fahrzeugen aufgenommen wurden. Der Objekttracker soll nach dem Prinzip „tracking by detection“ arbeiten, was bedeutet, dass im Sichtfeld zuerst Objekte (z. B. Fahrzeuge, Fußgänger) durch einen Detektor erkannt und klassifiziert werden [2].

Auf der Grundlage dieser soll ein Objekttracker erstellt werden. Da der Objekttracker mit Daten eines Detektors arbeitet, ist es ihm möglich auch Objektklassen zu tracken, welche bei der Entwicklung des Objekttrackers noch nicht in Betracht gezogen wurden, auch wenn das Tracking dieser Objektklassen nicht optimiert wär.

Als Detektor wird in erster Linie der von Phillipp Kleinhenz erstellte SSD-Detektor [1] genutzt.

Als softwaretechnisches Fundament für den Objekttracker wird das Nova-Framework verwendet, wodurch es mit geringem Zeitaufwand möglich ist, Testdaten von verschiedenen Quellen einzubinden und diese zum Evaluieren des Objekttrackers zu nutzen.

Der zugrunde liegende Objektdetektor erreicht eine Verarbeitungsgeschwindigkeit von 30 Bildern pro Sekunde, wenn die Berechnung auf einer Nvidia Titan X ausgeführt wird.

Der Objekttracker kann dabei auf dasselbe Leistungsniveau zugreifen und soll unter den gegebenen Bedingungen echtzeitfähig sein.

Außerdem soll der Objekttracker „Online“arbeiten. Das bedeutet, dass eine Verarbeitung Bild für Bild geschieht. Damit ist es möglich den Objekttracker in einem realen Szenario einzubinden.

1.3 Qualitätskriterien

Zur Evaluierung des Objekttrackers werden Statistiken wie die CLEAR MOT[3] oder MT/PT/ML[4] genutzt. Diese Statistiken werden in der Literatur häufig zur Evaluierung von Trackingalgorithmen eingesetzt, wie z. B. beim KITTI-Tracking-Benchmark[5] oder bei der MOT-Challenge[6]. Diese Benchmark-Tests machen es möglich, den Algorithmus mit bestehenden Algorithmen zu vergleichen. Als weiteres Bewertungskriterium wird die Performance des Algorithmus' genutzt. Hierfür wird im Kapiteln 6 auf die zu Grunde liegende Hardware eingegangen.

Die Evaluierung erfolgt, sofern möglich, mit Evaluatoren, welche von den Datensätzen bereitgestellt werden. Zusätzlich wird mit einen eigens entwickelten Evaluator evaluiert, welcher zusätzliche Statistiken bereitstellen soll, wie detaillierte Fehlerberichte und die Einbeziehung der Detektionsergebnisse zur Evlauierung.

1.4 Gliederung der Arbeit

Zu Beginn der Arbeit, in Kapitel 3 werden bestehende Verfahren vorgestellt.

Das Kapitel 4 erklärt das Konzept des Objekttrackers, welcher im Rahmen dieser Arbeit erstellt wird. Es wird auf die Architektur, sowie die Anbindung an bestehende Schnittstellen eingegangen. Außerdem wird in diesem Kapitel die Funktionsweise des Nova-Frameworks aufgezeigt.

Kapitel 5 wird die Umsetzung des Objekttrackers mithilfe der Konzepte aus Kapitel 4 beschreiben. Es wird auf Probleme eingegangen, welche während der Entwicklung aufgetreten sind.

Kapitel 6 enthält eine Evaluation des Objekttrackers. Hier wird der Objekttracker auf den genannten Datensätzen getestet und es werden, sofern möglich, Vergleichsergebnisse von anderen Algorithmen bereitgestellt.

Zuletzt wird es in Kapitel 7 eine Zusammenfassung der Ergebnisse und einen Ausblick geben.

2 Grundlagen

In diesem Kapitel werden grundlegende Informationen dargelegt, auf welche im weiteren Verlauf referenziert wird.

2.1 Softwaretechnische Grundlagen

In diesem Abschnitt werden softwaretechnische Grundlagen dargelegt. Dabei handelt es sich um Softwarebibliotheken oder Softwareumgebungen, welche zur Umsetzung genutzt werden.

2.1.1 Nova-Framework

Das Nova-Framework stellt die softwaretechnische Grundlage für den Objekttracker und dessen Anbindung an Datenquellen bereit. Daher wird in diesem Abschnitt auf die Funktionsweise des Frameworks eingegangen.

Das Nova-Framework ist ein von Sedenius Engineering entwickeltes Softwareframework. Die Funktionsweise ist mit anderen Frameworks wie dem Automotive Data and Time-Triggered Framework (ADTF) oder Robot Operating System (ROS) verwandt.

Im Nova-Framework werden verschiedene Softwarekomponenten als Plugins geladen. Plugins werden mit einem Interface-System wie bei Microsofts COM-Interface beschrieben. Dabei können je Plugin beliebig viele Klassen exportiert und von anderen Programmen genutzt werden.

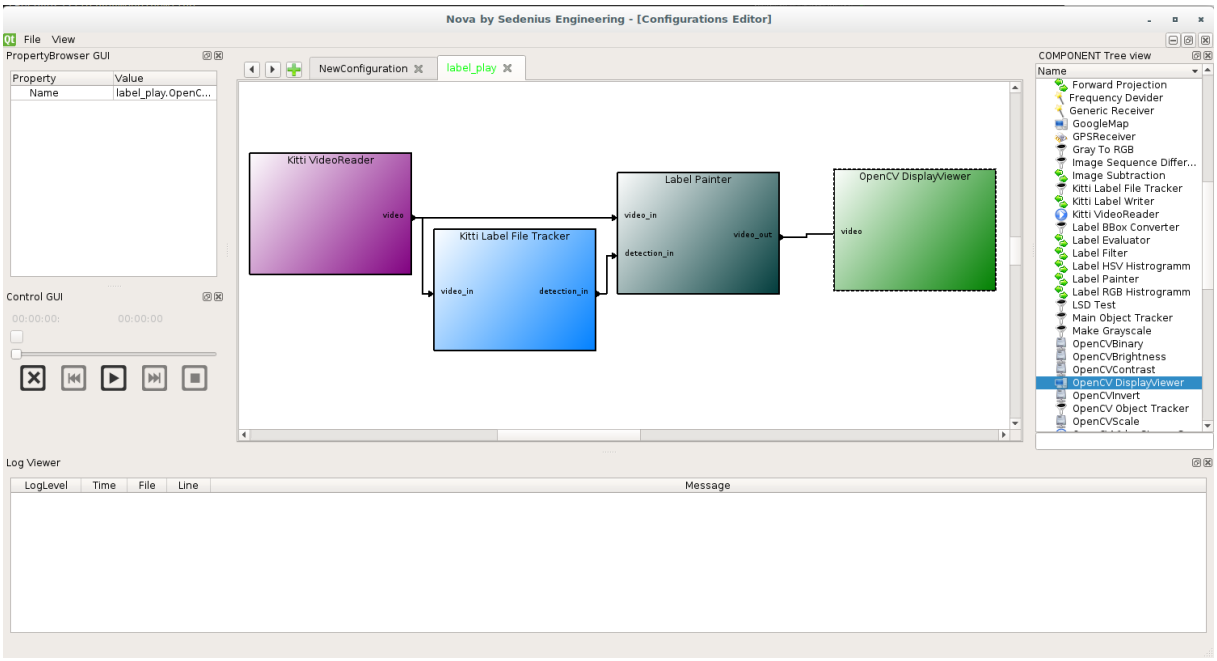

Abbildung 2.1: Diese Grafik stellt das User-Interface des Nova-Frameworks dar. Im Zentrum befindet sich der Konfigurations-Editor, in welchem Komponenten platziert und miteinander verbunden werden können. Am linken Rand befindet sich der Property-Browser und die Control-GUI. Am rechten Rand ist eine Liste aller Komponenten, welche per Drag-and-Drop in den Konfigurations-Editor eingefügt werden können. Am Unteren Rand befindet sich der Log-Viewer, welcher Lognachrichten filtert und anzeigt.

Architektur

Der Kern des Nova-Frameworks ist die Nova-Runtime. Diese lädt Plugins zu laden, verwaltet und generiert Klassen und Objektinstanzen und steuert das Zustandsverhalten des Programms.

Die Nova-Runtime selbst funktioniert wie ein Zustandsautomat. Folgende Zustände sind definiert:

Shutdown Das Programm wird sich in diesem Zustand beenden.

New Das Programm wurde soeben gestartet. Dies ist der Initiale Zustand.

Kernel In diesem Zustand werden die wichtigsten Systemservices geladen.

System Dies ist der Ruhezustand für Kommandozeilenanwendungen.

XSystem Dies ist der Ruhezustand für GUI-Anwendungen. Beim Erreichen dieses Zustandes werden zusätzlich die grafischen Oberflächen der Services gestartet.

Init Dieser Zustand ist für alle Komponenten in aktiven Konfigurationen relevant. Diese werden in diesem Zustand initialisiert.

Ready Dieser Zustand ist die Vorstufe zum Runlevel *Running*.

Running In diesem Zustand wird die Konfiguration gestartet, das bedeutet, dass Datenproduzenten beginnen Daten zu erzeugen und damit die Verarbeitung starten.

Ein Wechsel ist immer nur von einem Runlevel in das nächst höhere oder niedrigere möglich. Die Reihenfolge ist wie in der Aufzählung definiert. Das bedeutet, dass bei einem Wechsel von *New* zu *XSystem* auch die Runlevel *Kernel* und *System* durchlaufen werden.

Mithilfe der Nova-Runtime können Klasseninstanzen erzeugt werden. Diese können bei der Nova-Runtime mit einem eindeutigen Instanznamen registriert werden.

Zur Kommunikation zwischen Modulen gibt es im Nova-Framework zwei Methoden.

Direkt per Interface

Es ist möglich die Speicheradresse eines Objekts bei der Nova-Runtime zu erfragen und auf diesem Objekt eine bestimmte Methode auszuführen, von einem Interface, welches die Klasse unterstützt. Der Nachteil dieser Methode ist, dass der Aufruf nur an ein bestimmtes Objekt geht. Um eine Aktion bei mehreren auf Objekten auszuführen muss der Entwickler explizit eine Schleife erstellen und die Aktion bei jedem Objekt einzeln ausführen. Der Vorteil dieser Methode ist, dass die Verarbeitung garantiert beendet ist wenn die Ausführung der Methode abgeschlossen ist, sie wird synchron ausgeführt.

Per Runtime-Event

Die Nova-Runtime stellt eine Eventschleife bereit, in die jedes Module Events erzeugen kann. Empfänger müssen sich als Observer/Callbacks bei der Nova-Runtime registrieren und erhalten die Events. Das so gesendete Event wird an kein bestimmtes Objekt gesendet, sondern kann von allen verarbeitet werden. Der Nachteil an dieser Methode ist, dass die Verarbeitung asynchron geschieht. Diese Methode ist besonders für Zustandsänderungen wichtig, da diese Änderungen als Runtime-Event von verschiedenen Services empfangen und z. B. visualisiert werden können.

Services

Services dienen dazu die Funktionalität der Nova-Runtime dynamisch zu erweitern. Von jedem Service existiert dafür exakt eine Instanz. Ein Service ist dadurch gekennzeichnet, dass er das Interface nova::IService implementiert. Wird ein solches Objekt bei der Nova-Runtime registriert, erhält es von der Nova-Runtime automatisch ServiceEvents, wenn sich z. B. das Runlevel ändert.

Das Nova-Framework stellt verschiedene Systemservices bereit, welche unter Anderem Komponenten (mehr hierzu im nächsten Paragraphen), eine graphische Oberfläche oder Module zur Datenmanipulation hinzufügen.

Komponenten

Eine Komponente ist ein Softwaremodul, welches Daten erzeugt, graphisch darstellt oder

verarbeitet. Jede Komponente verfügt über eine Menge von Pins. Ein Pin stellt eine mögliche Datenverbindung zu einer anderen Komponente dar. Ein Pin wird dabei als Eingangs- oder als Ausgangspin klassifiziert. Es ist jedoch möglich Daten in beide Richtungen zu übertragen. Die Rückübertragung von Daten wird nicht von jeder Komponente unterstützt. Diese Komponenten ignorieren die eingehenden Daten dann.

Zur Konfiguration hat jede Komponente sog. Properties, welche über eine graphische Oberfläche (Property-Browser, siehe Abbildung 2.2) konfiguriert werden und von der Nova-Umgebung automatisch abgespeichert und wieder geladen werden.

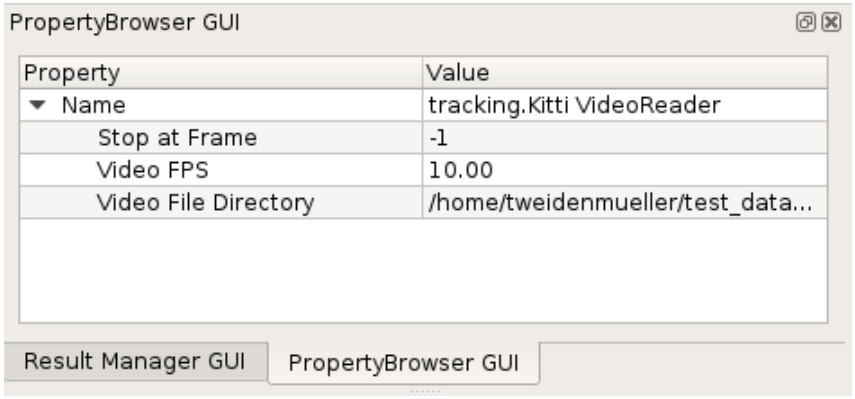

Abbildung 2.2: Die Abbildung zeigt die graphische Oberfläche des Property-Browsers, in welchem die Parameter von Komponenten konfiguriert werden können.

In Abbildung 2.3 sind zwei miteinander verbundene Nova-Komponenten zu sehen. Die linke Komponente(„Kitti VideoReader0“) liest ein Video von der Festplatte und gibt die Bilder auf dem Pin „video“ aus. Die rechte Komponente („OpenCV DisplayViewer0“) empfängt die Bilder über den Pin „video“ und gibt diese über eine graphische Oberfläche aus. Der zu sehende Aufbau ist ein sehr simpler und vermag nicht mehr zu tun als ein Video abzuspielen. Es können prinzipiell weitere Komponenten in den Datenstrom eingesetzt werden um z. B. das Bild zu verarbeiten oder es in eine Datei abzuspeichern. Dabei ist es möglich beliebig lange Verarbeitungsketten aufzubauen, sofern der Computer auf dem das Framework läuft über ausreichend Verarbeitungskapazitäten verfügt.

Abbildung 2.3: Diese Abbildung zeigt zwei Nova-Komponenten, welche miteinander verbunden sind.

Der Objekttracker, welcher im Rahmen dieser Arbeit entwickelt wird, soll selbst eine Komponente sein, um ihn so mit beliebigen Datenquellen verbinden zu können.

2.1.2 OpenCV

OpenCV ist eine Programmbibliothek, welche Funktionen zur Verarbeitung von Bildern enthält.

OpenCV ist in C++ programmiert. Es besteht auch eine Anbindung an Python [7].

Im Rahmen dieser Arbeit werden verschiedene Funktionen der OpenCV-Bibliothek, wie z. B. Bildkonvertierung oder optischer Fluss, genutzt.

Die, für diese Arbeit, wichtigsten Funktionen sind die Implementierung des Lucas-Kanade-Trackers, die Konvertierung von Bildern in verschiedene Formate sowie das einfache Ausschneiden vom Bildausschnitten. Es wird darüber hinaus auf verschiedene Merkmalsdetektoren (SIFT, SURF, ...) zum testen zurückgegriffen.

2.2 Algorithmische Grundlagen

In diesem Abschnitt werden Verfahren vorgestellt, welche im Rahmen dieser Arbeit Anwendung finden.

2.2.1 Medianflow-Tracker

Beim Medianflow-Tracker handelt es sich um einen Algorithmus zum Tracken eines einzelnen Objektes. Der Algorithmus baut auf Verfahren des optischen Flusses auf (häufig wird das Lukas-Kanade-Verfahren genutzt) [8].

Der Medianflow-Tracker berechnet in jedem Zeitschritt die Verschiebung und die Größenänderung eines Objektes. Zusätzlich wird ein Qualitätsmaß berechnet, mit dem festgestellt wird, wie stabil die Vorhersage der Position ist.

Der Algorithmus braucht in jedem Zeitschritt das Bild des vorherigen Zeitschritts (B_{alt}), das Bild des aktuellen Zeitschritts (B_{neu}) und die Bounding-Box des Objektes im vorherigen Zeitschritt.

Es wird eine Menge an Punkten innerhalb der alten Bounding-Box bestimmt. Dies kann über verschiedene Verfahren geschehen. Häufig wird hierbei ein Gitter über die Bounding-Box gelegt oder die Punkte werden mit einem Merkmalspunktedetektor, wie z. B. FAST, berechnet.

Positionsbestimmung

Danach werden diese Punkte mithilfe des Lucas-Kanade-Trackers in das nächste Bild vorausberechnet. Die Lucas-Kanade Methode berechnet die neue Position jedes einzelnen Punktes mithilfe des optischen Flusses [9]. Diese neuen Positionen werden hier durch V_n dargestellt, wobei n der Index des Punktes ist.

V_n ist die vorausberechnete Position von P_n.

V_n und P_n beschreiben spezifische Punkte, wohingegen P und V die Menge aller Punkte beschreiben.

$$P = \{P_1, P_2, ...\} \tag{2.1}$$

$$V = \{V_1, V_2, ...\} \tag{2.2}$$

Die Berechnung sieht wie folgt aus:

$$V = \text{LK\_track}(Bild_{\text{alt}}, Bild_{\text{neu}}, P) \tag{2.3}$$

Daraus entsteht eine Menge an Tupeln, bestehend aus dem Startpunkt P_n und seinem korrespondierenden neuen Punkt V_n. Aus den Punktemengen P und V lässt sich eine Menge an Bewegungsvektoren (BV) berechnen [10].

$$\text{BV} = \{P_1 - V_1; P_2 - V_2; ...\} \tag{2.4}$$

Aus diesen Bewegungsvektoren werden zwei Listen gebildet. Die der X-Werte und separat die der Y-Werte [10].

$$\text{BV}_x = \{P_{1x} - V_{1x}; P_{2x} - V_{2x}; ...\} \tag{2.5}$$

$$\text{BV}_y = \{P_{1y} - V_{1y}; P_{2y} - V_{2y}; ...\} \tag{2.6}$$

Als Gesamtverschiebung wird der Median aller X- und der Median aller Y-Verschiebung genutzt [8].

$$\text{Gesamtverschiebung} = (\text{median}(BV_x); \text{median}(BV_y)) \tag{2.7}$$

Die Gesamtverschiebung stellt die Bewegung dar, die ein Objekt seit dem letzten Bild durchgeführt hat im Bezugssystem des Bildes.

Größenänderung

Aufbauend auf den Berechnungen der Positionen werden zur Berechnung der Größenänderung Tupel aus je zwei Punkten gebildet. Für jeden Tupel wird die Distanz der beiden Punkte gebildet [8]. Um eine möglichst genaue Vorhersage zu treffen, werden alle möglichen Tupel aus den gegebenen Merkmalspunkten gebildet. Wie bereits bei der Positionsbestimmung beschreibt P die Menge der ursprünglichen Punkte und V die Menge der aus P vorausberechneten Punkte.

$$\text{Diff}_{Pi} = \text{länge}(P_n - P_m) \tag{2.8}$$

$$\text{Diff}_{Vi} = \text{länge}(V_n - V_m) \tag{2.9}$$

Dabei ist es nicht wichtig welche Punkte konkret voneinander subtrahiert werden, wichtig ist, dass für $Diff_{Pi}$ und $Diff_{Vi}$ dieselben Werte für n und m genutzt werden [8].

Im nächsten Schritt wird das Verhältnis von Differenzen mit dem selben Index gebildet und diese werden in eine Liste gespeichert [8].

$$\text{Skallierungsdifferenzen} = \left\{ \frac{\text{Diff}_{P1}}{\text{Diff}_{V1}}; \frac{\text{Diff}_{P2}}{\text{Diff}_{V2}}; ... \right\} \tag{2.10}$$

Die Gesamtskalierung ist der Median dieser Werte.

$$\text{Gesamtskalierung} = \text{median}(\text{Skallierungsdifferenzen}) \tag{2.11}$$

Die Gesamtskalierung ist der Faktor um den die Bounding-Box wächst (bei Gesamtskalierung > 1) oder schrumpft (bei Gesamtskalierung < 1).

Forward-Backward-Error (FB-Fehler)

Der FB-Fehler soll die Qualität des Trackings bestimmen. Zur Berechnung des FB-Fehlers werden die Berechnungen der Positionsänderung und der Größenänderung vorausgesetzt[8].

Die Merkmalspunkte, welche mit dem Lucas-Kanade-Tracker in das nächste Bild vorausberechnet wurden (V_n), werden wieder zurück in das alte Bild berechnet. Die Zurückberechneten Punkte werden hier mit W_n referenziert. Wenn die Vorhersage stabil ist, sollten die ursprünglichen Punkte den neuen Punkten entsprechen und der FB-Fehler ist möglichst nahe an der 0 [8].

Die Berechnung des FB-Fehlers sieht wie folgt aus:

$$W = \text{LK\_track}(\text{Bild}_{\text{Neu}}, \text{Bild}_{\text{Alt}}, V) \tag{2.12}$$

Zuerst wird die Liste aller Differenzen berechnet:

$$\text{FBVektor} = \{\text{länge}(W_1 - P_1); \text{länge}(W_2 - P_2); ...\} \tag{2.13}$$

Der Gesamtfehler ist der Median aller Teilfehler:

$$\text{FBFehler} = \text{median}(\text{FBVektor}) \tag{2.14}$$

Überschreitet der $\text{FB} - \text{Fehler}$ einen gegeben Grenzwert, wird das Tracking als instabil angesehen [8].

Nutzung des Medianflow

Der Medianflow-Algorithmus wird von dem Tracker, welcher im Rahmen dieser Arbeit entwickelt wird genutzt, mehr hierzu in Kapitel 5.

Der Medianflow-Algorithmus wird auch vom MDP-Verfahren (siehe Abschnitt 3.1) und in Teilen vom NOMT (Abschnitt 3.2) genutzt.

2.3 Evaluation

In diesem Abschnitt werden bestehende Evaluationsmethoden für Objekttracker beschrieben.

2.3.1 CLEAR MOT

Bei diesem Bewertungsmaß wird im ersten Schritt geprüft ob es für jedes Objekt das vom Detektor/Tracker erkannt wird ein entsprechendes Objekt in den Ground-Truth-Daten gibt. Dafür wird berechnet wie groß das Verhältnis der Fläche der Schnittmenge zweier Objekte zu deren Vereinigungsmenge ist (Überlappungsverhältnis). Eine genaue Erklärung zur Berechnung dieses Werten ist im Kapitel 5.2 zu finden. Ist ein bestimmter Wert überschnitten wird das Objekt als erkannt angesehen. Dabei kann jedem Ground-Truth-Objekt nur ein detektiertes Objekt zugewiesen werden [3].

Für alle korrekt zugewiesenen Objekte wird das Überlappungsverhältnis gespeichert und zu einem Gesamtdurchschnitt, genannt Multiple Object Tracking Precision (MOTP), zusammengefasst [3]. Es werden außerdem in jedem Frame die True Positives (TP), False Positives (FP), False Negatives (FN), ID-Switches (IDS) und die Gesamtzahl aller Ground-Truth-Objekte (GT) gezählt. Aus diesen Werten wird die Multiple Object Tracking Accuracy (MOTA) mit folgender Formel berechnet:

$$MOTA = 1.0 - \frac{\mathrm{FP} + \mathrm{FN} + \mathrm{IDS}}{\mathrm{GT}} \tag{2.15}$$

Hierbei kann es vorkommen, dass die Summe aus FP, FN und IDS größer wird als die Gesamtzahl der GT-Objekte, wodurch der MOTA-Wert negativ werden kann. Dieser Effekt tritt meist zu Beginn einer Testreihe für einige Sekunden ein, stabilisiert sich aber meist nach wenigen Sekunden. Diese Eigenschaft ist jedoch auch einer der größten Vorteile des MOTA-Wertes, da jeder Fehler in der Bewertung einen Einfluss hat [3].

Verschiedene Implementierung der CLEAR MOT-Statistik setzen unterschiedlichen Maßstäbe an, welche Objekte in die Evaluation mit einbezogen werden oder nicht. Daher kann die Menge der einzelnen Werte je nach Implementierung leicht variieren.

2.3.2 MT/PT/ML

Dieses Bewertungsmaß besteht im Wesentlichen aus den drei Werten Mostly Tracked (MT), Partly Tracked (PT) und Mostly Lost (ML). Um diese zu bestimmen werden alle Ground-Truth-Tracks Frame für Frame durchgegangen. Frames in denen das Objekt nicht in den Ground-Truth-Daten auftaucht werden ignoriert. Ist es vorhanden wird geprüft ob es in den Detektionen auch vorhanden ist. Wenn eine Übereinstimmung gefunden wurde, wird geprüft ob diese dieselbe Identifikationsnummer (ID) haben. Auf diese Weise wird für jeden Frame eines Tracks bestimmt, ob es korrekt erkannt (R), überhaupt nicht erkannt (F) oder mit neuer ID erkannt wurde (IDS). Anschließend wird gezählt, wie viele der

Frames der jeweiligen Klassifizierung angehören. Im Anschluss wird ein Trackverhältnis (TV) mit folgender Formel gebildet [4]:

$$TV = \frac{R}{R + F + IDS} \tag{2.16}$$

Ist das Trackverhältnis kleiner als 20%, wird der Track als ML eingestuft. Ist der Wert über 80% gilt der Track als MT. Liegt der Wert zwischen den beiden, gilt der Track als PT. Für jede dieser Klassifizierungen wird die Gesamtzahl der ihr zugehörigen Tracks gezählt und durch die absolute Gesamtzahl der GT-Tracks geteilt. Das Ergebnis ist ein Verhältnis wie hoch der Anteil der Tracks ist, welche größtenteils getrackt (MT), teilweise getrackt (PT) oder kaum bis gar nicht getrackt (ML) wurden [4].

2.3.3 genutzte Datensätze

Zur Evaluierung des Trackers wird auf zwei Datensätze zurückgegriffen.

KITTI-Benchmark

Der KITTI-Benchmark besteht auf 21 Trainingssequenzen, welche Ground-Truth-Daten enthalten und 29 Trainingssequenzen ohne Ground-Truth-Daten. Zu jeder Sequenz (sowohl Test- als auch Trainingssequenzen) gibt es zwei verschiedene vordetektierte Daten. Einmal die Resultate eines L-SVM-Detektors und die eines Regionlet-Detektors. Als zusätzliche Daten können Stereobilder, GPS/IMU-Daten oder Laserscanner-Pointclouds genutzt werden [11].

Wichtig ist beim KITTI-Benchmark zu erwähnen, dass die Ground-Truth-Daten an einigen Stellen fehlerhaft sind. Von jedem Objekt wird zusätzlich zur Position noch mit angegeben, wie stark das Objekt verdeckt ist. Diese Information ist in einigen Bildern sehr schlecht. Es gibt Objekte, welche vollständig verdeckt sind aber trotzdem als vollständig sichtbar angegeben werden. Dies verfälscht die Evaluation des Trackers [11].

Im KITTI-Benchmark sind verschiedene Klassen markiert, die Evaluation wird jedoch nur mit Autos und Fußgängern durchgeführt, da nur diese in ausreichendem Maße vorhanden sind [11].

MOT-Challenge

Der Datensatz, welchen die MOT-Challenge nutzt wird jedes Jahr etwas abgeändert. 2015 bestand der Datensatz aus 11 Test- sowie Trainingssequenzen. Die Trainignssequenzen waren auch hier mit Ground-Truth-Daten ausgestattet. Für alle Sequenzen wurde ein Detektionsergebnis mitgeliefert [6].

2017 bestand der Datensatz aus 7 Test- sowie Trainingssequenzen. Auch hier wurden Ground-Truth-Daten zu bei den Trainingsdaten mitgeliefert. Für alle Sequenzen wurden drei verschiedene Detektionsergebnisse mitgeliefert (SDP, FRCNN, DPM) [6].

In den Ground-Truth-Daten der MOT-Challenge sind ausschließlich Fußgänger markiert [6].

Im Rahmen dieser Arbeit wird dabei auf den Datensatz von 2015 zugegriffen, da der Referenzalgorithmus der im Rahmen dieser Arbeit ausgeführt werden konnte ausschließlich mit diesem kompatibel ist.

3 State of the Art

In diesem Kapitel werden Systeme und Methoden vorgestellt, welche interessante Konzepte darstellen. Es werden ausschließlich Methoden vorgestellt, welche in Ausreichendem Maße offengelegt werden und es damit möglich ist Teile dieser Methoden in eigenen Lösungen zu nutzen, bzw. deren Ergebnisse nachzuvollziehen. So listet z. B. die Website des KITTI-Tracking Benchmarks eine Vielzahl von Algorithmen auf, von denen jedoch die meisten ausschließlich aus Resultaten bestehen [11].

3.1 Markov Decision Process (MDP)

Das MDP-Tracking Verfahren ist ein Trackingslgorithmus, welcher 2015 vorgestellt wurde [10]. Es setzt das Tracking mit einem sog. Markov-Decision-Process um, welcher für den Tracker namensgebend ist. Der Markov-Decision-Process ist ein Verfahren um eine Entscheidungsfindung in teilweise zufälligen Systemen zu realisieren [12].

Das MDP-Tracking Verfahren arbeitet nach dem „tracking by detection“-Prinzip und teilt den Zustand eines Objekts in vier Gruppen auf [10].

Active Das Objekt wurde gerade neu entdeckt und muss noch auf Gültigkeit geprüft werden.

Tracked Das Objekt wurde vom Tracker als richtig Positives klassifiziert.

Lost Das Objekt wurde in einem früheren Frame getracked kann aber im aktuellen Frame nicht gefunden werden.

Inactive Das Objekt wurde entweder als invalide empfunden oder war zu lange im Zustand Lost. Objekte in diesem Zustand können als endgültig gelöscht angesehen werden.

Das MDP besteht aus dem Tupel (S, A, T(), R()):

S Menge an Zuständen in denen sich ein Objekt befinden kann. $s \in S$

A Aktionen $a \in a_1, \ldots, a_7$, welche auf einem Objekt ausgeführt werden können

T Zustandsübergangsfunktion $T : S \times A \to S$, beschreibt den Effekt jeder Aktion auf den Zustand eines Objekts

R Belohnungs(Reward-)funktion $R : S \times A \to \mathbb{R}$, bestimmt die Belohnung für das Ausführen einer Aktion

Für jeden Zustand (Active/Inactive/Tracked/Lost) gibt es eine sog. Policy (π), welche als Resultat eine auszuführende Aktion (a_1-a_7) bestimmt ($\pi : S \to A$). Das Problem des Objekttrackings wird als ein Datenassoziationsproblem behandelt. Diese Policies werden trainiert. Das Ziel des Policylearning ist es, eine Policy zu finden, welche die Gesamtbelohnung maximiert.

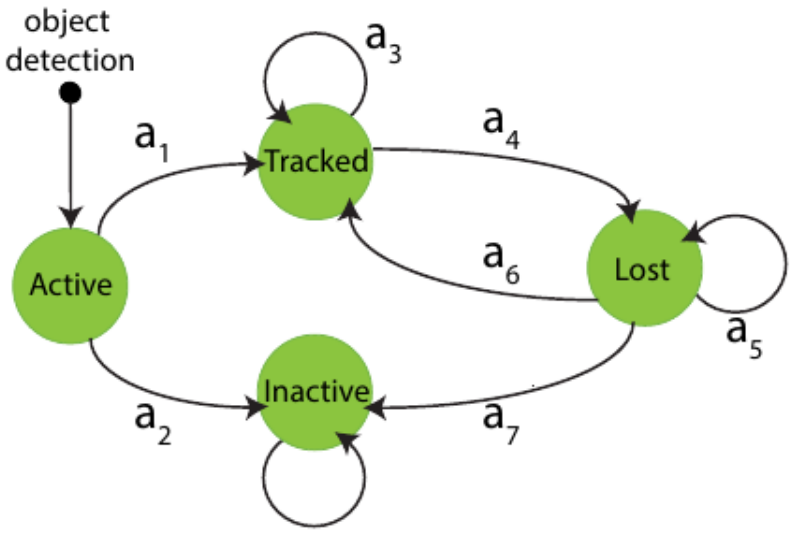

Abbildung 3.1: Dieses Bild zeigt die Zustände des MDP-Verfahrens, sowie die Zustandsübergänge (a_i, wobei i die Klassifizierung der Aktion, mit einem Start- und Endzustand darstellt). [10]

Um nicht detektierte Objekte weiter zu verfolgen nutzt das MDP-Verfahren den Medianflow-Tracker, welcher in Abschnitt 2.2.1 beschrieben ist.

In der MOT-Challange von 2015 erreicht der MDP-Tracker eine Verarbeitungsfrequenz von 1,1 Hz [6].
Für detailliertere Informationen wird auf die Veröffentlichung [10] verwiesen.

Die Einteilung des Zustandes in dem sich ein Objekt befinden kann wurde im Rahmen dieser Arbeit in den eigenen Tracker übernommen. Siehe hierfür Kapitel 4.6.

3.2 NOMT

Das NOMT-Verfahren ist ein Trackingalgorithmus, welcher sich eine bestimmte Eigenschaft des optischen Flusses zu Nutze macht. Diese Eigenschaft besagt, dass es im optischen Fluss immer Fehler in einzelnen Flussvektoren geben wird, über viele Flussvektoren hinweg ist das Resultat jedoch korrekt [13].

Eine besondere Eigenschaft des Verfahrens ist es, Resultate im Nachhinein, durch zusätzliche Informationen zu korrigieren. Das bedeutet, dass Daten, welche in der Vergangenheit liegen noch geändert werden können. Die Online-Fähigkeit des Verfahrens geht dabei jedoch nicht verloren, das bedeutet, die Ausgabe erfolgt nach wie vor Bild für Bild und kann somit in einer realen Anwendung eingesetzt werden [13].

Um bereits beobachtete Objekte mit neuen Detektionen zu vergleichen und zu assoziieren, wird der sog. Aggregated Local Flow Descriptor (ALFD) genutzt. Die Methode codiert relative Bewegungsmuster zwischen zwei Bounding-Boxen innerhalb eines bestimmten Zeitabstandes. Hierfür werden bestimmte Schlüsselpunkte beobachtet (sog. Interest Point

Trajectory (IPT)). Die Annahme hierbei ist, dass es bei gleichen Objekten viele IPTs gibt, welche sich relativ zur Bounding-Box an der selben Position befinden. Um diesen Vergleich durchzuführen, wird auf der Bounding-Box ein Gitter ausgelegt. Liegt ein Punkt im alten Bild im selben Feld wie im neuen Bild, wird der Punkt als passend angesehen. Abbildung 3.2 illustriert den Sachverhalt [13].

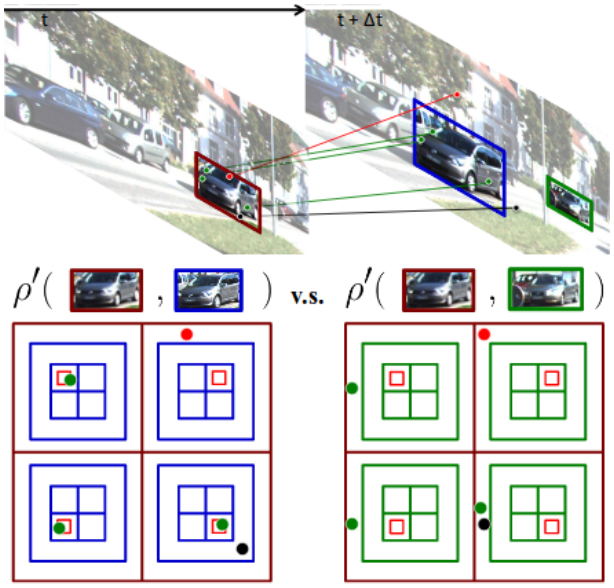

Abbildung 3.2: Dieses Bild zeigt den Vergleich zweier Objekte im NOMT-Verfahren. Auf der linken Seite ist das alte Bild zu sehen und auf der Rechten das Neue. Am unteren Rand ist das Gitter zu sehen, welches über die Bounding-Boxen gelegt wird. Grün sind die Punkte markiert, welche sich an der richtigen Position befunden haben. Punkte, welche nicht an der richtigen Position waren, sind rot markiert. Die Roten Rechtecke stellen die erwartete Position des Punktes dar. Zur Vereinfachung wurde in der Abbildung nur ein 2x2-Gitter genutzt. Im Betrieb wird dieses Gitter auf 4x4 erweitert [13].

Mithilfe dieser Methode wird eine Wahrscheinlichkeit berechnet, dass es sich bei den beiden Objekten um das Selbe handelt. IPTs werden mithilfe des FAST detektiert [13, 14]. Diese werden mit den bereits berechneten Punkten zusammengelegt, und es werden die Punkte entfernt, deren Distanz zu einem bereits bestehenden Punkt kleiner als 4 Pixel ist. Danach werden die Punkte mit der optischen Fluss-Methode von G. Farnbäck in das nächste Bild vorausprojeziert und von dort aus wieder zurück projeziert [15]. Mit den zurückprojezierten Punkten wird für jeden Punkt der FB-Fehler (genaue Erläuterung dazu in Kapitel 5.3.2) berechnet. Jeder Punkt, dessen FB-Fehler größer als 10 Pixel ist wird gelöscht.

Mit allen verbleibenden IPTs wird geprüft, wie viele der Punkte im alten Bild im selben Gitterfeld sind wie im neuen Bild.

Mithilfe dieser Informationen wird eine Menge von Hypothesen generiert, welche eine mögliche Kombination von bereits getrackten Objekten und neuen Detektionen darstellen. Auf diesen Informationen wird ein Conditional Random Field (CRF) Model erstellt [13, 16]. Daraus wird die plausibelste Lösung der Zuweisung als Output des Algorithmus gewählt. Abbildung 3.3 stellt die Funktionsweise in einzelnen Schritten dar.

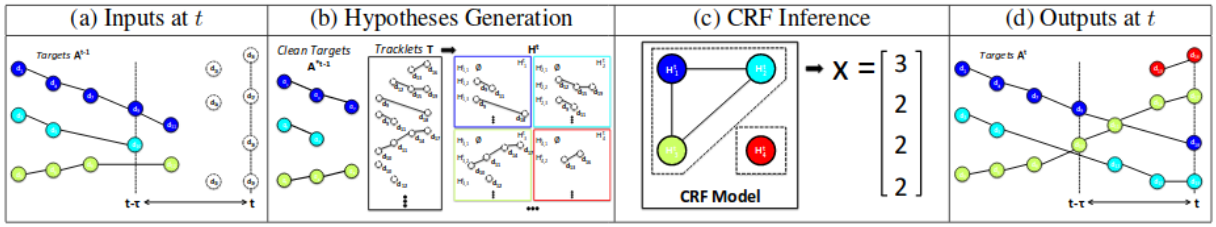

Abbildung 3.3: Dieses Bild zeigt die Arbeitsschritte, welche vom NOMT-Verfahren durchgeführt werden. (a) Für eine Menge an existierenden Objekten A^{t-1} und Detektionen $D^t_{t-\tau}$ generiert das Verfahren (b) eine Menge an Hypothesen $H^t(x)$ mit Hilfe der Tracklets τ. Aus diesen Hypothesen wird ein CRF-Model generiert. (c) Daraus wird die Konsistenteste Lösung x ausgewählt. (d) Die Ausgangsdaten werden generiert, indem die bisherigen Objekte A^{t-1} mit der Hypothese $H^t(x)$ zusammengesetzt wird [13].

3.3 Combined Image- and World-Space Tracking (CIWT)

Das CIWT-Verfahren zeichnet sich, von allen hier Vorgestellten, dadurch aus, dass es für den KITTI-Benchmark das einzige Verfahren ist, welches Stereo-Bilder nutzt und auf dieser Basis 3D-Bounding-Boxen generiert [17]. Aus diesem Grund wird es hier vorgestellt, da die Verarbeitung von Stereo-Bildern zur Weiterentwicklung des eigenen Trackers einen vielversprechenden Ansatz bietet.

Das Verfahren setzt eine Kombination aus 2D- und 3D-Tracking um. Objekte werden in 3D getrackt, sofern 3D-Informationen vorhanden sind. Sind diese nicht vorhanden, wird das Objekt in 2D getrackt. Für diesen Zweck wurde im Rahmen des CIWT-Verfahrens ein eigener 2D-3D Kalman-Filter entwickelt. Dieser verfügt über einen Zustand der aus einer 2D- und einer 3D-Komponente besteht. Damit ist es möglich, dass die 3D-Komponente die Positionsbestimmung in 2D unterstützt und umgekehrt [17].

Abbildung 3.4 zeigt die Verarbeitungspipeline des CIWT-Verfahrens. Im ersten Schritt werden Daten des Detektors mit den 3D-Informationen zusammengeführt. Dafür werden 3D-Vorschläge (sog. *3D-Proposals*) gemacht. Hierbei gilt es festzustellen, welche 3D-Punkte zum Objekt und welche zum Hintergrund oder einem anderen Vordergrundobjekt gehören. Abbildung 3.5 zeigt die Resultate dieses Verarbeitungsschrittes. Dabei entsteht eine Menge von sog. Observations. Eine Observation ist eine Kombination aus einem Detektionsergebnis und einem 3D-Objekt. Aus diesen Observations werden die Besten ausgewählt. Diese Observations werden im nächsten Schritt getrackt, um eine Menge an Hypothesen zu erstellen. Auch hier werden die am besten passenden Hypothesen ausgewählt. Die Auswahl der Hypothesen erfolgt mithilfe eines CRF-Models [17].

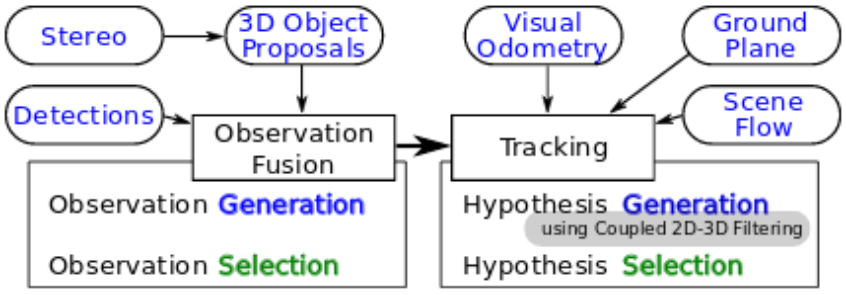

Abbildung 3.4: Diese Grafik zeigt die Verarbeitungspipeline im CIWT-Verfahren. Zu sehen ist, dass Daten aus verschiedenen Quellen fusioniert werden [17].

Abbildung 3.5: Die Grafik zeigt die Resultate der Zwischenschritte visuell dargestellt. Die erste Reihe ist eine Kombination aus Rohbildern, Detektionsergebnissen und den 3D-*Proposals*. Die mittlere Reihe stellt die 3D-Punktwolke dar, in die die einzelnen Punkte je nach Bounding-Box-Zugehörigkeit eingefärbt sind. Die dritte Reihe zeigt die 3D-Punktwolke in der die Punkte herausgefiltert wurden, welche nach Ansicht des Algorithmus zu dem eigentlichen Objekt gehören. Diese Reihe stellt die 3D-*Proposals* dar [17].

4 Konzeption

In diesem Kapitel wird auf der Basis des Nova-Frameworks ein Objekttracker entworfen. Hier werden Schnittstellen festgelegt, Kernprobleme dargestellt und Lösungsansätze vorgestellt.

4.1 Herausforderungen an den Algorithmus

Dieses Kapitel beschreibt Kernprobleme, welche der Tracker möglichst gut bewältigen soll. Dabei sind einige trivial, andere jedoch komplex.

Objekt-IDs beibehalten

Dieses Problem beschreibt den sog. ID-Switch. Das bedeutet, dass ein Objekt verloren und im nächsten Frame wiedererkannt wird, jedoch mit einer anderen Objekt-ID. Das bedeutet es wird nicht als das Objekt erkannt, welches bereits gesehen wurde. Diese Art der Fehler sind zu vermeiden. Aus diesem Problem ergibt sich teilweise das Nächste.

Wiedererkennen nach Trackingverlust (Recover)

Hierbei geht es darum, dass ein Objekt wiedererkannt wird, auch wenn es einige Frames nicht zu sehen ist.

Multi Object Tracking (MOT) fähig sein

Der Tracker muss dazu fähig sein viele Objekte gleichzeitig zu tracken.

Drift

Die Verfolgung des Objekts wird durch die kontinuierliche optische Veränderung eines Objekts erschwert [18].

4.2 Allgemein

Objekttracking nach dem Prinzip „tracking by detection“ ist ein Datenassoziation Problem [10]. Grundsätzlich geht es darum eine Assoziation zwischen detektierten Objekten über mehrere Frames hinweg zu schaffen. Hierfür sollte jedes Objekt mit bestimmten Merkmalen (Features) beschrieben werden, um Die Objekte anhand ihrer Merkmale im nächsten Frame wiederzufinden.

Die Merkmale sollten sich je nach dem Kontext des Objektes ändern. Details hierzu sind in den folgenden Unterkapiteln zu finden.

4.3 Modularer Aufbau

Die Anbindung des Trackers an verschiedene Datenquellen soll modular gestaltet werden. Damit ist es möglich den Tracker, ohne zusätzlichen Aufwand, in verschiedenen Szenarien zu testen und ihn somit an verschiedene Datenquellen anzubinden. Abbildung 4.1 zeigt die existierenden Module und deren Interaktion.

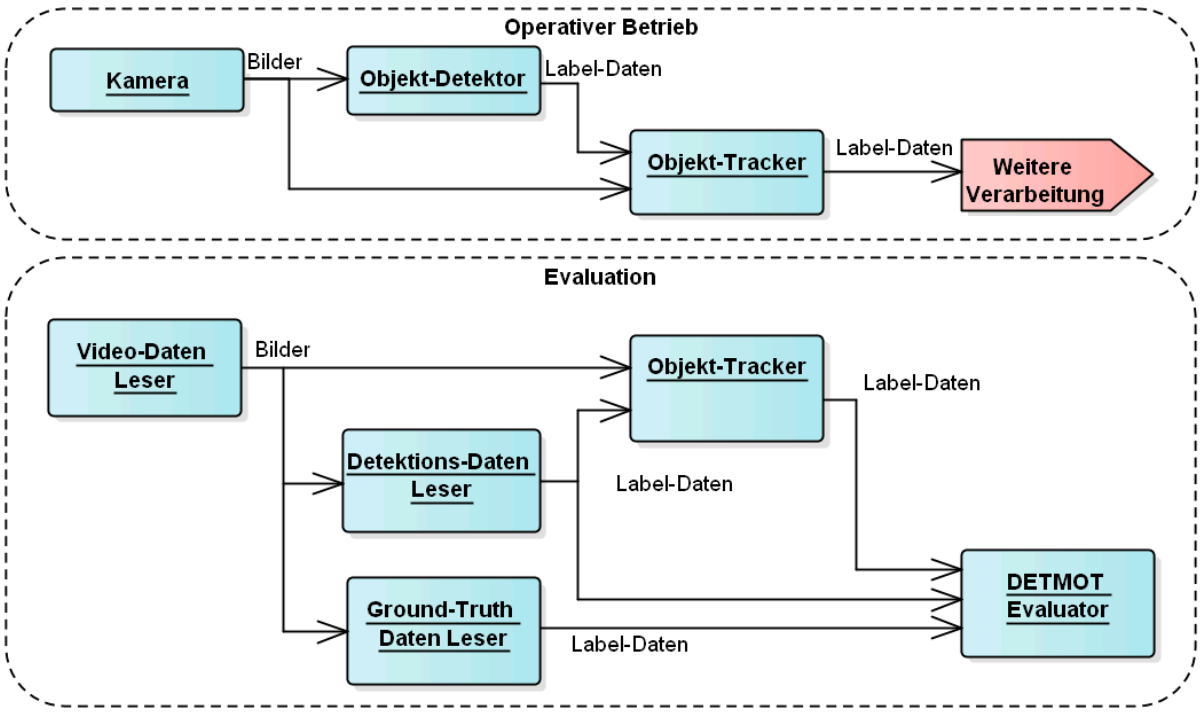

Abbildung 4.1: Diese Grafik stellt die verschiedenen Module und deren Interaktion miteinander dar. Dabei sind zwei Szenarien dargestellt. Im oberen Teil ist der operative Betrieb. Dabei handelt es sich um einen Anwendungsfall wie er auftritt, wenn der Tracker in einem realen Anwendungsfall eingesetzt wird. Die Daten erhält der Tracker von einer Kamera und die Detektionsdaten kommen von einem Objekt-Detektor. Die Resultate des Trackers werden von weiteren Modulen genutzt. Der zweite Anwendungsfall ist die Evaluation. Hierbei werden die Video-Daten und die Detektions-Daten aus vorgefertigten Dateien ausgelesen. Zusätzlich werden Ground-Truth-Daten eingelesen, welche die Trackingergebnisse enthalten die vom Tracker erwartet werden. Die Trackingergebnisse, Detektionsergebnisse und Ground-Truth-Daten werden an den DETMOT-Evaluator weitergegeben, um die Performance des Trackers zu bewerten. Der Fokus dieser Arbeit liegt auf den Modulen „Objekt Tracker“ und „DETMOT-Evaluator“.
Die Pfeile stellen Datenverbindungen dar. Die Art der Daten, die gesendet wird, ist an dem Pfeil notiert. Bilder beschreiben hierbei rohe Bilddaten und Labeldaten stellen Bounding-Boxen mit Klassifikation und (optional) Identifikationsnummern dar.

4.4 Datendarstellung Bounding-Box

Bei der Detektion und beim Tracking wird eine Menge an Objekten erkannt. Jedes Objekt wird dabei durch eine 2D-Bounding-Box definiert. Abbildung 4.2 zeigt eine Objekt mit dessen Bounding-Box.

Abbildung 4.2: Die Darstellung zeigt ein Objekt und die dazugehörige Bounding-Box. Zusätzlich ist die Konfidenz und die Klassifikation zu sehen.

Eine Bounding-Box wird durch 4 Werte definiert:

- Linker Rand
- Rechter Rand
- Oberer Rand
- Unterer Rand

Daraus lassen sich folgende zusätzliche Werte berechnen:

- Breite = Rechter Rand - Linker Rand
- Höhe = Unterer Rand - Oberer Rand

Das Koordinatensystem in dem sich diese Werte befinden ist ein zweidimensionales Koordinatensystem mit invertierter Y-Achse, dies ist in Abbildung 4.3 dargestellt.

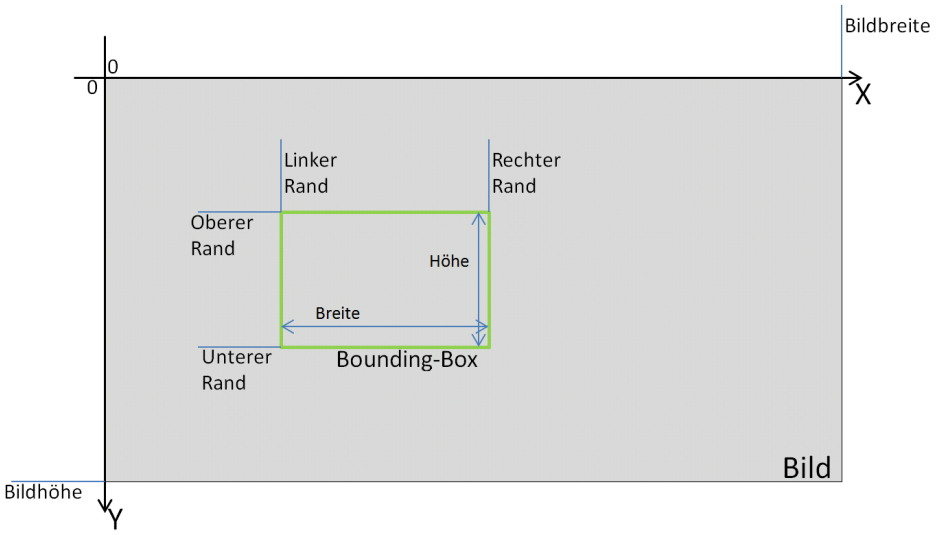

Abbildung 4.3: Diese Abbildung zeigt welche Informationen zu einer Bounding-Box gehören. Die Grenzen der Bounding-Box werden auf der X-Achse durch den linken und rechten Rand definiert. Auf der Y-Achse wird die Bounding-Box durch den oberen und unteren Rand begrenzt.

4.5 Schnittstellen

Wie bereits in der Einleitung (Abschnitt 1.2) erwähnt wird der Objekttracker auf dem Fundament des Nova-Frameworks aufgebaut. Hierfür ist es wichtig, klare Schnittstellen zur Kommunikation zwischen den einzelnen Komponenten zu erstellen. Im Rahmen dieser Arbeit gilt es dabei nicht nur den Objekttracker, sondern auch den dazugehörigen Objektdetektor in das Nova-Framework einzubinden. Des Weiteren wird es nötig sein, Labeldaten von verschiedenen Datensätzen einzulesen, um auf Ground-Truth-Daten zurückgreifen zu können.

Das Nova-Framework verfügt über eine sehr weitführende Interface-Hierarchie. Das Basisinterface, von welchem alle Klassen über verschieden lange Vererbungsketten ableiten ist *IObject*. Ein *IObject* verfügt nur über einige Basisfunktionen, wie das Löschen des Elements oder das Erfragen eines implementierte Interfaces. Jedes Interface bekommt eine sog. Interface-ID zugewiesen, mithilfe welcher es angefragt werden kann. So kann auf diese Art z. B. ein IObject gefragt werden, ob es auch ein IComponent ist und entsprechend behandelt werden. Die Interface-ID ist ein menschenlesbarer String aus Zeichen.

Alle in diesem Abschnitt beschriebenen Interfaces leiten ebenfalls von *IObject* ab. Um die Graphiken jedoch überschaubar zu halten, werden diese Vererbungen nicht mit dargestellt. Für genauere Informationen hierfür kann die Doxygen-Dokumentation des Projekts eingesehen werden, welche auf der diese Arbeit beiliegenden Disk enthalten ist.

4.5.1 Trackingdaten

Die Resultate des Tracking bestehen aus einer Menge von Trackingframes. Ein einzelnes Trackingframe besteht aus einer endlichen Menge detektierter, bzw. getrackter Objekten. Jedes Objekt stellt bestimmte Daten bereit, wie z.B. Boundingbox, Konfidenz oder die Klassifikation. Der einzige Unterschied zwischen Detektion- und Trackingdaten besteht darin, dass beim Tracking jedes Objekt eine ID zur Identifizierung der Spezifischen Instanz hat. Aus diesem Grund werden für das Tracking und die Detektion dieselben Datenstrukturen genutzt. Um eine bessere Kompatibilität zwischen verschiedenen Softwaremodulen zu schaffen, werden diese Daten mit C++-Interfaces codiert.

Die jeweiligen Interfaces sind IDetectionFrame und IDetectionObject, welche in Abbildung 4.4 dargestellt sind. Zusätzlich gibt es das Interface IClassMap, welches dafür zuständig ist, die Klassen-ID in einen String und umgekehrt zu übersetzen. Das Interface IBounds beschreibt die Boundingbox.

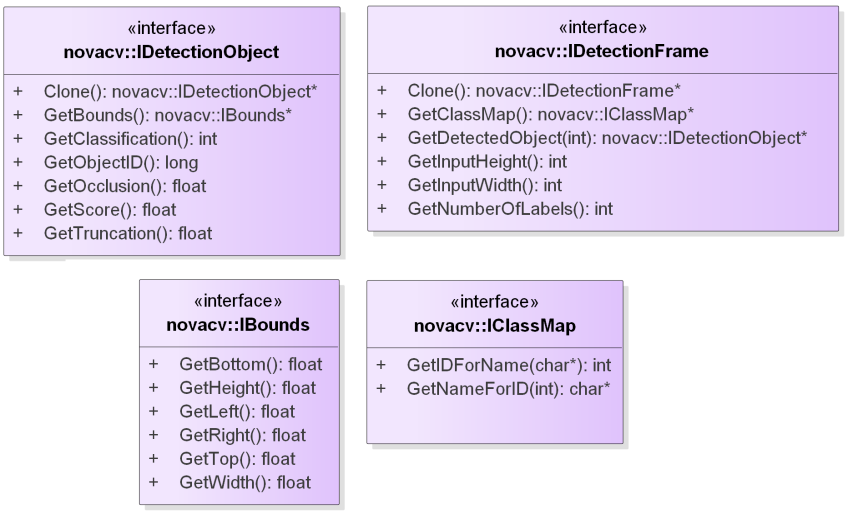

Abbildung 4.4: Das UML-Klassendiagramm zu den Daten-Interfaces. Zu sehen sind die zu Implementierenden Methoden für die Interfaces IDetectionObject, IDetectionFrame, IBounds und IClassMap.

4.5.2 Detektor

Der Objektdetektor soll Objekte in Bildern detektieren und deren Boundingbox markieren, sowie eine Klassifizierung in z. B. Auto oder Fußgänger vornehmen.

Der Lebenszyklus eines Detektors besteht aus der Initialisierung (mit der Methode *Initialize()*), aus der Detektion von Objekten in Eingangsbildern (durch die Methode *DetectImage()*) und dem Schließen des Detektors (mithilfe von *Close()*)). Zur Detektion eines einzelnen Frames ist nur das zugrunde liegende Bild von Nöten, welches als Parameter an *DetectImage(cv::Mat*, novacv::IDetectionFrame**)* übergeben wird.

Detektoren können sich sehr signifikant voneinander unterscheiden. So ist beispielsweise für den Objektdetektor von Herrn Kleinhenz [1] eine Netzbeschreibung für das neuro-

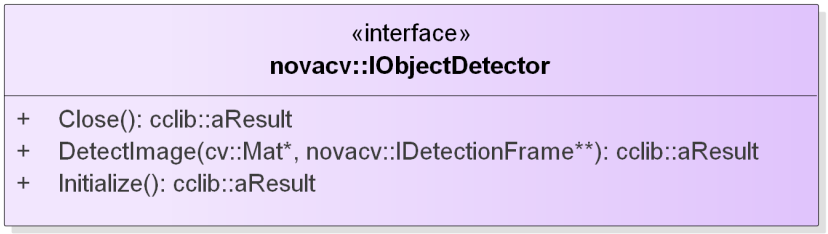

Abbildung 4.5: Das UML-Klassendiagramm zum Objektdetektor-Interface.

nale Netz nötig, sowie ein Modelfile mit den Gewichten anzugeben. Andere Detektoren wie z. B. der SedeniusLabelFile-Detektor, welcher die Detektionsdaten aus einer Datei einliest, benötigt ausschließlich den Dateinamen der Datei, welche die zu ladenden Labeldaten enthält, um zu funktionieren. Diese Daten können Detektionsergebnisse oder auch Ground-Truth-Daten enthalten.

Um diese Usecases abzubilden, erbt *IObjectDetector* vom Interface *IProperties*. *IProperties* bietet die Möglichkeit zur Konfiguration Properties zu setzen. Ein Property ist ein Tupel aus einem, innerhalb des Objekts, eindeutigen Namen und einem Wert, welcher von Folgendem Typ sein kann:

Bool Boolean Wert, True/False

Int32 32-Bit Integer Wert, mit Vorzeichen

UInt32 32-Bit Integer Wert, ohne Vorzeichen

Float32 32-Bit Gleitkommazahl

String Text

Die Werte der Properties werden automatisch mit dem Projekt gespeichert und geladen.

4.5.3 Tracker

Der Tracker ist das zentrale Herzstück dieser Arbeit. Seine Funktionsweise und sein Lebenszyklus unterscheiden sich nur leicht von dem eines Objekt-Detektors. Abb. 4.6 zeigt das UML-Klassendiagram des Tracking Interfaces. Auch der Tracker kann mit Properties

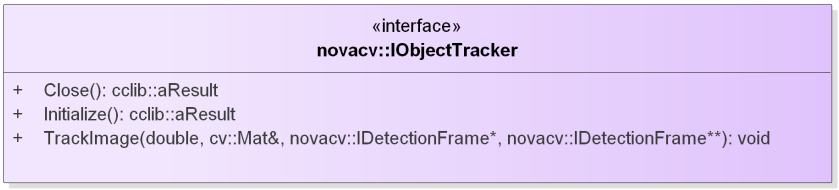

Abbildung 4.6: Das UML-Diagramm zum Objekttracker-Interface.

konfiguriert werden. Der Tracker unterscheidet sich vom Detektor in der Verarbeitungs-

methode, hier *TrackImage*. Als Parameter wird hierbei das zugrunde liegende Bild, sowie die Erbenmisse des Detektors für das angegebene Bild erwartet.

4.6 Lebenszyklus eines Objektes

Das Verfahren, welches im Rahmen dieser Arbeit entwickelt wird, wird den Zustand eines Objekts, im Hinblick auf das Tracking, auf ähnliche Weise modellieren wie das MDP-Verfahren [10].

Abbildung 4.7 zeigt die Zustände und die möglichen Übergänge.

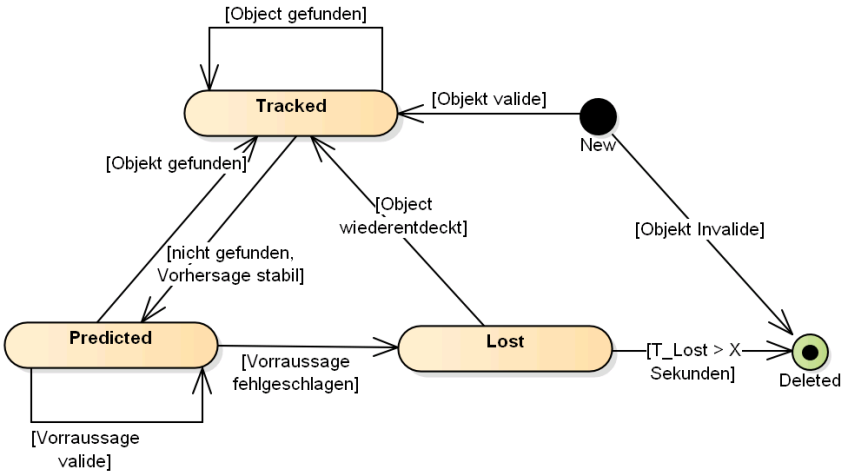

Abbildung 4.7: Zu sehen sind hier die Zustände und die Übergänge, mit denen die Entscheidung getroffen werden soll, wie ein Objekt behandelt wird. Es ist an das MDP-Verfahren angelehnt [10].

New Das Objekt wurde gerade gefunden und es muss noch geprüft werden ob die Boundingbox und die Konfidenz valide sind.

Tracked Objekte in diesem Zustand sind aktuell im Bild zu sehen und werden getrackt. Das bedeutet, dass sie in der Ausgabe des Trackers enthalten sind.

Predicted Das Objekt wurde vom Detektor nicht gefunden, die Position kann jedoch mit einer gewissen Sicherheit vorhergesagt werden.

Lost Das Objekt wurde nicht vom Detektor gefunden und es ist auch nicht möglich die Position vorherzusagen.

Deleted ist ein Objekt von Anfang an invalide oder befindet sich zu lange im Zustand Lost, wird es unwiderruflich gelöscht.

Für die Übergangsfunktionen werden verschiedene Ansätze genutzt. Hierfür werden Untersuchungen durchgeführt, die Ergebnisse sind in Kapitel 5 zu finden.

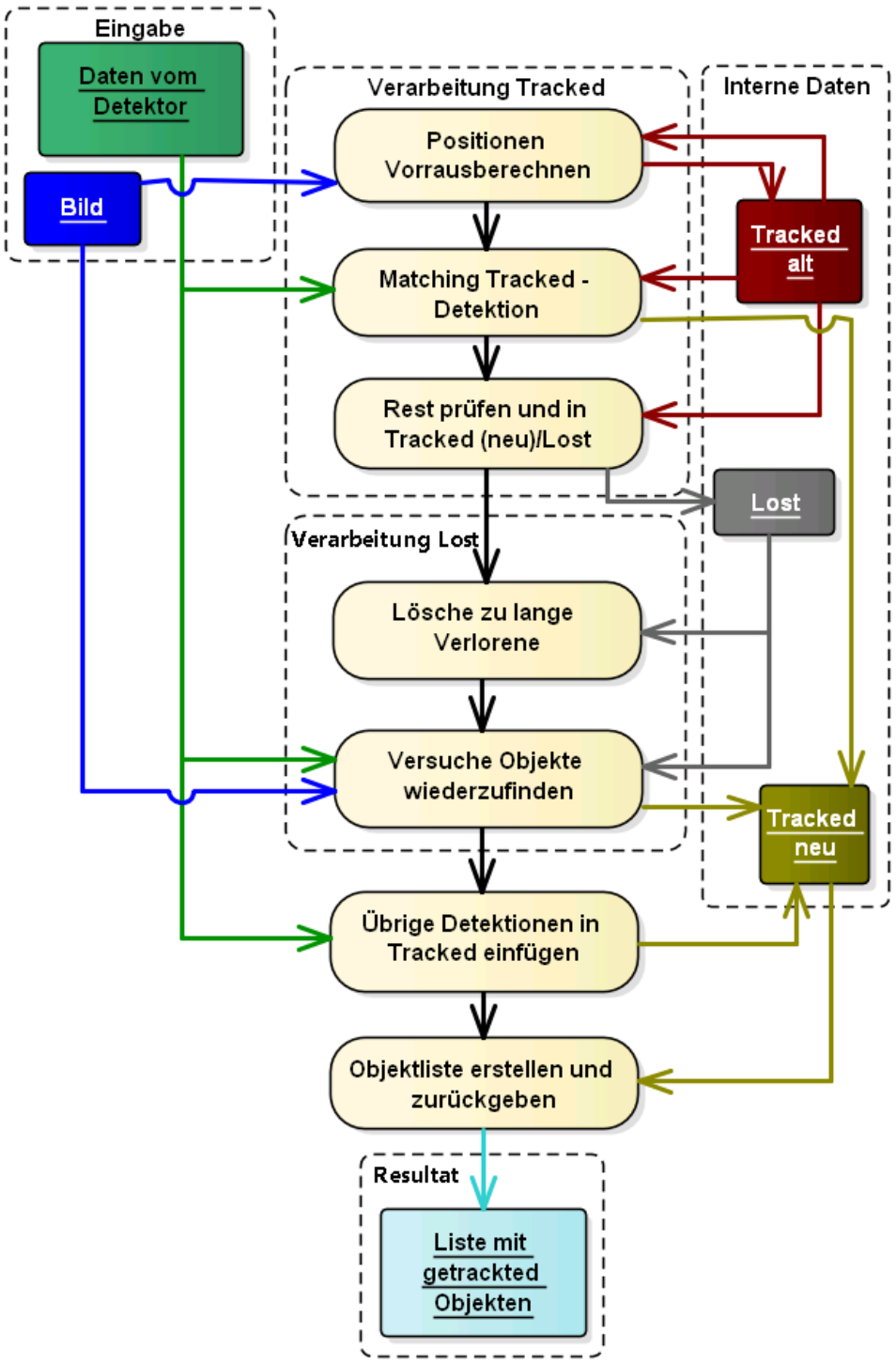

Abbildung 4.8: Dieses Bild zeigt den Workflow des Trackers.

4.7 Workflow pro Frame

Abbildung 4.8 zeigt die Arbeitsschritte, welche der Tracker in jedem Frame durchläuft. Auf der linken Seite in Abbildung 4.8 sind die Eingangsdaten zu sehen. Auf der rechten Seite sind die internen Daten dargestellt und am unteren Rand die Resultate. Die Farbigen

Boxen stellen Datenmengen dar. Diese enthalten Verweise auf Objekte oder andere Daten. *Bild* enthält z. B. die die Bilddaten des aktuellen Bildes. *Tracked alt* enthält die Objekte die zu Beginn der Berechnung im Zustand Tracked waren, wohingegen *Tracked alt* jene Objekte enthält, welche auch nach der Berechnung noch als getracked angesehen werden. *Lost* enthält die Objekte, die aktuell nicht zu sehen sind und *Daten vom Detektor* enthält die Objekte, welche vom Detektor erkannt wurden.

Die Pfeile stellen dar welche Art der Manipulation in dem jeweiligen Schritt vorgenommen wird. Ein eingehender Pfeil zeigt an, dass in diesem Schritt Objekte in die Datenmenge eingefügt werden, während ausgehende Pfeile bedeuten, dass Objekte aus der Menge entfernt oder gelesen werden.

Positionen vorausberechnen

In diesem Schritt werden die Positionen der Bounding-Boxen mithilfe der Bilddaten vorausberechnet. Dabei werden verschiedene Qualitätsmerkmale für die Vorhersage berechnet, auf deren Basis entschieden werden kann, ob die Vorhersage akzeptiert wird oder nicht, sofern kein Detektionsergebnis für das Objekt vorliegt. Mehr hierzu in Abschnitt 5.3.

Matching Tracked-Detektion

Im zweiten Schritt werden bereits getrackte Objekte mit den neusten Detektions-Daten zusammen geführt. Es wird geprüft, wo eine sinnvolle Assoziation möglich ist, also welche Objekte im aktuellen Frame nach wie vor präsent sind. Dabei muss sich grundsätzlich jedes im vorherigen Frame getrackte Objekt neu beweisen. Daher gibt es eine eindeutige Trennung von *Tracked alt* und *Tracked neu*. Objekte können nur mit guter Begründung von *Tracked alt* in *Tracked neu* verschoben werden. Dieser Schritt greift auf die Detektionsdaten zurück.

Alle gefundenen Objekte werden aus *Tracked alt* entfernt und in *Tracked neu* eingefügt.

Rest prüfen und in Tracked(neu)/Lost

Für alle verbleibenden Objekte in *Tracked alt* wird nun geprüft ob ihre Vorhersage stabil genug ist, um sie weiter zu tracken. Ist dies der Fall, wird das Objekt in *Tracked neu* verschoben, wenn nicht kommt es in *Lost*.

Lösche zu lange Verlorene

Alle Objekte die sich länger als eine vorgegebene Zeit in *Lost* befinden, werden gelöscht.

Versuche Objekte wiederzufinden

Für die verbleibenden detektierten Objekte wird geprüft ob sie einem der Objekte in *Lost* weit genug ähneln, dass es sich um dasselbe Objekt handeln könnte. Hierfür werden von beiden Objekten Features berechnet und miteinander verglichen. Sollte die Ähnlichkeit

groß genug sein, wird das Objekt aus *Lost* entfernt und in *Tracked neu* eingefügt. Diese Prüfung wird auch für die Objekte durchgeführt die im vorherigen Teilschritt erst in die *Lost*-Menge eingefügt wurden.

Übrige Detektionen in Tracked einfügen

Da die restlichen verbleibenden detektierten Objekte keinem bestehenden Objekt zugewiesen werden konnten muss es sich entweder um falsch Positive, d. h. einen Fehler von Seiten des Detektors, oder um ein neues Objekt handeln. Im Rahmen dieser Arbeit wird davon ausgegangen, dass die Filterung und Validierung der Detektionsergebisse Aufgabe des Detektors ist. Deshalb werden alle übrigen detektierten Objekte in *Tracked neu* eingefügt.

Objektliste erstellen und zurückgeben

Im letzten Schritt werden alle Objekte in *Tracked neu* in die Resultatliste eingefügt und zurückgegeben.

Die genaue Implementierung dieser Schritte ist im Kapitel 5 zu finden.

4.8 DETMOT Evaluator

Dieser Abschnitt beschreibt die Grundidee hinter dem DETMOT-Evaluator.

Die in Abschnitt 2.3 beschriebenen Evaluationsmethoden sind unter gewissen Vorbehalten dazu geeignet Trackingalgorithmen miteinander zu vergleichen.

Sie sind jedoch nicht in der Lage, ein absolutes Qualitätsmaß zu geben, welches unabhängig vom Detektor ist, da der Trackingalgorithmus immer zusammen mit seinem Detektor bewertet wird, wodurch die Ergebnisse des Trackers verfälscht werden können. Dieser Umstand wird bei der MOT-Challenge dadurch ausgeglichen, dass die Detektionsergebnisse vorgegeben werden. Beim KITTI-Benchmark wird nicht darauf eingegangen, welche Detektionsergebnisse von den Algorithmen genutzt werden sollen, was bei der MOT-Challenge explizit vermerkt wird. Des Weiteren ist es nicht möglich feststellen, wo die Stärken und Schwächen eines einzelnen Trackers liegen.

Aus diesem Grund wird ein weiterer Evaluator entworfen, welcher dazu dienen soll den Algorithmus detailliert auszuwerten, um so Schwächen in bestimmten Bereichen feststellen zu können und ein besseres Bewertungsmaß, auch im Bezug auf einzelne Teilaufgaben des Trackers, bereitzustellen. Hierzu werden zur Evaluation nicht nur die Trackingergebnisse, sondern auch die zu Grunde liegenden Detektionsergebnisse zurate gezogen.

Beim Evaluieren wird geprüft ob ein Ground-Truth-Element eine korrespondierende Detektion hat. Ab diesem Frame wird das Objekt in allen folgenden Frames, in denen es in den Ground-Truth-Daten enthalten ist als aktiv markiert. Nicht aktivierte Ground-Truth Elemente werden nicht in die Evaluation einbezogen, da der Tracker keine Objekte tracken kann, die nicht bereits detektiert wurden. Ist ein Objekt einmal aktiviert wird vom Tracker erwartet, dass dieser es weiterhin tracken kann. Das bedeutet, selbst wenn das

Objekt nicht mehr vom Detektor erfasst wird, bleibt das Objekt aktiviert. Verschwindet ein Objekt in den Ground-Truth-Daten, wird das Objekt wieder auf inaktiv geschaltet, bis es wieder von der Ground-Truth und dem Detektor erfasst wird. Auch dies basiert auf der Annahme, dass der Tracker keine Objekte tracken kann, welche nicht detektiert wurden.

Diese Änderung wird auf die beiden bereits genannten Evaluatoren angewandt. Zusätzlich wird für bestimmte Fehler, wie ID-Switches, noch eine Klassifikation durchgeführt, wodurch dieser verursacht wurde. Es werden Kategorien festgelegt um Fehlerursachen besser feststellen zu können (Siehe Abschnitt 5.4).

Eigenschaften wie die Vorhersage von Objekte, das heißt diese weiterhin zu tracken auch wenn der Detektor sie nicht sieht, werden detaillierter ausgewertet und es wird geprüft, wie gut ein Tracker diese Objekte weiter trackt. Die Notwendigkeit der Vorhersage ist sehr abhängig von der Qualität des Detektor.

Mithilfe dieser Methode werden für jeden Wert aus der CLEAR MOT, sowie MT/PT/ML Statistik mehrere korrigierte Versionen generiert, welche verschiedene Detektionsfehler ausgleichen. Die entstehenden Werte stellen ein besseres Bewertungsmaß dar und bieten zusätzlich eine Fehlerursachenanalyse. Es lässt sich detaillierter feststellen in welchen Teilgebieten ein bestimmter Objekttracker gut oder schlecht ist.

In Abschnitt 5.4 wird beschrieben welche Werte berechnet werden und was diese aussagen.

5 Realisierung

Dieses Kapitel beschreibt die konkrete Funktionsweise des Objekttrackers, sowie des DETMOT Evaluators.

5.1 Ablauf

Ein einzelner Schritt des Trackingverfahrens umfasst die Verarbeitung eines Bildes zusammen mit den dazugehörigen Detektionsergebnissen.

In jedem Schritt werden zuerst die Positionen der bereits bekannten Objekte, vorausberechnet. Objekte, welche als Verloren (*Lost*) gelten werden von dieser Vorausberechnung ausgenommen und werden nicht vorausberechnet.

Nach dem Vorbrauserechnen der Objekte folgt der Assoziationsprozess zwischen bekannten Objekten und Detektionsergebnissen. Hierbei werden ausschließlich *getrackte* Objekte berücksichtigt, das bedeutet verlorene Objekte werden von der Assoziation ausgenommen.

Für alle getrackten Objekten, welchen keine Detektion zugewiesen werden konnte, wird im nächsten Abschnitt geprüft, ob die Vorhersage, welche im ersten Teil der Verarbeitung durchgeführt wurde, ausreichend gut ist, um das Objekt weiterhin zu verfolgen. Ist eine ausreichende Stabilität der Vorhersage nicht gegeben, wird das Objekt als *Lost* eingestuft. Ist die Vorhersage jedoch stabil genug, verbleibt das Objekt in *Tracked*.

Im Weiteren wird eine Verarbeitung der *Lost*-Objekte durchgeführt. Der erste Schritt hierbei ist, alle Objekte in *Lost* zu löschen, welche für einen zu langen Zeitraum nicht erfasst wurden. Dieser Zeitraum kann per Parameter eingestellt werden. Danach wird für jedes übrige Objekt in *Lost* geprüft, ob es unter den detektierten Objekten eines gibt, welches eine hohe Ähnlichkeit zu dem gewählten *Lost*-Objekt aufweist. Ist die Ähnlichkeit groß genug, wird angenommen, dass es sich um dasselbe Objekt handelt und das Objekt wird wieder in *Tracked* überführt.

Im letzten Abschnitt der Verarbeitung eines Bildes wird jedes detektierte Objekt, welches noch nicht mit einem bestehenden Objekt assoziiert wurde, als neues Objekt im Zustand *Tracked* eingefügt.

Zuletzt wird eine Liste mit allen Objekten, welche sich im Zustand *Tracked* befinden, zusammengestellt. Diese Liste stellt das Trackingergebnis des aktuellen Zeitschritts dar und wird als Resultat zurückgegeben. Eine schematische Darstellung dieses Ablaufes ist in Abbildung 4.8 zu sehen.

5.2 Merkmalsauswahl

Der Tracker arbeitet nach den Paradigma *„tracking-by-detektion“*. Das bedeutet, dass zum Tracking Bild und Detektionsdaten genutzt werden. Das Tracking selbst wird damit

zu einem Assoziationsproblem, bei dem es darum geht, passende Assoziationen zwischen bekannten Objekten und neuen Detektionen zu schaffen.

Hierfür sind Merkmale notwendig, anhand derer ein Ähnlichkeitsgrad zwischen zwei Objekten festgestellt werden kann. Es gibt verschiedene Anwendungsfälle, welche unterschiedliche Merkmale benötigen. Folgende Anwendungsfälle müssen bedacht werden:

Zwischen Frames detektierten Objekten mit einem Objekt aus dem vorherigen Bild. Mehr hierzu in Kapitel 5.2.1.

Nach Trackingverlust Detektionsergebnisse mit verlorenen Objekten. Mehr hierzu in Kapitel 5.2.2.

Jeder Anwendungsfall hat bestimmte Eigenschaften, die zur Assoziation ausgenutzt werden können. Für den Anwendungsfall *Zwischen Frames* kann z. B. vorausgesetzt werden, dass der Zeitabstand zwischen zwei Objekten genau ein Zeitschritt ist. Auf dieser Basis kann die Annahme vorausgesetzt werden, dass sich Objekte in einem Zeitschritt weder weit bewegen, noch große Änderungen zum vorherigen Auftreten aufweisen.

5.2.1 Verarbeitung bereits bekannter Objekte

Wie bereits in Abschnitt 5.1 erwähnt, wird die Position eines Objekts, bevor es mit detektieren Objekten verglichen wird vorausberechnet. Der Vergleich von *Tracked*-Objekten wird anhand geometrischer Merkmale durchgeführt. Es wird geprüft, ob das Objekt sich an der Position befindet, wo es sich laut dem optischen Fluss befinden sollte. Dieser Vergleich ist besonders performant, da ausschließlich die Bounding-Boxen verglichen werden.

In diesem Vergleich wird nur das Überlappungsverhältnis betrachtet.

Überlappungsverhältnis

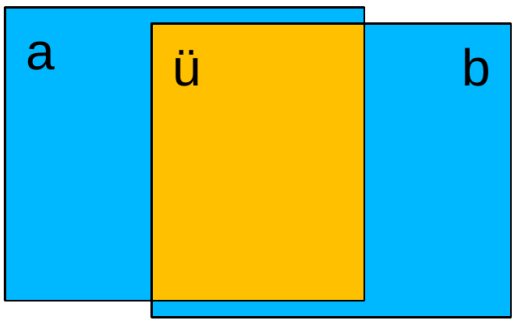

Abbildung 5.1: Die Abbildung zeigt die Schnittfläche $\ddot{u}$ der beiden Bounding-Boxen a und b. Daraus lässt sich mit den Formeln 5.1, 5.2 und 5.3 das Überlappungsverhältnis berechnen.

Abbildung 5.1 illustriert die Berechnung des Überlappungsverhältnisses (Üv). Es sei A_a die Fläche der Bounding-Box a und A_b die Fläche der Bounding-Box b. $A_{\text{ü}}$ sei die Schnittfläche beider Bounding-Boxen (in Abbildung 5.1 ü).

$$A_{all} = A_a \cup A_b \tag{5.1}$$

$$A_{\text{ü}} = A_a \cap A_b \tag{5.2}$$

$$\text{Ü}v = \frac{A_{\text{ü}}}{A_{all}} \tag{5.3}$$

Sofern das Üv den Schwellwert 0,3 überschreitet, wird von einer passenden Assoziation ausgegangen. Der Wert 0,3 ergab sich aus Tests, welche im Abschnitt 5.5.1 beschrieben werden.

Im Verlauf dieser Phase wird für jedes mögliche Tupel, bestehend aus einem Objekt in *Tracked* und einem Objekt aus den Detektionen des aktuellen Zeitschritts, geprüft wie groß das Überlappungsverhältnis ist. Ist der oben genannte Schwellwert überschritten, wird das Tupel zusammen mit dem Überlappungsverhältnis beider Objekte abgespeichert. Dabei entsteht eine Assoziationsliste mit allen Objekten die miteinander assoziiert werden könnten. Diese Assoziationsliste wird nach dem Überlappungsverhältnis absteigend sortiert und es wird nacheinander immer die Assoziation mit dem größten Überlappungsverhältnis aus der Liste entfernt und das dazugehörige Objekt in den Zustand *Tracked* überführt. Außerdem wird die Position des *Tracked*-Objekts mit der Position des Detektionsergebnisses aktualisiert. Um doppelte Assoziationen zu vermeiden wird jede weitere Assoziation, welche entweder ein anderes *Tracked*-Objekt mit dem selben Detektionsergebnis oder ein anderes Detektionsergebnis mit dem selben *Tracked* Objekt wie in dem gewählten Tupel assoziiert, aus der Assoziationsliste entfernt. Außerdem wird das *Tracked*-Objekt welches zu dem Tupel gehört aus der alten *Tracked*-Liste entfernt

Dieser Vorgang wird wiederholt bis die Assoziationsliste leer ist.

Die Grundidee hinter diesem Ähnlichkeitsfaktor ist die Annahme, dass sich Objekte nach einem Zeitschritt meist nicht weit von Ihrer vorherigen Position entfernt befinden.

Sollte dies doch der Fall sein, wird das Objekt in die *Lost*-Liste verschoben. Dort wird es unter Zuhilfenahme anderer Merkmale mit detektierten Objekten verglichen (siehe Kapitel 5.2.2) und es wird geprüft, ob das Objekt wieder gefunden werden kann.

5.2.2 Verarbeitung verlorener (*Lost*) Objekte

Um geeignete Merkmale zur Wiedererkennung verlorener Objekte zu finden werden im folgenden Kapitel verschiedene Merkmale vorgestellt. Diese werden statistisch auf der Grundlage einer Datensequenz des Kitti-Datensatzes darauf geprüft, ob sie geeignet sind, Lost-Objekte korrekt mit Detektionsergebnissen zu assoziieren.

Als Testdaten wird ein Differenzvektor zwischen den Merkmalen berechnet, so dass für jedes Merkmal ein oder mehrere Abstandsindikatoren entstehen. Um Abstandswerte für

richtige Zuordnungen zu erhalten wird jedes Objekt mit den Bounding-Boxen seines vorherigen Auftretens verglichen. Um Abstandswerte für falsche Zuordnungen zu erhalten wird jedes Objekt mit den Bounding-Boxen anderer Objekte verglichen. Für jedes Vergleichspaar wird bestimmt wie groß die Differenz der Zeitstempel beider Bounding-Boxen ist. Für diese zeitliche Differenz kann ein Minimal- und Maximalwert festgelegt werden. Die zugrunde liegenden Bounding-Boxen stammen aus den Ground-Truth-Daten.

Für jedes dieser Merkmale wird der Durchschnitt, sowie die positive und negative Standardabweichung berechnet. Dabei werden verschiedene Statistiken für richtige und für falsche Zuordnungen angelegt. Mithilfe dieser Statistiken kann festgestellt werden ob ein Merkmal für die Zuordnung von Objekten geeignet ist. Aus den Mittelwerten und den Standardabweichungen lässt sich für die richtigen und die falschen Zuordnungen jeweils ein Wertebereich bestimmen. Je kleiner die Schnittmenge beider Wertebereiche, desto besser ist das Merkmal zur Unterscheidung geeignet.

Merkmale

- Euklidischer Abstand / Differenz der Zeitstempel
- Verhältnis der Höhen der Bounding-Boxen (zur Definition der Höhe siehe Abschnitt 4.4)
- HSV-Histogramm mit 8 diskreten Werten je Kanal
- SIFT-Deskriptor
- SURF-Deskriptor

Für jedes dieser Merkmale werden sechs verschiedene Testreihen durchgeführt. Hierfür wird jeder Testdurchlauf zweimal klassifiziert. Die erste Klassifizierung besteht darin ob im Test das Tracking von Autos oder das Tracking von Fußgängern getestet wird. Die zweite Klassifizierung besteht darin, die Testreihen zeitlich zu unterteilen. Dafür wird für jedes Vergleichspaar (welches aus zwei Bounding-Boxen besteht) die Differenz zwischen den Zeitstempeln der beiden Bounding-Boxen bestimmt. Abhängig von dieser Zeitdifferenz wird das Vergleichspaar einer der folgenden Gruppen zugeteilt:

<1 Sekunde Die Zeitdifferenz beträgt weniger als eine Sekunde

1-5 Sekunden Die Zeitdifferenz beträgt zwischen 1 und 5 Sekunden

>5 Sekunden Die Zeitdifferenz beträgt mindestens 5 Sekunden

Jede Testreihe wird separat nach Autos und Fußgängern und zusätzlich nach den beschriebenen Zeitdifferenzen eingeteilt. Daraus ergeben sich 6 Testreihen pro Merkmal.

Der Grund für diese Unterscheidung ist, dass geprüft werden soll ob ein Merkmal unterschiedlich geeignet ist, je nach Klassifizierung des Objekts. Die zeitliche Einteilung basiert auf der Annahme, dass die Merkmale mit zunehmender Zeitdifferenz, auch bei gleichen Objekten, größere Unterschiede bei den Merkmalen hervorrufen werden. Dies liegt daran, dass ein Objekt bei größeren Zeitdifferenzen stärkeren Transformationen unterliegt.

Um die Qualität eines Merkmals zu prüfen wird für jedes Merkmal ein Differenzvektor gebildet. Dieser Differenzvektor gibt die Distanz beider Merkmalsvektoren zueinander an. Die Datenmenge wird dabei in richtige und falsche Zuordnungen unterteilt. Richtige Zuordnungen sind solche, die laut Ground-Truth garantiert dasselbe Objekt darstellen und falsche Zuordnungen die laut Ground-Truth verschiedene Objekte darstellen.

Für jedes Merkmal wird für jede der oben beschriebenen Testreihen eine Liste mit Differenzwerten zwischen zwei Merkmalsvektoren für richtige und für falsche Zuordnung erstellt. Für jede dieser Listen werden folgende Werte berechnet:

- Mittelwert
- positive Standardabweichung
- negative Standardabweichung

Aus dem Mittelwert, der positiven und der negativen Standardabweichung entsteht ein Wertebereich. Dieser Wertebereich wird für richtige und für falsche Zuordnungen berechnet. Je geringer die Schnittmenge der Wertebereiche für richtige und falsche Zuordnungen, desto besser ist das Merkmal dafür geeignet, zwei Objekte einander zuzuordnen. Ist keine Schnittmenge der beiden Bereiche vorhanden ist dies ein sehr positives Argument für den Einsatz des Merkmals. Je größer in diesem Fall die Differenz zwischen den Wertebereichen, desto besser ist das Merkmal geeignet. Abbildung 5.2 zeigt dies an einem Beispiel.

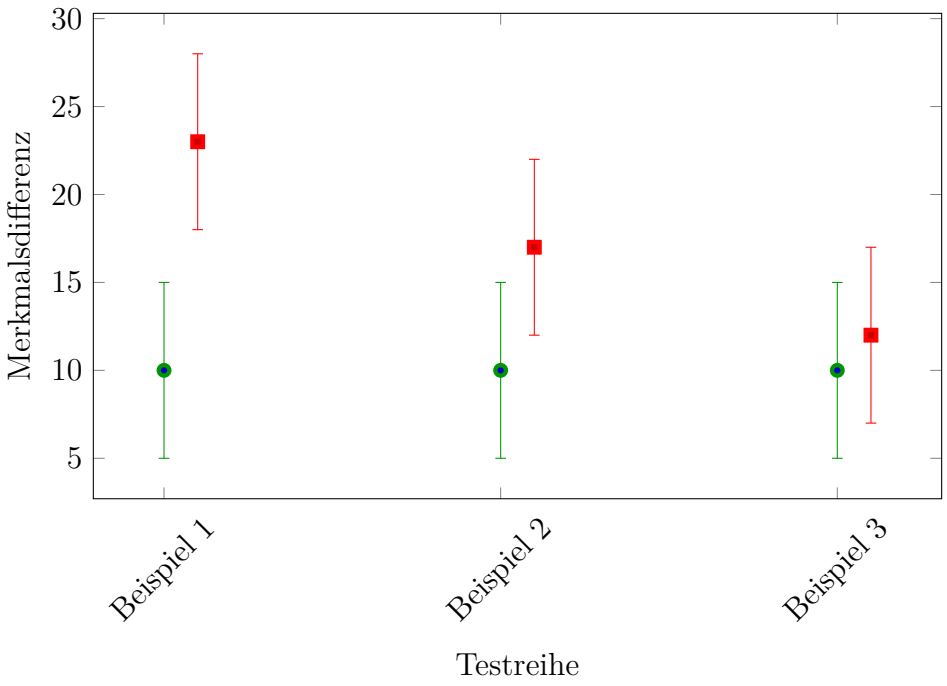

Abbildung 5.2: Statistische Analyse: Beispielstatistik. Grün: richtige Zuordnungen, Rot: falsche Zuordnungen. Diese Grafik stellt fiktive Resultate dar, um die Bedeutung bestimmter Statistiken zu erklären. Die Merkmalsdifferenz hat daher keine Einheit. In Beispiel 1 ist zu sehen, dass es zwischen den Wertebereichen der richtigen und der falschen Zuordnungen keine Schnittmenge gibt. Hier ist eine Unterscheidung sehr gut möglich. Beispiel 2 zeigt einen Fall, indem sich beide Wertebereiche zwar überschneiden, diese Schnittmenge ist jedoch gering und das Merkmal, welches der statistischen Analyse in Beispiel 2 zugrunde liegt, ist zur Unterscheidung geeignet. In Beispiel 3 sind sich beide Wertebereiche nahezu identisch. Es gibt keine statistische Tendenz des Merkmals zu bestimmten Werten auf der Grundlage von richtigen oder falschen Zuordnungen. Dieses Merkmal wäre zur Unterscheidung nicht geeignet.

Die beschriebenen Werte werden in einem Diagramm dargestellt (Abbildung 5.2 stellt dies Beispielhaft dar). Darin werden die statistischen Werte für richtige Zuordnungen durch einen grünen Balken und die falschen Zuordnungen durch einen roten Balken dargestellt. Der Mittelwert wird als grüner Punkt (richtige Zuordnung) oder rotes Quadrat (falsche Zuordnung) dargestellt. Die Standardabweichung wird durch Balken nach oben (positive Standardabweichung) und nach unten (negative Standardabweichung) dargestellt. Die jeweilig zusammen gehörenden richtigen und falschen Zuordnungen werden in Balken direkt nebeneinander dargestellt, so dass eine Überschneidung beider Bereiche optisch erkennbar wird.

Euklidischer Abstand / Zeitabstand (A_t)

Dieser Wert stellt die Geschwindigkeit dar, mit der sich das Objekt bewegen müsste um von seiner ursprünglichen Position an die neue Position, im Bezugssystem des Bildes, zu gelangen in Pixel pro Sekunde. Unter der Annahme, dass sich ein Objekt zwischen zwei

Bildern nur begrenzt weit bewegen kann, kann diesem Wert eine obere Schranke zugewiesen werden, welche nicht überschritten werden darf, da sonst keine plausible Assoziation vorliegen kann.

Die konkrete Berechnungsformel für A_t lautet:

$$A_t = \frac{\text{Distanz}}{\Delta t * X + 1.0s} \tag{5.4}$$

Distanz ist der euklidische Abstand zwischen den Mittelpunkten beider Bounding-Boxen. Δt ist die Differenz der Zeitstempel beider Bounding-Boxen. Hierbei wird immer der neuere Zeitstempel minus den Älteren gerechnet, damit Δt nicht negativ wird. X stellt einen variablen Faktor dar, welcher dazu dient den Verfall der Gesamtdifferenz über die Zeit zu verstärken oder abzuschwächen. Da eine Division durchgeführt wird, kann es passieren, dass der Gesamtwert bei geringer Zeitdifferenz gegen 0 konvergiert. An diesem Punkt ist eine Unterscheidung nicht mehr sinnvoll durchführbar. Aus diesem Grund wird zum Zeitabstand 1 Sekunde addiert. Durch diese Maßnahme wird auch vermieden, dass es zu einer Division durch 0 kommt.

Je länger ein Objekt nicht sichtbar ist, desto weiter kann es sich von seiner ursprünglichen Position entfernt haben. Durch das Verwenden der Zeitdifferenz wird bewirkt, dass das beschriebene Abstandsmaß diesen Sachverhalt mit abbilden kann. Die Multiplikation mit dem konstanten Wert X stellt das erwartete Verhältnis zwischen dem euklidischen Abstand und der Zeitdifferenz dar. Damit wird bestimmt wie viel Verschiebung in Pixel pro Sekunde durch das Einbinden des Zeitabstandes kompensiert werden können. Dieser Wert ist abhängig vom Anwendungskontext und von der Bildauflösung. Je größer X desto schneller verfällt der Abstand mit steigender Zeitdifferenz.

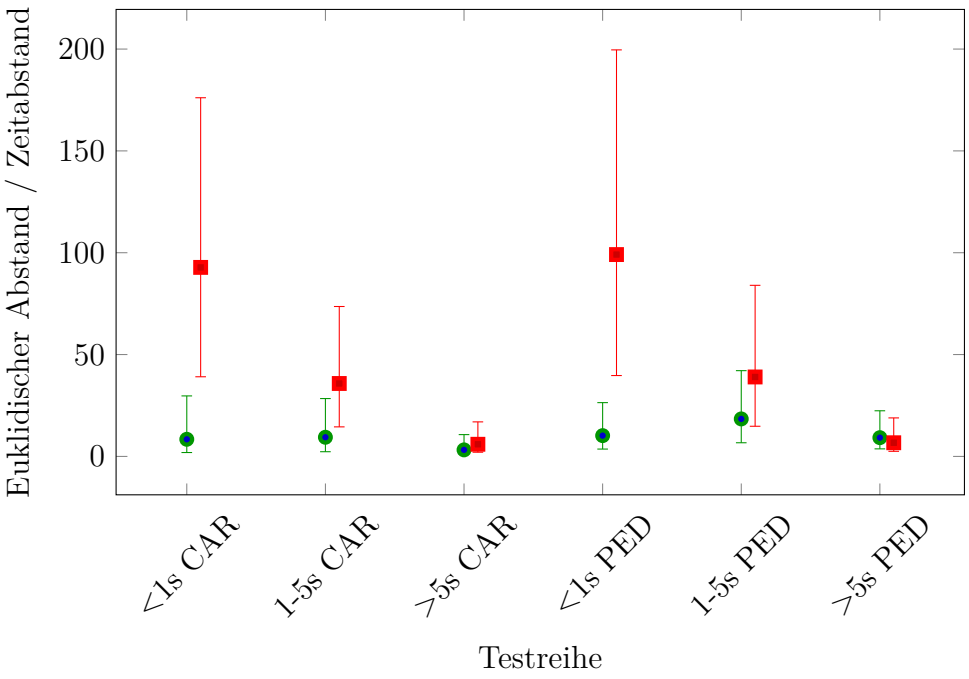

Abbildung 5.3: Statistische Analyse: Euklidischer Abstand / Zeitabstand. Grün: richtige Zuordnungen, Rot: falsche Zuordnungen

Je kleiner der Wert (A_t) ist, desto ähnlicher sind sich die zwei verglichenen Objekte. Dies zeigt sich in der Statistik in Abbildung 5.3. Die Wertebereiche der Differenzen für richtige Zuordnungen liegen sehr niedrig, wohingegen der Wertebereich der falschen Zuordnungen sehr hoch liegen. Dies gilt jedoch nur für kleine Δt (kleiner als 1 Sekunde).

Abbildung 5.3 zeigt die statistische Analyse des Features *Euklidischer Abstand / Zeitabstand*. Zu sehen ist, dass sowohl bei Fußgängern (PED) als auch bei Autos (CAR) die Trends sehr ähnlich sind. In den ersten Sekunden nach dem Verschwinden des Objekts gehen die Differenzen für die richtigen Zuweisungen und die für die Falschen sehr weit auseinander. Die Bereiche, welche durch die Standardabweichung markiert werden, berühren sich nicht. Damit ist das Merkmal in dieser Zeit sehr gut zur Unterscheidung geeignet.

In dem Bereich zwischen 1 und 5 Sekunden kommen sich beide Zahlenbereiche jedoch bereits merklich näher und es gibt eine große Überschneidung zwischen richtigen und falschen Zuweisungen. Ab 5 Sekunden sind sich beide Wertebereiche nahezu identisch und es ist kaum noch ein Unterschied auszumachen.

Daraus folgt, dass dieses Merkmal in den ersten Sekunden nach dem Verschwinden sehr nützlich ist, nach 2 - 3 Sekunden jedoch zur Unterscheidung nicht mehr geeignet ist. Der Vorteil an diesem Merkmal ist, dass es ungeachtet von der Bildauflösung immer konstanten Berechnungsaufwand hat.

Höhenverhältnis zwischen Bounding-Boxen

Dieses Merkmal setzt die Höhen von zwei Bounding-Boxen ins Verhältnis zueinander.

Dafür wird die Höhe beider Bounding-Boxen bestimmt. Daraufhin wird von diesen Werten der Größere durch den Kleineren geteilt um zu gewährleisten, dass das Resultat immer zwischen 0 und 1 liegt. Die Berechnung läuft wie folgt:

$$H_1 = \min\left(H_{\text{BBox},1}, H_{\text{BBox},2}\right) \tag{5.5}$$

$$H_2 = \max\left(H_{\text{BBox},1}, H_{\text{BBox},2}\right) \tag{5.6}$$

$$H_{\text{Ratio}} = \frac{H_1}{H_2} \tag{5.7}$$

Wobei $H_{\text{BBox},i}$ die Höhe der Boundingbox i ist.

In diesen Wert wird, wie bereits beim Merkmal *Euklidischer Abstand / Zeitabstand*, die Zeitdifferenz zwischen beiden Bounding-Boxen mit eingerechnet. Das Ziel hier ist es, dass der Merkmalswert mit steigender Zeitdifferenz gegen 1 konvergiert, da es auch hier so sein kann, dass sich die Höhe zweier Objekte größere Unterschiede aufweisen kann, wenn diese zeitlich weiter auseinander liegen.

Der finale Wert wird folgendermaßen gebildet:

$$H_{\text{final}} = 1 - \frac{1 - H_{\text{Ratio}}}{\Delta t * X + 1.0s} \tag{5.8}$$

Das nicht normierte Höhenverhältnis wird invertiert. Im Nenner wird das invertierte Höhenverhältnis durch den Zeitfaktor geteilt. Dieser errechnet sich indem die Zeitdifferenz zwischen zwei Objekten mit einem konstanten Wert X multipliziert und das Resultat mit 1 Sekunde addiert wird. Das Resultat dieser Division wird noch einmal invertiert. Das Resultat ist das Höhenverhältnis welches bei steigender Zeit gegen 1 konvergiert.

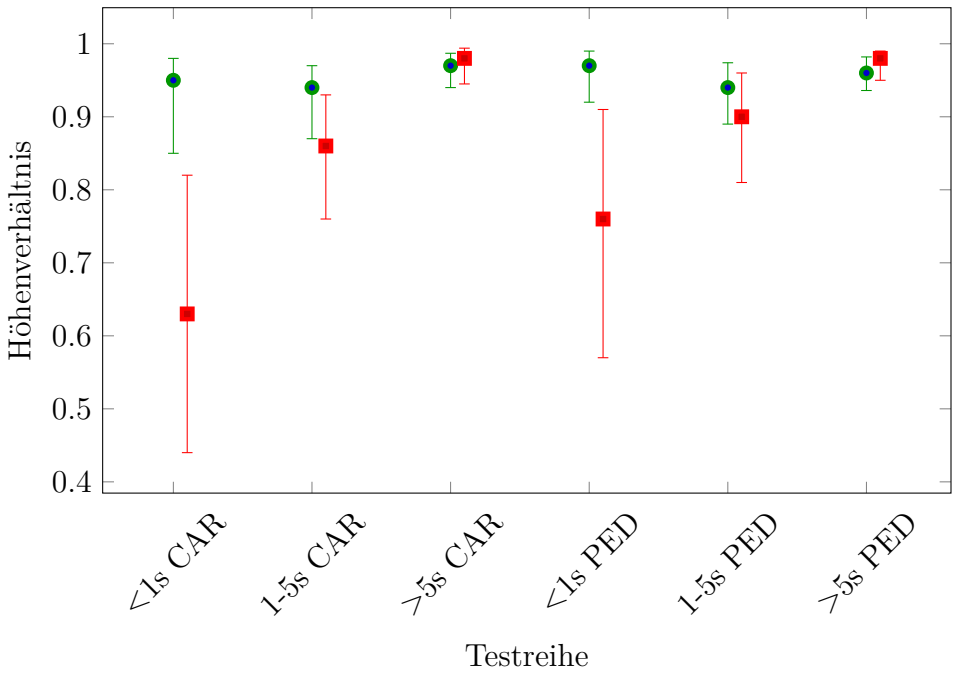

Abbildung 5.4: Statistische Analyse: Höhenverhältnis in Abhängigkeit zur Zeit. Grün: richtige Zuordnungen, Rot: falsche Zuordnungen

Abbildung 5.4 zeigt die statistische Auswertung des Merkmals.

Wie bereits beim Merkmal *Euklidischer Abstand / Zeitabstand* (siehe Abbildung 5.3) ist unter einer Sekunde noch eine gute Unterscheidung möglich. Bei 1-5 Sekunden besteht schon eine große Überschneidung zwischen den Wertebereichen und ab 5 Sekunden sind sich beide Bereiche nahezu identisch. In den ersten Sekunden ist das Merkmal daher gut nutzbar. Ab 2-3 Sekunden jedoch sind auch hier die Differenzwerte nicht mehr zur Unterscheidung von richten und falschen Zuordnungen geeignet. Auch dieses Merkmal benötigt sehr wenig Rechenaufwand um berechnet zu werden.

HSV-Histogramm mit 8 diskreten Farbwerte je Kanal

Zur Berechnung des HSV-Histogramms wird zuerst der Bildausschnitt, welchen die Bounding-Box markiert, ausgeschnitten und von BGR (Blue-Green-Red) in das HSV (Hue-Saturation-Value)-Format umgewandelt. Danach wird für diesen Bildausschnitt ein Histogramm berechnet. Jeder Kanal wird dabei in 8 diskrete Werte unterteilt.

Alle Werte werden normiert, d. h. jeder Wert wird durch die Gesamtanzahl an Pixel geteilt. Damit wird das Histogramm zu einer Verteilungsfunktion und die Summe aller Werte ergibt 1.

Beim Vergleichen von zwei Histogrammen wird die Differenz von jedem der acht Werte in einem Histogramm zu seinem korrespondierenden Wert im anderen verglichen. Dabei wird eine Differenz gebildet. Die Gesamtdifferenz ist die Summe aller acht Einzeldifferenzen. Dieser Wert liegt zwischen 0 (beide Histogramme sind gleich) und 2 (maximale Differenz).

Diese Gesamtdifferenz wird separat für den Hue-, Saturation- und den Value-Kanal berechnet. Je kleiner dieser Wert ist, desto ähnlicher sind sich beide Objekte.

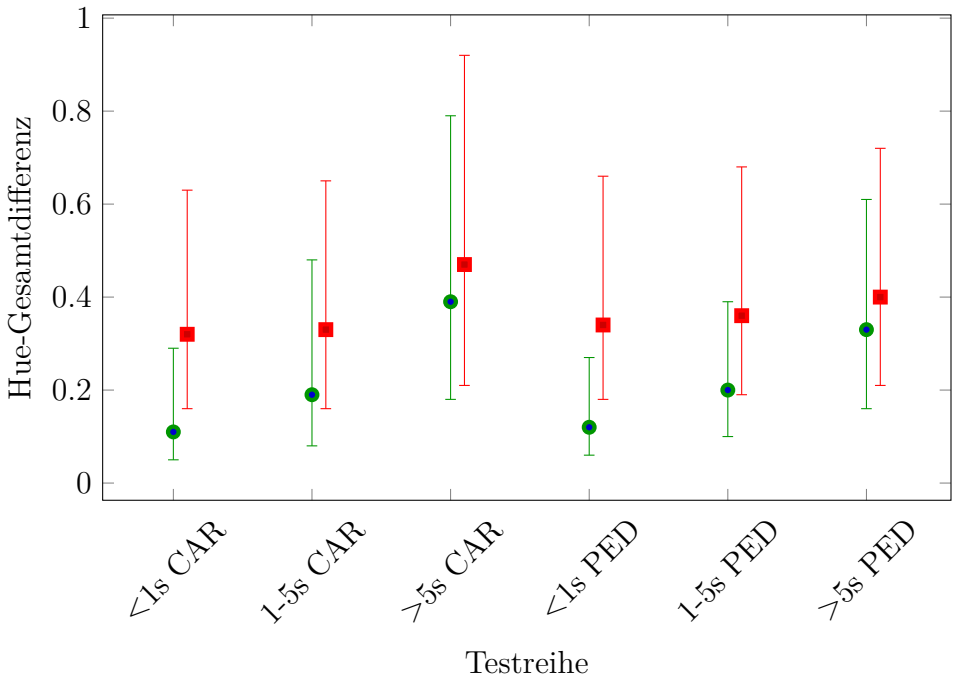

Abbildung 5.5: Statistische Analyse: Hue-Gesamtdifferenz. Grün: richtige Zuordnungen, Rot: falsche Zuordnungen

Abbildung 5.5 zeigt die statistische Auswertung der Differenz im Hue-Kanal. In den Testreihen unter 1 Sekunde ist eine große Überschneidung gegeben, welche im Bereich zwischen 1 und 5 Sekunden größer wird. Ab 5 Sekunden sind sich beide Wertebereiche nahezu identisch. Darauf folgt, dass auch dieses Merkmal nur begrenzt zur Unterscheidung bei Objekten, die weniger als 5 Sekunden verschwunden waren, geeignet ist.

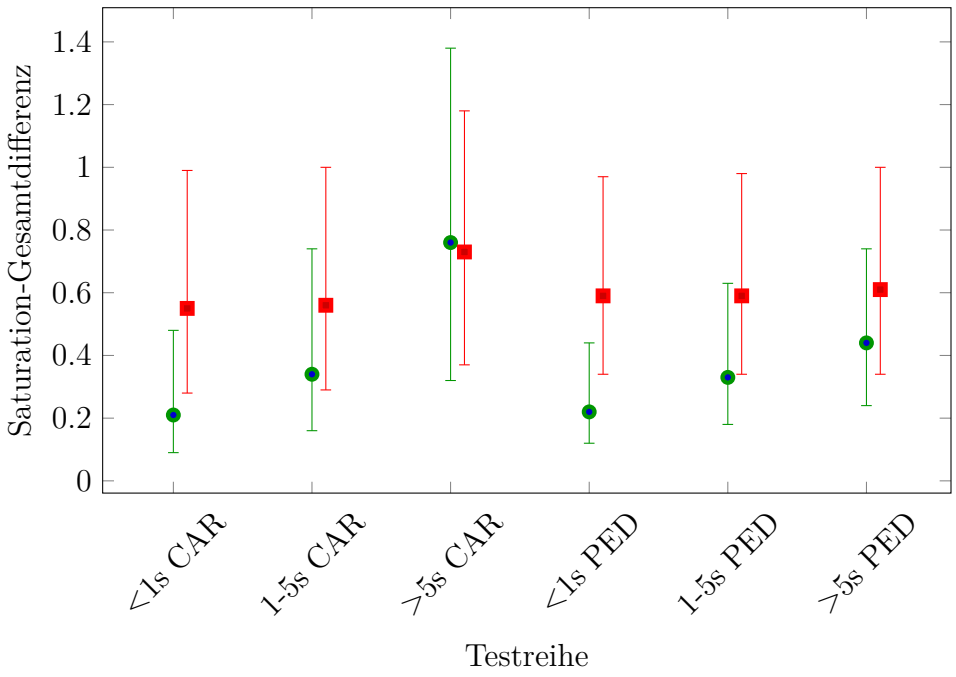

Abbildung 5.6: Statistische Analyse: Saturation-Gesamtdifferenz. Grün: richtige Zuordnungen, Rot: falsche Zuordnungen

Abbildung 5.6 zeigt die statistische Auswertung der Differenz im Saturation-Kanal. Die Resultate sind den Ergebnissen Hue-Kanals sehr ähnlich. Daher wird hier dieselbe Schlussfolgerung wie beim Hue-Kanal gezogen.

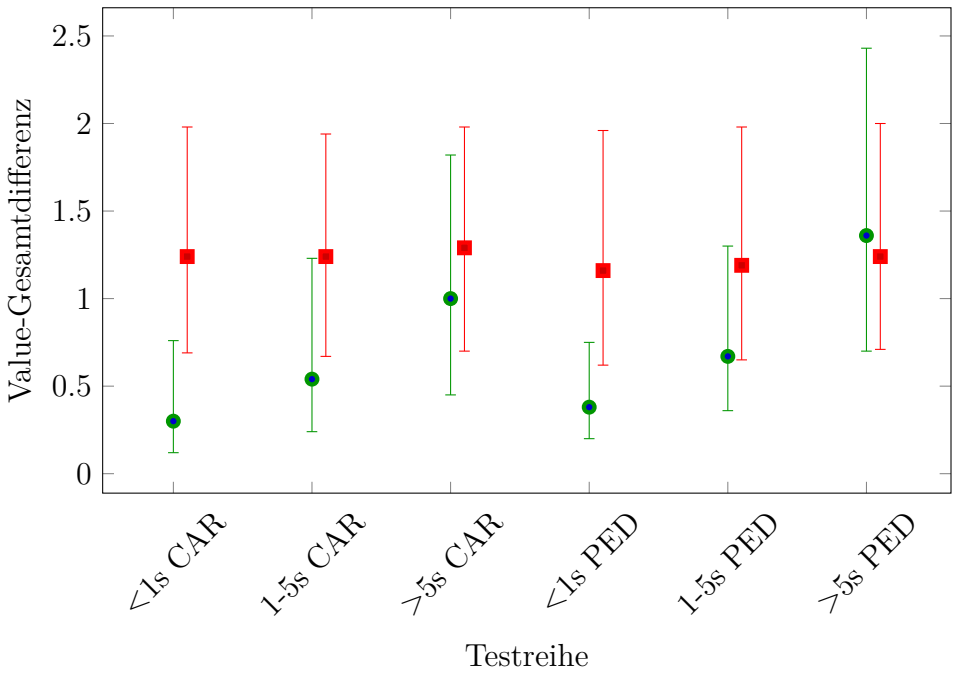

Abbildung 5.7: Statistische Analyse: Value-Gesamtdifferenz. Grün: richtige Zuordnungen, Rot: falsche Zuordnungen

Abbildung 5.7 zeigt die statistische Auswertung der Differenz im Value-Kanal. Auch hier ist in den Resultaten eine große Ähnlichkeit zu den Diagrammen des Hue- und des Saturation-Kanals zu sehen. Daher wird auch hier die Schlussfolgerung übernommen, die für den Hue-Kanal gezogen wurde.

Da sich die Ergebnisse aller drei Kanäle sehr ähnlich sind wird für alle drei folgende Schlussfolgerung gezogen: Im Bereich unter einer Sekunde ist die Überschneidung der Wertebereichen noch gering wodurch die Merkmale zur Unterscheidung geeignet sind. Zwischen 1 und 5 Sekunden gibt es bereits große Überschneidungen zwischen den Wertebereichen, zur Unterscheidung zwischen richtigen und falschen Zuordnungen ist das Merkmal jedoch, mit Einschränkungen, geeignet. Ab 5 Sekunden ist die Überschneidung zu groß, um eine gute Unterscheidung durchzuführen.

Scale-invariant feature transform (SIFT)/Speeded Up Robust Features (SURF)

SIFT und SURF sind Algorithmen zur lokalen Merkmalsfindung in Bildern. Dabei werden markante Punkte in einem Bild gesucht und es werden für diese Punkte Merkmale berechnet. Diese Merkmale verhalten sich weitgehend invariant gegenüber Transformationen. Außerdem sind sie sehr robust gegenüber Beleuchtungsunterschiede, Rauschen und leichten geometrischen Deformationen [19, 20].

Beide Algorithmen bestehen jeweils aus einem Detektor, welcher markante Punkte im Bild findet und einem Deskriptor, welcher für die gefundenen Punkte Merkmale berechnet.

Die Merkmale eines Objekts werden berechnet, indem zuerst im Bild innerhalb der Bounding-Box des Objekts Punkte mithilfe des Detektor des ausgewählten Algorithmus (SIFT oder SURF) gesucht werden. Für jeden dieser Punkte werden mit dem ausgewählten Algorithmus (SIFT oder SURF) die Merkmale (Deskriptor) berechnet. Die Punkte und ihre Deskriptoren werden als Merkmale an das Objekt angehängt.

Beim Vergleich werden zwei Objekte (Objekt A und Objekt B) miteinander Verglichen. Dafür werden mithilfe eines FLANN-basierten Matchers die Deskriptoren der Punkte von Objekt A mit den Deskriptoren der Punkte von Objekt B verglichen. Das Matching wird per K-Nearest Neighbour (KNN) mit k=2 durchgeführt [21]. Daraus resultiert eine Menge an Matches. Jeder Match enthält einen Merkmalspunkt von Objekt A (Punkt X_A) und die zwei Merkmalspunkte von Objekt B (Punkt U_B mit der höchsten Übereinstimmung zu X_A und Punkt V_B mit der zweithöchsten Übereinstimmung zu X_A), welche die höchste Übereinstimmung zu Punkt X_A haben. Es werden die Matches gelöscht, für die die Übereinstimmung zwischen X_A und U_B und die Übereinstimmung zwischen X_A und V_B sich zu ähnlich sind und die Zuweisung somit mehrdeutig sein könnte [22, 23]. Folgende Überprüfung wird durchgeführt:

$$\text{Uebereinstimmung}(X_A, U_B) < D * \text{Uebereinstimmung}(X_A, V_B) \tag{5.9}$$

Wobei Uebereinstimmung eine Funktion des Matchers ist, welcher für zwei Deskriptoren einen einzelnen Abstandwert berechnet. Je geringer dieser Abstand, desto größer ist die Ähnlichkeit. Wobei D ein beliebig gewählter Faktor ist. Je kleiner dieser Wert ist, desto weniger werden falsche Punkte assoziiert und mehr richtige Assoziationen werden entfernt.

D. Lowe beschreibt in seinem Paper, dass ein Wert von 0,8 90% falsche und nur 5% richtige Assoziationen aussortiert [24]. Zur Erstellung der Statistik wurde ein Wert von 0,7 gewählt. Dadurch werden mehr Punkte aussortiert, was notwendig war, da es zu viele falsche Assoziationen gegeben hat.

Im Letzten Schritt wird die Anzahl der übrigen Übereinstimmungen gezählt. Diese Anzahl ist der Ähnlichkeitsfaktor. Je größer dieser Wert, desto ähnlicher sind sich Objekt A und Objekt B.

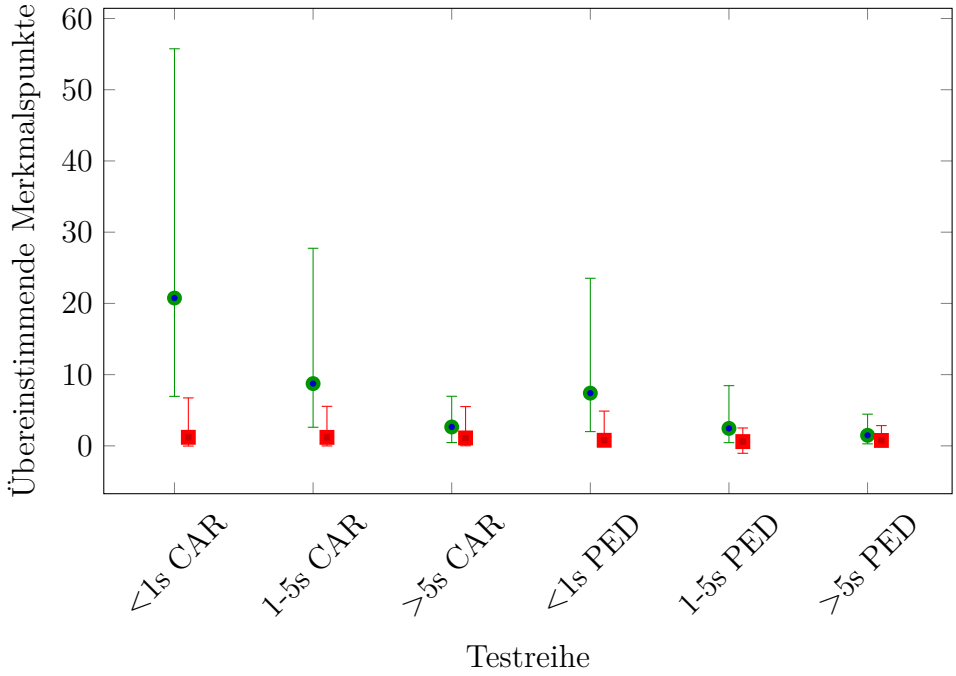

Abbildung 5.8: Statistische Analyse: Anzahl übereinstimmende Merkmalspunkte SIFT. Grün: richtige Zuordnungen, Rot: falsche Zuordnungen

Abbildung 5.8 zeigt die statistische Auswertung für die SIFT Merkmalspunkte. Bei einem Zeitabstand von weniger als 1 Sekunde überschneiden sich die beiden Wertebereiche nicht. Bei einem Zeitabstand von 1 bis 5 Sekunden gibt es nur eine kleine Überschneidung. Ab 5 Sekunden sind sich beide Wertebereiche nahezu identisch. Es gibt auch einen signifikanten Unterschied in den Resultaten für Autos und für Fußgänger. Bei Fußgängern liegen die Wertebereiche für richtige und falsche Zuweisungen wesentlich näher beieinander.

Daraus lässt sich schließen, dass der SIFT-Algorithmus für den Vergleich von Objekten gut geeignet ist, wenn diese weniger als 5 Sekunden verschwunden waren. Obwohl die Resultate für Fußgänger schlechter sind als die für Autos, ist eine Unterscheidung nach wie vor möglich. Ab 5 Sekunden ist keine sinnvolle Unterscheidung mehr möglich.

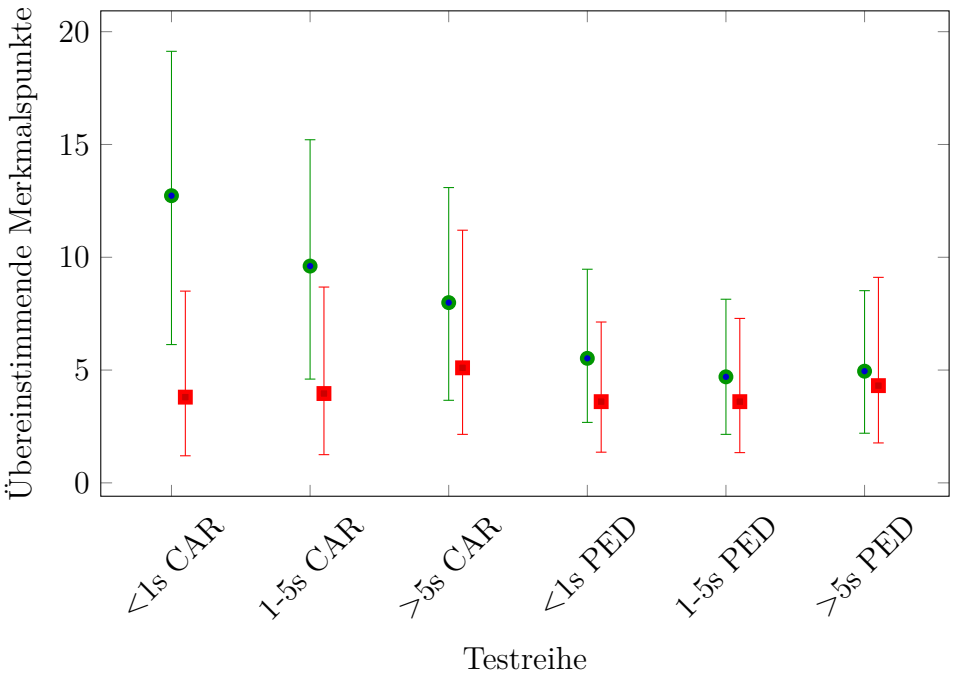

Abbildung 5.9: Statistische Analyse: Anzahl übereinstimmende Merkmalspunkte SURF. Grün: richtige Zuordnungen, Rot: falsche Zuordnungen

Abbildung 5.9 zeigt die statistische Auswertung für die SURF Merkmalspunkte. Die Resultate für den SURF-Algorithmus zeigen für Autos ähnliche Resultate. Ist ein Auto weniger als 5 Sekunden verschwunden, ist die Methode für die Zuordnung geeignet, bei größeren Zeitabständen jedoch nicht. Die Resultate für Fußgänger zeigen, dass der SURF-Algorithmus nicht für die Zuordnung von Fußgängern geeignet ist. In allen Testreihen ist die Überschneidung so groß, dass eine Unterscheidung kaum möglich ist, auch wenn das Objekt weniger als 1 Sekunde verschwunden ist.

Beim Vergleich der Statistiken ergibt sich, dass der SIFT-Algorithmus (besonders bei Fußgängern) bessere Resultate liefert als der SURF-Algorithmus. Der Vorteil des SURF-Algorithmus liegt darin, dass dieser nach eigenen Angaben weniger Rechenaufwand bedarf als der SIFT-Algorithmus [20]. Im Rahmen dieser Arbeit ist der SIFT-Algorithmus besser geeignet, da die besseren Resultate des SIFT-Algorithmus wichtiger sind als die bessere Performance des SURF-Algorithmus.

Weitere lokale Merkmale

Zusätzlich zu SIFT und SURF wurden noch die Merkmalsextraktoren ORB und AKAZE getestet. Beide sind in der OpenCV-Bibliothek enthalten. Da die Ergebnisse dieser Beiden im Test jedoch weitaus schlechter ausfielen als SIFT und SURF wird hier nicht weiter darauf eingegangen.

Fazit
Alle getesteten Ähnlichkeitsmaße haben zunehmend Schwierigkeiten je länger das Objekt verschwunden ist. Um ein Objekt jedoch erfolgreich wiederzufinden, ist es notwendig eine ausreichend gute Assoziation auch bei langen Zeitabständen zu schaffen.

Der DETMOT-Evaluator, welcher in Abschnitt 5.4 beschrieben ist, hat zu dem ergeben, dass die Menge an notwendigen Wiedererkennungen an Relevanz weit hinter anderen Problemen, wie der zu großen Anzahl Falsch-Negativer oder Falsch-Positiver liegt.

Die Implementierung des Trackers nutzt im aktuellsten Stand zur Wiedererkennung das HSV-Histogramm.

5.3 Vorhersage

Die Vorhersage von Objektpositionen stellt ein Objekttracking ohne unterstützende Detektionsinformationen dar. Es kommt immer dann zum Einsatz, wenn der Detektor ein Objekt nicht durchgehend erkennt, es sich jedoch trotzdem nach wie vor im Bild befindet. Das Ziel ist es dabei sowohl die neue Position des Objekts zu berechnen, als auch ein Qualitätsmaß zu schaffen, mit dem geprüft werden kann ob, die Vorhersage korrekt ist oder nicht.

Zur Vorhersage der Positionen wird das Medianflow-Verfahren genutzt. Im folgenden Abschnitt wird die Bestimmung der Merkmalspunkte für den Medianflow-Tracker beschrieben. Für eine genaue Beschreibung hiervon siehe Abschnitt 2.2.1.

5.3.1 Bestimmung der Merkmalspunkte

Im Rahmen dieser Arbeit wird der FAST-Algorithmus genutzt. Diese Punkte werden hier mit P_n notiert, wobei n der Index des Punktes ist [10].

Die damit berechneten Punkte werden noch nach ihrer Entfernung zum Mittelpunkt der Bounding-Box gefiltert. Hierfür wird zuerst die relative Position zum Mittelpunkt berechnet. Abbildung 5.10 illustriert die Funktionalität des Filters. Alle Merkmalspunkte die im Filterkreis liegen werden übernommen, der Rest verworfen.

Die X- und Y-Distanz werden durch die Höhe und Breite der Bounding-Box geteilt und mit 2 multipliziert. Damit liegen sie im Wertebereich [-1, 1]. Die Werte werden folgendermaßen berechnet:

$$XDist_{Rel} = \frac{2 * XDistanz}{Breite} \tag{5.10}$$

$$YDist_{Rel} = \frac{2 * YDistanz}{Höhe} \tag{5.11}$$

Aus diesen beiden Werten wird die Filterdistanz (FD) berechnet:

$$FD = \sqrt{XDist_{Rel}{}^2 + YDist_{Rel}{}^2} \tag{5.12}$$

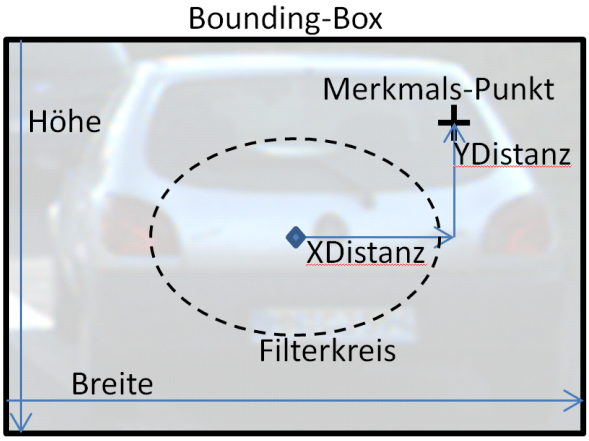

Abbildung 5.10: Distanzberechnung zur Filterung der Merkmalspunkte. Alle Punkte, welche innerhalb des Filterkreises liegen werden zur Positionsbestimmung genutzt. Der Filterkreis ist immer relativ zur Bounding-Box.

Ist FD kleiner als ein vorgegebener Schwellwert, wird der Merkmalspunkt anerkannt, sonst nicht. Diese Filterung ermöglicht es Teile des Hintergrundes (in der im Falle einer Verdeckung Teile des anderen Objektes) auszublenden, da diese bei der Vorausberechnung der Position des Objektes stören.

Für die verbleibenden Punkte wird geprüft, ob genügend Punkte zur Vorhersage vorhanden sind. Ist die Anzahl übriger Merkmalspunkte kleiner als ein vorgegebener Grenzwert wird die Vorhersage als zu instabil angesehen und das Objekt wird in *Lost* verschoben. Dies geschieht meist, wenn eine Bounding-Box zu klein ist.

Mit den übrigen Punkten wird das Medianflow-Verfahren wie in Abschnitt 2.2.1 beschrieben durchgeführt.

5.3.2 Validierung der Vorhersage

Zur Validierung der Trackingergebnisse wird der FB-Fehler berechnet, wie er in Abschnitt 2.2.1 beschrieben ist. Dieser Wert wird jedoch zusätzlich durch die Höhe des Bildes und durch die Zeitdifferenz zwischen den beiden verarbeiteten Bildern geteilt um so die Grenzwerte unabhängiger von der Bildgröße und der Bildrate zu haben.

Die Berechnung geschieht wie folgt:

$$\text{FBFehler}_{\text{normiert}} = \frac{\text{FBFehler}}{\text{höhe(Bild)} * \Delta t} \tag{5.13}$$

Wobei Δt die Differenz der Zeitstempel von zwei aufeinander folgenden Bildern ist (Bild_{Neu} und Bild_{Alt}). Höhe(Bild) ist die Höhe des alten oder des neuen Bildes in Pixel. Es wird davon ausgegangen, dass sich die Auflösung der Bilder nicht ändert.

Überschreitet diese Fehlergröße ($\text{FBFehler}_{\text{normiert}}$) einen festgelegten Grenzwert, wird die

Vorhersage als ungültig angesehen und das Objekt wird in *Lost* verschoben.

Out-of-Frame-Quote
Dieser Wert beschreibt das Verhältnis der Punkte, welche nicht mehr im Bild liegen zu der Gesamtmenge an Punkten. Das LK-Verfahren kann Bewegungsvektoren auch für Punkte berechnen, welche im neuen Bild außerhalb des Bildes liegen. Diese Punkte werden gezählt und die Anzahl wird hier durch N_{OOF} dargestellt. Die Out-of-Frame (OOF)-Quote berechnet sich foldendermaßen:

$$\mathrm{OOF} - \mathrm{Quote} = \frac{N_{\mathrm{OOF}}}{\mathrm{GesamtzahlPunkte}} \tag{5.14}$$

Die OOF-Quote wird über aufeinander folgende Bilder integriert und es entsteht eine Gesamtsumme.

Überschreitet die Gesamtsumme einen bestimmten Grenzwert, wird die Vorhersage als ungültig erklärt.

5.4 DETMOT Evaluator

Die Grundidee hinter dem DETMOT-Evaluator ist es konkrete Fehlerquellen innerhalb eines Trackers ausfindig zu machen. Die Bewertung soll auf der Grundlage der Resultate des Trackers und des dazugehörigen Detektors, sowie auf den Ground-Truth-Daten basieren. Damit soll festgestellt werden, welche der Trackingfehler ihren Ursprung in Fehlern des Detektors haben. Ein Objekt, welches zwar in den Ground-Truth-Daten ist, jedoch nie detektiert wurde, kann vom Tracker nicht gesehen werden, da dieser die Objekte vom Detektor übernimmt. Dies würde bei der Evaluation als Trackingfehler gewertet. Solche Fehler werden gezählt und in einem separaten Wert in die Evaluierung mit einbezogen.

Für die Trackingfehler wird eine detaillierte Analyse durchgeführt um, den Ursprung eines Fehlers einzugrenzen und eine Verbesserung zu ermöglichen.

Um die Formeln zu verkürzen wird jedem Bewertungsmaß eine Kurzform zugewiesen, welches in Klammern hinter dem Wert in der Beschreibung steht.

5.4.1 Auswertung Detektor

Der DETMOT-Evaluator führt zusätzlich zur Evaluierung des Trackers eine Evaluierung des Detektors durch und gibt die Ergebnisse davon aus.

Hierbei werden Folgende Ereignisse gezählt:

richtig Positive (TP) Ein Elemente taucht sowohl in den GT-Daten, als auch in den Detektionsergebnissen auf

falsch Positive (FP) Ein Elemente taucht nicht in den GT-Daten auf, wird jedoch vom Detektor erkannt

falsch Negative (FN) Ein Elemente das in den GT-Daten auftauchen, jedoch nicht in den Detektionsergebnissen

Ground-Truth-Daten (GT) Gesamtmenge der GT-Elemente

Aus diesen Zahlen werden folgende Werte berechnet:

Falsch Positive pro Frame (FAF)

$$FAF = \frac{FP}{\text{AnzahlFrames}} \tag{5.15}$$

Der Wert gibt an wie viele Falsch Positive pro Bild durchschnittlich erkannt werden.

Genauigkeit ($Accuracy$)

$$\text{Accuracy} = 1 - \frac{FN + FP}{GT} \tag{5.16}$$

Dieser Wert stellt eine Vereinigung von falsch Negativen und falsch Positiven dar. Dieser Wert ist kein reiner Prozentanteil, da der Wertebereich von 1 bis -∞ reicht. Die Genauigkeit ermöglicht jedoch eine Abschätzung der „Falschheit“, da dieser Wert das Verhältnis aller Trackingfehler zur Anzahl der Ground-Truth-Daten angibt. Die Berechnungsformel wird auch in ähnlicher Form im CLEAR MOT, zur Berechnung des MOTA-Wertes genutzt [3].

Empfindlichkeit ($Recall$)

$$\text{Recall} = \frac{TP}{TP + FN} \tag{5.17}$$

Der Wert gibt an, wie gut Objekte erkannt werden. Das bedeutet, wie viele der zu erkennenden Objekte wurden tatsächlich erkannt.

Präzision ($Precision$)

$$\text{Precision} = \frac{TP}{TP + FP} \tag{5.18}$$

Der Wert gibt an, wie verlässlich die Resultate sind. Das bedeutet, wie viele der der erkannten Objekte sind tatsächlich vorhanden.

Bounding-Box-Prezision ($BBPrecision$)

$$\text{BBPrecision} = \frac{\text{SummeallerÜberlappungsverhältnisse}}{TP} \tag{5.19}$$

Dieser Wert gibt an, wie genau die Bounding-Boxen sind für Objekte, welche richtig erkannt wurden.

5.4.2 Filterung Ground-Truth

Der erste Schritt bei der Evaluation ist die Filterung der Ground-Truth Elemente. Hierbei wird geprüft welche der Gound-Truth-Daten für den Tracker überhaupt trackbar sind. Da der Tracker auf einem Objektdetektor aufbaut, kann der Tracker keine Objekte tracken, welche nicht vom Detektor erkannt werden. Dies wirkt sich auf alle Bewertungsmaße aus.

Jedem Ground-Truth-Element wird ein Zustand zugewiesen. Abbildung 5.11 zeigt die Zustände und die Übergänge.

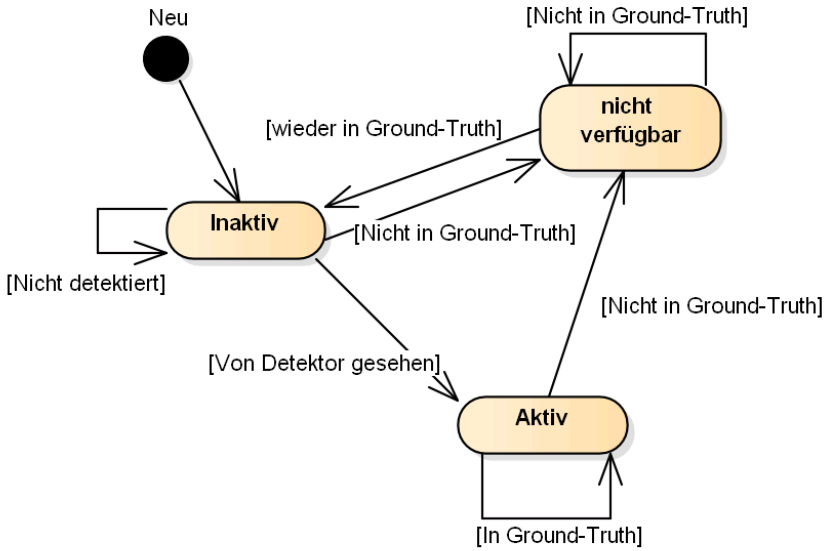

Abbildung 5.11: Zustände Ground-Truth Elemente.

Kommt ein Ground-Truth Element neu hinzu, gelangt es zuerst in den Zustand *Inaktiv*. Solange es sich in diesem Zustand befindet, werden Falsch-Negative, welche dieses Objekt betreffen separat gezählt, um daraus später ein korrigiertes Fehlermaß zu berechnen.

Wird das Objekt vom Detektor erfasst, wechselt es in den Zustand *Aktiv*. In diesem Zustand wird vom Tracker erwartet, dass er das Objekt verfolgt. Ist es in den Trackingergebnissen nicht vorhanden, wird dies als aktiver Falsch-Negativ Fehler angesehen und als solcher gezählt.

Ist das Objekt in *Inaktiv* und verschwindet aus den Ground-Truth-Daten, wechselt es in den Zustand *nicht verfügbar*. Hier wird weder vom Detektor, noch vom Tracker erwartet, dass sie das Objekt sehen. Sollten sie das Objekt trotzdem erkennen, handelt es sich um einen inaktiven Falsch-Positiv Fehler und wird als solcher gezählt.

Ist das Objekt in *Aktiv*, verbleibt es dort, solange, bis es aus den Ground-Truth-Daten verschwindet. Vom Tracker wird erwartet, dass er das Objekt nach wie vor erfasst, auch wenn es nicht vom Detektor erkannt wird. Sobald das Objekt aus den Ground-Truth-Daten verschwindet, wechselt es in *nicht verfügbar*.

Mithilfe der Ground-Truth-Filterung werden die Fehler gezählt, für welche nicht der Tracker verantwortlich ist. Diese korrigierten Werte werden genutzt um die Werte der CLEAR MOT und MT/PT/ML Statistiken separat zu berechnen. Dabei werden die Werte sowohl korrigiert, als auch nicht korrigiert ausgegeben, da beide Formen zur Evaluierung wichtig sind.

5.4.3 Zuweisung

Ein wichtiger Punkt zur Evaluierung ist die Zuweisung von detektierten/getrackten Objekten zu Ground-Truth-Daten. Dafür gibt es zwei verschiedene Use-Cases in denen dies benötigt wird:

1. Zuweisung von Ground-Truth-Objekten zu detektierten Objekten
2. Zuweisung von Ground-Truth-Objekten zu getrackten Objekten

Zuweisung von Ground-Truth-Objekten zu detektierten Objekten

Hierbei geht es darum eine Zuweisung zwischen Detektionsergebnissen und GroundTruth-Ergebnissen vorzunehmen, um so zu prüfen welches Objekt aktiviert werden muss. Hierbei ergibt sich ein wichtiges Problem: Wenn ein Detektionsergebnis zwei verschiedenene GT-Elementen zugewiesen werden könnte, kann der Evaluator nicht eine der beiden Zuweisungen vornehmen. Ziel dieser Zuweisung ist es zu prüfen, welche Trackingprobleme von Detektor kommen und welche nicht. Wenn der Evaluator in mehrdeutigen Fällen willkürlich eine bestimmte Zuweisung auswählen würde, kann es passieren, dass der Tracker genau die andere Zuweisung vornimmt. Hierbei handelt es sich nicht um einen Fehler, da das Detektionsergebnis zu beiden GT-Elementen passen würde. Der Evaluator würde es jedoch als Fehler werten.

Aus diesem Grund wird nach folgender Regel gehandelt: Gibt es für ein GT-Element ein beliebiges Detektionsergebnis, welches ihm zugewiesen werden könnte, wird das Objekt aktiviert, ungeachtet ob dasselbe Detektionsergebnis auch zu einem anderen GT-Element passen würde. Wichtig hierbei ist es festzustellen, ob ein GT-Element vom Tracker erfasst werden kann oder nicht und diese Information ist mithilfe der genannten Methode möglich.

Als Ähnlichkeitsmaß für die Zuordnung wird das Überlappungsverhältnis genutzt. Ist das Überlappungsverhältnis größer als 0,3 wird die Zuweisung als gültig angesehen.

Zuweisung von Ground-Truth-Objekten zu getrackten Objekten

In diesem Teil geht es darum zu prüfen, welches getrackte Objekt welchem GT-Element zugewiesen wird. Die Implementierung des CLEAR MOT, welche für den KITTI-Benchmark genutzt wird, erreicht dies über die Zuhilfenahme der Ungarischen Methode [11]. Diese weist jeder Assoziation einen Kostenwert zu und bestimmt für jedes Element eine Zuweisung, so dass die Gesamtkosten minimal sind [25].

Das Problem an dieser Methode ist, dass Objekten, welche keine ausreichende Assoziation aufweisen ebenfalls Kosten zugewiesen werden müssen. In der CLEAR MOT Implementierung des KITTI-Benchmark [11] werden diese Kosten auf 10^9 gesetzt, während die Kosten für Assoziationen zwischen 0 und 1 liegen. Der Wert entspricht dem Überlappungsverhältnis der verglichenen Bounding-Boxen. Die extrem hohen Kosten bei fehlenden Assoziationen haben zur Folge, dass jede zusätzliche Assoziation die Kosten so drastisch senkt, dass der Evaluator bei der Zuweisung eine größere Menge an Zuweisungen immer einer qualitativ besseren Zuweisung vorzieht.

Im DETMOT-Evaluator wird daher anders vorgegangen. Zuerst wird, wie bei der Ungarischen Methode [25] eine Assoziationsmatrix erstellt, in die für jede Kombination die Ähnlichkeit (in diesem Fall das Überdeckungsverhältnis) eingetragen wird. Jede Spalte markiert in dieser Matrix ein Tracking-Objekt und jede Zeile ein GT-Element. Danach wird in dieser Matrix die Zelle mit dem größten Wert (also der größten Ähnlichkeit) gesucht. Die Spaltenzahl dieser Zelle markiert ein Objekt in den Trackingdaten und die Zeilenzahl ein GT-Element. Diese beiden Objekte werden miteinander assoziiert. Gleichzeitig werden in allen Zellen, welche sich in derselben Zeile oder der selben Spalte wie die gewählte Zelle befinden die Ähnlichkeit auf 0 gesetzt.

Diese Methode ist performant, da die Assoziationen in der finalen Implementierung nicht als Matrix, sondern als Liste angeordnet werden kann. Diese Liste wird aufsteigend nach der Ähnlichkeit sortiert, so dass die Suche nach dem größten Element eine Operation mit konstanter Laufzeit wird. Ein weiterer Vorteil dieser Methode ist, dass immer die Bounding-Boxen mit der größten Ähnlichkeit miteinander assoziiert werden. Der Nachteil ist, dass es hier häufiger vorkommen kann das Objekte nicht assoziiert werden als bei der ungarischen Methode. Der Vorteil ist jedoch, dass die Objekte, welche die größte Ähnlichkeit haben auch einander zugewiesen werden.

5.4.4 Auswertung Tracker

Für den Tracker werden die Fehler gezählt, wie bereits beim Detektor. Dabei werden folgende Ereignisse gezählt:

richtig Positive (TP) Anzahl der Elemente die sowohl in den GT-Daten, als auch in den Trackingergebnissen auftauchen

falsch Positive (FP) Anzahl der Elemente die nicht in den GT-Daten sind, jedoch vom Tracker erkannt werden

falsch Negative (FN) Anzahl der Elemente die in den GT-Daten auftauchen, jedoch nicht in den Trackingergebnissen

Ground-Truth-Daten (GT) Anzahl der GT-Elemente

ID-Switches (IDS) Anzahl der ID-Wechsel, das bedeutet ein Objekt, welches in den GT-Daten die selbe Identifikationsnummer hat, hat in den Ergebnissen des Trackers plötzlich eine andere.

inaktive falsch Negative (IFN) Anzahl der Falsch negativen, bei denen das korrespondierende GT-Element inaktiv war. Siehe hierfür Abschnitt 5.4.2.

aktive detektierte falsch Negative ($ADFN$) Anzahl der falsch Negativen, welche aktiv waren und vom Detektor erkannt wurden, jedoch nicht vom Tracker.

aktiv nicht Detektierte (AD) Anzahl der Objekte, welche aktiv waren, aber nicht vom Detektor erfasst wurden. Vom Tracker wird bei diesen Objekten erwartet, dass er sie weiterhin trackt.

Aus diesen Zahlen werden folgende Statistiken berechnet:

CLEAR MOT-basiert

Der DETMOT-Evaluator soll auf den statistischen Werten der CLEAR MOT-Statistik aufbauen, da sich diese zur Evaluierung von Objekttrackern weitgehend durchgesetzt haben. In diesem Abschnitt werden die Werte beschrieben, die ihren Ursprung in der CLEAR MOT-Statistik haben und die, welche zusätzliche Informationen darstellen.

Weitere Informationen zur CLEAR MOT Statistik können in Abschnitt 2.3.1 und in dem Paper von Bernardin und Stiefelhagen nachgelesen werden [3].

Die CLEAR MOT Implementierung des KITTI-Benchmarks implementier einige Zusatzregeln, welche in der Ursprungsform des CLEAR MOT nicht vorgesehen sind. Folgende Regeln werden hinzugefügt [11]:

1. Verwandte Klassen werden nicht als Falsch Positive gewertet (z. B. *Van* nicht als Falsch Positiv für *Car*).
2. Falsch Positive die zu 50% von einem sog. *Don't Care*-Label überlagert sind, werden nicht als Fehler gewertet. *Don't Care*-Areale sind oft scheinbar willkürlich gesetzt und markieren Bereiche die für lernfähige Algorithmen uneindeutige Szenen darstellen.
3. Objekte die nicht vollständig im Bild sind werden ignoriert.

Zusatzregel Nr. 1 und 3 können im Evaluator optional eingeschaltet werden. Für Regel 3 kann explizit angegeben werden, wie viel des Objekts im Bild sein muss (Flächenanteil in %) um es in die Evaluierung einzubeziehen.

Zusatzregel Nr. 2 ist im DETMOT-Evaluator nicht umgesetzt, da diese im Kontext dieser Arbeit nicht als sinnvoll erachtet wurde. *Don't Care*-Areale bedecken häufig sehr verschiedene Teile des Bildes und dienen als Hinweise für den Lernalgorithmus. Zur Evaluierung ist es jedoch nicht sinnvoll, Fehler nach willkürlichen Maßstäben zu ignorieren. Abbildung 5.12 zeigt ein Beispiel hierfür. Das *Don't Care*-Label am linken Rand dient dazu dem Lernalgorithmus mitzuteilen, dass das Fahrzeug, welches sich unter dem Label befindet, weniger relevant ist, da es größtenteils verdeckt ist. Bei der Evaluierung kann es jedoch in diesem Bereich sehr schnell zu falsch Positiven kommen, welche in der Fehlerstatistik durch das *Don't Care* ignoriert werden.

Abbildung 5.12: Bildausschnitt, welcher eine Szene mit mehreren *Don't Care*-Labels zeigt. Diese liegen an Bereichen im Bild verteilt, welche leicht zu Problemen bei Objektdetektoren führen können und sollen Lernalgorithmen helfen, die Relevanz von Fehlern einzuschätzen.

MOTA

Die Multi-Object-Tracking-Accuracy berechnet sich folgendermaßen:

$$\text{MOTA} = 1 - \frac{FN + FP + IDS}{GT} \tag{5.20}$$

Der MOTA-Wert ist das Verhältnis aller Trackingfehler zu der Menge an Ground-Truth-Daten. Er bietet daher eine Grundlage, um Trackingalgorithmen miteinander zu vergleichen.

Im nächsten Schritt wird die Anzahl an Trackingfehlern (FN, FP, IDS) um die *inaktive falsch Negative* (IFN) korrigiert, wodurch viele Fehler ausgenommen werden, welche ihren Ursprung nicht im Tracker sondern im Detektor haben. Die Berechnung sieht wie folgt aus:

$$\text{MOTA}_{\text{Korrigiert}} = 1 - \frac{(FN + FP + IDS) - IFN}{GT} \tag{5.21}$$

Ein $MOTA_{Korrigiert}$ von 1,0 würde bedeuten, dass der Tracker alle Objekte die er tracken kann auch trackt. Das bedeutet nicht, dass die Tracking-Resultate keine Fehler mehr enthalten sondern, dass die Fehler, welche übrig sind, nicht durch den Tracker, sondern durch den Detektor verursacht werden.

Beide Werte ($MOTA$ und $\text{MOTA}_{\text{Korrigiert}}$) erlauben keine absolute Einschätzung des Algorithmus sondern bieten ausschließlich ein Bewertungsmaß, um Algorithmen miteinander zu vergleichen.

MOTP

Die Multi-Object-Tracking-Precision beschreibt die Genauigkeit der Bounding-Boxen. Hierfür wird für jedes richtig Positive das Überlappungsverhältnis zwischen dem dazugehörigen Ground-Truth-Element und dem damit assoziierten Trackingergebnis bestimmt. Diese Werte werden für alle richtig Positiven miteinander addiert. Die Gesamtsumme wird durch die Anzahl an richtig Positiven geteilt. Die Berechnungsformel sieht wie folgt

aus:

$$\text{MOTP} = \frac{\sum \text{Überlappungsverhältnisse}}{TP} \tag{5.22}$$

Dieser Wert liegt zwischen 0 (keine Überlappung) und 1 (vollständige Überlappung). Dieser Wert gibt an wie genau Größe und Positionen von richtigen Objekten durch den Tracker bestimmt werden.

5.4.5 Auswertung Vorhersage

Ein Objekt, welches nicht vom Detektor erkannt wird, vom Tracker jedoch trotzdem weiterhin beobachtet wird gilt als Vorhersage. Die Voraussetzung hierfür ist, dass das Objekt aktiv ist, es wurde also bereits mindestens einmal detektiert. Es wird vom Objekttracker erwartet, dass dieser Objekte auch dann weiter verfolgen kann, wenn diese nicht durchgehend detektiert werden.

Zur detaillierten Auswertung der Qualität der Vorhersage werden zunächst folgende Ereignisse gezählt:

Vorherzusagende (V) Anzahl der vorhergesagten Objekte. Hierfür werden alle Trackingresultate gezählt, für welche kein passendes Detektionsergebnis vorliegt.

richtig Vorhergesagte (RV) Anzahl der Vorhersagen, welche korrekt sind. Das bedeutet ein Objekt befindet sich in den GT-Daten, wurde vom Detektor nicht erfasst, ist jedoch trotzdem in den Trackingergebnissen.

falsch Vorhergesagte (FV) Anzahl Objekte, welche nicht aktiv sind aber trotzdem vorhergesagt wurden. Hierbei handelt es sich um Objekte, welche vom Tracker erfasst wurden, jedoch nicht vom Detektor und die auch nicht in den GT-Daten auftauchen.

falsch Nicht-Vorhergesagte (FNV) Anzahl Objekte, welche vorhergesagt werden sollten, es aber nicht wurden. Hierbei handelt es sich um Objekte, welche in den GT-Daten vorhanden sind, jedoch weder vom Detektor, noch vom Tracker erfasst wurden. Diese Objekte wurden in einem vorherigen Zeitschrit mindestens einmal detektiert und sind so aktiv.

Aus diesen Zählungen werden folgende Werte berechnet:

Vorhersagenverhältnis

Das Vorhersagenverhältnis beschreibt den relativen Anteil der GT-Elemente, welche vorhergesagt werden müssen. Dieser Wert dient dazu, festzustellen, wie relevant die Vorhersage bei der Bewertung ist. Dies dient dazu festzustellen wo ein Tracker das größte Verbesserungspotential aufweist.

$$\text{Vorhersagenverhältnis} = \frac{RV + FNV}{GT} \tag{5.23}$$

Ein hohes Vorhersagenverhältnis lässt auf einen schlechten Objektdetektor schließen.

Anteil richtiger Vorhersagen (RVRel)

RVRel repräsentiert den Anteil der Vorherzusagenden Elemente, die korrekt vorhergesagt wurden. Die Berechnung sieht folgendermaßen aus:

$$\text{RVRel} = \frac{RV}{V} \tag{5.24}$$

Der Wert liegt zwischen 0 und 1. Je größer dieser Wert ist, desto verlässlicher sind die Vorhersagen, welche vom Tracker getroffen werden.

Anteil falscher Vorhersagen (FVRel)

FVRel gibt an wie groß der Anteil der falschen Vorhersagen zur Anzahl der vorhergesagten Elemente ist.

$$\text{FVRel} = \frac{FV}{V} \tag{5.25}$$

Dieser Wert liegt zwischen 0 und 1 und sollte im Best-Case bei 0 liegt. Mit diesem Wert lässt sich abschätzen, wie gut der Tracker erkennt wann er aufhören muss ein Objekt weiter zu verfolgen.

Vorhersage Empfindlichkeit (Recall)

$$\text{Recall}_{\text{Vorhersage}} = \frac{RV}{RV + FNV} \tag{5.26}$$

Dieser Wert gibt an, wie viele der vorherzusehenden Objekte tatsächlich vorhergesehen wurden. Der Wert liegt zwischen 0 und 1.

Vorhersage Präzision ($Precision$)

$$\text{Precision}_{\text{Vorhersage}} = \frac{RV}{RV + FV} \tag{5.27}$$

Mit diesem Wert lässt sich feststellen, wie viele der vorhergesehenen Objekte auch in den GT-Daten vorhanden sind. Der Wertebereich liegt zwischen 0 und 1.

5.4.6 Auswertung ID-Switches

Dieser Abschnitt beschreibt die Auswertung von ID-Switches. Dazu zählt die Auswertung verschwundener und wiedergefundener Objekte.

Im ersten Schritt werden folgende Ereignisse gezählt:

ID-Switches (IDS) Anzahl der Objekte, welche ihre Identifikationsnummer zwischen zwei Zeitschritten gewechselt haben.

Nach Verschwinden (NV) Anzahl ID-Switches, die auftraten, nachdem ein Objekt verschwunden (in *Lost*) war. Das bedeutet, dass mehrere Zeitschritte zwischen dem Verschwinden und dem Auftauchen des Objektes mit anderer ID liegen.

Zwischen Frames (ZF) Anzahl ID-Switches, die zwischen zwei Bildern aufgetreten sind, obwohl das Objekt durchgehend erfasst wurde.

richtig Wiedererkannte (RW) Anzahl der Objekte, die nach Ihrem Verschwinden erfolgreich wiedergefunden und mit der richtigen Identifikationsnummer versehen wurden.

Tracks welche mindestens einmal in *Lost* waren (TMV) Anzahl an Tracks, welche mindestens einmal verschwunden und wieder aufgetaucht sind.

Fälschlich als Neu hinzugefügt (FNH) Anzahl an Objekten, welche beim Tracker verschwunden sind (in *Lost* verschoben wurden), wieder aufgetauchten und nicht mit ihrer früheren Identifikationsnummer, sondern mit einer Neuen versehen wurden.

Gestohlener Track (GS) Anzahl der Objekte, welche verschwanden, mehr als einen Zeitschritt verschwunden blieben, wieder aufgetauchten und mit einer anderen, bereits existierenden, Identifikationsnummer assoziiert wurden. Diese Objekte haben damit einen anderen Track übernommen.

Gestohlener Track zwischen Bildern ($GSZF$) Anzahl der Objekte, die von einem Frame auf den anderen (ohne weitere Zeitschritte dazwischen) die ID eines anderen Tracks übernommen haben.

Aus diesen Werten werden folgende Statistiken berechnet:

ID-Switches Relativ ($IDSRel$)

Dieser Wert gibt das relative Verhältnis zwischen ID-Switches und GT-Daten an. Berechnung:

$$IDSRel = \frac{IDS}{GT} \tag{5.28}$$

Dieser Wert stellt die Relevanz der Fehlerbehandlung von ID-Switches dar. Dieser Wert dient dazu, den Fokus bei der Entwicklung des Trackers auf die Teile zu lenken, welche die größte Fehlerquellen darstellen, ähnlich wie der Wert $TVRel$.

Track mit ID-Switches (TIS)

Dieser Wert gibt den Anteil an Tracks an, welche mindestens einen ID-Switch enthalten. Berechnung:

$$TIS = \frac{T_{IDS}}{T_{All}} \tag{5.29}$$

Wobei T_{IDS} die Anzahl an Tracks ist, welche mindestens einen ID-Switch enthalten. T_{All} ist die Gesamtanzahl aller Tracks.

Anteil nach Verschwinden (NV_{Rel})

Dieser Wert stellt den Anteil der ID-Switches dar, welche aufgetreten sind nachdem das

Objekt verschwunden (in *Lost*) war. Berechnung:

$$NV_{Rel} = \frac{NV}{IDS} \tag{5.30}$$

Zusammen mit dem nächsten Wert (ZF_{Rel}) dient dieser dazu, festzustellen welcher Assoziationsprozess innerhalb des Trackers mehr Fehler verursacht. Ist der Wert NV_{Rel} größer als ZF_{Rel}, bedeutet dies, dass die Assoziation von Verschwundenen Objekten mit Detektionen die größere Fehlerquelle darstellt. Ist jedoch ZF_{Rel} größer als NV_{Rel}, liegt der Fehler in der Assoziation von Objekten zwischen zwei Bildern, also die Assoziation von aktuell getrackten Objekten mit Detektionen.

Für alle ID-Switches, welche in diese Klasse fallen wird zusätzlich geprüft ob die neue Identifikationsnummer des Objektes bereits existierte oder nicht. Hat sie bereits existiert, bedeutet dies, dass beim ID-Switch ein anderer Track übernommen (gestohlen) wurde. Existierte die Identifikationsnummer noch nicht, handelt es sich um ein neues Objekt.

Anteil zwischen Frames (ZF_{Rel})
Dieser Wert beschreibt den Anteil an ID-Switches, die zwischen zwei Frames aufgetreten sind, obwohl das Objekt durchgehend erfasst war. Berechnung:

$$ZF_{Rel} = \frac{ZF}{IDS} \tag{5.31}$$

Die Aussagekraft diese Wertes wurde bereits im vorherigen Wert (NV_{Rel}) näher erläutert.

Auch hier wird für jeden ID-Switch, dieser Klasse, geprüft ob die neue Identifikationsnummer bereits vergeben war oder nicht, um auf diesem Weg festzustellen, ob ein anderer Track übernommen oder ein neuer angelegt wurde.

5.4.7 MT/PT/ML basiert

In diesem Abschnitt werden die statistischen Werte beschrieben, welche ihren Ursprung in der MT/PT/ML, welche bereits in Abschnitt 2.3.2 beschrieben wurde. Nähere Informationen hierzu können in dem entsprechenden Paper nachgeschlagen werden [4].

Diese Werte werden als Grundlage für die Evaluierung herangezogen, da sie zur Evaluierung von Trackingalgorithmen sehr häufig genutzt werden [11, 6].

Im ersten Schritt werden wie in Kapitel 2.3.2 beschrieben, folgende Werte bestimmt:

Mostly Tracked (MT) Anzahl an Tracks, welche mindestens 80% der Zeit korrekt getracked wurden.

Mostly Lost (ML) Anzahl an Tracks, welche weniger als 80% der Zeit korrekt getracked wurden.

Partly Tracked (PT) Anzahl Tracks, welche weder zu *Mostly Tracked* oder *Mostly Lost* gehören.

Mit den Werten MT, ML und PT werden die relativen Anteile bestimmt:

$$MT_{Rel} = \frac{MT}{MT + PT + ML} \tag{5.32}$$

$$ML_{Rel} = \frac{ML}{MT + PT + ML} \tag{5.33}$$

$$PT_{Rel} = \frac{PT}{MT + PT + ML} \tag{5.34}$$

Zusätzlich wird vom DETMOT-Evaluator eine durchschnittliche Abdeckung berechnet. Hierfür wird für jeden Track das Trackverhältnis (TV siehe Abschnitt 2.3.2) bestimmt. Um die Abdeckung zu berechnen wird nun der Durchschnitt über all diese Trackverhältnisse gebildet.

Es wird außerdem für jeden Track ein aktives Trackverhältnis (siehe Formel 5.35) berechnet. Hierfür wird folgendes zusätzliches Ereignisse innerhalb eines Tracks gezählt:

Inaktive Falsch Negative (T_{IFN}) Wie viele der falsch Negative waren nicht aktiv

Das aktive Abdeckungsverhältnis ergibt sich mit folgender Formel:

$$TV_{\text{Aktiv}} = \frac{R}{(R + F + IDS) - T_{\text{IFN}}} \tag{5.35}$$

Wobei R die Anzahl der Bilder ist, in denen das Objekt getrackt ist, F ist die Anzahl der Bilder in denen es nicht getrackt wurde und IDS ist die Anzahl der ID-Switches. TV_{Aktiv} wird für jeden Track berechnet und es wird der Durchschnitt über alle Werte gebildet.

Bei der Berechnung dieser Abdeckung besteht das Problem, dass jeder Track paritätisch in diese Berechnung eingeht, ungeachtet ob er über 100 oder nur 3 Bilder zu sehen war. Aus diesem Grund wird die aktive Gesamtabdeckung zusätzlich mit einer Gewichtung der Tracks korrigiert.

Dafür wird zur Bildung des Durchschnittes aller Abdeckungen jeder Wert mit der Anzahl seiner aktiven Ground-Truth-Vorkommen multipliziert. Die Berechnungsformel sieht folgendermaßen aus:

$$\text{Gesamtabdeckung}_{\text{Gewichtet}} = \frac{\sum_{i=1}^{n}(R_i)}{\sum_{i=1}^{n}((R_i + F_i + IDS_i) - T_{IFNi})} \tag{5.36}$$

Wobei n die Anzahl an Tracks ist. R_i ist die Menge an richtigen Erkennungen des Tracks mit dem Index i. F_i ist die Menge an falschen Erkennungen des Tracks mit dem Index i. IDS_i ist die Menge an ID-Switches des Tracks mit dem Index i. T_{IFNi} ist die Menge an inaktiven falsch Positiven des Tracks mit dem Index i.

Die $\text{Gesamtabdeckung}_{\text{Gewichtet}}$ gibt an, zu welchem Anteil ein Track im Durchschnitt ge-

trackt ist.

5.4.8 Fazit

Der DETMOT-Evaluator hat sich als sehr hilfreich beim Identifizieren von Problemen erwiesen. So wurde beispielsweise festgestellt, dass die korrekte Assoziation von Objekten nach dem Verschwinden von geringer Relevanz ist, als das Auftreten von falsch Positiven oder falsch Negativen. Dies hat geholfen den Fokus der Arbeit auf die größte Problemquelle zu lenken.

Der DETMOT-Evaluator hat dabei geholfen Probleme bei der Assoziation zu finden und es konnte geprüft werden, ob die Mehrzahl der Fehler durch die falsche Assoziation mit einer Detektion oder durch die falsche Vorhersage hervorgerufen wurde.

Quelltext 7.1 im Anhang ist ein Beispiel-Bericht, welcher vom DETMOT-Evaluator erstellt wurde. Dieser stellt beispielhaft die Ausgabe des DETMOT-Evaluators dar.

5.5 Bestimmung Grenzwerte

Wie bereits in Abschnitt 5.2.1, 5.3 und 5.2.2 beschrieben, geht es beim Objekttracking um Datenassoziation. Dafür wurden Methoden vorgestellt, mit denen die Ähnlichkeit zweier Objekten nach verschiedenen Merkmalen berechnet werden kann.

Dabei ist es wichtig einen Grenzwert abhängig vom Anwendungsfall festzulegen, ab welchem eine Ähnlichkeit als Assoziation angesehen wird. Der Grenzwert soll dabei allgemein genug festgelegt werden, dass er (im Falle des Objekttrackings) in möglichst jeder Verkehrssituation zu richtigen Ergebnissen führt (z. B. Landstraßen, Autobahnen, Stadtfahrten, uvm.). Der Grenzwert sollte jedoch an Umstände angepasst werden, welche sich im laufenden Betrieb nicht ändern, wie z. B. die Kameraauflösung, den genutzten Detektor, die Relative Position der Kamera zum Fahrzeug oder die Aufnahmefrequenz. Diese Anpassung ist beim KITTI-Benchmark möglich, da die genannten Variablen über den gesamten Datensatz konstant sind. Dies ist bei der MOT-Challenge nicht möglich, da die meisten Datensequenzen unterschiedliche Auflösungen, Kamerapositionen oder Aufnahmefrequenzen haben.

Grenzwerte sind z. B.:

FB-Fehlerwert Um, wie in Abschnitt 5.3 beschriebene, die Stabilität einer Vorhersage zu bestimmen, muss eine Obergrenze für den FB-Fehler festgelegt werden, ab dem der Track als zu instabil zur weiteren Verfolgen angesehen wird.

Überlappungsverhältnis Zum Matching von detektierten und getrackten Objekten wird wie in Abschnitt 5.2.1 das Überlappungsverhältnis berechnet. Unterschreitet dieser Wert einen festgelegten Grenzwert, wird von einer gültigen Assoziation ausgegangen.

Zur Bestimmung dieser Grenzwerte wird der Tracker mit verschiedenen Parametern getestet und evaluiert. Hierfür werden die zu variierenden Parameter ausgewählt und es wird für jeden Parameter ein Wertebereich sowie eine Schrittweite für die Abtastung

ausgewählt. Im Anschluss wird jede mögliche Parameterkonfiguration, welche mit den gewählten Parametern und Wertebereichen möglich ist durchgetestet. Da jede Kombination gestestet wird, kann innerhalb des gewählten Parameterraums eine optimale Konfiguration gefunden werden. Hierbei kann es nach wie vor passieren, dass durch die Wahl der Schrittweite ein evtl. besseres Ergebnis übersprungen wird. Es ist auch möglich, dass eine bessere Parameterkonfiguration außerhalb des Parameterraumes liegt. Dieses wird mit der beschriebenen Methode ebenfalls nicht gefunden.

Um die Wertebereiche für jeden einzelnen Parameter zu wählen, wird jeder Parameter einzeln in einem großen Wertebereich abgetastet und es wird geprüft ab welchen Werten die Trackingqualität nicht mehr besser sondern nur noch schlechter wird. Mithilfe dieser Werte wird der Wertebereich bestimmt.

Als Gesamtmaß für die Qualität wird hierfür der MOTA-Wert genutzt. Dieser bietet kein absolutes Qualitätsmaß. Zum Vergleich verschiedener Durchläufe auf demselben Datensatz ist er jedoch geeignet.

5.5.1 Tracking

Die in diesem Abschnitt zu bestimmenden Grenzwerte dienen zur Prüfung der Stabilität der Vorhersage. Hierfür sind folgende Grenzwerte festzulegen:

Überlappungsverhältnis Maximalwert den das Überlappungsverhältnis (siehe Abschnitt 5.2.1) haben darf, um eine Assoziation zu bestätigen.

FB-Error Maximalwert des FB-Fehlers (siehe Abschnitt 5.3.2)

Merkmale Punktdistanz Der maximale relative Abstand eines Merkmalspunktes zum Mittelpunkt der Bounding-Box (siehe Abschnitt 5.3.1).

Minimale Anzahl Merkmalspunkte Die Mindestanzahl an Merkmalspunkten, wie in Abschnitt 5.3.1 beschrieben

Um die Werte für diese Parameter zu ermitteln wurde eine Testkonfiguration erstellt, in der für jeden Parameter ein festgelegter Wertebereich in bestimmten Teilschritten abgetastet wird. Zur Auswertung werden die genutzten Parameter zusammen mit der Evaluation, bestehend aus MOTA, MOTP, FPS (Frames-Per-Second), MT, ML und PT, in eine CSV-Datei geschrieben. Nachdem die Testreihen durchgelaufen sind kann in der CSV-Datei geprüft werden, welche Parameterkonfiguration zur Optimierung welches Wertes die beste ist.

Für die Parametertests wurde vom Kitti-Datensatz die Trainingssequenz Nr. 20 genutzt, da diese besonders viele Fahrzeuge auf engem Raum enthielt.

Im folgenden werden die Konfiguration für den Durchlauf mit dem besten Trackingergebniss (größter MOTA) und den mit der besten Performance (größte FPS) hier dargestellt.

Wertekonfiguration mit dem größten MOTA:

Wertbezeichner	**Wert**
Überlappungsverhältnis	0,3
maximaler Forward-Backward-Fehler	0,1
Maximale Punktdistanz	0,7
Minimale Anzahl Merkmalspunkte	6

Mit dieser Parameterkonfiguration lag der MOTA auf seinem Maximum von 0,611315. Die Performance lag im Durchschnitt bei 10 FPS.

Wertekonfiguration mit dem größten FPS:

Wertbezeichner	**Wert**
Überlappungsverhältnis	0,5
maximaler Forward-Backward-Fehler	0,8
Maximale Punktdistanz	0,1
Minimale Anzahl Merkmalspunkte	14

Mit diesen Parametern lag die durchschnittliche Verarbeitungsfrequenz bei 19,9 FPS, der MOTA lag jedoch nur bei 0,578073. Es ist damit praktisch möglich den Leistungsbedarf des Trackers um ~50% zu reduzieren was jedoch auf Kosten der Genauigkeit (um ~3 Prozentpunkte Verlust) geht.

Alle Parameterkonfigurationen mit Evaluationsresultaten können auf der Disk, welche dieser Arbeit beiliegt eingesehen werden. Die Datei befindet sich auf der Disk im Ordner „Resultate/Parametertests“.

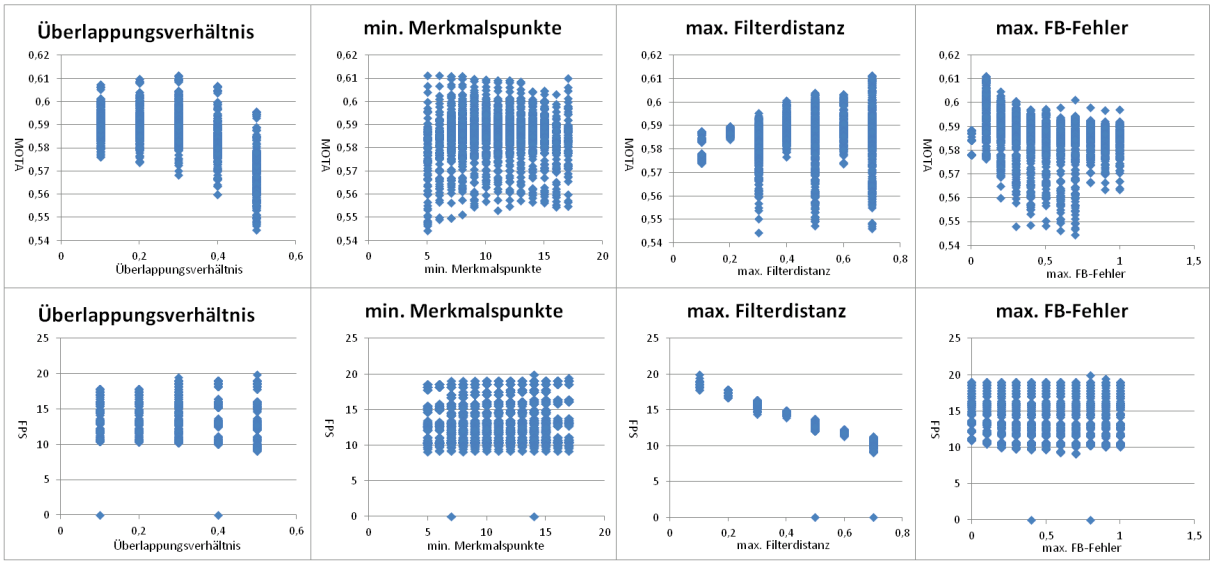

Abbildung 5.13: Diese Grafik zeigt alle Durchläufe der Parameteroptimierung, gruppiert nach den jeweiligen Parametern. In den oberen Diagrammen stellt die X-Achse den Schwellwert und die Y-Achse den MOTA-Wert für den jeweiligen Testdurchlauf dar. Die unteren Diagramme zeigen den Zusammenhang zwischen den Parametern und der Performance des Trackers. Die X-Achse stellt den Schwellwert und die Y-Achse die durchschnittliche Bearbeitungsrate in Bildern pro Sekunde dar. *Überlappungsverhältnis* ist das minimale Überlappungsverhältnis zur Assoziation von getrackten und detektieren Objekten (Abschnitt 5.2.1). *Min. Merkmalspunkte* ist die minimale Anzahl an Merkmalspunkten, welche der FAST-Algorithmus ergeben sollte, damit ein Objekt weiter verfolgt wird (Abschnitt 5.3.1). *Max. Filterdistanz* ist die maximale Filterdistanz (Abschnitt 5.3.1). *Max. FB-Fehler* ist der maximale FB-Fehler ab dem eine Vorhersage als instabil angesehen wird (Abschnitt 5.3.2).

Abbildung 5.13 stellt eine grafische Darstellung der Ergebnisse der Parameteroptimierung dar. Zu sehen ist, dass die minimalen Merkmalspunkte weder auf die Trackingqualität, noch auf die benötigte Rechenleisung einen Einfluss haben. Beim Überlappungsverhältnis ist bei der Trackingqualität eine Tendenz zu erkennen, dass die MOTA-Werte bei etwa 0,3 ihr Maximum erreichen und bei kleineren und größeren Werten abfallen. Auf den Rechenaufwand hat das Überlappungsverhältnis keinen Einfluss. Die maximale Filterdistanz hat einen großen Einfluss sowohl auf die benötigte Rechenleistung als auch auf die Trackingqualität. Hierbei ist zu erkennen, dass ein größerer Schwellwert die Trackingqualität verbessert, die Verarbeitungsrate jedoch verschlechtert.

Der Schwellwert für den FB-Fehler hat auf die benötigte Rechenleistung keinen Einfluss, auf die Trackingqualität wirkt er sich jedoch leicht aus.

Über alle Diagramme ist bei dem MOTA-Wert zu erkennen, dass die Schwankung nur sehr gering ist. Der Minimalwert liegt bei 54% und der Maximalwert bei 61%. Die Spanne dazwischen sind 7 Prozentpunkte. Das bedeutet innerhalb des gewählten Parameterraumes ist der Einfluss den die Parameter auf die Trackigqualität haben sehr gering. Weitere Parameteroptimierungen könnten dieses Ergebnis zwar verbessern, jedoch liegt der Schluss

nahe, dass der Einfluss eines größeren Wertebereichs für Parameter ebenso klein ausfallen würde, wie die bereits Getesteten. Zur weiteren Verbesserung der Trackingqualität sollte daher an diesem Punkt darauf gesetzt werden neue Verfahren zu testen.

Die Spanne bei der Verarbeitungsfrequenz ist wesentlich größer als beim MOTA-Wert. Hier ist der Minimalwert bei 10 FPS und der Maximalwert bei 20 FPS.

6 Auswertung

In diesem Kapitel wird der Tracking-Algorithmus, welcher im Rahmen dieser Arbeit entwickelt wurde, evaluiert. Wie getestet wird, ist in den jeweiligen Unterkapiteln beschrieben.

Sowohl beim KITTI-Benchmark, als auch bei der MOT-Challenge wird die Evaluierung im Rahmen dieses Kapitels gegen die Trainingsdatensätze durchgeführt, da nur diese Ground-Truth-Daten enthalten. Es kann nicht gegen die Testdatensätze evaluiert werden, da diese nicht über Ground-Truth-Daten verfügen. Zur Evaluierung wird die Statistik über alle Trainingssequenzen gebildet.

6.1 KITTI-Benchmark

Der KITTI-Datensatz wird mit dem DETMOT-Evaluator und mit dem Evaluator, welcher vom KITTI-Benchmark mitgeliefert wird, getestet.

Der KITTI-Trainingsdatensatz besteht aus 21 Trainingssequenzen, welche Sequenzen mit Stadtfahrten, Autobahnfahrten, Landstraßenfahrten und Fußgängerzonenfahrten enthalten. Einige dieser Sequenzen sind besser geeignet, das Tracking von Fußgängern zu evaluieren, andere enthalten mehr Fahrzeuge [11]. In den Evaluierungsdatensatz werden alle KITTI-Trainingssequenzen aufgenommen, mit Ausnahme von Sequenz 0020, da diese bereits bei der Parameteroptimierung genutzt wurde. Auf diese Weise soll einer Überanpassung vorgebeugt werden.

Dabei wird von den Autoren des Benchmarks angegeben, dass ausschließlich Autos und Fußgänger zur Evaluierung genutzt werden, da nur diese ausreichend gut in den Ground-Truth-Daten vorhanden sind [11].

Bei der Evaluation werden unterschiedliche Statistiken für das Tracking von Fußgängern und das Tracking von Autos erstellt.

6.1.1 Resultate KITTI-Benchmark-Evaluator

Im folgenden Abschnitt werden die Resultate des KITTI-Benchmarks angegeben. Die Werte tragen die Bezeichnung, welche vom Benchmark-Tool angegeben wird.

Resultate des eigenen Trackers

Resultate für Autos:

Bewertungsmaß	**Wert**
Multiple Object Tracking Accuracy (MOTA)	77,1 %
Multiple Object Tracking Precision (MOTP)	82,5 %
Multiple Object Tracking Accuracy (MOTAL)	77,4 %
Multiple Object Detection Accuracy (MODA)	77,4 %
Multiple Object Detection Precision (MODP)	87,6 %
Recall	86,6 %
Precision	93,4 %
F1	89,9 %
False Alarm Rate	0,189125
Mostly Tracked	63,2 %
Partly Tracked	31,4 %
Mostly Lost	4,7 %
ID-Switches	57
Anzahl GT-Elemente	24342

Resultate für Fußgänger:

Bewertungsmaß	**Wert**
Multiple Object Tracking Accuracy (MOTA)	54,2 %
Multiple Object Tracking Precision (MOTP)	77,0 %
Multiple Object Tracking Accuracy (MOTAL)	55,9 %
Multiple Object Detection Accuracy (MODA)	55,9 %
Multiple Object Detection Precision (MODP)	92,9 %
Recall	77,9 %
Precision	78,3 %
F1	78,1 %
False Alarm Rate	0,336810
Mostly Tracked	54,5 %
Partly Tracked	40,7 %
Mostly Lost	4,8 %
ID-Switches	185
Anzahl GT-Elemente	11470

Zum Vergleich: Ergebnisse des MDP-Verfahrens

Resultate für Autos:

Bewertungsmaß	**Wert**
Multiple Object Tracking Accuracy (MOTA)	79,0 %
Multiple Object Tracking Precision (MOTP)	81,0 %
Multiple Object Tracking Accuracy (MOTAL)	79,7 %
Multiple Object Detection Accuracy (MODA)	79,7 %
Multiple Object Detection Precision (MODP)	86,4 %
Recall	87,0 %
Precision	94,8 %
F1	90,7 %
False Alarm Rate	0,144764
Mostly Tracked	64,1 %
Partly Tracked	27,9 %
Mostly Lost	8,1 %
ID-Switches	132
Anzahl GT-Elemente	24342

Resultate für Fußgänger:

Bewertungsmaß	**Wert**
Multiple Object Tracking Accuracy (MOTA)	62,9 %
Multiple Object Tracking Precision (MOTP)	76,6 %
Multiple Object Tracking Accuracy (MOTAL)	63,7 %
Multiple Object Detection Accuracy (MODA)	63,7 %
Multiple Object Detection Precision (MODP)	92,7 %
Recall	74,6 %
Precision	87,6 %
F1	80,6 %
False Alarm Rate	0,165207
Mostly Tracked	53,9 %
Partly Tracked	35,9 %
Mostly Lost	10,2 %
ID-Switches	85
Anzahl GT-Elemente	11470

Auswertung

Bei Autos ist der Tracker, welcher im Rahmen dieser Arbeit entwickelt wurde, gleich auf mit dem MDP-Verfahren. Die Werte der CLEAR MOT-Statistik liegen etwa gleich auf. Die MT/PT/ML-Statistik des eigenen Trackers ist im *Mostly Tracked* leicht schlechter, im *Partly Tracked* und *Mostly Lost* jedoch leicht besser als der MDP-Algorithmus.

Bei Fußgängern sieht das Ergebnis signifikant anders aus. In den Werten der CLEAR MOT-Statistik hat der MDP-Algorithmus einen Vorsprung von 10 Prozentpunkten, wohingegen der eigene Tracker bei den Werten der MT/PT/ML-Statistik einen kleinen Vorsprung aufzuweisen hat, da weniger Ziele vollständig verloren wurden.

6.1.2 Resultate DETMOT-Evaluator

In diesem Abschnitt werden die Resultate des DETMOT-Evaluators auf dem KITTI-Trainingsdatensatz angegeben. Auch hier werden verschiedene Statistiken für Fußgänger und Autos angelegt.

Resultate des eigenen Trackers

Resultate für Autos:

Bewertungsmaß	**Wert**
MOTA	64,8 %
MOTP	80,8 %
ID-Switches	348
Tracks mit ID-Switch (Rel.)	34,8 %
False Alarm Rate	0,356873
Mostly Tracked	47,2 %
Mostly Lost	7,1 %

Resultate für Fußgänger:

Bewertungsmaß	**Wert**
MOTA	36,5 %
MOTP	71,4 %
ID-Switches	502
Tracks mit ID-Switch (Rel.)	61,1 %
False Alarm Rate	0,561460 %
Mostly Tracked	41,9 %
Mostly Lost	8,4 %

Zum Vergleich: Ergebnisse des MDP-Verfahrens

Das MDP-Verfahren nutzt eine eigene Methode, um die Detektionsergebnisse zu filtern. Dieser Filter wird im Rahmen dieser Arbeit nicht als Teil des Trackers, sondern als Teil des Detektors angesehen. Da die Detektionsergebnisse jedoch innerhalb des MDP-Trackers gefiltert wurden, bestand kein Zugriff auf die gefilterten Ergebnisse, was zur Folge hat, dass der DETMOT-Evaluator keine verlässlichen Detektionsergebnisse zur Evaluation des Trackers zur Verfügung stehen hat. Alle weiteren Werte, welche der DETMOT-Evaluator produziert werden daher nicht angegeben, da diese nicht verlässlich genug sind.

Resultate für Autos:

Bewertungsmaß	Wert
MOTA	64,2 %
MOTP	79,3 %
ID-Switches	349
Tracks mit ID-Switch (Rel.)	33,0 %
False Alarm Rate	0,380839
Mostly Tracked	48,1 %
Mostly Lost	9,6 %

Resultate für Fußgänger:

Bewertungsmaß	Wert
MOTA	44,0 %
MOTP	72,7 %
ID-Switches	313
Tracks mit ID-Switch (Rel.)	49,1 %
False Alarm Rate	0,418432 %
Mostly Tracked	41,9 %
Mostly Lost	9,0 %

Auswertung

Wie auch schon beim KITTI-Evaluator, zeigt sich, dass beide Algorithmen bei Autos gleichauf liegen und auch bei Fußgängern zeigt sich das gleiche Ergebnis, der MDP-Algorithmus hat etwa 10 Prozentpunkte Vorsprung.

Zusätzliche Auswertung des eigenen Trackers

In diesem Abschnitt wird noch einmal auf die Resultate des DETMOT-Evaluators eingegangen. Es werden erweiterte Statistiken dargestellt und es wird gezeigt, wo die Stärken und Schwächen des Algorithmus liegen.

Erweiterte Resultate für Autos:

Bewertungsmaß	**Wert**
MOTA	64,8 %
MOTA (Aktiv)	76,4 %
MOTP	80,8 %
ID-Switches	348
Tracks mit ID-Switch (Rel.)	34,8 %
False Alarm Rate	0.356873
Mostly Tracked	47,2 %
Mostly Lost	7,1 %
Mostly Tracked (Aktiv)	63,8 %
Mostly Lost (Aktiv)	2,6 %
Durchschnittliche aktive Abdeckung	82,1 %
Richtig vorhergesehen	808
Vorhergesehene Objekte	1019
Objekte die vorhergesehen werden müssen	4476
Falsch nicht vorhergesehen	2668
Falsch vorhergesehen	211
IDS zwischen Bildern	34,5 %
IDS nach Track-Verlust	65,5 %

Der Wert *MOTA (Aktiv)* zeigt an das unter Einbeziehung der Detektionsergebnisse 76% der Objekte richtig getracked wurden. Wenn diese Aktiv-Filterung auch auf die MT/PT/ML-Statistik angewendet wird, so werden 63% der Tracks größtenteils getracked in den Abschnitten in denen es dem Tracker möglich ist das Objekt zu tracken. Im Durchschnitt sind aktive Tracks zu 82% getracked, wie die *Durchschnittliche aktive Abdeckung* zeigt.

Die Resultate der Vorhersage von Objekten ohne Detektion sind verlässlich, 808 von insgesamt 1019 (79,3%) vorhergesagten Objekten sind korrekte Vorhersagen. Es werden jedoch 2668 der 4476 (59,6%) vorherzusagenden Objekte nicht vorhergesagt. Das bedeutet, dass vorhergesagte Objekte zwar weitgehend korrekt sind, jedoch in vielen Fällen nicht vorhergesagt wird. Es gibt sehr viele falsch Negative unter den Vorhersagen.

Von den ID-Switches treten 66% auf, nachdem das Objekt verschwunden war und 34% entstehen zwischen zwei Bildern, indem das Objekt fälschlicherweise verloren und als neues oder anderes, bereits bestehendes, Objekt wiedergefunden wird. Daraus ergibt sich, dass der Tracker bei Fahrzeugen Schwierigkeiten hat, verlorene Objekte mit ihrem vorherigen Track zu identifizieren.

Erweiterte Resultate für Fußgänger:

Bewertungsmaß	Wert
MOTA	36,5 %
MOTA (Aktiv)	40,7 %
MOTP	71,3 %
ID-Switches	502
Tracks mit ID-Switch (Rel.)	61,1 %
False Alarm Rate	0,561460
Mostly Tracked	41,9 %
Mostly Lost	8,4 %
Mostly Tracked (Aktiv)	49,4 %
Mostly Lost (Aktiv)	3,0 %
Durchschnittliche aktive Abdeckung	74,7 %
Richtig vorhergesehen	328
Vorhergesehene Objekte	391
Objekte die vorhergesehen werden müssen	1646
Falsch nicht vorhergesehen	1318
Falsch vorhergesehen	63
IDS zwischen Bildern	68,1 %
IDS nach Track-Verlust	31,9 %

Die Resultate für Fußgänger zeigen, dass der Objekttracker Probleme hat, Fußgänger zu verfolgen. Der MOTA-Wert mit Aktiv-Filterung liegt bei 41%. Das bedeutet, dass nur 41% der Objekte, die getrackt werden können, auch getrackt werden. Außerdem werden etwa 49% (siehe *Mostly Tracked (Aktiv)*) der aktiven Tracks größtenteils richtig getrackt. Aktive Tracks sind im Durchschnitt zu 75% richtig getrackt. Auch hier zeigt sich, dass bei der Vorhersage mithilfe des Medianflow-Trackers 328 von insgesamt 391 (83,9%) vorhergesagten Objekten stimmen, wohingegen nur 328 der 1646 (20%) vorherzusagenden Objekte vorhergesagt werden. Auch hier zeigt sich, wie bei den Autos, dass Vorhersagen, welche getroffen werden zwar größtenteils stimmen, der Tracker jedoch oft zu früh aufhört Objekte zu tracken und keine Vorhersage trifft.

68% der ID-Switches geschehen zwischen zwei Bildern und 32% nachdem ein Objekt verschwunden war. Dies legt nahe, dass die Assoziation der Objekte von einem Bild ins nächste, bei Fußgängern, nicht ausreichend gut funktioniert.

6.2 MOT-Challenge

Die MOT-Challenge von 2015 stellt 11 Trainingsdatensätze bereit, welche Gound-Truth-Daten enthalten. Die Ground-Truth-Daten enthalten ausschließlich Daten für Fußgänger [6].

Die Evaluaierung wird ausschließlich mit dem DETMOT-Evaluator durchgeführt, da es im Rahmen dieser Arbeit nicht möglich war, den Evaluator, welcher mitgeliefert wurde, auszuführen.

Des weiteren wird bei diesem Datensatz sowohl für den eigenen Tracker, als auch für den

Referenztracker (MDP-Verfahren) eine vollständige Analyse, mit Detektionsergebnissen durchgeführt.

Resultate des eigenen Trackers

Bewertungsmaß	**Wert**
Detektor Recall	38,6 %
Detektor Precision	84,8 %
Detektor Accuracy	31,7 %
MOTA	25,8 %
MOTA (Aktiv)	53,6 %
MOTP	68,5 %
ID-Switches	793
Tracks mit ID-Switch (Rel.)	32,7 %
False Alarm Rate	0,561273
Mostly Tracked	4,9 %
Mostly Lost	65,4 %
Mostly Tracked (Aktiv)	14,9 %
Mostly Lost (Aktiv)	65,4 %
Durchschnittliche aktive Abdeckung	45,3 %
Richtig vorhergesehen	339
Falsch nicht vorhergesehen	14412
Falsch vorhergesehen	5
IDS zwischen Bildern	31,4 %
IDS nach Track-Verlust	68,6 %

Zuerst eine kurze Auswertung des Detektors. 85% der vom Detektor erkannten Objekte sind korrekt. Es werden jedoch nur 39% der zu erkennenden Objekte korrekt detektiert. Die Genauigkeit des Detektors, das bedeutet das Verhältnis aller Detektionsfehler zu den Ground-Truth-Daten, beträgt 31,7%.

Die Detektionsergebnisse wurden bereits nach den Objekten gefiltert, welche mindestens eine Konfidenz von 30% haben.

Der *MOTA (Aktiv)*-Wert zeigt, dass 53.6% der Objekte, die getrackt werden können, auch getrackt werden. 32,7% der Tracks haben mindestens einen ID-Switch. Die Werte der MT/PT/ML-Statistik sehen hier sehr kritisch aus. Nur 5% der Tracks wurden größtenteils getrackt und 65,4% der Tracks werden gar nicht bis schlecht getrackt. Werden hier nur aktive Tracks betrachtet, zeigt sich keine signifikante Verbesserung, was darauf schließen lässt, dass die Fehlerursache zum größten Teil beim Tracker liegt.

339 der 344 (98,5%) Objekte, welche der Tracker vorhergesehen hat, sind korrekte Vorhersagen, wohingegen nur 339 der 14751 (2,3%) Objekte, welche der Tracker vorhersehen sollte auch tatsächlich vorhergesehen wurden. Hier zeigt sich derselbe Trend wie beim KITTI-Datensatz, auch wenn die Vorhersagen, welche vom Tracker vorgenommen wurden größtenteils stimmen, werden nur wenige der vorherzusagenden Objekte überhaupt

vorhergesagt.

ID-Switches treten zu 68.6% nach dem Verschwinden eines Objekts auf. Das bedeutet, dass es Probleme beim Wiederfinden von Verlorenen Objekten, besonders bei Fußgängern, gibt.

Zum Vergleich: Ergebnisse des MDP-Verfahrens

Zur vollständigen Evaluierung werden die rohen Detektionsergebnisse zur Evaluierung mit dem DETMOT-Evaluator herangezogen.

Bewertungsmaß	**Wert**
Detektor Recall	54,8 %
Detektor Precision	61,3 %
Detektor Accuracy	20,1 %
MOTA	33,8 %
MOTA (Aktiv)	47,0 %
MOTP	70,1 %
ID-Switches	422
Tracks mit ID-Switch (Rel.)	26,7 %
False Alarm Rate	1,388546
Mostly Tracked	17,9 %
Mostly Lost	42,2 %
Mostly Tracked (Aktiv)	33,8 %
Mostly Lost (Aktiv)	17,6 %
Durchschnittliche aktive Abdeckung	59,2 %
Richtig vorhergesehen	2312
Falsch nicht vorhergesehen	11620
Falsch vorhergesehen	193
IDS zwischen Bildern	41,2 %
IDS nach Track-Verlust	58,8 %

Da die Detektionsergebnisse hier, im Vergleich zur Evaluierung des eigenen Trackers, ungefiltert evaluiert wurden, ist dessen Präzision niedriger, da mehr falsch Positive ausgewertet wurden. Die Empfindlichkeit (Recall) ist jedoch niedriger, da weniger richtige Ergebnisse mit geringer Konfidenz herausgefiltert werden und es so zu mehr falsch Positiven kommt.

Der MOTA des MDP-Verfahrens liegt bei 33,8% was ein besseres Ergebnis darstellt, als die 25,8% des eigenen Trackers. Werden jedoch nur die aktiven Elemente evaluiert, liegt der MDP-Algorithmus bei 47% im Vergleich zu 53.6% des eigenen Trackers. Das bedeutet, dass der MDP-Algorithmus zwar mehr Fehler macht als der eigene Tracker, jedoch wesentlich besser darin ist, Fehler des Detektors herauszufiltern.

18% der Tracks werden größtenteils getrackt, 42% werden größtenteils verloren. Auch hier zeigt sich, dass der MDP-Tracker einen Vorsprung vor dem eigenen Tracker hat.

Mit der Vorhersage von Objekten ohne Detektionsergebnisse hat auch der MDP-Tracker Probleme. 2312 der 2505 (92,3%) der vorhergesehenen Objekte sind korrekt, während

2312 der 13932 (16,6%) vorherzusehenden Objekte vorhergesehen werden. Damit ist der MDP-Algorithmus, mit 16,6% korrekt Vorhergesagten Objekten, dem eigenen Tracker, mit 2,6% reichlich überlegen, was die Vorhersage von Fußgängern angeht.

Der MDP-Algorithmus hat die meisten ID-Switches mit 58,8% nach einem Trackingverlust.

6.3 Fazit

Die Evaluierung des Trackers hat gezeigt wo dessen Stärken und Schwächen liegen. Beim Tracken von Autos kann es der eigene Tracker mit dem Referenztracker aufnehmen und erreicht ähnliche Ergebnisse. Hierzu ist zu erwähnen, dass der MDP-Tracker nicht der beste Trackingalgorithmus ist, welcher auf der Website des KITTI-Benchmarks genannt wird [11]. Er ist der einzige, für den Quellcode bereitgestellt wurde und der somit unter den gleichen Bedingungen getestet werden konnte wie der eigene Tracker. Dies ist notwendig, da sowohl der KITTI-Benchmark, als auch die MOT-Challenge keine Ergebnisse für die Trainingsdatensätze bereitstellen. Ohne diese ist es notwendig den Tracker selbstständig ausführen und testen zu können, was nur beim MDP-Algorithmus möglich war.

Die Evaluierung von Fußgängern (sowohl im KITTI-Benchmark als auch in der MOT-Challenge) hat gezeigt, dass der eigene Tracker noch erhebliche Probleme mit dem Tracken von Fußgängern hat. Der Grund hierfür ist eventuell, dass sich Autos zwar bewegen, sich ihre Form jedoch nicht in so großem Maße verändert. Ein Fußgänger ist dagegen von starken Formveränderungen betroffen, während er läuft. Dies führt zu einem wesentlich instabileren optischen Fluss und somit zu einem verfrühten Abbruch des Trackings, sofern keine Detektionsergebnisse vorliegen.

Hinzu kommt, dass bei der MOT-Challenge von 2015 die mitgelieferten Detektionsergebnisse von sehr schlechter Qualität waren, was sich sowohl auf den eigenen, als auch auf den MDP-Tracker auswirkt. Beide lieferten sehr schlechte Trackingergebnisse.

7 Zusammenfassung und Ausblick

In diesem Kapitel werden kurz die Ergebnisse der Arbeit zusammengefasst. Es wird darauf eingegangen, inwiefern die Ziele der Arbeit erreicht wurden und es wird einen Ausblick geben, darüber wie die Thematik weitergeführt werden könnte.

7.1 Zusammenfassung

Das Ziel der Arbeit war es, einen Objekttracker zu entwickeln, welcher auf der Grundlage der Bild- und Detektionsdaten Objekte trackt.

Hierfür wurde eine Schnittstelle entwickelt, über die beliebige Tracker angebunden werden konnten. Diese Anbindung betreibt die Tracker online. Die Schnittstelle des Trackers ist so konzipiert, dass er Tracker leicht an die Datenquellen des Nova-Frameworks angebunden werden kann. Dadurch ist es einfach auf verschiedene Datenquellen zurückzugreifen.

In Kapitel 5 wurde dargelegt welche Tests und statistischen Auswertung durchgeführt wurden, um den Tracker zu entwickeln. Es wurde auch darauf eingegangen, wie der DETMOT-Evaluator, welcher eine detaillierte Auswertung der Trackingdaten ermöglichen sollte, seine statistischen Werte berechnet und was diese Werte über den Tracker aussagen. Der Evaluator hat außerdem dabei geholfen den Fokus der Entwicklungsarbeit zu lenken. So konnte mit dem Evaluator festgestellt werden, dass die größten Qualitätsverluste beim Tracking nicht an der korrekten Zuordnung verlorener Objekte liegt, sondern an der mangelhaften Vorhersage schlecht detektierter Objekte. In Kapitel 6 wurde dargestellt, dass eben diese Funktion nicht nur für den eigenen Tracker eine Herausforderung darstellt, sondern auch für den Referenztracker, das MDP-Verfahren.

Wie im Kapitel 6 dargestellt, reicht die Qualität des Trackers bei Fahrzeugen an die Qualität bestehender Lösungen heran. Bei Fußgängern liegt der Tracker jedoch hinter bestehenden Lösungen.

7.2 Ausblick

Eine weitere Arbeit an dem Objekttracker sollte sich dem Verbessern der Trackingergebnisse widmen. Hierfür gibt es verschiedene Ansätze, wie z. B. den Einsatz von künstlicher Intelligenz (Künstliche neuronale Netze) oder das Einbinden zusätzlicher Sensorinformationen wie z. B. Stereokameraaufnahmen um 3D-Informationen zu gewinnen.

Eine weitere Möglichkeit den Tracker zu verbessern ist, eine Verbesserung der Detektionsergebnisse. Dies könnte erreicht werden, indem z. B. der Tracker die Filterung der Detektionsergebnisse übernimmt und zusätzliche Informationen in die Entscheidung, ob es sich bei einem Objekt um ein falsch Positives oder ein tatsächlich existierendes Objekt handelt, mit einbeziehen. Hierbei besteht auch die Möglichkeit das Detektieren und das Tracking besser miteinander zu vernetzen, das bedeutet einen Detektor zu entwickeln,

welcher z. B. Zusatzdaten zur Identifizierung von Objekten generiert. Durch eine Rückführung der Detektionsergebnisse eines neuronalen Netzes in das Netz, könnte es sogar möglich sein, auch das Tracking im Detektor auszuführen.

Die wichtigste Komponente hierbei ist das Tracking von Objekte, auch wenn die Detektionsergebnisse ausbleiben. Der Medianflow-Tracker, welcher in dieser Arbeit genutzt wird, löst diese Aufgabe (wie in Kapitel 6 zu sehen ist) nicht ausreichend gut. Das Tracking von Objekten, welche ihre Form nicht verändern, wie Autos, sind die Ergebnisse noch gut, bei Objekten die ihre Form jedoch verändern, wie Fußgänger, ist das Tracking ohne Detektionsergebnisse nur schwer möglich. Daher wäre es hier notwendig mit anderen Ansätzen zu experimentieren. Eventuell ist es bereits ausreichend den Integritätscheck der Vorhersage zu verbessern. Das bedeutet besser festzustellen, ob ein Objekt weiter getrackt werden soll oder ob dies nicht möglich ist.

Akronyme

ADTF Automotive Data and Time-Triggered Framework. 3

ALFD Aggregated Local Flow Descriptor. 14

CIWT Combined Image- and World-Space Tracking. II, 16

CRF Conditional Random Field. 15, 16

FN False Negatives. 10

FP False Positives. 10

IDS ID-Switches. 10

IPT Interest Point Trajectory. 14, 15

KNN K-Nearest Neighbour. 41

MDP Markov Decision Process. II, 13, 14, 24, 80

ML Mostly Lost. 10, 11

MOT Multi Object Tracking. 1, 18

MOTA Multiple Object Tracking Accuracy. 10, 47, 52

MOTP Multiple Object Tracking Precision. 10, 52

MT Mostly Tracked. 10, 11

NOMT Near Online Multi-target Tracking. II, 14

PT Partly Tracked. 10, 11

ROS Robot Operating System. 3

SIFT Scale-invariant feature transform. 41

SURF Speeded Up Robust Features. 41

TP True Positives. 10

Glossar

Automotive Data and Time-Triggered Framework Ein Softwareframework zur dynamischen Datenstromverarbeitung von Audi und Elektrobit [26]. 75

Boundingbox Eine Boundingbox beschreibt den Rahmen mit dem die Position, sowie die Ausmaße eines Objekts angegeben werden. Sie besteht aus den rechten, linken, oberen und unteren Rand. 22, 24

False Negatives Beschreibt Objekte, welche in der Ground-Truth enthalten sind, vom Detektor jedoch nicht erkannt wurden. 10, 75

False Positives Beschreibt Objekte die vom Detektor erkannt wurden, jedoch nicht inder Ground-Truth enthalten sind. 10, 75

FLANN Eine sehr effiziente Implementierung einer KNN (k neares neighbour) Zuweisung. Mehr Infos hierzu sind in dem entwsprechenden Paper zu finden [21]. 41

Frame Ein Frame beschreibt einen einzelnen Zeitschrit, welcher mit der Aufnahme eines Bildes beginnt. II, 18, 22, 25, 26

Ground-Truth Kombination aus Video und Labeldaten, welche als Grundlage zur Bewertung oder zum Trainieren dienen. Ground-Truth-Daten zeichnen sich durch eine sehr hohe Genauigkeit aus. 21

ID-Switches Ein ID-Switch liegt vor, wenn ein Objekt in seiner Trajektorie die ID wechselt, also vom Tracker fälschlicherweise als neues Objekt erkannt wurde. 10, 75

Konfidenz Gibt an, wie sicher sich ein Detektor oder ein Tracker über die Richtigkeit eines Labels ist. Der Wertebereich kann je nach Verfahren schwanken. 22, 24

Markov Decision Process Methode um Sachverhalte zu modellieren, die von zufälligen Ereignissen abhängen [10]. 13, 14, 24, 75, 80

Multi Object Tracking Das verfolgen von mehreren Objekten gleichzeitig in einem Videodatenstrom. 75

Robot Operating System Ein Softwareframework welches einen Kommunikationsbus, sowie verschiedenste Softwarepakete zur Verfügung stellt um Software für die Robotik zu entwickeln [27]. 75

Track Ein Track ist eine Menge an Detektionsergebnissen, welche zu dem selben Objekt gehören und damit die selbe Identifikationnummer haben. 10, 11

True Positives Beschreibt Objekte, welche in der Ground-Truth enthalten sind und vom Detektor erkannt wurden. 75

Literaturverzeichnis

[1] Kleinhenz, Philipp: *Detektion und Klassifikation von Objekten im Fahrbahnbereich mit Verfahren des maschinellen Lernens*. Bachelorarbeit, Hochschule für Technik, Wirtschaft und Kultur Leipzig, Okt. 2016.

[2] Xiang, Yu, Changkyu Song, Roozbeh Mottaghi und Silvio Savarese: *Monocular Multiview Object Tracking with 3D Aspect Parts*. In: *European Conference on Computer Vision (ECCV)*, 2014.

[3] Bernardin, Keni und Rainer Stiefelhagen: *Evaluating Multiple Object Tracking Performance: The CLEAR MOT Metrics*. EURASIP Journal on Image and Video Processing, 2008(1):246309, May 2008, ISSN 1687-5281. `https://doi.org/10.1155/2008/246309`.

[4] Li, Yuan, Chang Huang und Ram Nevatia: *Learning to associate: Hybridboosted multi-target tracker for crowded scene*. In: *In CVPR*, 2009.

[5] Geiger, Andreas, Philip Lenz und Raquel Urtasun: *Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite*. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[6] Leal-Taixé, L., A. Milan, I. Reid, S. Roth und K. Schindler: *MOTChallenge 2015: Towards a Benchmark for Multi-Target Tracking*. arXiv:1504.01942 [cs], April 2015. `http://arxiv.org/abs/1504.01942`, arXiv: 1504.01942.

[7] *OpenCV library*. `https://opencv.org/`, besucht: 01.02.2018.

[8] Kalal, Zdenek, Krystian Mikolajczyk und Jiri Matas: *Forward-Backward Error: Automatic Detection of Tracking Failures*. In: *International Conference on Pattern Recognition*, Aug 2010.

[9] Lucas, Bruce D. und Takeo Kanade: *An Iterative Image Registration Technique with an Application to Stereo Vision*. In: *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'81, Seiten 674–679, San Francisco, CA, USA, 1981. Morgan Kaufmann Publishers Inc. `http://dl.acm.org/citation.cfm?id=1623264.1623280`.

[10] Xiang, Yu, Alexandre Alahi und Silvio Savarese: *Learning to Track: Online Multi-Object Tracking by Decision Making*. In: *International Conference on Computer Vision (ICCV)*, 2015.

[11] Geiger, Andreas, Philip Lenz und Raquel Urtasun: *Object Tracking Evaluation 2012*. `http://www.cvlibs.net/datasets/kitti/eval_tracking.php`, besucht: 19.01.2018.

[12] Bellman, Richard: *A Markovian Decision Process*. Indiana Univ. Math. J., 6:679–684, 1957, ISSN 0022-2518.

[13] Choi, Wongun: *Near-Online Multi-target Tracking with Aggregated Local Flow Descriptor*. CoRR, abs/1504.02340, 2015. http://arxiv.org/abs/1504.02340.

[14] Rosten, E. und T. Drummond: *Fusing points and lines for high performance tracking*. In: *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, Band 2, Seiten 1508–1515 Vol. 2, Oct 2005.

[15] Farneback, G.: *Very high accuracy velocity estimation using orientation tensors, parametric motion, and simultaneous segmentation of the motion field*. In: *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, Band 1, Seiten 171–177 vol.1, 2001.

[16] Lafferty, John D., Andrew McCallum und Fernando C. N. Pereira: *Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data*. In: *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, Seiten 282–289, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc., ISBN 1-55860-778-1. http://dl.acm.org/citation.cfm?id=645530.655813.

[17] Osep, Aljoša, Wolfgang Mehner, Markus Mathias und Bastian Leibe: *Combined image-and world-space tracking in traffic scenes*. In: *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, Seiten 1988–1995. IEEE, 2017.

[18] Lee, Byungjae, Enkhbayar Erdenee, Songguo Jin, Mi Young Nam, Young Giu Jung und Phill Kyu Rhee: *Multi-class Multi-object Tracking Using Changing Point Detection*, Seiten 68–83. Springer International Publishing, Cham, 2016, ISBN 978-3-319-48881-3. https://doi.org/10.1007/978-3-319-48881-3_6.

[19] Lowe, D. G.: *Object recognition from local scale-invariant features*. In: *Proceedings of the Seventh IEEE International Conference on Computer Vision*, Band 2, Seiten 1150–1157 vol.2, 1999.

[20] Bay, Herbert, Tinne Tuytelaars und Luc Van Gool: *SURF: Speeded Up Robust Features*, Seiten 404–417. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, ISBN 978-3-540-33833-8. https://doi.org/10.1007/11744023_32.

[21] Muja, Marius und David G. Lowe: *Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration*. In: *International Conference on Computer Vision Theory and Application VISSAPP'09)*, Seiten 331–340. INSTICC Press, 2009.

[22] Muja, Marius und David G. Lowe: *Scalable Nearest Neighbor Algorithms for High Dimensional Data*. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 36, 2014.

[23] Muja, Marius und David G. Lowe: *Fast Matching of Binary Features*. In: *Computer and Robot Vision (CRV)*, Seiten 404–410, 2012.

[24] Lowe, David G.: *Distinctive Image Features from Scale-Invariant Keypoints*. International Journal of Computer Vision, 60(2):91–110, Nov 2004, ISSN 1573-1405. https://doi.org/10.1023/B:VISI.0000029664.99615.94.

[25] Munkres, James: *Algorithms for the Assignment and Transportation Problems.* 5, März 1957.

[26] GmbH, Elektrobit Automotive: *EB Assist ADTF - Elektrobit.* `https://www.elektrobit.com/products/eb-assist/adtf/`, besucht: 26.10.2017.

[27] `http://www.ros.org/`, besucht: 26.10.2017.

Abbildungsverzeichnis

Quelltextverzeichnis

Anhang

```
1  ####################################
2  ++++++++++++Detector++++++++++++
3   TruePositives ..........      3306
4   FalsePositives .........      1740
5   FalseNegatices .........      1555
6   GroundTruth ............      7294
7   GroundTruth Acccepted ..      4861
8   False Alarms per Frame.  2.081340
9   Recall .................  0.680107
10  Accuracy ...............  0.548259
11  Precision ..............  0.655172
12  BB-Precision ...........  0.849281
13  ++++++++++++Tracker++++++++++++
14  TruePositive ...........      3105
15  FalsePositive ..........       827
16  FalseNegative ..........      1733
17  IDSwitches .............        23
18  GroundTruth ............      7294
19  GroundTruth Accepted ...      4861
20  False Alarms per Frame.  0.989234
21  Active False Negative ..       216
22  ++++++++++++CLEAR MOT++++++++++++
23  ----------Not Corrected----------
24  MOTA....................  0.468628
25  Precision ..............  0.638757
26  MOTP....................  0.851429
27  Accuracy ...............  0.473359
28  ----------Corrected----------
29  Inactivated FN .........      1124
30  Detector FP ............      1740
31  Detector FN ............      1555
32  ActiveDetectedFN .......       229
33  Not Detected, Active ...       380
34  Accuracy ...............  0.830880
35  MOTA(Cleanded) .........  0.799973
36  ++++++++++Predictions++++++++++
37  Predicted Objects ......        51
38  Correctly Predicted ....        51
39  Falsely Predicted ......         0
40  Falsely Not Predicted ..       380
41  ----------Relative----------
42  Predicted Objects ......  0.013655
43  Correctly Predicted ....  1.000000
44  Falsely Predicted ......  0.000000
45  Predicted Recall .......  0.118329
46  Predicted Precision ....  1.000000
47  ++++++++++ID-Switches++++++++++
48  IDS ....................        23
49  After Lost .............        21
50  Between Frames .........         2
51  Tracks with Lost .......        13
52  Tracks with Lost Rel ...  0.116071
53  IDS Rel ................  0.004732
```

```
54 Tracks With IDS ..... 0.196429
55 ---------IDS-Analysis (Rel)---------
56 After being lost ..... 0.913043
57 Between Frames ..... 0.086957
58 Properly Recovered ..... 76
59 Stolen Tracks(Lost) .... 5
60 Falsely seen as new .... 16
61 Stolen Tracks(Frames) .. 1
62 ----------Between Frames----------
63 Stolen ..... 0.043478
64 New ..... 0.043478
65 ----------After Lost----------
66 Stolen ..... 0.217391
67 New ..... 0.695652
68 --------Reappear no Track---------
69 Stolen ..... 0
70 New ..... 0
71 ++++++++++MT/PT/ML++++++++++++
72 ----------Not Corrected----------
73 Mostly Tracked ..... 23
74 Mostly Lost ..... 31
75 All Tracks ..... 112
76 ----------Relative----------
77 Mostly Tracked ..... 0.205357
78 Mostly Lost ..... 0.276786
79 ----------Corrected----------
80 Mostly Tracked ..... 42
81 Mostly Lost ..... 10
82 All Tracks ..... 97
83 ----------Relative----------
84 Mostly Tracked ..... 0.432990
85 Mostly Lost ..... 0.103093
86 Activated Tracks ..... 0.866071
87 ----------Coverage----------
88 Averge Coverage ..... 0.493407
89 Average Active Coverage 0.657346
90 ----------Weighted----------
91 Average Active Coverage 0.831325
92 ####################################
```

Quelltext 7.1: Beispielbericht des DETMOT-Evaluators