TUHH
Technische Universität Hamburg-Harburg

TUHH STS
Software
Technology
Systems

Über die Kopplung von Bildverarbeitung und Bildverstehen

erstellt von
Andreas Jungbluth
Matr.Nr.: 20522852

betreut durch
Prof. Dr. Ralf Möller
Anahita Nafissi

Technische Universität Hamburg-Harburg
Institut für Softwaresysteme (STS)

Ehrenwörtliche Erklärung

Ich erkläre hiermit ehrenwörtlich, dass ich die vorliegende Arbeit selbständig angefertigt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde weder einer anderen öffentlichen oder privaten Institution vorgelegt noch veröffentlicht.

Ort, Datum und Unterschrift

Inhaltsverzeichnis

Abbildungsverzeichnis

1. Einführung

Der Informationsumfang im „World Wide Web“ nimmt immer mehr zu. Da viele Daten uneinheitlich gespeichert werden, ist es nur dem Menschen möglich, diese in vollem Umfang zu erfassen. Eine gezielte Suche nach bestimmtem Informationen ist somit nur sehr schwer möglich.

Um dieses wachsende Problem zu lösen, soll das WWW um das „Semantic Web“ erweitert werden. Ziel ist es, alle Informationen, die bisher nur dem Menschen zugänglich sind, auch für Maschinen verständlich zu machen. So sollen Computer die Bedeutung von Texten verstehen, Bildinhalte erkennen etc.

Die Vorteile dieser automatischen Wissensextraktion beschränken sich dabei nicht nur auf die Informationssuche. Genau wie dem Menschen ist es den Maschinen möglich, anhand vorhandener Informationen Rückschlüsse zu ziehen und dadurch neues Wissen zu generieren. Durch derart neu gewonnene Assoziationen kann der Benutzer z.B. beim virtuellen Einkauf optimal mit Informationen versorgt werden.

1.1. Motivation

Der Informationsfluss im „Semantic Web“ geht bisher nur in eine Richtung. Auf der untersten Ebene werden aus Multimediadaten wie Bildern oder Texten Informationen extrahiert. Diese werden an die nächste Ebene weitergereicht, auf der mittels logischer Rückschlüsse eine Aussage über diese Informationen generiert wird.

Der Nachteil dieses Vorgehens liegt darin begründet, dass die finale Aussage nur so gut sein kann, wie die Eingabedaten, die auf der untersten Ebene gewonnen werden. Ein

beispielweise auf einem Bild falsch erkanntes Objekt kann somit zu einer vollständig falschen Interpretation führen. Vor allem die Ergebnisse der Bilderkennung produzieren derzeit noch viele Fehler in ihrer Klassifizierung.

Genau an diesem Problempunkt setzt das Prinzip der Kopplung von Bilderverarbeitung und Bildverstehen an. Neben dem bisherigen einseitigen Informationsfluss wird ein Feedback von der Logikebene zurück zur Bilderkennung modelliert. Ziel dieses Aufbaus ist es, die Prüfverfahren der Logikebene zu nutzen, um eventuelle Fehldetektionen in dem von der Bilderkennung übermittelten Datensatz zu erfassen. Im Falle eines so ermittelten Widerspruchs soll es der Bilderkennung möglich sein, ihr Ergebnis entsprechend zu verbessern.

1.2. Ziel der Arbeit

Das Ziel dieser Studienarbeit ist es, die Auswirkungen einer derartigen Kopplung anhand einer konkreten Implementierung zu untersuchen. Dazu werden zunächst in Kapitel 2 die notwendigen Grundlagen im Bereich der Wissenrepräsentation erläutert. In Kapitel 3 erfolgt dann eine Beschreibung der im Projekt verwendeten Softwarekomponenten. Die konkrete Umsetzung des Projektes wird in Kapitel 4 beschrieben. Abschließend erfolgt in Kapitel 5 die Bewertung der implementierten Funktionen anhand einer Datenerhebung.

2. Theoretische Grundlagen

2.1. Beschreibungslogik

Die im „Semantic Web“ geforderte Modellierung und Auswertung von Informationen wird durch die Beschreibungslogik realisert. Dabei handelt es sich um einen Formalismus zur Repräsentation von Wissen in einer bestimmten Domäne, indem zunächst die relevanten Konzepte in dieser Domäne definiert und anhand dieser Konzepte die Eigenschaften von Objekten und Individuen innerhalb der Domäne spezifiziert werden (vgl. [BCM+03]).

2.1.1. Syntax

Es werden folglich zwei Typen von Informationen unterschieden. Zunächst existieren die grundlegenden Konzepte, die die Terminologie der Domäne beschreiben und deshalb in der TBox (Terminological Box) zusammengefasst werden. Neue Konzepte werden dabei in Abhängigkeit von bereits existierenden definiert, so dass eine logische Verknüpfung entsteht. Diese Verknüpfung kann entweder durch eine Äuqivalenz

$$Woman \equiv Person \sqcap Female$$

oder in schwächerer Form durch eine Inklusion erfolgen. Jede Äquivalenz lässt sich dabei durch zwei Inklusionen darstellen:

$$Woman \sqsubseteq Person \sqcap Female$$
$$Person \sqcap Female \sqsubseteq Woman$$

Der zweite Typ von Informationen stellt Aussagen über bestimme Individuen dar und wird deshalb in der ABox (Assertional Box) zusammengefasst. Dabei werden neben der Klassenzugehörigkeit eines Individuums

$$Female(alice)$$

auch mehrstellige Relationen zwischen diesen modelliert:

$$Married(alice, bob)$$

Die grundlegenden Operatoren zur Verknüpfung mehrerer Ausdrücke sind neben der Konjunktion ($\sqcap$) und Disjunktion ($\sqcup$) auch der Existenz- ($\exists$) und Allquantor ($\forall$) sowie die Negation ($\neg$). Diese Operatoren werden in einem Standard namens ALC (Attributive Language with Complements) zusammengefasst. Wird zusätzlich noch die Benutzung von Kardinalitäten erlaubt, so spricht man von ALCQ. Darauf basierend gibt es noch eine Reihe weiterer Ergänzungen der beschreibungslogischen Syntax (z.B. SHIQ), wobei der Umfang von ALCQ für dieses Projekt ausreichend ist.

2.1.2. Reasoning

Der wesentliche Unterschied der Beschreibungslogik im Vergleich zu anderen Wissensrepräsentationen ist ihre Entscheidbarkeit, da sie semantisch eindeutig ist. Hierdurch ergibt sich die Möglichkeit Schlussfolgerungen zu ziehen. Bei dem als *Reasoning* (bzw. Inferenz) bekannten Verfahren werden die explizit modellierten Daten genutzt, um implizit vorhandene Informationen zu erhalten. Am Beispiel der folgenden TBox wird dies verdeutlicht:

$$Woman \sqsubseteq Person$$

$$Person \sqsubseteq Being$$

Es ist sofort ersichtlich, dass „Woman“ ebenfalls eine Unterklasse von „Being“ ist.

Das beschriebene Beispiel stellt nur einen möglichen Fall dar, wie aus vorhandenen Konzepten neues Wissen erlangt werden kann. Einige dieser als „Inferenzprobleme“ bekannten Fragestellungen sind u.a.:

- Konsistenz - stellt der Inhalt der ABox einen Widerspruch zur TBox dar?
- Subsumption - ist ein Konzept „D“ eine Verallgemeinerung eines anderen Konzeptes „C“ (d.h. gilt $C \sqsubseteq D$)?
- Äquivalenz - sind zwei Konzepte identisch (d.h. gilt $C \equiv D$)?
- Instanz - ist ein Individuum „a“ eine Instanz eines Konzeptes „C“ (d.h. gilt C(a))?

Um deratige Rückschlüsse auch bei umfangreicheren Domänen zu erzielen, gibt es entsprechende Algorithmen, wie z.B. den Tableau-Algorithmus. Aufgrund der Komplexität der Beschreibungslogik wächst die Laufzeit der Algorithmen jedoch exponentiell mit der Anzahl der Einträge in der T- und ABox.

3. Analyse

Die grundlegenden Komponenten des Projektes sind in Abbildung 3.1 dargestellt. Auf der Ebene der Informationsextraktion arbeiten sowohl ein Programm zur Bilderkennung (*Object-Detector*, s. Kap. 3.1) als auch eines zur Texterkennung (*Text-Detector*). Dabei wird der *Text-Detector* nicht explizit modelliert, sondern lediglich als externer Input aufgefasst, dem manuell Informationen dargereicht werden.

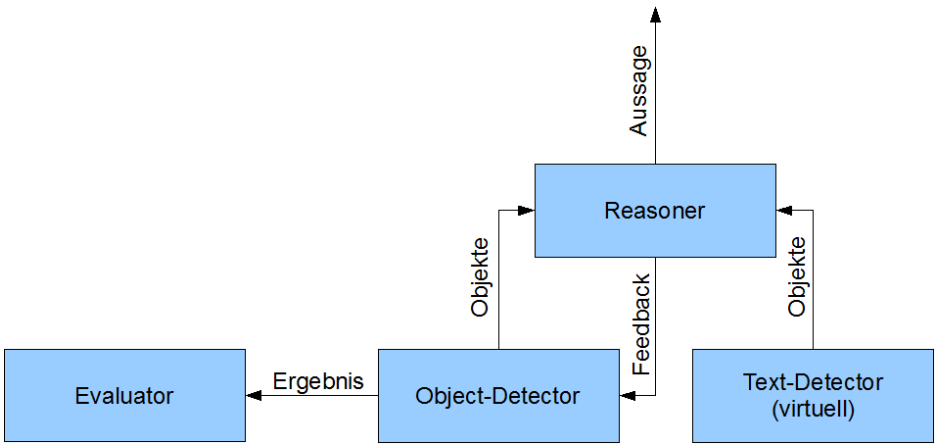

Abbildung 3.1.: Versuchsaufbau

Informationen, die auf diese Weise extrahiert werden, werden auf die nächste Ebene weitergeleitet, auf der ein Progamm zur logischen Auswertung (*Reasoner*, s. Kap. 3.2) angesiedelt ist, welches auch das Feedback zum *Object-Detector* ausgibt.

Zur Auswertung werden die Ergebnisse des *Object-Detectors* an einen *Evaluator* (s. Kap. 3.3) weitergegeben. Dessen Aufgabe ist es, nach der Durchführung des Experimentes eine objektive Aussage darüber zu treffen, ob und in welchem Umfang das Ergebnis verbessert werden konnte.

3.1. OpenCV

Zur Implementierung des *Object-Detectors* wird OpenCV (Open Computer Vision) eingesetzt. Dabei handelt es sich um eine freie, von Intel zur Verfügung gestellte Bibliothek, die vorgefertigte Komponenten zur Bildverarbeitung enthält.

OpenCV wurde ursprünglich im Hinblick auf Recheneffizienz und dem dazugehörigen Fokus auf Echtzeitanwendungen entwickelt und bietet daher nicht zuletzt dank seiner Multi-Core Unterstützung eine sehr gute Performance. Eines der Hauptziele von OpenCV ist es, dem Anwender zu ermöglichen, sehr komplexe Applikationen in möglichst kurzer Zeit umzusetzen. Dafür stehen über 500 fertige Funktionen in Bereichen wie z.B. Kamerakalibrierung, User-Interfaces und Robotik zur Verfügung (vgl. [GB08]).

3.1.1. Haar-Features

Die Objekterkennung in OpenCV arbeitet nach dem Prinzip der sogenannten *Haar-Features*. Dabei handelt es sich um definierte geometrische Formen, welche ihren Namen aufgrund der Ähnlichkeit zu *Haar-Wavelets* aus der Signalverarbeitung haben.

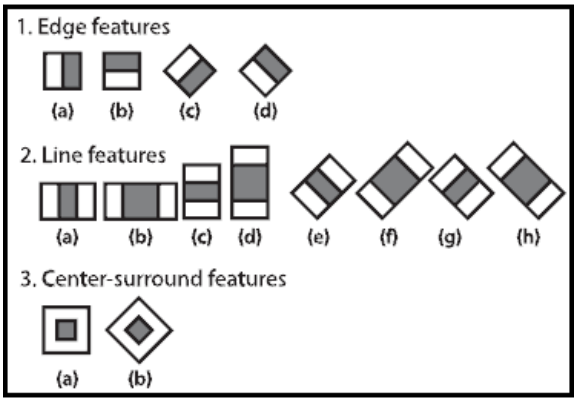

Abbildung 3.2.: *Haar-Features* in OpenCV [GB08, S. 509]

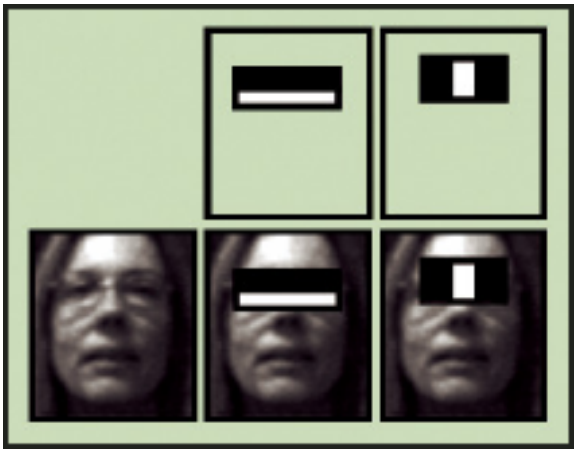

Abbildung 3.3.: *Haar-Features* in der Gesichtserkennung [Hew]

Zur Objekterkennung besitzt jedes dieser Features zwei unterschiedliche Zonen, welche in Abbildung 3.2 durch die weißen und grauen Flächen dargestellt sind. In der Anwendung wird jedes einzelne dieser Features in der Größe skaliert und über sämtliche Abschnitte des zu untersuchenden Bildes gelegt (vgl. Abb. 3.3). Dabei wird jeweils die Differenz zwischen den durchschnittlichen Pixelwerten unter der grauen und weißen Region gebildet. Liegt diese Differenz über einem, im vorangegangenen Schritt des Trainings, festgelegten Schwellenwert, so wird dieses Feature als vorhanden erkannt.

Ursprünglich wurde dieses Verfahren für die Gesichtserkennung entwickelt, lässt sich jedoch auf nahezu jedes starre Objekt (wie z.B. Autos oder Körper) übertragen.

Da bei der Anwendung aller möglichen *Haar-Features* auf einem Bild bis zu über einer Million Möglichkeiten durchprobiert werden können, wird ein Verfahren genutzt, um den Prozess der Entscheidungsfindung zu beschleunigen. Dazu werden auftrainierte *Haar-Features* jeweils in Knoten angeordnet, welche wiederum hintereinander geschaltet werden und somit eine Kette bilden (vgl. Abb. 3.4).

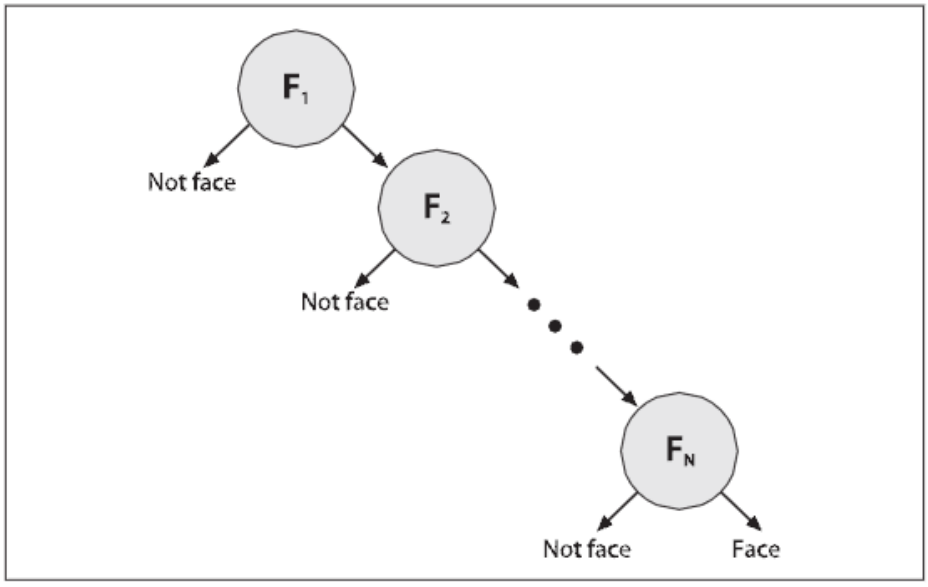

Abbildung 3.4.: Mehrstufige Verkettung von Knoten [GB08, S. 510]

Beim Absuchen eines Bildes wird mit der Evaluation des ersten Knoten begonnen. Liefert dieser ein „Nein“ zurück, so wird die Suche unmittelbar beendet und somit

verkürzt. Nur in dem Fall, dass sämtliche Knoten bei der Entscheidungsfindung mit einem „Ja“ passiert werden, wird ein detektiertes Objekt gemeldet. Ein Beispiel aus [GB08, Chapter 13] verdeutlicht den Vorteil dieses Verfahrens:

> Es sei angenommen, dass jeder dieser Knoten mit einer Wahrscheinlichkeit von 99.9% ein vorhandenes Objekt korrekt erkennt und mit einer Wahrscheinlichkeit von 50% nicht vorhandene Objekte fälschlicherweise Weise als vorhanden detektiert. Bei einer Anzahl von 20 Knoten wird ein vorhandenes Objekt daher mit einer Wahscheinlichkeit von $0.999^{20} \approx 98\%$ erkannt. Die Quote für einen Fehlalarm liegt bei lediglich $0.5^{20} \approx 0.0001\%$.

3.1.2. Das Arbeiten mit OpenCV

Die auf den *Haar-Features* basierende Objekterkennung in OpenCV beruht auf einem Prozess des „supervised Learning“. Dabei werden Trainingsdaten verwendet, von denen die korrekte Klassifizierung bereits bekannt ist. Anhand der gewünschten Ausgabe können nun die entsprechenden *Haar-Features* und ihre Schwellenwerte ermittelt werden. Die Merkmale des Objektes, welche durch dieses Training gesammelt werden, werden in einer speziellen Datei (dem *Classifier*) gespeichert, welcher nun auf neue, unbekannte Daten angewendet werden kann.

Der Trainingsablauf lässt sich dabei in drei wensentliche Abschnitte einteilen:

1. Es werden Bilder gesammelt, die das zu identifizierende Objekt beinhalten. Die Auswahl dieser Bilder ist nicht trivial, da z.B. ein sich ständig ändernder Blickwinkel zu einem schlechteren Resultat führen kann. Die Menge dieser Bilder sollte möglichst groß (mehrere Hundert) sein und wird in zwei Teile aufgesplittet. Dabei werden ca. 95% der Bilder dazu genutzt, einen passenden *Classifier* für OpenCV aufzutrainieren. Die restlichen 5% der Bilder dienen dann zur Verifikation des so erhaltenen *Classifiers*.

Zum bereits erwähnten Auftrainieren ist es zunächst notwendig, von Hand mittels eines mitgelieferten *Object-Markers* die Objekte auf jedem Trainingsbild zu markieren, wobei sich auch mehrere Objekte auf einem Bild befinden können. Dabei wird jedes Objekt durch ein es umrandendes Rechteck identifiziert, dessen Koordinaten gespeichert werden. Diese Auswahl stellt die sogenannten „positiven“ Bilder dar.

Zusätzlich bedarf es noch einer etwa gleich großen Menge an „negativen“ Bildern. Diese sollten alle möglichen Formen an Hintergründen beinhalten, ohne jedoch das zu klassifizierende Objekt darzustellen. Da hierfür der gesamte Bildinhalt relevant ist, müssen diese nicht gesondert markiert werden.

2. Nachdem die notwendigen Eingabedaten gesammelt und in entsprechenden Dateien gespeichert worden sind, wird das Training von OpenCV gestartet. Dieses läuft automatisch in mehreren Stufen ab. Während die Genauigkeit mit jeder weiteren trainierten Stufe zunimmt (vgl. S. 9), erhöht sich auch die zur Berechnung notwendige Zeit. Als Ergebnis wird ein *Classifier* geliefert, der nach einer Umwandlung in das XML-Format für den Einsatz in der Objekterkennung zur Verfügung steht.

3. Da die Möglichkeit besteht, dass im Training ein spezialisierter *Classifier* erzeugt wurde, der lediglich auf den Trainingsdaten korrekt arbeitet, ist es notwendig, mit einem unabhängigen Datensatz seine Qualität zu beurteilen. Dafür werden die im Schritt der Datenerhebung zur Verifikation reservierten Bilder genutzt. Für eine grobe Beurteilung des Ergebnisses genügt es hier, die Ergebnisse der Klassifizierung von Hand zu kontrollieren. Wurden beispielsweise zu wenige oder nicht den Anforderungen entsprechende Trainingsbilder verwendet, so erkennt man dies frühzeitig am stark abweichenden Ergebnis und muss den Schritt des Trainings wiederholen.

Ein fertiger *Classifier* kann nun auf ein unbekanntes Bild angewendet werden, um dort das auftrainierte Objekt zu detektieren. Das Ergebnis, welches man dabei erhält, ist

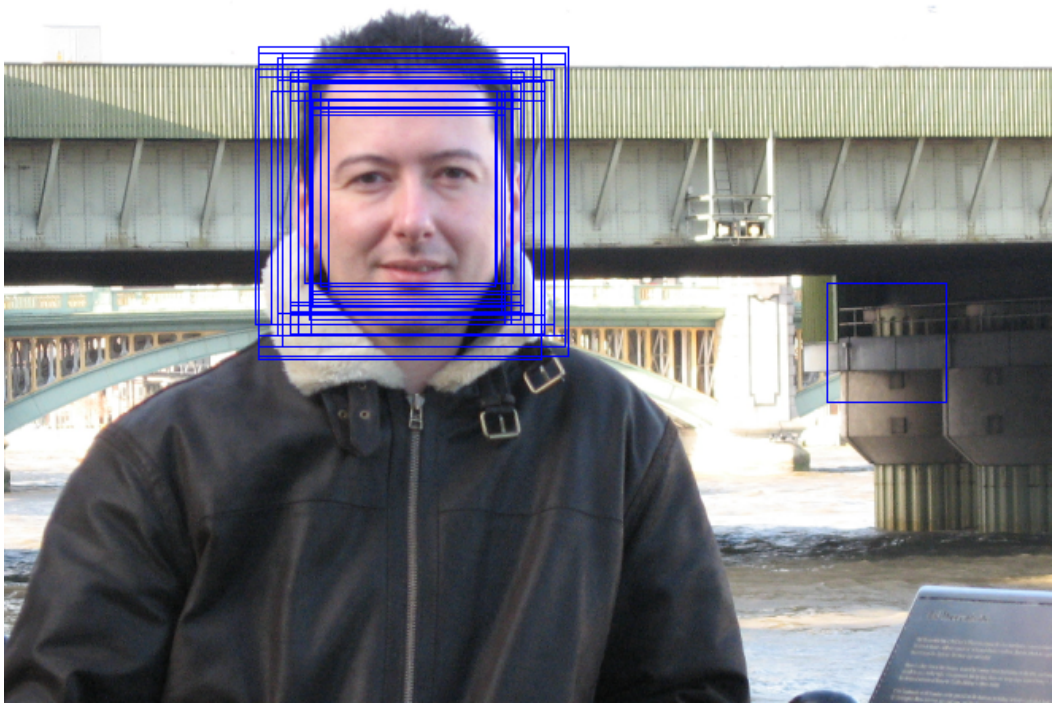

Abbildung 3.5.: Ungefiltertes Ergebnis eines *Classifiers*

am Beispiel einer Gesichtserkennung in Abbildung 3.5 dargestellt. Dabei ist ersichtlich, dass für jedes einzelne erkannte Objekt ein Rechteck in das Bild eingezeichnet wurde. Liegen mehrere dieser Rechtecke übereinander, so ist die Wahrscheinlich größer, dass sich an dieser Stelle tatsächlich ein Objekt befindet.

Aufgrund des hohen erforderlichen Datenvolumens zur Erstellung eines oder mehrerer *Classifier* werden in diesem Projekt vorgefertigte *Classifier* genutzt. Als Quelle dient dabei eine Sammlung verschiedener Autoren, welche unter [Rei] einzusehen ist.

3.2. RacerPro

Das Programm zur logischen Auswertung der vom *Object-Detector* gelieferten Informationen stellt der *Reasoner* dar. Zum Einsatz kommt dabei RacerPro (Renamed ABox and Concept Expression Reasoner), ein kommerzieller *Reasoner* für Beschreibungslogik, der von der Racer Systems GmbH & Co. KG vertrieben wird.

RacerPro arbeitet mit der SHIQ Beschreibungslogik. Dabei handelt es sich um eine Erweiterung von ALC um Kardinalitäten sowie hierarchische, transitive und inverse Rollen. Hierbei können gleichzeitig mehrere T- und ABoxen verwaltet werden (vgl. [rac]).

Sämtliche Modellierungen, die in RacerPro vorgenommen werden, geschehen unter der sogenannten „Open World Assumption“. Diese Annahme ist der Beschreibungslogik immanent und wird daher auch von vergleichbaren Programmen verwendet. Dabei wird davon ausgegangen, dass nie alle Individuen, Konzepte und Relationen der betrachteten Welt erfasst werden können. Wird z.B. ein bestimmtes Individuum erfragt, welches (noch) nicht modelliert wurde, so wird diese Anfrage nicht negativ beantwortet. Daher kann aus der Abwesenheit von Informationen kein Wissen abgeleitet werden, was in Kontrast zu den Konzepten von Datenbanken steht.

Als Besonderheit bietet RacerPro jedoch die Möglichkeit, auch bei derartigen Anfragen dem Benutzer eine verwertbare Aussage zu liefern. Die dabei verwendete logische Abduktion wird in Abschnitt 4.3.2 erläutert.

Die Kommunikation mit anderen Programmen ist in mehreren Formaten möglich. Neben dem weit verbreitenen Standard DIG, welcher auf HTTP basiert, bietet RacerPro auch eine eigene Anfragesprache (Racer Query Language) sowie einer Erweiterung derselben namens nRQL (new Racer Query Language) an, welche noch umfangreichere Funktionalitäten bietet.

3.3. DetEval

Zur Bewertung der erzielten Resultate wird das Programm DetEval (veröffentlicht unter [Wol]) eingesetzt. Dabei handelt es sich um eine für diesen Zweck selbst entwickelte Software von Christian Wolf, Dozent an der Université de Lyon. Der Algorithmus zur Auswertung basiert auf einem von ihm veröffentlichten wissenschaftlichen Artikel (s. [WJ06]).

3.3.1. Eingabeformat

Zur Bewertung des Ergebnisses eines *Classifiers* werden insgesamt zwei Eingabedaten benötigt. Zunächst muss von Hand eine korrekte Lösung, die sogenannte *Ground-Truth* erstellt werden, welche das optimale Ergebnis repräsentiert. Zusätzlich wird noch das vom *Object-Detector* unter Einbezug des *Classifiers* erstellte Ergebnis (*Result*) benötigt. Beide Eingaben weisen dabei dieselbe Struktur auf und bestehen jeweils aus einer XML-Datei. Das dabei verwendete Format stellt eine Erweiterung jenes Formates dar, welches für den ICDAR[1] Wettbewerb 2003 verwendet wurde.

Der Aufbau der beiden Dokumente setzt sich aus Einträgen für jedes untersuchte Bild zusammen (s. List. 3.1). Innerhalb dieser Einträge werden die einzelnen markierten Objekte in Form ihrer Koordinaten und Ausmaße angegeben. Der Typus der Objektes ist bei diesem Projekt nicht relevant.

```
<?xml version="1.0" encoding="UTF-8"?>
<tagset>
  <image>
    <imageName>image001.bmp</imageName>
    <taggedRectangles>
      <taggedRectangle x="174" y="21" width="131" height="168" modelType="0" />
      <taggedRectangle x="320" y="7" width="150" height="178" modelType="0" />
    </taggedRectangles>
  </image>
  <image>
    <imageName>image002.bmp</imageName>
    <taggedRectangles>
      <taggedRectangle x="291" y="267" width="377" height="377" modelType="0" />
    </taggedRectangles>
  </image>
</tagset>
```

Listing 3.1: Eingabeformat

[1] „International Conference on Document Analysis and Recognition“, internationales Forum für Forscher und Anwender um Ideen und Erfahrungen im Bereich von Dokumentenanalyse, -verstehen und -auswertung auszutauschen

3.3.2. Parameter zur Bewertung

Unter Verwendung der *Ground-Truth* und des *Results* wird eine Aussage über die Qualität der gefundenen Objekte generiert. Dazu werden bildweise die einzelnen markierten Objekte miteinander verglichen.

Weisen Objekte der *Ground-Truth* und des *Results* ähnliche Koordinaten und Abmessungen auf, so identifizieren sie dasselbe Objekt auf dem Bild. In diesem Fall ist das erkannte Objekt korrekt. Ebenso kann es sein, dass ein Objekt lediglich in der *Ground-Truth* vorhanden ist, also vom *Object-Detector* nicht erkannt wurde. In diesem Fall spricht man von einem „false negative“, also einem Objekt, das fälschlicherweise nicht erkannt wurde. Analog dazu gibt es Objekte, die zwar im *Result* vorhandenen sind, nicht jedoch in der *Ground-Truth*. Dabei handelt es sich um „false positives“, also fälschlicherweise erkannte Objekte.

Um anhand dieser verschiedenen Möglichkeiten zu einer eindeutigen, objektiven Aussage zu gelangen, werden die Ergebnisse mittels zweier Parameter beurteilt.

Precision

Die *Precision* gibt an, wie treffsicher die Ergebnisse des *Object-Detectors* sind. Werden beispielsweise 5 Objekte gefunden, von denen 3 korrekt und die anderen beiden „false positives“ sind, so hat die *Precision* nach Formel 3.1 einen Wert von $\frac{3}{5}$.

$$Precision = \frac{|\ \{relevante\ Objekte\}\ \cap\ \{gefundene\ Objekte\}\ |}{|\ \{gefundene\ Objekte\}\ |} \tag{3.1}$$

Recall

Der *Recall* gibt an, wieviele der tatsächlich vorhandenen Objekte vom *Object-Detector* auch gefunden wurden. Findet dieser bei 7 vorhandenen Objekten lediglich 5, von denen nur 3 korrekt sind, so beträgt der *Recall* nach Formel 3.2 genau $\frac{3}{7}$.

$$Recall = \frac{\mid \{relevante\ Objekte\} \cap \{gefundene\ Objekte\} \mid}{\mid \{relevante\ Objekte\} \mid} \tag{3.2}$$

Anhand dieser Beispiele lässt sich auch nachvollziehen, warum ein Parameter alleine nicht ausreichend ist, um ein Ergebnis vollständig zu beschreiben. So lässt sich zwar eine sehr hohe *Precision* erreichen, indem stets nur das beste Objekt markiert wird, jedoch führen die dadurch verpassten restlichen Objekte zu einem schlechten Wert des *Recalls*. Umgekehrt lässt sich durch die Markierung aller möglichen Objekte ein sehr hoher *Recall* erreichen, jedoch sinkt durch die Hinzunahme vieler „false positives“ wiederum die *Precision*.

Beide Parameter in Kombination erfüllen jedoch die Anforderungen an eine objektive und vollständige Beurteilung der Ergebnisse.

4. Implementierung

Der wesentliche Programmablauf zur Ermittlung der auf einem Bild vorhandenen Objekte ist in Abbildung 4.1 dargestellt. Dabei handelt es sich um einen fortlaufenden Dialog zwischen dem *Object-Detector* und dem *Reasoner*. Die einzelnen Schritte dabei werden im Folgenden erläutert.

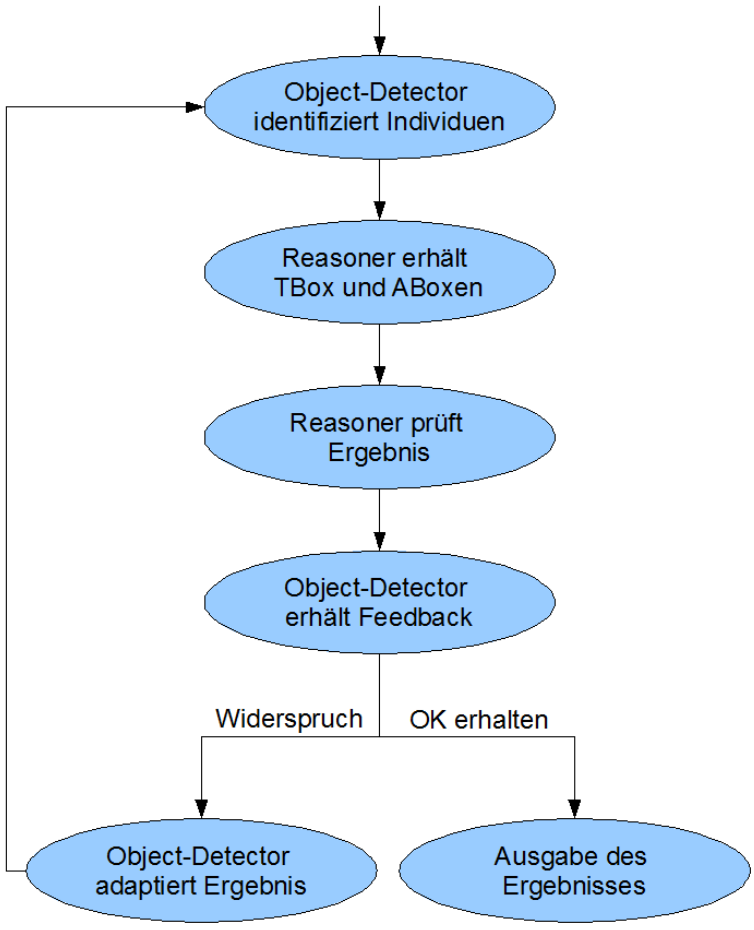

Abbildung 4.1.: Programmablauf

4.1. Objektdetektion

Als erstes vollzieht der *Object-Detector* eine vorläufige Detektion aller auf dem Bild vorhandenen Personen. Das daraus resultierende Ergebnis dient dann als Ausgangspunkt für die anschließende Adaption.

Im Zuge der Arbeit mit verschiedenen *Classifiern* hat sich herausgestellt, dass jeder *Classifier* in unterschiedlichen Situationen zu Fehldetektionen neigt. Um das Gesamtergebnis möglichst unabhängig von der Qualität eines einzelnen *Classifiers* zu erhalten, werden mehrere dieser *Classifier* parallel eingesetzt. So können individuelle Fehldetektionen ausgeglichen werden und insgesamt ist mit einem besseren Ergebnis zu rechnen. Dabei erfolgt die Objektdetektion in zwei Stufen, wobei zunächst die Ergebnisse jedes einzelnen *Classifiers* gesammelt und schließlich zu einem finalen Ergebnis kombiniert werden.

4.1.1. Auswertung eines Classifiers

Der *Classifier* wird durch Aufruf der entsprechenden Funktion aus OpenCV (s. List. 4.1) auf das Bild angewendet. Als Ergebnis wird eine Liste mit vielen einzelnen Objekten zurückgeliefert. Diese Objekte können dabei übereinander liegen, also inhaltlich für dasselbe Objekt auf dem Bild stehen. Zu diesem Zweck bietet OpenCV eine Methode, mit deren Hilfe die überlappenden Objekte zusammengefasst werden können. Da bei diesem Vorgang jedoch die Information verloren geht, wieviele der Objekte auf diese Art miteinander verschmolzen werden, wird diese Funktionalität eigenständig implementiert.

```
// Detect the objects and store them in the sequence
CvSeq* objects = cvHaarDetectObjects( img, cascade, storage,
                                      1.1, 0, CV_HAAR_DO_CANNY_PRUNING,
                                      cvSize(40, 40));
```

Listing 4.1: Aufruf der Funktion zur Objekterkennung

Dafür wird jedes einzelne erkannte Objekt mit allen anderen verglichen. In Abhängigkeit der überlappenden Fläche der beiden Objekte sowie dem Verhältnis ihrer Größe wird entschieden, ob diese dasselbe Objekt repräsentieren. Ist dies der Fall, so werden beide Objekte verschmolzen und der Durchschnitt ihrer Koordinaten gebildet. Zusätzlich wird in dem so neu entstandenen Objekt die Anzahl der darin verschmolzenen Objekte vermerkt, was die objekteigene *Score* darstellt.

In OpenCV ist die Anzahl der auf derselben Position erkannten Objekte proportional zu der Wahrscheinlichkeit, dass sich an dieser Stelle tatsächlich ein Objekt befindet. Daher ist die auf diese Art und Weise ermittelte *Score* eines Objektes ein guter Indikator für seine Qualität.

Das Ergebnis eines solchen Durchlaufes ist in Abbildung 4.2 dargestellt. Dabei beinhaltet jedes eingezeichnete Objekt in seiner linken oberen Ecke eine eindeutige ID und in der rechten oberen Ecke die beschriebene *Score*. Dieselben Informationen sind noch einmal der Deutlichkeit halber in der Konsole dargestellt.

Abbildung 4.2.: Ergebnis eines einzelnen *Classifiers*

4.1.2. Kombination mehrerer Classifier

Nachdem mit den Ergebnissen jedes einzelnen *Classifiers* wie in Abschnitt 4.1.1 beschrieben verfahren wurde, müssen diese noch zu einem Gesamtergebnis kombiniert werden. Das Prinzip dabei ist nahezu dasselbe wie zuvor. Erneut werden alle Objekte miteinander verglichen und im Bedarfsfall verschmolzen. Wichtig hierbei ist jedoch, dass die Information, welche *Classifier* an der Erkennung eines Objektes beteiligt waren, erhalten bleibt. Zusätzlich ist es möglich, die Gewichtung der einzelnen *Classier* zu verändern, wovon in diesem Projekt jedoch kein Gebrauch gemacht wird.

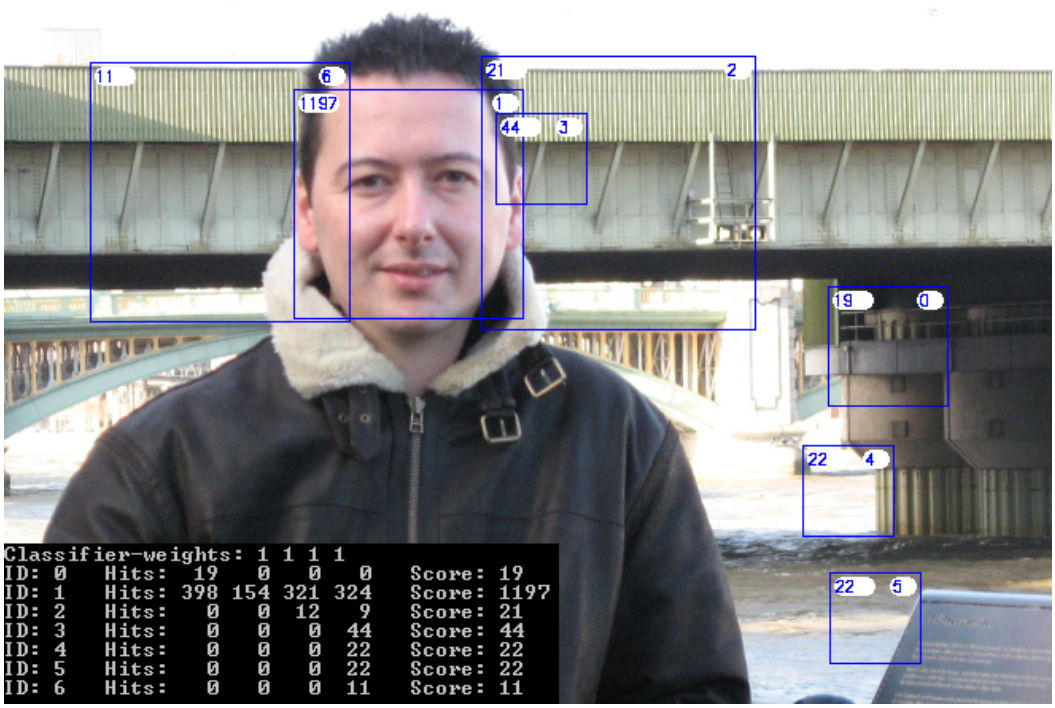

Abbildung 4.3.: Kombiniertes Ergebnis mehrerer *Classifier*

Als Ergebnis der Anwendung von 4 verschiedenen *Classifiern* zur Gesichtserkennung erhält man Abbildung 4.3. Hierbei gibt die Konsole genaueren Aufschluss über die Entstehung der Ergebnisse, da die *Scores* jedes einzelnen *Classifiers* noch einmal separat aufgelistet sind. Am gezeigten Beispiel ist die Streuung der Ergebnisse sehr gut sichtbar. Während der zweite *Classifier*, dargestellt durch die zweite Spalte der „Hits“, lediglich 1 Objekt identifizieren konnte, sind es beim vierten *Classifier* (vierte Spalte)

hingegen 6. Da sich diese Situation von Bild zu Bild verändert, entsteht durch die Kombination der Einzelresultate ein insgesamt unabhängigeres Ergebnis.

4.1.3. Auswahl der Objekte - Referenzmethode

Bisher wurde lediglich betrachtet, wie sämtliche von den *Classifiern* identifizierten Objekte erfasst und bewertet werden. Da es nach einer Betrachtung von Abbildung 4.3 aber nicht sinnvoll erscheint, davon auszugehen, dass alle erkannten Objekte relevant sind, muss ein Selektionsverfahren gewählt werden, um aus der erhaltenen Liste von Objekten die plausibelsten auszuwählen.

Wie bereits in Abschnitt 4.1.1 erläutert wurde ist die ermittelte *Score* eines jeden Objektes ein guter Indikator für dessen Qualität. Daher ist es naheliegend, lediglich diejenigen Objekte auszuwählen, die eine ausreichend hohe *Score* aufweisen. Dafür wird zunächst ein fixer Schwellenwert festgelegt. Jedes Objekt, dessen *Score* oberhalb dieses Schwellenwertes liegt, bekommt den Status „erkannt“, alle anderen erhalten den Status „nicht erkannt“. Diese Form des Auswahlverfahrens stellt später die Referenzmethode dar, anhand derer die Ergebnisse des rückgekoppelten Systems beurteilt werden.

Die Einschränkung dieser Methode liegt in der starren Auswahl des Schwellenwertes begründet. Während in Abbildung 4.3 ein Schwellenwert von 200 zum korrekten Ergebnis geführt hätte, kann dies bei einem anderen Bild grundsätzlich differieren. Ohne weiterführende Informationen über den Bildinhalt lässt sich auch nicht bestimmen, auf welche Weise dieser Schwellenwert angepasst werden muss.

4.2. Beschreibung der Wissensbasis

4.2.1. Szenario

Das für die Durchführung des Experimentes gedachte Szenario bezieht sich auf das am weitesten verbreitete Themengebiet der Objekterkennung: die Erkennung von Personen. Die grundlegende Idee dieses Szenarios ist es, ein Bild anhand der Anzahl vorhandener Personen zu klassifizieren. Dabei wird lediglich ein Typ von *Classifiern* zur Gesichtserkennung benötigt. Basierend auf diesem Parameter werden insgesamt drei verschiedene Möglichkeiten (*Scenes*) von Bildern unterschieden: das Portrait (1 Person), das Interview (2 Personen) und das Gruppenfoto (3 oder mehr Personen). Die Darstellung in beschreibungslogischer Syntax lautet entsprechend:

$$Portrait \sqsubseteq Scene \sqcap (=_1 hasParticipant.Person) \tag{4.1}$$

$$Interview \sqsubseteq Scene \sqcap (=_2 hasParticipant.Person) \tag{4.2}$$

$$Groupshot \sqsubseteq Scene \sqcap (\geq_3 hasParticipant.Person) \tag{4.3}$$

Dabei ist vorgesehen, dass die Information, um welchen Bildtyp es sich handelt, von dem virtuellen *Text-Detector* stammt. Dieser könnte z.B. aus der Bildunterschrift das Wort „Klassenfoto“ extrahieren und darauf basierend zu der Aussage gelangen, dass es sich um ein Gruppenfoto handelt.

4.2.2. Informationsübermittlung

Die Informationen, welche dem *Reasoner* zur Verfügung gestellt werden, stammen aus unterschiedlichen Quellen. Zunächst wird die TBox, welche sämtliche Informationen über die Konzepte des vorher definierten Szenarios enthält, aus einer externen Datei eingelesen und übermittelt:

```
(implies Portrait Scene)
(implies Interview Scene)
(implies Groupshot Scene)

(implies Portrait(exactly 1 hasParticipant))
(implies Interview(exactly 2 hasParticipant))
(implies Groupshot(at-least 3 hasParticipant))

(define-rule (?x full-Portrait)(and (?x Portrait)(?x \$?u hasParticipant)))
(define-rule (?x full-Interview)(and (?x Interview)(?x \$?u hasParticipant)
                                                    (?x \$?v hasParticipant)))
(define-rule (?x full-Groupshot)(and (?x Groupshot)(?x \$?u hasParticipant)
                                                    (?x \$?v hasParticipant)
                                                    (?x \$?w hasParticipant)))
```

Listing 4.2: TBox-Einträge

Ähnliches geschieht mit der Eingabe durch den virtuellen *Text-Detector*. Seine Informationen liegen ebenfalls in einer externen Datei vor und beschreiben einen zu jedem Bild passenden Input in Form eines Szenentyps. Daraus wird dann der entsprechende Eintrag in die Wissensbasis aufgebaut:

```
(instance scene0 Interview)
```

Listing 4.3: ABox-Einträge des *Text-Detectors*

Die vom *Object-Detector* gewonnenen Resultate hingegen stehen erst zur Laufzeit zur Verfügung. So wird für jedes erkannte Objekt ein passendes Individuum unter Berücksichtigung der notwendigen Relationen erzeugt. Die daraus resultierenden Einträge in der ABox werden ebenfalls an den *Reasoner* übermittelt:

```
(instance person1 Person)
(instance person2 Person)
(different-from person1 person2)
(related scene0 person1 hasParticipant)
(related scene0 person2 hasParticipant)
```

Listing 4.4: ABox-Einträge des *Object-Detectors*

4.3. Reasoning

Hat der *Reasoner* alle Informationen erhalten, stellt der *Object-Detector* verschiedene Anfragen (*Queries*) an diesen, um die Korrektheit der übermittelten Wissensbasis zu überprüfen. Das Format, in welchem diese *Queries* getätigt werden, entspricht dabei der „Racer Query Language“.

Bei dem in Kapitel 4.2.1 vorgestellten Szenario sind genau zwei verschiedene Formen von *Queries* vorgesehen.

4.3.1. Konsistenz

Bei der Anfrage auf Konsistenz wird überprüft, ob sich in der übermittelten Wissensbasis ein Widerspruch befindet. Da die in der TBox definierten Konzepte widerspruchsfrei sind und im Programmablauf ihr Inhalt nicht verändert wird, muss lediglich der Inhalt der ABox auf Widersprüche getestet werden:

```
(abox-consistent-p)
```

Listing 4.5: Konsistenzanfrage

Ein derartiger Widerspruch kann in dem beschriebenen Szenario lediglich auftreten, falls auf einem Bild mehr Objekte erkannt wurden als es der identifizierte Typ dieses Bildes zulässt. Die folgende Situation dient hierfür als Beispiel:

> Der *Text-Detector* liefert als Bildtyp ein „Interview“, wohingegen der *Object-Detector* 3 verschiedene Personen identifiziert hat. Diese Information kollidiert jedoch mit dem zuvor in Definiton 4.2 formulierten Konzept, dass ein Interview genau 2 Teilnehmer besitzt. Diesen Widerspruch teilt der *Reasoner* dem *Object-Detector* mit, indem er meldet, dass die eingegebene Wissensbasis inkonsistent ist.

Sind auf einem Bild jedoch weniger Objekte erkannt worden als gefordert, so führt dies zu keinem Widerspruch. Der Grund für dieses Verhalten liegt in der durch den *Reasoner* angenommenen „Open World Assumption“, welche in Abschnitt 3.2 bereits dargestellt wurde. Aufgrund dieser Tatsache ist das Erkennen von nicht modellierten, also vom *Object-Detector* nicht gefundenen, Objekten mittels einer Prüfung auf Konsistenz nicht möglich.

4.3.2. Abduktion

Für den Fall, dass vom *Object-Detector* zu wenige Objekte identifiziert wurden, wird eine Prüfung auf Abduktion durchgeführt. Eine detaillierte Erläuterung zu dem in RacerPro verwendeten Prinzip findet sich in [Möl]. Im Vergleich zu der Konsistenzanfrage ist es dabei notwendig, eine zusätzliche Bedingung in Form einer zu erfüllenden Regel anzugeben:

```
(retrieve-with-explanation()(scene0 full-Interview):final-consistency-checking-p NIL)
```

Listing 4.6: Beispiel zur Abduktionsanfrage

Dafür ermittelt der *Reasoner*, ob alle zur Erfüllung der Bedinung notwendigen Individuen in der Wissensbasis explizit modelliert wurden oder ob zusätzlich Individuen unter der Annahme der „Open World Assumption“ hypothetisiert werden müssen. Das folgende Beispiel soll dies verdeutlichen:

> Vom *Text-Detector* wird ein Interview identifiziert, vom *Object-Detector* jedoch nur eine Person erkannt. Der *Reasoner* prüft diese Informationen und stellt fest, dass zur Erfüllung der geforderten Regel eines vollständigen Interviews ein weiteres Individuum vorhanden sein muss und hypothetisiert dieses. Die Anzahl der hypothetisierten Individuen wird dem *Object-Detector* daraufhin mitgeteilt.

4.4. Adaption der Objekterkennung

Im Anschluss an die logische Prüfung der Wissenbasis durch den *Reasoner* erhält der *Object-Detector* die entsprechenden Resultate. In Abhängigkeit der beiden in Kapitel 4.3 beschriebenen Fälle muss der *Object-Detector* sein eigenes Ergebnis verändern.

Zu dieser Anpassung werden hier zwei verschiedene Methoden beschrieben. Anhand der gegebenen Daten und Funktionen der Implementierung sind aber auch andere Ansätze denkbar.

Methode 1

Als Basis dient die im ersten Schritt der Objektdetektion gewonnene Liste mit Objekten, denen jeweils eine *Score* zugeordnet ist (vgl. Kap. 4.1). Hierbei wurden zunächst alle Objekte als „erkannt“ markiert, deren *Score* über einem zuvor festgelegten Schwellenwert lag. Wird durch den *Reasoner* nun übermittelt, dass die Anzahl der erkannten Objekte nicht korrekt ist, so wird dieser Schwellenwert implizit verändert, damit mehr bzw. weniger Objekte markiert werden.

Im Falle einer gemeldeten Inkonsistenz wird dafür das Objekt ausgewählt, welches unter denen als „erkannt“ markierten Objekten die geringste Punktzahl aufweist. Der Status dieses Objektes wird nun auf „nicht erkannt“ gesetzt, wodurch sich die Anzahl der als „erkannt“ markierten Objekte um genau 1 verringert.

Bei einer gemeldeten Abduktion wird analog das Objekt ausgewählt, welches unter denen als „nicht erkannt“ markierten Objekten die höchste Punktzahl aufweist und sein Status auf „erkannt“ gesetzt. Sollte kein Objekt mehr vorhanden sein, welches als „nicht erkannt“ markiert ist, wird die Adaption beendet.

In beiden Fällen wird also lediglich 1 erkanntes Objekt hinzugefügt bzw. entfernt. Bei einer Abweichung von z.B. 2 Objekten zu einem logisch korrekten Ergebnis werden die entsprechenden Funktionen in einer Schleife wiederholt.

Methode 2

Um einen zusätzlichen Nutzen aus der Kombination von mehreren *Classifiern* zu ziehen, wird ein zweites Anpassungsverfahren betrachtet. Im Gegensatz zum ersten Ansatz wird hierbei die *Score* der erkannten Objekte verändert anstatt den Schwellenwert anzupassen. Dafür wird zunächst untersucht, wieviele der in diesem Fall 4 verwendeten *Classifier* an der Erkennung eines Objektes beteiligt sind. Dabei liegt die Idee zugrunde, dass neben einer hohen *Score* auch eine hohe Anzahl an beteiligten *Classifiern* ein Indikator für ein qualitativ gutes Ergebnis sein kann.

Zur Manipulation der *Score* eines Objektes wird diese mit einem hinzukommenden Multiplikator neu gewichtet. Dabei werden wieder die beiden möglichen Fälle der Inkonsistenz und Abduktion unterschieden.

Da bei der Inkonsistenz die Anzahl an erkannten Objekten zu hoch ist, wird ein Multiplikator gewählt, der kleiner als 1 ist und somit die *Score* eines jeden Objektes verringert. Zur erwähnten Abstufung in Abhängigkeit der beteiligten *Classifier* werden folgende Multiplikatoren verwendet:

- 0.1 bei einem *Classifier*
- 0.25 bei zwei *Classifiern*
- 0.5 bei drei *Classifiern*
- 0.9 bei vier *Classifiern*

Die Anwendung der Multiplikatoren auf die *Score* eines jeden Objektes erfolgt so lange, bis die *Score* wenigstens eines Objektes unter den Schwellenwert sinkt. Da die notwendige Anzahl dieses Schrittes vorher nicht bekannt ist, wird der Vorgang solange wiederholt, bis der *Reasoner* keine Inkonsistenz mehr meldet.

Die Möglichkeit der Abduktion wird demzufolge mit Multiplikatoren behoben, welche größer als 1 sind und somit die *Score* der Objekte erhöhen:

- 1.25 bei einem *Classifier*
- 1.5 bei zwei *Classifiern*
- 2 bei drei *Classifiern*
- 4 bei vier *Classifiern*

Im Gegensatz zu Methode 1 wird bei diesem Verfahren im Falle einer Inkosistenz oder Abduktion nicht zwingend der Status genau eines Objektes verändert. Weisen zwei Objekte sehr ähnliche *Scores* auf, so können diese im selben Schritt durch eine Multiplikation unter bzw. über den Schwellenwert gelangen. Der dabei gegebenenfalls auftretende neue Widerspruch wird durch das anschließende Anwenden von Methode 1 gelöst.

5. Experiment

Zur Überprüfung, ob die in Kapitel 1.1 erwarteten Effekte erzielt werden, wird eine statistische Erhebung durchgeführt. Grundlage dieser Erhebung stellt ein Datensatz von insgesamt 75 Bildern dar, welche zu gleichen Teilen aus den drei verschiedenen in Abschnitt 4.2.1 beschriebenen *Scenes* bestehen. Auf diese Bilder werden die beiden in Kapitel 4.4 beschriebenen Methoden angewendet und mit der in Abschnitt 4.1.3 dargestellten Referenzmethode verglichen. Als Bewertungsmaßstab dienen die in Abschnitt 3.3.2 eingeführten Parameter.

5.1. Visualisierung

Zur besseren Nachvollziehbarkeit des Zustandekommens einer Bildklassifizierung werden die ermittelten Objekte farblich unterschieden. Die dabei verwendete Codierung lautet wie folgt:

- Rot: Das Objekt gilt sowohl vor als auch nach dem Adaptionsschritt als „nicht erkannt“.
- Grün: Das Objekt gilt sowohl vor als auch nach dem Adaptionsschritt als „erkannt“.
- Blau: Das Objekt galt zunächst als „nicht erkannt“, änderte seinen Status nach der Adaption jedoch auf „erkannt“.
- Orange: Das Objekt galt zunächst als „erkannt“, änderte seinen Status nach der Adaption jedoch auf „nicht erkannt“.

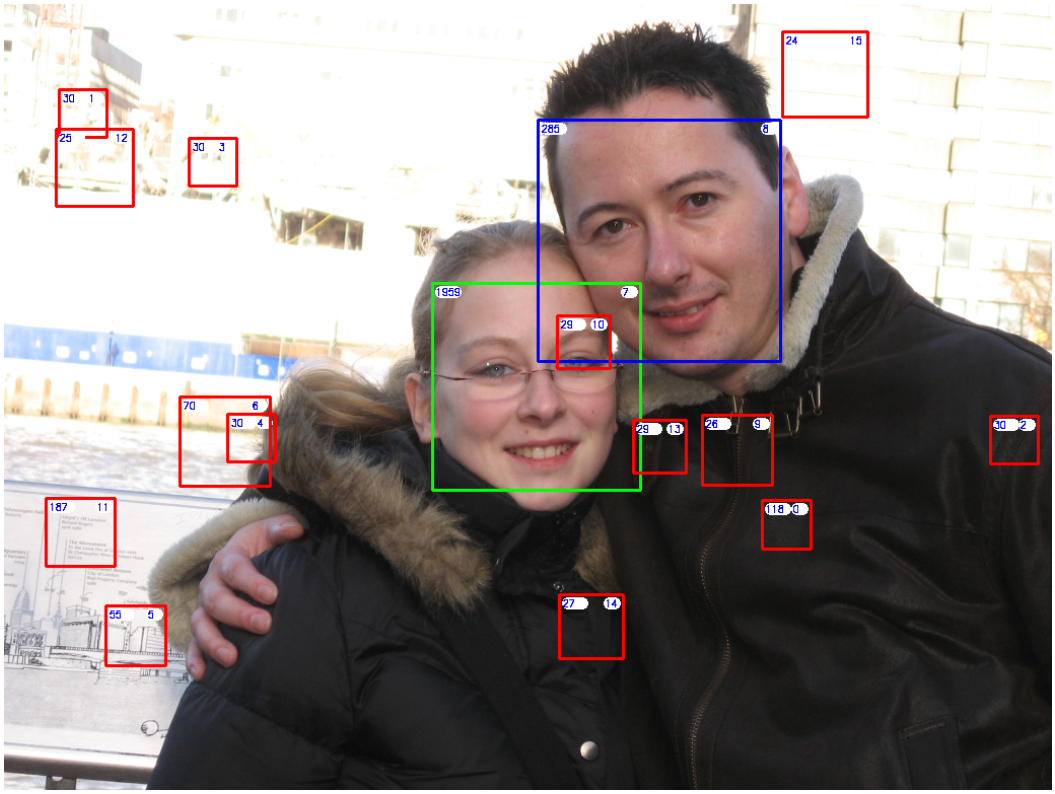

Abbildung 5.1.: Resultat einer Abduktion

Als Beispiel sind zwei verschiedene Situationen abgebildet. Es handelt sich dabei um dasselbe Bild, jedoch wurde die Objekterkennung mit unterschiedlichen Schwellenwerten initialisiert. So ist in Abbildung 5.1 ersichtlich, dass zunächst nur eine von zwei Personen erkannt wurde. Im Schritt der Abduktion (s. Kap. 4.3.2) wurde dann die in diesem Fall blau markierte Person nachträglich hinzugefügt.

In Abbildung 5.2 hingegen wurden bedingt durch einen niedrigeren Schwellenwert zunächst 4 Objekte detektiert. Von diesen wurden bei der Prüfung der Konsistenz (s. Kap. 4.3.1) die beiden orange markierten entfernt.

Das Result ist in beiden Situationen identisch, da beide vorhandenen Personen korrekt klassifiziert wurden.

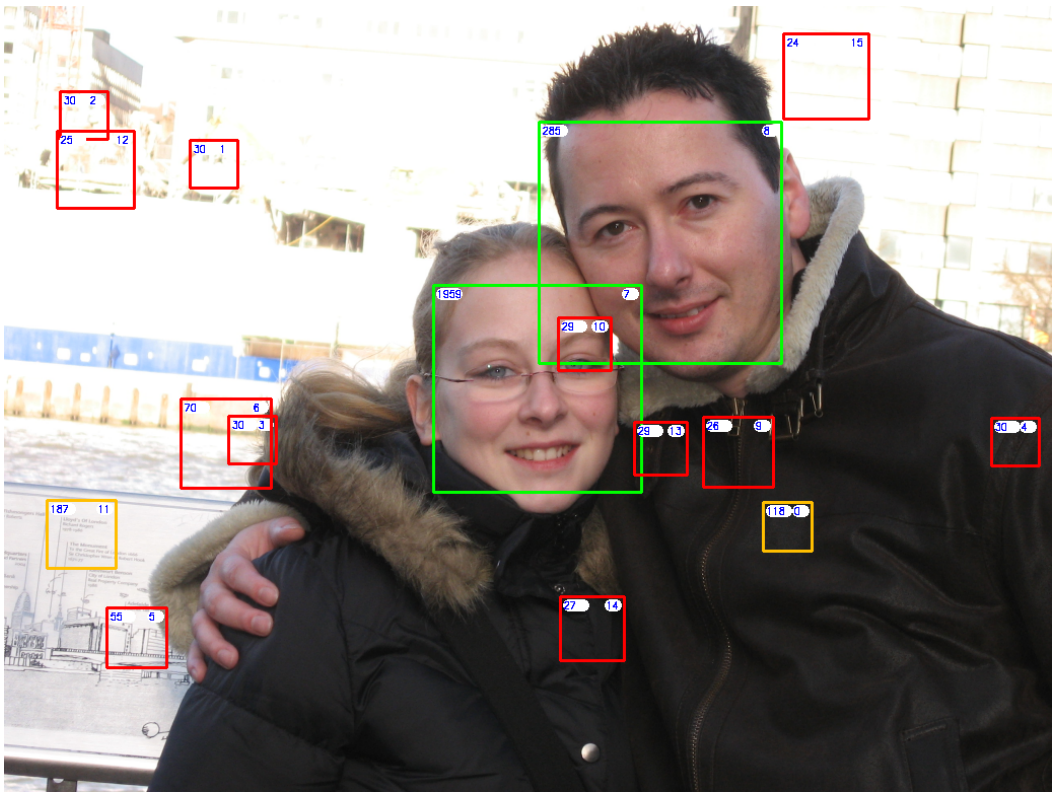

Abbildung 5.2.: Resultat einer Inkonsistenz

5.2. Ergebnisse und Bewertung

Zur Vergleichbarkeit der verschiedenen Verfahren zur Objekterkennung werden die Ergebnisse für unterschiedliche Schwellenwerte ermittelt. Dieser Schwellenwert ist für die Referenzmethode maßgeblich, da nur solche Objekte als „erkannt“ markiert werden, deren *Score* über diesem Schwellenwert liegt. Für die beiden rückgekoppelten Methoden dient das mittels dieses Schwellenwertes ermittelte Ergebnis lediglich als Ausgangssituation für den nachfolgenden Schritt der Adaption.

Die numerischen Werte, anhand derer die im folgenden Abschnitt dargestellten Graphen erzeugt wurden, sind tabellarisch im Anhang aufgeführt.

Referenzmethode

Zunächst erfolgt eine Betrachtung der Referenzmethode. Die in Abbildung 5.3 dargestellte Auswertung der *Precision* und des *Recalls* zeigt einen erwarteteten Verlauf. Bei einem sehr niedrigen Schwellenwert werden sehr viele Bilder als „erkannt“ markiert, was zu einem hohen *Recall* aber einer niedrigen *Precision* führt. Mit steigendem Schwellenwert wird die Anzahl der erkannten Objekte immer geringer, dafür jedoch genauer.

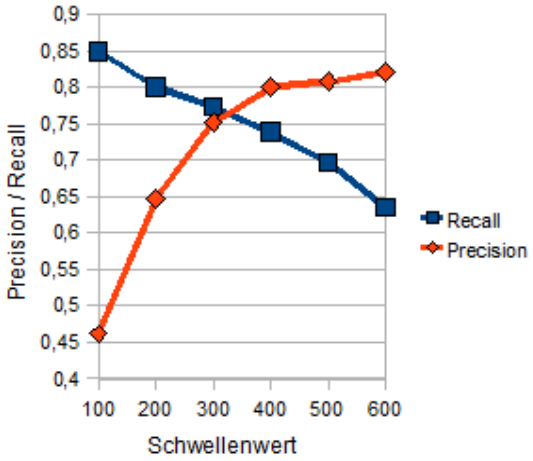

Abbildung 5.3.: Referenzmethode

Bei einer gleichgewichteten Einbeziehung beider Parameter scheint das beste Ergebnis bei einem Schwellenwert zwischen 300 und 400 erreicht, da hier der gemittelte Wert aus *Precision* und *Recall* bei ca. 0.76 liegt.

Rückgekoppelte Methoden

Die Auswertungen der beiden rückgekoppelten Methoden in den Abbildungen 5.4 und 5.5 zeigen im direkten Vergleich einen sehr ähnlichen Verlauf. Ab einem Schwellenwert von 300 zeigt sich nahezu keine Veränderung der beiden Parameter mehr.

Die bis zu einem Schwellenwert von 300 zunächst geringe *Precision* erklärt sich durch das gewählte Szenario. Da bei der Definiton des „Gruppenfotos“ (s. Formel 4.3) die Anzahl der auf dem Bild vorhandenen Personen lediglich nach unten hin begrenzt ist, führt ein niedriger Schwellenwert dazu, dass auf diesem Bildtyp fälschlicherweise zu viele Objekte erkannt werden. Daher erscheint es sinnvoll, mit einem bereits bei der Auswertung der Referenzmethode als sinnvoll ermittelten Schwellenwert von mindestens 300 zu arbeiten.

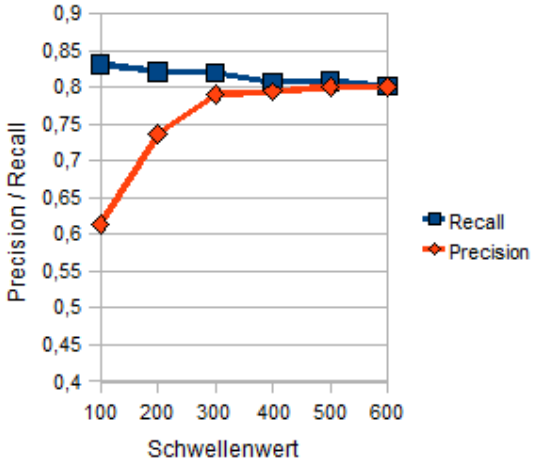

Abbildung 5.4.: Methode 1

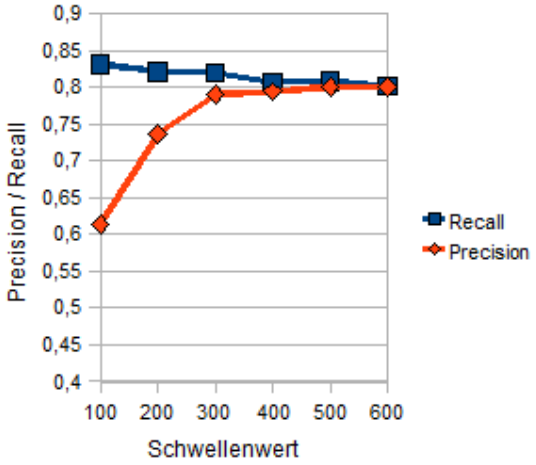

Abbildung 5.5.: Methode 2

Im Vergleich zur Referenzmethode weisen die beiden rückgekoppelten Methoden nahezu konstant hohe Wert bei *Precision* und *Recall* auf. Die Referenzmethode hingegen beinhaltet eine umgekehrt proportionale Abhängigkeit der beiden Parameter.

Ingesamt ergibt sich bei Methode 1 ein Optimum, welches sowohl für die *Precision* als auch den *Recall* einen Wert von ca. 0.8 aufweist und somit über den Werten der Referenzmethode liegt.

Um mögliche Unterschiede zwischen Methode 1 und Methode 2 besser erkennen zu können, werden alle drei Ergebnisse in den Abbildungen 5.7 und 5.6 direkt miteinander verglichen. Dabei ist ersichtlich, dass die *Precision* bei Methode 2 nahezu keine

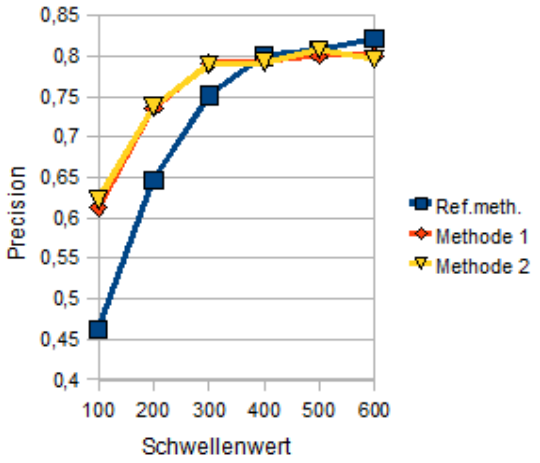

Abbildung 5.6.: Vergleich der *Precision*

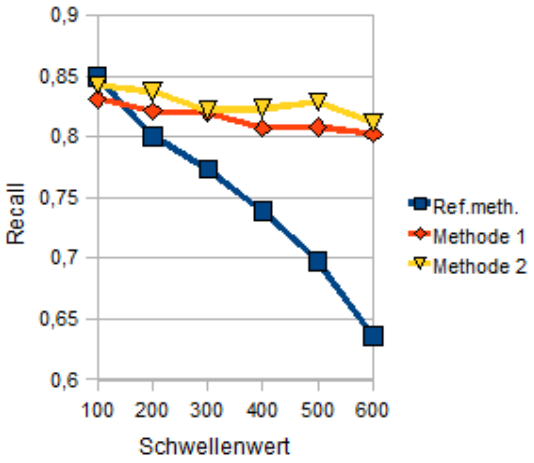

Abbildung 5.7.: Vergleich des *Recalls*

Veränderung gegenüber Methode 1 aufzeigt. Beim *Recall* hingegen lässt sich eine Steigerung um einen Wert von bis zu 0.02 erkennen.

Fazit

Insgesamt lässt sich bei beiden rückgekoppelten Methoden eine Steigerung der Parameterwerte gegenüber der Referenzmethode feststellen. Ebenso wird ein nahezu vom Schwellenwert unabhängiges Ergebnis erzielt. Die rückgekoppelte Methode 2 erweist sich gegenüber der Methode 1 bezüglich der *Precision* als geringfügig besser.

6. Zusammenfassung und Ausblick

Im Anschluss an die erhaltenen Ergebnisse gilt es zu klären, inwieweit die in der Einleitung gestellte Zielsetzung erfüllt wurde. Zu diesem Zweck wird der Inhalt des Projektes noch einmal kurz zusammen gefasst. Anschließend werden die sich daraus ergebenen Möglichkeiten der Weiterentwicklung aufgezeigt.

6.1. Zusammenfassung

Das Ziel dieses Projektes war es, zu untersuchen, ob durch eine logische Rückkopplung zwischen *Reasoner* und *Object-Detector* das Ergebnis der Objekterkennung verbessert werden kann.

Dazu wurde ein System implementiert, welches sich aus dem *Reasoner* in Form von RacerPro, dem *Object-Detector* als ein selbst entworfenes Programm auf der Basis von OpenCV sowie einem virtuellen *Text-Detector* zusammensetzt. Zur Evaluation der erhaltenen Ergebnisse diente das Programm DetEval.

Das gewählte Szenario setzt auf die Erkennung von Personen. Die vom *Text-Detector* stammende Information über die Anzahl der auf dem Bild vorhandenen Personen führt zu einer Klassifizierung des Bildes. Diese Information wird mit den vom *Object-Detector* erkannten Individuen fusioniert und an den *Reasoner* übermittelt. Nach erfolgter logischer Prüfung erhält der *Object-Detector* eine Rückmeldung vom *Reasoner* über die Plausibilität der übermittelten Daten. Als Reaktion auf ein unplausibles Ergebnis adaptiert der *Object-Detector* sein Verfahren zur Objekterkennung, um zu einem besseren Ergebnis zu gelangen.

Zu diesem Zweck wurden zwei verschiedene Methoden dargestellt, welche mittels unterschiedlicher Ansätze die Informationen der Rückkopplung zur Adaption nutzen.

Die Bewertung des so aufgestellten System erfolgte anhand einer Testreihe mit 75 Bildern. Dabei wurden die beiden rückgekoppelten Methoden mit einer Referenzmethode verglichen, welche nicht das Prinzip der logischen Rückmeldung nutzt.

In der nach einem standartisierten Verfahren erfolgten Evaluation der Ergebnisse ist eine messbare Verbesserung der Objekterkennung durch die Rückkopplung ersichtlich geworden. Somit ist auch mit einer qualitativen Verbesserung von logischen Aussagen zu rechnen, die auf diesen Ergebnissen basieren.

6.2. Ausblick

Das in diesem Projekt verwendete Szenario bedient sich eines virtuellen *Text-Detectors* als sekundäre Informationsquelle. Bei entsprechendem Design der Wissenbasis ist es jedoch möglich, nur anhand der Ergebnisse des *Objetdetectors* ein ähnliches Verfahren zu verwenden. Dabei wäre es erforderlich, verschiedene Objekttypen voneinander zu unterscheiden und anhand der modellierten Abhängigkeiten zwischen diesen Objekten die Rückschlüsse zu ziehen.

Mit Hinblick auf ein derart neu gestaltetes Szenario ergäbe sich auch das Erfordernis, Wahrscheinlichkeiten in die logischen Prüfungen mit einzubeziehen. Diese würden eine realitätsnähere Modellierung der beschriebenen Abhängigkeiten ermöglichen. Eine Analyse für die Erweiterung der Beschreibungslogik um Wahrscheinlichkeiten wird z.B. in [Nae07] durchgeführt.

Bei Erweiterungen der Domäne ist jedoch auch die Performance des verwendeten *Reasoners* zu beachten. Während dieses Projekt keinen Fokus auf die Ausführungszeit legt, gilt es bei der Betrachtung von Echtzeitproblemen herauszufinden, ob eine Übertragung der gewählten Methoden in Bezug auf die erzielte Bearbeitungsdauer sinnvoll erscheint.

A. Anhang

Wertetabellen

Schwellenwert	100	200	300	400	500	600
Recall	0.85	0.8	0.77	0.74	0.7	0.64
Precision	0.46	0.65	0.75	0.8	0.81	0.82

Tabelle A.1.: Referenzmethode

Schwellenwert	100	200	300	400	500	600
Recall	0.83	0.82	0.82	0.81	0.81	0.8
Precision	0.61	0.74	0.79	0.79	0.8	0.8

Tabelle A.2.: Methode 1

Schwellenwert	100	200	300	400	500	600
Recall	0.84	0.84	0.82	0.82	0.83	0.81
Precision	0.62	0.74	0.79	0.79	0.81	0.8

Tabelle A.3.: Methode 2

Literaturverzeichnis

[BCM+03] Baader, Franz (Hrsg.) ; Calvanese, Diego (Hrsg.) ; McGuinness, Deborah L. (Hrsg.) ; Nardi, Daniele (Hrsg.) ; Patel-Schneider, Peter F. (Hrsg.): *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003

[GB08] Gary Bradski, Adrian K.: *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly Media, 2008

[Hew] Hewitt, Robin: *How Face Detection Works*. `http://www.cognotics.com/opencv/servo_2007_series/part_2/sidebar.html`, . – [Online; Zugriff am 21.05.2009]

[Möl] Möller, Ralf: *Rule formalism extended to support abduction*. `http://www.sts.tu-harburg.de/~r.f.moeller/racer/Racer-1-9-2-beta-Release-Notes/release-notes-1-9-2se6.html`, . – [Online; Zugriff am 14.06.2009]

[Nae07] Naeth, Tobias H.: *Analysis of the average-case behavior of an inference algorithm for probabilistic description logics*, TU Hamburg-Harburg, Diplomarbeit, Februar 2007

[rac] *What is RacerPro?* `http://www.racer-systems.com/products/racerpro/index.phtml`, . – [Online; Zugriff am 10.06.2009]

[Rei] Reimondo, Alejandro F.: *Haar Cascades*. `http://alereimondo.no-ip.org/OpenCV/34`, . – [Online; Zugriff am 25.05.2009]

[WJ06] Wolf, C. ; Jolion, J.-M.: Object count/Area Graphs for the Evaluation of Object Detection and Segmentation Algorithms. In: *International Journal on Document Analysis and Recognition* 8 (2006), Nr. 4, S. 280–296

[Wol] Wolf, Christian: *DetEval - Evaluation software for object detection algorithms*. `http://liris.cnrs.fr/christian.wolf/software/deteval/index.html`, . – [Online; Zugriff am 11.06.2009]