ALBERT-LUDWIGS-UNIVERSITÄT
FREIBURG
INSTITUT FÜR INFORMATIK

Arbeitsgruppe Autonome Intelligente Systeme

Prof. Dr. Wolfram Burgard

Bild- und laserbasierte Klassifikation von Umgebungen mit mobilen Robotern

Diplomarbeit

Axel Rottmann

Januar 2005 - Juli 2005

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt und alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder unveröffentlichten Schriften entnommen wurden, als solche kenntlich gemacht habe. Außerdem erkläre ich, dass die Diplomarbeit nicht, auch nicht auszugsweise, bereits für eine andere Prüfung angefertigt wurde.

(Axel Rottmann)
Freiburg, den 5. Juli 2005

Danksagung

An dieser Stelle möchte ich mich bei den Personen bedanken, die mich bei der Erstellung meiner Diplomarbeit unterstützt haben. Ganz besonders danken möchte ich Herrn Prof. Dr. Wolfram Burgard, dass er mich an die wissenschaftliche Arbeitsweise herangeführt hat und mir jederzeit für offene Fragen hilfreich zur Seite stand. Auch bei Cyrill Stachniss und Óscar Martínez Mozos möchte ich mich für deren Unterstützung und die Anregungen bedanken. Ebenfalls danke ich Rudolph Triebel, Christian Plagemann und Patrick Pfaff für viele kleine Tipps und die gute Atmosphäre in der Arbeitsgruppe. Des Weiteren Danke ich Renate und Daniel für ihre Kritikpunkte während der schriftlichen Ausarbeitung. Schließlich bedanke ich mich bei meiner Mutter für Ihr Verständnis und Ihre Unterstützung während des gesammten Studiums und speziell in der heißen Endphase.

Inhaltsverzeichnis

1. Einleitung **9**
- 1.1. Zielsetzung 10
- 1.2. Der Aufbau dieser Arbeit 11

2. Verwandte Arbeiten **13**

3. Der AdaBoost Algorithmus **17**
- 3.1. Einführung in Boosting 17
- 3.2. AdaBoost als binärer Klassifikator 18
- 3.3. Bestimmung der Gewichte α_t 19
- 3.4. Bildung der schwachen Klassifikatoren 22
- 3.5. Multi-Klassen Klassifikator 26
  - 3.5.1. Sequenz binärer AdaBoost Klassifikatoren 27
  - 3.5.2. Reihenfolge der Klassifikatoren in einer Entscheidungsliste 29

4. Merkmale zur Erzeugung der schwachen Klassifikatoren **35**
- 4.1. Merkmale der Laserdaten 37
- 4.2. Merkmale der Kameradaten 41
- 4.3. Verfahren zur Erstellung der Merkmale aus den Kameradaten 43

5. Wahrscheinlichkeitsbasierte Klassifikation von Beobachtungen **49**
- 5.1. Das Bewegungsmodell $P(\xi_t \mid \xi_{t-1}, u_{t-1})$ des Roboters 50
- 5.2. Das Beobachtungsmodell $P(z_t \mid \xi_t)$ des Roboters, basierend auf Klassifikationen 51
- 5.3. Das Beobachtungsmodell $P(z_t \mid \xi_t)$ des Roboters, basierend auf Wahrscheinlichkeiten 52

6. Experimente **55**
- 6.1. Details zum Trainieren der Klassifikatoren und Bestimmen der Entscheidungsliste 55
  - 6.1.1. Trainieren der Klassifikatoren mit dem AdaBoost Algorithmus . . 55
  - 6.1.2. Bestimmung der Reihenfolge der Entscheidungsliste 58

6.2. Klassifikation von Beobachtungen entlang einer Trajektorie 59
6.3. Verwendung von verschiedenen schwachen Klassifikatortypen $h_j(x)$ . . . 62
6.4. Vergleich mit vorherigen Ansätzen 64
6.5. Lokalisierung mit Hilfe semantischer Informationen von Roboterpositionen 65
6.5.1. Lokalisierung mobiler Roboter 66
6.5.2. Ergebnisse 72

7. Zusammenfassung **75**
7.1. Ausblick 76

A. Technische Daten **79**
A.1. Der Roboter 79
A.2. Der Laserscanner 80
A.3. Die Kamera 81

1. Einleitung

In der Vergangenheit beschäftigten sich viele Forschungsarbeiten mit dem Problem der Bildung präziser topologischer und metrischer Karten von Umgebungen aus Sensordaten, die mit einem Roboter gesammelt wurden. Von all diesen Arbeiten betrachten jedoch nur wenige den Ansatz, semantische Informationen in diese Karten zu integrieren. Da viele Roboter konstruiert wurden, um mit Menschen zu interagieren, stellt die semantische Information von Positionen einen wichtigen Faktor dar. Sobald Roboter autonom agieren sollen und auf äußere Einflüsse reagieren müssen, kann durch solche Informationen ihre Leistungsfähigkeit erheblich gesteigert werden. Das macht sich besonders bemerkbar, wenn sie innerhalb von Gebäuden, die für Menschen entworfen und strukturiert wurden, sicher navigieren sollen. Dort kann es hilfreich sein, wenn der Roboter die verschiedenen Räume unterscheiden kann und erkennt, dass er sich beispielsweise in einem Türdurchgang befindet. Ähnliches gilt, wenn der Roboter nach seiner aktuellen Position gefragt wird. Liefert der Roboter die exakten Koordinaten, ist mit dieser Information meist wenig anzufangen. Antwortet er dagegen, er befände sich in der Küche, hat das einen wesentlich höheren Informationsgehalt für einen Menschen. Andererseits kann dem Roboter auch gesagt werden, er soll sich in den Korridor begeben. Der Roboter ist dann imstande diesen Befehl auszuführen, da ihm bekannt ist, welchen Ort er mit dem Begriff „Korridor“ in Verbindung zu bringen hat.

In dieser Arbeit möchten wir einen neuen Ansatz vorstellen, der Beobachtungen eines Roboters mit semantischen Informationen in Verbindung setzt. Dazu werden innerhalb von Gebäuden, wie beispielsweise in Abbildung 1.1 dargestellt, mit einem Laserscanner und einer Kamera Sensordaten aufgenommen. Diese werden dann anhand eines Klassifikators in typische semantische Klassen, wie Korridore, Türdurchgänge, Seminarräume oder Küchen eingeordnet. Einige dieser Orte sind dabei eher an ihrer individuellen Struktur zu erkennen, andere eher anhand den jeweiligen Einrichtungsgegenständen. So zeichnet sich beispielsweise ein Korridor durch eine Vielzahl langer Distanzmessungen und einer geringen Abweichung in der Länge zweier Messungen aus. Ebenso ist eine Kaffeemaschine typischerweise in einer Küche zu finden.

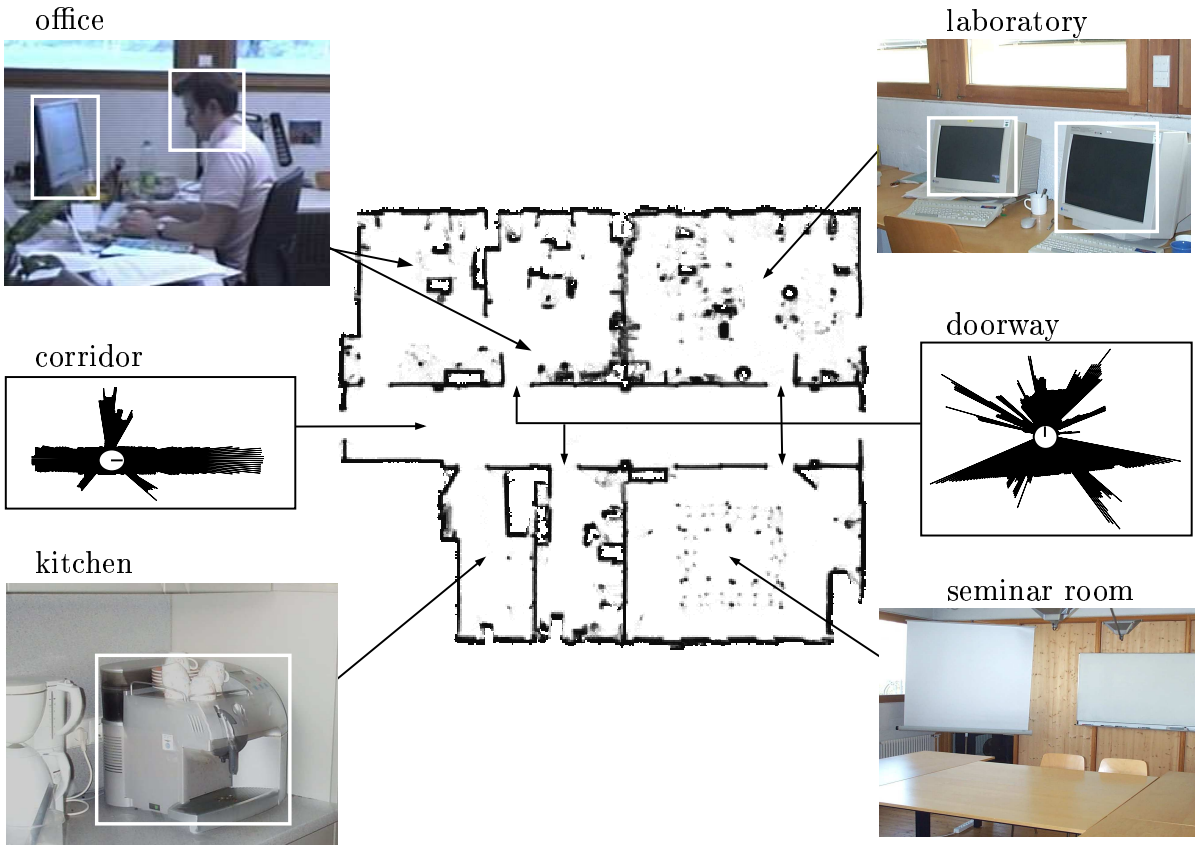


Abbildung 1.1.: Eine Büroumgebung mit verschiedenen Klassen von Räumen. Dabei sind typische Beobachtungsmerkmale abgebildet, mit Hilfe welcher auf die jeweilige Klasse geschlossen werden kann.

1.1. Zielsetzung

Die Aufgabe dieser Arbeit ist die Entwicklung eines robusten Klassifikators, um Orte in verschiedene Klassen einer typischen Büroumgebung einzuordnen. Die Grundidee besteht darin, die Position des Roboters anhand von Objekten, die aus den Kameradaten extrahiert werden, und mittels Distanzmessungen eines Laserscanners in die einzelnen Klassen einzustufen. Dazu verwenden wir den *AdaBoost* Algorithmus [Schapire und Singer, 1999] und kombinieren einfache Merkmale, die allein für eine Klassifikation nicht ausreichend sind, zu einem robusten Klassifikator. Typische Klassen und entsprechende Merkmale der verschiedenen Klassen sind in Abbildung 1.1 illustriert. Da der originale AdaBoost Algorithmus nur einen binären Klassifikator liefert, erzeugen wir aus diesen binären Klassifikatoren eine Entscheidungsliste, um zwischen mehreren verschiedenen Klassen zu unterscheiden. Dabei werden die einzelnen binären Klassifikatoren linear angeordnet, um einen Multi-Klassen Klassifikator zu erhalten.

Um die Anzahl der Fehlklassifikationen entlang der Trajektorie eines Roboters zu verringern, haben wir ein sogenanntes *Hidden Markov Model* eingesetzt. Dieses entspricht einer wahrscheinlichkeitsbasierten Klassifikation basierend auf der aktuellen Beobachtung und der Klassifikationen der vorherigen Positionen.

1.2. Der Aufbau dieser Arbeit

Diese Arbeit ist, wie folgt, gegliedert: Nach der Einleitung werden in Kapitel 2 bisherige Arbeiten vorgestellt und mit dieser Arbeit verglichen. Danach folgt ein Kapitel über den AdaBoost Algorithmus, wobei zuerst eine kleine Einführung in das Boosting allgemein gegeben wird. Ebenso werden einzelne Komponenten des AdaBoost Algorithmus mit ihren jeweiligen Anpassungen an unser Problem vorgestellt, sowie die Entscheidungsliste zum Klassifizieren mehrerer Klassen. In Kapitel 4 werden die Merkmale der Laserdaten sowie die Objekte, die aus den Kameradaten extrahiert werden, präsentiert. Kapitel 5 beschreibt dann den Einsatz des Hidden Markov Models und in Kapitel 6 werden verschiedene Experimente präsentiert. In den Experimenten zeigen wir dann, dass es möglich ist, Beobachtungen von Robotern in verschiedene Klassen einer typischen Büroumgebung, basierend auf Laser- und Kameradaten, zu unterteilen und eine Klassifikationrate von bis zu 91,2% zu erreichen. Ebenso werden wir Vergleiche zu vorherigen Arbeiten anstellen und ein Anwendungsbeispiel präsentieren, in dem die semantische Information von Roboterpositionen erfolgreich eingesetzt werden kann. Im letzten Kapitel wird schließlich die gewonnene Kenntnis zusammengefasst und ein Ausblick auf zukünftige Arbeiten und Erweiterungen gegeben.

2. Verwandte Arbeiten

Um einen Überblick über das Thema der Klassifikation von Beobachtungen eines Roboters zu geben, werden in diesem Kapitel einige Arbeiten aus der Vergangenheit vorgestellt und erläutert. Ebenso wird deren Ansatz mit unserem verglichen, um auf die Gemeinsamkeiten sowie auf die Unterschiede aufmerksam zu machen.

Es gibt einige Arbeiten, die sich mit dem Problem beschäftigen, semantische Informationen mit Orten in Verbindung zu setzen. In der Arbeit von Buschka und Saffiotti [2002] wird beispielsweise ein virtueller Sensor beschrieben, der die Umgebung automatisch in Raum- und Korridor-Regionen unterteilt. Dazu wird die Umgebung zuerst in verschiedene Regionen segmentiert, um danach für jeden Raum charakteristische Parameter zu bestimmen. Dadurch wird eine topologische Karte erzeugt und bereits besuchte Räume können wiedererkannt werden. Althaus und Christensen [2003] verwenden die Hough-Transformation um von den Sonardaten zwei parallele Linien zu detektieren, die zur Klassifikation eines Korridors verwendet werden. Anhand des Abstands dieser beiden Linien kann ebenfalls ein Türdurchgang erkannt werden. Mit diesem Erkennungssystems von Korridoren und Türdurchgängen kann eine topologische Karte zur Navigation und Kollisionsvermeidung generiert werden. Ebenso erkennen Koenig und Simmons [1998] Türdurchgänge aus Entfernungsmessungen. Im Vergleich zu all diesen Arbeiten erlaubt der Ansatz dieser Arbeit jedoch zwischen mehr Klassen zu unterscheiden, beispielsweise zwischen Doorway, Corridor, Office, Laboratory, Seminar Room und Kitchen. Zur Klassifikation ist nur eine Beobachtung einer Roboterposition notwendig. Es werden die Merkmale aus dieser Beobachtung bestimmt und nicht aus mehreren Positionen, die in der gleichen Klasse liegen.

Des Weiteren werden in der Arbeit von Kuipers und Beeson [2002] verschiedene Lerntechniken zur Erkennung markanter Positionen in Umgebungen vorgestellt. Damit können topologische Karten von Umgebungen erlernt werden. Dabei ist es zwar möglich eine topologische Karte erzeugen, aber nicht diese mit semantischen Informationen zu ergänzen. Unser Ansatz kann dagegen jede Position des Roboters mit einer Bedeutung, wie Küche oder Büro, versehen und so eine Karte mit semantischen Informationen erzeugen.

In der Arbeit von Meir und Rätsch [2003] wird eine Einführung in Boosting gegeben und diese Technik mit anderen Lernalgorithmen verglichen. Dazu wollen wir kurz auf den Vergleich zwischen Boosting und Support Vector Machinen eingehen. Beim Boosting wird eine bestimme Anzahl von Merkmalen zu einer robusten Vorhersage der Klasse kombiniert. Bei einer Support Vector Machine hingegen wird versucht im hochdimen-

sionalen Merkmalsraum eine Hyperebene zur Trennung der Klassen zu bestimmen. In beiden Vorgehensweisen wird dabei versucht den Trainingsfehler auf einer Trainingsmenge zu minimieren. In der Arbeit von Rätsch *et al.* [2002] wird dazu gezeigt, dass es eine äquivalente Transformation von Support Vector Machinen zum Boosting und umgekehrt gibt. Ebenso werden in den Arbeiten von Shumeet und Rowley [2005] und Silapachote *et al.* [2004] diese beiden Lernalgorithmen in einer Anwendung miteinander verglichen. Es werden in beiden Arbeiten vergleichbar gute Klassifikationsraten mit beiden Algorithmus erzielt. Jedoch ist die Klassifikationszeit des AdaBoost Klassifikators schneller. Ebenso ist die Wahl der Kernelfunktion und der Parameter der Support Vector Machinen kritischer im Bezug auf die Klassifikationsrate.

In dieser Arbeit haben wir uns für den AdaBoost Algorithmus entschieden, wobei es ebenso Arbeiten gibt, die andere Lernalgorithmen verwenden. Beispielsweise in der Arbeit von Anguelov *et al.* [2002] wird der EM-Algorithmus angewandt, um das Objekt Wand und Tür in einer Korridor-Umgebung anhand von Sensordaten zu erkennen. Dabei kalibriert der EM-Algorithmus die Objektparameter derart, dass alle Trainingsbeispiele bestmöglichst repräsentiert werden. Um dies zu erreichen, müssen alle Objekte vorher mit einer Wahrscheinlichkeitsverteilung modelliert werden. Im Gegensatz zu diesen Arbeiten, wird beim AdaBoost Algorithmus kein wahrscheinlichkeitsbasiertes Modell der Objekte benötigt. Stattdessen werden einfache geometrische Merkmale ohne Vorwissen vom AdaBoost Algorithmus zu einem robusten Klassifikator kombiniert. In der Arbeit von Oore *et al.* [1997] wird ein Neuronales Netz trainiert um die Position des Roboters anhand von Odometrie- und Ultraschalldaten zu schätzen. In der Arbeit von Torralba *et al.* [2003] werden globale Kameramerkmale, bestehend aus Waveletzerlegungen verschiedener Orientierungen und Skalierungen, verwendet. Diese werden dann in Ihre Hauptkomponenten zerlegt, um zwischen verschiedenen Klassen wie Büro, Korridor und Straße zu klassifizieren. Danach werden Testdaten anhand der Hauptkomponenten klassifiziert und mit Hilfe eines Hidden Markov Models versucht Fehlklassifikationen zu reduzieren. Im Vergleich zu unserem Ansatz, werden jedoch keine Objekte an den verschiedenen Orten erkannt und somit nicht aufgrund deren Auftreten auf eine Klasse geschlossen.

Im Besonderen wurde auch bereits der AdaBoost Algorithmus erfolgreich bei verschiedenen Klassifikationsproblemen verwendet. In der Arbeit von Viola und Jones [2001] werden einfachste geometrische Merkmale verwendet, um sie mit dem AdaBoost Algorithmus zu kombinieren und Gesichter in Bildern zu erkennen. In [Tieu und Viola, 2000] wird der AdaBoost Algorithmus verwendet, um ähnliche Bilder in einer Datenbank zu finden. Ebenso verwenden ihn Treptow *et al.* [2003], um die Laufbahn eines Fußballes ohne Farbinformation im Rahmen von RoboCup zu verfolgen. Die Grundidee in all diesen Arbeiten ist es, einfachste Merkmale, die aus der Differenz von Grauwerten verschiedener Bildbereiche bestehen, zu verwenden. Diese Merkmale können mit sogenannten „Integralbildern“ sehr effizient bestimmt werden. In unserer Arbeit verwenden wir jedoch Kamera- und Laserdaten in Kombination. Dennoch ist die Grundidee der

Erzeugung einfacher und schnell berechenbarer Merkmale sehr ähnlich.

In der Arbeit von Martínez Mozos *et al.* [2005] wird ein Ansatz präsentiert, der Merkmale von Laserdaten berechnet und unter Verwendung von AdaBoost Roboterpositionen in verschiedene Klassen klassifiziert. Dort werden einfache und schnell berechenbare Merkmale von Laserdaten vorgestellt, anhand derer robust zwischen den Klassen Room, Doorway und Corridor unterschieden werden kann. Im Vergleich zu unserer Arbeit werden jedoch keine Kameradaten verwendet, weshalb auch eine Klassifikation von Räumen etwa gleicher Struktur, wie Büro und Küche, nicht möglich ist. Ebenso verwenden sie kein Hidden Markov Model um die Ausgabe zu glätten. Unser Ansatz kann zwischen mehr Klassen unterscheiden und im direkten Vergleich haben wir durch eine höhere Anzahl von Merkmalen eine robustere Klassifikation. Hierzu sei auf Abschnitt 6.4 verwiesen. Da die Lasermerkmale jedoch schnell zu berechnen sind und robuste Ergebnisse liefern, konnten wir diese für unseren Ansatz übernehmen.

Die Grundidee unseres Ansatzes besteht darin, die Position des Roboters anhand von Objekten, die aus den Kameradaten extrahiert werden, und von Laserdaten des Laserscanners in die verschiedenen Klassen einzustufen. Da bereits viele unterschiedliche Verfahren für dieses Problem der Objekterkennung bekannt sind, wird nun ein kurzer Überblick der verschiedenen Ansätze gegeben. Es wird jedoch das Augenmerk auf die Erkennung von Objekten in einem zweidimensionalen Grauwertbild gelegt. Verfahren, die sich speziell mit dem Einsatz mehrerer Kameras, Bildfolgen, oder mehrdimensionalen Aufnahmen auseinandersetzen, sind dabei ausgeklammert.

Die Aufgabe eines Systems zur Objekterkennung besteht darin, reale Objekte in einem digitalen Bild automatisch zu erkennen. Üblicherweise werden aus dem Eingangsbild Merkmale extrahiert, um dann anhand eines Verfahrens auf die Zugehörigkeit zu einem Objekt zu schließen. Im Grunde lassen sich die Objekterkennungsverfahren in drei Kategorien einteilen:

- Die einfachsten Algorithmen basieren auf dem Mittelwert von Pixelwerten, wie das bereits bei dem Verfahren von Viola und Jones [2001] beschrieben wurde. Ebenso erkennen [Agarwal und Roth, 2002] die Seitenansicht von Autos in Graubildern. Dort werden ähnliche Bestandteile der Objekte gruppiert, wie beispielsweise Räder und Kotflügel, um dann das gesammte Objekt anhand des Auftretens und der Anordnung dieser Teile zu erkennen. Andererseits wird in der Arbeit von Murphy *et al.* [2003] nicht versucht Objekte vom Hintergrund zu unterscheiden, sondern das gesammte Bild als Merkmal zu verwenden, um Objekte darin zu klassifizieren.

- Bei komplexeren Algorithmen wird meist die Hauptachsentransformation oder ein Neuronales Netzwerk verwendet. In der Arbeit von Rowley *et al.* [1998] wird zum Beispiel ein Gesichtserkennungssystem basierend auf einem Neuronales Netzwerk vorgestellt. Ebenso stellen Heisele *et al.* [2001] einen verbesserten Ansatz zur Objekterkennung mit Hilfe von Support Vektor Machinen vor und in der Arbeit von

Innes *et al.* [2004] wird genetisches Programmieren verwendet um die Fehlklassifikationsraten zu verringern.

- Die letzte Kategorie beinhaltet medienübergreifende Algorithmen. Diese basieren meist auf Kontur- und Farberkennung, wie in [Garcia *et al.*, 1999]. In dieser Arbeit werden die Bilder zuerst in Regionen mit Hautfarben ähnlichen Farben eingeteilt, um dann anhand einer Waveletzerlegung die Form zu bestimmen. Es wird ein Merkmalsvektor gebildet und durch eine Metrik auf Gesichts- und Nicht-Gesichts Regionen geschlossen.

In unserem Fall sollte das System entscheiden, ob sich in der aufgenommenen Szene ein bestimmtes Objekt befindet und die jeweilige Anzahl erkannter Objekte als Ergebnis liefern. Wir haben uns deshalb für ein Verfahren nach der Idee von Viola und Jones [2001] entschieden. Es liefert gute Klassifikationsraten und ist heutzutage schon auf üblichen Rechnern in Echtzeit anwendbar. Das ist wichtig, da wir für eine robuste Klassifikation mehrere Objekte in einer Rundumsicht, bestehend aus acht Bilder, erkennen müssen. In der Arbeit von Lienhart *et al.* [2003] wird dazu ein robustes Verfahren vorgestellt, das bereits in der Open Computer Vision Library [OpenCV, 2001] implementiert ist. Wir verwenden dieses Verfahren um binäre Klassifikatoren von verschiedenen Objekten zu trainieren, die typischerweise in den verschiedenen Klassen einer Büroumgebung zu finden sind.

3. Der AdaBoost Algorithmus

In diesem Kapitel wird zuerst das Boosting erklärt, das die Grundidee des AdaBoost Algorithmus darstellt. Danach wird der AdaBoost Algorithmus selbst vorgestellt sowie einzelne Komponenten davon. Zuletzt wird auf die Entscheidungsliste eingegangen, mit der die binären Klassifikatoren des AdaBoost Algorithmus zu einem Multi-Klassen Klassifikator kombiniert werden.

3.1. Einführung in Boosting

Das generelle Problem bei der Klassifikation ist die Vorhersage einer Klasse $y \in \mathcal{Y}$ zu einem Beispiel $x \in \mathcal{X}$. Dabei entspricht $\mathcal{X}$ dem Raum aller Beispiele und $\mathcal{Y}$ den möglichen Klassen.

Für dieses Problem stellt *Boosting* eine generelle Methode bereit, um die Präzision eines gegebenen Lernalgorithmus zu verbessern [Freund und Schapire, 1999]. Die Grundidee von Boosting besteht darin, verschiedene schwache Hypothesen $\{h_1, h_2, \ldots, h_S\}$ zu einer starken Hypothese H zu kombinieren. Eine Hypothese im allgemeinen stellt dabei eine Funktion $h : \mathcal{X} \to \mathcal{Y}$ dar, die die Klassenzugehörigkeit von Beispielen x vorhersagt. Dabei sei vorausgesetzt, dass die schwachen Hypothesen eine bessere Güte als eine Zufallsentscheidung haben. Durch diese Kombination der schwachen Hypothesen soll erreicht werden, dass die Effizienz der starken Hypothese besser ist als die Effizienz jeder einzelnen schwachen Hypothese.

Im Allgemeinen läuft Boosting wie folgt ab: Der Boosting Algorithmus benötigt eine Menge von Trainingsbeispielen $(x_1, y_1), \ldots, (x_N, y_N)$. Dabei entspricht x_n einem Element aus dem Beispielraum $\mathcal{X}$ und y_n einer Zuweisung zu einer Klasse aus $\mathcal{Y}$. Im binären Fall setzen wir $\mathcal{Y} = \{+1, -1\}$, entsprechend für positive und negative Beispiele. In jeder Iteration $t = 1, \ldots, T$ ruft das Boosting einen gegebenen sogenannten *schwachen Lerner* auf. Ein schwacher Lerner entspricht dabei einer Prozedur, die aus der Menge aller schwachen Hypothesen eine Hypothese h_t nach einem bestimmten Kriterium bestimmt. All diese schwachen Hypothesen werden letztlich zu einer starken Hypothese H kombiniert. Der Grundgedanke des Boosting besteht nun darin, eine Verteilung von Gewichten über den Trainingsbeispielen zu erstellen. Das Gewicht eines Beispiels x_n in der Iteration t ist gegeben durch $D_t(n)$ und soll den Einfluss des jeweiligen Beispiels bei der Wahl der schwachen Hypothese beschreiben. Anfangs werden alle Gewichte gleich initialisiert.

In den einzelnen Iteration werden sie derart modifiziert, dass der schwache Lerner seine Wahl einer schwachen Hypothese nach diesen Gewichten richtet. So werden schwachen Hypothese bevorzugt, die schwierige korrekt zu klassifizierende Trainingsbeispiele richtig bewerten. Die Güte jeder schwachen Hypothese h_t wird durch einen Fehler

$$\epsilon_t = \sum_{i \in \{k | h_t(x_k) \neq y_k\}} D_t(i)$$

dargestellt. Es sei angemerkt, dass der Fehler ϵ_t hinsichtlich der Verteilung D_t der Trainingsbeispiele bemessen wird. Anhand dieses Fehlers wird schließlich für jede schwache Hypothese h_t eine Gewichtung α_t bestimmt. Letztlich werden die schwachen Hypothesen h_t mit den geringsten Fehlern ϵ_t aller Iterationen zu einer starken Hypothese H kombiniert. Im binären Fall kann dies durch eine gemischte Summe

$$H(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$$

erfolgen.

Ebenso kann gezeigt werden, dass, falls die schwachen Hypothesen eine besser Güte als eine Zufallsentscheidung haben, der Trainingsfehler exponentiell schnell mit der Zahl der Iterationen fällt [Schapire, 2001].

3.2. AdaBoost als binärer Klassifikator

In diesem Abschnitt wollen wir uns mit dem *AdaBoost* Algorithmus beschäftigen. Er wurde erstmals von Freund und Schapire [1995] vorgestellt und ist heute einer der bekanntesten Boosting Algorithmen. Bezogen auf das im vorherigen Abschnitt 3.1 vorgestellte Boosting entsprechen die im folgenden auftretenden schwachen und starken Klassifikatoren den dort bereits erwähnten Hypothesen.

Nach der Idee des Boosting werden aus vielen verschiedenen schwachen Klassifikatoren h_j eine geringe Anzahl ausgewählt, um daraus einen starken Klassifikator H zu generieren. Der AdaBoost Algorithmus benötigt dafür als Eingabe eine Menge von Trainingsbeispielen $(x_n, y_n), n = 1, \ldots, N$, wobei jedes x_n aus dem Beispielraum $\mathcal{X}$ kommt und y_n das Beispiel x_n mit einer Bezeichnung aus $\mathcal{Y}$ kennzeichnet. Da wir uns im folgenden mit binären Klassifikatoren beschäftigen, setzen wir $\mathcal{Y} = \{+1, -1\}$, entsprechend für positive und negative Beispiele. In jeder Iteration $t = 1, \ldots, T$ verwendet AdaBoost einen schwachen Lerner um den jeweils besten schwachen Klassifikator zu bestimmen. Generell hat ein schwacher Klassifikator die Form $h : \mathcal{X} \to \mathbb{R}$. Dabei entspricht das Vorzeichen von $h(x)$ der Zuordnung des Beispiels x zu einer Klasse aus $\mathcal{Y} = \{+1, -1\}$ und $|h(x)|$ der Überzeugung der Klassifikation. In dieser Arbeit haben wir die Zielmenge von h auf

einen bestimmten Bereich $h : \mathcal{X} \to \{-1, +1\}$ begrenzt. Die Wahl des besten schwachen Klassifikators geschieht mit Hilfe der Trainingsbeispiele und einer darüber gewichteten Verteilung D_t. Um die Verteilung D_t über den Beispielen in jeder Iteration anzupassen, ist die Berechnung eines Fehlers ϵ_t ein wichtiger Bestandteil des Verfahrens. Es gibt dazu verschiedene Ansätze und in dieser Arbeit wird kein Fehler ϵ_t, sondern ein allgemeiner Faktor α_t verwendet. Die allgemeine Version des AdaBoost Algorithmus von Schapire und Singer [1999] ist in Algorithmus 1 abgebildet.

Noch offene Fragen sind die Funktionsweise des schwachen Lerners zur Wahl eines schwachen Klassifikators h_j in Schritt 2 und die Bestimmung des Faktors α_t in Schritt 3 des Algorithmus 1. Darauf wird in Kapitel 3.3 eingegangen. Des Weiteren werden in Kapitel 3.4 die von uns verwendeten schwachen Klassifikatoren h_j vorgestellt.

Um letztlich eine robuste Klassifikation zu erhalten, ist es manchmal von Vorteil, ein Maß für die Klassifikation zu einer bestimmten Klasse zu haben. Dieses Maß soll dabei eine Art Wahrscheinlichkeit darstellen, mit der die Klassifikation sich des Ergebnisses sicher ist. Dabei kann sowohl ein Maß für die positive als auch für die negative Klassifikation von Interesse sein. Friedman *et al.* [1998] haben dafür eine Methode für den AdaBoost entwickelt, um die Wahrscheinlichkeit der Klassifikation zu schätzen. Im Folgenden setzen wir zur vereinfachten Schreibweise

$$g(x) = \sum_{t=1}^{T} \alpha_t h_t(x), \tag{3.1}$$

wodurch sich für den starken Klassifikator $H(x) = sign\big(g(x)\big)$ ergibt. Zu einem Beispiel x kann die Wahrscheinlichkeit bei einer positiven Klassifikation $H(x) = 1$ durch

$$P[H(x) = 1 \mid x] = \frac{e^{g(x)}}{e^{g(x)} + e^{-g(x)}} \tag{3.2}$$

geschätzt werden. Ebenso für eine negative Klassifikation $H(x) = -1$ durch

$$P[H(x) = -1 \mid x] = \frac{e^{-g(x)}}{e^{g(x)} + e^{-g(x)}}. \tag{3.3}$$

3.3. Bestimmung der Gewichte α_t

Das Ziel des AdaBoost Algorithmus besteht darin, einen robusten Klassifikator H zu erzeugen, der die Trainingsbeispiele bestmöglichst klassifiziert. In der vorherigen Sektion wurde der allgemeine AdaBoost Algorithmus vorgestellt und im Folgenden wollen wir nun den Trainingsfehler des Klassifikators H bei den Trainingsbeispielen abschätzen. Das Ziel dieses Vorgehens liegt darin, eine geeignete Berechnung des Faktors α_t zu erhalten und ein Kriterium für die beste Wahl der schwachen Klassifikatoren h_t in den einzelnen Iterationen.

Algorithmus 1 Der allgemeine AdaBoost Algorithmus

Eingabe: Menge von Trainingsbeispielen $(x_1, y_1), \ldots, (x_N, y_N)$,
$x_n \in \mathcal{X}$ und $y_n \in \{+1, -1\}$,
entsprechend für ein positives oder negatives Beispiel.

Initialisierung der Gewichte $D_1(n) = \frac{1}{2l}, \frac{1}{2m}$, abhängig vom Wert y_n.
Dabei sei l und m die Anzahl der positiven bzw. negativen Beispiele.

for $t = 1, \ldots, T$ **do**

1. Trainieren der schwachen Klassifikatoren h_j
anhand der Gewichte D_t.

2. Wähle einen schwachen Klassifikator h_j aus und setzte $h_t = h_j$

3. Berechne $\alpha_t \in \mathbb{R}$

4. Aktualisierung der Gewichte $D_{t+1}(n)$

$$D_{t+1}(n) = \frac{D_t(n) e^{-\alpha_t y_n h_t(x_n)}}{A_t},$$

wobei A_t einem Normalisierungsfaktor

$$A_t = \sum_{i=0}^{N} D_t(i) e^{-\alpha_t y_i h_t(x_i)}$$

entspricht.

end for

Ausgabe: Der starke Klassifikator ist letztlich gegeben durch

$$H(x) = \operatorname{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x) \right).$$

Der Trainingsfehler kann durch

$$\frac{F(H)}{N} \leq \prod_{t=1}^{T} A_t \tag{3.4}$$

abgeschätzt werden, wobei N der Anzahl der Trainingsbeispiele, A_t dem Normalisierungsfaktor aus Schritt 4 des Algorithmus 1 und

$$F(H) = |\{n \mid H(x_n) \neq y_n\}|$$

der Anzahl der Fehlklassifikationen auf den Trainingsbeispielen entspricht.

Die Beweisidee wird nun kurz erklärt, ist aber auch in [Schapire und Singer, 1999] nachzulesen. Im Falle einer Fehlklassifikation $H(x_n) \neq y_n$ gilt $y_n g(x_n) \leq 0$. Dabei entspricht $g(x)$ der in Gleichung (3.1) gesetzten Kurzschreibweise. Daraus folgt für solch einen Fall, dass $1 \leq e^{-y_n g(x_n)}$. Da $e^z > 0 \; \forall z \in \mathbb{R}$ kann die Anzahl aller Fehlklassifikationen durch

$$F(H) \leq \sum_{i=0}^{N} e^{-y_i g(x_i)} \tag{3.5}$$

abgeschätzt werden. Ebenfalls kann man $D_{T+1}(n)$ rekursiv auflösen und erhält

$$D_{T+1}(n) = \frac{e^{-\sum_t \alpha_t y_n h_t(x_n)}}{N \prod_t A_t} = \frac{e^{-y_n g(x_n)}}{N \prod_t A_t}. \tag{3.6}$$

Durch Einsetzen von Gleichung (3.6) in (3.5) erhält man die Abschätzung aus Gleichung (3.4).

Daraus folgt nun, dass für einen bestmöglichen Klassifikator H der Normalisierungsfaktor A_t in jeder Iteration zu minimieren ist. Auf Grund dieser Kenntnis wird nun der Faktor α_t berechnet.

In Anlehnung an [Schapire und Singer, 1999] wird versucht A_t als Funktion von α_t zu minimieren. Sind die schwachen Klassifikatoren auf den Bereich $h \in [-1, +1]$ beschränkt, läßt sich A_t wie folgt umformen

$$\begin{aligned} A_t &= \sum_{i=0}^{N} D_t(i) e^{-\alpha_t y_i h_t(x_i)} \\ &\leq \sum_{i=0}^{N} D_t(i) \left(\frac{1 + y_i h_t(x_i)}{2} e^{-\alpha_t} + \frac{1 - y_i h_t(x_i)}{2} e^{\alpha_t} \right). \end{aligned} \tag{3.7}$$

Falls, wie in dieser Arbeit, $h \in \{-1, +1\}$, gilt sogar die Gleichheit. Wird nun der Wert von α_t, der den rechten Ausdruck in Gleichung (3.7) minimiert, analytisch ermittelt, ergibt sich

$$\alpha_t = \frac{1}{2} ln \left(\frac{1 + r_t}{1 - r_t} \right), \tag{3.8}$$

dabei entspricht

$$r_t = \sum_{i=0}^{N} D(i) y_i h_t(i). \tag{3.9}$$

Durch Einsetzen in Gleichung (3.7) erhält man eine obere Schranke

$$A_t \leq \sqrt{1 - r_t^2}.$$

In Verbindung mit Gleichung (3.4) ergibt sich somit ein Trainingsfehler von

$$\frac{F(H)}{N} \leq \prod_{t=1}^{T} \sqrt{1 - r_t^2}. \tag{3.10}$$

Als Ergebnis dieser Überlegungen erhalten wir ein Kriterium für die Wahl des schwachen Klassifikators h_t in jeder Iteration und eine Vorschrift zur Berechnung des Faktors α_t. Wir berechnen α_t nach Gleichung (3.8) um einen möglichst geringen Trainingsfehler zu erhalten. Der Trainingsfehler kann dann durch Gleichung (3.10) abgeschätzt werden. Um wiederum diese Abschätzung zu minimieren, ist $|r_t|$ zu maximieren. Es wird $|r_t|$ nach Gleichung (3.9) berechnet und der schwache Klassifikator h_j in den einzelnen Iterationen gewählt, der $|r_t|$ maximiert. Dadurch wird der bestmöglichste Klassifikator H bezüglich der Trainingsmenge gebildet. Dabei entspricht r_t einem natürlichen Maß der Korrelation zwischen der Entscheidung des Klassifikators h_t und den Kennzeichnungen der Beispiele y_n, basierend auf der Verteilung D_t. Der in dieser Arbeit verwendete AdaBoost Algorithmus ist in Algorithmus 2 abgebildet.

3.4. Bildung der schwachen Klassifikatoren

Bisher wurde der AdaBoost Algorithmus vorgestellt, auf welche Weise die Gewichte α_t berechnet und welche schwachen Klassifikatoren in den einzelnen Iterationen gewählt werden. Zur Ausführung des AdaBoost Algorithmus fehlen uns jetzt nur noch die schwachen Klassifikatoren, auf die wir in diesem Kapitel eingehen wollen.

Ein schwacher Klassifikator ist durch eine Funktion $h_t : \mathcal{X} \rightarrow \mathcal{Y}$ gegeben, die einem Beispiel x eine Klasse zuordnet. Unser Klassifikator soll anhand von Laser- und Kameradaten eine Beobachtung eines Roboters einer Klasse zuweisen. Daher wurde zur Bildung der schwachen Klassifikatoren h_j die Idee von Viola und Jones [2001] aufgegriffen und skalare Merkmale f_j der Laser- und Kameradaten verwendet. Ein Merkmal besteht dabei aus einer Funktion, die basierend auf den Laser- oder Kameradaten einen skalaren Faktor liefert. Diese Merkmale werden im einzelnen im folgenden Kapitel 4 vorgestellt.

Algorithmus 2 Der verwendete AdaBoost Algorithmus

Eingabe: Menge von Trainingsbeispielen $(x_1, y_1), \ldots, (x_N, y_N)$,
$x_n \in \mathcal{X}$ und $y_n \in \{+1, -1\}$,
entsprechend für ein positives oder negatives Beispiel.

Initialisierung der Gewichte $D_1(n) = \frac{1}{2l}, \frac{1}{2m}$, abhängig vom Wert y_n.
Dabei sei l und m die Anzahl der positiven bzw. negativen Beispiele.

for $t = 1, \ldots, T$ **do**

1. Trainieren der schwachen Klassifikatoren h_j
anhand der Gewichte D_t.

2. Berechne für jeden schwachen Klassifikator h_j:

$$r_j = \sum_{i=0}^{N} D_t(i) y_i h_j(x_i),$$

wobei $h_j(x_n) \in \{-1, +1\}$.

3. Wähle den schwachen Klassifikator h_j aus, der $|r_j|$ maximiert,
und setze $(h_t, r_t) = (h_j, r_j)$.

4. Aktualisierung der Gewichte $D_{t+1}(n)$

$$D_{t+1}(n) = \frac{D_t(n) e^{-\alpha_t y_n h_t(x_n)}}{A_t},$$

wobei

$$\alpha_t = \frac{1}{2} \log \left(\frac{1 + r_t}{1 - r_t} \right)$$

und A_t einem Normalisierungsfaktor

$$A_t = \sum_{i=0}^{N} D_t(i) e^{-\alpha_t y_i h_t(x_i)}$$

entspricht.

end for

Ausgabe: Der starke Klassifikator ist letztlich gegeben durch

$$H(x) = \operatorname{sign} \left(\sum_{t=1}^{T} \alpha_t h_t(x) \right).$$

Für den AdaBoost Algorithmus wurden dann aus diesen Merkmalen zwei verschiedene Arten von schwachen Klassifikatoren erzeugt. Der erste Typ der schwachen Klassifikatoren wird in verschiedenen Arbeiten verwendet (vergleiche dazu [Viola und Jones, 2001], [Martínez Mozos *et al.*, 2005], [Treptow *et al.*, 2003]) und hat die Form

$$h_j^1(x) = \begin{cases} +1 & \text{falls } p_j f_j(x) < p_j \theta_j \\ -1 & \text{sonst,} \end{cases} \tag{3.11}$$

wobei θ_j ein Schwellwert darstellt und p_j mit -1 oder $+1$ die Ausrichtung der Ungleichung präsentiert. Der zweite Typ der schwachen Klassifikatoren hat die Form

$$h_j^2(x) = \begin{cases} p_j & \text{falls } \theta_j^1 < f_j(x) < \theta_j^2 \\ -p_j & \text{sonst,} \end{cases} \tag{3.12}$$

wobei θ_j^1 und θ_j^2 als Schwellwerte zur Begrenzung eines Intervalls dienen und p_j mit -1 oder $+1$ festlegt, ob eine positive Klassifikation außerhalb oder innerhalb des Intervalls erfolgt.

Für beide Typen von schwachen Klassifikatoren $h(x)$ entspricht $+1$ oder -1 einer positiven beziehungsweise negativen Klassifikation des Beispiels x. Aus allen Merkmalen der Laser- und Kameradaten ergeben sich dadurch zwei verschiedene schwache Klassifikatoren, einen in der Form des ersten Typs und einen in der Form des zweiten. In jeder Iteration t werden für die schwachen Klassifikatoren $h_j^1(x)$ und $h_j^2(x)$ die Parameter (p_j,θ_j) oder $(p_j,\theta_j^1,\theta_j^2)$ derart ermittelt, dass eine möglichst geringe Anzahl an Fehlklassifikationen der Trainingsbeispiele basierend auf der Verteilung D_t auftritt.

Es sei darauf hingewiesen, dass die Erzeugung der schwachen Klassifikatoren $h_j^2(x)$ nach Gleichung (3.12) einen neuen Ansatz darstellt, der bisher in keiner Arbeit Verwendung fand. Die Erzeugung dieses weiteren Typs basiert auf der Idee, dass bestimmte Klassen anhand eines Intervalls besser beschrieben werden können. So wird erwartet, dass in einem Büro mehr Monitore als in einer Küche zu finden sind, jedoch weniger als in einem Großraumbüro. Diese Eigenschaft könnte nicht ausschließlich mit schwachen Klassifikatoren des ersten Typs $h_j^1(x)$ modelliert werden. Dazu sei auf Abbildung 3.1 verwiesen, die diese Eigenschaft grafisch veranschaulicht. Anhand der oberen beiden Grafiken aus Abbildung 3.1 können die Parameter (p_j,θ_j) und $(p_j,\theta_j^1,\theta_j^2)$ der schwachen Klassifikatoren bezüglich der skalaren Merkmale $f_l(x)$ und $f_m(x)$ bestimmt werden. Es wird für diese Merkmale der skalare Wert für alle Trainingsdaten bestimmt und in sortierter Reihenfolge auf der x-Achse angezeigt. Entsprechend ergibt sich der dazugehörige Wert auf der y-Achse durch die Klasse $y_n \in \{-1,+1\}$ des Beispiels x_n und dessen Gewichtung $D_t(n)$ der entsprechenden Iteration. Nun werden die Parameter der schwachen Klassifikatoren $h_l^1(x), h_l^2(x), h_m^1(x)$ und $h_m^2(x)$ derart gewählt, dass möglichst viele Beispiele richtig klassifiziert werden. Die klassifizierten Bereiche der einzelnen schwachen Klassifikatoren sind jeweils in den darunterliegenden Grafiken zu sehen. Dabei sei angemerkt, dass die schwachen Klassifikatoren $h^2(x)$ im Vergleich zu $h^1(x)$ jeweils nur ein Intervall als positiv oder

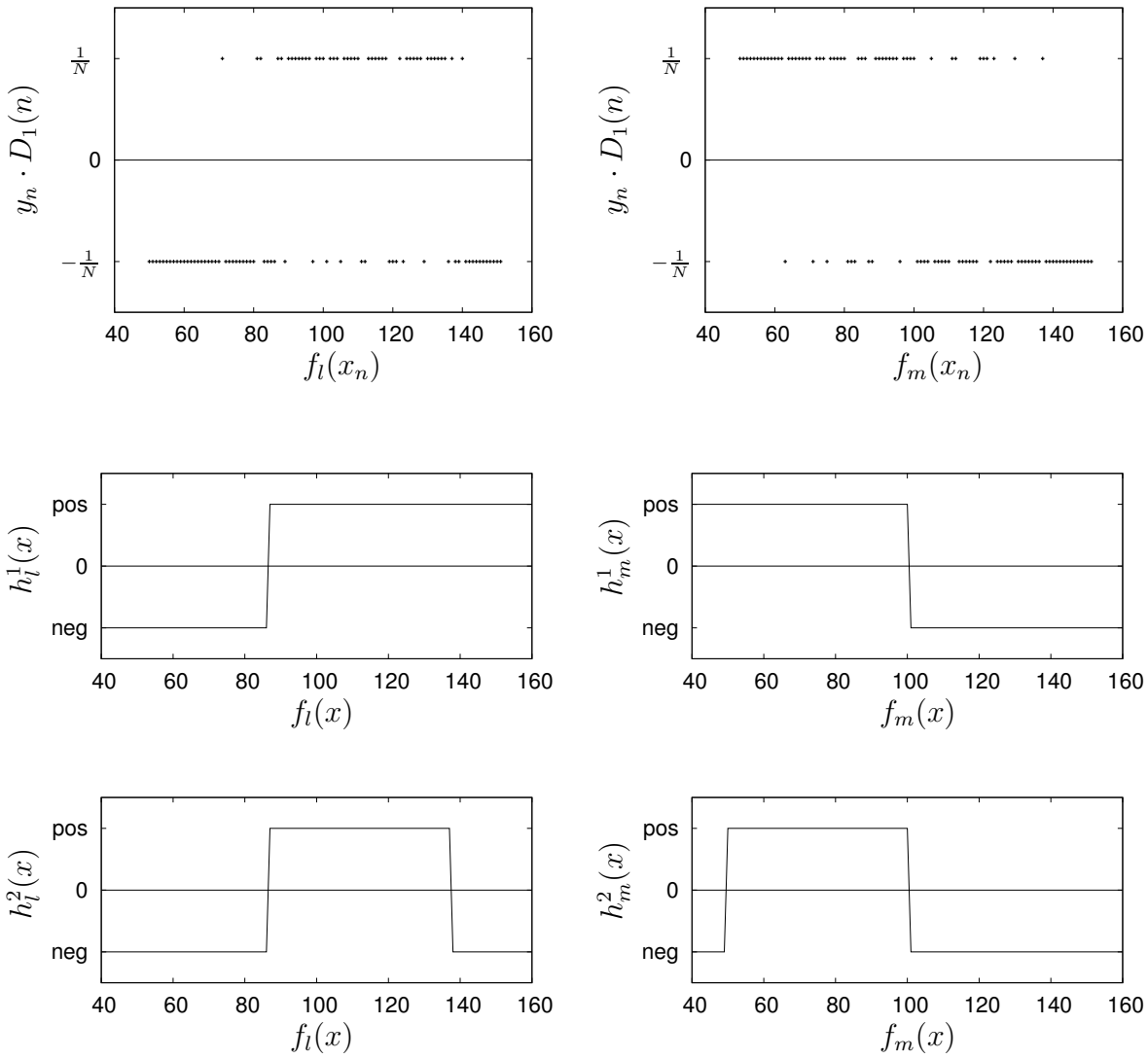


Abbildung 3.1.: In der oberen Zeile ist je eine Grafik zur Bestimmung der Parameter der schwachen Klassifikatoren $h_l(x)$ und $h_m(x)$ abgebildet. Auf der x-Achse sind die Werte der skalaren Merkmale $f_l(x_n)$ und $f_m(x_n)$ für alle Trainingsdaten x_n in sortierter Reihenfolge dargestellt. Auf der y-Achse sind die entsprechenden Klassen y_n multipliziert mit der Gewichtung $D_1(n)$ aufgezeigt. Dabei entsprechen in diesen Beispielen alle Gewichte den Werten $\frac{1}{N}$. Darunter sind jeweils die schwachen Klassifikatoren $h_l(x)$ und $h_m(x)$ mit den Bereichen, in denen eine positive und negative Klassifikation für ein Testbeispiel x erfolgt, zu sehen. Dabei erkennt man deutlich, dass die schwachen Klassifikatoren $h^2(x)$ im Vergleich zu $h^1(x)$ jeweils nur ein Intervall als positiv oder negativ klassifizieren.

Klassifikator	Parameter			Fehler
$h_l^1(x)$	θ_l=86	–	p_l=1	27.45%
$h_l^2(x)$	θ_l^1=86	θ_l^2=138	p_l=1	14.71%
$h_m^1(x)$	θ_m=101	–	p_m=1	17.65%
$h_m^2(x)$	θ_m^1=49	θ_m^2=101	p_m=1	17.65%

Tabelle 3.1.: Die Parameter der schwachen Klassifikatoren und der jeweilige Trainingsfehler, bestimmt anhand der oberen beiden Grafiken aus Abbildung 3.1.

negativ klassifizieren. Die Parameter der einzelnen Klassifikatoren sind in Tabelle 3.1 zu sehen.

Hierbei sei darauf hingewiesen, dass der Klassifikator $h_l^2(x)$ die Trainingsdaten besser approximiert als $h_l^1(x)$ und die positive und negative Klasse exakter repräsentiert. Dies kann bei der Klassifikation von Roboterpositionen anhand von Objekten zu besseren Resultaten führen, da einige Objekte in verschiedenen Klassen in bestimmter Anzahl vorkommen. In einigen Fällen kann dies jedoch zuviel an Information beinhalten. Bei Anwendung auf begrenzte Mengen, wie es bei der Verwendung mit AdaBoost der Fall ist, stellt der erste Typ nach Gleichung (3.11) eine Teilmenge des zweiten Typs dar. Dazu sei auf die rechte Seite der Abbildung 3.1 verwiesen. Beide schwachen Klassifikatoren beschreiben die Trainingsdaten gleich und erzielen denselben Klassifikationfehler. Jedoch würde Klassifikator $h_m^2(x)$ ein Testbeispiel x' mit dem Merkmal $f_m(x') = 45$ als negativ einstufen. Schaut man sich aber die dazugehörenden Werte der Trainingsdaten an, würde man intuitiv eine solches Beispiel als positiv einstufen. Dies würde Klassifikator $h_m^1(x)$ in diesem Fall auch tun und liefert in solchen Fällen eine robustere Klassifikation. Bei der Programmierung sollte man darauf achten, dass in diesen Fällen der Klassifikator $h^1(x)$ nach Gleichung (3.11) bevorzugt verwendet werden sollte. Des Weiteren sei an dieser Stelle auf das Experiment 6.3 verwiesen, wo unter Verwendung der verschiedenen schwachen Klassifikatoren die Klassifikationsraten miteinander verglichen werden.

3.5. Multi-Klassen Klassifikator

Der Klassifikator, der bisher mit dem AdaBoost Algorithmus erhalten wird, ist ausschließlich in der Lage, zwischen zwei verschiedenen Klassen zu unterscheiden, nämlich positiven und negativen Beispielen. In typischen Indoor Umgebungen muss der Roboter jedoch in der Lage sein, mehr als nur zwei Klassen zu erkennen. Aus diesem Grund muß ein Multi-Klassen Klassifikator konstruiert werden. In der Arbeit von Freund und Schapire [1997] wird zum Beispiel eine Erweiterung des AdaBoost Algorithmus zur Verarbeitung mehrerer Klassen vorgestellt. Dieser Algorithmus wird *AdaBoost.M2* genannt

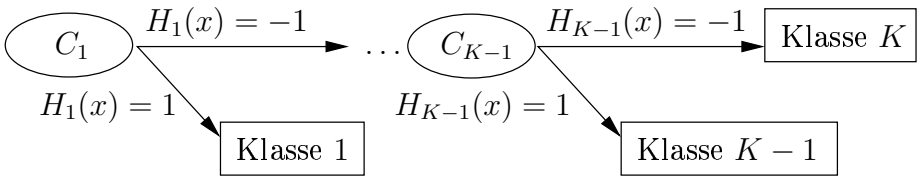


Abbildung 3.2.: Eine Entscheidungsliste H für K Klassen unter Verwendung von $K-1$ binären Klassifikatoren $\{C_1, C_2, \ldots, C_{K-1}\}$. Jeder Klassifikator C_k entspricht dabei einem binären AdaBoost Klassifikatoren. Die Ausgabe „Klasse k“ repräsentiert dabei die Ausgabe entsprechend des Klassifikators C_k, in dieser Arbeit somit der Bezeichnung der Klasse k.

und trainiert einen starken Klassifikator H, der in der Lage ist, zwischen mehreren Klassen $\mathcal{Y} = \{1, 2, \ldots, K\}$ zu unterscheiden. Ein anderer Ansatz besteht darin, aus einzelnen binären Klassifikatoren eine Entscheidungsliste zu erzeugen, die *Sequentieller AdaBoost* genannt wird. Martínez Mozos *et al.* [2005] vergleichen bei der Klassifikation von Indoor Umgebungen diese beiden Verfahren miteinander und kommen zu dem Ergebnis, dass der Sequentielle AdaBoost die bessere Klassifikationsrate erreicht. Aus diesem Grund haben wir dieses Verfahren gewählt. Im folgenden Kapitel werden die Grundideen dieses Verfahrens kurz vorgestellt.

3.5.1. Sequenz binärer AdaBoost Klassifikatoren

Eine Möglichkeit einen Multi-Klassen Klassifikator zu erzeugen besteht darin, $K-1$ binäre Klassifikatoren $\{C_1, C_2, \ldots, C_{K-1}\}$ zu verwenden, um daraus eine Entscheidungsliste H zur Klassifikation von K Klassen zu konstruieren. Diese Liste entscheidet für eine Beobachtung x, welcher Klasse $\mathcal{Y} = \{1, 2, \ldots, K\}$ diese Beobachtung angehört. Zur Klassifikation einer Beobachtung verwenden die einzelnen binären Klassifikatoren C_k einen binären starken Klassifikator $H_k : \mathcal{X} \to \mathcal{Y} = \{-1, +1\}$, der mit dem AdaBoost Algorithmus aus Kapitel 3.2 trainiert wurde. Zum Trainieren dieser binären Klassifikatoren C_k wurden positive Trainingsbeispiele (x_p, y_p) mit $y_p = k$ und negative (x_n, y_n) mit $y_n \in \mathcal{Y} \backslash \{k\}$ verwendet. In Abbildung 3.2 ist eine solche Entscheidungsliste dargestellt.

Um die Klasse einer Beobachtung x durch eine solche Entscheidungsliste zu bestimmen, können zwei Vorgehensweisen angewandt werden. Zum einen könnte jedes Element C_k der Entscheidungsliste als binärer Klassifikator entscheiden, ob die Beobachtung x ein positives oder negatives Beispiel der Klasse $k \in \mathcal{Y}$ darstellt. Falls der Klassifikator C_k eine Beobachtung für positiv befindet ($H_k(x) = 1$), wird die Klassifikation beendet und die Bezeichnung der Klasse k als Ergebnis geliefert. Andernfalls wird die Beobachtung x an das nächste Element C_{k+1} in der Entscheidungsliste weitergegeben. Ist der letzte binäre Klassifikator C_{K-1} der Liste erreicht, wird bei einer positiven Klassifikation die

Algorithmus 3 Die Prozedur zur Klassifikation der Klasse einer Beobachtung x durch eine Entscheidungsliste

Eingabe: ein Beobachtung x,
eine Bezeichnung für jede Klasse $k \in \mathcal{Y} = \{1, 2, \ldots, K\}$,
binären Klassifikatoren $\{C_1, C_2, \ldots, C_{K-1}\}$
for $k = 1, \ldots, K-1$ **do**
if $H_k(x) = 1$ **then**
return Bezeichnung der Klasse k
end if
end for
return Bezeichnung der Klasse K

Bezeichnung der Klasse $K-1$ als Ergebnis geliefert, andernfalls die der Klasse K. Im Vergleich zur Verwendung von K binärer Klassifikatoren, liegt der Unterschied darin, dass bei einer negativen Klassifikation aller Elemente die Klassifikation der Beobachtung x nicht in die Klasse unknown fallen würde. Der grundlegende Ablauf zur Klassifikation einer Beobachtung x ist in Algorithmus 3 dargestellt.

Ebenso könnte man eine solche Entscheidungsliste verwenden, um eine Wahrscheinlichkeitsverteilung über alle Klassen zu ermitteln. Der Unterschied zur vorher beschriebenen Vorgehensweise liegt darin, dass nicht nur eine Klasse als Ergebnis geliefert wird, sondern ein Wahrscheinlichkeitswert für jede einzelne Klasse. Unter Verwendung der Gleichungen (3.2) und (3.3) könnte für jeden binären Klassifikator C_k die Wahrscheinlichkeit der Überzeugung berechnet werden. Dann hätte man jedoch nur einzelne Werte von jeder Klasse, aber keine globale Aussage über alle Klassen. Verwendet man jedoch eine Entscheidungsliste, erhält man basierend auf den Wahrscheinlichkeiten der einzelnen binären Klassifikatoren, durch

$$P(k) = P(C_k) \cdot \prod_{i=0}^{k-1} (1 - P(C_i)) \tag{3.13}$$

eine Wahrscheinlichkeitsverteilung über alle Klassen. Dabei entspricht $P(k)$ der Klassenzugehörigkeit zur Klasse k und $P(C_k)$ der positiven Klassifikationswahrscheinlichkeit von Klasse k, die sich durch

$$P(C_k) = \begin{cases} P[H_k(x) = 1 \mid x] & \text{falls } H_k(x) = 1 \\ 1 - P[H_k(x) = -1 \mid x] & \text{falls } H_k(x) = -1 \end{cases}$$

ergibt. Dabei wird $P[H_k(x) = 1 \mid x]$ und $P[H_k(x) = -1 \mid x]$ entsprechend nach Gleichung (3.2) oder (3.3) berechnet. In diesem Fall würde nach einer positiven Klassifikation der Beobachtung x die Klassifikation nicht beendet werden, sondern würde immer bis

Algorithmus 4 Die Prozedur zur Berechnung einer Verteilung über alle Klassen bezüglich einer Beobachtung x

Eingabe: ein Beobachtung x,
binären Klassifikatoren $\{C_1, C_2, \ldots, C_{K-1}\}$
Initialisierung der Verteilung $D = \{d_1, d_2, \ldots, d_K\}$ mit $d_k = 1{,}0$
for $k = 1, \ldots, K-1$ **do**
Berechne $H_k(x)$
Setze

$$P(C_k) = \begin{cases} P[H_k(x) = 1 \mid x] & \text{falls } H_k(x) = 1 \\ 1 - P[H_k(x) = -1 \mid x] & \text{falls } H_k(x) = -1 \end{cases}$$

$d_k = d_k \cdot P(C_k)$
for $l = k+1, \ldots, K$ **do**
$d_l = d_l \cdot \big(1 - P(C_k)\big)$
end for
end for
Ausgabe: Verteilung D

zum letzten Element weitergehen. Durch aufmultiplizieren der jeweiligen Wahrscheinlichkeiten der einzelnen Klassifikatoren ist auch gewährleistet, dass eine Wahrscheinlichkeitsverteilung über alle Klassen erhalten wird. Der grundlegende Ablauf zur Berechnung einer Verteilung von einer Beobachtung x ist in Algorithmus 4 dargestellt.

Im Folgenden können nun beide Varianten des Multi-Klassen Klassifikator H, verwendet werden, um einer Beobachtung x eine Klasse aus $\mathcal{Y}$ zuzuordnen. Dabei wird zur Bestimmung der Klasse eines Beispiels bei der ersten Variante die entsprechende Klasse direkt geliefert. In der zweiten Variante wird dementsprechend eine Verteilung über alle Klassen berechnet, wovon das Maximum zu Klassifikation des Beispiels verwendet wird. Bei der späteren Verwendung eines Multi-Klassen Klassifikators wird an den entsprechenden Stellen jeweils angegeben, ob eine Klassifikator nach dem Algorithmus 3 angewandt oder eine Verteilung über alle Klassen nach Algorithmus 4 berechnet wird.

3.5.2. Reihenfolge der Klassifikatoren in einer Entscheidungsliste

Eine wichtige Fragestellung im Bezug auf eine solche Entscheidungsliste, betrifft die Reihenfolge in der die einzelnen binären Klassifikatoren $\{C_1, C_2, \ldots, C_{K-1}\}$ angeordnet werden. Diese Reihenfolge kann einen großen Einfluß auf die Klassifikationsrate der Entscheidungsliste haben, da die Klassifikatoren C_k in der Regel mit einer Fehlerrate behaftet sind. An einem kleinen Beispiel für die Klassifikation einer Klasse nach Algorithmus 3 wird dieses Problem nun kurz erläutert. Bei der Berechnung einer Verteilung nach Algorithmus 4 ist dies ebenfalls leicht ersichtlich, da die Wahrscheinlichkeiten von

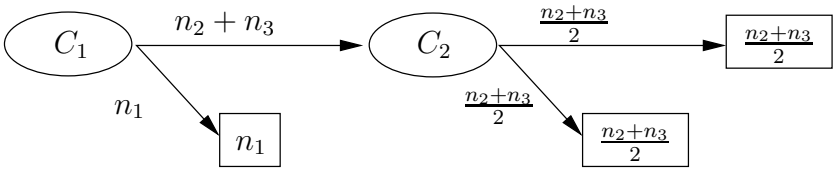


Abbildung 3.3.: Entscheidungsliste $\{C_1, C_2\}$ und die Anzahl der jeweiligen positiv und negativ klassifizierten Beispiele.

vorangegangenen Klassifikatoren in der Berechnung jedes nachfolgenden Klassifikators mit eingehen.

Angenommen man hätte drei Klassen y_1, y_2 und y_3, zwei dazugehörenden Klassifikatoren C_1 und C_2 und entsprechende Fehlklassifikationsraten $\epsilon_1 = 0\%$ und $\epsilon_2 = 50\%$. Betrachten wir im ersten Fall die Entscheidungsliste $\{C_1, C_2\}$ aus Abbildung 3.3. Verwenden wir diese Liste zur Klassifikation einer Menge von Testbeispielen, bestehend aus n_1 Beobachtung der Klasse y_1 bzw. n_2 und n_3 der Klassen y_2 und y_3. Der erste Klassifikator C_1 würde alle n_1 Elemente der Klasse y_1 als positiv klassifizieren und die restlichen $n_2 + n_3$ Elemente als negativ. Der Fehler des ersten Klassifikators auf den Testbeispielen beträgt somit

$$error_{C_1} = 0{,}0.$$

Die als negativ klassifizierten Beispiele würden dann an den zweiten Klassifikator C_2 weitergegeben werden. Dieser hat eine Fehlklassifikationsrate von 50% und würde statistisch gesehen einen Fehler von

$$error_{C_2} = \frac{n_2 + n_3}{2}$$

erzeugen. Es würden $error_{C_2}$ viele Beispiele als positiv und genau so viele als negativ erkannt werden. Dabei wären $\frac{n_3}{2}$ viele fälschlicherweise als positiv und $\frac{n_2}{2}$ als negativ klassifiziert worden. Der Gesamtfehler dieser Entscheidungsliste beträgt somit

$$error_{\{C_1,C_2\}} = \frac{n_2 + n_3}{2}.$$

Betrachten wir nun im zweiten Fall die Entscheidungsliste $\{C_2, C_1\}$ mit den gleichen Testbeispielen, siehe Abbildung 3.4. Der erste Klassifikator C_2 würde sich statistisch gesehen in 50% der Fälle falsch entscheiden und daher einen Fehler von

$$error'_{C_2} = \frac{n_1 + n_2 + n_3}{2}$$

erzeugen. Es würden $error'_{C_2}$ viele Beispiele als positiv klassifiziert werden, wovon $\frac{n_1+n_3}{2}$ eine Fehlklassifikation darstellen. Ebenso würden $error'_{C_2}$ viele als negativ erkannt werden, von denen $\frac{n_2}{2}$ viele falsch klassifiziert wären. Alle negativ Klassifizierten würden

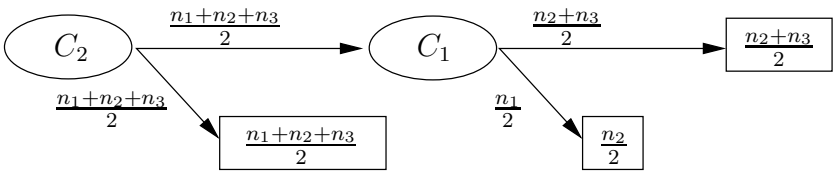


Abbildung 3.4.: Entscheidungsliste $\{C_2, C_1\}$ und die Anzahl der jeweiligen positiv und negativ klassifizierten Beispiele.

an den zweiten binären Klassifikator C_1 weitergegeben werden. Dieser ist ein perfekter Klassifikator und würde daher keine weiteren Fehlklassifikationen

$$error'_{C_1} = 0{,}0$$

erzeugen. Der Gesamtfehler dieser Entscheidungsliste würde somit

$$error_{\{C_2,C_1\}} = \frac{n_1 + n_2 + n_3}{2}$$

betragen, im Vergleich zu $\frac{n_2+n_3}{2}$ der ersten Sequenz.

An diesem Beispiel erkennt man, dass die Qualität einer Entscheidungsliste von der Reihenfolge der binären Klassifikatoren abhängt. Dieses Problem ist jedoch schwierig zu lösen, da es insgesamt $K!$ verschiedene Anordnungen für K Klassifikatoren gibt. Uns war keine Algorithmus bekannt, der die beste Lösung, bezüglich einer Menge von Testdaten, in polynomieller Zeit bestimmt. Daher mussten wir für diese beste Sequenz alle Permutationen testen und den Klassifikationsfehler minimieren. Da in dieser Arbeit jedoch die Anzahl der Klassen $\mathcal{Y} = \{$ Corridor, Doorway, Laboratory, Kitchen, Office, Seminarroom $\}$ auf sechs beschränkt ist, wurde der Aufwand für eine Trainingsmenge in Kauf genommen. Anhand der so ermittelten Entscheidungsliste, wurden dann die restlichen Experimente durchgeführt. Für mehr Informationen sei auf Kapitel 6.1.2 verwiesen.

Ebenso haben wir die Reihenfolge für die Entscheidungsliste, die eine Verteilung über alle Klassen nach Algorithmus 4 bestimmt, nach dem gleich Kriterium bestimmt. Dadurch soll gewährleistet sein, dass ebenfalls durch möglichst wenig Fehlklassifikationen, eine eindeutige Verteilung bezüglich der richtigen Klassen berechnet wird.

Des Weiteren ist es uns gelungen, eine Heuristik zu ermitteln, anhand derer eine gute Sequenz von Klassifikatoren in linearer Zeit ermittelt werden kann. Diese Heuristik liefert zwar nicht die beste Sequenz bezüglich einer Menge von Testdaten, jedoch bekamen wir in unseren Experimenten vergleichbar gute Ergebnisse.

Für diese Heuristik werden alle Testbeispiele mit allen Klassifikatoren $\{C_1, C_2, \ldots, C_K\}$ klassifiziert und entsprechend in vier Klassen: korrekt positiv, falsch positiv, korrekt negativ und falsch negativ unterteilt. Basierend auf diesen Klassen wird eine Gewichtung $W_k, k = 1, \ldots, K$ von jedem Klassifikator berechnet. Zu Beginn werden diese mit Null

initialisiert und dann entsprechend der Klassifikation jedes Beispiels nach folgender Regel modifiziert:

Klassifikation eines Beispiels	**Veränderung der Gewichte W_k**
korrekt positiv	$+1$
falsch positiv	-1
korrekt negativ	$+1$
falsch negativ	-1

Dabei wird die Idee verfolgt, dass ein Klassifikator, der ein Beispiel korrekt klassifiziert, weiter vorne in der Entscheidungsliste verwendet werden sollte. Es ist dann wahrscheinlicher, dass dieses Beispiel bis zu diesem Klassifikator gelangt und dort korrekt klassifiziert wird. Entsprechend wird der Klassifikator mit einer falschen Klassifikation weiter hinten verwendet. Dabei hofft man, dass dieses Beispiel nicht bis zu diesem Klassifikator gelangt und bereits vorher korrekt positiv klassifiziert wird und nicht mehr zu betrachten ist. Durch eine absteigende Sortierung der Gewichte, ergibt sich letztlich die Anordnung der Klassifikatoren. Der genaue Ablauf dieser Heuristik zur Bestimmung der Reihenfolge der Entscheidungsliste ist in Algorithmus 5 zu sehen.

Algorithmus 5 Der Algorithmus, der nach einer Heuristik in linearer Laufzeit eine Reihenfolge für die Entscheidungsliste bestimmt.

Eingabe: eine Menge von Trainingsbeispielen $(x_n, y_n), n = 1, \ldots, N$
binäre Klassifikatoren $\{C_1, C_2, \ldots, C_K\}$
Initialisierung der Gewichte $W_i = 0, i = 1, \ldots, K$
for all Trainingsbeispielen x_n **do**
for all Klassifikatoren C_k **do**
Klassifiziere x_n mit C_k
if Klassifikation ist korrekt positiv **then**
$W_k = W_k + 1$
else if Klassifikation ist falsch positiv **then**
$W_k = W_k - 1$
else if Klassifikation ist korrekt negativ **then**
$W_k = W_k + 1$
else if Klassifikation ist falsch negativ **then**
$W_k = W_k - 1$
end if
end for
end for
Ausgabe: Reihenfolge entsprechend der absteigend sortierten W_k

4. Merkmale zur Erzeugung der schwachen Klassifikatoren

In diesem Kapitel werden die skalaren Merkmale beschrieben, die zur Erzeugung der schwachen Klassifikatoren des AdaBoost Algorithmus aus Kapitel 3.4 benötigt werden. Die Klassifikation einer Beobachtung des Roboters wird basierend auf den Laser- und Kameradaten bestimmt. Daher werden zuerst die Merkmale der Laserdaten in Abschnitt 4.1 vorgestellt und danach die der Kameradaten in Abschnitt 4.2.

Der verwendetet Roboter ist mit einem Lasersensor und einer Kamera ausgerüstet, wobei der Lasersensor Entfernungsdaten rund um den Roboter liefert. Dazu werden L Laserstrahlen in unterschiedliche Richtungen abgesendet und die Zeit bis zur Reflektion gemessen. Anhand dieser Zeit kann auf die Entfernung zu einem Objekt geschlossen werden. Die Kamera des Roboters hat eine maximale Auflösung von 640x480 Pixel. Um eine komplette Umgebungsaufnahme zu erhalten, werden M Bilder in unterschiedlichen Orientierungen des Roboters aufgenommen. Damit ist gewährleistet, dass eine Beobachtung mit der Kamera rotationsinvariant ist. Der Grund dafür ist naheliegend, da die Position des Roboters unabhängig von seiner Orientierung klassifiziert werden soll. Daher muss auch bei der Wahl der Lasermerkmale darauf geachtet werden, dass diese ebenfalls rotationsinvariant sind. Eine Beobachtung des Roboters kann somit durch ein Tripel

$$E = \{(l, v, y) \mid y \in Y = \{\text{Office}, \text{Corridor}, \ldots\}\}$$

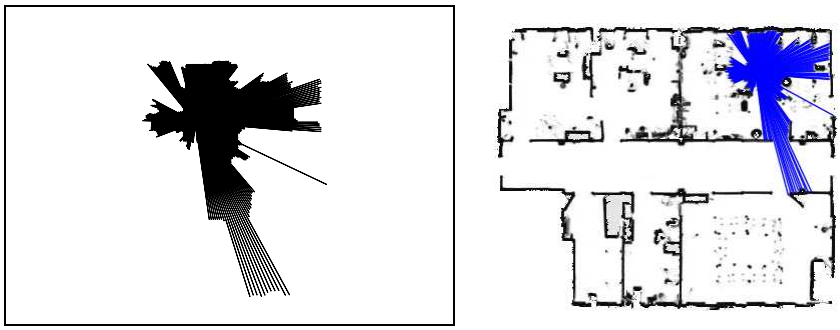
angegeben werden, wobei l den Laserdaten und v den Kameradaten entspricht. Ein Beispiel einer solchen Beobachtung aus einem Großraumbüro ist in Abbildung 4.1 abgebildet.

In den folgenden beiden Kapiteln werden die skalaren Merkmale beschrieben, die von den Laserdaten, wie auch von den Kameradaten extrahiert werden. Diese Merkmale werden, wie bereits in Kapitel 3.4 erklärt, verwendet um zwei verschiedene Typen von schwachen Klassifikatoren zu erzeugen. Alle im Folgenden beschriebenen Merkmale werden somit doppelt genutzt. Es wird aus jedem Merkmal ein schwacher Klassifikator $h(x)$ nach Gleichung (3.11) und einer nach Gleichung (3.12) aus Kapitel 3.4 erzeugt.

(a) v^0 (b) v^1 (c) v^2

(d) v^3 (e) v^4 (f) v^5

(g) v^6 (h) v^7



(i) l (j) Umgebungskarte

Abbildung 4.1.: Beispiel einer Beobachtung E in einem Großraumbüro. Die Bilder 4.1(a) bis 4.1(h) entsprechen einer Umgebungsaufnahme v mit der Kamera und 4.1(i) den Laserdaten l. 4.1(j) zeigt die Postion der Beobachtung in der entsprechenden Umgebungskarte.

4.1. Merkmale der Laserdaten

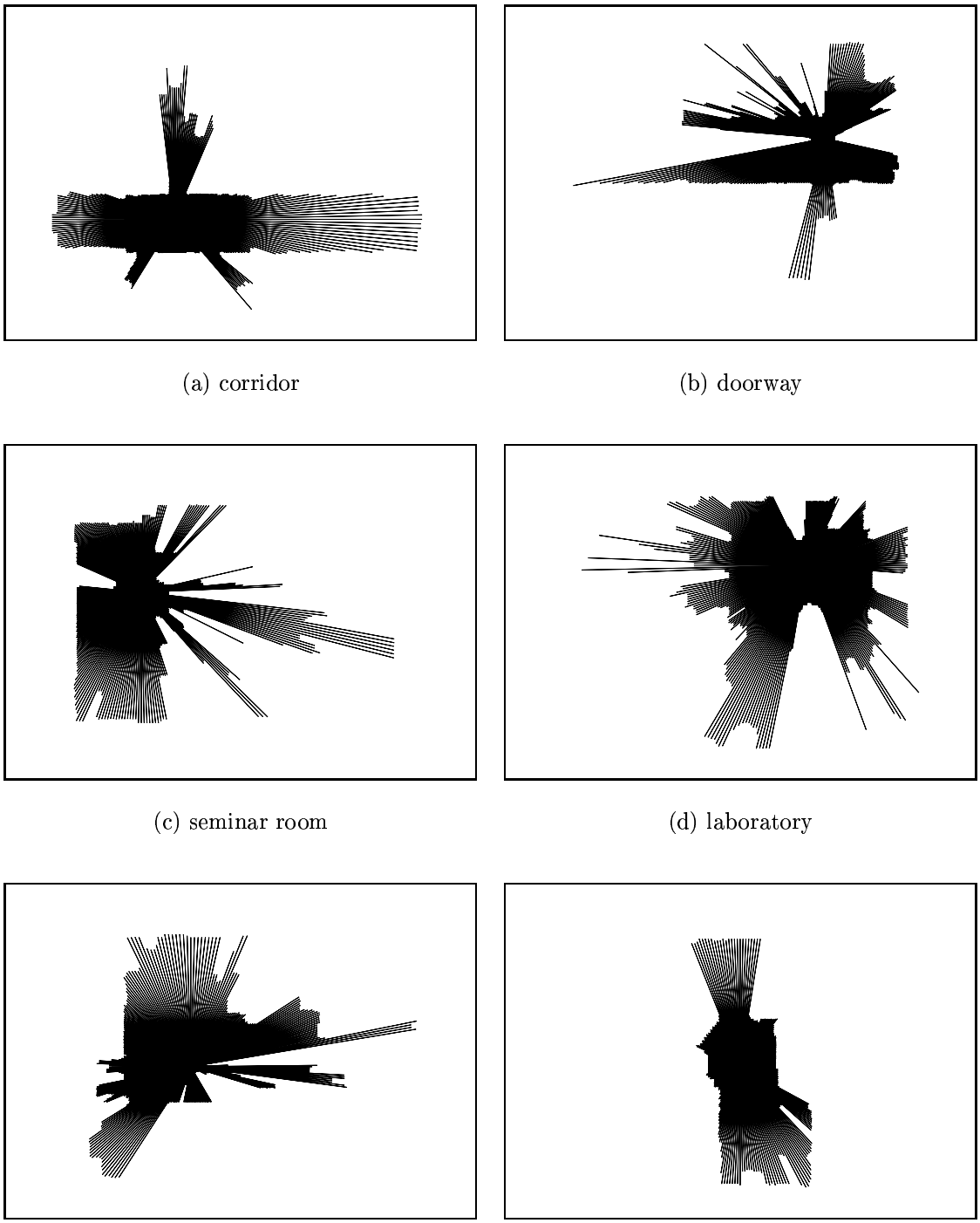
In diesem Kapitel werden die skalaren Merkmale $f^l(l)$, die von den Laserdaten des Roboters extrahiert werden, beschrieben. Der Roboter ist mit einem Lasersensor ausgerüstet, der eine 360 Grad umfassende Messung von Entfernungsdaten liefert. Jede Messung $l = \{b_0, \ldots, b_{L-1}\}$ enthält L Laserstrahlen b_i. Jeder Laserstrahl b_i besteht aus einem Tupel (α_i, d_i), wobei α_i dem Winkel des Strahls relativ zur Orientierung des Roboters und d_i der Distanzmessung des Strahls entspricht. In dieser Arbeit bestehen alle Laserdaten aus $L = 360$ Laserstrahlen mit einem Abstand von je einem Grad untereinander. Beispiele von Laserdaten einer typischen Indoor Umgebung sind in Abbildung 4.2 abgebildet.

In der Arbeit von Martínez Mozos *et al.* [2005] wurde ein Ansatz präsentiert, um anhand von Laserdaten die Position eines Roboters zu klassifizieren. Dort wurden ausschließlich Laserdaten verwendet, aber es wurde auch lediglich zwischen den Klassen Corridor, Doorway und Room unterschieden. Die Klassifikation dieser drei Klassen erzielte sehr gute Ergebnisse und wir können die Merkmale der Laserdaten direkt für unseren Ansatz übernehmen. Im folgenden werden die Merkmale der Laserdaten vorgestellt, können im Detail aber auch in [Martínez Mozos, 2004] nachgelesen werden.

Formal kann ein Lasermerkmal $f^l(l)$ als eine Funktion angesehen werden, die die Laserdaten l einer Beobachtung E als Argument nimmt und einen reellen Wert liefert: $f^l : Z_l \rightarrow \mathbb{R}$, wobei Z_l dem Raum aller möglichen Laserdaten entspricht. Die meisten Merkmale sind einfache geometrische Eigenschaften, die schnell berechnet werden können und oft in der Formanalyse (siehe auch [Gonzalez und Wintz, 1987], [Haralick und Shapiro, 1992], [Loncaric, 1998], [O'Rourke, 1998] und [Russ, 1992]) verwendet werden. Abgesehen davon ist zu beachten, dass alle Merkmale rotationsinvariant sein müssen.

Es werden zwei verschiedene Typen skalarer Merkmale $f^l(l)$ von den Laserdaten gebildet. Der erste Typ B verwendet direkt die unbearbeiteten Laserdaten, wie sie in Abbildung 4.2 dargestellt sind. Im folgenden ist die Liste der Merkmale zu sehen, die zu diesem ersten Typ B gehören:

1. Durchschnittliche Längendistanz zweier aufeinanderfolgender Laserstrahlen. Durch Begrenzung der maximalen Länge werden mehrere Merkmale erzeugt.
2. Standardabweichung der Längendistanz zweier aufeinanderfolgender Laserstrahlen.
3. Durchschnittliche Länge der Strahlen.
4. Standardabweichung der Länge der Strahlen.
5. Anzahl zweier aufeinanderfolgender Strahlen, deren Längendistanz größer als ein gegebener Schwellwert ist. Hierbei werden mehrere Merkmale mit verschiedenen Schwellwerten erzeugt.



(a) corridor

(b) doorway

(c) seminar room

(d) laboratory

(e) office

(f) kitchen

Abbildung 4.2.: Beispiele von unbearbeiteten Laserdaten, bestehend aus 360 Grad umfassenden Distanzmessungen. Aufgenommen an verschiedenen Orten einer typischen Indoor Umgebung.

6. Anzahl der Strahlen, die auf bestimmten Linien liegen (berechnet nach [Sack und Burgard, 2004]).

7. Euklidische Distanz zweier aufeinanderfolgender lokalen Minima.

8. Winkel zwischen den lokalen Minima aus 7.

Der zweite Typ P von Lasermerkmalen wird aus der Polygonapproximation $\mathbf{P}(l)$, die die Laserdaten l umgibt, berechnet. Die Punkte des Polygonzugs $\mathbf{P}(l)$ werden durch die Endpunkte der Laserstrahlen b_i entsprechend der Position des Roboters berechnet durch

$$\mathbf{P}(l) = \{(d_i \cos \alpha_i, d_i \sin \alpha_i) \mid i = 0, \ldots, M-1\}.$$

Beispiele solcher Polygonzüge sind in Abbildung 4.3 dargestellt. Im Folgenden ist die Liste der Merkmale zu sehen, die zu diesem zweiten Typ P gehören:

1. Fläche von $\mathbf{P}(l)$.

2. Umfang von $\mathbf{P}(l)$.

3. Fläche von $\mathbf{P}(l)$ dividiert durch den Umfang von $\mathbf{P}(l)$.

4. Durchschnittlicher Abstand der Distanzmessungen vom Schwerpunkt des Polygonzugs $\mathbf{P}(l)$.

5. Standardabweichung der Abstände der Distanzmessungen vom Schwerpunkt des Polygonzugs $\mathbf{P}(l)$.

6. 200 ähnlichkeitsinvariante Deskriptoren der Fouriertransformation.

7. Betrag der Hauptachse der Ellipse der ersten beiden Fourierkoeffizienten.

8. Betrag der Nebenachse der Ellipse der ersten beiden Fourierkoeffizienten.

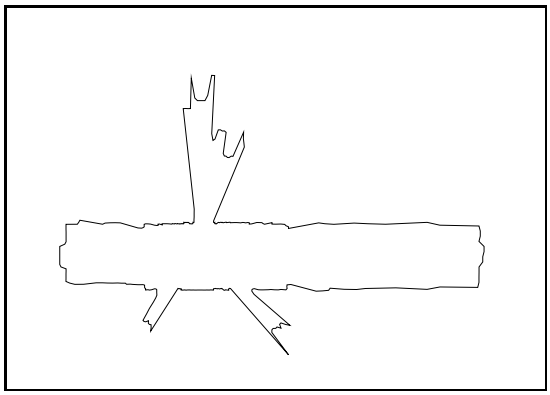
9. Betrag der Hauptachse dividiert durch Betrag der Nebenachse.

10. Sieben Invarianten berechnet aus den zentralen Momenten.

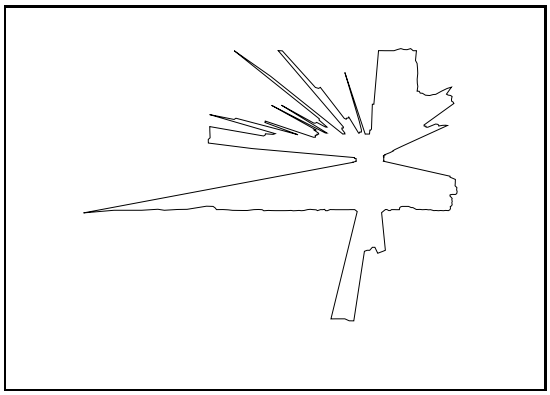
11. Normalisiertes Merkmal der Kompaktheit von $\mathbf{P}(l)$.

12. Normalisiertes Merkmal der Exzentrizität von $\mathbf{P}(l)$.

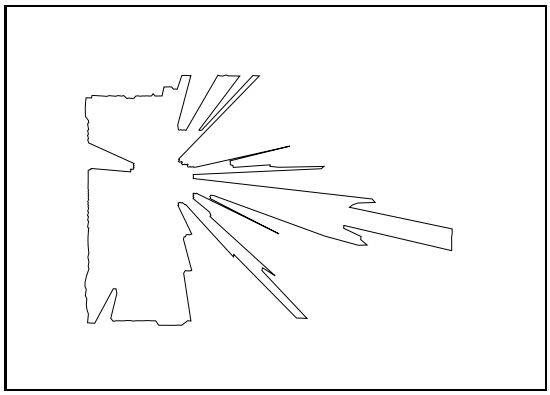
13. Formfaktor von $\mathbf{P}(l)$.



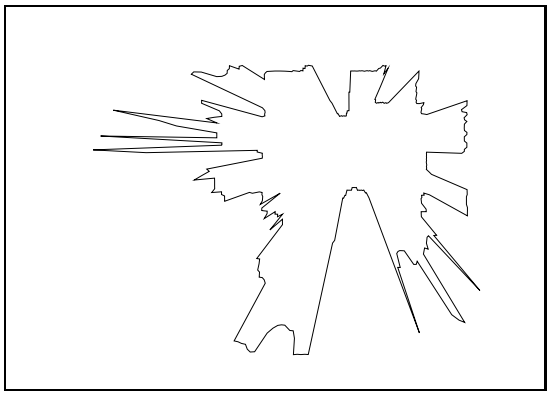
(a) corridor



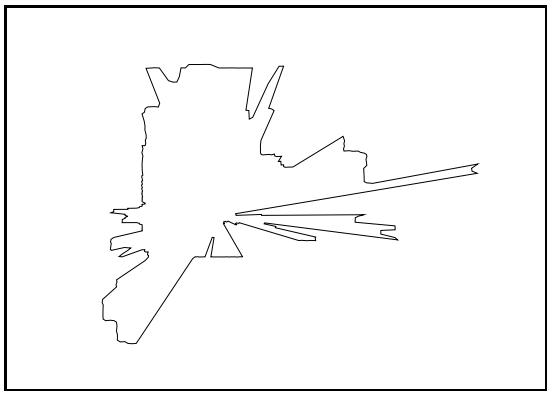
(b) doorway



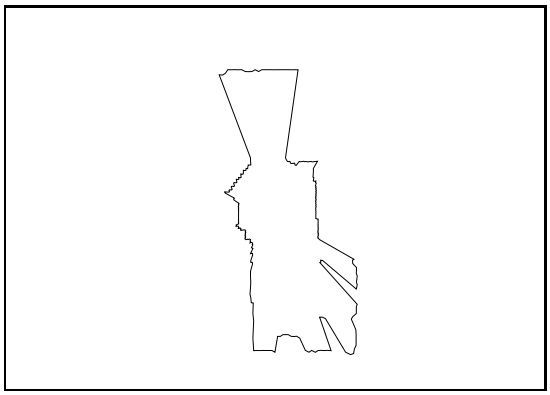
(c) seminar room



(d) laboratory



(e) office



(f) kitchen

Abbildung 4.3.: Polygonapproximationen $\mathbf{P}(l)$ der Laserdaten aus Abbildung 4.2.

Für die exakte Berechnung der Merkmale der Typen B und P sei auf [Martínez Mozos, 2004] verwiesen. Aus all diesen skalaren Merkmalen der Laserdaten werden die schwachen Klassifikatoren aus Kapitel 3.4 erzeugt. Dabei kann jedes Merkmal sowohl in Gleichung (3.11) als auch in Gleichung (3.12) aus Sektion 3.4 verwendet werden, um einen schwachen Klassifikator $h(x)$ zu bilden.

Bei genauerer Betrachtung der Laserdaten aus Abbildung 4.2 fällt auf, dass es selbst für den Betrachter keine einfache Aufgabe darstellt, den Laserdaten eine Klasse zuzuordnen. Was zeichnet beispielsweise die Laserdaten in Abbildung 4.2(c) aus, sie mit der semantischen Information „Seminarraum“ in Verbindung zu setzen. Martínez Mozos *et al.* [2005] haben gezeigt, dass unter der Voraussetzung von drei Klassen $\mathcal{Y} = \{\text{Corridor, Doorway, Room}\}$, die Laserdaten zur Klassifikation ausreichend sind. Die Daten der Klasse Corridor zeichnen sich durch eine geringe Abweichung der Längen zweier Distanzmessungen aus, die sich aufgrund der Form eines Korridors bilden. Die Klasse Doorway kann anhand ihrer beiden Minima mit einer Winkeldifferenz von 180°, die den Türpfosten entsprechen, erkannt werden. Ebenso enthält die Klasse Room viele stark unterschiedlich lange Strahlen, die durch Stühle, Tische und andere Hindernisse entstehen.

In dieser Arbeit wurde die Klasse Room jedoch in die Klassen Laboratory, Seminar Room, Office und Kitchen unterteilt. Dabei sei angemerkt, dass die Klassen Seminar Room und Laboratory sowie Office und Kitchen jeweils etwa eine gleiche Raumgröße und Raumstruktur aufweisen und daher nicht ausschließlich anhand von Laserdaten robust unterschieden werden können. Um dies dennoch zu bewerkstelligen, haben wir zusätzlich Kameradaten verwendet. So kann man zum Beispiel durch die Erkennung einer Kaffeemaschine auf eine Küche und bei Monitoren auf ein Großraumbüro schließen. Genaueres zu den verwendeten Merkmalen der Kameradaten wird im folgenden Kapitel beschrieben.

4.2. Merkmale der Kameradaten

In diesem Kapitel werden die skalaren Merkmale $f^v(v)$, die aus den Kameradaten extrahiert werden, beschrieben. Der Roboter ist mit einer Kamera ausgerüstet, die eine maximale Auflösung von 640x480 Pixel hat. Jede Beobachtung mit der Kamera $v = (v^0, \ldots, v^{M-1})$ besteht aus M Bildern, die jeweils in einer anderen Orientierung des Roboters aufgenommen werden. Damit ist gewährleistet, dass die Klassifikation der Position des Roboters unabhängig von der Orientierung geschieht. In dieser Arbeit bestehen alle Kameradaten aus $M = 8$ Bilder. Ein Beispiel der Kameradaten v ist in Abbildung 4.1 zu sehen.

Die Auswahl der Merkmale aus den Kameradaten basiert auf der Idee, dass bestimmte Objekte typischerweise mit unterschiedlichen Wahrscheinlichkeiten in den verschiedenen Klassen aus $\mathcal{Y}$ zu finden sind. So ist es sehr viel wahrscheinlicher einen Monitor in

einem Büro zu detektieren, als beispielsweise in einer Küche oder einem Seminarraum. Ebenso kann anhand der Anzahl detektierter Monitore zwischen einem Büro und einem Großraumbüro unterschieden werden. In der Regel sind in einem Großraumbüro mehr Monitore zu finden als in einem Büro. Dadurch kann durch mehrere Objekte, die je einem Merkmal der Kameradaten entsprechen, ein robuster Klassifikator erzeugt werden. Ein Merkmal aus den Kameradaten ist daher gegeben durch eine Funktion $f^v : Z_v \rightarrow \mathbb{N}$, wobei Z_v die Menge aller möglichen Kameradaten v darstellt. Als Ergebnis wird die Summe

$$f^v(v) = \sum_{i=0}^{M-1} f^v(v^i)$$

aller detektierter Objekte eines Typs in allen Bilder $v^i, i = 0, \ldots, M-1$ aus v geliefert.

Es gibt verschiedene Arbeiten ([Viola und Jones, 2001], [Treptow *et al.*, 2003], [Schneiderman und Kanade, 2000]) zur Objekterkennung aus Bildern. Ein verbreiteter Ansatz basiert auf dem in Kapitel 3.2 vorgestellten AdaBoost Algorithmus. Meist werden die schwachen Klassifikatoren aus leicht berechenbaren Merkmalen gebildet. So zum Beispiel einer Summe von bestimmten Pixelregionen, die in der Arbeit von Papageorgiou *et al.* [1998] erstmals vorgestellt wurden. Dabei werden die Grauwerte bestimmter Pixel von den Werten anderer Pixel abgezogen, um einen skalaren Faktor zu erhalten. Diese Idee wurde hier aufgegriffen und für verschiedene Objekte einen Klassifikator mit einem bereits implementierten Algorithmus von Lienhart *et al.* [2003] trainiert. Dieser Algorithmus ist in der Open Computer Vision Library [OpenCV, 2001] enthalten und wird in Kapitel 4.3 kurz vorgestellt. Ein Beispiel eines detektierten Objekts ist in Abbildung 4.4 zu sehen. Im folgenden sind die Objekte, von denen ein Klassifikator trainiert wurde, aufgelistet:

1. Monitor ausgeschaltet
2. Monitor eingeschaltet
3. Kaffeemaschine
4. Büroschrank
5. Seifenspender
6. Gesicht frontal
7. Gesicht im Profil
8. Oberkörper einer Person
9. gesammter Körper einer Person

Dabei wurden die bereits trainierten Klassifikatoren 6. bis 9. aus der Open Computer Vision Library [OpenCV, 2001] übernommen. Die Idee dabei ist, dass in einem Großraumbüro in der Regel mehr Menschen zu finden sind und daher auch mehr Gesichter erkannt werden. Ebenso ist in einem Korridor eher der gesammter Körper einer Person zu sehen als in einer anderen Klasse.

4.3. Verfahren zur Erstellung der Merkmale aus den Kameradaten

In diesem Kapitel wird das Verfahren zum Trainieren der Klassifikatoren, aus denen die Merkmale der Kameradaten berechnet werden, vorgestellt. Dabei sei im folgenden zwischen dem Klassifikator H, der die Position des Roboters schätzt, und den Klassifikatoren H' zur Erkennung der Objekte aus den Kameradaten unterschieden.

Ein Merkmal $f^v(v)$ liefert die Anzahl erkannter Objekte eines Typs aus allen Bildern $v^i, i = 0, \ldots, M-1$. Anhand der Anzahl der jeweiligen Objekte, kann der Klassifikator H auf eine bestimmte Klasse aus $\mathcal{Y}$ schließen. Für jedes Objekt wird daher ein Klassifikator H' benötigt. Um einen robusten Klassifikator H zur Klassifikation der Umgebung des Roboters zu erhalten, sind gute Klassifikatoren H' zur Objekterkennung nötig. Zum Trainieren dieser Klassifikatoren H' haben wir den vorprogrammierten Algorithmus von Lienhart *et al.* [2003] aus der Open Computer Vision Library [OpenCV, 2001] verwendet.

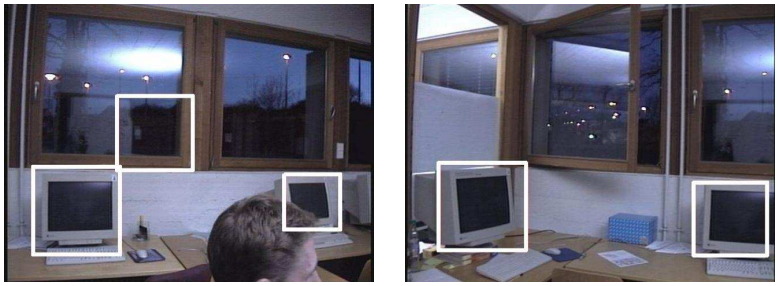
OpenCV ist eine Open-Source Bibliothek, die viele Funktionen für den Bereich Computersehen bereitstellt. Das Softwarepaket wurde von Intel für Linux und Windows OS entwickelt und im Jahr 2001 veröffentlicht. Die meisten Funktionen wurden für Prozessoren der Intel-Architektur optimiert und direkt auf Assembler-Ebene programmiert.

In dieser OpenCV Bibliothek ist ein Trainings- wie auch ein Erkennungsalgorithmus für Objekte in Kamerabildern enthalten. Er wurde von Lienhart *et al.* [2003] entwickelt und basiert auf der Idee von Viola und Jones [2001]. Zum Trainieren der Klassifikatoren H' wird ein Boosting Algorithmus verwendet, wobei zwischen verschiedenen Varianten gewählt werden kann. In [Lienhart *et al.*, 2003] wird gezeigt, dass mit dem Gentle AdaBoost die besten Ergebnisse zur Gesichtserkennung erzielt werden, wobei auch ein Diskreter und ein Real AdaBoost Algorithmus verfügbar sind. Diese sind alles spezielle Versionen des AdaBoost Algorithmus, verfahren aber nach dem gleichen Prinzip, wie es in Kapitel 3.2 vorgestellt wird. Informationen zu den einzelnen Algorithmen sind in der Arbeit von Friedman *et al.* [1998] zu finden. Die schwachen Klassifikatoren all dieser Algorithmen werden jeweils nach Gleichung (3.11) aus Kapitel 3.4 berechnet.

Als Merkmale zur Objekterkennung aus Bildern werden Haar-Transformations ähnliche Merkmale $\mathcal{H}$ verwendet. Diese bestehen aus der Summe von Pixelwerten bestimmter Bildbereiche und die Grundtypen sind in Abbildung 4.5(a) zu sehen. Dabei erhalten die

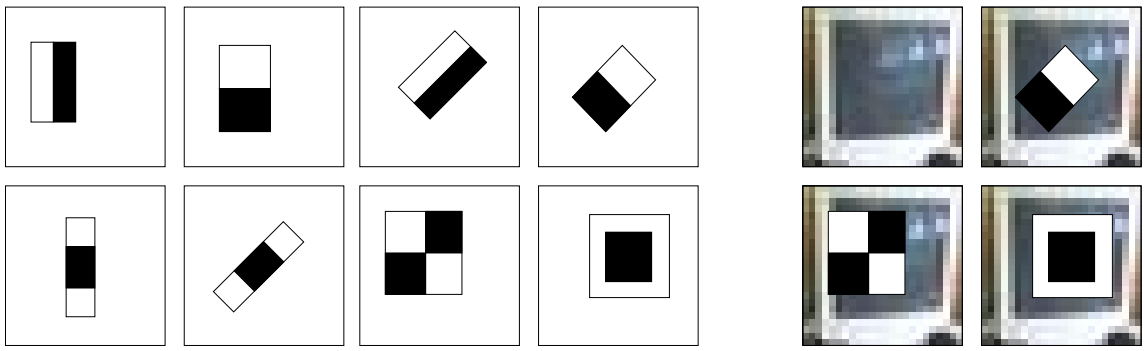
(a) v^0 (b) v^1 (c) v^2

(d) v^3 (e) v^4 (f) v^5



(g) v^6 (h) v^7

Abbildung 4.4.: Detektion des Objektes „Monitor ausgeschaltet“ in der Beobachtung E aus Abbildung 4.1. Das Ergebnis dieses Merkmals für diese Beobachtung wäre damit $f^v(v) = 10$.



(a) Merkmale $\mathcal{H}$ (b) Beispiel x_n

Abbildung 4.5.: 4.5(a): Die Grundtypen der Merkmale $\mathcal{H}$ zur Erzeugung der schwachen Klassifikatoren des Algorithmus von Lienhart *et al.* [2003]. Die Summe der Pixel aus den schwarzen Bereichen wird von der Summe der weißen Bereiche abgezogen. 4.5(b): Positives Beispiel eines Monitors von 24x24 Pixel und darauf angewandte Merkmale.

Bildpunkte in den schwarzen Bereichen eine negative Gewichtung. Der Wert eines solchen Merkmals kann mit Hilfe eines Integralbildes II in kürzester Zeit berechnet werden. Der Wert eines Integralbildes an Position (x, y) besteht aus der Summe der Grauwerte der Pixel aus dem Bereich, der sich von der linken oberen Ecke $(0, 0)$ bis zur Position (x, y) erstreckt, siehe dazu Abbildung 4.6(a). Der Wert $II(x, y)$ berechnet sich somit durch

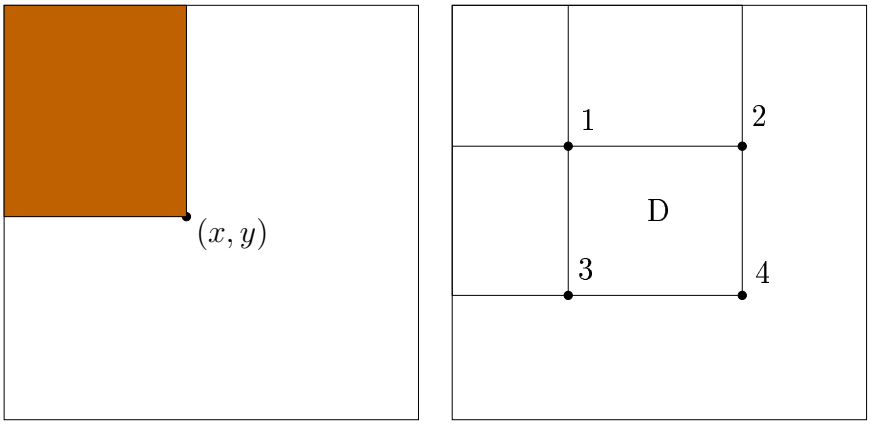
$$II(x, y) = \sum_{x' \leq x, y' \leq y} I(x', y'),$$

wobei $I(x', y')$ dem Grauwert des Pixels (x', y') aus dem Bild I entspricht. Das Erstellen eines solchen Integralbildes ist in einem Durchlauf möglich. Jedes Merkmal aus $\mathcal{H}$ kann damit aus wenigen Werten des Integralbildes berechnet werden, siehe dazu Abbildung 4.6(b). Ist beispielweise die Summe der Pixelwerte aus dem Bereich D gesucht, der sich durch die Punkte $1, 2, 3$, und 4 ergibt, so kann dieser durch die vier Werte des Integralbildes mit

$$D = II(4) + II(1) - II(2) - II(3)$$

berechnet werden.

Durch verschiedene Größen und Positionen der Merkmale $\mathcal{H}$, kann eine Vielzahl unterschiedlicher Merkmale erzeugt werden. Zum Beispiel können insgesamt 117.941 verschiedene Merkmale in einem Bildbereich von lediglich 24x24 Pixel konstruiert werden. Dies entspricht etwa dem 200-fachen der Anzahl der Pixel in diesem Bereich. Um daraus dennoch einen robusten Klassifikator zu konstruieren, werden mit Hilfe des AdaBoost



(a) (b)

Abbildung 4.6.: 4.6(a): Der Wert des Integralbildes II an Position (x, y) besteht aus der Summe aller Pixel aus dem Bereich von (x, y) bis zur linken oberen Ecke $(0, 0)$. 4.6(b) Ein Merkmal kann mit Hilfe eines Integralbildes schnell aus wenigen Werten berechnet werden. Sei D die Summe des gesuchten Bildbereichs, so ergibt sich $D = II(4) + II(1) - II(2) - II(3)$.

Algorithmus die besten schwachen Klassifikatoren bestimmt und zu einem starken Klassifikator H' kombiniert.

Zum Trainieren dieses starken Klassifikatoren wird eine Menge von Trainingsbeispielen $(x_n, y_n), n = 1, \ldots, N$ benötigt. Dabei entspricht jedes x_n einem Beispiel aus dem Raum $\mathcal{X}$ aller möglichen Bilder und y_n einer Klasse aus $\mathcal{Y}$. Da wir nur binäre Klassifikatoren trainieren, setzen wir $\mathcal{Y} = (-1, +1)$, entsprechend einem negativen oder positiven Beispiel. In der Regel bestehen diese aus einem kleinen Bild weniger Pixel, zum Beispiel der Monitor der Größe 24x24 Pixel aus Abbildung 4.5(b). In diesem Bild ist dann entsprechend y_n entweder ein Beispiel eines zu klassifizierenden Objekts enthalten oder keines. Der AdaBoost versucht dann die beste Teilmenge der schwachen Klassifikatoren auszuwählen, die alle Trainingsbeispiele bestmöglichst klassifiziert. Die Grundidee des AdaBoost Verfahren ist in Kaptitel 3.2 bereits vorgestellt worden.

Der trainierte Klassifikator H' ist danach in der Lage, Objekte in einem Bild der Größe der trainierten Maske, zum Beispiel 24x24 Pixel, zu erkennen. Da die Größe von Objekten in Testbildern jedoch normalerweise nicht dieser Auflösung entsprechen, wird der Klassifikator mehrmals für verschiedene Skalierungen angewandt. Dazu wird ein entsprechender Bildausschnitt auf die Größe des Klassifikators skaliert. Da in dieser Arbeit Kamerabilder mit einer Auflösung von 320x240 Pixel verwendet werden, ist der Klassifikator ebenso als Maske über das gesamte Bild zu führen. Eine Routine zur Klassifikation

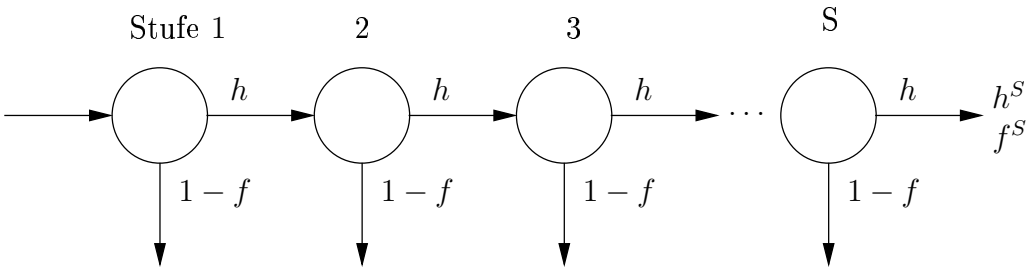


Abbildung 4.7.: Kaskade von S Stufen zur Beschleunigung der Klassifikation eines Bildausschnitts. In jeder Stufe wird ein Klassifikator mit einer Trefferrate h und einer falschem Positivrate f trainiert.

der Objekte ist ebenfalls in der OpenCV Bibliothek enthalten und liefert als Ergebnis die Anzahl erkannter Objekte und ihre jeweilige Lage in dem Testbild. Durch die notwendige Skalierung und Verschiebung der Klassifikatormaske kommt es jedoch zu einer Vielzahl von Bildausschnitten, die nach einem Objekt durchsucht werden müssen.

In dieser Arbeit besteht eine Kamerabeobachtung aus M Bildern, die ebenfalls alle zu durchsuchen sind. Das kann so zeitaufwendig werden, dass keine Echtzeitberechnung mehr möglich ist. Um die Geschwindigkeit des Klassifikators dennoch etwas zu erhöhen, wird dieser beim Training aus einer sogenannten *Kaskade* zusammengesetzt. Diese Kaskade entspricht dabei einer Entscheidungsliste, bestehend aus mehreren Stufen, die nacheinander durchlaufen werden. Wird in einer Kaskade ein Bildausschnitt als negativ klassifiziert, wird dieser Ausschnitt als negativ zurückgewiesen. Im Falle einer positiven Klassifikation, wird er an die nächste Stufe weitergegeben. Liefern alle Stufen eine positive Klassifikation, wird für diesen Ausschnitt ein Objekt als detektiert zurückgeliefert. Eine solche Kaskade ist in Abbildung 4.7 abgebildet. Jede einzelne Stufe besteht dabei aus einem trainierten Klassifikator, der eine vorher festgelegte Trefferrate h und eine falsche Positivrate f hat. Der Vorteil gegenüber eines einzigen Klassifikators besteht darin, dass die falsche Positivrate nicht besonders hoch sein muß. Bei den in dieser Arbeit verwendeten Klassifikatoren wurden zum Beispiel eine falsche Positivrate von $f = 0{,}5$, eine Trefferrate von $h = 0{,}999$ und $S = 20$ Stufen gewählt. Durch die Verwendung einer solchen Kaskade enthält man aber dennoch einen robusten Klassifikator. In jeder Stufe werden nur $f = 0{,}5$ der negativen Ausschnitte durchgelassen, was zu einem erwarteten Gesamtfehler von $0{,}5^{20} \approx 9{,}6\text{e} - 07$ führt. Dennoch wird sich eine erwartete Trefferrate von $0{,}995^{20} \approx 0{,}98$ ergeben. Die beschleunigte Klassifikation ergibt sich dadurch, dass bereits manche Ausschnitte in den ersten Stufen als negativ erkannt werden und nicht mehr weiter betrachtet werden müssen.

5. Wahrscheinlichkeitsbasierte Klassifikation von Beobachtungen

Der bisher beschriebene Ansatz ist ausschließlich in der Lage, eine einzige Beobachtung zu klassifizieren und ihr eine Klasse aus $\mathcal{Y}$ zuzuweisen. Es ist bisher nicht möglich, vorangegangene Klassifikationen in der Berechnung der aktuellen Beobachtung zu berücksichtigen. Dies kann jedoch in vielen Umgebungen einen Vorteil bringen und zu einer robusteren Klassifikation führen. Zum Beispiel sind in Indoor Umgebungen die Klassen von nahe beieinanderliegenden Beobachtungen meist identisch. Ebenso sind Übergänge zwischen bestimmten Klassen sehr unwahrscheinlich. So wird beispielsweise bei einer derzeitigen Position in der Klasse Kitchen, mit einer sehr geringen Wahrscheinlichkeit die nachfolgende Beobachtung in der Klasse Office sein, vorausgesetzt der Roboter bewegt sich keine allzugroße Strecke. Um von der Klasse Kitchen in die Klasse Office zu gelangen, müßte der Roboter zum Beispiel erst die Klasse Doorway durchfahren.

Um eine solche Abhängigkeit zwischen den verschiedenen Klassen zu simulieren, wird ein sogenanntes *Hidden Markov Model* (HMM) [Russell und Norvig, 1995] verwendet. Ein HMM ist ein zeitliches Wahrscheinlichkeitsmodell, in dem die Zustände des Prozesses durch skalare Variablen ξ_t beschrieben werden. Diese Variablen entsprechen dabei den möglichen Weltzuständen, in unserem Fall den Klassen $\xi_t \in \mathcal{Y}$. Dabei wird eine Wahrscheinlichkeit $Bel(\xi_t)$ für den *Glauben*, der Roboter befinde sich zum Zeitpunkt t in der Klasse ξ_t, berechnet durch

$$Bel(\xi_t) = \alpha \cdot P(z_t \mid \xi_t) \sum_{\xi_{t-1}} P(\xi_t \mid \xi_{t-1}, u_{t-1}) Bel(\xi_{t-1}). \tag{5.1}$$

In dieser Gleichung entspricht α einer Normalisierungskonstante, die dafür sorgt, dass die Summe über alle Klassen immer eins ergibt. Um ein solches HMM verwenden zu können, sind drei Komponenten nötig. Zum einen wird das Beobachtungsmodell $P(z_t \mid \xi_t)$ benötigt, das die Wahrscheinlichkeit der Klassifikation von $z_t \in \mathcal{Y}$, gegeben der Roboter befindet sich in Klasse ξ_t, wiedergibt. Des Weiteren muss das Bewegungsmodell $P(\xi_t \mid \xi_{t-1}, u_{t-1})$ des Roboters bekannt sein. Dabei entspricht $P(\xi_t \mid \xi_{t-1}, u_{t-1})$ der Wahrscheinlichkeit, dass der Roboter durch Ausführen der Aktion u_{t-1} von der Klasse ξ_{t-1} in die Klasse ξ_t gelangt. Und zuletzt muss der Initialisierungwert der Wahrscheinlichkeiten

$Bel(\xi_0)$ gewählt werden. Durch solch einen Ansatz kann der Glaube $Bel(\xi_t)$ über die Position des Roboters für jede Klasse ξ_t berechnet werden. Es wird dementsprechend der aktuellen Position die Klasse mit dem größten Glauben zugewiesen.

In dieser Arbeit haben wir zur Initialisierung der Werte eine Gleichverteilung

$$Bel(\xi_0) = \frac{1}{|\mathcal{Y}|}$$

gewählt, wobei $|\mathcal{Y}|$ der Anzahl verschiedener Klassen entspricht.

5.1. Das Bewegungsmodell $P(\xi_t \mid \xi_{t-1}, u_{t-1})$ des Roboters

Um das Bewegungsmodell $P(\xi_t \mid \xi_{t-1}, u_{t-1})$ des Roboters zu erfassen, wurden lediglich die beiden Aktionen $u_{t-1} \in \{\text{BEWEGEN}, \text{WARTEN}\}$ berücksichtigt. Die Wahrscheinlichkeit von einer Klasse zu einer anderen zu gelangen, wurde dabei durch die simulierte Bewegung des Roboters in 1.000 Experimenten ermittelt. Dazu wurde der Roboter in der entsprechenden Klasse ξ_{t-1} an einen zufällig bestimmten Ort in einer zufälligen Orientierung positioniert, um danach eine Bewegung von $20 - 50$ cm nach vorne auszuführen. Dies entspricht etwa der Distanz, die der Roboter in den späteren Experimenten zwischen den einzelnen Beobachtungen zurücklegte. Die Klassen der jeweiligen Endpositionen wurden dann in eine Wahrscheinlichkeitsverteilung mit addiertem Dirichlet-Term übertragen. Es wird jeweils ein Dirichlet-Term

$$P(c) = \delta \frac{1}{|\mathcal{Y}|} + (1 - \delta) P(c)$$

addiert, damit sich keine Wahrscheinlichkeit von Null ergibt. Dabei entspricht δ einem Faktor aus dem Intervall $\delta \in [0, 1]$. Das Ergebnis der einzelnen Klassen ist in einer Wahrscheinlichkeitsmatrix in Abbildung 5.1 für die Aktion $u_{t-1} = \text{BEWEGEN}$ zu sehen. Dabei entsprechen dunklere Werte einer geringeren Übergangswahrscheinlichkeit.

Des Weiteren sei darauf hingewiesen, dass die Wahrscheinlichkeit, in der gleichen Klasse zu bleiben, für alle Klassen am größten ist. Ebenso ist die Wahrscheinlichkeit, in der Klasse Doorway zu bleiben, etwas geringer als bei den anderen Klassen. Das resultiert daraus, dass die Klasse Doorway normalerweise einem sehr kleinen Bereich entspricht, in dem sich der Roboter meist nur eine kurze Zeit aufhält. Des Weiteren ist die Wahrscheinlichkeit von einer der Klassen Room in die Klasse Doorway zu gelangen, wesentlich größer als in eine andere Klasse. Das folgt aus der Tatsache, dass man nur mit Durchfahren der Klasse Doorway einen anderen Klasse erreichen kann. Ebenso ist die Wahrscheinlichkeit von der Klasse Laboratory und Seminar Room die Klasse Doorway zu erreichen etwas

Abbildung 5.1.: Das Bewegungsmodell $P(\xi_t \mid \xi_{t-1}, u_{t-1})$ des Roboters, repräsentiert durch die Wahrscheinlichkeiten aller Übergänge der einzelnen Klassen. Um die Sichtbarkeit zu erhöhen, wurde eine logarithmische Skalierung gewählt, wobei dunklere Werte einer geringeren Wahrscheinlichkeit entsprechen.

geringer. Das beruht darauf, dass die Raumgröße dieser Klassen größer ist, und damit die Wahrscheinlichkeit in die Klasse Doorway zu gelangen etwas geringer.

Im Falle der Aktion $u_{t-1} =$ WARTEN wird keine Aktualisierung der Werte $Bel(\xi_t)$ der einzelnen Klassen vorgenommen. Solange sich der Roboter nicht bewegt, wird sich die Beobachtung E nicht verändern und es entstehen somit keine neuen Informationen zur Klassifikation der aktuellen Position.

5.2. Das Beobachtungsmodell $P(z_t \mid \xi_t)$ des Roboters, basierend auf Klassifikationen

Es gibt zwei Möglichkeiten das Beobachtungsmodell des Roboters darzustellen. Zum einen könnte zur Klassifikation eines Beispiels der Algorithmus 3 verwendet werden. Dafür wird der Wert des Beobachtungsmodells $P(z_t \mid \xi_t)$ durch eine Statistik über die Ausgabe des sequentiellen Multi-Klassen Klassifikators H, gegeben der Roboter befindet sich in der entsprechenden Klasse ξ_t, beschrieben. Dazu werden von jeder Klasse 50 Beobachtungen klassifiziert und die Ergebnisse in einer Wahrscheinlichkeitsverteilung gespeichert. Zusätzlich wird wiederum ein Dirichlet-Term addiert, damit sich keine Wahrscheinlichkeit von Null ergibt. Die Ergebnisse der einzelnen Klassen sind in einer Wahrscheinlichkeitsmatrix in Abbildung 5.2 zu sehen.

Abbildung 5.2.: Das Beobachtungsmodell $P(z_t \mid \xi_t)$ des Roboters. Für jeweils 50 Beobachtungen einer bestimmten Klasse $\xi_t \in \mathcal{Y}$ wurde die Klassifikation des sequentiellen Multi-Klassen Klassifikators $H(x_n)$, $y_n = \xi_t$ durch eine Wahrscheinlichkeitsverteilung dargestellt. Dabei entsprechen dunklere Werte einer geringeren Wahrscheinlichkeit.

Wie in dieser Matrix zu sehen ist, liefert der Klassifikator für die verschiedenen Klassen unterschiedlich gute Klassifikationraten. Dabei werden für die Klassen Corridor, Doorway, Seminar Room und Laboratory die robustesten Entscheidungen geliefert. Diese Eigenschaft beruht zum einen darauf, dass Corridor und Doorway bereits anhand der Laserdaten sehr gut klassifiziert werden können. Zum anderen haben die Klassen Seminar Room und Laboratory eine ähnliche Raumgröße und Struktur und unterscheiden sich somit anhand der Laserdaten von den Klassen Kitchen und Office. Des Weiteren werden von fast jeder Position in einem Laboratory viele Monitore erkannt, wodurch robust zwischen diesen beiden Klassen unterschieden werden kann. Ebenso fällt auf, dass die Klassifikation einer Beobachtung aus der Klasse Kitchen in der Hälfte aller Fälle als Ergebnis die Klasse Office liefert. Das basiert auf der Raumaufteilung einer Küche, denn es ist nicht von allen Positionen eine Kaffeemaschine zu sehen. Aus diesem Grund werden die Hälfte der Beobachtungen einer Küche der Klasse Office zugeordnet.

5.3. Das Beobachtungsmodell $P(z_t \mid \xi_t)$ des Roboters, basierend auf Wahrscheinlichkeiten

Zum anderen gibt es die Möglichkeit, an dieser Stelle den Multi-Klassen Klassifikator H unter Verwendung von Algorithmus 4 zu nutzen, um eine Wahrscheinlichkeitsverteilung

über alle Klassen zu erhalten. Dazu würde man von den entsprechenden Klassen ξ_t eine Verteilung bestimmen, indem man wiederum 50 Beobachtungen klassifiziert und die erhaltenen Verteilungen zu einer Durchschnittsverteilung vereinigt. Ebenfalls würde der Multi-Klassen Klassifikator H von der aktuellen Beobachtung eine Verteilung über alle Klassen berechnen. Um nun einen Wert für das Beobachtungsmodell $P(z_t \mid \xi_t)$, anhand dieser beiden Verteilungen zu erhalten, müsste man diese lediglich miteinander vergleichen. Dazu kann man das Maß der *KL Divergenz*

$$D(z_t \parallel \xi_t) = \sum_{i=0}^{K} z_t(i)\ log\frac{z_t(i)}{\xi_t(i)}, \tag{5.2}$$

verwenden [Cover und Thomas, 1991], wobei $z_t(i)$ beziehungsweise $\xi_t(i)$ die Wahrscheinlichkeit der positiven Klassenzugehörigkeit der Klassen i aus der entsprechenden Verteilung wiedergibt. Die KL Divergenz ist dabei ein nicht negatives Maß und nur im Falle der Gleichheit Null. Sie ist jedoch keine echt Distanz zwischen zwei Verteilungen, da sie nicht symmetrisch ist und nicht die Dreiecksungleichung erfüllt. Ebenso wurde entsprechend den Werten $z_t(k) = 0$ das Maß auf $D(z_t \parallel \xi_t) = 0$ und für $\xi_t(k) = 0$ auf $D(z_t \parallel \xi_t) = \infty$ gesetzt. Um daraus eine Verteilung entsprechend einer Wahrscheinlichkeit zwischen Null und Eins zu erhalten, wurde

$$P(z_t \mid \xi_t) = e^{-D(z_t \parallel \xi_t)^2}$$

gesetzt.

6. Experimente

In diesem Kapitel werden die verschiedenen Experimente präsentiert, die in Simulation wie auch mit echten Robotern in einer typischen Büroumgebung durchgeführt wurden. Die beschriebenen Methoden wurden implementiert und mit einem B21r Roboter getestet. Dieser Roboter ist mit einem SICK Laserscanner und einer Kamera ausgerüstet. Anhand dieser Experimente soll nun demonstriert werden, dass laser- und kamerabasierte Merkmale für eine robuste Klassifikation von Beobachtungen verwendet werden können. Dazu wird in Kapitel 6.1 das Trainieren der Klassifikatoren mit dem AdaBoost Algorithmus erklärt, sowie die Bestimmung der Entscheidungsliste vorgestellt. Danach werden in je zwei Experimenten die Klassifikationsraten der Entscheidungsliste mit und ohne Verwendung des HMM präsentiert. Ebenso werden die Klassifikationsraten unter Verwendung von verschiedenen schwachen Klassifikatoren vorgestellt. Des Weiteren wird dieser Ansatz mit vorherigen verglichen, um die Verbesserung durch die Erweiterung von Kameradaten zu veranschaulichen. Zuletzt wird ein Anwendungsbeispiel präsentiert, in dem die Klassifikation von Beobachtungen für das Problem der Lokalisierung von mobilen Robotern erfolgreich Verwendung finden kann.

6.1. Details zum Trainieren der Klassifikatoren und Bestimmen der Entscheidungsliste

In diesem Abschnitt wird im Detail erläutert, wie die Klassifikatoren mit Hilfe des AdaBoost Algorithmus trainiert wurden und wie die notwendigen Trainingsbeispiele erzeugt werden können. Ebenso wird noch einmal kurz erklärt, wie die optimale Entscheidungsliste bestimmt wurde.

6.1.1. Trainieren der Klassifikatoren mit dem AdaBoost Algorithmus

Um die einzelnen binären Klassifikatoren zu trainieren, ist eine größere Menge von Trainingsdaten nötig. Es wurden verschiedene Größenordnungen getestet und die besten Resultate mit einer Anzahl von 38.500 positiven und negativen Trainingsbeispielen erzielt. Um eine solch große Menge an Trainingsdaten zu erhalten, ist es ratsam, die Laser-

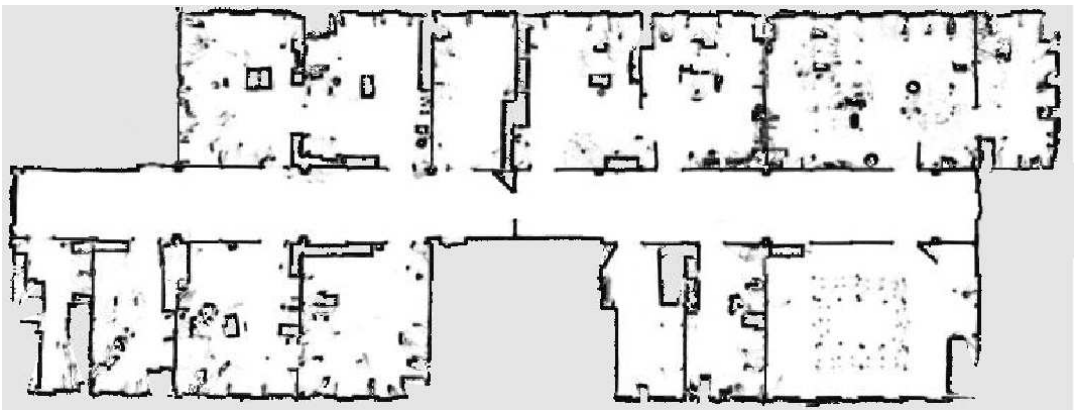


Abbildung 6.1.: Occupancy Grid Map einer typischen Büroumgebung

und Kameradaten in einer Simulation zu erzeugen.

Um die Laserdaten zu erzeugen, wurde dafür eine Belegungsgitter-Karte (*Occupancy Grid Map*) verwendet. Dazu wurde mit Hilfe des Carnegie Mellon Robot Navigation Toolkit (CARMEN) [Montemerlo *et al.*, 2003] der Roboter virtuell an verschiedenen Positionen in einer Umgebungskarte positioniert und die Laserdaten des Scanners simuliert. Dies ist eine typische Vorgehensweise um eine größere Anzahl unterschiedlicher Laserdaten aus beliebigen Umgebungen zu erhalten. Eine solche Occupancy Grid Map von unserer Büroumgebung ist in Abbildung 6.1 zu sehen.

Um die Merkmale der dazugehörenden Kameradaten zu simulieren, wurden insgesamt 350 Panoramaaufnahmen mit der Kamera des Roboters in der Umgebung aus Abbildung 6.1 von allen Klassen aufgenommen. Jede dieser Panoramaaufnahmen besteht aus 8 einzelnen Bildern, wobei jedes in einer anderen Orientierung des Roboters aufgenommen wurde. Dadurch ist gewährleistet, dass die Kameradaten einer kompletten Rundumsicht des Roboters entsprechen und somit rotationsinvariant sind. Auf diese Weise wurde eine Datenbank von Kameradaten erstellt, aus der zu jedem simulierten Laserscan eine Panoramaaufnahme der entsprechenden Klasse zufällig gezogen werden kann. Anhand dieser Aufnahmen werden dann die Merkmale der Kameradaten $f^v(v)$ für jedes Objekt bestimmt. Um die Trainingsprozedur etwas zu beschleunigen, werden alle Merkmale vorher berechnet und in eine Verteilung gespeichert. Damit ist die zeitaufwendige Bestimmung der Merkmale nur einmal auszuführen. Das führt schließlich dazu, dass für jede Klasse und jedes Merkmal zwei Verteilungen erzeugt wurden. Eine entsprechend für positive und eine für negative Trainingsdaten. Die beiden Verteilungen für die Klasse Kitchen und das Merkmal „Kaffeemaschine“ sind in Abbildung 6.2 zu sehen. Letztlich werden so zu jedem simulierten Laserscan die Merkmale der Kameradaten aus den entsprechenden Verteilung zufällig gezogen, um schließlich ein Trainingsbeispiel x_n zu erzeugen.

Da die verschiedenen Klassen unterschiedliche Raumgrößen haben, kann zu den jewei-

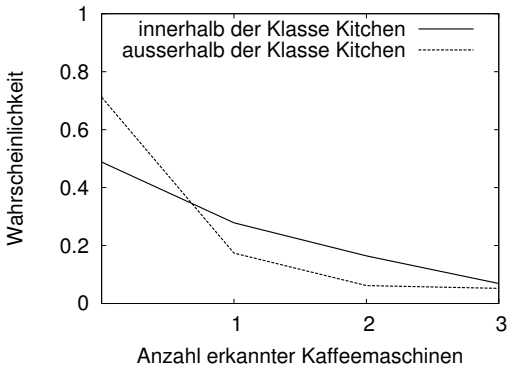


Abbildung 6.2.: Die Verteilung erwarteter Kaffeemaschinen innerhalb und außerhalb der Klasse Kitchen. Dazu wird der Klassifikators „Kaffeemaschine“ auf die positiven und negativen Panoramaaufnahmen der Datenbank angewandt.

Klasse	Anzahl positiver Trainingsdaten
laboratory	10000
corridor	6000
doorway	1500
kitchen	3000
seminar	6000
office	12000

Tabelle 6.1.: Die Anzahl positiver Trainingsdaten jeder Klasse.

ligen Klassen nur eine unterschiedliche Anzahl von paarweise verschiedenen Laserdaten erzeugt werden. Die Anzahl positiver Daten der einzelnen Klassen ist in Tabelle 6.1 zu sehen. Zum Trainieren eines binären Klassifikators der Klasse k wurden als negative Trainingsdaten alle positiven Daten der Klassen $k' \in \{\mathcal{Y} \backslash k\}$ verwendet.

Ein wichtiger Parameter des AdaBoost Algorithmus ist die Anzahl der Iterationen T und damit die Menge schwacher Klassifikatoren, die zum Trainieren eines starken binären Klassifikators verwendet werden. Mit jedem binären Klassifikator wurden verschiedene Experimente unter Verwendung von maximal 500 schwachen Klassifikatoren durchgeführt. Eine Grafik mit den Klassifikationsfehlern für den Klassifikator Doorway ist in Abbildung 6.3 zu sehen. Letztlich legte das erste globale Minimum dieser Fehlerfunktion die Anzahl T der zu verwendenden schwachen Klassifikatoren fest. In Tabelle 6.2 ist die jeweilige Anzahl T verwendeter schwacher Klassifikatoren der einzelnen binären Klassifikatoren, sowie der entsprechende Trainingsfehler abgebildet.

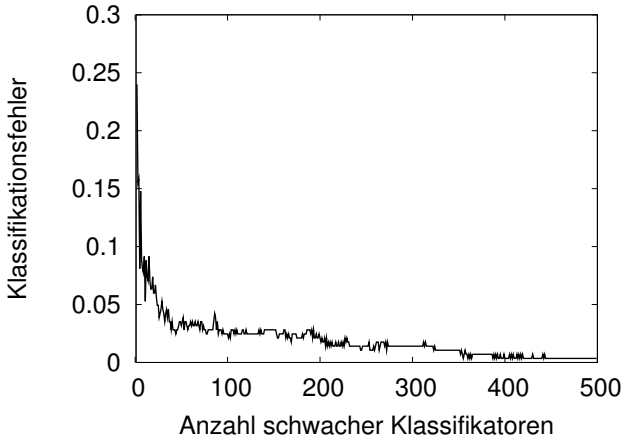


Abbildung 6.3.: Die Fehlklassifikation unter Verwendung von unterschiedlich vielen schwachen Klassifikatoren T.

Binärer Klassifikator	T	Trainingsfehler [%]
kitchen	6	0,30
corridor	3	0,31
doorway	472	0,35
laboratory	380	2,47
seminar room	452	7,29
office	3	8,39

Tabelle 6.2.: Die Anzahl verwendeter schwacher Klassifikatoren T der jeweiligen binären Klassifikatoren und der entsprechende Trainingsfehler.

6.1.2. Bestimmung der Reihenfolge der Entscheidungsliste

Ebenso wurde die beste Sequenz der Entscheidungsliste H, bestehend aus den starken binären Klassifikatoren, anhand einer Testmenge bestimmt. Dafür wurden die Trainingsdaten mit allen Permutationen der Sequenz klassifiziert, um so das Minimum der Fehlklassifikationen zu bestimmen. Dieses Vorgehen hat exponentielle Laufzeit, da es $K!$ verschiedene Permutationen gibt. Da die Anzahl verwendeter Klassen in dieser Arbeit jedoch auf sechs begrenzt ist, war die Bestimmung der optimale Sequenz innerhalb von 6 Stunden möglich. Die optimale Sequenz lautet: Corridor, Doorway, Kitchen, Laboratory, Seminar Room, Office. Diese Sequenz wurde dann in den Experimenten mit den beiden Beobachtungsmodellen aus den Sektionen 5.2 und 5.3 verwendet. Des Weiteren hat sich gezeigt, dass in unseren Experimenten, falls man die binären Klassifikatoren nach der Heuristik aus Algorithmus 5 bestimmt, ebenfalls sehr gute Klassifikationsraten erreicht werden.

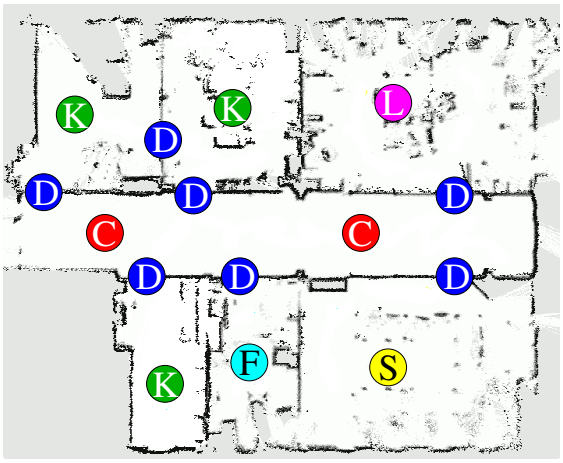
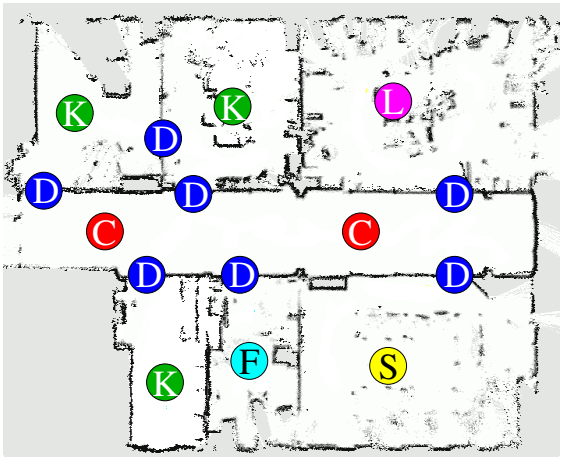


Abbildung 6.4.: Die korrekten Bezeichnungen der individuellen Bereiche in der Trainingsumgebung.

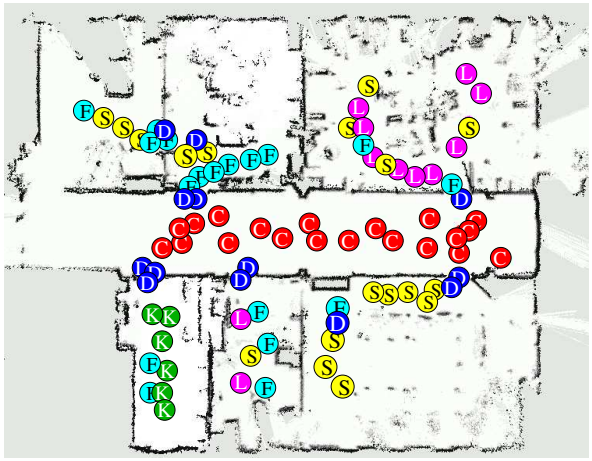
6.2. Klassifikation von Beobachtungen entlang einer Trajektorie

In diesem ersten Experiment soll demonstriert werden, dass die Entscheidungsliste, ohne und in Kombination mit einem HMM, eine robuste Klassifikation von Positionen eines Roboters entlang einer Trajektorie liefert. Dafür wurde die Entscheidungsliste nach dem eben beschriebenen Verfahren mit Laser- und Kameradaten aus der rechten Hälfte der Umgebungskarte aus Abbildung 6.1 erzeugt. Diese Umgebung besteht aus den sechs Klassen Corridor Doorway, Kitchen, Laboratory, Office sowie Seminar Room und die wahren Bereiche der einzelnen Klassen sind in Abbildung 6.4 gekennzeichnet.

Um die anhand dieser Daten erzeugten Klassifikatoren und Entscheidungsliste zu testen, wurde der Roboter durch die rechte, wie auch durch die linke Hälfte der Karte aus Abbildung 6.1 gesteuert, um echte Testdaten entlang einer Trajektorie zu sammeln. Diese Testdaten wurden dann mit und ohne HMM klassifiziert und mit den korrekten Bezeichnungen verglichen. Die Klassifikationsrate in der rechten Hälfte der Karte ohne Verwendung eines HMM lag bei 73,7%. Die Positionen und ihre klassifizierten Bezeichnungen sind in Abbildung 6.5(b) zu sehen. Wurde zusätzlich ein HMM mit dem Beobachtungsmodell aus Sektion 5.2 verwendet, verbesserte sich die Rate auf 87,9%. Diese Positionen sind in Abbildung 6.6(a) abgebildet. Bei der Klassifikation mit dem Beobachtungsmodell aus Sektion 5.3 ergab sich eine Rate von 90,9%, wobei diese Positionen in Abbildung 6.6(b) zu sehen sind.



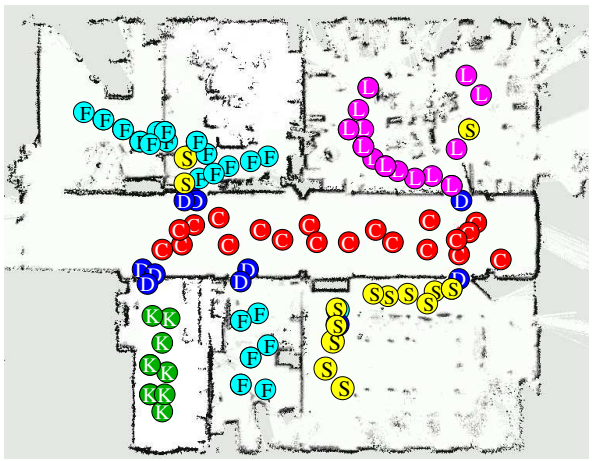
(a) Korrekte Bezeichnung der einzelnen Bereiche



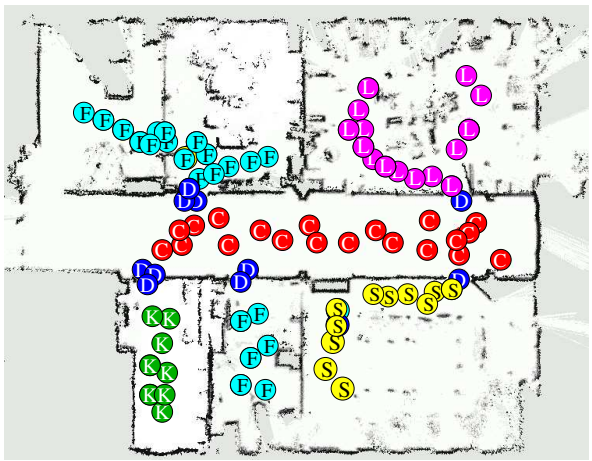
(b) Klassifikation ohne ein HMM

C Corridor D Doorway L Laboratory
S Seminar F Office K Kitchen

Abbildung 6.5.: Klassifikationen von Beobachtungen entlang einer Trajektorie eines Roboters in der rechten Hälfte von Abbildung 6.1 Dazu sind in 6.5(a) die korrekten Bezeichnungen der einzelnen Bereiche und in 6.5(b) die Klassifikationen der einzelnen Beobachtungen ohne ein HMM dargestellt.



(a) Klassifikation mit einem HMM und dem Beobachtungsmodell aus Sektion 5.2



(b) Klassifikation mit einem HMM und dem Beobachtungsmodell aus Sektion 5.3

C Corridor D Doorway L Laboratory
S Seminar F Office K Kitchen

Abbildung 6.6.: Klassifikationen von Beobachtungen entlang einer Trajektorie eines Roboters in der rechten Hälfte von Abbildung 6.1. Dazu sind in 6.6(a) die Klassifikationen der Beobachtungen mit ein HMM und dem Beobachtungsmodell von Sektion 5.2 und in 6.6(b) die Klassifikationen mit dem Beobachtungsmodell von Sektion 5.3 dargestellt.

Vergleicht man diese Klassifikationen miteinander, ist sehr schön zu erkennen, dass mit Hilfe eines HMM die Anzahl der Ausreißer drastisch reduziert wird. Dabei ist der Unterschied zwischen den beiden Beobachtungsmodellen jedoch nicht besonders groß. In diesem Experiment werden beispielsweise die Fehlklassifikationen in der Klasse Kitchen komplett durch das HMM verbessert. Die meisten Fehlerklassifikationen mit einem HMM liegen danach vorwiegend an zweideutigen Stellen in Räumen. So beispielsweise in den beiden nördlichen Büros. An diesen Stellen ist unklar, ob es sich um einen Türdurchgang oder einen Raum handelt.

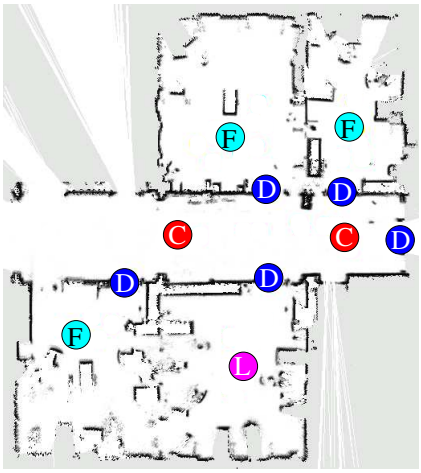
Des Weiteren hat sich mit Hilfe eines two-sample t test [Cohen, 1995] gezeigt, dass diese Verbesserung durch das HMM signifikant auf einem Level von $\alpha = 0{,}01$ ist. Jedoch kann nicht gezeigt werden, dass der Unterschied zwischen den beiden Beobachtungsmodellen aus Abbildung 6.6 auf Signifikanz schließen lässt. In unseren Experimenten sind die Klassifikationsraten mit dem Beobachtungsmodellen aus Sektion 5.3 etwas besser. Jedoch waren unsere Testdaten nicht umfangreich genug, um auf eine signifikante Verbesserung schließen lässt.

Zusätzlich wurde ebenfalls in der linken Hälfte der Umgebungskarte aus Abbildung 6.1 eine Trajektorie von Beobachtungen mit dem Roboter aufgenommen. Für dieses Experiment wurden dieselben binären Klassifikatoren wie auch dieselbe Entscheidungliste verwendet, die auch zur Klassifikation der Trajektorie der rechten Hälfte verwendet wurde. Dabei wurde mit der Entscheidungliste eine Klassifikationrate von 75,4% erreicht, was in Abbildung 6.7(b) zu sehen ist. Wird nun ebenfalls eine HMM mit dem Beobachtungsmodell aus Sektion 5.2 verwendet werden 91,2% erreicht, entsprechend 91,2% mit dem Beobachtungsmodell nach Sektion 5.3. Die einzelnen Positionen sind in Abbildung 6.7(c) und 6.7(d) zu sehen. Diese Verbesserung unter Verwendung eines HMM ist ebenfalls signifikant auf einem Level von $\alpha = 0{,}01$.

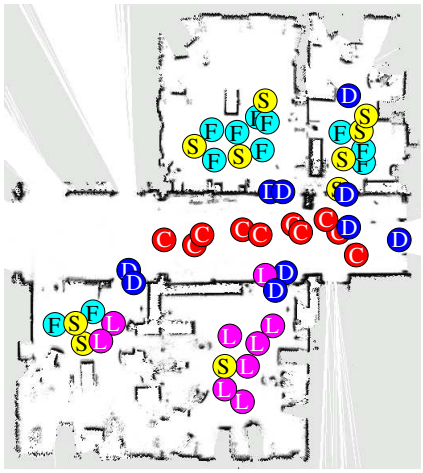
Durch dieses Experiment wurde gezeigt, dass basierend auf Laser- und Kameradaten eine robuste Klassifikation von Beobachtungen eines Roboters erreicht werden kann. Des Weiteren kann unter Verwendung eines HMM die Gesammtklassifikationsrate noch einmal deutlich verbessert werden.

6.3. Verwendung von verschiedenen schwachen Klassifikatortypen $h_j(x)$

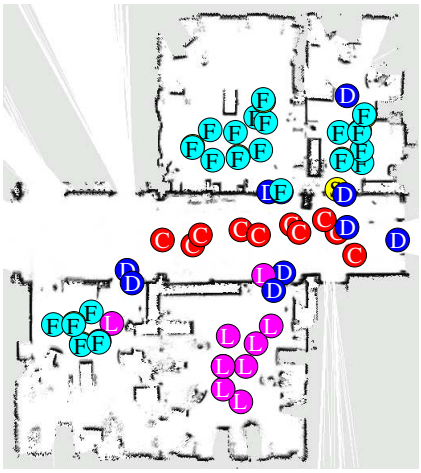
In diesem Experiment werden die Klassifikationsraten unter Verwendung von verschiedenen schwachen Klassifikatoren aus Kapitel 3.4 miteinander verglichen. In dem vorherigen Kapitel wurden die verwendeten binären Klassifikatoren je mit dem kompletten Satz schwacher Klassifikatoren trainiert. Dabei wurden schwache Klassifikatoren nach Gleichung (3.11) und (3.12), sowohl für die Merkmale der Laserdaten wie auch der Kameradaten verwendet. Dieses Vorgehen wird nun mit den Klassifikationsraten unter



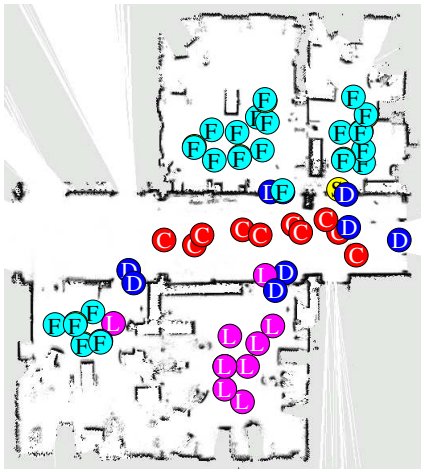
(a) Korrekte Bezeichnung der einzelnen Bereiche



(b) Klassifikation ohne ein HMM



(c) Klassifikation mit einem HMM und dem Beobachtungsmodell aus Sektion 5.2



(d) Klassifikation mit einem HMM und dem Beobachtungsmodell aus Sektion 5.3

C Corridor D Doorway L Laboratory
S Seminar F Office K Kitchen

Abbildung 6.7.: Klassifikationen von Beobachtungen entlang einer Trajektorie eines Roboters in der linken Hälfte von Abbildung 6.1. Dabei sind in 6.7(a) die korrekten Bezeichnungen der Bereiche dargestellt. In 6.7(b) ist die Klassifikation der Beobachtungen ohne ein HMM und in 6.7(c) und 6.7(d) die Klassifikation mit einem HMM und den Beobachtungsmodellen aus Sektion 5.2 und 5.3 zu sehen.

Trajektorie aus Abbildung	Laser $h_j^1(x)$	Laser $h_j^2(x)$	Kamera $h_j^1(x)$	Kamera $h_j^2(x)$	Klassifikationsraten [%]
6.5	✓	✓	✓	✓	90,9
6.7	✓	✓	✓	✓	91,2
6.5		✓		✓	73,7
6.7		✓		✓	70,2
6.5	✓		✓		89,9
6.7	✓		✓		91,2

Tabelle 6.3.: Die Klassifikationsraten unter Verwendung von verschiedenen Kombinationen der schwachen Klassifikationstypen. Dabei werden $h_j^1(x)$ nach Gleichung (3.11) und $h_j^2(x)$ nach Gleichung (3.12) gebildet.

Verwendung verschiedener Kombinationen dieser schwachen Klassifikatortypen verglichen. Dazu wurden die gleichen Trainingsdaten wie im vorherigen Kapitel verwendet und jeweils binäre Klassifikatoren mit den verschiedenen Kombinationen von schwachen Klassifikatoren trainiert. Danach wurde ebenfalls die beste Sequenz der Entscheidungsliste nach dem gleichen Verfahren ermittelt und die gleichen Trajektorien zur Klassifikation verwendet. Die Klassifikationsraten der jeweiligen Multi-Klassen Klassifikatoren ist in Tabelle 6.3 zu sehen.

In dieser Tabelle sieht man deutlich, dass die Verwendung von schwachen Klassifikatoren nach der Gleichung (3.12) einen Vorteil für die Klassifikationsraten bringen. Es werden besser oder mindestens gleich gute Klassifikationsraten erreicht.

6.4. Vergleich mit vorherigen Ansätzen

In der Arbeit von Martínez Mozos *et al.* [2005] wurden die Klassen Corridor, Doorway und Room lediglich anhand von Lasermerkmalen erkannt. In diesem Experiment wollen wir zeigen, dass unter Verwendung von Kameradaten selbst diese Klassen robuster klassifiziert werden können. Dazu werden nun die beiden Ansätze miteinander verglichen, wobei im einen Fall, nur Laserdaten verwendet werden. Um ein repräsentatives Ergebnis zu erhalten, wurden lediglich binäre Klassifikatoren für die Klassen Corridor, Doorway und Room trainiert. Die Klassen Kitchen, Office, Laboratory und Seminar wurden dafür zur Klasse Room vereint, da diese schlecht anhand der Lasermerkmale klassifiziert werden können. Danach wurden die Klassifikationsraten beider Ansätze für alle möglichen Permutationen der Entscheidungsliste verglichen. Die Ergebnisse sind in Tabelle 6.4 zu sehen. Man sieht daran, dass die Kombination von Laser- und Kameradaten in allen Fällen eine bessere oder wenigstens eine gleich gute Klassifikationsrate erreicht.

Ebenso kann man zeigen, dass die Erweiterung um Kamerainformationen die Klassifi-

Entscheidungsliste	Fehler [%] Laser	Fehler [%] Laser & Kamera
corridor-doorway	3,21	1,87
doorway-room	3,74	2,67
doorway-corridor	3,21	2,14
room-corridor	1,60	1,34
corridor-room	1,60	1,34
room-doorway	1,60	1,60
Durchschnitt	2,50	1,83

Tabelle 6.4.: Klassifikationsfehler unter Verwendung von ausschließlich Laserdaten im Vergleich zur Verwendung von Laser- und Kameradaten.

kationsrate von Klassen, die allein mit Laserdaten nur schlecht erkannt werden können, extrem verbessert. Ein typisches Beispiel dafür sind die Klassen Seminar Room und Laboratory. Sie haben etwa eine gleiche Raumstruktur und enthalten beide zum größten Teil Tische und Stühle. Es wurde jeweils ein binärer Klassifikator trainiert, wobei nur Trainingsdaten aus diesen beiden Klassen verwendet wurden. Eine Entscheidungsliste mußte nicht bestimmt werden, da eine negative Klassifikation der einen Klasse, sofort der anderen Klasse zugeordnet wird. Die Fehlklassifikation der Klasse Seminar Room fällt von 46,9% auf 6,3% bei zusätzlicher Verwendung von Kameradaten. Entsprechend sinkt die Fehlklassifikation der Klasse Laboratory von 34,4% auf 3,1%. Dies ist eine beträchtliche Verbesserung, was auf der Tatsache beruht, dass diese beiden Räume hauptsächlich durch Objekte wie „Monitor“ unterscheidbar sind. Dieses Merkmal ist ausschließlich mit einem Laserscanner nicht zu erkennen. Hier liefert ebenfalls ein two-sample t test ein Level von $\alpha = 0{,}01$, was auf eine signifikante Verbesserung schließen läßt.

6.5. Lokalisierung mit Hilfe semantischer Informationen von Roboterpositionen

In diesem Experiment wird ein Anwendungsbeispiel präsentiert, in dem die Klassifikation von Beobachtungen des Roboters erfolgreich Verwendung finden kann. Es soll dabei illustriert werden, wie semantische Informationen der Beobachtungen eines Roboters in der Lokalisierung von mobilen Robotern verwendet werden kann. Dazu wird die bisherige Vorgehensweise, die auf der Odometrie basiert, um die Klassifikation von Beobachtungen erweitert. Daher wird im Folgenden eine kleine Einführung in die Lokalisierung mobiler Roboter gegeben, um danach in einem Experiment die Verbesserung zu präsentieren.

6.5.1. Lokalisierung mobiler Roboter

Das Problem bei der Lokalisierung mobiler Roboter besteht darin, die aktuelle Position des Roboters relativ in seiner Umgebung zu schätzen und stellt eines der Schlüsselprobleme der mobilen Robotik dar. Zum Beispiel spielt dieses Problem in den Arbeiten von Cox und Wilfong [1990], Leonard *et al.* [1992] oder Simmons *et al.* [1997] eine zentrale Rolle.

Die Lokalisierung mobiler Roboter kann in verschiedene Teilprobleme aufgeteilt werden [Thrun *et al.*, 2001]. Das wohl einfachste und bisher auch am meisten beachtete Lokalisierungsproblem ist die sogenannte *Positionsverfolgung* [Borenstein *et al.*, 1996]. Hierbei ist die Startposition des Roboters in seiner Umgebung bekannt und das Problem besteht darin, die inkrementellen Fehler der Odometriedaten des Roboters zu kompensieren. Die meisten Algorithmen machen dazu eine Abschätzung des Fehlers um ein Gebilde der Unsicherheit der aktuellen Roboterposition zu generieren. Das zweite und etwas anspruchsvollere Problem ist die *Globale Lokalisierung* [Burgard *et al.*, 1998], da hier die Startposition des Roboters ebenfalls unbekannt ist. Das dritte und schwierigste Problem, ist das sogenannte *Entführter Roboter* Problem [Fox *et al.*, 1999], in dem der Roboter während der Lokalisierung an eine unbekannte Position versetzt wird. Dieses Problem unterscheidet sich von der globalen Lokalisierung, da der Roboter nichts von der Entführung weiß und weiterhin glaubt, er sei in der Nähe der vorherigen Position. All diese Probleme sind natürlich besonders schwer zu lösen, falls sich der Roboter in einer dynamischen Umgebung befindet [Thrun *et al.*, 2000]. Dabei könnten beispielsweise die Sensormessungen durch sich bewegende Menschen gestört werden.

In dieser Arbeit ist ausschließlich die globale Lokalisierung des Roboters in einer typischen Büroumgebung von Interesse. Im folgenden Abschnitt wird daher die Idee der Lokalisierung an einem Beispiel erläutert und danach der verwendete Ansatz der Monte-Carlo Lokalisierung [Dellaert *et al.*, 1998] vorgestellt.

Grundidee der wahrscheinlichkeitsbasierten Lokalisierung

Bevor genauere Details der globalen Lokalisierung mobiler Roboter vorgestellt werden, wird zuerst die Grundidee an einem Beispiel erläutert. Der Grundgedanke der wahrscheinlichkeitsbasierten Lokalisierung besteht darin, die Unsicherheit über die Position des Roboters durch eine Wahrscheinlichkeitsverteilung über alle möglichen Positionen zu beschreiben. Im Initialisierungszustand sind alle Positionen gleich wahrscheinlich und werden daher auch gleich gewichtet. Der grundlegende Ablauf sieht dann wie folgt aus: Basierend auf den Sensor- und Odometriedaten wird die Verteilung iterativ im Laufe der Zeit verfeinert. Dabei sollen wahrscheinlichere Positionen eine höhere Gewichtung erhalten.

Sei der Roboter nun an einer ihm unbekannten Stelle in der Umgebung aus Abbil-

dung 6.8(a) plaziert. Zur Vereinfachung sei die Umgebung eindimensional und der Roboter bewegt sich ausschließlich in der Horizontalen. In dem unteren Teil dieser Abbildung ist die Wahrscheinlichkeitsverteilung über alle möglichen Positionen dargestellt. Hierbei sind im Initialisierungszustand alle Positionen gleich wahrscheinlich und daher auch gleich gewichtet.

Nehmen wir nun an, der Roboter würde durch seine Sensoren an seiner derzeitigen Position eine Tür in seiner unmittelbaren näheren Umgebung erkennen. Bei der wahrscheinlichkeitsbasierten Lokalisierung werden nun die Positionen, die sich in der Nähe einer Tür befinden, derart modifiziert, dass sich die Gewichtung dieser Positionen erhöht und alle anderen Gewichtungen verringern. Dies ist in Abbildung 6.8(b) zu sehen. Dabei sei angemerkt, dass die Positionen, die sich nicht in unmittelbarer Umgebung einer Tür befinden, keine Gewichtung von Null erhalten. Das beruht auf der einfachen Tatsache, dass die Sensoren der Roboter immer mit einer gewissen Unsicherheit behaftet sind. Durch dieses Vorgehen ist gewährleistet, dass bei falschen Sensorinformationen die Lokalisierung nicht sofort divergiert. Es besteht nach einer Fehlklassifikation somit immer noch die Möglichkeit, durch richtig folgende Sensorinformationen die korrekte Position des Roboters zu lokalisieren.

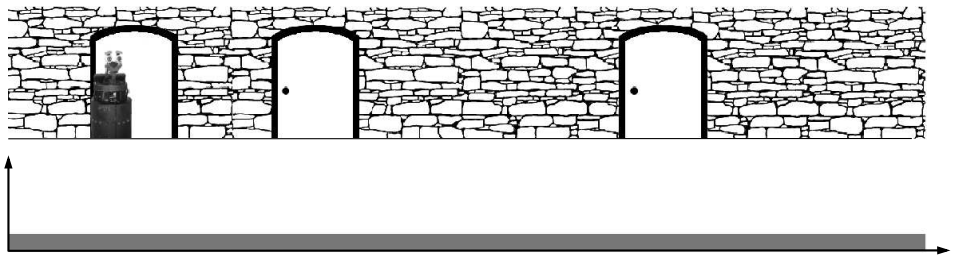
Als nächstes würde sich der Roboter eine gewissen Strecke nach rechts bewegen, wie es in Abbildung 6.8(c) zu sehen ist. In der wahrscheinlichkeitsbasierten Lokalisierung wird diese Information verwendet, um die möglichen Positionen und ihre Gewichte dementsprechend zu verschieben. Hierbei sei angemerkt, dass aufgrund der Unsicherheit in der Roboterbewegung die Verteilung der Gewichte geglättet werden.

Nehmen wir nun wiederum an, der Roboter würde an seiner neuen Position eine weitere Tür erkennen. Diese Information wird nun ebenfalls in den Gewichten der einzelnen Positionen verarbeitet, was zur letztlichen Verteilung aus Abbildung 6.8(d) führt. Zu diesem Zeitpunkt sind nun nach lediglich zwei Sensormessungen und einer Bewegung des Roboters die wahrscheinlichsten Positionen um einen einzigen Standort zentriert. Der Roboter kann sich daher sehr sicher über seine derzeitige Position sein. Des Weiteren sei darauf hingewiesen, dass die letzte Verteilung über allen Positionen ebenfalls nur noch aus fünf verschiedenen Spitzen besteht.

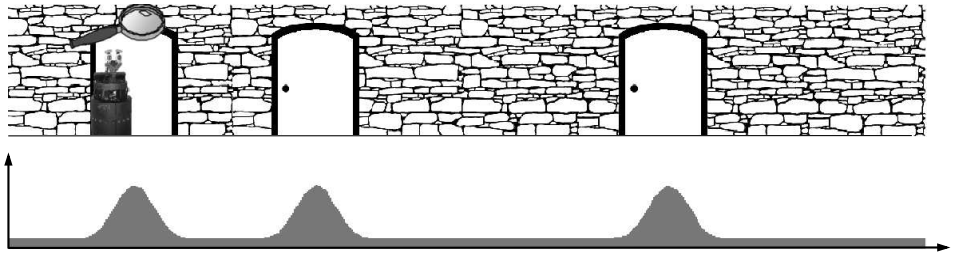
Monte-Carlo Lokalisierung

Im vorherigen Abschnitt wurde die Grundidee der wahrscheinlichkeitsbasierten Lokalisierung von mobilen Robotern an einem Beispiel kurz erläutert. In diesem Abschnitt wird nun das Verfahren der Monte-Carlo Lokalisierung vorgestellt, das für die Experimente in dieser Arbeit verwendet werden.

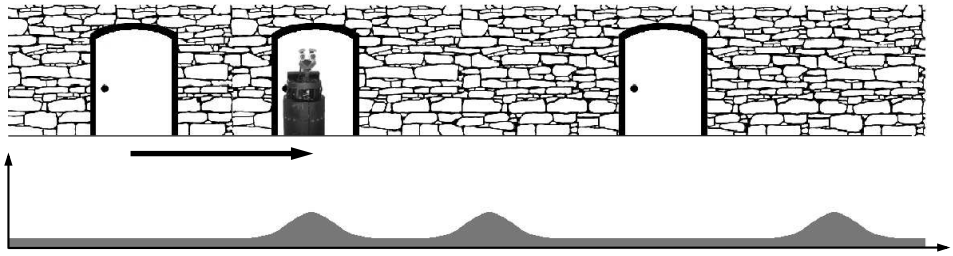
Die Monte-Carlo Lokalisierung (MCL) ist ein rekursiver Bayes-Filter, der die posteriori Wahrscheinlichkeit der globalen Position des Roboters anhand der Informationen über die Umgebung und der eigenen Bewegung schätzt. Die Grundidee des Bayes-Filters be-



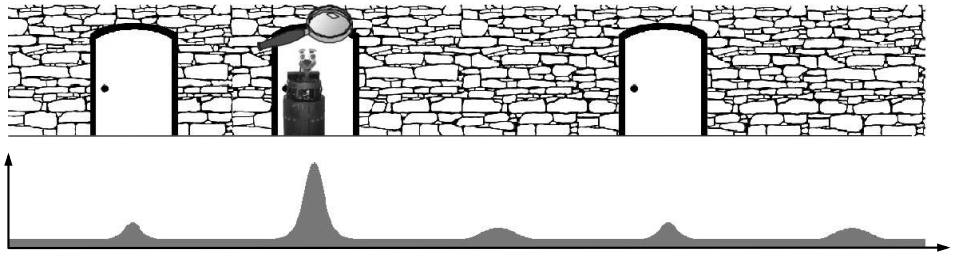
(a) Anfangszustand



(b) Durch die Sensoren wird eine Tür erkannt



(c) Der Roboter bewegt sich eine gewisse Strecke nach rechts



(d) Durch die Sensoren wird wieder eine Tür erkannt

Abbildung 6.8.: Ein Beispiel zur wahrscheinlichkeitsbasierten Lokalisierung mobiler Roboter.

steht darin, eine Wahrscheinlichkeitsverteilung für alle möglichen Roboterpositionen zu bestimmen. Dazu wird der *Glaube* für jede Position

$$P(\mathbf{x_t}) = p(\mathbf{x_t} \mid d_{0\ldots t})$$

anhand der Informationen über die Umgebung und der eigenen Bewegung bestimmt. Dabei ist $\mathbf{x_t}$ ein drei-dimensionaler Vektor $\mathbf{x_t} = [x, y, \theta]^T$ der Position des Roboters, entsprechend den Koordinaten (x, y) und der Orientierung θ, und $d_{0\ldots t}$ sind die Informationen, die vom Zeitpunkt 0 bis t gesammelt wurden. Im einzelnen bestehen die Informationen $d_{0\ldots t} = (z_t, u_{t-1}, z_{t-1}, u_{t-2}, \ldots, z_0)$ aus den Sensordaten z und den Odometriedaten u. Des Weiteren wird angenommen, dass die Positionen $\mathbf{x_t}$ einem Markov-Prozess erster Ordnung entsprechen und nur von der vorangegangenen Position $\mathbf{x_{t-1}}$ und der Aktion u_{t-1} abhängen. Somit ergibt sich die folgende Gleichung

$$P(\mathbf{x_t}) = p(\mathbf{x_t} \mid z_t, u_{t-1}, z_{t-1}, u_{t-2}, \ldots, z_0). \tag{6.1}$$

Die Berechnung des Glaubens $P(\mathbf{x_t})$ jeder Position wird rekursiv zu jedem Zeitpunkt t ausgeführt. Dadurch ergibt sich von den möglichen Positionen des Roboters eine Wahrscheinlichkeitsverteilung, die im Laufe der Zeit zu wahrscheinlicheren Positionen tendiert. Zum Zeitpunkt t_0 wird der Glaube $Bel(\mathbf{x_0})$, entsprechend dem Wissen bei der globalen Lokalisierung durch eine Gleichverteilung initialisiert.

Die Gleichung (6.1) kann mit Hilfe des Bayes-Theorem umgeformt werden zu

$$\begin{aligned}
P(\mathbf{x_t}) &= \frac{p(z_t \mid \mathbf{x_t}, u_{t-1}, z_{t-1}, u_{t-2}, \ldots, z_0) p(\mathbf{x_t} \mid u_{t-1}, z_{t-1}, u_{t-2}, \ldots, z_0)}{p(z_t \mid u_{t-1}, z_{t-1}, u_{t-2}, \ldots, z_0)} \\
&= \eta \cdot p(z_t \mid \mathbf{x_t}, u_{t-1}, z_{t-1}, u_{t-2}, \ldots, z_0) p(\mathbf{x_t} \mid u_{t-1}, z_{t-1}, u_{t-2}, \ldots, z_0) \\
&= \eta \cdot p(z_t \mid \mathbf{x_t}) p(\mathbf{x_t} \mid u_{t-1}, z_{t-1}, u_{t-2}, \ldots, z_0) \\
&= \eta \cdot p(z_t \mid \mathbf{x_t}) \int_{\mathbf{x_{t-1}}} p(\mathbf{x_t} \mid \mathbf{x_{t-1}}, u_{t-1}, \ldots, z_0) p(\mathbf{x_{t-1}} \mid u_{t-1}, \ldots, z_0)\, d\mathbf{x_{t-1}} \\
&= \eta \cdot p(z_t \mid \mathbf{x_t}) \int_{\mathbf{x_{t-1}}} p(\mathbf{x_t} \mid \mathbf{x_{t-1}}, u_{t-1}) P(\mathbf{x_{t-1}})\, d\mathbf{x_{t-1}},
\end{aligned} \tag{6.2}$$

wobei η einem Normalisierungsfaktor entspricht. In manchen Umformungen wurde die Eigenschaft der Unabhängigkeit einzelner Variablen von anderen ausgenutzt, beispielsweise dass $\mathbf{x_t}$ einem Markov Prozeß erster Ordnung entspricht. Detailierte Informationen zu den einzelnen Umformungen sind in Thrun *et al.* [2001] zu finden. Die letztlich resultierende Gleichung (6.2) besteht zum einen aus dem Sensormodell $p(z_t \mid \mathbf{x_t})$, das die Wahrscheinlichkeit angibt, dass der Roboter die Sensordaten z_t misst, gegeben er befindet sich an Position $\mathbf{x_t}$. Zum anderen beinhaltet die Gleichung das Bewegungsmodell $p(\mathbf{x_t} \mid \mathbf{x_{t-1}}, u_{t-1})$, das die Wahrscheinlichkeit angibt, dass der Roboter von Position $\mathbf{x_{t-1}}$ durch Aktion u_{t-1} an Position $\mathbf{x_t}$ gelangt. Diese beiden Modelle sind stationär und somit

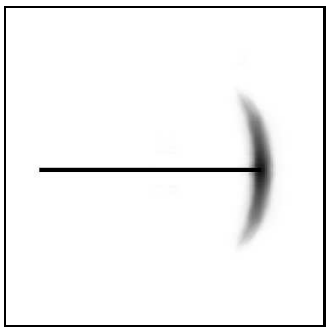


Abbildung 6.9.: Das Bewegungsmodell $p(\mathbf{x_t} \mid \mathbf{x_{t-1}}, u_{t-1})$ für eine Aktion von 20 Metern nach vorne. Dabei entsprechen jeweils dunklere Bereiche einer höheren Wahrscheinlichkeit.

unabhängig von der aktuellen Zeit t. Die beiden verwendeten Modelle werden im Folgenden kurz vorgestellt, werden aber auch in der Arbeit von Fox *et al.* [1999] ausführlich erklärt.

Das Bewegungsmodell $p(\mathbf{x_t} \mid \mathbf{x_{t-1}}, u_{t-1})$ ist eine generelle Bewegungswahrscheinlichkeit des Roboters. Die Odometriedaten u_{t-1} beschreiben dabei die ausgeführte Aktion des Roboters und somit wie sich die Position $\mathbf{x_t}$ relativ zur vorherigen $\mathbf{x_{t-1}}$ geändert hat. Diese Angaben sind oft fehlerhaft, da sie physikalisch gemessen werden. Eine Position des Roboters ist durch den drei-dimensionalen Vektor $\mathbf{x_t}$ eindeutig beschrieben. Somit bestehen die Odometriedaten aus drei Parametern. Dabei beschreiben die Parameter zum einen die Distanz zwischen den beiden Positionen $\mathbf{x_t}$ und $\mathbf{x_{t-1}}$, den Winkel, in dem sich Positionen $\mathbf{x_t}$ relativ zur ersten $\mathbf{x_{t-1}}$ befindet, sowie die Orientierung an der neuen Position. Um diese Unsicherheit der Parameter zu beschreiben, wird der Fehler durch je eine bei Null zentrierte Gaußverteilungen simuliert. Je eine Gaußverteilung ist dabei entsprechend dem Entfernungfehler und eine dem Rotationsfehler ausgerichtet. Ein Beispiel eines solchen Bewegungsmodells ist in Abbildung 6.9 zu sehen. Dabei ist die Position $\mathbf{x_{t-1}}$ links und die möglichen Positionen $\mathbf{x_t}$ durch eine Wahrscheinlichkeitsverteilung rechts dargestellt. Dabei entsprechen dunklere Werte einer höheren Wahrscheinlichkeit.

Das Sensormodell $p(z_t \mid \mathbf{x_t})$ dagegen ist ein Maß für die Wahrscheinlichkeit, dass der Roboter die Sensordaten z_t wahrnimmt, gegeben er befindet sich an Position $\mathbf{x_t}$. Dazu müssen natürlich die Sensordaten $z_{\mathbf{x}_t}$ der Position $\mathbf{x_t}$ bekannt sein, um sie mit den Sensordaten z_t vergleichen zu können. Für dieses Experiment bestehen die Sensordaten jedoch lediglich aus den semantischen Information entsprechend der Position des Roboters. Somit vereinfacht sich das Sensormodell zu $p(z_t \mid \xi_t)$, wobei $z_t \in \mathcal{Y}$ die Klasse des Klassifikators H und $\xi_t \in \mathcal{Y}$ die Klasse entsprechend der Position $\mathbf{x_t}$ in der Umgebungskarte darstellt. Der Klassifikator H liefert somit basierend auf den Laser- und Kameradaten eine Klasse aus $\mathcal{Y}$, die mit der Klasse der Position $\mathbf{x}_t$ verglichen wird. Die Wahrscheinlichkeiten des Sensormodells entsprechen somit den Klassifikationswahr-

scheinlichkeiten des Klassifikator H und wurden im Beobachtungsmodell in Kapitel 5.2 bereits vorgestellt. Im Detail kann das Sensormodell nun wie folgt angegeben werden

$$p(z_t \mid \mathbf{x_t}) = p(z_t \mid \xi_t) = p(z_t \mid m, \mathbf{x_t}) \cdot p(\mathbf{x_t} \mid m).$$

Dabei ist m die Umgebungskarte, damit entsprechend der Position $\mathbf{x_t}$ die Klasse ξ_t bestimmt werden kann. Des Weiteren gibt der Term $p(\mathbf{x_t} \mid m) \in \{0, 1\}$ an, ob es sich bei der Position $\mathbf{x_t}$ um eine reguläre Position in der Umgebungskarte m handelt. So wäre dieser Wert beispielsweise 0, falls die Postion $\mathbf{x_t}$ mit einer Wand oder einem Hindernis übereinstimmt.

Die Positionsschätzung wird somit letztlich durch die Formel

$$P(\mathbf{x_t}) = \eta \cdot p(z_t \mid m, \mathbf{x_t})\; p(\mathbf{x_t} \mid m) \int_{\mathbf{x_{t-1}}} p(\mathbf{x_t} \mid \mathbf{x_{t-1}}, u_{t-1}) P(\mathbf{x_{t-1}})\; d\mathbf{x_{t-1}} \tag{6.3}$$

berechnet.

Im Falle eines kontinuierlichen Zustandraums, wie bei der mobilen Roboter Lokalisierung, ist es jedoch nicht trivial einen effizienten Algorithmus für dieses Problem zu implementieren. Aus diesem Grund wird bei der MCL der Glaube $P(\mathbf{x_t})$ lediglich für eine begrenzte Menge von p gewichteten Stichproben

$$P(\mathbf{x_t}) \approx \left\{ \mathbf{x_t^{(i)}}, w_t^{(i)} \right\}_{i=1,\ldots,p},$$

sogenannten *Partikeln*, dargestellt. Dabei entspricht $\mathbf{x^{(i)}}$ einer Stichprobe und repräsentiert eine Position und $w^{(i)}$ einer nicht negativen Gewichtung, die sich insgesamt zu eins aufsummieren. Diese Partikel werden zu Beginn gleichmäßig über die gesammte Karte verteilt und sollen somit die möglichen Positionen des Roboters approximieren. Das Ziel dabei liegt darin, die Partikel im Laufe der Zeit entsprechend dem Sensormodell $p(z_t \mid \mathbf{x_t})$ zu gewichten und nach dem Bewegungsmodell $p(\mathbf{x_t} \mid \mathbf{x_{t-1}}, u_{t-1})$ zu verschieben. Für dieses Vorgehen wurden in der Vergangenheit bereits verschiedene Algorithmen vorgestellt. Generell ist diese Methode jedoch als *Partikle Filter* bekannt und dazu wird in [Doucet, 1998] ein Überblick gegeben.

Das Vorgehen sieht im Detail nun wie folgt aus: Die Partikel werden zu Beginn gemäß einer Gleichverteilung über die gesammte Karte verteilt. Danach werden sie in Analogie zur Gleichung (6.3) nach dem Sensormodell $w_t^{(i)} = p(z_t \mid m, \mathbf{x_t^{(i)}})$ gewichtet und nach dem Bewegungsmodell $p(\mathbf{x_t^{(i)}} \mid \mathbf{x_{t-1}^{(i)}}, u_{t-1})$ verschoben. Damit sich die Partikel im Laufe der Zeit auf wahrscheinlichere Bereiche konzentrieren, werden sie nach diesen beiden Schritten neu verteilt. Dazu werden p neue Partikel gemäß der Gewichtung $w_t^{(i)}$ zufällig aus den alten Partikeln gezogen. Durch diesen iterativen Durchlauf überleben wahrscheinlichere Partikel nach dem Prinzip *Überleben der Tüchtigsten* und unwahrscheinlichere Partikel sterben im Laufe der Zeit aus.

6.5.2. Ergebnisse

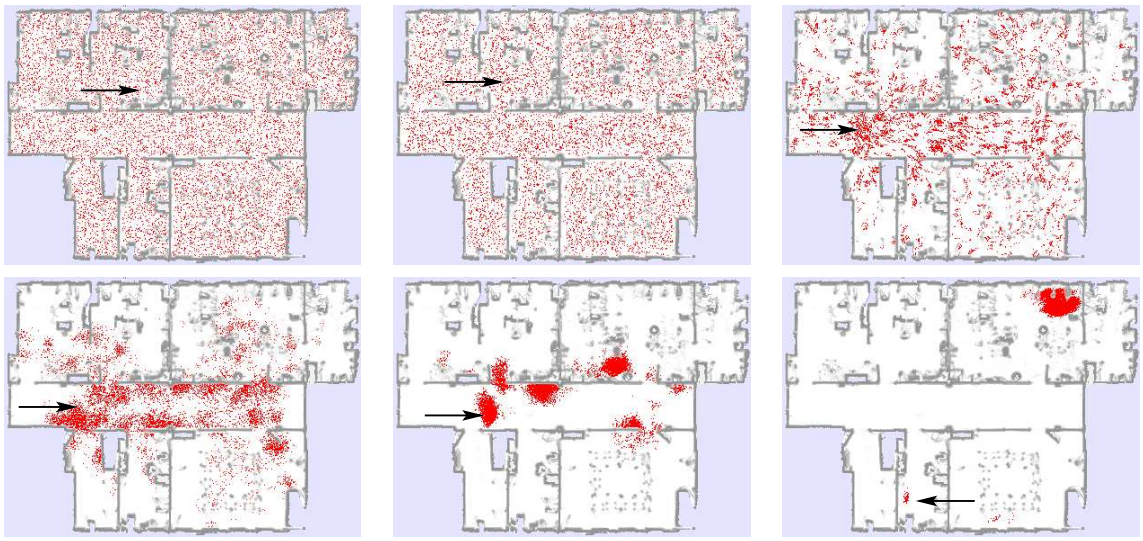
In dem vorangegangenen Abschnitt wurde eine kleine Einführung in die Lokalisierung mobiler Roboter gegeben. Nun wird dieses Verfahren angewandt, um ein Beispiel anzugeben, in dem die Klassifikation der Beobachtungen von Roboterpositionen erfolgreich angewendet werden kann.

Dazu wird der Ansatz, der lediglich auf den Odometriedaten basiert, mit dem Ansatz unter Verwendung von semantischen Informationen verglichen. Dementsprechend vereinfacht sich für den ersten Ansatz das Sensormodell aus Gleichung (6.3) zu

$$p(z_t \mid \mathbf{x_t}) = p(\mathbf{x_t} \mid m).$$

Da hierbei nur die Odometriedaten verwendet werden, ist lediglich zu überprüfen, ob es sich um eine korrekte Position in der Karte handelt. Die Gewichtung der Partikel wird somit auf Null gesetzt, sobald sie sich gegen eine Wand oder ein Hindernis verschieben. Unter Verwendung der semantischen Informationen würde dagegen auch die Gewichtung von Partikeln, die sich in unwahrscheinlichen Klassen befinden, herabgesetzt. Bei diesem Experiment wird somit erwartet, dass der zweite Ansatz schneller gegen wahrscheinlichere Positionen konvergiert. Das Ergebnis des Experiments ohne semantische Informationen ist in Abbildung 6.10(a) und mit semantischer Information in Abbildung 6.10(b) zu sehen. Dabei verläuft die zeitliche Reihenfolge von links oben nach rechts unten und jeweils gleichpositionierte Bilder der Abbildungen 6.10(a) und 6.10(b) entsprechen gleichen Zeitpunkten. Das Experiment dauerte insgesamt 428 Sekunden, wobei etwa jede Sekunde einer Iteration entspricht. Die einzelnen Bilder wurden jeweils nach 0, 4, 27, 50, 115, 160 Iterationen erstellt.

In beiden Fällen werden 10.000 Partikel verwendet und die Ausgangsposition des Roboters ist im zweiten Büro von links, nördlich des Korridors. Diese semantische Information führt im zweiten Bild von Abbildung 6.10(b) bereits nach wenigen Iterationen dazu, dass die Partikel in den Klassen Office vermehrt auftreten. Des Weiteren ist sehr schön zu erkennen, dass die korrekte Position des Roboters (Pfeil) mit Hilfe der semantischen Information in Bild 5 bereits sehr gut lokalisiert wurde. Dagegen ist bei dem odometriebasierten Ansatz die Position zum gleich Zeitpunkt noch recht ungenau. Des Weiteren ist in Bild 6 zu sehen, dass ausschließlich mit den Odometriedaten die Position nicht sehr robust lokalisiert werden kann und in diesem Fall sogar fast divergiert.



(a) Globale Lokalisierung mit Odometriedaten

(b) Globale Lokalisierung mit Odometriedaten und semantischer Information

Abbildung 6.10.: Globale Lokalisierung unter Verwendung von semantischen Informationen und Odometriedaten im Vergleich zur Verwendung von ausschließlich Odometriedaten. Jeweils entsprechende Bilder wurden zur gleichen Zeit aufgenommen. Der Pfeil entspricht jeweils der wahren Position des Roboters.

7. Zusammenfassung

In dieser Arbeit haben wir einen neuen Ansatz vorgestellt, um Positionen eines Roboters an verschiedenen Orten zu klassifizieren. Dabei werden die Beobachtungen des Roboters mit semantischen Informationen von typischen Büroumgebungen versehen. Diese Technik basiert auf einfachen geometrischen Merkmalen die aus Laserdaten extrahiert werden, sowie Merkmalen, die aus den Kameradaten des Roboters gewonnen werden. Da diese Merkmale allein für eine robuste Klassifikation nicht geeignet sind, werden sie mit Hilfe des AdaBoost Algorithmus zu einem starken Klassifikator kombiniert. Der originale AdaBoost Algorithmus liefert allerdings nur binäre Klassifikatoren, weshalb wir eine Entscheidungsliste aus diesen Klassifikatoren gebildet haben, um zwischen mehr als nur zwei Klassen unterscheiden zu können. Ebenso wurde die räumliche Abhängigkeit zwischen einzelnen Beobachtungen des Roboter entlang einer Trajektorie ausgenutzt um die Klassifikationsrate zu verbessern. Dafür wurde ein Hidden Markov Model verwendet, das eine wahrscheinlichkeitsbasierte Klassifikationsverteilung über alle Klassen, basierend auf den Sensorinformationen wie auch auf den Odometriedaten, berechnet.

Die vorgestellten Ansätze wurden implementiert und mit einem mobilen Roboter, der mit einem Laserscanner und einer Kamera ausgerüstet war, getestet. In den Experimenten wurde dann mit echten Robotern, wie auch in Simulation demonstriert, dass diese Vorgehensweise für eine robusten Klassifikation von Beobachtungen eines Roboters in Indoor Umgebungen gut geeignet ist. Es wurde dafür eine typische Büroumgebung mit den sechs Klassen Doorway, Corridor, Office, Kitchen, Seminar Room und Laboratory verwendet. Ebenso zeigte sich in den Experimenten, dass die Anwendung eines Hidden Markov Models die Anzahl der Fehlklassifikationen von Beobachtungen entlang einer Trajektorie deutlich verringert und eine signifikant Verbesserung darstellt.

In einem weiteren Experiment wurde illustriert, wie die semantische Information von den Positionen eines Roboters effizient in einem Anwendungsbeispiel verwendet werden kann. Dazu wurde die Lokalisierungstechnik um diese Informationen erweitert, was in unseren Experimenten zu einem schneller konvergierenden wie auch robusteren Verfahren führte. Ebenso wurde in den Experimenten diese Vorgehensweise mit vorherigen Ansätzen verglichen, was zeigte, dass die Hinzunahme von Kamerainformationen die Klassifikationsrate erheblich verbessert. Es kann dadurch nicht nur zwischen mehr Klassen unterschieden werden, sondern steigert auch die Klassifikationsraten von solchen Klassen, die mit den Laserdaten bereits gut klassifiziert werden konnten.

7.1. Ausblick

An dieser Stelle möchten wir einige Punkte unserer Arbeit ansprechen, die man noch verbessern könnte oder um welche Ideen sich der jetzige Ansatz erweitern ließe.

Die Grundbausteine bei der Klassifikation in dieser Arbeit, sind die Merkmale aus den Laser- und Kameradaten. Die einfachste Möglichkeit eine robustere Klassifikation zu erhalten, läge darin, mehr Merkmale für die Klassifikation bereitzustellen. Denkbar wären Ansätze aus der Signaltheorie oder Bildvorverarbeitung, um mehr Merkmale der Laserdaten zu erzeugen. Ebenso könnten durch neue Merkmale aus den Kameradaten mehr Klassen in der Klassifikation berücksichtigt werden. Denkbar wären da beispielsweise die Klassen Wohnzimmer, Schlafzimmer oder Badezimmer und Klassifikatoren der Objekte Bett, Couch, Waschbecken und Toilette.

Andererseits besteht auch die Möglichkeit, ganz neue Merkmale bereitzustellen. So sind beispielsweise die Merkmale der Laserdaten 2-dimensionale Entfernungsmessungen aus der Umgebung. Daher sind diese Merkmal zwar schnell berechenbar, liefern aber jeweils nur Informationen über einen Schnitt durch einen 3-dimensionalen Raum. Indoor Umgebungen sind 3-dimensionale Räume und daher wäre es vorstellbar, einen 3-dimensionalen Laserscan mit höher dimensionalen geometrischen Merkmalen zu verwenden. Dadurch wäre mehr Information über die Umgebung bekannt und die Klassifikation robuster gegen Störungen. Des Weiteren könnte man einen 3-dimensionalen Scan der Umgebung mit den Kamerainformationen kombinieren. So könnten Objekte durch Ihre 3-dimensionale Struktur erkannt werden oder es wäre die Information der Entfernung zu einzelne Objekte bekannt. So wäre beispielsweise ein Monitor in einer Entfernung von mehr als 15 Metern sehr wahrscheinlich in einem anderen Raum und wäre daher von der Klassifikation auszuschließen werden.

Ebenso wäre es denkbar noch mehr Klassifikatoren für die Objekterkennung aus den Kameradaten zu trainieren. Da sich in diesem Bereich der Objekterkennung bereits viele Arbeiten mit dem Thema beschäftigt haben, gibt es auch viele verschiedene Ansätze dazu. Da gäbe es beispielsweise die Möglichkeit konturbasierte Merkmale zu verwenden, um komplexere Objekte zu erkennen. So wären beispielsweise Stühle und Tische zu klassifizieren, die meist zu einem gewissen Teil durch andere Objekte verdeckt sind.

Eine ganz andere Möglichkeit, die Anzahl an Merkmalen auf einen Schlag drastisch zu erhöhen, liegt in der Anwendung von Radio Frequency Identification (RFID). Die RFID ist ein drahtloses Übertragungssystem zur Identifikation von Personen oder Objekten, in Form einer einmaligen seriellen Nummer. Dieses System würde beispielsweise den Barcode von Produkten in Supermärkten ablösen. Ebenso könnte diese System für unseren Ansatz verwendet werden, um festzustellen, welche Produkte sich in unmittelbarer Umgebung des Roboter befinden. Anhand dieser Objekte wäre somit in der gleichen Art und Weise auf eine Klasse zu schließen.

Ein anderes anwendungsbasiertes Beispiel wäre die Verwendung unseres Klassifikators,

um topologische Karten von Umgebungen zu erstellen. Dazu wäre eine Grip Map basierend auf den Laserdaten zu erzeugen. In dem selben Durchlauf wäre es ebenfalls möglich, verschiedenen Räume anhand von Türdurchgängen voneinander zu trennen und somit eine topologische Karte aufzubauen. Ebenso wäre es möglich innerhalb der einzelnen Räume mit unserem Klassifikator die Klasse zu bestimmen. Durch dieses Vorgehen wäre in einem Durchlauf aus einer unbekannten Umgebung eine topologische Umgebungskarte zu erzeugen, die Informationen über einzelne Räume beinhaltet.

A. Technische Daten

A.1. Der Roboter

Typ: B21r von RWI

(a) Der Roboter Albert

Parameter	Wert
Anzahl Sonarsensoren	48
Anzahl Infrarotsensoren	24
Anzahl taktile Sensoren	56
CPU	Pentium
Translationsgeschwindigkeit	max. $90\,cm/s$
Rotationsgeschwindigkeit	max. $167\,°/s$
Antrieb	4-Rad Synchro-Drive
Ladegewicht	$90\,kg$
Durchmesser	$52,5\,cm$
Höhe	$106\,cm$
Gewicht	$122,5\,kg$

(b) Die Daten

Abbildung A.1.: Der mobile Roboter Albert. Sämtliche Daten sind Angaben des Herstellers.

A.2. Der Laserscanner

Typ: PLS der Firma SICK

(a) Der Laserscanner PLS der Firma SICK

Parameter	Wert
Abmessungen	$155\,mm \times 185\,mm \times 156\,mm$
Ansprechzeit	$80\,ms$
Auflösung	$70\,mm$ (in $4\,m$ Entfernung)
Reichweite	max. $50\,m$
Öffnungswinkel	bis zu 180°
Versorgungsspannung	24 V DC
Gewicht	ca. $4.5\,kg$

(b) Die Daten

Abbildung A.2.: Der Laserscanner PLS der Firma SICK. Daten sind Angaben des Herstellers.

A.3. Die Kamera

Typ: Sony XC-999

(a) Die CCD-Kamera der Firma SONY

Parameter	Symbol	Wert
Sensorelement horizontal	n_{cx}	768
Sensorelement vertikal	n_{cy}	494
Breite des CCD-Sensors	b_s	6.4 mmm
Höhe des CCD-Sensors	h_s	4.8 mm
Brennweite der Linse	f	6.17 mm
Verzerrungsfaktor	κ_1	0,007657
hor. Bildhauptpunktverschiebung	c_x	293.621 Pixel
vert. Bildhauptpunktverschiebung	c_y	224,697 Pixel

(b) Die Daten

Abbildung A.3.: Die CCD-Kamera der Firma SONY. Daten sind Angaben des Herstellers.

Literaturverzeichnis

[Agarwal und Roth, 2002] S. Agarwal and D. Roth. Learning a sparse representation for object detection. In *Proceedings of the European Conference on Computer Vision*, volume 4, pages 113–130, Copenhagen, Denmark, May 2002. Springer-Verlag.

[Althaus und Christensen, 2003] P. Althaus and H.I. Christensen. Behaviour coordination in structured environments. *Advanced Robotics*, 17(7):657–674, 2003.

[Anguelov *et al.*, 2002] D. Anguelov, R. Biswas, D. Koller, B. Limketkai, S. Sanner, and S. Thrun. Learning hierarchical object maps of non-stationary environments with mobile robots. In *Proc. of the Conf. on Uncertainty in Artificial Intelligence (UAI)*, 2002.

[Borenstein *et al.*, 1996] J. Borenstein, B. Everett, and L. Feng. *Navigating Mobile Robots: Systems and Techniques.* A. K. Peters, Ltd., Wellesley, MA, 1996.

[Burgard *et al.*, 1998] W. Burgard, A. Derr, D. Fox, and A.B. Cremers. Integrating global position estimation and position tracking for mobile robots: The dynamic markov localization approach. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 1998.

[Buschka und Saffiotti, 2002] P. Buschka and A. Saffiotti. A virtual sensor for room detection. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 637–642, 2002.

[Cohen, 1995] Paul R. Cohen. *Empirical methods for artificial intelligence.* MIT Press, Cambridge, MA, USA, 1995.

[Cover und Thomas, 1991] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory.* John Wiley & sons, 1991.

[Cox und Wilfong, 1990] I.J. Cox and G.T. Wilfong, editors. *Autonomous Robot Vehicles.* Springer Verlag, 1990.

[Dellaert *et al.*, 1998] F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte carlo localization for mobile robots. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 1998.

[Doucet, 1998] A. Doucet. On sequential simulation-based methods for bayesian filtering. Technical report, Signal Processing Group, Departement of Engeneering, University of Cambridge, 1998.

[Fox *et al.*, 1999] D. Fox, W. Burgard, and S. Thrun. Markov localization for mobile robots in dynamic environments. *Journalof Artificial Intelligence Research*, 11, 1999.

[Freund und Schapire, 1995] Y. Freund and R.E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Computational Learning Theory (Eurocolt)*, 1995.

[Freund und Schapire, 1997] Y. Freund and R.E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[Freund und Schapire, 1999] Y. Freund and R. Schapire. A short introduction to boosting, 1999.

[Friedman *et al.*, 1998] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting, 1998.

[Garcia *et al.*, 1999] Christophe Garcia, G. Zikos, and George Tziritas. Face detection in color images using wavelet packet analysis. In *ICMCS, Vol. 1*, pages 703–708, 1999.

[Gonzalez und Wintz, 1987] R.C. Gonzalez and P. Wintz. *Digital Image Processing.* Addison-Wesley Publishing Inc., 1987.

[Haralick und Shapiro, 1992] R.M. Haralick and L.G. Shapiro. *Computer and Robot Vision.* Addison-Wesley Publishing Inc., 1992.

[Heisele *et al.*, 2001] B. Heisele, T. Serre, S. Mukherjee, and T. Poggio. Feature reduction and hierarchy of classifiers for fast object detection in video images, 2001.

[Innes *et al.*, 2004] Andrew Innes, Victor Ciesielski, John Mamutil, and Sabu John. Reducing false alarms using genetic programming in object detection. In *IC-AI*, pages 569–574, 2004.

[Koenig und Simmons, 1998] S. Koenig and R. Simmons. Xavier: A robot navigation architecture based on partially observable markov decision process models. In D. Kortenkamp, R. Bonasso, and R. Murphy, editors, *Artificial Intelligence Based Mobile Robotics: Case Studies of Successful Robot Systems*, pages 91–122. MIT-Press, 1998.

[Kuipers und Beeson, 2002] B. Kuipers and P. Beeson. Bootstrap learning for place recognition. In *Proc. of the Nat. Conf. on Artificial Intelligence (AAAI)*, 2002.

[Leonard *et al.*, 1992] J.J. Leonard, H.F. Durrant-Whyte, and I.J. Cox. Dynamic map building for an autonomous mobile robot. *International Journal of Robotics Research*, 11(4):89–96, 1992.

[Lienhart *et al.*, 2003] R. Lienhart, A. Kuranov, and V. Pisarevsky. Empirical analysis of detection cascades of boosted classifiers for rapid object detection. In *DAGM, 25th Pattern Recognition Symposium*, 2003.

[Loncaric, 1998] S. Loncaric. A survey of shape analysis techniques. *Pattern Recognition*, 31(8), 1998.

[Martínez Mozos *et al.*, 2005] O. Martínez Mozos, C. Stachniss, and W. Burgard. Supervised learning of places from range data using adaboost. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Barcelona, Spain, April 2005.

[Martínez Mozos, 2004] O. Martínez Mozos. Supervised learning of places from range data using adaboost. Master's thesis, University of Freiburg, December 2004.

[Meir und Rätsch, 2003] R. Meir and G. Rätsch. *An Introduction to Boosting and Leveraging*, pages 119–184. Number LNAI 2600. Springer, January 2003.

[Montemerlo *et al.*, 2003] M. Montemerlo, N. Roy, and S. Thrun. Perspectives on standardization in mobile robot programming. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2003.

[Murphy *et al.*, 2003] K. Murphy, A. Torralba, and W. Freeman. Using the forest to see the tree: a graphical model relating features, objects and the scenes, 2003.

[Oore *et al.*, 1997] S. Oore, G.E. Hinton, and G. Dudek. A mobile robot that learns its place. *Neural Computation*, 9(3):683–699, 1997.

[OpenCV, 2001] OpenCV. Open computer vision library. URL, 2001. http://sourceforge.net/projects/opencvlibrary.

[O'Rourke, 1998] J. O'Rourke. *Computational Geometry in C (2nd Edition)*. Cambridge University Press, 1998.

[Papageorgiou *et al.*, 1998] Constantine P. Papageorgiou, Michael Oren, and Tomaso Poggio. A general framework for object detection. In *ICCV '98: Proceedings of the Sixth International Conference on Computer Vision*, page 555, Washington, DC, USA, 1998. IEEE Computer Society.

[Rätsch *et al.*, 2002] G. Rätsch, S. Mika, B. Schölkopf, and K.-R. Müller. Constructing boosting algorithms from svms: an application to one-class classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(9):1184–1199, 2002.

[Rowley *et al.*, 1998] Henry A. Rowley, Shumeet Baluja, and Takeo Kanade. Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):23–38, 1998.

[Russ, 1992] J.C. Russ. *The Image Processing Handbook*. CRC Press, 1992.

[Russell und Norvig, 1995] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ, 1995.

[Sack und Burgard, 2004] D. Sack and W. Burgard. A comparison of methods for line extraction from range data. In *Proc. of the 5th IFAC Symposium on Intelligent Autonomous Vehicles (IAV)*, 2004.

[Schapire und Singer, 1999] Robert E. Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. *Mach. Learn.*, 37(3):297–336, 1999.

[Schapire, 2001] R. Schapire. The boosting approach to machine learning: An overview, 2001.

[Schneiderman und Kanade, 2000] H. Schneiderman and T. Kanade. A statistical method for 3d object detection applied to faces and cars. In *CVPR00*, pages I: 746–751, 2000.

[Shumeet und Rowley, 2005] B. Shumeet and H. Rowley. Boosting sex identification performance. In *Proc. of the Seventeenth Innovative Applications of Artificial Intelligence Conference (IAAI)*, Pittsburgh, PA, USA, 2005.

[Silapachote *et al.*, 2004] P. Silapachote, D. R. Karuppiah, and A. Hanson. Feature selection using adaboost for face expression recognition. In *Proceedings of the Fourth IASTED International Conference on Visualization, Imaging, and Image Processing*, pages 84–89, Marbella, Spain, September 2004.

[Simmons *et al.*, 1997] R. Simmons, R. Goodwin, K. Haigh, S. Koenig, and J. O'Sullivan. A layered architecture for office delivery robots. In *Proc. of the First Int. Conf. on Autonomous Agents*, 1997.

[Thrun *et al.*, 2000] S. Thrun, M. Beetz, M. Bennewitz, W. Burgard, A.B. Cremers, F. Dellaert, D. Fox, C. Haehnel, D. andRosenberg, N. Roy, J. Schulte, and D. Schulz. Probabilistic algorithms and the interactive museum tour-guide robot Minerva. volume 19, 2000.

[Thrun *et al.*, 2001] S. Thrun, D. Fox, W. Burgard, and Dellaert. F. Robust monte carlo localization for mobile robots. *Artificial Intelligence*, 128(1-2), 2001.

[Tieu und Viola, 2000] K. Tieu and P. Viola. Boosting image retrieval. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 228–235, 2000.

[Torralba *et al.*, 2003] A. Torralba, K. Murphy, W. Freeman, and M. Rubin. Context-based vision system for place and object recognition. In *Proc. of the Int. Conf. on Computer Vision (ICCV)*, 2003.

[Treptow *et al.*, 2003] A. Treptow, A. Masselli, and A. Zell. Real-time object tracking for soccer-robots without color information. In *Proc. of the Europ. Conf. on Mobile Robots (ECMR)*, 2003.

[Viola und Jones, 2001] P. Viola and M.J. Jones. Robust real-time object detection. In *Proc. of IEEE Workshop on Statistical and Theories of Computer Vision*, 2001.