Technische Universität Ilmenau
Fakultät für Informatik und Automatisierung
Fachgebiet Neuroinformatik und Kognitive Robotik

Klassifikation von Fahrzeugen nach Typ und Modell anhand von Videodaten

Bachelorarbeit zur Erlangung des akademischen Grades Bachelor of Science

Manuel Zellhöfer

Betreuer: Dr. Gerald Schweighofer (JOANNEUM RESEARCH, Graz)
Dipl. Inf. Ronny Stricker
Verantwortlicher Hochschullehrer:
Prof. Dr. H.-M. Groß, FG Neuroinformatik und Kognitive Robotik

Die Bachelorarbeit wurde am 12.07.2011 bei der Fakultät für Informatik und Automatisierung der Technischen Universität Ilmenau eingereicht.

Erklärung: „Hiermit versichere ich, dass ich diese Bachelorarbeit selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe. Alle von mir aus anderen Veröffentlichungen übernommenen Passagen sind als solche gekennzeichnet.“

Ilmenau, 12.07.2011 .....

Manuel Zellhöfer

Inhaltsverzeichnis

Kapitel 1

Einleitung

Automatische Klassifikation ist ein wichtiger Bestandteil kognitiver technischer Systeme. Sie ist in vielen Bereichen anzufinden, wie etwa dem maschinellen Sehen oder der Spracherkennung. Dies liegt daran, dass die Klassifikation die zentrale Komponente am Ende jedes Mustererkennungsprozesses darstellt. Klassifikation von Fahrzeugen nach Typ und Modell, *engl.:* Make and Model Recognition (MMR), ist ein spezielles Beispiel für ein Problem des maschinellen Sehens.

Geht man von beliebigen Aufnahmen von Verkehrsszenen aus in denen MMR durchgeführt werden soll, trifft man an zwei Stellen auf Klassifikationsaufgaben. Zuerst muss im Bild, das der Sensor liefert, ein Fahrzeug gefunden werden. Ein geeigneter Detektionsalgorithmus muss dabei Bildausschnitte nach *Fahrzeug / kein Fahrzeug* klassifizieren. Der nächste Schritt ist die Klassifikation des detektierten Fahrzeugs. Diese stellt den Kern der vorliegenden Arbeit dar.

Es gibt eine Vielzahl an Algorithmen die in der Lage sind Datenreihen zu klassifizieren (Farb- oder Grauwertbilder sind eine Form von Datenreihen). Von einfachen Methoden wie Nearest Neighbour Klassifikation (NN) über etablierte Techniken wie Support-Vector-Machine (SVM) oder neuronale Netze. In der vorliegenden Arbeit soll ein Klassifikationsalgorithmus basierend auf dem Prinzip von *Sparse Representations* untersucht werden – Sparse Representation Classification (SRC).

Ausgespart bleiben soll die Detektion von Fahrzeugen in den Bildern sowie die Regis-

trierung oder Ausrichtung an den Bildern in der Datenbank. Auch auf die Verwendung von Bildsequenzen, wie der Begriff *Videodaten* implizieren könnte, wird in dieser Arbeit nicht eingegangen.

Motivation

Anwendungsgebiete von MMR-Systemen finden sich überwiegend im Bereich der Sicherheitstechnik. So lässt sich die Sicherheit von auf Nummernschilderkennung basierenden Identifikationssystemen verbessern. Einen Schritt weiter könnten polizeilich gesuchte Fahrzeuge auch anhand der Fahrzeugbeschreibung automatisch identifiziert werden. Sei es als Hilfe für Polizisten in Streifenwägen oder ortsgebunden, an Schlüsselpunkten im Straßennetz.
Ein weiteres Einsatzgebiet findet sich im Sicherheitssystem von Fahrzeugen. Das Insassenrückhaltesystem besteht üblicherweise aus Gurt, automatischem Gurtstraffer und mehreren Airbags. Obwohl es zum Schutz der Insassen vorhanden ist, stellt es vor allem bei Unfällen mit niedrigen Geschwindigkeiten ein zusätzliches Verletzungsrisiko dar. Durch die adaptive Gestaltung des Insassenrückhaltesystems lässt sich dieses deutlich verringern. Ein MMR-System wird hier benötigt um bei einer bevorstehenden Kollision Informationen über den Aufprallpartner zu ermitteln. Ist Marke und Modell des Aufprallpartners bekannt, kann Rückschluss gezogen werden auf Gewicht, Geometrie und Verformungsverhalten. Anhand dieser Größen können dann Parameter des Rückhaltesystems so eingestellt werden, dass das Verletzungsrisiko verringert wird.

Aufbau der Arbeit

Im nächsten Kapitel werden existierende MMR-Verfahren vorgestellt und die vorliegende Arbeit entsprechend eingeordnet. Das darauf folgende Kapitel stellt recherchierte Methoden zur Merkmalsextraktion im Bereich der MMR vor. Danach wird das angesprochene Klassifikationsverfahren SRC vorgestellt, welches bisher nur im zur Gesichtserkennung eingesetzt wurde. Kapitel 5 erläutert Details zur Implementierung des Algorithmus, der als Grundlage für die Experimente aus Kapitel 6 dient. Kapitel 5

erläutert Details zur Implementierung mit der die in Kapitel 6 vorgestellten Experimente durchgeführt werden. Im letzten Kapitel wird eine Zusammenfassung über die Arbeit gegeben und mögliche zukünftige Entwicklungen angesprochen.

Kapitel 2

Stand der Technik

Der grundlegende Ablauf bei der Mustererkennung lässt sich in die Schritte Vorverarbeitung, Merkmalsextraktion und Klassifikation aufteilen. Die Vorverarbeitung dient der Aufbereitung der Sensordaten zur weiteren Verarbeitung. Durch die Merkmalsextraktion soll die darauffolgende Klassifikation durch die Wahl geeigneter Merkmale vereinfacht oder überhaupt erst ermöglicht werden. Die Klassifikation weist dann einer Eingabedatenreihe die entsprechende Klasse zu.

Verglichen mit Gesichtserkennung und deren Detektion oder Fahrzeug- oder generell Objektdetektion existieren verhältnismäßig wenige Veröffentlichungen, die MMR-Systeme vorstellen. Dennoch unterscheiden sich die wenigen gefundenen Verfahren grundlegend in der verwendeten Klassifikationsmethode und den Methoden zur Merkmalsextraktion. Die Klassifikationsmethoden schließen Bayes-Klassifikation, (k-)NN, Neuronale Netze und Entscheidungsbäume ein. Zur Merkmalsextraktion werden diverse Kantendetektoren und auf solchen aufbauende, verfeinerte Extraktionsverfahren vorgeschlagen.

Das folgende Kapitel soll einen Überblick über recherchierte MMR-Systeme liefern und Bezugspunkte zur untersuchten Klassifikationsmethode zeigen. Tabelle 2.1 zeigt eine Übersicht über diese Methoden mit den jeweils verwendeten Techniken und den dabei erreichten Erkennungsraten (P_{Er}). Die Arbeiten von [Dlagnekov und Belongie, 2005], [Sarfraz et al., 2009] und [Petrovic und Cootes, 2004] sollen dabei

exemplarisch näher betrachtet werden. Erstere wegen des –verglichen mit den übrigen Methoden– ungewöhnlichen Ansatzes zur Merkmalsextraktion durch Scale Invariant Feature Transform (SIFT). Die Arbeit von [Sarfraz et al., 2009] verwendet als Alleinstellungsmerkmal eine Bayes'sche Klassifikationsmethode zusammen mit einem Histogramm-basierten Merkmalsextraktor. Dahingegen verwendet die letzte vorgestellte Veröffentlichung klassische, bildbasierte[1] Merkmale, wie sie auch für die untersuchte Klassifikationsmethode geeignet sind.

Tabelle 2.1: Übersicht über Recherchierte MMR-Systeme

Veröffentlichung	**Merkmalsextr.**	**Klassifikator**	P_{Er} **in %**
[Dlagnekov und Belongie, 2005]	SIFT	Matching	89.7
[Sarfraz et al., 2009]	LESH	Bayes	94
[Clady et al., 2008]	Gradientenbasiert	Voting	90
[Petrovic und Cootes, 2004]	verschiedene	NN	97.7
[Kazemi et al., 2007]	Curvelets	SVM	99
[Zafar et al., 2009]	Contourlets (+LDA)	SVM	96
[Munroe und Madden, 2005]	Gradientenbasiert	NN	99.5

2.1 Klassifikation mit SIFT-Merkmalen

Mithilfe des SIFT-Algorithmus (Scale Invariant Feature Transform) lassen sich aus Grauwertbildern markante Punkte (Keypoints) und Umgebungsinformation (Vorzugsrichtung, Skalierung) zu diesen als Features extrahieren. Der Algorithmus basiert auf Filterung zur Detektion von Keypoint-Kandidaten und anschließender Aussortierung von wenig aussagekräftigen Kandidaten bzw. Gruppierung ähnlicher Kandidaten. Die resultierenden Features sind invariant gegenüber Translation, Rotation und Skalierung des Ursprungsbildes und haben einen hohen Wiedererkennungswert. [Lowe, 2004]

[1] Bildbasiert in dem Sinne, dass der Merkmalsvektor die Struktur eines Bildes beibehält und kein „Bag-of-Features" oder Histogramm verwendet wird

[Dlagnekov und Belongie, 2005] stellten einen Ansatz vor, der mit diesem Algorithmus gefundene Punkte in Grauwertbildern von Fahrzeugen vergleicht und diese dadurch Klassifiziert. Dieser besteht aus drei Schritten:

1. Extraktion der SIFT-Features (Keypoints) für alle Bilder in der Datenbank und das Testbild
2.
   - Vergleich aller Keypoints des Testbilds mit den Keypoints aller Bilder in der Datenbank
   - Finden von übereinstimmenden Keypoints anhand eines Abstandsmaßes
3. Das Bild aus der Datenbank mit den meisten übereinstimmenden Keypoints gibt Auskunft über die Klasse des Testbilds

Die Methode wurde zusammen mit einem Nummernschilderkennungsverfahren entwickelt. Die verwendete Datenbank bestand aus 1140 über die Position des Nummernschilds registrierten Rückansichten von Fahrzeugen in 38 Klassen. Getestet wurde lediglich mit 38 ausgewählten Rückansichten, eine je Klasse, wobei eine Erkennungsrate von 89.5% erreicht wurde [Dlagnekov, 2005].
Der Ansatz unterscheidet sich in der Herangehensweise deutlich von der in dieser Arbeit untersuchten Klassifikationsmethode. Die Klassifikation erfolgt anhand von Koordinaten und der Umgebung markanter Punkte und nicht auf der Gesamtansicht der Fahrzeuge. Dabei ist ein großer Vorteil das Wegfallen der Registrierung der Testsamples. Dieser ist mit der recht aufwändigen Merkmalsextraktion erkauft.

2.2 Klassifikation mit dem Bayes Theorem

In [Sarfraz et al., 2009] wurde eine Klassifikationsmethode basierend auf dem Bayes-Theorem vorgeschlagen.
Zur Klassifikation wird zuerst das Mehrklassen-Problem auf ein Zwei-Klassen Problem reduziert. Dies geschieht, indem zwei Wahrscheinlichkeitsverteilungen über der Ähnlichkeit zwischen zwei Samples definiert werden. Die Ähnlichkeit zweier Samples r und t, $\chi_{r,t}$ wird durch ein geeignetes Abstandsmaß ermittelt. Die beiden Verteilungen sind

- $P(\chi = \chi_{r,t}|$„Sample r ist in der selben Klasse wie t") $= P(\chi_{r,t}|C_{selbe})$ und
- $P(\chi = \chi_{r,t}|$„Sample r ist in einer anderen Klasse wie t") $= P(\chi_{r,t}|C_{andere})$.

Beide werden aus dem Trainingsdatensatz berechnet und durch Normalverteilungen (mit Mittelwert und Varianz) repräsentiert. Die Größe, die zur Klassifikation benötigt wird, ist

$$P(\text{„Sample } r \text{ ist in der selben Klasse wie } t\text{"}|\chi = \chi_{r,t}) = P(C_{selbe}|\chi_{r,t}).$$

Mithilfe des Bayes-Theorems und den obigen Verteilungen ergibt sich Gleichung 2.1 zur Berechnung eben dieser Wahrscheinlichkeit

$$P(C_{selbe}|\chi_{r,t}) = \frac{P(C_{selbe})P(\chi_{r,t}|C_{selbe})}{P(\neg C_{selbe})P(\chi_{r,t}|\neg C_{selbe}) + P(C_{selbe})P(\chi_{r,t}|C_{selbe})} \tag{2.1}$$

Es gilt $\neg C_{selbe} = C_{andere}$. Die Wahrscheinlichkeit $P(C_{selbe})$ wird vorher als $\ll 1$ festgelegt. Dementsprechend ist $P(\neg C_{selbe}) = 1 - P(C_{selbe})$. Das Trainingssample für welches $P(C_{selbe}|\chi_{r,t})$ den Höchstwert einnimmt gibt Rückschluss auf die Klasse des Testsamples.

Der verwendete Algorithmus zur Merkmalsextraktion (Local Energy based Shape Histogram (LESH)) basiert auf der lokalen Signalenergie im Bild und wird in Abschnitt 3.3 erläutert. Diese Methode wurde auf einen Datensatz von 300 Frontansichten bei unterschiedlichen Wetter- und Beleuchtungsverähltnisen angewandt. Im Datensatz waren 25 unterschiedliche Modelle vorhanden. Mit vier Samples je Klasse wurden die Wahrscheinlichkeitsverteilungen berechnet. Ein Sample je Klasse wurde als Referenz verwendet, um die Distanzen zu berechnen. Mit den übrigen sieben Samples wurde getestet. Dabei wurde eine Erkennungsrate von 94% erreicht.

Im Vergleich mit der Vorgehensweise aus dem vorherigen Abschnitt ähnelt diese eher der untersuchten Methode, da registrierten Frontansichten klassifiziert wurden. Die verwendeten Bilder stammen aus dem Datensatz der auch in der vorliegenden Arbeit genutzt wird. Näheres zu diesem Datensatz findet sich in Abschnitt 6.1.

2.3 Nearest Neighbour und Kantendetektoren

[Petrovic und Cootes, 2004] untersuchten eine Vielzahl von gradientenbasierten Kantendetektoren zur Merkmalsextraktion. Einige dieser sind in Abschnitt 3.1 näher Erläutert. Zur Klassifikation wurde der NN-Algorithmus verwendet. Der Verwendete Datensatz bestand aus 1132 Bildern von 77 unterschiedlichen Fahrzeugtypen bei variierenden Licht- und Wetterverhältnissen. 105 Bilder wurden zum Training verwendet und die übrigen 1027 zum Testen. Die Bilder wurden über die Position des Nummernschildes registriert. Die erreichte Erkennungsrate betrug 97.7% Die Anwendung der Hauptkomponentenanalyse zur Merkmalsreduktion brachte keine Verbesserung der Klassifikationsleistung. Einer der in der vorliegenden Arbeit verwendeten Datensätze wurde ursprünglich für diese Veröffentlichung aufgenommen und auch in einem Großteil der anderen Arbeiten genutzt. Eine nähere Beschreibung findet sich in Abschnitt 6.1.

Die übrigen Arbeiten in Tabelle 2.1 können als Abwandlung dieser Arbeit betrachtet werden, da die Innovationen aus der Verwendung anderer Merkmalsextraktoren bestehen. So wurden etwa Wavelet, Curvelet oder Contourlet-Transformation benutzt oder modifizierte Kantendetektoren. Als Klassifikatoren wurden überwiegend Standardverfahren (SVM, NN) verwendet.

Zusammenfassung

Die drei vorgestellten Methoden stellen einen Querschnitt der vorhanden Literatur zur MMR dar. Die untersuchte Klassifikationsmethode SRC stellt eine Neuerung dar. SRC ist in der Lage, Verfälschungen auf Pixelebene in den Testsamples direkt zu kompensieren. In [Wright et al., 2008] wurden entsprechend klare Vorteile bezüglich der Robustheit verglichen mit NN oder SVMs nachgewiesen. Die Methode wurde dabei zur Gesichtserkennung angewendet. Es wird angenommen, dass diese Robustheit auch bei der MMR vorhanden ist.

Während sich die in diesem Kapitel vorgestellte Methode die sich der SIFT bedient von

den anderen beiden abhebt, wurden letztere in der vorliegenden Arbeit insofern verarbeitet, dass die Merkmalsextraktoren (oder Teile davon im Fall von LESH) verwendet wurden. Inwiefern genau beschreibt das nächste Kapitel.

Kapitel 3

Verwendete Methoden zur Merkmalsextraktion

Bei der Klassifikation von Fahrzeugen anhand ihrer Frontansicht sind einige spezifische Schwierigkeiten zu berücksichtigen. So kann die Beleuchtung je nach Wetterlage und Tageszeit deutlich variieren. Zwei Aufnahmen des selben Fahrzeugs am Morgen und am Abend unterscheiden sich voneinander ebenso wie sich auch zwei Aufnahmen bei Regen oder bei Sonnenschein unterscheiden. Umgebende Lichtquellen werden von den glatten Oberflächen der Fahrzeuge stark reflektiert oder es spiegeln sich Teile der Umgebung auf der Motorhaube.

Ein weiteres Problem ist, dass Fahrzeuge des selben Modells mit unterschiedlichen Farben existieren. Da Farbe kein Kriterium zur Klassifikation darstellt, sondern sie im Gegensatz erschwert, werden gewichtete Grauwertbilder verwendet. Abbildung 3.1 zeigt Beispiele für Unterschiede innerhalb derselben Klasse eines Fahrzeugs.

Die Verfälschungen, die Reflektionen auf der Oberfläche oder Modifikationen der Front verursachen, sollte die verwendete Klassifikationsmethode berücksichtigen. Um die wegen der unterschiedlichen Farben vorhandene Varianz innerhalb der Klassen zu reduzieren, werden in den in Kapitel 2 aufgeführten Verfahren überwiegend Merkmalsextraktoren verwendet, die Kanten hervorheben. Die geometrischen Strukturen der Fahrzeugfront (Scheinwerfer, Kühlergrill, Stoßstange) sind bei aufmerksamer Betrach-

Abbildung 3.1: Schwierigkeiten bei der Klassifikation von Fahrzeugen
Zu sehen sind drei Fahrzeuge des selben Modells (Klasse). Sichtbar sind Unterschiede aufgrund der Beleuchtung und deutlicher, aufgrund der Farbe. Im linken Bild sieht man reflektiert die Umgebung des Fahrzeugs und deutliche Spiegelungen auf der Motorhaube. Das mittlere Bild zeigt ein Beispiel für eine Modifizierung der Front.

tung durch sichtbare Kanten begrenzt. In der vorliegenden Arbeit werden ebenfalls nur Merkmalsextraktoren verwendet, die Kanten hervorheben (siehe Tabelle 3.1). Sie lassen sich dabei in solche einteilen, die den Gradienten des Grauwertbildes berechnen und solche, die mittels geeigneter Filterung Information über die Ortsfrequenzen des Bildes gewinnen. Dieses Kapitel stellt die verwendeten Methoden vor.

Tabelle 3.1: Übersicht über verwendete Merkmale

	Merkmal
Frequenzbasiert	Gabor-Filter
	Lokale Energie
	Phasenkongruenz
Gradientenbasiert	Sobel-Filter
	Square-Mapped Gradients

3.1 Gradientenbasierte Merkmale

Eine einfache Methode Kanten in einem Bild zu detektieren besteht aus dem Ableiten des Helligkeitsverlaufs. Weitestgehend etabliert ist hier die Faltung des Bildes mit dem Sobel-Operator, wie in Abbildung 3.2 gezeigt. Dabei entstehen zwei weitere Bilder: Die Ableitung in x- und in y-Richtung.

$$S_x = \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix} S_y = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$$

Abbildung 3.2: Faltungskerne des Sobel-Operators
Diese beiden Matrizen stellen den Faltungskern des Sobel-Operators in x- bzw. y-Richtung dar. Bei der Faltung eines Grauwertbildes mit diesen Kernen enthält das resultierende Bild die geglättete Ableitung des Ursprungsbild in x- bzw. y-Richtung.

Basierend auf den resultierenden Gradientenbildern, im folgenden G_x bzw. G_y genannt, können verschiedene weitere Merkmale gewonnen werden. Beispiele hierfür sind die Orientierung der Kanten, über das gesamte Bild normierte Gradienten oder lokal normierte Gradienten. [Petrovic und Cootes, 2004]
Als effektiv für MMR hat sich die Verwendung von *Square-Mapped Gradients* erwiesen. Die Gradienten werden auf Einheitslänge gebracht und so transformiert, dass die Orientierung auf Modulo π eingeschränkt wird. Die Transformation besteht darin, die Gradienten in Polarkoordinaten (Betrag G und Winkel ϕ, siehe (3.1)) darzustellen und den Winkel zu verdoppeln. Dadurch werden Winkel $\phi \in [\pi, 2\pi]$ in den Bereich $[0, \pi]$ abgebildet. Durch Anwendung der Additionstheoreme findet sich Gleichung 3.2.
Dabei stellt $G_{SM,x}$ den achsen-parallelen und $G_{SM,y}$ den -diagonalen Anteil dar. Die Repräsentation wird invariant gegenüber der Richtung des Gradienten (dunkel zu hell oder hell zu dunkel), nicht aber gegenüber der Orientierung. Abbildung 3.3 veranschaulicht dies anhand einiger beispielhafter Grauwertverläufe. [Cootes und Taylor,

2001]

$$G_x = G\cos(\phi) \qquad G_y = G\sin(\phi) \tag{3.1}$$

$$\begin{aligned} G_{SM,x} &= \cos(2\phi) & G_{SM,y} &= \sin(2\phi) \\ &= \cos(\phi)\cos(\phi) - \sin(\phi)\sin(\phi) & &= \cos(\phi)\sin(\phi) + \sin(\phi)\cos(\phi) \\ &= \frac{G_x^2 - G_y^2}{G^2} & &= \frac{2 \cdot G_x \cdot G_y}{G^2} \\ G_{SM,x} &= \frac{G_x^2 - G_y^2}{G_x^2 + G_x^2} & G_{SM,y} &= \frac{2 \cdot G_x \cdot G_y}{G_x^2 + G_x^2} \end{aligned} \tag{3.2}$$

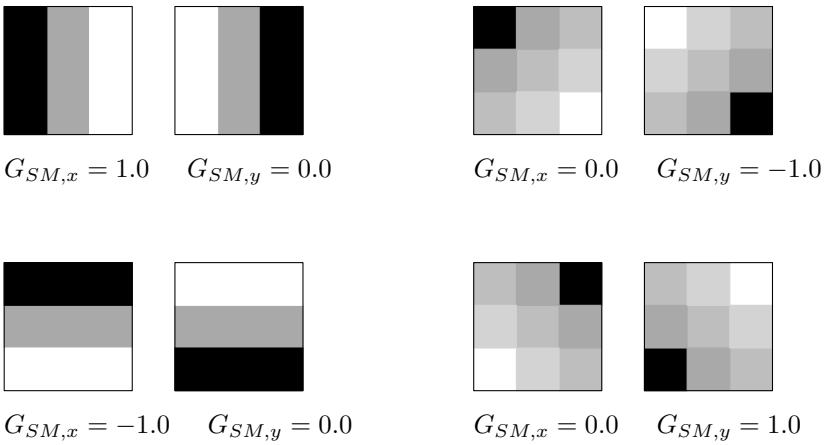


Abbildung 3.3: Square-Mapped Gradient für verschiedene Grauwertverläufe
Das Bild zeigt Werte des Square-Mapped Gradienten für den mittleren Bildpunkt in jeweils zwei Grauwertbildern. Die Werte sind invariant gegenüber der Richtung des Gradienten (dunkel zu hell oder hell zu dunkel), berücksichtigen aber die Orientierung.

Durch diese Transformation wird der Einfluss der farblichen Varianz verringert, da sich diese in Grauwertbildern nur durch die Richtung des Helligkeitsgradienten ausdrückt und nicht durch die Orientierung. [Petrovic und Cootes, 2004] In [Clady et al., 2008] wird das gleiche Prinzip *Oriented Contour Points* genannt und ebenfalls erfolgreich zur MMR verwendet.

3.2 Wavelets

Wavelets bilden ein viefältiges Werkzeug zur Signalanalyse und -verarbeitung. Wavelet ist ein Überbegriff für eine Art von Signalen die sich als Basis für die Wavelet-Transformation verwenden lassen. Diese wird unter anderem zur Datenkompression, Entrauschung und Frequenzanalyse verwendet. Ein Wavelet besteht aus einer örtlich begrenzten Schwingung. Durch Verschiebung und Skalierung wird daraus eine Basis zur Transformation erzeugt.
Bei der Wavelet-Transformation wird Information über die Frequenz eines Signals lokalisierbar im Zeitbereich gewonnen[1], weshalb sie sich für die Kantendetektion eignet: Einzelne Wavelets lassen sich als Faltungskern verwenden und stellen dann einen Bandpassfilter dar. [Lee, 1996] [Movellan, 2002]

3.2.1 Wavelets als Faltungskern: Der Gabor-Filter

Der Gabor-Filter ist eng verbunden mit der Wavelet-Transformation. Das Gabor-Wavelet besteht aus einem komplexen Träger, eingehüllt von einer Gaußglocke. Bei der Erweiterung dieses Wavelets auf zwei Dimensionen wird es zusätzlich noch rotiert (siehe Abbildung 3.4). Gleichung 3.3 zeigt die mathematische Beschreibung des 2D-Gabor-Wavelets.

$$
\begin{aligned}
g_{\omega,\varphi}(x,y) &= \tfrac{1}{\sqrt{2\pi}\sigma} \underbrace{e^{\frac{1}{2\sigma^2}(x'^2+y'^2)}}_{\text{Gaußglocke}} \underbrace{e^{j2\pi\omega(x'+y')}}_{\text{Schwingung}} \\
\text{wobei} \\
x' &= x\cos(\varphi) + y\sin(\varphi) \\
y' &= -x\sin(\varphi) + y\cos(\varphi)
\end{aligned}
\tag{3.3}
$$

Bei der Faltung mit einem Gabor-Wavelet treten Kanten hervor die die selbe Orientierung besitzen wie das jeweilige Wavelet selbst. Zur Kantendetektion können die einzelnen Bilder summiert werden (siehe Abbildung 3.5). Ein weiteres Indiz für Kanten

[1] Anders als bei der Fourier-Transformation, bei der nur Information über die Frequenzen eines Signals gewonnen wird, nicht jedoch wann oder wo diese vorkommen.

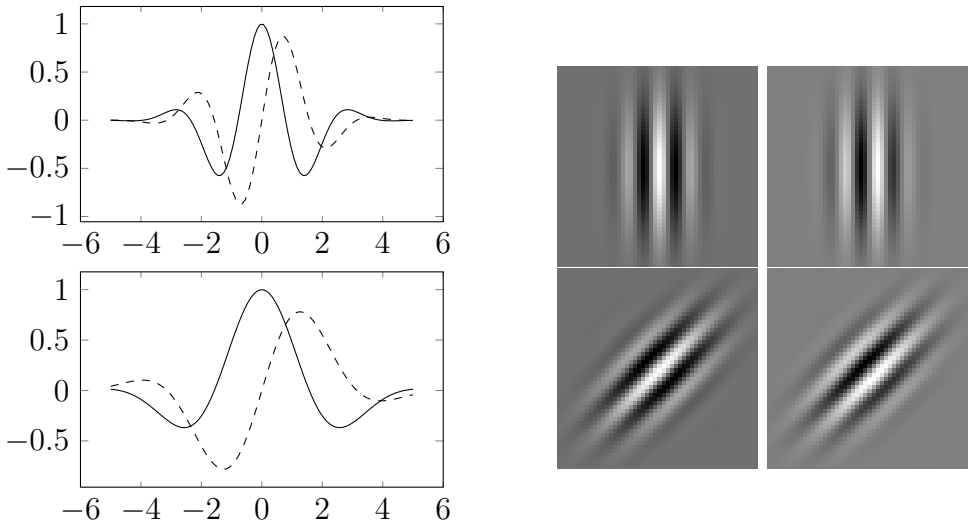


Abbildung 3.4: 1D und 2D Gabor-Wavelets

Links: *Realteil (—) und Imaginärteil (- -) des 1D-Gabor-Wavelets für zwei Skalierungen.* ***Rechts:*** *Real- und Imaginärteil des 2D-Gabor-Wavelets für zwei Rotationen. Gabor-Wavelets unterschiedlicher Rotation und Skalierung stellen Faltungskerne für die Gabor-Filterung dar. Die jeweils benutzte Menge von Wavelets wird Filterbank genannt.*

in einem Bild ist der Betrag der komplexen Filterantwort für mehrere Skalierungen. Diese Größe ist ein Maß für die im Bild vorhandene Signalenergie und wird in Abschnitt 3.3 erläutert. [Lee, 1996] [Kyrki, 2002]

3.2.2 Wavelets zur Dekomposition

Die Wavelet-Transformation ermöglicht bestimmte Signale effizienter darzustellen und eignet sich dadurch wie die Fouriertransformation zur Datenkompression. Besonders Signale in denen Unstetigkeiten vorkommen lassen sich durch Wavelets besser darstellen als durch Sinusoide. Eine spärliche Darstellung kann Vorteilhaft für die Klassifikation sein, weshalb die Wavelet-Transformation auch direkt zur Merkmalsextraktion eingesetzt wird. Auch Modifikationen der 2D-Wavelet-Transformation fanden in der MMR Anwendung:

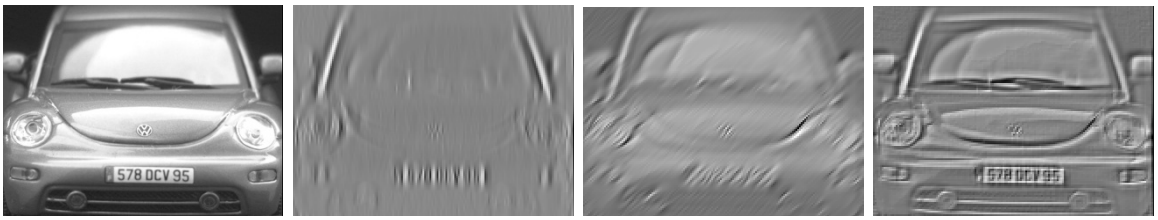


Abbildung 3.5: Bilder eines Fahrzeugs nach Gabor-Filterung
Von Links nach Rechts: *Originalbild, Bild nach Filterung mit vertikal orientierten Gabor-Wavelets (90°), Bild nach Filterung mit diagonal orientierten Gabor-Wavelets (45°), Überlagerung von Filterantworten in drei Orientierungen (0°, 45° und 90°).*

- Die *Contourlet-Transformation* eignet sich für Bilder in denen scharfe Konturen vorkommen. [Zafar et al., 2009]

- Die *Curvelet-Transformation* erweitert die Contourlet-Transformation auf weiche Kanten. [Kazemi et al., 2007]

3.3 Lokale Energie und Phasenkongruenz

Die Filterantwort eines Bildes auf den komplexen Gaborfilter lässt sich zur Berechnung der lokalen Energie und der lokalen Phase heranziehen. Diese sind, analog zur Fourieranalyse, Betrag und Phase der komplexen Filterantwort. Um mehrere Raumrichtungen und Frequenzen zu berücksichtigen, kann vereinfacht ein Filter aus einer Filterbank akkumuliert werden:

$$g(x,y) = \sum_{\varphi,\omega} g_{\omega,\varphi}(x,y) \tag{3.4}$$

Dabei werden meist etwa sechs bis acht Richtungen verwendet ($\varphi \in \{0, \pi/4, \pi/2, \pi \dots\}$). Die Frequenzen ω werden anhand der benötigten Auflösung festgelegt. Damit ergeben sich für Filterantwort G, lokale Signalenergie E und lokale Phase ϕ eines Grauwertbildes $I(x,y)$ die Gleichungen (3.5) bis (3.7).

$$G(x,y) = g(x,y) * I(x,y) \tag{3.5}$$

$$E(x,y) = |G(x,y)| = \sqrt{Re\{G(x,y)\}^2 + Im\{G(x,y)\}^2} \tag{3.6}$$

$$\phi(x,y) = \tan^{-1}\left(\frac{Re\{G(x,y)\}}{Im\{G(x,y)\}}\right) \tag{3.7}$$

Stellen im Bild, an denen die Phasen der Frequenzanteile stark übereinstimmen, werden als deutliches Merkmal oder Unstetigkeit wahrgenommen. Dies lässt sich anhand der Synthese einer Kante durch Sinusschwingungen, wie in Abbildung 3.6 gezeigt, verdeutlichen.

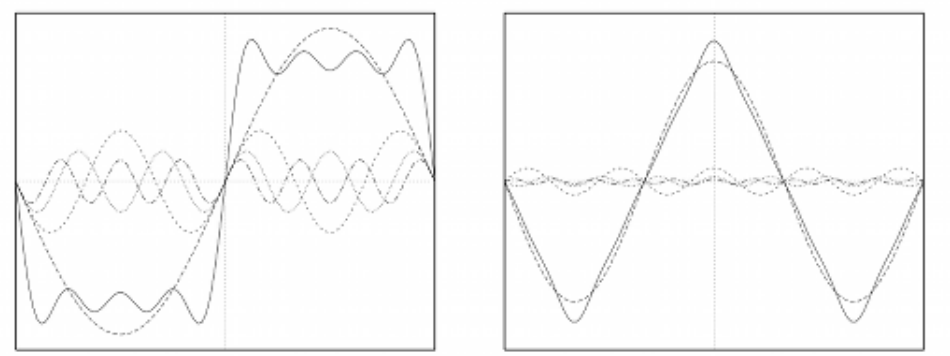


Abbildung 3.6: Synthese von Unstetigkeiten durch Sinusschwingungen
Das Bild zeigt wie eine Kante und ein Dreieckssignal durch Überlagerung von Sinusschwingungen gebildet wird. An Unstetigkeiten stimmen die Phasen der Schwingungen jeweils überein. Hohe Phasenkongruenz ist also ein Anzeichen für Unstetigkeiten. (Aus [Kovesi, 1999]*)*

Der Betrag der komplexen Filterantwort (3.6) ist proportional zur Phasenkongruenz: Dort wo die Phasen stark übereinstimmen ist die Signalenergie hoch. Deshalb eignet sich $|G(x,y)|$ zur Kantendetektion. Die Energie ist abhängig von der Signalamplitude, welche mit der Farbe schwankt. Folglich ist die Signalenergie empfindlich gegenüber farblicher Varianz. Wird die Energie mit der Summe der Amplituden des Frequenzspektrums normiert, ensteht eine dimensionslose Größe, die den Grad der Übereinstimmung der Phasen, die sogenannte Phasenkongruenz angibt. Da die Amplituden

frequenz- und richtungsabhängig sind, kann die Vereinfachung (3.4) nicht angewendet werden. Für den Wert der Phasenkongruenz ergibt sich folgende Formel:

$$P(x,y) = \frac{E(x,y)}{\sum_{\omega,\varphi} A_{\omega,\varphi}} = \frac{E(x,y)}{\sum_{\omega,\varphi} |g_{\omega,\varphi}(x,y) * I(x,y)|} \tag{3.8}$$

Um die Rauschempfindlichkeit zu reduzieren und die Auflösung zu erhöhen werden in [Kovesi, 1999] noch ein Gewichtungsfaktor $W(x,y)$ und zwei Schwellwerte T und ε eingeführt:

$$P(x,y) = \frac{W(x,y) \cdot \max(E(x,y) - T, 0)}{\sum_{\omega,\varphi} A_{\omega,\varphi} + \varepsilon} \tag{3.9}$$

Der Term $\max(E(x,y) - T, 0)$ entspricht der Entrauschung durch Wavelet-Transformation mit Schwellwert. Der Wert ε verhindert Instabilitäten falls die Amplituden des Frequenzsspektrums sehr niedrig sind. Die Funktion $W(x,y)$ gewichtet die Bildpunkte nach der Anzahl der auftretenden Frequenzen. Treten nur wenige Frequenzen auf, verliert die Phasenkongruenz an Aussagekraft. Abbildung 3.7 zeigt Lokale Energie und Phasenkongruenz für die Frontansicht eines Fahrzeuges. [Kovesi, 1999]

Abbildung 3.7: Lokale Energie und Phasenkongruenz
Zwei weitere Beispiele für die Merkmalsextraktion per Frequenzinformation. ***Links:*** *Originalbild* ***Mitte:*** *Lokale Energie des Originalbildes* ***Rechts:*** *Phasenkongruenz*

In [Sarfraz et al., 2009] wird aus der Lokalen Energie ein Histogramm gewonnen. Hierfür wird das Bild in gleichförmige Unterregionen aufgeteilt und für jede Region ein Histogramm erstellt. Um die Nachbarschaft unter den Regionen zu erhalten, werden die einzelnen Histogramme in gleicher Reihenfolge zu einem Merkmalsvektor konkateniert.

Zusammenfassung

Dieses Kapitel stellt drei frequenzbasierte und zwei gradientenbasierte Merkmale zur MMR vor: Sobel-Filter und Square-Mapped-Gradients sowie Gabor-Filter, Lokale Energie und Phasenkongruenz. Alle fünf werden in Abschnitt B.1 als Merkmalsextraktoren zur Klassifikation mit NN, SVM und SRC untersucht.
Dimensionsreduktion auf Basis der Hauptkomponentenanalyse soll dabei nicht zur Anwendung kommen. Zum einen hat diese in [Petrovic und Cootes, 2004] keinen Vorteil bei der Klassifikation gebracht und zum anderen ist die Klassifikation mit SRC unempfindlich gegenüber der Methode zur Dimensionsreduktion wie in [Wright et al., 2008] gezeigt und in Abschnitt 4.2.3 erläutert.
Nach der Merkmalsextraktion ist der nächste Schritt zur MMR die Klassifikation. Das nächste Kapitel stellt demzufolge die untersuchte Methode, SRC vor.

Kapitel 4

Sparse Representation Classification

Spärlichkeit oder Sparsamkeit spielt in der automatischen Klassifikation eine wichtige Rolle. Bei der Merkmalsextraktion etwa wird häufig eine niedrigdimensionale Repräsentation der Eingabendatenreihe gesucht, die dafür einen Großteil der relevanten Information enthaltält. Die unverarbeiteten Daten sind dann meist nicht aussagekräftig genug oder erschweren die Modellbildung durch zu viel redundante Information.

Ein anderes Beispiel für das Ausnutzen von Spärlichkeit bietet die SVM. Ihr Trick ist die Reduktion der Menge an vorhandenen Trainingsdatenreihen (Vektoren) auf die Menge der *Support-Vektoren* für das jeweilige Klassifikationsproblem. Die Support-Vektoren sind die Vektoren des Trainingssatzes, die im Merkmalsraum nahe der Klassengrenze liegen und definieren die Entscheidungsfunktion, die einer unbekannten Datenreihe die Klasse zuweist. Die Wahl einer kompakten Entscheidungsbasis macht die Klassifikation robuster und effizienter.

Eine andere Form von Spärlichkeit wird bei der SRC ausgenutzt. Sie hat ihren Ursprung in einer jungen Theorie aus der Signalverarbeitung: Dem *Compressed Sensing*[1]. Sie macht sich den Umstand zunutze, dass bei Mehrklassen-Problemen ein Sample oft durch eine Linearkombination nur weniger Trainingssamples dargestellt werden kann. Die Grundidee des *Compressed Sensing* soll im ersten Teil dieses Kapitels vorgestellt

[1] In der Literatur auch häufig *Compressed Sampling*, *Compressive Sensing* oder *Sampling* oder *Sparse Representation* oder *Reconstruction* genannt.

werden. Der zweite Teil schildert die Anwendung des Compressed Sensing zur Klassifikation (SRC). Algorithmen die dabei eine wichtige Rolle Spielen werden im letzten Teil erläutert: l_1-Minimierer.

4.1 Compressed Sensing

Die Theorie hinter Compressed Sensing beruht auf der Tatsache, dass die meisten Signale in bestimmten Basissystemen[2] eine spärliche Darstellung besitzen. Bilder können beispielsweise durch die Wavelet-Transformation kompakt dargestellt werden. Audiosignale werden mithilfe der Cosinus-Transformation komprimiert.

4.1.1 Transformationskodierung

Spärlich bedeutet, dass nur wenige Koeffizienten des transformierten Signals signifikant ungleich Null sind. Sei beispielsweise $\underline{\boldsymbol{x}} \in \mathbb{R}^N$ eine Darstellung, die durch eine Transformation $\boldsymbol{\Phi} \in \mathbb{R}^{N \times N}$ aus einem Signal $\underline{\boldsymbol{s}} \in \mathbb{R}^N$ hervorgegangen ist:

$$\underline{\boldsymbol{x}} = \boldsymbol{\Phi}\underline{\boldsymbol{s}} = \sum_i^N s_i \phi_i \tag{4.1}$$

Die Matrix $\boldsymbol{\Phi}$ kann je nach Anwendung beispielsweise eine DCT- oder Wavelet-Matrix sein, mit den Zeilenvektoren ϕ_i in Form von Sinusschwingungen oder Wavelets. Wenn die l_0-Norm von $\underline{\boldsymbol{x}}$, also die Anzahl an Komponenten des Vektors ungleich Null,

$$||\underline{\boldsymbol{x}}||_0 = |supp(\underline{\boldsymbol{x}})| = |\{j : x_j \neq 0\}|,$$

viel kleiner ist als N, ist $\underline{\boldsymbol{s}}$ über die Transformation durch $\boldsymbol{\Phi}$ komprimierbar. Der Vektor $\underline{\boldsymbol{x}}$ heißt dann *dünn besetzt* oder *spärlich*. In der Praxis kommt es kaum vor, dass die Transformierte eines Signals direkt spärlich im obigen Sinne ist. In der Regel werden Koeffizienten die einen Schwellwert unterschreiten gleich Null gesetzt oder nur die größten Koeffizienten behalten. Da in diesem Fall Information verloren geht, spricht man von *verlustbehafteter Kompression*.

[2]Gemeint sind nicht nur Basen im mathematischen Sinn sondern auch Wörterbücher aus denen sich ein Signal zusammensetzen lässt.

Die Methode der Datenkompression durch Transformationskodierung besteht im wesentlichen aus

- der Aufnahme des kompletten Signals ($\underline{\boldsymbol{s}}$),
- der Transformation in eine kompakte Darstellung ($\boldsymbol{\Phi}\underline{\boldsymbol{s}}$) und
- der Kodierung der relevanten Information

Große Datenmengen, die bei der Aufnahme des kompletten Signals entstehen, werden letztendlich verworfen.

4.1.2 Signale komprimiert erfassen durch Minimierung der l_1-Norm

Die Frage stellt sich, ob es möglich ist, sich die Aufnahme des kompletten Signals zu ersparen und nur Datenmengen in der Größenordnung des nachher komprimierten Signals aufzunehmen. Die Theorie des Compressive Sensing bejaht diese Frage und nennt eine Methode mit der sich stark unterabgetastete Signale rekonstruieren lassen. Angenommen obiger, spärlich besetzter Vektor $\underline{\boldsymbol{x}}$ wird durch eine Messung von M Werten in Form des Vektors $\underline{\boldsymbol{y}} \in \mathbb{R}^M$, $M \ll N$ wie folgt dargestellt:

$$\underline{\boldsymbol{y}} = \mathbf{A}\underline{\boldsymbol{x}} \tag{4.2}$$

Der Vektor $\underline{\boldsymbol{x}}$ ist unbekannt. Das Signal $\underline{\boldsymbol{s}} = \boldsymbol{\Phi}^{-1}\underline{\boldsymbol{x}}$ gilt es zu rekonstruieren. Die Matrix $\mathbf{A}$ (in der Literatur *Sensing Matrix*, *Measurement Matrix* oder kurz *CS-Matrix* genannt) hat mehr Zeilen als Spalten, das Gleichungssystem (4.2) ist also unbestimmt und hat unendlich viele Lösungen. Mit dem Vorwissen, dass der Vektor $\underline{\boldsymbol{x}}$ spärlich besetzt ist, kann dieser durch folgendes Minimierungsproblem bestimmt werden:

$$\arg\min_{\underline{\boldsymbol{x}}} ||\underline{\boldsymbol{x}}||_0 : \underline{\boldsymbol{y}} = \mathbf{A}\underline{\boldsymbol{x}} \tag{4.3}$$

Die Lösung dieses Problems ist NP-Schwer. Sie würde darin bestehen, in allen Kombinationen von gültigen Koeffizientenbelegungen eine passende Lösung mit minimaler l_0-Norm zu suchen.

Ersetzt man die l_0-Norm mit der l_1-Norm,

$$||\underline{\boldsymbol{x}}||_1 = \sum_j |x_j|,$$

erhält man ein lineares Optimierungsproblem:

$$\arg\min_{\underline{\boldsymbol{x}}} \sum_j |x_j| : \underline{\boldsymbol{y}} = \mathbf{A}\underline{\boldsymbol{x}} \tag{4.4}$$

Die Lösung dieses Problems wird l_1-Minimierung genannt. Ist $||\underline{\boldsymbol{x}}||_0$ klein genug, so lautet eine wichtige Erkenntnis die *Compressive Sensing* ermöglicht, dann stimmt der Lösungsvektor des Problems (4.4) mit dem von Problem (4.3) überein. Ist also $\underline{\boldsymbol{x}}$ „spärlich genug" besetzt, lässt sich anstelle des NP-Schweren Problems (4.3) äquivalent Problem (4.4) lösen. Die spärliche Darstellung lässt sich bei Kenntnis nur eines Bruchteils des zugrundeliegenden Signals $\underline{\boldsymbol{s}}$ rekonstruieren. [Candes et al., 2006]
Häufiger wird eine abgeschwächte Variante von (4.4) verwendet, die Ungenauigkeiten in der Messung berücksichtigt:

$$\arg\min_{\underline{\boldsymbol{x}}} \sum_j |x_j| : ||\underline{\boldsymbol{y}} - \mathbf{A}\underline{\boldsymbol{x}}||_2 \leq \varepsilon \tag{4.5}$$

Wobei $||\cdot||_2$ die Euklidische Norm ist und ε die Fehlertoleranz.

4.1.3 Vorraussetzungen

Nicht für jede beliebige Matrix $\mathbf{A}$ stimmen (4.3) und (4.4) überein. Sie muß gewisse Eigenschaften besitzen die aber allesamt sehr theoretischer Natur sind und für gegebene Matrizen teils nur schwer zu überprüfen sind. Praktisch sind „fast alle Matrizen CS-Matrizen" ([Donoho, 2006]). Für den, im vorigen Abschnitt betrachteten, Fall lässt sich konkret eine mögliche Matrix $\mathbf{A}$ konstruieren:

$$\mathbf{A} = \mathbf{\Psi}\mathbf{\Phi}^{-1} \tag{4.6}$$

Dabei dient $\mathbf{\Psi}$ der zufälligen Auswahl von Basisvektoren aus $\mathbf{\Phi}$ und sollte möglichst inkohärent sein. Betrachtet man das Problem (4.4) isoliert, ist dies bereits ein Sonderfall:

Laut [Candes et al., 2006] genügt es, wenn $\mathbf{A}$ aus normalverteilten Zufallsvektoren besteht damit eine Rekonstruktion des Vektors $\underline{\boldsymbol{x}}$ möglich ist.

Die Anzahl K an Elementen ungleich Null in $\underline{\boldsymbol{x}}$ ($K = ||\underline{\boldsymbol{x}}||_0$) spielt dabei ebenfalls eine gewisse Rolle. Für zufällige Matrizen $\mathbf{A}$ existiert etwa Ungleichung (4.7), die K mit der Signallänge N und der Länge der Messung $\underline{\boldsymbol{y}}$, M in Beziehung setzt.

$$2K \log(N/M) \leq M \tag{4.7}$$

Diese Ungleichung ist nicht als scharfe Grenze zu sehen. Vielmehr stellt sie eine Schwelle dar, ab der die Äquivalenz von (4.3) und (4.4) mit „überwältigender Wahrscheinlichkeit" ([Donoho, 2006]) vorliegt. Im Fall (4.6) verschiebt sich diese Schwelle zugunsten der Anzahl Messungen M, weniger Messungen werden nötig oder weniger dünn besetzte $\underline{\boldsymbol{x}}$ möglich. [Fornasier und Rauhut, 2010] [Baraniuk et al., 2011]

4.2 Sparse Representations zur Klassifikation

Zur Klassifikation lässt sich die l_1-Minimierung nutzen, wenn sich zu klassifizierende Samples als Linearkombination von Trainingssamples darstellen lassen. Bei bildbasierter Objekterkennung ist dies im Allgemeinen der Fall. Desweiteren sollten genügend Klassen existieren, sodass die Anzahl an Trainingssamples einer Klasse niedrig ist gegenüber der Gesamtanzahl an Trainingssamples.

SRC wurde in [Wright et al., 2008] erstmals zur Gesichtserkennung angewandt. Dort wurde die Tatsache ausgenutzt, dass Gesichter des selben Individuums bei unterschiedlicher Beleuchtung und wechselnder Mimik einen Unterraum im Merkmalsraum aufspannen. Analog wird dies hier für Frontansichten von Fahrzeugen bei unterschiedlichen Beleuchtungsverhältnissen angenommen. Verdeckungen und Modifikationen (wie in Abbildung 3.1 gezeigt) sollen durch eine ebenfalls in [Wright et al., 2008] vorgestellte und in Abschnitt 4.2.2 erläuterte Erweiterung berücksichtigt werden können.

4.2.1 Die Grundidee

Die Grundidee der SRC ist schnell erklärt. Sei der Ausgangspunkt ein Mehrklassen-Problem mit K Trainingssamples je Klasse und C Klassen. Die Trainingssamples, also die Grauwertbilder einer Klasse $i \in \{1, \ldots, C\}$ werden durch zeilenweises Auslesen der Bildpunkte als Spaltenvektoren $\underline{\boldsymbol{a}}_{i,1}, \underline{\boldsymbol{a}}_{i,2}, \ldots, \underline{\boldsymbol{a}}_{i,K} \in \mathbb{R}^M$ dargestellt. Es ist Vorraussetzung, dass sich ein Testsample $\underline{\boldsymbol{y}} \in \mathbb{R}^M$ als Linearkombination dieser Trainingssamples darstellen lässt:

$$\underline{\boldsymbol{y}} = \sum_{j=1}^{K} \beta_{i,j} \underline{\boldsymbol{a}}_{i,j}$$

oder in Matrix-Vektor schreibweise, wobei $\mathbf{A}_i \in \mathbb{R}^{M \times K}$ aus den Trainingssamples der Klasse i besteht :

$$\underline{\boldsymbol{y}} = \left[\underline{\boldsymbol{a}}_{i,1} \underline{\boldsymbol{a}}_{i,2} \cdots \underline{\boldsymbol{a}}_{i,K}\right] \underline{\boldsymbol{\beta}}_i = \mathbf{A}_i \underline{\boldsymbol{\beta}}_i \qquad \underline{\boldsymbol{\beta}}_i \in \mathbb{R}^K$$

Alle Matrizen $\mathbf{A}_i$ werden zu einer großen Matrix $\mathbf{A} \in \mathbb{R}^{M \times KC}$ konkateniert, dann lässt sich das Testsample $\underline{\boldsymbol{y}}$ folgendermaßen darstellen:

$$\underline{\boldsymbol{y}} = [\mathbf{A}_1, \mathbf{A}_2, \cdots, \mathbf{A}_C] \underline{\boldsymbol{x}} = \mathbf{A} \underline{\boldsymbol{x}} \qquad \underline{\boldsymbol{x}} \in \mathbb{R}^{KC} \tag{4.8}$$

Wobei der Vektor $\underline{\boldsymbol{x}}$ genau K Einträge verschieden von Null hat:

$$\underline{\boldsymbol{x}} = [0, \ldots, 0, \underline{\boldsymbol{\beta}}_i^T, 0, \ldots, 0]^T$$

für ein Testsample $\underline{\boldsymbol{y}}$ aus der Klasse i. In der Regel ist dieser also dünn besetzt.
Um ein Testsample zu klassifizieren wird das Problem (4.5) mit dem GLS (4.8) gelöst:

$$\arg\min_{\underline{\boldsymbol{x}}} \sum_l |x_l| : ||\underline{\boldsymbol{y}} - \mathbf{A}\underline{\boldsymbol{x}}||_2 \leq \varepsilon \tag{4.9}$$

Laut [Wright et al., 2008] ist dabei eine „lose obere Schranke" für K

$$K < \lfloor (M+1)/3 \rfloor \tag{4.10}$$

Die Lösung $\underline{\boldsymbol{x}}'$ dieses Problems erlaubt dann, Rückschluss auf die Klasse von $\underline{\boldsymbol{y}}$ zu nehmen (siehe Abb. 4.2).

Hierzu wird eine Funktion $\delta_i(\underline{\boldsymbol{x}}) : \mathbb{R}^{KC} \rightarrow \mathbb{R}^{KC}$ definiert, die die Elemente des Vektors $\underline{\boldsymbol{x}}$ zu Null setzt, welche nicht zur Klasse i gehören. Auf diese Weise lässt sich das Testsample $\underline{\boldsymbol{y}}$ genähert darstellen als $\underline{\boldsymbol{y}} \approx \mathbf{A}\delta_i(\underline{\boldsymbol{x}}')$. Die Klasse i, mit der diese Approximation den geringsten Fehler aufweist sollte der Klasse des Testsamples entsprechen. Der Pseudocode in Abb. 4.1 fasst den Klassifikationsprozess zusammen. [Wright et al., 2008]

Eingaben

1 $\mathbf{A} = [\mathbf{A}_1, \mathbf{A}_2, \cdots, \mathbf{A}_C] \in \mathbb{R}^{M \times KC}$ *// Matrix mit KC Trainingssamples für C Klassen*

2 $\underline{\boldsymbol{y}} \in \mathbb{R}^M$ *// Testsample*

3 $\varepsilon > 0$ *// Fehlertoleranz*

Algorithmus

4 $\underline{\boldsymbol{x}}' = \arg\min_{\underline{\boldsymbol{x}}} ||\underline{\boldsymbol{x}}||_1 : ||\mathbf{A}\underline{\boldsymbol{x}} - \underline{\boldsymbol{y}}||_2 < \varepsilon$; *// l1-Minimierungsproblem (4.5) lösen*

5 $r_i(\underline{\boldsymbol{y}}) = ||\underline{\boldsymbol{y}} - \mathbf{A}\delta_i(\underline{\boldsymbol{x}}')||_2$ `for` $i = 1, \ldots, C$; *// Fehler je Klasse berechnen*

Rückgabe

6 $id(\underline{\boldsymbol{x}}') = \arg\min_i r_i(\underline{\boldsymbol{y}})$ *// Klasse des Testsamples*

Abbildung 4.1: Pseudocode Sparse Representation Classification
Algorithmus zur Klassifikation mit SRC durch (4.5)

4.2.2 Robustheit gegenüber Verfälschungen

Mit dem Algorithmus in Abbildung 4.1 ist man in der Lage, durch l_1-Minimierung zu klassifizieren. Die angesprochene Robustheit gegenüber Verfälschungen der Testsamples auf Pixelebene wird durch eine Erweiterung von (4.2) mit einem Fehlerterm $\underline{\boldsymbol{e}} \in \mathbb{R}^M$ erreicht:

$$\underline{\boldsymbol{y}} = \underline{\boldsymbol{y}}_0 + \underline{\boldsymbol{e}} = \mathbf{A}\underline{\boldsymbol{x}} + \underline{\boldsymbol{e}} \tag{4.11}$$

Der Vektor $\underline{\boldsymbol{e}}$, der die selbe Dimension wie $\underline{\boldsymbol{y}}$ hat stellt genau die Fehler auf Pixelebene im Testsample $\underline{\boldsymbol{y}}$ dar. Es wird angenommen, dass nur wenige Pixel des Testsamples ver-

fälscht sind, weshalb der Fehler wie der Lösungsvektor $\underline{\boldsymbol{x}}$ als dünnbesetzt anzunehmen ist. Gleichung (4.11) lässt sich wie folgt umschreiben:

$$\underline{\boldsymbol{y}} = [A, I] \begin{bmatrix} \underline{\boldsymbol{y}}_0 \\ \underline{\boldsymbol{e}} \end{bmatrix} := \mathbf{B}\underline{\boldsymbol{w}} \tag{4.12}$$

Wobei I die Einheitsmatrix ist. Durch diese einfache Erweiterung bleibt das Minimierungsproblem (4.5) an sich das gleiche, lediglich die Matrix $\mathbf{A}$ wird durch $\mathbf{B}$ ausgetauscht und der Lösungsvektor $\underline{\boldsymbol{x}}$ durch die Konkatenation $\underline{\boldsymbol{w}}$ von Lösung und Fehlerterm. Über (4.10) und der Annahme, dass $M \gg K$, lässt sich laut [Wright et al., 2008] eine grobe obere Schranke für den Anteil $|supp(\underline{\boldsymbol{e}})|/M$ an verdeckten oder verfälschten Pixeln zu

$$|supp(\underline{\boldsymbol{e}})|/M < 1/3 = 33\%$$

angeben. In Abbildung 4.2 ist das Gleichungssystem (4.11) nach der Lösung des Optimierungsproblems bildhaft dargestellt. Abbildung 4.3 zeigt weitere Beispiele für fehlerbehaftete Testamples mit dem vom Minimierer ermittelten Fehlerbild.

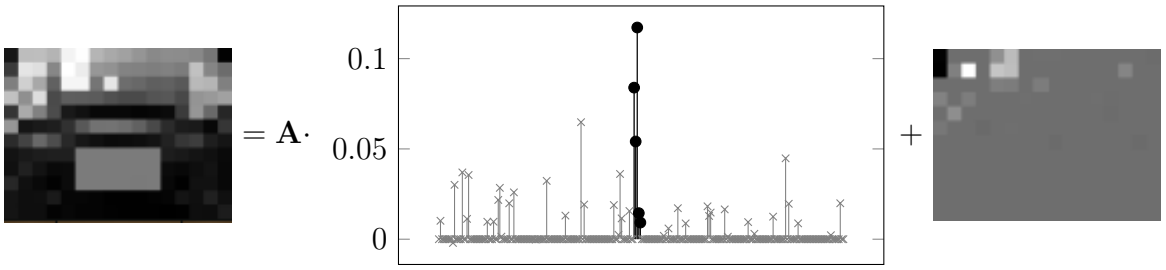


Abbildung 4.2: Sparse Representation Classification

Das Bild zeigt das Ergebnis des Minimieralgorithmus für einen gegebenen Datensatz $\mathbf{A}$ *und ein Testsample wie auf der linken Seite gezeigt. Die Elemente, die laut dem Algorithmus in Abb. 4.1 zur ermittelten Klasse gehören sind hervorgehoben. Zur Dimensionsreduktion wurden die Samples hier auf eine größe von* 16×12 *Pixel gebracht. Die Klassifikation war korrekt.*

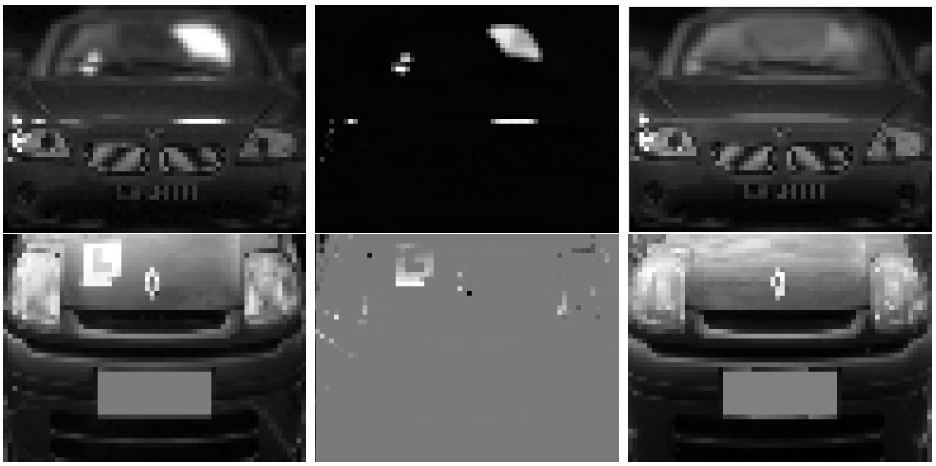


Abbildung 4.3: Fehlerkorrektur bei SRC
Links: *Originalbilder* ***Mitte:*** *Vom l_1-Minimierer ermitteltes Fehlerbild. Dieses ist auf den Bereich $[0;1]$ normiert. Im oberen Bild entsprechen folglich schwarze Pixel Fehlern von 0, im unteren graue.* ***Rechts:*** *Anhand des Trainingsdatensatzes rekonstruiertes Bild*

4.2.3 Dimensionsreduktion

Bilder in ihrer vollen Auflösung zu klassifizieren führt zu hochdimensionalen Merkmalsvektoren. Bei einer relativ niedrigen Auflösung von 320×240 Pixeln hat ein Merkmalsvektor eine Dimension von $M = 76800$. Solche Größenordnungen sind wie [Wright et al., 2008] angibt und Abschnitt B.2 bestätigt auch für spezialisierte $l1$-Minimierer nicht vernünftig realisierbar.

Es stellt sich also die Frage nach einer geeigneten Dimensionsreduktion. In [Wright et al., 2008] wurden hierzu diverse Methoden, wie die Hauptkomponentenanalyse die zu den bekannten „Eigenfaces" führt oder die Lineare Diskriminanzanalyse („Fisherfaces") untersucht. Ebenfalls untersucht wurden simples Resampling der Bilder auf Auflösungen in der Größenordnung von 10×12 Pixeln oder Transformationen mit einer Matrix deren Zeilenvektoren aus auf Einheitslänge gebrachte Zufallsvektoren (dort genannt „Randomfaces") besteht.

Interessanterweise führten erstere, eher aufwändige Methoden bei der Verwendung zu-

sammen mit SRC zu keinem Vorteil gegenüber den letzten, sehr simplen und einfach umzusetztenden Methoden. Aufgrunddessen und der in Kapitel 3 angesprochenen Tatsache, dass die Hauptkomponentenanalyse keinen Vorteil auch bei der Klassifikation mit NN liefert, wird diese auch in der vorliegenden Arbeit nicht verwendet. Die einzige verwendete Methode zur Dimensionsreduktion ist das Resampling auf eine niedrigere Auflösung.

4.3 Algorithmen zur l_1-Minimierung

Im folgenden soll stabilisierte Problem (4.5) betrachtet werden. Dieses ist wie sich zeigt, ein Beispiel für ein konvexes Optimierungsproblem.
Bei konvexen Optimierungsproblemen wird eine Lösung gesucht, die eine konvexe Zielfunktion $f(\cdot)$ über einer konvexen Menge $D \subseteq \mathbb{R}^N$ minimiert. Abbildung 4.4 veranschaulicht die Begriffe *konvexe Menge* und *konvexe Funktion*. In (4.5) entspricht $f(\cdot)$

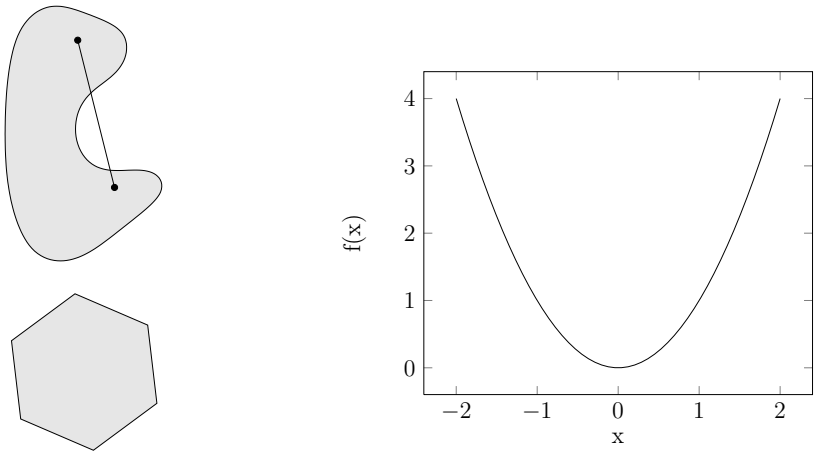


Abbildung 4.4: Konvexe Mengen und Funktionen
Links Oben: *Ein Sechseck als konvexe Menge.* ***Links Unten:*** *Diese Menge ist nicht konvex, da eine Gerade zwischen zwei Punkten der Menge existiert, die nicht vollständig in der Menge selbst enthalten ist. (aus* [Boyd und Vandenberghe, 2004]*)* ***Rechts:*** *Eine konvexe Funktion. Eine nach unten geöffnete Parabel wäre konkav.*

der l_1-Norm, wie sich zeigen lässt einer konvexen Funktion. Die Menge D ist, definiert durch die Nebenbedingung $||\mathbf{A}\underline{\boldsymbol{x}} - \underline{\boldsymbol{y}}||_2 \leq \varepsilon$ eine Ebene und folglich eine konvexe Menge.

Was konvexe Optimierung handlich macht, ist die Tatsache, das ein gefundenes lokales Optimum stets mit dem globalen Optimum übereinstimmt. [Boyd und Vandenberghe, 2004] [Baraniuk et al., 2011]

Etablierte Verfahren zur Lösung konvexer Optimierungsprobleme mit Nebenbedingungen nutzen den Umstand, dass die Lösung spärlich ist nicht aus und sind deshalb verhältnismäßig ineffektiv. Mit dem Aufkommen des *Compressed Sensing* wurden in den letzten Jahren Algorithmen entwickelt, die die Spärlichkeit berücksichtigen und dadurch kürzere Laufzeiten ermöglichen. Hier soll lediglich auf letztere eingegangen werden. Tabelle 4.1 zeigt einen Überblick über $l1$-Minimierer, die bei der Literaturrecherche gefunden wurden und zur SRC eingesetzt wurden.

In [Yang et al., 2010] findet sich eine ausführliche Evaluation vorhandener l_1-Minimierer auch bezüglich der Anwendung zur Gesichtserkennung. Demnach haben sich zwei Methoden hervorgetan:

- Ein *Homotopie-Methode* genannter Algorithmus und
- eine Erweiterung der Lagrange-Multiplikator-Methode

Beide sollen hier kurz erläutert werden. In Anhang A befindet sich erläuterter Pseudocode für beide Algorithmen.

Im allgemeinen wird das Problem (4.5) umformuliert, um ein Minimierungsproblem ohne Einschränkungen zu erhalten. In der Regel wird ein Lagrange-Multiplikator λ eingeführt, über den die Nebenbedingungen in die Zielfunktion integriert werden. Das Resultat ist die Lagrangefunktion $F(\underline{\boldsymbol{x}}, \lambda)$. Die Lösung des Minimierungsproblem zeichnet sich dadurch aus, Sattelpunkt dieser Lagrangefunktion zu sein.

Tabelle 4.1: Übersicht über Veröffentlichungen zu l_1-Minimierung bei SRC

Verfahren	Bemerkungen	Veröffentlichung
CVX / CVXOPT[a]	Modellierungs und -lösungssystem für MATLAB / Python auf Basis des Inneren-Punkte-Verfahrens nach [Boyd und Vandenberghe, 2004]	[Wright et al., 2008]
GPSR[b]	Gradientenprojektionsverfahren nach [Figueiredo et al., 2007]	[Huang et al., 2008]
l_1-LS[c]	l_1-regularisierte kleinste Quadrate. Ebenfalls auf Basis des Inneren-Punkte-Verfahrens nach [Kim et al., 2007].	[Mei und Ling, 2011]
l_1-Benchmark[d]	Evaluation diverser l_1-Minimierer	[Yang et al., 2010]

[a] `http://cvxr.com/cvx/` bzw. `http://abel.ee.ucla.edu/cvxopt/`
[b] `http://www.lx.it.pt/~mtf/GPSR/`
[c] `http://www.stanford.edu/~boyd/l1_ls/`
[d] `http://www.eecs.berkeley.edu/~yang/software/l1benchmark/`

Homotopie

Bei dieser Methode wird das uneingeschränkte Problem (4.13) betrachtet.

$$\arg\min_{\underline{\boldsymbol{x}}} F(\underline{\boldsymbol{x}}) = \frac{1}{2}||\mathbf{A}\underline{\boldsymbol{x}} - \underline{\boldsymbol{y}}||_2^2 + \lambda||\underline{\boldsymbol{x}}||_1 \tag{4.13}$$

Eine Bedingung für die Optimalität einer Lösung $\underline{\boldsymbol{x}}$ ist, dass der Gradient der Zielfunktion $\nabla F(\underline{\boldsymbol{x}})$ Null ist. Der Algorithmus startet bei $\underline{\boldsymbol{x}} = \underline{\mathbf{0}}$ und findet sukzessive eine Lösung durch hinzufügen und entfernen von Elementen des Supports $\mathcal{I} = \text{supp}(\underline{\boldsymbol{x}})$. Er vollzieht dabei einen Gradientenabstieg eingeschränkt auf die Koordinaten in $\mathcal{I}$, bis die obige Bedingung nicht mehr erfüllt ist, und ein Element $\mathcal{I}$ hinzugefügt oder entfernt werden muss.

Dieser Algorithmus ist einer der schnellsten, unter der Vorraussetzung, dass die Lösung spärlich genug ist. Die Laufzeit erhöht sich jedoch schnell je mehr Elemente im

Lösungsvektor enthalten sind. [Donoho und Tsaig, 2008]

Erweiterter Lagrange-Multiplikator

Die Methode des erweiterten Lagrange-Multiplikators, *engl.:* Augmented Lagrangian Multiplicator (ALM), formuliert das Problem wie in (4.14) wobei $\mu \gg 0$ eine Konstante darstellt, die Lösungen mit großer Abweichungen von der ursprünglichen Nebenbedingung abstraft.

$$\arg\min_{\underline{\boldsymbol{x}}} F(\underline{\boldsymbol{x}}, \underline{\boldsymbol{\lambda}}) = ||\underline{\boldsymbol{x}}||_1 + \underline{\boldsymbol{\lambda}}^T(\underline{\boldsymbol{y}} - \mathbf{A}\underline{\boldsymbol{x}}) + \frac{\mu}{2}||\underline{\boldsymbol{y}} - \mathbf{A}\underline{\boldsymbol{x}}||_2^2 \tag{4.14}$$

Die Lösung wird iterativ durch Aktualisieren der Lösung $\underline{\boldsymbol{x}}$ und des Lagrange-Multiplikators $\underline{\boldsymbol{\lambda}}$ gefunden. Während dabei in jedem Schritt für $\underline{\boldsymbol{\lambda}}$ eine analytische Lösung existiert, muss für $\underline{\boldsymbol{x}}$ wieder ein Optimierungsproblem gelöst werden. Dieses ist dabei so konditioniert, dass es effizient lösbar ist. Dieser Algorithmus konvergiert schnell und ist robuster gegenüber Rauschen und variierender Spärlichkeit der Lösungsvektoren.[Yang et al., 2010] [Wagner et al., 2010]

Zusammenfassung

Dieses Kapitel stellt den untersuchten Klassifikationsalgorithmus SRC vor, erklärt seine Wurzeln im *Compressed Sensing* und geht auf das essenzielle Verfahren der l_1-Minimierung ein. Mit den Merkmalsextraktionsverfahren aus Kapitel 3 sind nun alle benötigten Komponenten erläutert. Das nächste Kapitel schildert das Vorgehen zur Implementierung und geht auf die Parameter- und Algorithmenwahl ein.

Kapitel 5

Implementierung

Dieses Kapitel dient der Erläuterung der Realisierung der Umgebung mit der die Experimente durchgeführt wurden. Hierbei soll im ersten Teil auf die verwendete Software eingegangen werden. Die übrigen Abschnitte erläutern Details zu den Parametern der Merkmalsextrahierung und der Klassifikationsmethoden.

5.1 Werkzeuge

Hauptwerkzeug für die Durchführung der Experimente war die Scriptsprache *Python*. Sie wurde aufgrund der hohen Verfügbarkeit von wissenschaftlichen Erweiterungen (*NumPy*, *SciPy*, *mlpy*) und der hohen Flexibilität ausgewählt. Durch das Paket *mlabwrap* kann MATLAB-Code bei einer vorhanden MATLAB-Installation direkt aus *Python* heraus ausgeführt werden. Desweiteren wurde für Aufgaben der Bildverarbeitung die Python-Schnittstelle der OpenCV-Bibliothek verwendet.

5.2 Merkmalsextraktion

Zur Mermkalsextraktion werden fünf Verfahren implementiert. Die Samples werden mit Methoden der OpenCV-Bibliothek als gewichtete Grauwertbilder geladen. Die Gradientenbasierten verfahren machen dabei vom in OpenCV implementierten Sobel-Filter Gebrauch. Verwendung findet der in Abb. 3.2 gezeigte Filterkern.

Tabelle 5.1: Übersicht über verwendete Software

Paket/Software	Version	Quelle
Python	2.6.4	
NumPy	1.3.0	
SciPy	0.7.0	
mlpy	2.2.2	`http://mlpy.fbk.eu/`
mlabwrap	1.1	`http://mlabwrap.sourceforge.net/`
MATLAB	7.8.0.347 (R2009a)	
OpenCV	SVN-Rev 4815	

Der Gabor-Filter und die Lokalen Energie werden über die Faltung mit einem entsprechenden Filterkern implementiert. Eine Python-Funktion erstellt den Kern mit gegebenen Ortsfrequenzen und Orientierungen. Die Frequenzen sind in beiden Verfahren $f \in \{0.33, 0.16, 0.08, 0.04\}$. Acht Orientierungen werden im Bereich $\phi = 0...\pi$ Gleichmäßig verteilt. Der Parameter σ der Gaußglocke beträgt 1.0, die Größe des Kerns 24 Pixel.
Die Phasenkongruenz wird durch ein vom Autor von [Kovesi, 1999] bereitgestelltes MATLAB-Skript berechnet ([Kovesi, 2011]). Dabei werden die voreingestellten Parameterwerte verwendet.

5.3 Klassifikation

Zur Klassifikation mit NN und SVM wird die Python-Bibliothek *mlpy* verwendet. Als Distanzmaß für den NN-Algorithmus wird in [Petrovic und Cootes, 2004] das Skalarprodukt verwendet, um welches die *mlpy*-Bibliothek erweitert wurde. *mlpy* implementiert den kNN-Algorithmus. Die Verwendung von mehr als einem Nachbar führte jedoch zu keinem Vorteil, weshalb der einfache NN-Algorithmus verwendet wird. Bei der Klassifikation durch SVM besteht in *mlpy* die Möglichkeit, die libSVM-Bibliothek ([Chang und Lin, 2011]) zu verwenden. Diese hat Mehrklassen-Funktionalität über den „one-against-one"-Ansatz implementiert. Verwendet wird eine

lineare SVM, der Regularisierungsparameter C wurde empirisch auf $C = 50$ festgelegt. Die l_1-Minimierer die zur SRC verwendet werden, sind

- l_1-LS aus dem gleichnamigen MATLAB-Paket,
- Homotopy und
- Dual Augmented Lagrange Multiplier (DALM) aus dem l_1-Benchmark MATLAB-Paket

wie in Tabelle 4.1 aufgelistet. Tabelle 5.2 listet die zu jedem Verfahren verfügbaren Parameter und die in den Experimenten verwendeten Werte, falls vom vordefinierten Wert abweichend. Die Werte wurden experimentell ermittelt.

Tabelle 5.2: Parameter zur l_1-Minimierung

Parameter	**Bemerkungen**	**Wert**
l_1-LS		
λ	Regularisierungsparameter	$0,01$
`rel_tol`	Toleranz	$0,01$
Homotopy		
λ	Kleinster Wert für den Regularisierungsparameter bei dem die Optimierung beendet wird.	$0,001$
`stoppingcriterion`	Haltekriterium für die Optimierung	2
`tolerance`	Toleranz	$0,001$
Dual Augmented Lagrange Multiplier		
λ	–	$0,01$
`tolerance`	Toleranz	$0,01$

Kapitel 6

Experimente

Dieses Kapitel stellt die durchgeführten Experimente und die erhaltenen Ergebnisse vor. Ziel der Experimente war es, das am besten für SRC geeignete Merkmalsextraktionsverfahren zu ermitteln und das Laufzeitverhalten und die Robustheit des SRC-Algorithmus zu untersuchen. Zunächst werden die Datensätze erläutert, mit denen die Experimente durchgeführt wurden. Anschließend werden die Experimente erläutert und die Ergebnisse gezeigt.

6.1 Datensätze

Es werden zwei Datensätze verwendet: Ein selbst aufgenommener Datensatz (eigener Datensatz) und ein Datensatz, der von den Autoren von [Petrovic und Cootes, 2004] zur Verfügung gestellt wurde (Petrovic-Datensatz).
Die Erstellung des eigenen Datensatzes war durch [Wagner et al., 2010] motiviert, wo Aufnahmen von Gesichtern bei unterschiedlichen Beleuchtungsverhältnissen zur Identifikation mit SRC verwendet werden. Es wurden nach dem selben Prinzip Frontansichten von Modellfahrzeugen aufgenommen, die über einen kleinen Beleuchtungsaufbau unterschiedlich ausgeleuchtet wurden. Diese Aufnahmen führten zu einem Datensatz mit 10 unterschiedlichen Modellen, bei allerdings nur geringfügigen Unterschieden innerhalb der Modellklassen. Die Samples haben eine Auflösung von 320×240 Pixel.

Aufgrund der niedrigen Inner-Klassen-Varianz stellt das Klassifikationsproblem mit diesem Datensatz ein sehr leichtes dar, weshalb dieser lediglich zur Untersuchung der Robustheit des SRC verwendet wird. Abbildung 6.1 zeigt einen Ausschnitt daraus.

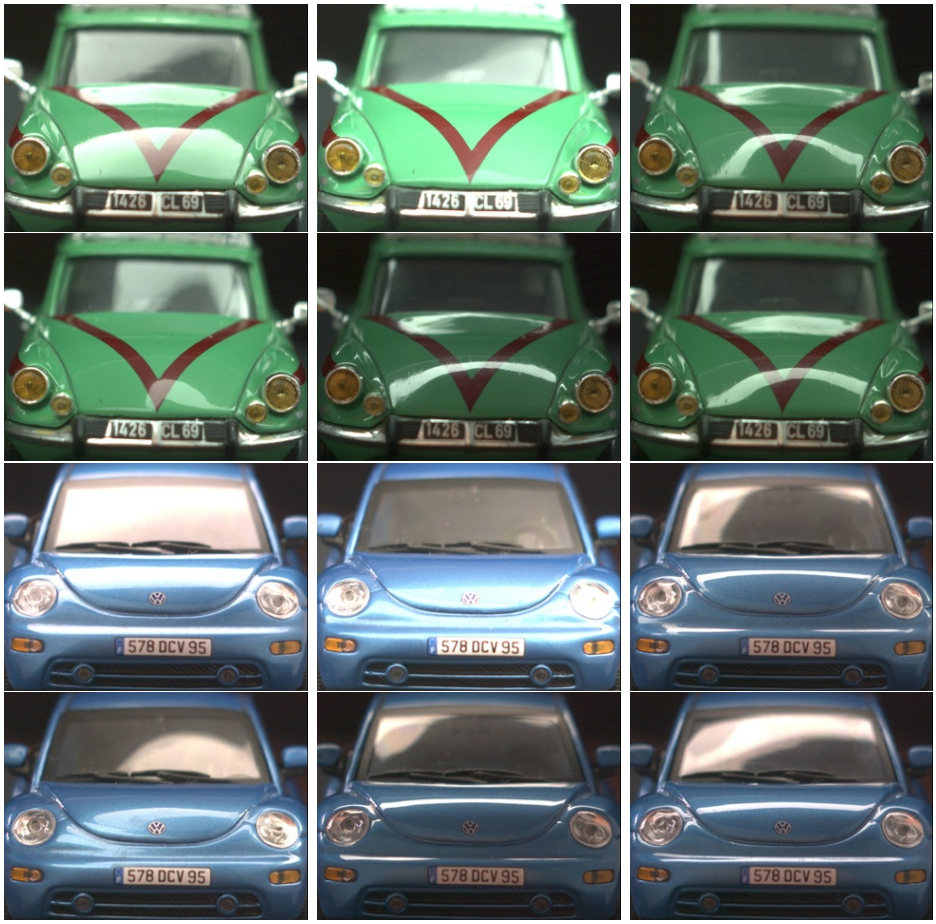


Abbildung 6.1: Ausschnitt aus dem selbst aufgenommenen Datensatz
Gezeigt werden jeweils sechs Samples von zwei Modellen des Datensatzes. Im Datensatz existieren nur zwei Modelle, die in unterschiedlicher Lackierung vorhanden sind. Die Inner-Klassen-Varianz ist gering, weshalb dieser Datensatz nur in den Experimenten verwendet wird, die die Testsamples verfälschen um die Robustheit von SRC zu untersuchen.

Beim Petrovic-Datensatz handelt es sich um den Datensatz, der in Abschnitt 2.3 vorgestellt wurde. Dabei werden nur die 52 der 77 vorhandenen Modelle verwendet, von denen mindestens zehn Samples vorhanden sind. Bei den Aufnahmen handelt es sich um Frontansichten von parkenden Autos bei unterschiedlichen Witterungsbedingungen, Lichtverhältnissen und mit verschiedenen Lackierungen. Die Bilder sind über die Position des Kennzeichens registriert und haben eine Auflösung von 320×240 Pixeln. Dieser Datensatz kommt bei allen Experimenten zum Einsatz. Abbildung 6.2 zeigt einen Ausschnitt daraus.

Abbildung 6.2: Ausschnitt aus dem Petrovic-Datensatz
*Auch dieses Bild zeigt sechs Samples von zwei Klassen des Datensatzes. Dieser besteht aus einer weit größeren Zahl von Modellen (*52*) mit vielfältigeren Samples in jeder Klasse. Diese sind durch die Position des Kennzeichens registriert.*

6.2 Vorgehensweise

Mit den Merkmalsextraktoren und den Klassifikatoren aus den vorigen Kapiteln und dem Datensatz können nun die Experimente erläutert werden. Diese werden so ausgelegt, dass eine Aussage getroffen werden kann über

1. die Eignung der in Kapitel 3 vorgestellten Verfahren zur MMR mit SRC,
2. das Laufzeitverhalten von SRC in Abhängigkeit von der Anzahl an Klassen und der Merkmalsdimension und
3. die Robustheit von SRC gegenüber Verfälschungen.

Die nächsten drei Abschnitte befassen sich jeweils mit einem dieser drei Punkte und erläutern im Detail das Vorgehen und die Ergebnisse. In Anhang B finden sich die genauen Werte, die den Diagramme in den folgenden Abschnitten zugrundeliegen.

6.3 Vergleich der Merkmalsextraktoren

Für den ersten Punkt werden alle Methoden aus Kapitel 3 verwendet, nämlich

- Gabor-Filter (GAB),
- Lokale Energie (LE),
- Phasenkongruenz (PC),
- Sobel-Filter (SOB),
- Square-Mapped-Gradient (SMG) und
- das Ursprungsbild (RAW) als Referenz.

Die Samples des Petrovic-Datensatzes werden vorher auf eine einheitliche Größe von 17×40 Pixeln gebracht. Die Wahl dieser Größe wird in Abschnitt 6.5.1 erläutert. Die Klassifikatoren werden mit fünf zufällig gewählten Samples je Modell trainiert.

Dieser Trainingssatz wird vier mal neu ausgewürfelt und zum Schluss der Mittelwert der ermittelten Erkennungsraten ($P_{Er} = \frac{\#\text{korrekt klassifizierte Samples}}{\#\text{Testsamples}}$) berechnet. Als Minimierer für SRC kommt der *DALM*-Algorithmus zum Einsatz. Abbildung 6.3 stellt die erhaltenen Ergebnisse dar: Als bester Merkmalsextraktor hat sich der Square-Mapped-Gradient gezeigt. Ein Ergebnis, das auch in [Petrovic und Cootes, 2004] für NN ermittelt wurde. In den übrigen Experimenten werden deswegen das Ursprungsbild (RAW) und der Square-Mapped-Gradient (SMG) verglichen.

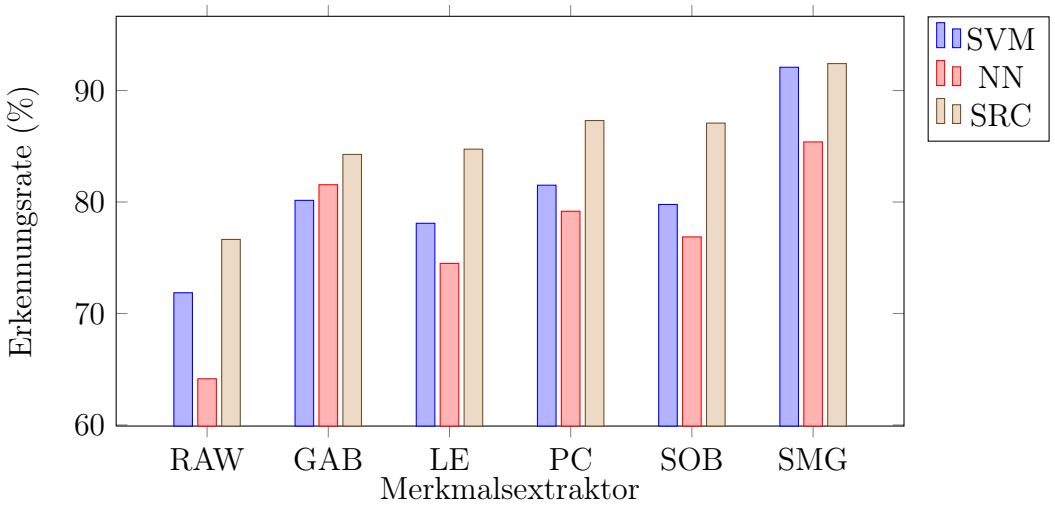


Abbildung 6.3: Vergleich der Merkmalsextraktoren

Das Ergebnis der Untersuchung der Merkmalsextraktoren: Die drei Frequenzbasierten Merkmalsextraktoren halten sich mehr oder weniger auf dem selben Niveau. Eindeutig am besten eignet sich bei allen Klassifikatoren der Square-Mapped-Gradient. SRC liefert bei allen Extraktoren das beste Ergebnis. Bei der Verwendung des SMG ist die SVM allerdings fast gleich auf.

6.4 Laufzeit

Für den zweiten Punkt werden die drei in Abschnitt 5.3 erwähnten l_1-Minimierer l_1-*LS*, *Homotopy* und *DALM* verwendet. Als Merkmalextraktoren kommt einmal reines Resampling (RAW) und einmal der Square-Mapped-Gradient (SMG) zum Einsatz.

Es wird die Laufzeit in Abhängigkeit der Bildgröße auf der Gesamten Datenbank (52 Klassen) und in Abhängigkeit der Anzahl an bekannten Klassen in der Datenbank bei fixer Merkmalsdimension (17×40 Pixel) untersucht. Bei der Ermittlung der Laufzeit wird über fünf Klassifiziervorgänge gemittelt.
Bei der Untersuchung der Bildgröße, muss die Klassifikationsleistung auch mit in Betracht gezogen werden. Die Abbildungen 6.4 und 6.5 zeigen die Ergebnisse dieser Untersuchungen.
Dabei sei erwähnt, dass bei einer Auflösung von 50×120 die Klassifikation mit SRC eine übermäßig lange Laufzeit bei allen Minimierern aufweist. In Verbindung mit dem Square-Mapped-Gradient als Merkmalsextraktor reichte bei dieser Bildgröße zudem der Arbeitsspeicher nicht mehr aus. Ein guter Kompromiss zwischen Laufzeit und Klassifikationsleistung ist eine Bildgröße von 17×40 Pixeln. Die Laufzeiten von SVM oder NN-Klassifikatoren fallen im Rahmen der hier untersuchten Merkmalsdimensionen hingegen kaum ins Gewicht. Dies ist auch der Grund für die in Abschnitt 6.5.3 gezeigten Experimente.
Beim Vergleich der Minimierer in Abhängigkeit von der Anzahl an bekannten Klassen in der Datenbank (Abb. 6.6) zeigen sich kaum unterschiede zwischen dem Homotopy-Algorithmus und dem DALM-Algorithmus. Beide bleiben nahezu konstant obwohl die Anzahl der Klassen zunimmt. Der l_1-LS-Algorithmus weist als einziger eher lineares Verhalten auf.

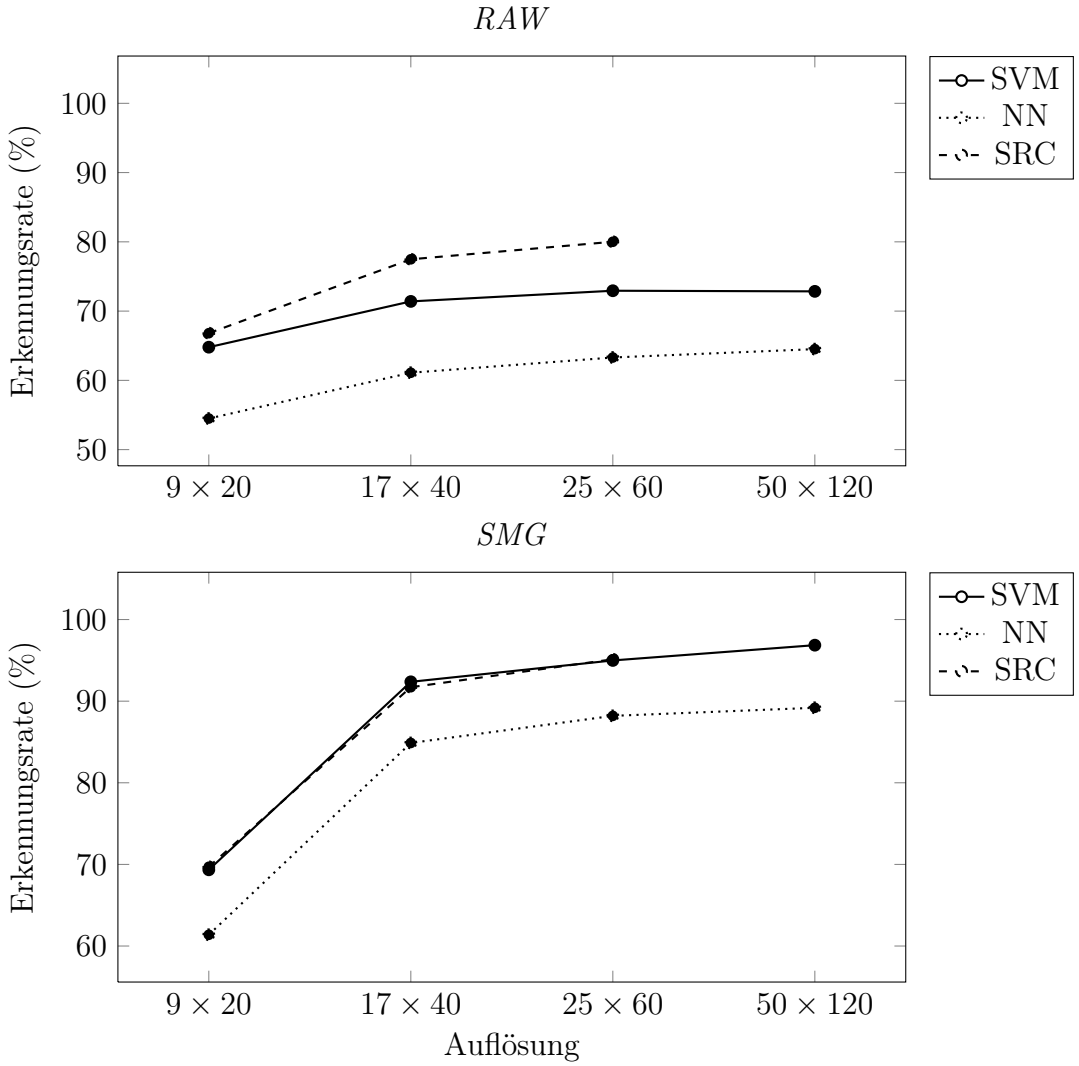


Abbildung 6.4: P_{Er} in Abhängigkeit der Bildgröße

Die Klassifikationsleistung verbessert sich bei Bildgrößen überhalb von 17×40 *nur noch in Schritten von* $\sim 2\%$*. Bei einer Auflösung von* 50×120 *ist eine Klassifikation mit SRC effektiv nicht mehr möglich.*

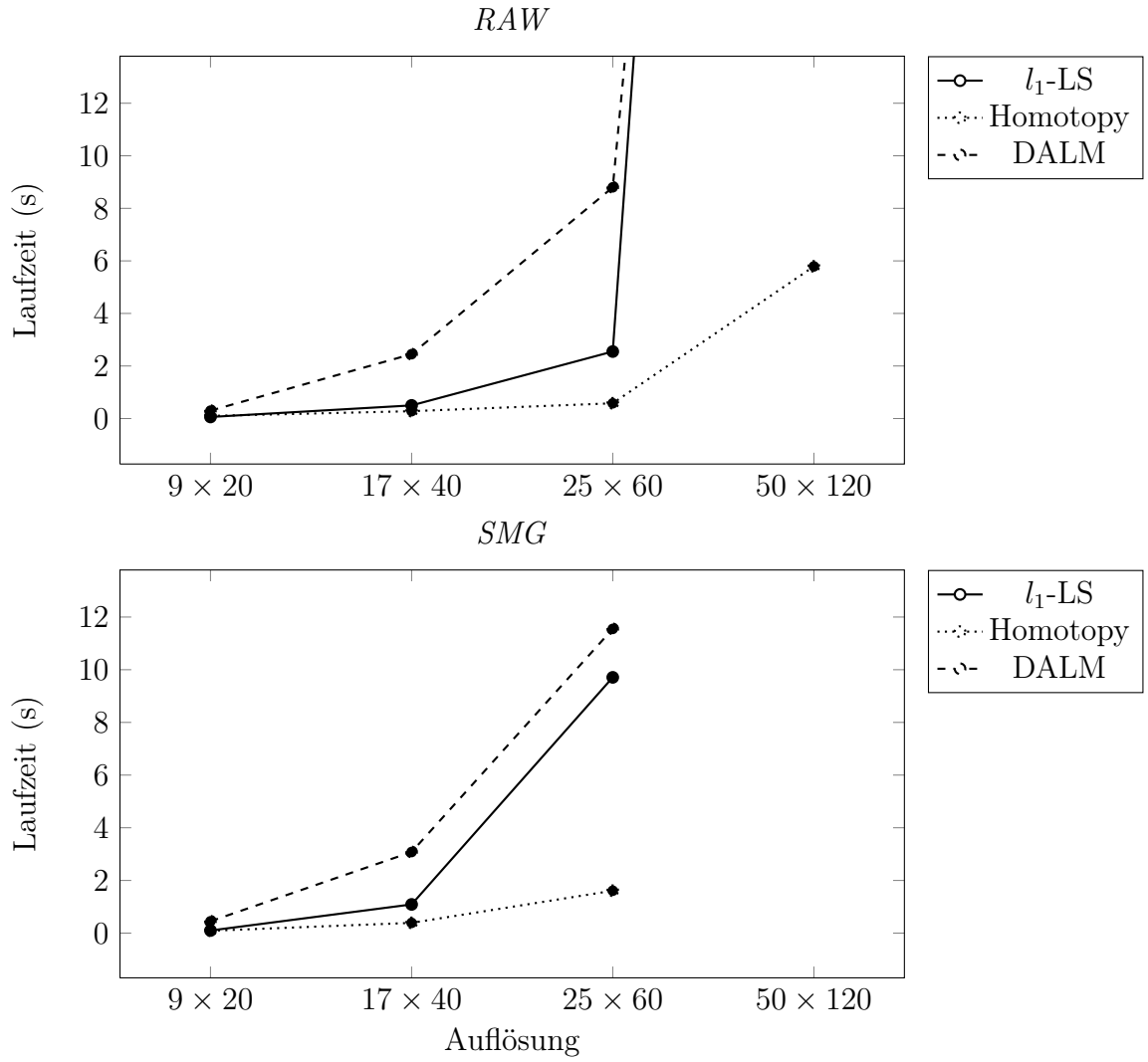


Abbildung 6.5: Laufzeit in Abhängigkeit der Bildgröße

Es wurden drei l_1-Minimierer verglichen. Der Homotopy-Algorithmus hat sich wie erwartet als der schnellste herausgestellt. Bei einer Auflösung von 50×120 ist die Laufzeit übermäßig hoch. Für den Square-Mapped-Gradient reichte zudem der Arbeitsspeicher nicht mehr aus (Ein Merkmalsvektor hat dann die doppelte Größe, da er aus zwei Gradientenbildern besteht).

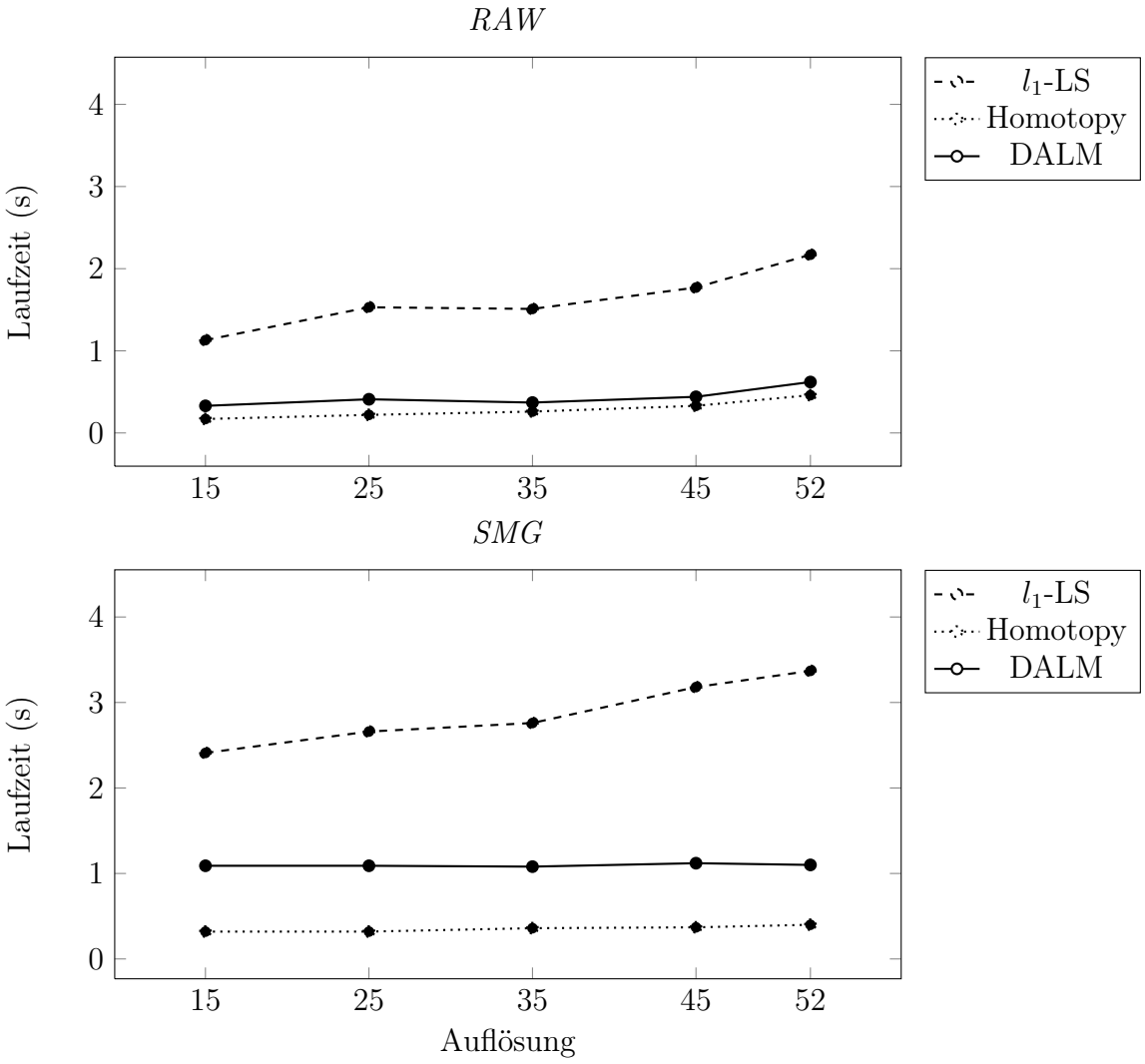


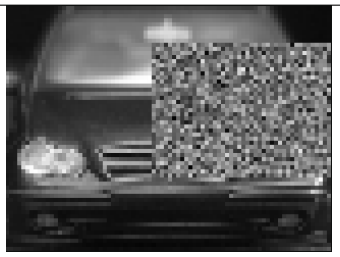
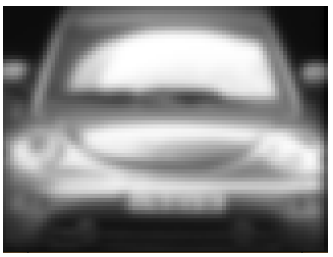
Abbildung 6.6: Laufzeit in Abhängigkeit der Anzahl an Klassen

Alle Minimierer skalieren gut mit der Anzahl an Klassen in der Datenbank. Bei Homotopy- und DALM-Algorithmus bleibt die Laufzeit sogar nahezu konstant.

6.5 Robustheit gegenüber Verfälschungen

Um die Robustheit der Klassifikatoren gegen über Verfälschungen zu untersuchen, wird das Testsample in vier Experimenten vor der Klassifikation zunehmend verfälscht. Tabelle 6.1 zeigt eine Übersicht über die Verfälschungen die den Samples zugefügt wurden.

Tabelle 6.1: Verfälschungen der Tastsamples

Verfälschung	**Beispielbild**
Zusammenhängende Verdeckung Ein zusammenhängender Teil des Samples an zufälliger Stelle wird mit Zufallswerten überschrieben.	
Salt-and-Pepper Rauschen Zufällige Pixel des Samples werden mit Zufallswerten überschrieben.	
Unschärfe Das Sample wird mit einem Gaußfilter weichgezeichnet.	
Verschiebung Das Sample wird in horizontaler und vertikaler Richtung verschoben, um Registrierungsfehler zu simulieren.	

Die Robustheit wurde in drei verschiedenen Konfigurationen untersucht:

- Mit dem eigenen Datensatz unter Verwendung der Originalbilder bei einer Auflösung von 32×24.
- Mit dem Petrovic-Datensatz unter Verwendung der Originalbilder und des Square-Mapped-Gradienten bei einer Auflösung von 17×40.
- In der selben Konstellation wie der vorherigen, mit dem Unterschied, dass die Auflösung für SVM und NN 50×120 beträgt.

Letztere wurde untersucht, da SRC bei der höheren Auflösung zwar nicht mehr verwendet werden kann, die anderen beiden Klassifikatoren davon aber durchaus profitieren.

6.5.1 Eigener Datensatz

Zunächst wurden die erwähnten Experimente mit dem eigenen Datensatz durchgeführt. Die Testsamples werden bei einer Auflösung von 320×240 verfälscht und anschließend auf eine Größe von 32×24 skaliert. Die Trainingssamples wurden nicht wie in den Experimenten mit dem Petrovic-Datensatz zufällig ermittelt. Sie wurden so ausgewählt, dass möglichst unterschiedliche Beleuchtungen vorkommen. Auf Cross-Validation wie in den übrigen Experimenten wurde verzichtet. Abbildungen 6.7 bis 6.10 zeigen die Ergebnisse.

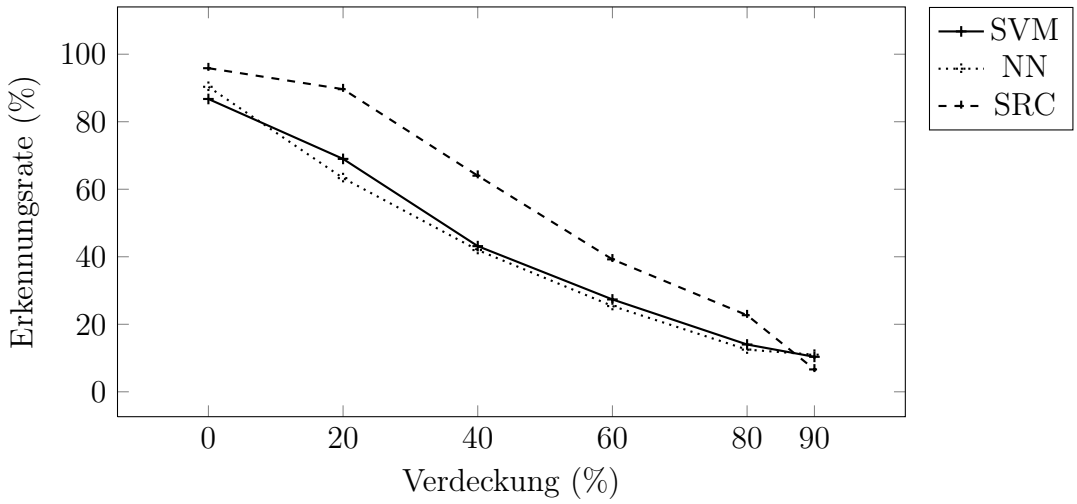


Abbildung 6.7: P_{Er} bei Verdeckungen auf eigenem Datensatz
SVM und NN weisen das gleiche Verhalten bei zunehmender Verfälschung auf. SRC reagiert etwas robuster, die Klassifikationsleistung nimmt schwächer ab.

In dieser Konstellation schneidet der SRC-Algorithmus durchweg besser ab als die SVM oder der NN-Klassifikator. Auf Verdeckung, Unschärfe und horizontale Verschiebung reagiert der SRC besser als die anderen beiden. Bei den übrigen Verfälschungen lässt die Klassifikationsleistung mit zunehmender Verfälschung in etwa gleichermaßen nach, wie sie es bei den übrigen Klassifikatoren tut.
Bemerkenswert ist an dieser Stelle die relative Unempfindlichkeit aller Klassifikatoren gegenüber horizontaler Verschiebung. Wo die Klassifikationsleistung bei vertikaler Ver-

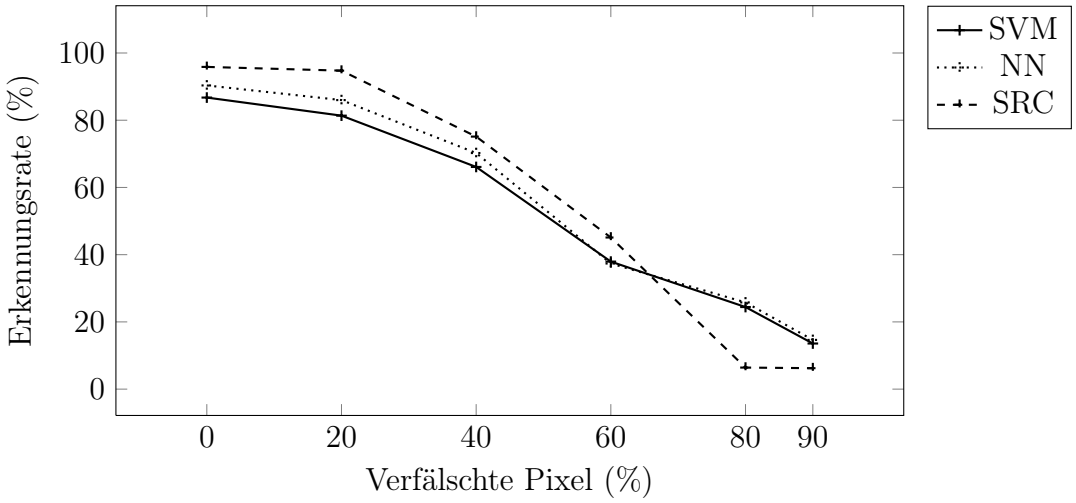


Abbildung 6.8: P_{Er} bei Salt-and-Pepper-Rauschen auf eigenem Datensatz
Alle drei Klassifikationsalgorithmen reagieren hier gleich auf die Verfälschung. Erst bei einer Verfälschungsrate von 80% *zeigen sich unterschiede in der Klassifikationsleistung.*

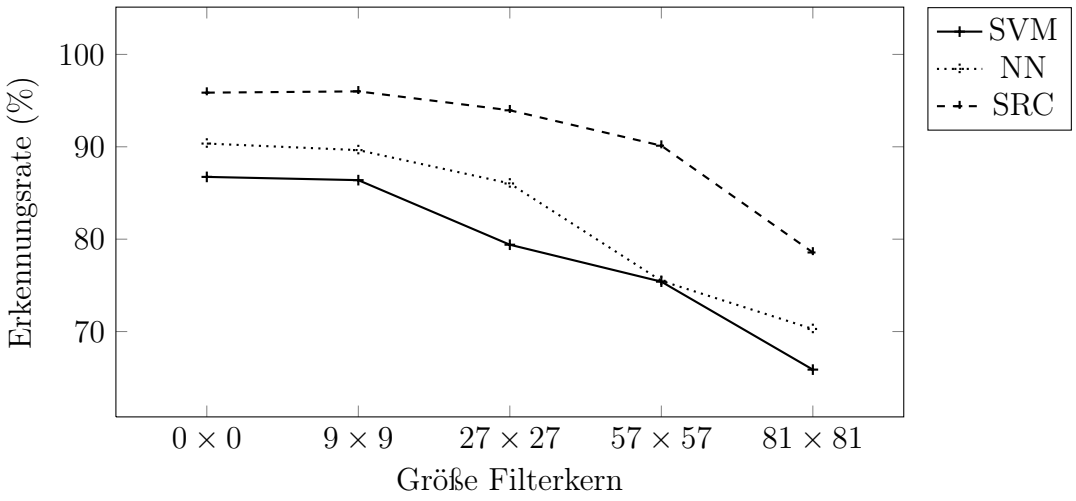


Abbildung 6.9: P_{Er} bei Unschärfe auf eigenem Datensatz
Gegenüber Unschärfe sind alle Methoden weniger empfindlich. Der SRC-Algorithmus verhält sich einen Tick robuster als die anderen beiden.

schiebung bereits auf nahezu 0% gefallen ist (45 Pixel) befindet sie sich bei horizontaler Verschiebung noch bei etwa 60% oder 80% im Fall der SRC.

In [Petrovic und Cootes, 2004] werden die Testsamples auf eine Auflösung von 50×120 skaliert, mit der Begründung, dass sich in horizontaler Richtung mehr redundante Information befindet. Aus diesen Gründen wurden in den übrigen Experimenten dieses Seitenverhältnis verwendet.

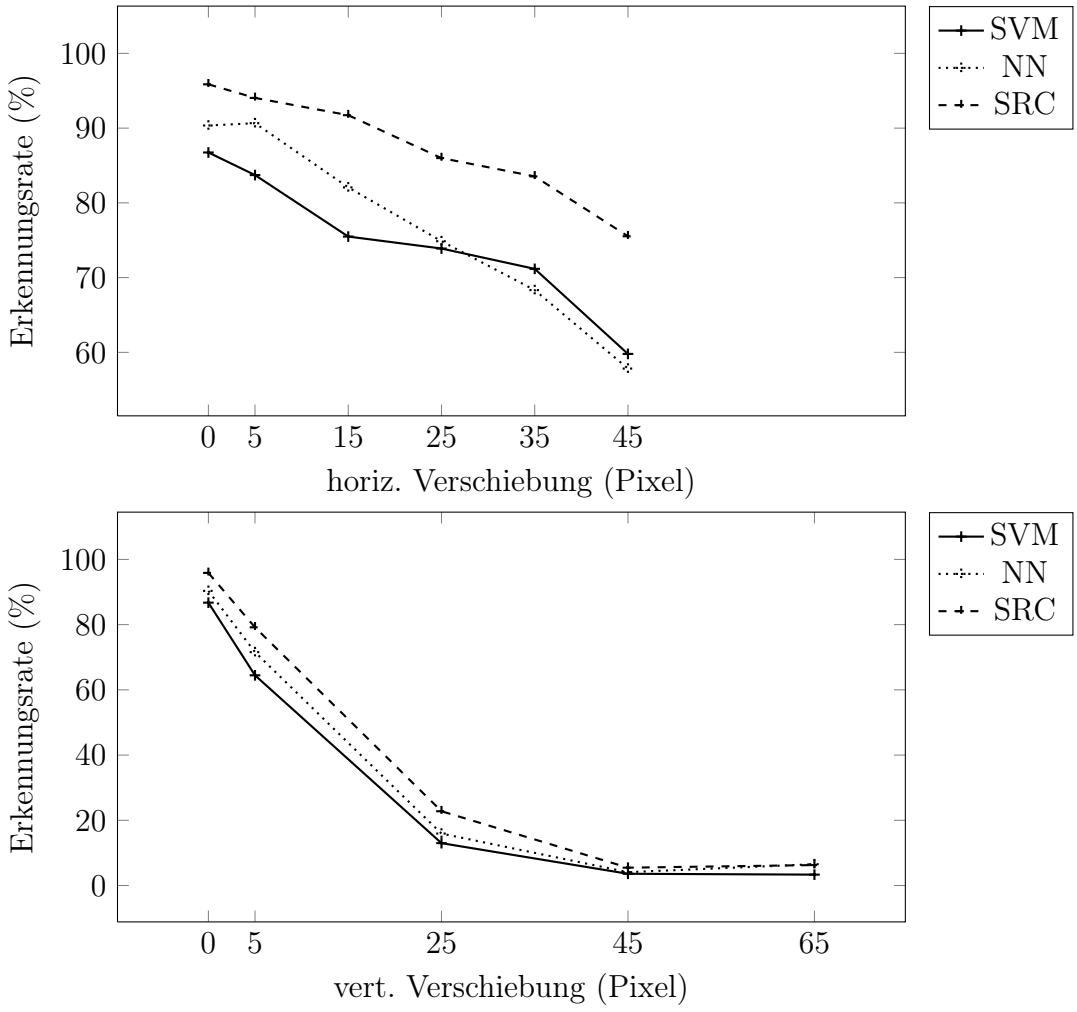


Abbildung 6.10: P_{Er} bei Verschiebungen auf eigenem Datensatz

Alle Klassifikatoren reagieren auf vertikale Verschiebung empfindlicher als auf horizontale. Der SRC-Algorithmus verhält sich bei letzterer außerdem etwas robuster als die anderen beiden.

6.5.2 Petrovic-Datensatz

Die Testsamples des Petrovic-Datensatzes haben vor der Verfälschung ebenfalls eine Auflösung von 320×240 Pixeln. Bei diesem Versuch werden die Testsamples für alle Klassifikatoren nach der Verfälschung auf eine Größe von 17×40 Pixeln gebracht. Zum Vergleich werden die Versuche zwei mal durchgeführt: Einmal unter Verwendung des Square-Mapped-Gradienten und einmal ohne. Die Ergebnisse sind in den Abbildungen 6.11 - 6.14 dargstellt.

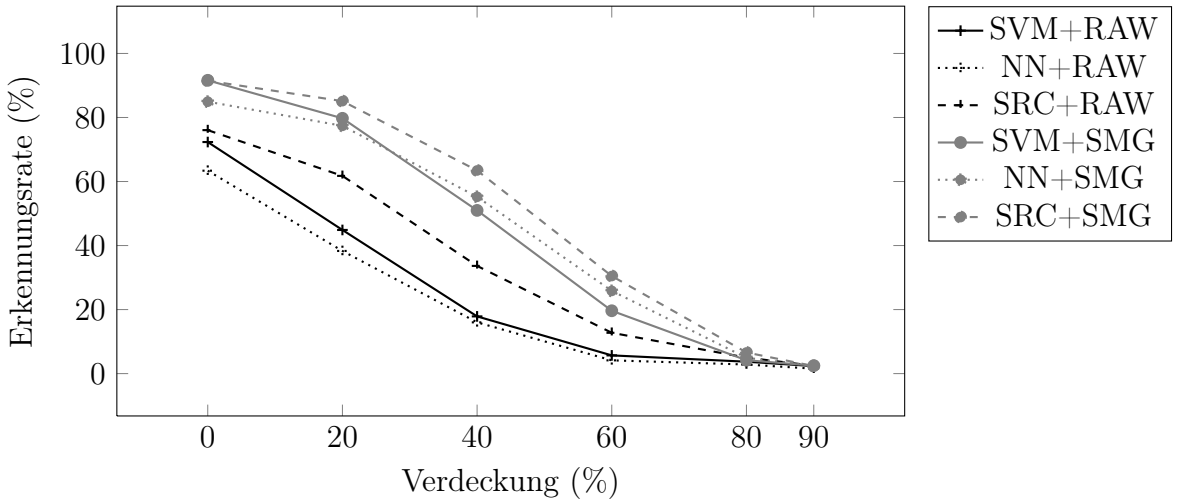


Abbildung 6.11: P_{Er} bei Verdeckungen auf dem Petrovic-Datensatz
Ohne Verwendung des SMG verhält sich der SRC-Algorithmus merklich robuster. Wird er verwendet, verringert sich dieser Vorteil.

Wird der Square-Mapped-Gradient nicht verwendet, also nur mit den skalierten Bildern klassifiziert, zeigt sich ähnliches Verhalten wie im vorigen Experiment: Der SRC-Algorithmus zeigt im Schnitt bessere Klassifikationsleistung bei geringfügig robusterem Verhalten.

Bei verwendung des SMG allerdings zeigt sich eine Angleichung der Klassifikationsleistung der SVM an die des SRC-Algorithmus. Dies hat sich bereits in Abschnitt B.1 angedeutet. Nicht nur nähert sich die SVM an den SRC-Algorithmus an, auch der geringfügige Vorteil bezüglich der Robustheit verringert sich. Dies zeigt sich daran, dass

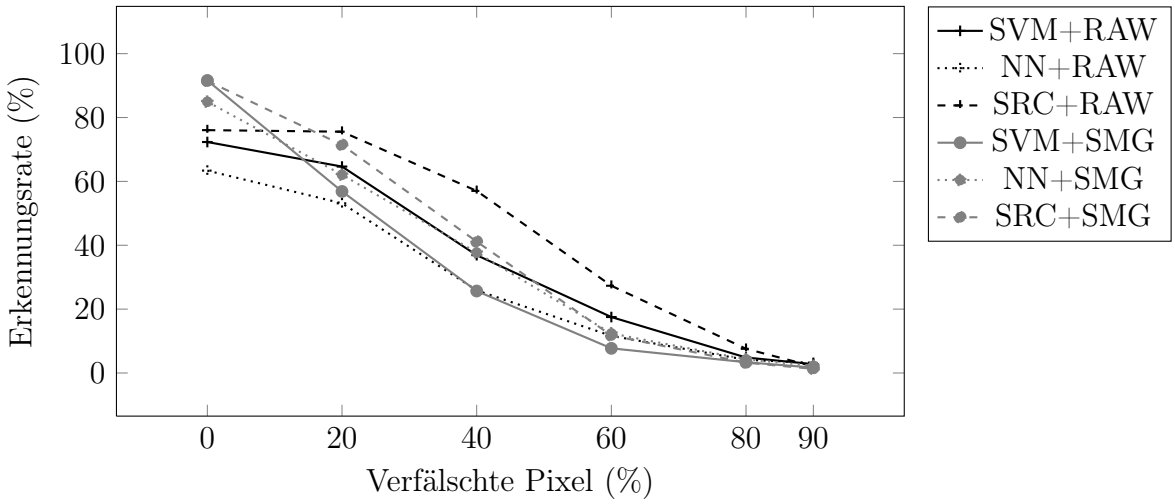


Abbildung 6.12: P_{Er} bei Salt-and-Pepper-Rauschen auf dem Petrovic-Datensatz
Die Hinzunahme des Square-Mapped-Gradienten verringert sogar die Robustheit der Klassifikatoren. Die Klassifikationsleistung sinkt dann schneller mit zunehmender Verfälschung. Der SRC-Algorithmus klassifiziert jeweils besser als SVM oder NN.

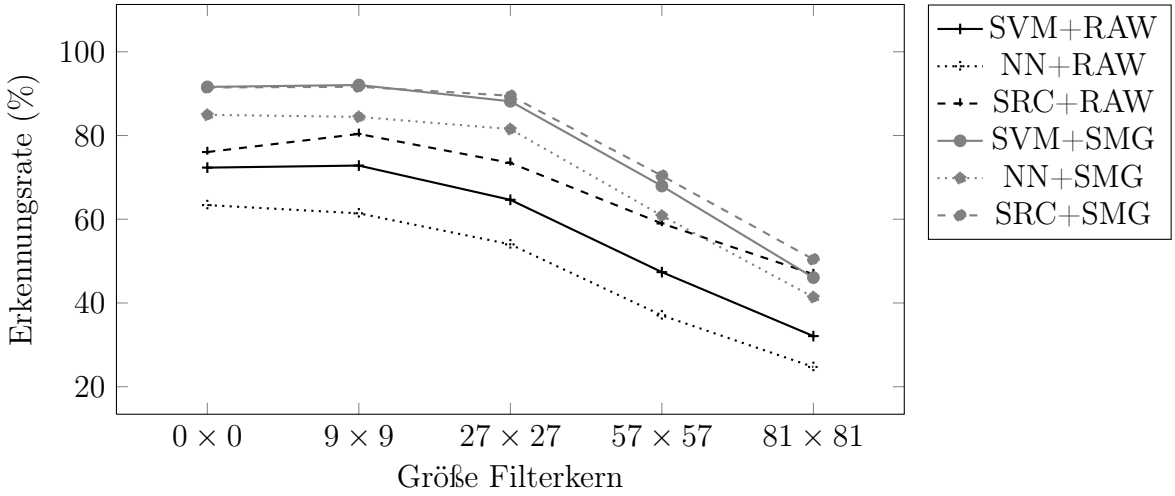


Abbildung 6.13: P_{Er} bei Unschärfe auf dem Petrovic-Datensatz
Bei Verwendung des SMG verringert sich der Vorteil des SRC-Algorithmus deutlich. Ohne weist dieser robusteres Verhalten auf. In beiden Fällen jedoch ist er in der Klassifikationsleistung überlegen.

der Verlauf beider Kurven überwiegend übereinstimmt. Die Verwendung des Square-Mapped-Gradienten scheint den Vorteil des SRC-Algorithmus in puncto Robustheit zu neutralisieren.

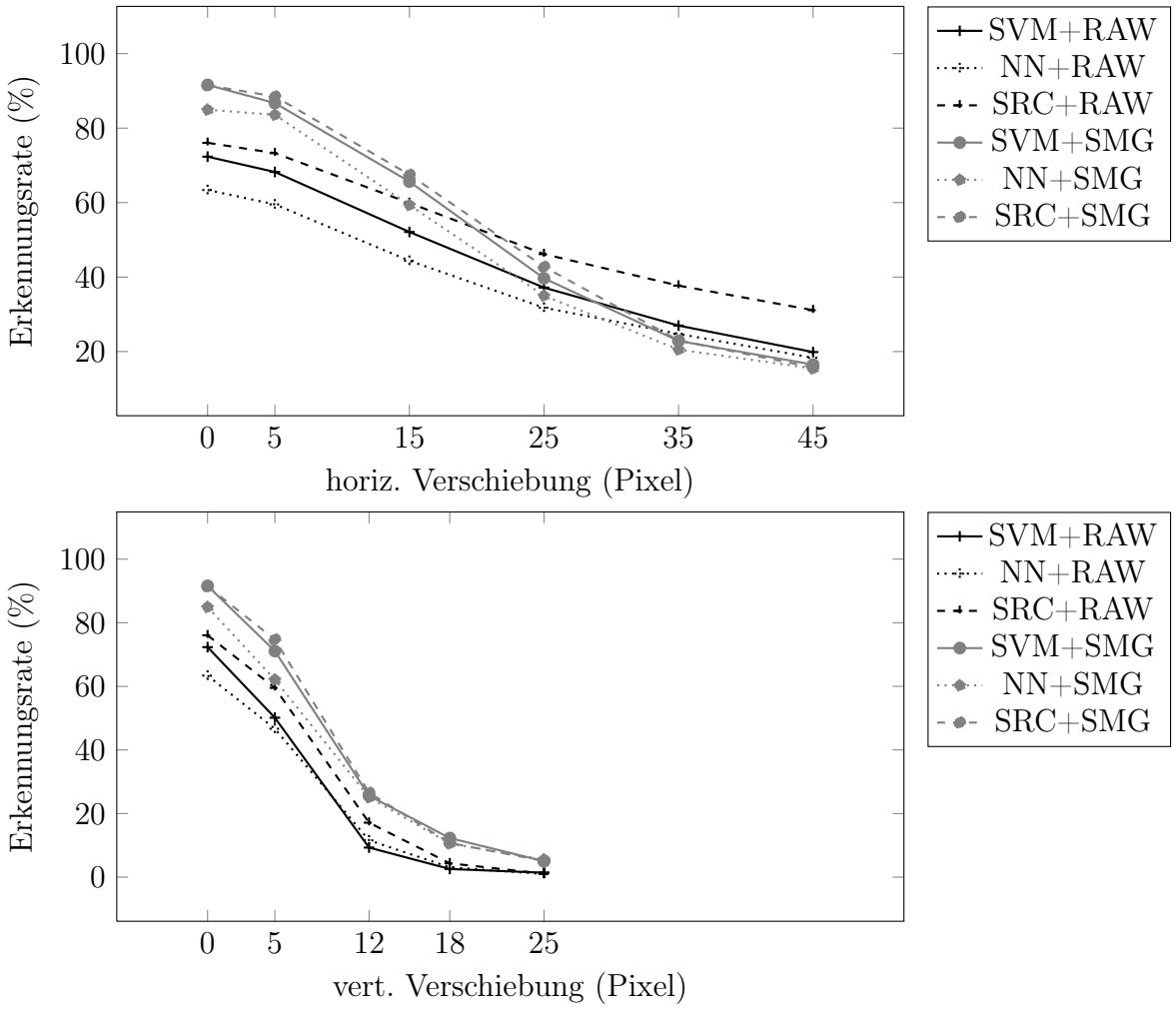


Abbildung 6.14: P_{Er} bei Verschiebungen auf dem Petrovic-Datensatz
Ein ähnliches Bild wie bei den übrigen Experimenten dieser Konfiguration: Geringfügige Vorteile des SRC-Algorithmus ohne SMG; Angleichung der SVM an den SRC-Algorithmus mit SMG.

6.5.3 Petrovic-Datensatz mit hoher Auflösung

In Abschnitt B.2 zeigt sich, dass Auflösungen größer als 17×40 für den SRC-Algorithmus deutlich längere Laufzeiten bedeuten. Die SVM und der NN-Klassifikator können allerdings von höherer Auflösung profitieren, ohne dass sich die Laufzeit merklich verschlechtert. Deshalb wurden alle Experimente zur Untersuchung der Robustheit für die beiden konventionellen Algorithmen noch einmal mit einer Auflösung von 50×120 durchgeführt. Die Daten des SRC-Algorithmus sind die selben wie im vorigen Experiment.

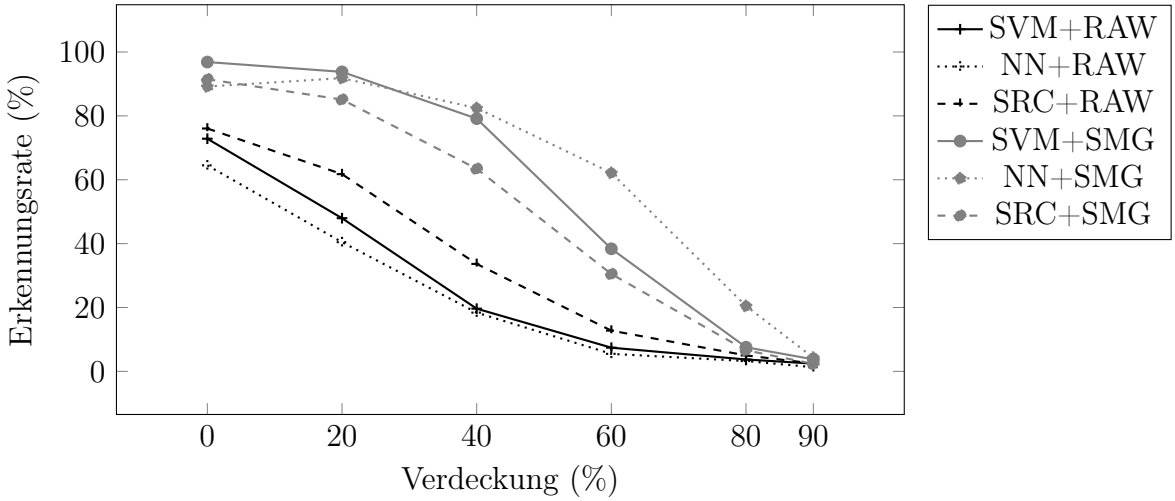


Abbildung 6.15: P_{Er} bei Verdeckungen auf dem Petrovic-Datensatz mit hoher Auflösung

Für Robustheit und Klassifikationsleistung ohne SMG zeigt sich ein ähnliches Bild wie bei den übrigen Experimenten: Der SRC-Algorithmus ist einen Tick besser als die anderen beiden Klassifikatoren. Die hohe Auflösung zusammen mit dem SMG verschafft der SVM und dem NN allerdings einen sichtbaren Vorteil. Am unempfindlichsten verhält sich hier sogar der NN-Klassifikator.

Ohne Verwendung des SMG zeigt sich das selbe Verhalten wie bei den anderen beiden Konfigurationen: Trotz der erhöhten Merkmalsdimension für SVM und NN-Klassifikator ist der SRC-Algorithmus ersteren überlegen. Auch weist er wieder etwas robusteres Verhalten gegenüber Verfälschungen auf.

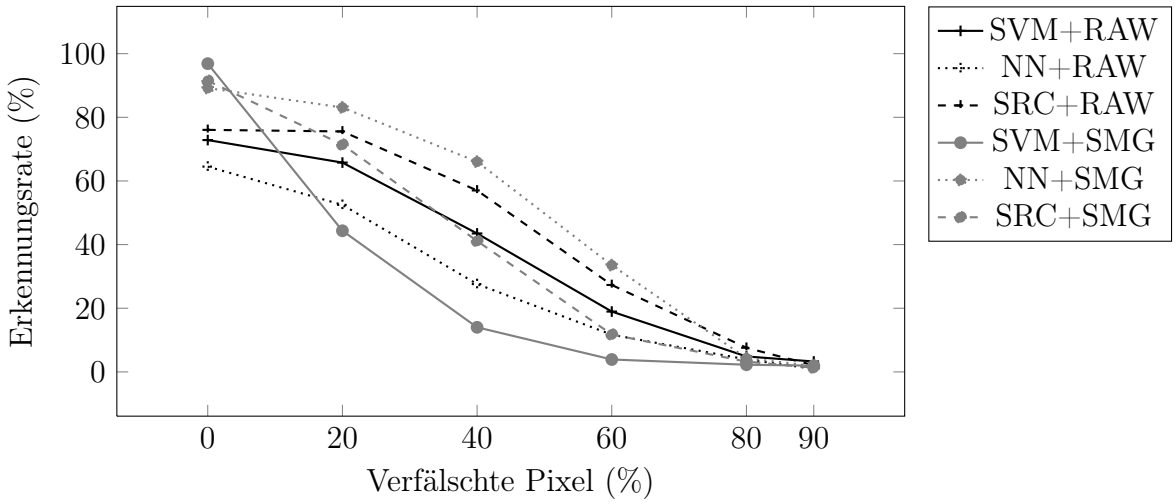


Abbildung 6.16: P_{Er} bei Salt-and-Pepper-Rauschen auf dem Petrovic-Datensatz mit hoher Auflösung

Bei Klassifizierung ohne SMG zeigen sich nur geringfügige Abweichungen der Klassifikationsleistung verglichen mit der vorherigen Konfiguration. Bei Verwendung des SMG liegt der SRC-Algorithmus zwischen SVM und dem NN-Klassifikator wobei letzterer deutlich robuster ist.

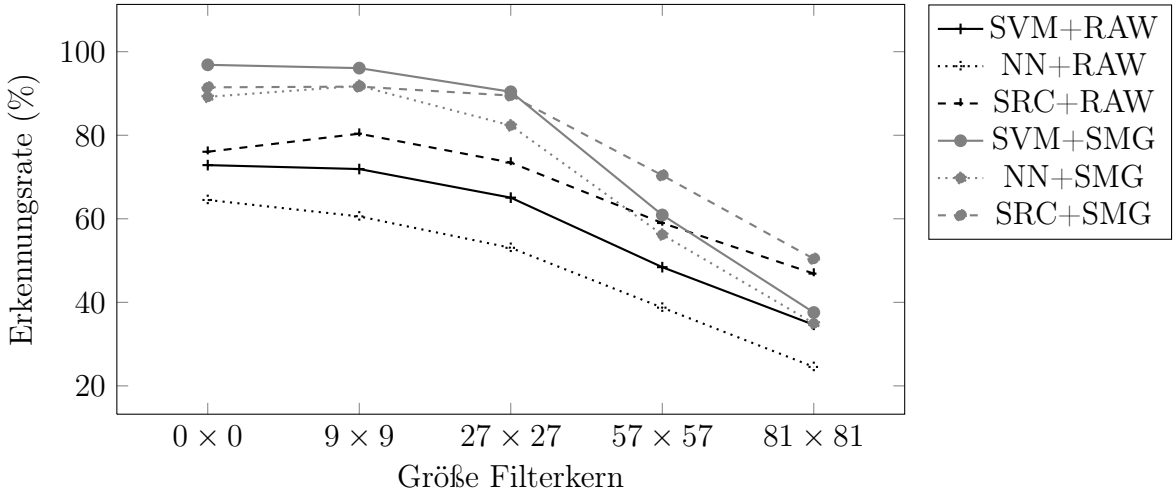


Abbildung 6.17: P_{Er} bei Unschärfe auf dem Petrovic-Datensatz mit hoher Auflösung

Verglichen mit der vorigen Konfiguration zeigt sich wieder ein ähnlicher Verlauf der Kurven falls der SMG weggelassen wird. Mit SMG liegt die SVM wieder vorne, der SRC-Algorithmus ist etwas robuster.

Die Spitze verliert der SRC-Algorithmus wieder, sobald der SMG eingesetzt wird. Dann zeigt die SVM bei fast jedem Experiment geringere Empfindlichkeit gegenüber den Verfälschungen und merklich bessere Klassifikationsleistung als der SRC-Algorithmus.

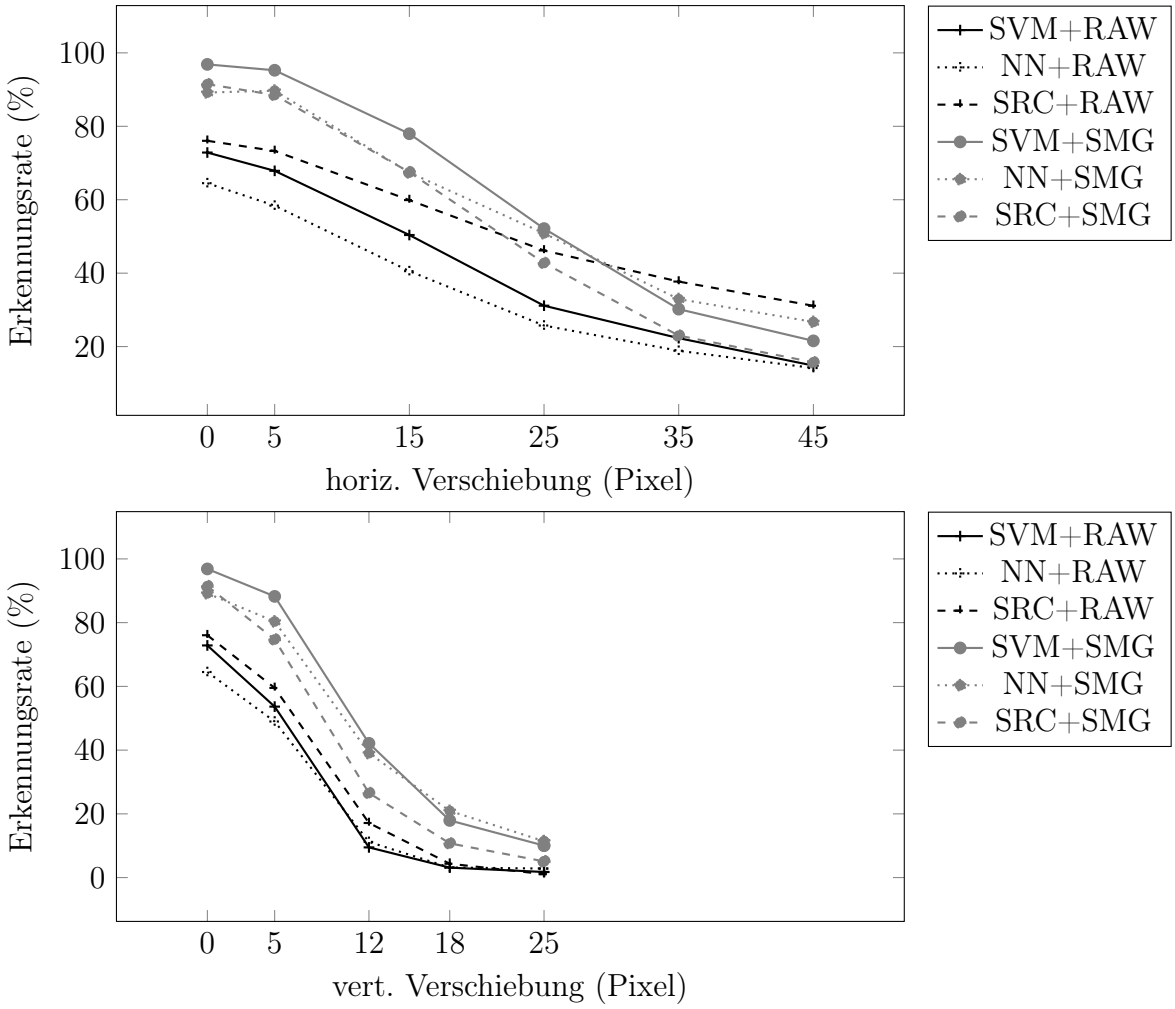


Abbildung 6.18: P_{Er} bei Verschiebungen auf dem Petrovic-Datensatz mit hoher Auflösung

Unter Verwendung des SMG setzt sich die SVM an die Spitze. Ohne bleibt der SRC-Algorithmus an erster Stelle bezüglich Klassifikationsleistung und Robustheit.

Kapitel 7

Fazit

In der vorliegenden Arbeit werden drei Klassifikationsalgorithmen und fünf Merkmalsextraktionsverfahren zur Klassifikation und Erkennung von Marke und Modell von Fahrzeugen verglichen. Dabei liefert sie einen Überblick über bildbasierte Merkmale die im Bereich der MMR verwendet wurden. Im Fokus des Vergleichs liegt der SRC-Algorithmus. Ein neuartiger Klassifikationsalgorithmus, der sich die l_1-Minimierung zunutze macht um mithilfe von spärlich besetzten Lösungen eines linearen Gleichungssystems zu klassifizieren.

Bezüglich der Merkmalsextraktionsverfahren wird ein eindeutiges Ergebnis geliefert: Der Square-Mapped-Gradient eignet sich am besten zur Klassifikation, unabhängig davon welcher Algorithmus verwendet wird. Verglichen mit den übrigen untersuchten Extraktoren ist die Berechnung dieses Merkmals auch relativ einfach.

Was den SRC-Algorithmus betrifft, ist es nicht leicht eine Aussage zu treffen. Er wurde mit der linearen SVM und dem NN-Algorithmus verglichen. Unter gleichen Vorraussetzungen (Dimension des Merkmalsvektors) ist der SRC-Algorithmus, soweit anwendbar, der SVM und dem NN stets knapp überlegen. Bei den Verfälschungsexperimenten zeigt er dabei auch überwiegend geringere Empfindlichkeit. Der Vorteil des SRC-Algorithmus zeigt sich eher wenn kein Merkmalsextraktor verwendet wird.

Berücksichtigt man die erhöhte Laufzeit und vor allen Dingen die Einschränkung der Merkmalsdimension verschwimmt der Vorteil des SRC-Algorithmus. Bei Verwendung

des Square-Mapped-Gradienten liegen SVM und NN-Algorithmus bei Klassifikationsleistung und Robustheit größtenteils vor dem SRC-Algorithmus, wenn die Merkmalsdimension für die erstere ausreichend erhöht wird. Ohne den Square-Mapped-Gradienten ist der SRC-Algorithmus den anderen beiden allerdings auch trotz niedrigerer Merkmalsdimension überlegen.
Der SRC-Algorithmus stellt eine vielversprechende Klassifikationsmethode dar, die sich zur MMR eignet. Die Funktionsweise und – bei gegebenem Optimierungsalgorithmus zur l_1-Minimierung – die Implementierung bestechen durch Einfachheit und leichte Nachvollziehbarkeit. Die Schlüsselkomponente ist der l_1-Minimierer, dessen Weiterentwicklung und Optimierung gegenstand aktueller Forschungsarbeiten ist (siehe in den Quellen von Tabelle 4.1). Der SRC-Algorithmus profitiert dabei von regem Interesse am Themengebiet des Compressed Sensing.

Ausblick

Zur Vervollständigung eines einsetzbaren MMR-Systems ist ansonsten noch ein geeigneter Fahrzeugdetektor nötig und ein Algorithmus um detektierte Fahrzeuge mit den Samples im Datensatz zu registrieren. In [Huang et al., 2008] wird gezeigt, dass die Registrieraufgabe mit dem SRC-Algorithmus kombiniert werden kann. Zur Fahrzeugdetektion bietet sich [Wagner et al., 2010] folgend die Verwendung des Viola-Jones Objektdetektors nach [Viola und Jones, 2004] an.
Im Bereich der l_1-Minimierung birgt der SRC noch einiges Potential. Die Untersuchten Algorithmen sind zwar aus numerischer sicht optimiert, die Implementierung in MATLAB allerdings lässt sicherlich noch Spielraum zu. Eine effiziente Implementierung der Minmierer in C oder unter Verwendung von CUDA zur Parallelisierung könnte das Bild doch ein Stück weit zugunsten der SRC verschieben.

Anhang A

l_1-Minimieralgorithmen

Dieser Abschnitt beinhält den Pseudocode zweier Algorithmen zur l_1-Minimierung:

1. Homotopy und
2. Augmented Lagrange Multiplier.

Sie wurden beide während der durchführung der Arbeit implementiert, jedoch wegen erhöhter Laufzeit nicht für die Durchführung der Experimente verwendet.

Bemerkungen zu A.1

Die Supportmenge $\mathcal{I}$ bezeichnet die Menge an Indizes zu Elementen des Lösungsvektors, die zur Lösung beitragen. Sie entspricht dem Support $supp(X) = \{i, x_i \in X \neq 0\}$. $\mathbf{A}_\mathcal{I}$ ist die Matrix die aus den Spaltenvektoren $\underline{\boldsymbol{a}}_i$ mit $i \in \mathcal{I}$ besteht. Analog ist $\underline{\boldsymbol{c}}_\mathcal{I}$ der Vektor mit Elementen $c_i, i \in \mathcal{I}$. In Zeile 11 werden entsprechend nur die Elemente $d_i, i \in \mathcal{I}$ des Vektors überschrieben. Beim Vektor $\underline{\boldsymbol{c}}$ in Zeilen 5 und 17 handelt es sich um den negativen Gradienten des ersten Terms der Kostenfunktion $F(\underline{\boldsymbol{x}})$ aus (4.13):

$$\underline{\boldsymbol{c}} = -\nabla 1/2||\mathbf{A}\underline{\boldsymbol{x}} - \underline{\boldsymbol{y}}||_2^2 = -\nabla 1/2(\mathbf{A}\underline{\boldsymbol{x}} - \underline{\boldsymbol{y}})^T(\mathbf{A}\underline{\boldsymbol{x}} - \underline{\boldsymbol{y}}) = -\mathbf{A}^T(\mathbf{A}\underline{\boldsymbol{x}} - \underline{\boldsymbol{y}})$$

In Zeile 11 wird die Richtung des Gradienten unter berücksichtigung nur der über $\mathcal{I}$ definierten Elemente berechnet. Die Zeilen 12-15 berechnen die Schrittlänge so, dass die Bedingung $\nabla F(\underline{\boldsymbol{x}}) = 0$ gilt.

Eingaben

1 $\mathbf{A} = [a_1, \ldots, a_n] \in \mathbb{R}^{m \times n}$

2 $\underline{\boldsymbol{y}} \in \mathbb{R}^m$ // *Testsample*

3 $\varepsilon > 0 \in \mathbb{R}$ // *Toleranz*

Initialisierung

4 $\underline{\boldsymbol{x}} \leftarrow \underline{\mathbf{0}} \in \mathbb{R}^n$; // *Lösungsvektor*

5 $\underline{\boldsymbol{c}} \leftarrow \mathbf{A}^T\underline{\boldsymbol{y}} - \mathbf{A}^T\mathbf{A}\underline{\boldsymbol{x}} \in \mathbb{R}^n$; // *Negativer Gradient der Kostenfunktion (4.13)*

6 $\lambda \leftarrow ||\mathbf{A}^T\underline{\boldsymbol{y}}||_1 \in \mathbb{R}$; // *Lagrange-Parameter*

7 $\mathcal{I} \leftarrow \{\arg\max_j \underline{\boldsymbol{a}}_j^T\underline{\boldsymbol{y}}\} \subset \mathbb{N}$; // *Supportmenge*

Algorithmus

8 `do;`

9 $\bar{\mathcal{I}} \leftarrow \{1, ..., n\} \setminus \mathcal{I}$;

/ Richtung $\underline{\boldsymbol{d}}$ bestimmen */*

10 $\underline{\boldsymbol{d}} \leftarrow \underline{\mathbf{0}} \in \mathbb{R}^n$;

11 $\underline{\boldsymbol{d}}_{\mathcal{I}} \leftarrow \mathbf{A}_{\mathcal{I}}^T\mathbf{A}_{\mathcal{I}}\underline{\boldsymbol{c}}_{\mathcal{I}}$;

/ Schrittlänge γ bestimmen */*

12 $\gamma^- \leftarrow \min_{i\in\mathcal{I}}\{-x_i/d_i\}; i^- \leftarrow \arg\min_{i\in\mathcal{I}}\{-x_i/d_i\}$;

13 $v \leftarrow \mathbf{A}_{\bar{\mathcal{I}}}\mathbf{A}_{\mathcal{I}}\underline{\boldsymbol{d}}_{\mathcal{I}}$;

14 $\gamma^+ \leftarrow \min_{i\in\bar{\mathcal{I}}}\left\{\frac{\lambda-c_i}{1-v}, \frac{\lambda+c_i}{1+v}\right\}; i^+ \leftarrow \arg\min_{i\in\bar{\mathcal{I}}}\left\{\frac{\lambda-c_i}{1-v}, \frac{\lambda+c_i}{1+v}\right\}$;

15 $\gamma \leftarrow \min\{\gamma^+, \gamma^-\}$;

/ Variablen aktualisieren */*

16 $\underline{\boldsymbol{x}} \leftarrow \underline{\boldsymbol{x}} + \gamma\underline{\boldsymbol{d}}$;

17 $\underline{\boldsymbol{c}} \leftarrow \mathbf{A}^T\underline{\boldsymbol{y}} - \mathbf{A}^T\mathbf{A}\underline{\boldsymbol{x}}$;

18 $\lambda \leftarrow \lambda - \gamma$;

19 `if` $\gamma = \gamma^+$;

20 $\mathcal{I} \leftarrow \mathcal{I} \cup \{i^+\}$;

21 `else if` $\gamma = \gamma^-$;

22 $\mathcal{I} \leftarrow \mathcal{I} \setminus \{i^-\}$;

23 `while` $(||\gamma\underline{\boldsymbol{d}}||_2 > \varepsilon)$;

Rückgabe

24 $\underline{\boldsymbol{x}}$

Abbildung A.1: Pseudocode Homotopie l_1-Minimierer

Bemerkungen zu A.2

Der in Abbildung A.2 beschriebene Algorithmus löst das in Abschnitt 4.2.2 erläuterte robuste Minimierungsproblem. Der Fehlervektor $\underline{\boldsymbol{e}}$ wird durch den Algorithmus separat berechnet, die Umformulierung wie in (4.12) entfällt. Gleichung (4.14) wird durch dem Fehlervektor ergänzt:

$$\arg\min_{\underline{\boldsymbol{x}},\underline{\boldsymbol{e}}} F(\underline{\boldsymbol{x}},\underline{\boldsymbol{e}},\underline{\boldsymbol{\lambda}}) = ||\underline{\boldsymbol{x}}||_1 + ||\underline{\boldsymbol{e}}||_1 + \underline{\boldsymbol{\lambda}}^T(\underline{\boldsymbol{y}} - \mathbf{A}\underline{\boldsymbol{x}} - \underline{\boldsymbol{e}}) + \frac{\mu}{2}||\underline{\boldsymbol{y}} - \mathbf{A}\underline{\boldsymbol{x}} - \underline{\boldsymbol{e}}||_2^2$$

Dieses Problem wird gelöst durch iteratives Lösen der folgenden Teilprobleme:

$$\underline{\boldsymbol{e}} = \arg\min_{\underline{\boldsymbol{e}}} F(\underline{\boldsymbol{x}},\underline{\boldsymbol{e}},\underline{\boldsymbol{\lambda}}) \tag{A.1}$$

$$\underline{\boldsymbol{x}} = \arg\min_{\underline{\boldsymbol{x}}} F(\underline{\boldsymbol{x}},\underline{\boldsymbol{e}},\underline{\boldsymbol{\lambda}}) \tag{A.2}$$

$$\underline{\boldsymbol{\lambda}} = \underline{\boldsymbol{\lambda}} + \mu(\underline{\boldsymbol{y}} - \mathbf{A}\underline{\boldsymbol{x}} - \underline{\boldsymbol{e}}) \tag{A.3}$$

Die in Zeilen 10 und 13 verwendete Funktion *shrink* ist wie folgt definiert:

$$\text{shrink}(\underline{\boldsymbol{x}},\alpha) = \text{sign}(\underline{\boldsymbol{x}}) \cdot \max\{|\underline{\boldsymbol{x}}| - \alpha, 0\} \tag{A.4}$$

Eingaben

1 $\mathbf{A} = [a_1, \ldots, a_n] \in \mathbb{R}^{m \times n}$

2 $\underline{\boldsymbol{y}} \in \mathbb{R}^m$ *// Testsample*

3 $\varepsilon > 0 \in \mathbb{R}$ *// Toleranz*

Initialisierung

4 $\underline{\boldsymbol{x}} \leftarrow \underline{\mathbf{0}} \in \mathbb{R}^n$; *// Lösungsvektor*

5 $\underline{\boldsymbol{e}} \leftarrow \underline{\boldsymbol{y}} \in \mathbb{R}^m$; *// Fehlervektor*

6 $\underline{\boldsymbol{\lambda}} \leftarrow \underline{\mathbf{1}} \in \mathbb{R}^m$; *// Lagrange-Multiplikator*

7 $\mu \in \mathbb{R} \leftarrow 2m/||\underline{\boldsymbol{y}}||_1$;

8 $\gamma \in \mathbb{R} \leftarrow$ Größter Eigenwert von $\mathbf{A}^T\mathbf{A}$;

Algorithmus

9 `do;`

/ Aktualiseren des Fehlervektors $\underline{\boldsymbol{e}}$, (A.1) */*

10 $\underline{\boldsymbol{e}} \leftarrow \text{shrink}\left(\underline{\boldsymbol{y}} - \mathbf{A}\underline{\boldsymbol{x}} + \frac{\underline{\boldsymbol{\lambda}}}{\mu}, \frac{1}{\mu}\right)$; *// siehe (A.4)*

/ Aktualisieren des Lösungsvektors $\underline{\boldsymbol{x}}$, (A.2) */*

11 $\underline{\boldsymbol{z}} \leftarrow \underline{\boldsymbol{x}}, \underline{\boldsymbol{w}} \leftarrow \underline{\boldsymbol{x}}, t \leftarrow 1$;

12 `do;`

13 $\underline{\boldsymbol{w}}^* \leftarrow \text{shrink}\left(\underline{\boldsymbol{z}} + \frac{1}{\gamma}\mathbf{A}^T\left(\underline{\boldsymbol{y}} - \mathbf{A}\underline{\boldsymbol{z}} - \underline{\boldsymbol{e}} + \frac{1}{\mu}\underline{\boldsymbol{\lambda}}\right), \frac{1}{\mu\gamma}\right)$;

14 $t^* \leftarrow \frac{1}{2}\left(1 + \sqrt{1 + 4t^2}\right)$;

15 $\underline{\boldsymbol{z}} \leftarrow \underline{\boldsymbol{w}} + \frac{t-1}{t^*}(\underline{\boldsymbol{w}}^* - \underline{\boldsymbol{w}})$;

16 $\underline{\boldsymbol{w}} \leftarrow \underline{\boldsymbol{w}}^*, t \leftarrow t^*$;

17 `while (` $\frac{||\underline{\boldsymbol{w}}^* - \underline{\boldsymbol{w}}||_2}{||\underline{\boldsymbol{w}}||_2} > \varepsilon$ `);`

18 $\Delta\underline{\boldsymbol{x}} \leftarrow \underline{\boldsymbol{x}} - \underline{\boldsymbol{w}}$;

19 $\underline{\boldsymbol{x}} \leftarrow \underline{\boldsymbol{w}}$;

/ Aktualisieren des Lagrange-Multiplikators $\underline{\boldsymbol{\lambda}}$, (A.3) */*

20 $\underline{\boldsymbol{\lambda}} \leftarrow \underline{\boldsymbol{\lambda}} + \mu(\underline{\boldsymbol{y}} - \mathbf{A}\underline{\boldsymbol{x}} - \underline{\boldsymbol{e}})$;

21 `while (`$||\Delta\underline{\boldsymbol{x}}||_2 > \varepsilon$`);`

Rückgabe

22 $\underline{\boldsymbol{x}}, \underline{\boldsymbol{e}}$

Abbildung A.2: Pseudocode ALM l_1-Minimierer

Anhang B

Tabellen

Auf den folgenden Seiten befinden sich die Werte, denen die Diagramme in Kapitel 6 zugrundeliegen. Die meisten Werte sind Mittelwerte, die durch Cross-Validation entstanden sind. Wo dies zutrifft gibt der Wert in Klammern die Standardabweichung (σ) an. Die Werte die mit *NaN* ausgezeichnet sind, wurden aufgrund überlanger Berechnungsdauer nicht berücksichtigt.

B.1 Vergleich der Merkmalsextraktoren

Tabelle B.1: P_{Er} in % für verschiedene Merkmale

Feature	SVM	(σ)	NN	(σ)	SRC	(σ)
RAW	71.86	(1.98)	64.14	(4.34)	76.64	(2.07)
GAB	80.15	(0.95)	81.55	(1.87)	84.27	(1.85)
LE	78.09	(0.91)	74.49	(2.01)	84.74	(1.93)
PC	81.51	(0.96)	79.17	(1.7)	87.31	(1.67)
SOB	79.78	(1.8)	76.87	(0.93)	87.08	(0.4)
SMG	92.09	(0.37)	85.39	(0.41)	92.42	(0.34)

B.2 Laufzeit

Tabelle B.2: P_{Er} in % für SRC für verschiedene Merkmalsdimensionen mit simplem Resampling (RAW)

Dimension	SVM	(σ)	NN	(σ)	SRC	(σ)
9×20	64.79	(0.35)	54.49	(1.7)	66.81	(2.03)
17×40	71.4	(2.06)	61.1	(1.44)	77.48	(0.18)
25×60	72.94	(0.54)	63.3	(1.83)	80.01	(1.33)
50×120	72.85	(0.73)	64.51	(1.3)	NaN	(NaN)

Tabelle B.3: P_{Er} in % für SRC für verschiedene Merkmalsdimensionen mit Square-Mapped-Gradienten (SMG)

Dimension	SVM	(σ)	NN	(σ)	SRC	(σ)
9×20	69.34	(1.64)	61.38	(0.6)	69.71	(1.4)
17×40	92.37	(0.13)	84.88	(2.19)	91.71	(1.13)
25×60	94.99	(0.24)	88.2	(1.15)	95.13	(1.09)
50×120	96.86	(0.63)	89.19	(1.13)	NaN	(NaN)

Tabelle B.4: Laufzeit in s für SRC für verschiedene l_1-Minimierer mit simplem Resampling (RAW)

Dimension	DALM	(σ)	Homotopy	(σ)	l_1-LS	(σ)
9×20	$6 \cdot 10^{-2}$	($3 \cdot 10^{-2}$)	0.1	($2 \cdot 10^{-2}$)	0.3	($3 \cdot 10^{-2}$)
17×40	0.5	(0.13)	0.28	($7 \cdot 10^{-2}$)	2.45	(0.47)
25×60	2.55	(0.34)	0.58	(0.11)	8.8	(1.84)
50×120	109.56	(10.54)	5.79	(0.23)	88.93	(10.48)

Tabelle B.5: Laufzeit in s für SRC für verschiedene l_1-Minimierer mit Square-Mapped-Gradienten (SMG)

Dimension	DALM	(σ)	Homotopy	(σ)	l_1-LS	(σ)
9×20	0.1	(0.1)	$9 \cdot 10^{-2}$	$(5 \cdot 10^{-2})$	0.45	$(5 \cdot 10^{-2})$
17×40	1.09	$(3 \cdot 10^{-2})$	0.39	$(3 \cdot 10^{-2})$	3.08	(0.11)
25×60	9.71	(0.12)	1.61	(0.12)	11.55	(0.21)
50×120	NaN	(NaN)	NaN	(NaN)	NaN	(NaN)

Tabelle B.6: Laufzeit in s für SRC in Abhängigkeit der Datenbankgröße mit simplem Resampling (RAW)

Klassen	DALM	(σ)	Homotopy	(σ)	l_1-LS	(σ)
15	0.33	$(6 \cdot 10^{-2})$	0.17	$(2 \cdot 10^{-2})$	1.13	(0.13)
25	0.41	$(6 \cdot 10^{-2})$	0.22	$(2 \cdot 10^{-2})$	1.53	(0.19)
35	0.37	$(4 \cdot 10^{-2})$	0.26	$(1 \cdot 10^{-2})$	1.51	(0.13)
45	0.44	$(3 \cdot 10^{-2})$	0.33	$(2 \cdot 10^{-2})$	1.77	(0.17)
52	0.62	$(8 \cdot 10^{-2})$	0.46	$(5 \cdot 10^{-2})$	2.17	(0.39)

Tabelle B.7: Laufzeit in s für SRC in Abhängigkeit der Datenbankgröße mit Square-Mapped-Gradienten (SMG)

Klassen	DALM	(σ)	Homotopy	(σ)	l_1-LS	(σ)
15	1.09	$(9 \cdot 10^{-2})$	0.32	$(5 \cdot 10^{-2})$	2.41	(0.15)
25	1.09	$(3 \cdot 10^{-2})$	0.32	$(1 \cdot 10^{-2})$	2.66	(0.19)
35	1.08	$(2 \cdot 10^{-2})$	0.36	$(3 \cdot 10^{-2})$	2.76	(0.22)
45	1.12	$(7 \cdot 10^{-2})$	0.37	$(4 \cdot 10^{-2})$	3.18	$(6 \cdot 10^{-2})$
52	1.1	$(3 \cdot 10^{-2})$	0.4	$(2 \cdot 10^{-2})$	3.37	(0.12)

B.3 Robustheit gegenüber Verfälschungen

Eigener Datensatz

Tabelle B.8: P_{Er} in % bei Verdeckung mit simplem Resampling (RAW) auf eigenem Datensatz

Verdeckung (%)	SVM	NN	SRC
0	86.74	90.35	95.86
20	68.99	63.3	89.64
40	43.15	41.99	64.01
60	27.36	25.4	39.23
80	14.01	12.5	22.69
90	10.36	10.99	6.63

Tabelle B.9: P_{Er} in % bei Salt-and-Pepper-Rauschen mit simplem Resampling (RAW) auf eigenem Datensatz

Verfälschung (%)	SVM	NN	SRC
0	86.74	90.35	95.86
20	81.36	85.99	94.75
40	66.1	70.24	75.13
60	37.94	37.37	45.11
80	24.42	25.8	6.41
90	13.57	14.64	6.23

Tabelle B.10: P_{Er} in % bei Unschärfe mit simplem Resampling (RAW) auf eigenem Datensatz

Filterkern	SVM	NN	SRC
0×0	86.74	90.35	95.86
9×9	86.39	89.64	96
27×27	79.4	86.03	93.95
57×57	75.4	75.49	90.12
81×81	65.88	70.28	78.56

Tabelle B.11: P_{Er} in % bei horizontaler (links) und vertikaler (rechts) Verschiebung mit simplem Resampling (RAW) auf eigenem Datensatz

Pixel	SVM	NN	SRC	Pixel	SVM	NN	SRC
0	86.74	90.35	95.86	0	86.74	90.35	95.86
5	83.72	90.66	94.04	5	64.46	71.57	79.18
15	75.49	82.07	91.73	25	12.99	15.93	22.78
25	73.89	74.87	85.99	45	3.6	4.09	5.47
35	71.17	68.33	83.54	65	3.34	6.54	6.27
45	59.79	57.83	75.53				

Petrovic-Datensatz

Tabelle B.12: P_{Er} in % bei Verdeckung mit simplem Resampling (RAW) auf dem Petrovic-Datensatz

Verdeckung (%)	SVM	(σ)	NN	(σ)	SRC	(σ)
0	72.33	(0.46)	63.38	(1.34)	76.05	(1.46)
20	44.87	(1.13)	38.31	(1.25)	61.76	(0.99)
40	17.87	(1.7)	16.01	(0.48)	33.64	(0.94)
60	5.69	(0.87)	4.14	(0.62)	12.75	(0.49)
80	3.72	(0.31)	2.84	(0.32)	4.99	(0.1)
90	2.35	(0.44)	1.54	(0.43)	2.46	(0.71)

Tabelle B.13: P_{Er} in % bei Verdeckung mit Square-Mapped-Gradienten (SMG) auf dem Petrovic-Datensatz

Verdeckung (%)	SVM	(σ)	NN	(σ)	SRC	(σ)
0	91.61	(1.03)	84.94	(2.16)	91.47	(2.29)
20	79.74	(1.08)	77.39	(0.79)	85.11	(1.55)
40	50.98	(1.79)	55.23	(0.46)	63.45	(0.67)
60	19.63	(2.53)	25.84	(1.47)	30.48	(1.78)
80	4.18	(0.78)	4.95	(0.53)	6.6	(0.1)
90	2.49	(0.29)	2.39	(0.64)	2.35	(0.81)

Tabelle B.14: P_{Er} in % bei Salt-and-Pepper-Rauschen mit simplem Resampling (RAW) auf dem Petrovic-Datensatz

Verfälschung (%)	SVM	(σ)	NN	(σ)	SRC	(σ)
0	72.33	(0.46)	63.38	(1.34)	76.05	(1.46)
20	64.64	(1.85)	53.13	(1.21)	75.56	(1.77)
40	36.9	(3.54)	25.88	(1.85)	57.02	(1.34)
60	17.52	(3.16)	11.62	(1.33)	27.32	(1.53)
80	4.85	(0.46)	4.14	(1.04)	7.58	(0.6)
90	2.77	(0.95)	3.16	(1.47)	2.07	(1.32)

Tabelle B.15: P_{Er} in % bei Salt-and-Pepper-Rauschen mit Square-Mapped-Gradienten (SMG) auf dem Petrovic-Datensatz

Verfälschung (%)	SVM	(σ)	NN	(σ)	SRC	(σ)
0	91.61	(1.03)	84.94	(2.16)	91.47	(2.29)
20	56.85	(0.83)	62.08	(2.34)	71.42	(1.01)
40	25.67	(2.11)	37.71	(1.11)	41.15	(1.07)
60	7.72	(1.05)	12.39	(0.75)	11.76	(0.46)
80	3.37	(1.03)	4.28	(0.21)	3.23	(0.59)
90	1.69	(0.57)	2.25	(0.26)	1.3	(0.3)

Tabelle B.16: P_{Er} in % bei Unschärfe mit simplem Resampling (RAW) auf dem Petrovic-Datensatz

Filterkern	SVM	(σ)	NN	(σ)	SRC	(σ)
0×0	72.33	(0.46)	63.38	(1.34)	76.05	(1.46)
9×9	72.82	(1.48)	61.41	(1.16)	80.37	(1.41)
27×27	64.64	(2.12)	54	(1.67)	73.46	(0.79)
57×57	47.37	(1.27)	37.04	(1.95)	58.99	(1.37)
81×81	32.09	(2.04)	24.72	(2.4)	46.88	(2.06)

Tabelle B.17: P_{Er} in % bei Unschärfe mit Square-Mapped-Gradienten (SMG) auf dem Petrovic-Datensatz

Filterkern	SVM	(σ)	NN	(σ)	SRC	(σ)
0×0	91.61	(1.03)	84.94	(2.16)	91.47	(2.29)
9×9	92.06	(0.76)	84.45	(1.39)	91.64	(0.43)
27×27	88.17	(0.61)	81.53	(2.44)	89.5	(0.75)
57×57	67.94	(1.79)	60.85	(1.28)	70.4	(0.87)
81×81	46.07	(1.07)	41.4	(2.12)	50.42	(1.32)

Tabelle B.18: P_{Er} in % bei horizontaler Verschiebung mit simplem Resampling (RAW) auf dem Petrovic-Datensatz

Verschiebung (Pixel)	SVM	(σ)	NN	(σ)	SRC	(σ)
0	72.33	(0.46)	63.38	(1.34)	76.05	(1.46)
5	68.22	(1.34)	59.52	(0.69)	73.28	(1.37)
15	52.11	(1.74)	44.38	(2.52)	59.9	(1.06)
25	37.18	(2.03)	31.78	(1.8)	46.17	(1.3)
35	26.97	(0.68)	24.75	(1.46)	37.71	(1.07)
45	19.87	(0.79)	18.33	(1.09)	31.11	(1.95)

Tabelle B.19: P_{Er} in % bei horizontaler Verschiebung mit Square-Mapped-Gradienten (SMG) auf dem Petrovic-Datensatz

Verschiebung (Pixel)	SVM	(σ)	NN	(σ)	SRC	(σ)
0	91.61	(1.03)	84.94	(2.16)	91.47	(2.29)
5	86.73	(1.5)	83.57	(0.6)	88.52	(0.94)
15	65.63	(1.01)	59.34	(0.66)	67.49	(0.58)
25	39.68	(1.76)	35.01	(1)	42.77	(0.69)
35	22.86	(2.13)	20.54	(1.1)	23	(1.38)
45	16.5	(2.25)	15.45	(1.28)	15.73	(0.99)

Tabelle B.20: P_{Er} in % bei vertikaler Verschiebung mit simplem Resampling (RAW) auf dem Petrovic-Datensatz

Verschiebung (Pixel)	SVM	(σ)	NN	(σ)	SRC	(σ)
0	72.33	(0.46)	63.38	(1.34)	76.05	(1.46)
5	50.14	(0.95)	46.66	(2.28)	59.52	(2.26)
12	9.3	(0.96)	11.59	(2.21)	17.13	(1.74)
18	2.56	(0.92)	3.09	(0.96)	4.35	(0.41)
25	1.4	(0.25)	0.98	(0.33)	1.09	(0.27)

Tabelle B.21: P_{Er} in % bei vertikaler Verschiebung mit Square-Mapped-Gradienten (SMG) auf dem Petrovic-Datensatz

Verschiebung (Pixel)	SVM	(σ)	NN	(σ)	SRC	(σ)
0	91.61	(1.03)	84.94	(2.16)	91.47	(2.29)
5	71.1	(0.46)	62.11	(3.04)	74.72	(1.16)
12	25.6	(1.04)	25.11	(0.47)	26.54	(2.13)
18	12.29	(0.88)	10.6	(0.29)	10.74	(0.73)
25	5.02	(0.52)	5.44	(0.96)	5.09	(0.41)

Petrovic-Datensatz mit hoher Auflösung

Tabelle B.22: P_{Er} in % bei Verdeckung mit simplem Resampling (RAW) auf dem Petrovic-Datensatz mit hoher Auflösung

Verdeckung (%)	SVM	(σ)	NN	(σ)	SRC	(σ)
0	72.85	(0.73)	64.51	(1.3)	76.05	(1.46)
20	47.96	(1.63)	40.45	(0.67)	61.76	(0.99)
40	19.63	(0.87)	18.4	(1.39)	33.64	(0.94)
60	7.41	(0.52)	5.48	(0.7)	12.75	(0.49)
80	3.72	(0.38)	3.23	(0.34)	4.99	(0.1)
90	2.42	(0.62)	1.3	(0.47)	2.46	(0.71)

Tabelle B.23: P_{Er} in % bei Verdeckung mit Square-Mapped-Gradienten (SMG) auf dem Petrovic-Datensatz mit hoher Auflösung

Verdeckung (%)	SVM	(σ)	NN	(σ)	SRC	(σ)
0	96.86	(0.63)	89.19	(1.13)	91.47	(2.29)
20	93.79	(0.83)	91.82	(0.87)	85.11	(1.55)
40	79.14	(1)	82.44	(1.81)	63.45	(0.67)
60	38.38	(2.22)	62.15	(2.65)	30.48	(1.78)
80	7.55	(0.9)	20.47	(0.96)	6.6	(0.1)
90	3.76	(1.45)	4.28	(0.97)	2.35	(0.81)

Tabelle B.24: P_{Er} in % bei Salt-and-Pepper-Rauschen mit simplem Resampling (RAW) auf dem Petrovic-Datensatz mit hoher Auflösung

Verfälschung (%)	SVM	(σ)	NN	(σ)	SRC	(σ)
0	72.85	(0.73)	64.51	(1.3)	76.05	(1.46)
20	65.77	(2.55)	52.49	(0.52)	75.56	(1.77)
40	43.5	(3.41)	27.6	(2.57)	57.02	(1.34)
60	18.96	(3.39)	11.69	(1.62)	27.32	(1.53)
80	4.85	(1.27)	3.93	(1.71)	7.58	(0.6)
90	3.27	(0.3)	0.95	(0.18)	2.07	(1.32)

Tabelle B.25: P_{Er} in % bei Salt-and-Pepper-Rauschen mit Square-Mapped-Gradienten (SMG) auf dem Petrovic-Datensatz mit hoher Auflösung

Verfälschung (%)	SVM	(σ)	NN	(σ)	SRC	(σ)
0	96.86	(0.63)	89.19	(1.13)	91.47	(2.29)
20	44.35	(6.51)	83.08	(1.16)	71.42	(1.01)
40	14.01	(4.13)	66.05	(1.7)	41.15	(1.07)
60	3.9	(0.5)	33.53	(1.19)	11.76	(0.46)
80	2.25	(0.17)	4.32	(0.25)	3.23	(0.59)
90	1.97	(0.1)	1.86	(0.35)	1.3	(0.3)

Tabelle B.26: P_{Er} in % bei Unschärfe mit simplem Resampling (RAW) auf dem Petrovic-Datensatz mit hoher Auflösung

Filterkern	SVM	(σ)	NN	(σ)	SRC	(σ)
0×0	72.85	(0.73)	64.51	(1.3)	76.05	(1.46)
9×9	71.91	(1.45)	60.57	(0.4)	80.37	(1.41)
27×27	65.06	(0.82)	53.05	(1.52)	73.46	(0.79)
57×57	48.42	(1.77)	38.73	(1.53)	58.99	(1.37)
81×81	34.62	(1.65)	24.54	(1.37)	46.88	(2.06)

Tabelle B.27: P_{Er} in % bei Unschärfe mit Square-Mapped-Gradienten (SMG) auf dem Petrovic-Datensatz mit hoher Auflösung

Filterkern	SVM	(σ)	NN	(σ)	SRC	(σ)
0×0	96.86	(0.63)	89.19	(1.13)	91.47	(2.29)
9×9	96.07	(0.46)	91.82	(2.02)	91.64	(0.43)
27×27	90.41	(0.82)	82.34	(2.72)	89.5	(0.75)
57×57	60.92	(0.99)	56.21	(0.82)	70.4	(0.87)
81×81	37.57	(1.16)	34.97	(0.65)	50.42	(1.32)

Tabelle B.28: P_{Er} in % bei horizontaler Verschiebung mit simplem Resampling (RAW) auf dem Petrovic-Datensatz mit hoher Auflösung

Verschiebung (Pixel)	SVM	(σ)	NN	(σ)	SRC	(σ)
0	72.85	(0.73)	64.51	(1.3)	76.05	(1.46)
5	67.87	(1.4)	58.39	(1.13)	73.28	(1.37)
15	50.39	(1.14)	40.55	(2)	59.9	(1.06)
25	31.14	(3.02)	25.74	(2.29)	46.17	(1.3)
35	22.3	(0.95)	18.86	(0.7)	37.71	(1.07)
45	14.82	(2.58)	14.15	(0.93)	31.11	(1.95)

Tabelle B.29: P_{Er} in % bei horizontaler Verschiebung mit Square-Mapped-Gradienten (SMG) auf dem Petrovic-Datensatz mit hoher Auflösung

Verschiebung (Pixel)	SVM	(σ)	NN	(σ)	SRC	(σ)
0	96.86	(0.63)	89.19	(1.13)	91.47	(2.29)
5	95.26	(0.58)	89.71	(1.63)	88.52	(0.94)
15	77.98	(1.4)	67.42	(1.24)	67.49	(0.58)
25	52.14	(1.42)	50.84	(1.89)	42.77	(0.69)
35	30.2	(2.3)	32.94	(1.16)	23	(1.38)
45	21.56	(1.18)	26.72	(0.32)	15.73	(0.99)

Tabelle B.30: P_{Er} in % bei vertikaler Verschiebung mit simplem Resampling (RAW) auf dem Petrovic-Datensatz mit hoher Auflösung

Verschiebung (Pixel)	SVM	(σ)	NN	(σ)	SRC	(σ)
0	72.85	(0.73)	64.51	(1.3)	76.05	(1.46)
5	53.65	(0.84)	48.98	(2.36)	59.52	(2.26)
12	9.48	(0.42)	11.2	(1.41)	17.13	(1.74)
18	3.13	(0.68)	3.16	(0.38)	4.35	(0.41)
25	1.76	(0.49)	2.84	(0.63)	1.09	(0.27)

Tabelle B.31: P_{Er} in % bei vertikaler Verschiebung mit Square-Mapped-Gradienten (SMG) auf dem Petrovic-Datensatz mit hoher Auflösung

Verschiebung (Pixel)	SVM	(σ)	NN	(σ)	SRC	(σ)
0	96.86	(0.63)	89.19	(1.13)	91.47	(2.29)
5	88.24	(1.49)	80.37	(3.69)	74.72	(1.16)
12	42.13	(1.85)	39.04	(1.3)	26.54	(2.13)
18	17.94	(2.57)	20.93	(1.48)	10.74	(0.73)
25	10.04	(1.02)	11.41	(0.6)	5.09	(0.41)

Anhang C

Quellenangaben

Konferenzen							
Jahr	01	04	05	07	08	09	10
IAPR-ANNPR - IAPR Workshop on Artificial Neural Networks in Pattern Recognition	0	0	0	0	1	0	0
CVPR - IEEE Computer Society Conference on Computer Vision and Pattern Recognition	1	0	0	0	1	1	0
ITNG - International Conference on Information Technology: New Generations	0	0	0	1	0	0	0
AICS - Irish Conference on Artificial Intelligence and Cognitive Science	0	0	1	0	0	0	0
BMVC - British Machine Vision Conference	0	1	0	0	0	0	0
FIT - International Conference on Frontiers of Information Technology	0	0	0	0	0	1	0
ICIP - International Conference on Image Processing	0	0	0	0	0	0	1
SPIE - Symposium on Electronic Imaging	0	0	0	0	0	1	0
Eintrag: Anzahl der relevanten Beiträge							

Zeitschriften								
Jahr	96	99	06	07	08	09	10	11
TIT - IEEE Transactions on Information Theory	0	0	2	0	1	0	0	0
JSTSP - IEEE Journal of Selected Topics in Signal Processing	0	0	0	1	0	0	1	0
JCVR - Videre: Journal of Computer Vision Research	0	1	0	0	0	0	0	0
TPAMI - IEEE Transactions on Pattern Analysis and Machine Intelligence	1	0	0	0	0	1	0	1
Eintrag: Anzahl der relevanten Beiträge								

Webressourcen				
Peter Kovesi	*	Functions for Computer Vision	*	04.2011
http://www.csse.uwa.edu.au/~pk/research/matlabfns				
Allen Y. Yang	*	Fast l-1 Minimization Algorithms	*	04.2011
http://www.eecs.berkeley.edu/~yang/software/l1benchmark/				
Igor Carron	*	Compressive Sensing: The Big Picture	*	03.2011
http://sites.google.com/site/igorcarron2/cs				
N. Petkov	*	Gabor filter for image processing and computer vision	*	04.2011
http://matlabserver.cs.rug.nl/edgedetectionweb/web/index.html				

Abkürzungsverzeichnis

ALM Augmented Lagrangian Multiplicator

DALM Dual Augmented Lagrangian Multiplicator

GLS Gleichungssystem

LESH Local Energy based Shape Histogram

MMR Make and Model Recognition

NN Nearest Neighbour Klassifikation

PCA Principal Component Analysis

SIFT Scale Invariant Feature Transform

SRC Sparse Representation Classification

SVM Support-Vector-Machine

Abbildungsverzeichnis

Literaturverzeichnis

[Baraniuk et al., 2011] Baraniuk, R., M. Davenport, M. Duarte und C. Hegde (2011). *An Introduction to Compressive Sensing*. Connexions Web site. http://cnx.org/content/col11133/1.5/.

[Boyd und Vandenberghe, 2004] Boyd, S. P. und L. Vandenberghe (2004). *Convex Optimization*. Cambridge University Press.

[Candes et al., 2006] Candes, E. J., J. Romberg und T. Tao (2006). *Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information*. Information Theory, IEEE Transactions on, 52(2):489–509.

[Chang und Lin, 2011] Chang, Chih-Chung und C.-J. Lin (2011). *LIBSVM: A library for support vector machines*. ACM Transactions on Intelligent Systems and Technology, 2(3):27–1. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[Clady et al., 2008] Clady, X., P. Negri, M. Milgram und R. Poulenard (2008). *Multi-class vehicle type recognition system*. Artificial Neural Networks in Pattern Recognition, S. 228–239.

[Cootes und Taylor, 2001] Cootes, T.F. und C. Taylor (2001). *On representing edge structure for model matching*. In: *Proceedings of the 2001 IEEE Computer Society, Conference on*, S. 1114–1119.

[Dlagnekov und Belongie, 2005] Dlagnekov, L. und S. Belongie (2005). *Recognizing Cars*. Technischer Bericht CS2005-0833, University of California San Diego, CSE.

[Dlagnekov, 2005] Dlagnekov, Louka (2005). *Video-based Car Surveillance: License Plate, Make, and Model Recognition*. Diplomarbeit, University of California, San Diego.

[Donoho, 2006] Donoho, D. L. (2006). *Compressed Sensing*. Information Theory, IEEE Transactions on, 52(4):1289–1306.

[Donoho und Tsaig, 2008] Donoho, D. L. und Y. Tsaig (2008). *Fast Solution of l_1-Norm Minimization Problems When the Solution May Be Sparse*. Information Theory, IEEE Transactions on, 54(11):4789–4812.

[Figueiredo et al., 2007] Figueiredo, M. A. T., R. D. Nowak und S. J. Wright (2007). *Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems*. Selected Topics in Signal Processing, IEEE Journal of, 1(4):586–597.

[Fornasier und Rauhut, 2010] Fornasier, M. und H. Rauhut (2010). *Compressive Sensing*, Bd. 1 d. Reihe *Handbook of Mathematical Methods in Imaging*, Kap. 6, S. 187–228. Springer Verlag.

[Huang et al., 2008] Huang, Junzhou, X. Huang und D. Metaxas (2008). *Simultaneous image transformation and sparse representation recovery*. Computer Vision and Pattern Recognition, IEEE Computer Society Conference on, 0:1–8.

[Kazemi et al., 2007] Kazemi, Farhad Mohamad, S. Samadi und H. R. Poorreza (2007). *Vehicle Recognition Using Curvelet Transform and SVM*. Fifth International Conference on Information Technology: New Generations, S. 515–521.

[Kim et al., 2007] Kim, S. J., K. Koh, M. Lustig, S. Boyd und D. Gorinevsky (2007). *An interior-point method for large-scale l1-regularized least squares*. Selected Topics in Signal Processing, IEEE Journal of, 1(4):606–617.

[Kovesi, 1999] Kovesi, P. (1999). *Image features from phase congruency*. Videre: Journal of Computer Vision Research, 1(3):1–26.

[Kovesi, 2011] Kovesi, P. D. (2011). *MATLAB and Octave Functions for Computer Vision and Image Processing*. online. Available from: <http://www.csse.uwa.edu.au/∼pk/research/matlabfns/>.

[Kyrki, 2002] Kyrki, Ville (2002). *Local and Global Feature Extraction for Invariant Object Recognition*. Doktorarbeit, Lappeenranta University of Technology, Lappeenranta, Finland.

[Lee, 1996] Lee, T. S. (1996). *Image representation using 2D Gabor wavelets*. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 18(10):959–971.

[Lowe, 2004] Lowe, D. G. (2004). *Distinctive image features from scale-invariant keypoints*. International Journal of Computer Vision, 60(2):91–110.

[Mei und Ling, 2011] Mei, Xue und H. Ling (2011). *Robust Visual Tracking and Vehicle Classification via Sparse Representation*. Pattern Analysis and Machine Intelligence (PAMI), IEEE Trans. on, (to appear).

[Movellan, 2002] Movellan, J. R. (2002). *Tutorial on gabor filters*. Open Source Document http://mplab.ucsd.edu/wordpress/?page_id=75.

[Munroe und Madden, 2005] Munroe, D. T. und M. G. Madden (2005). *Multi-class and single-class classification approaches to vehicle model recognition from images*. Proceedings of IEEE AICS.

[Petrovic und Cootes, 2004] Petrovic, V. S. und T. F. Cootes (2004). *Analysis of features for rigid structure vehicle type recognition*. 2:587–596.

[Sarfraz et al., 2009] Sarfraz, M. S., A. Saeed, M. H. Khan und Z. Riaz (2009). *Bayesian prior models for vehicle make and model recognition*. In: *Proceedings of the 6th International Conference on Frontiers of Information Technology*, S. 1–6. ACM.

[Viola und Jones, 2004] Viola, P. und M. J. Jones (2004). *Robust Real-Time Face Detection*. International Journal of Computer Vision, 57(2):137–154.

[Wagner et al., 2010] Wagner, A., J. Wright, A. Ganesh, Z. Zhou und Y. Ma (2010). *Towards a practical face recognition system: robust registration and illumination by sparse representation*. S. 597–604.

[Wright et al., 2008] Wright, J., A. Y. Yang, A. Ganesh, S. S. Sastry und Y. Ma (2008). *Robust face recognition via sparse representation*. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 31(2):210–227.

[Yang et al., 2010] Yang, A. Y., S. S. Sastry, A. Ganesh und Y. Ma (2010). *Fast l1-minimization algorithms and an application in robust face recognition: A review*. In: *Image Processing (ICIP), 2010 17th IEEE International Conference on*, S. 1849–1852.

[Zafar et al., 2009] Zafar, I., E. A. Edirisinghe und B. S. Acar (2009). *Localized contourlet features in vehicle make and model recognition*. In: *Proceedings of SPIE*, Bd. 7251, S. 725105–1–725105–9.