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

Objektklassifizierung anhand der Modalität Textur

Bachelorarbeit zur Erlangung des akademischen Grades Bachelor of Science

Daniel Maximilian Hundsdörfer

Betreuer: Dipl.-Ing. Richard Bormann, M. Sc., Fraunhofer Institut für Produktionstechnik und Automatisierung
Prof. Dr. H.-M. Groß, FG Neuroinformatik und Kognitive Robotik
Verantwortlicher Hochschullehrer:
Prof. Dr. H.-M. Groß, FG Neuroinformatik und Kognitive Robotik

Die Bachelorarbeit wurde am 26.11.2014 bei der Fakultät für Informatik und Automatisierung der Technischen Universität Ilmenau eingereicht.

urn:nbn:de:gbv:ilm1-2015200021

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, 26.11.2014 .....

Daniel Maximilian Hundsdörfer

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Motivation

Die Servicerobotik hat das Potential, in naher Zukunft eine Vielzahl von Dienstleistungen in den unterschiedlichsten Bereichen anzubieten. Dazu muss sich ein Serviceroboter sicher in einer häuslichen Umgebung bewegen sowie Manipulationen und Interaktionen durchführen können. Bei der Navigation unterscheidet ein Roboter innerhalb einer kartografierten Umgebung zwischen befahrbaren und nicht befahrbaren Regionen und bewegt sich mit Bezug zu den Raumkoordinaten [DeSouza und Kak, 2002]. Ebenfalls ist es möglich, während der Navigation eine Karte zu erstellen, diese zu aktualisieren und Hindernissen auszuweichen [Thrun et al., 2002]. Diese Art der Navigation ist allerdings nicht intuitiv und somit für Menschen, die nicht mit Robotersystemen vertraut sind, schwer nachvollziehbar. Ein Intuitiver Ansatz ist die funktionale Navigation [DeSouza und Kak, 2002]. Diese ordnet Gegenständen eine Bedeutung zu, sodass gesprochene Befehle wie "Bringe mir das Telefon aus der Küche." interpretiert und ausgeführt werden können. Die dafür benötigte maschinelle Wahrnehmung ist allerdings noch nicht im Stande, eine häuslicher Umgebung derart wahrzunehmen [Thrun et al., 2002][DeSouza und Kak, 2002].

Für Menschen dagegen ist das Erkennen und somit die Klassifizierung von bekannten Objekten trotz Veränderung der Erscheinung kein Problem. Die Fähigkeit wichtige Merkmale zu erkennen und deren Funktion zu interpretieren ermöglicht es selbst völlig

unbekannte Gegenstände zumindest in grobe Klassen einzuordnen. Für diese Kategorisierung sind unter anderem Geometrie, Farbe und Textur wichtig. Diese Merkmale ermöglichen es, die Menge der für das Objekt infrage kommenden Kategorien stark einzuschränken und selbst auf Basis weniger Informationen bereits gute Klassifizierungsergebnisse zu erzielen [ANDERSON et al., 1989].
Durch verbesserte Sensoren und innovative Systeme, die angelehnt an menschliche Fähigkeiten entwickelt werden, ist es möglich die maschinelle Wahrnehmung zu verbessern um so die Herausforderungen der Servicerobotik zu bewältigen.
Dazu wird hier der Versuch unternommen, eine Objektklassifizierung auf Basis von menschenverständlichen Farb- und Textureigenschaften zu realisieren. Dies ermöglicht ein intuitives Erlernen von neuen Klassen anhand einer verbalen Beschreibung und reduziert gleichzeitig die Menge an benötigten Trainingsdaten. Ebenfalls ist es möglich mehrere Objekte einer Klasse anhand von einzelnen Eigenschaften zu differenzieren.

1.2 Ziel der Arbeit

In dieser Arbeit soll ein System entwickelt werden, das anhand von Textur- und Farbeigenschaften, die dem menschlichen Empfinden entsprechen, sowie Informationen anhand von 3D-Tiefendaten, eine Klassifizierung von Objekten in einem RGB-Farbbild vornimmt. Dazu soll das Farbbild in Segmente unterteilt werden, die in sich homogene Eigenschaften aufweisen. Zur optimalen Auswertung dieser Segmente sollen diese mit Hilfe der Tiefeninformationen aus der Kameraperspektive in eine kanonische Darstellung überführt werden. Auf den normalisierten Bildausschnitten werden die Farb-, Textur- und Tiefeneigenschaften ausgewertet. Anhand dieser Daten werden die Bildausschnitte mit Hilfe von statistischen maschinellen Lernverfahren klassifiziert.

1.3 Lösungsskizze

Die Segmentierung der RGB-D-Bilder findet in zwei Schritten statt. Im ersten Schritt werden aus dem Tiefenbild Objektoberflächen segmentiert. Dies geschieht, indem über das gesamte Bild Normalen anhand der Tiefendaten berechnet werden und anschlie-

ßend eine Segmentierung durch ein Region-Growing Verfahren vorgenommen wird. Für eine bessere Auswertung der erhaltenen Segmente werden diese in eine normalisierte Ansicht transformiert. Im zweiten Schritt werden die Segmente durch einen Split and Merge Algorithmus in homogene Bereiche unterteilt. Die Homogenität der Regionen wird dabei über mehrere Farb- und Texturmerkmale festgestellt. Nach Beendigung des Segmentierungsalgorithmus werden die Segmente in deren ursprüngliche Ansicht transformiert um eine Einteilung des Originalbildes vornehmen zu können.
Für jedes Segment werden die Hauptkomponenten anhand einer entsprechenden Hauptkomponentenanalyse ermittelt. Die Hauptkomponenten werden zur Erzeugung der Koordinatensysteme der Segmente genutzt. Anhand der Koordinatensysteme der Kamera und des Segments wird die Homographiematrix erzeugt und die Segmente in eine kanonische Darstellung transformiert. Durch die normalisierte Sicht auf die Segmente können die Farb- und Textureigenschaften ohne den Einfluss perspektivischer Verzerrung erzeugt werden. Diese Eigenschaften beschreiben Merkmale einer Textur die vom menschlichen Sehsystem wahrgenommen werden. Dazu gehört die Größe und Anzahl von kleinen, sich wiederholenden Texturmustern (engl. Primitives), deren Anordnung, sowie die statistische Untersuchung der auftretenden Grauwerte. Die ermittelten Ergebnisse werden in Merkmalsvektoren abgelegt und ermöglichen es einem maschinellen Lernverfahren eine Klassifizierung nach vordefinierten Klassen vorzunehmen.

1.4 Aufbau der Arbeit

In Kapitel 2 wird der aktuelle Stand der Technik sowie Forschungsarbeiten aus den für diese Arbeit thematisch relevanten Bereichen vorgestellt. Kapitel 3 stellt das Lösungskonzept sowie die Systemumgebung vor und geht auf die Theorie der gewählten Methoden und deren Implementierung ein. Die einzelnen Komponenten des Systems werden in Kapitel 4 evaluiert. Die Ergebnisse werden anschließend in Kapitel 5 diskutiert sowie ein Ausblick für zukünftige Aufgaben präsentiert.

Kapitel 2

Stand der Technik

In dieser Arbeit soll ein System entwickelt werden, das anhand von Textur- und Farbmerkmalen, die aus Farbbildern und Tiefeninformationen gewonnen werden, eine Klassifizierung der dargestellten Objekte und Oberflächen durchführt. Um gleichartige Bereiche im Farbbild zu erhalten, die entsprechenden Klassen zugeordnet werden können, muss eine Segmentierung des Bildes vorgenommen werden. Zur einheitlichen Auswertung werden die Segmente in eine kanonische Perspektive transformiert. Die Auswertung der Segmente anhand von Farb-, Textur- und Geometrieeigenschaften liefert eine Reihe von Werten, die in Attributvektoren dargestellt werden. Durch die Analyse der Vektoren anhand von statistischen, maschinellen Lernverfahren können die Segmente Klassen zugeordnet werden [Voss und Süsse, 1991]. Das folgende Kapitel beschreibt den aktuellen Stand der Technik in den Forschungsgebieten der Segmentierung (Kapitel 2.1), Texturanalyse (Kapitel 2.2) und Klassifizierung (Kapitel 2.3).

2.1 Segmentierung

Angesichts der unterschiedlichsten Anforderungen im Bereich der Bildverarbeitung wurden eine Vielzahl von Segmentierungstechniken entwickelt. Dabei können alle Techniken der folgenden grundlegenden Verfahren oder einer Kombination aus diesen zugeordnet werden [Tuceryan und Jain, 1998].

- Pixelorientierte Segmentierung

- Kantenbasierte Segmentierung
- Regionenorientierte Verfahren
- Modellbasierte Segmentierung

Bei der pixelorientierten Segmentierung wird jedes Pixel dahingehend überprüft, ob dieses Teil eines Segments ist. Dazu wird entweder nur das Pixel selbst oder auch dessen direkte Umgebung betrachtet. Diese lokale Betrachtung der Pixel führt allerdings zu Fehlern wenn weiche Übergänge zwischen Segment und Hintergrund vorhanden sind. Die kantenbasierten Segmentierungsverfahren dagegen können diese Übergänge besser erkennen. Dazu wird ein maximaler Wert der Ableitung erster Ordnung über das Grau- oder Farbbild gesucht und aus den angrenzenden Maxima auf die Segmentenkanten geschlossen.
Regionenorientierte Verfahren betrachten nicht nur die direkte Umgebung eines Pixels sondern fügen dieses zu einer Region hinzu, wenn ein entsprechendes Kriterium, wie z.B. der Grauwert, den Anforderungen der Region genügt. Durch dieses Vorgehen werden homogene Bereiche erzeugt und einzelne isolierte, sowie sehr kleine Regionen vermieden.
Sind Vorkenntnisse über die Geometrie der zu erwartenden Segmente vorhanden können mit der modellbasierten Segmentierung entsprechende Geometrien gesucht und erkannt werden [JÄHNE, 2005].
Die Segmentierung anhand von Grau- und Farbbildern findet ihre Grenzen wenn Segmente keine homogenen Grau- oder Farbwerte aufweisen. Diese Bereiche weisen meist Texturen auf, die eine Segmentierung anhand entsprechender Texturmerkmale erforderlich macht.

2.1.1 Split and Merge

Ein regionenbasiertes Verfahren ist Split and Merge [WANG, 2010]. Dieses eignet sich besonders für Bilddaten mit starkem Rauschen, da es diese in möglichst homogene Bildregionen zusammenfügt. Die Homogenität wird anhand von Kriterien bewertet. Diese sind häufig anhand der Grauwertinformationen wie Mittelwert, Varianz und den

Eigenschaften des Grauwert-Histogramms definiert. Es ist ebenfalls möglich Kriterien über Farbwerte und Texturmerkmale zu definieren. Das Verfahren kann in eine Split und eine Merge Phase unterteilt werden und beginnt mit der Betrachtung des gesamten zu segmentierenden Bildes:

- Split

    1. Unterteile den aktuellen Bereich in vier gleich große Teilbereiche und überprüfe die Homogenität.
    2. Wenn die Homogenität gegeben ist, den Splitvorgang beenden, sonst Teilbereiche in den Quadtree hinterlegen und Punkt 1 für jeden Teilbereich ausführen.

- Merge

    3. Suche zwei benachbarte Regionen, die das Homogenitätskriterium erfüllen. Ist die Suche erfolglos beende Algorithmus.
    4. Verschmelze die gefundenen Regionen und aktualisiere den Quadtree. Führe Punkt 3 aus.

Es ist sinnvoll für die Split und die Merge Phase unterschiedlich starke Homogenitätskriterien zu wählen. Ein schwaches Kriterium in der Split Phase kann zu einer Untersegmentierung führen, die das Ergebnis verfälscht. Eine Übersegmentierung dagegen kann von der Merge Phase ausgeglichen werden. Dies geschieht allerdings auf Kosten der Laufzeit des Algorithmus. In der Merge Phase entscheidet die Stärke des Kriteriums über die Anzahl der zu erhaltenen Regionen [Whelan und Ghita, 2008].

2.1.2 Mean Shift

Mean Shift [Comaniciu und Meer, 2002] ordnet Punkte aus einem Merkmalsraum in Gruppen ein. Dazu wird innerhalb eines Fensters um den betrachteten Punkt die Position der maximalen Verteilungsdichte gesucht und diese als neues Zentrum des Fensters verwendet (Abbildung 2.1). Diese Prozedur wird so lange wiederholt bis das

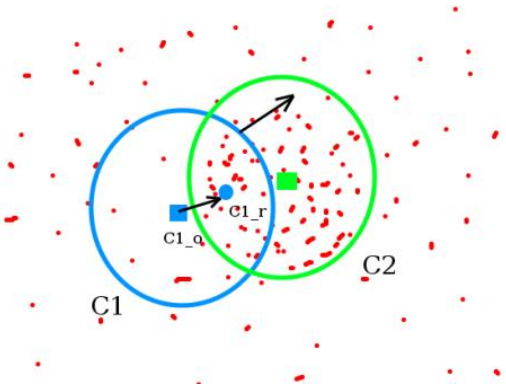


Abbildung 2.1: Mean Shift: Verschiebung des Pixelfensters in Richtung der maximalen Verteilungsdichte [OpenCV, 2014b].

Zentrum des Fensters mit dessen maximaler Verteilungsdichte übereinstimmt und somit das Zentrum einer Gruppe darstellt. Dieser Vorgang wird für alle Punkte bzw. Pixel durchgeführt. Ausführlicher wird das Verfahren in [Comaniciu und Meer, 2002] beschrieben.

Mean Shift baut auf den gradientenbasierten Verfahren auf, benötigt allerdings keine aufwendige Abschätzung der Verteilungsdichte und Berechnung der Gradienten. Der einfach zu berechnende Mean-Shift-Vektor zeigt in die gleiche Richtung wie der Gradient und benötigt durch seine adaptive Größe keine Schrittweite. Wie auch die gradientenbasierten Verfahren benötigt die Mean Shift Methode eine glatte Funktion, die aus dem diskreten Merkmalsraum geschätzt wird. Hierzu werden Kernel Dichte-Schätzer verwendet. Die Methode der Kernel Dichte-Schätzer ermittelt eine Funktion der Verteilungsdichte aus dem Mittel aller Verteilungen von Beobachtungen in einem festgelegten Fenster. Für ein Punkt $x = (x_1, ..., x_d, ...x_D) \in \Re^D$ im D-dimensionalen Merkmalsraum wird zur Bestimmung dessen Verteilungsdichte die N unterschiedlichen Beobachtungen innerhalb des betrachteten Fensters analysiert. Der Mittelwert der Verteilungsdichten der N Beobachtungen ergeben sich zu der Verteilungsdichte im Punkt x. Dabei ist h der Radius des Fensters und die Bandbreite des Kernels. Der Kernel hat zudem die Aufgabe die Beobachtungen anhand ihrer Nähe zu dem Mittelpunkt zu bewerten. Für eine gegebene Menge aus N D-dimensionalen Punkten ergibt

sich der Kernel Dichte-Schätzer zu:

$$f(x) = \frac{1}{Nh^D} \sum_{n=1}^{N} K\left(\frac{x - x_n}{h}\right). \tag{2.1}$$

Nach Cheng in [Cheng, 1995] ist der Kernel K(x) eine Funktion von $\| x \|^2$ unter der Bedingung, dass eine Funktion $k : [0, \infty] \to \Re$ als Profil zu Kernel K existiert, so dass

$$K(x) = c_{k,D} k(||x||^2). \tag{2.2}$$

Dabei muss k stückweise stetig, nicht negativ und nicht zunehmend sein. Zudem muss gelten $\int_0^N k(r)dr < \infty$. Ist $c_{k,D}$ positiv wird $K(x)$ zu eins integriert. Dadurch ist es möglich das Profil des Kernels und den Radius h im Dichte-Schätzer 2.1 direkt zu erkennen:

$$f(x)_{h,K} = \frac{c_{k,D}}{Nh^D} \sum_{n=1}^{N} k\left(\left|\left|\frac{x - x_n}{h}\right|\right|\right) \tag{2.3}$$

Im Gegensatz zu gradientenbasierten Verfahren muss bei Mean Shift keine Schrittgröße festgelegt werden, da diese bezüglich der lokalen Steigung der Verteilungsdichte adaptiv festgelegt wird. Der Mean Shift-Vektor, der in die gleiche Richtung wie der Gradient an einem Punkt $x = (x_1, ..., x_d, ...x_D) \in \Re^D$ zeigt ist:

$$m_{h,K}(x) = \frac{\sum_{n=1}^{N} x_n k\left(\left|\left|\frac{x-x_n}{h}\right|\right|^2\right)}{\sum_{n=1}^{N} k\left(\left|\left|\frac{x-x_n}{h}\right|\right|^2\right)} - x \tag{2.4}$$

mit Fensterradius h und N gegebenen D-dimensionalen Beobachtungen. Wird das Mean Shift Verfahren für die Segmentierung von Bildern genutzt, beinhalten der Mean Shift-Vektor häufig die Bildkoordinaten u und v sowie die Farbraumwerte R, G und B. Der Kernel K wird meist zum Gauss-Kernel gewählt. Werden die Koordinaten und die Farbraumwerte im Mean Shift-Vektor genutzt, können zwei Kernel eingesetzt werden um zwei unterschiedliche Bandbreiten für die räumliche und farbliche Entfernungen zu erhalten.

2.1.3 Region Growing

Das Region Growing gehört wie das Split and Merge zu den regionenorientierten Verfahren. Im Gegensatz zu dem Split and Merge Verfahren beginnt dieses Verfahren die Segmentierung auf Pixelebene. Es werden im Bild sogenannte Keimpunkte oder Seeds gesetzt, die als Startpunkt für das Wachstum einer Region dienen. Im Wachstumsvorgang wird anhand eines Ähnlichkeitskriteriums ein Wert eines benachbarten Pixels ermittelt und mit dem Mittelwert der in der Region vorhanden Pixel verglichen. Ist das Kriterium erfüllt, wird der Punkt der Region hinzugefügt und der Mittelwert der Region neu berechnet. Es werden zudem die Nachbarregionen verglichen und bei ausreichender Ähnlichkeit verschmolzen. Die Keimpunkte spielen somit eine große Rolle für die Qualität der Segmentierung, da die Anzahl eine obere Grenze der Segmentanzahl darstellt und eine ungünstige Verteilung im Bild zu fehlerhaften Segmenten führen kann [Adams und Bischof, 1994].

2.1.4 Tiefensegmentierung

Durch die steigende Verbreitung von Tiefenkameras wie Time-of-Flight Kamerasystemen oder der Microsoft® Kinect stehen in vielen Bereichen nun Tiefeninformationen der Umgebung des Robotersystems zur Verfügung. In unterschiedlichen Arbeiten wurden diese Tiefeninformationen genutzt um bestehende Segmentierungsverfahren um ein weiters Merkmal zu erweitern und somit das Ergebnis der Segmentierung zu verbessern [Bleiweiss und Werman, 2009] oder um neue Konzepte zu entwickeln, die eine Segmentierung nur anhand der Tiefendaten vornehmen können [Holz et al., 2012].
In der Arbeit von Bleiweiss und Werman [Bleiweiss und Werman, 2009] wird zur Verbesserung des Mean Shift-Verfahrens der Mean Shift-Vektor um eine Dimension erweitert und mit den Tiefendaten einer Time-of-Flight Kamera gefüllt. Unter Verwendung des $L * u * v*$ Farbraums und Koordinatenachsen x, y und z ergibt sich ein 6-Dimensionaler Mean Shift-Vektor. Unter Beachtung des Rauschverhaltens der Tiefenmessung werden die Tiefendaten nach ihrer Qualität gewichtet und können so die Segmentierung des Mean-Shift Verfahrens verbessern. Besonders bei bewegten Bilddaten die zu verwischten Farben neigen, können Verbesserungen gegenüber der reinen

Farbsegmentierung erkannt werden.

Eine Methode zur Segmentierung anhand der Tiefeninformationen wird von Holz in [Holz et al., 2012] vorgestellt. Dabei sollen die Normalen der Oberflächen ermittelt und ähnliche zusammengeführt werden. Dazu wird die Normale n_i anhand einer Hilfsebene, die durch die Nachbarschaft P_i definiert wird, approximiert. P_i kann dabei die k nächsten Nachbarn um p_i oder alle im Radius r um p_i liegenden Nachbarn sein. Anhand der Eigenvektoren der Kovarianzmatrix $C_i \in \Re^{3\times3}$ die über die Nachbarschaft P_i ermittelt wird ist es möglich die Normale n_i zu schätzen. Diese wird durch den Eigenvektor $v_{i,2}$, der dem kleinsten Eigenwert $\lambda_{i,2}$ am nächsten ist, ausreichend dargestellt. Die Krümmung der Oberfläche lässt sich durch das Verhältnis zwischen $\lambda_{i,2}$ und der Summe der Eigenwerte ermitteln. Die Schätzung der Normalen und der Krümmung wird durch die Wahl der Anzahl k der Nachbarn und dem Radius r der Nachbarschaft beeinflusst. Große Werte glätten kleine Störungen, können allerdings zu einem Verlust von lokalen Maxima oder dem verwischen von Kanten führen. Zu klein gewählte Werte erhöhen die Anfälligkeit gegenüber Messrauschen.

Serviceroboter, im speziellen der Care-o-bot®, werden in häuslicher Umgebung eingesetzt. Eine wichtige Aufgabe des Systems ist die Interaktion mit Objekten, was eine möglichst genaue Identifizierung und somit eine vorhergehende Segmentierung der Bilddaten voraussetzt. Um dies zu ermöglichen, sollen die vorhandenen Farb- und Tiefendaten genutzt werden. Dazu wird in dieser Arbeit eine zweistufige Segmentierung gewählt, die nach [Arbeiter et al., 2014] die Segmentierung im ersten Schritt anhand von Tiefendaten vornimmt. Diese liefert gute Ergebnisse, da in häuslichen Umgebungen viele gleichmäßige Flächen und deutlichen Kanten vorkommen. Im zweiten Schritt werden unterschiedliche Texturen identifiziert, indem auf den Segmenten ein Split and Merge Algorithmus angewendet wird. Der regionenortientierte Segmentierungsalgorithmus ermöglicht es, die Farb- und Texturturmerkmale dieser Arbeit als Homogenitätskriterium zu nutzen. Dadurch wird sichergestellt, dass nicht mehrere Texturen in einem Segment die Berechnung der Merkmale für die Klassifizierung verfälschen.

2.2 Texturanalyse

Das Erkennen von Mustern und deren Unterscheidung ist für Menschen eine einfache Aufgabe. Die Beschreibung von Texturen durch Parameter, die maschinell zuverlässig ausgewertet werden können, ist hingegen ein schwieriges Unterfangen, das vielseitig gelöst werden kann. In der Literatur können viele unterschiedliche Ansätze zur Texturanalyse gefunden werden [JÄHNE, 2005]. Bereits bei der Beschreibung von Textur ist noch keine einheitliche Definition vorhanden. In [COGGINS, 1983] wurden Versuche zusammengetragen Textur zu definieren. Einige werden im Folgenden erwähnt.

"We may regard texture as what constitutes a macroscopic region. Its structure is simply attributed to the repetitive patterns in which elements or primitives are arranged according to a placement rule."[TAMURA et al., 1978]

"The image texture we consider is nonfigurative and cellular[...]. An image texture is described by the number and types of its (tonal) primitives and the spatial organization or layout of its (tonal) primitives[...]. A fundamental characteristic of texture: it cannot be analyzed without a frame of reference of tonal primitive being stated or implied. For any smooth gray-tone surface, there exists a scale such that when the surface is examined, it has no texture. Then as resolution increases, it takes on a fine texture and then a coarse texture."[HARALICK, 1979]

"Die Strukturierung des Grauwerts wird allgemein als Textur bezeichnet. Texturen sind gerade dadurch gekennzeichnet, dass es nicht leicht ist, sie mit einem einfachen Merkmal zu charakterisieren, obwohl wir sie mit unserem visuellen System leicht erfassen und unterscheiden können."[VOSS und SÜSSE, 1991]

"Strukturen, die sich aus sich wiederholenden kleineren Mustern aufbauen, die sich wiederum aus sich wiederholenden Grauwerten in einer lokalen Nachbarschaft zusammensetzen, wobei die Grauwerte in einer solchen lokalen Nachbarschaft auch variieren dürfen, können als Textur bezeichnet werden."[HERMES, 1999]

Den meisten Definitionen ist gleich, dass sich Textur aus einem kleinen, sich regelmäßig wiederholenden Muster aufbaut. Dieses elementare Muster wird je nach Autor als Primitive [VAN GOOL et al., 1985], Texel [DAVIES, 2008] oder Texton [JULESZ, 1981] bezeichnet. Welche Eigenschaften ein Element aufweist und wie sich diese im Bild wiederholen muss um eine Textur darzustellen wird dagegen unterschiedlich eingeschätzt. Ein weiterer Ansatz zur Definition ist, dass Texturen sich hierarchisch aufbauen. Satellitenaufnahmen z.B. weisen bewohnte Regionen auf die wiederum in Gebäude, Straßen und weiteren Texturen wie Vegetation unterteilt werden können [JÄHNE, 2005].

Unabhängig von den Definitionen können die Verfahren an ihrem Verhalten gegenüber Rotation, Translation und Skalierung der auszuwertenden Bilddaten unterschieden werden. Ansätze die nicht invariant sind beziehen die Informationen der Lage der Textur in die Analyse mit ein und bieten die Möglichkeit nach Texturen in bestimmten Positionen zu suchen. Das ermöglicht Abweichungen der analysierten Daten gegenüber erwarteten Strukturen zu erkennen. Hingegen werden bei invarianten Ansätzen Texturen unabhängig von ihrer Position ausgewertet, so dass identische allerdings unterschiedlich angeordnete Texturen ebenfalls einander zugeordnet werden können [JÄHNE, 2005][ESSLINGER, 2014].

Texturen können anhand geeigneter Parameter beschrieben und unterschieden werden. Die Identifizierung solcher Merkmale ist Aufgabe der Texturanalyse. Aktuelle Verfahren können in statistische, strukturelle und modellbasierte Methoden eingeteilt werden und sind Gegenstand der folgenden Abschnitte dieses Kapitels.

2.2.1 Statistische Methoden der Texturanalyse

Nach Julesz [JULESZ, 1962] kann die Wahrnehmung des Menschen in die lokale Wahrnehmung von Textureigenschaften und die globale Wahrnehmung von Texturmustern aufgeteilt werden. Die lokale Wahrnehmung beruht auf der Fähigkeit statistische Eigenschaften zu erkennen. Die Erkennung von Texturmustern hingegen nutzt die Fähigkeit Kontextinformationen einzubeziehen und ermöglicht somit z.B. zwischen Vorder-

und Hintergrund zu unterscheiden. Das menschliche System ist in der Lage Dichteschwankungen und Schwankungen der Körnigkeit zu erkennen, die als Statistik erster bzw. zweiter Ordnung beschrieben werden.

Statistik erster Ordnung

Die statistische Untersuchung einzelner Pixel anhand deren Grauwerte wird als Statistik erster Ordnung bezeichnet. Entsprechende Texturparameter sind gegenüber Permutationen der Pixel invariant und daher rotations- und skaleninvariant. Dies führt allerdings dazu, dass unterschiedliche Texturen mit ähnlicher Verteilung der Grauwerte nicht unterschieden werden können.

Mathematisch ist die Statistik erster Ordnung durch den Mittelwert und die Varianz der Verteilung beschreibbar. In der Texturanalyse handelt es sich dabei um die Verteilung der diskreten Grauwerte im Bild.

Ist die Anzahl der möglichen Grauwerte $g_q(q = 0, 1, 2, ..., Q)$ in einem Bild durch Q gegeben, so ergibt sich für die relative Häufigkeit

$$f(q), \quad 1 \leq q \leq Q \tag{2.5}$$

die Bedingung

$$0 \leq f(q) \leq 1 \, \forall q, \quad \sum_{q=1}^{Q} f(q) = 1, \tag{2.6}$$

da die Wahrscheinlichkeit einen beliebigen Grauwert zu erhalten mit eins definiert ist. Der Mittelwert μ ist das Moment 1-ter Ordnung. Die Momente n-ter Ordnung ergeben sich aus

$$\mu_n = \sum_{q=1}^{Q} f(q) q^n, \quad n \in N \tag{2.7}$$

und die zentralen Momente n-ter Ordnung σ_n^2 aus

$$\sigma_n^2 = \sum_{q=1}^{Q} (q - \mu)^n f(q). \tag{2.8}$$

Das zentrale Moment erster Ordnung ist per Definition null, währen das zweite zentrale Moment der Varianz entspricht [Weis, 2000].

Statistik zweiter Ordnung

Statistiken die lediglich ein Pixel betrachten, können meist Texturen nicht ausreichend gut für einen praktischen Einsatz beschreiben [Davies, 2008]. Die Statistik zweiter Ordnung betrachtet Pixelpaare und die räumlichen Zusammenhänge der Grauwerte. Eine etablierte Methode ist die Co-ocurrence-Matrix, auch Grauwertmatrix genannt. Diese beschreibt die relativen Häufigkeiten der auftretenden Grauwertverhältnisse zwischen zwei Pixeln. Die Anzahl der Matrixelemente C_{ij} der Grauwertmatrix M wird durch die Anzahl der Grauwerte G zu $N = G \times G$ bestimmt. Alle Pixel eines $n \times m$ Bildes werden zur Erstellung der Grauwertmatrix betrachtet. Ein Pixelpaar besteht dabei aus dem betrachteten Pixel und dem Pixel der sich im Abstand d mit dem Winkel θ befindet. Somit können beispielsweise die horizontalen und vertikalen Nachbarn durch den Abstand $d = 1$ und den Winkeln $\theta_H = 0°$ und $\theta_V = 90°$ bestimmt werden. Die Anzahl der aufgetretenen Grauwertkombinationen ist P_{ij} wobei i der Grauwert des betrachteten Pixels und j der Grauwert des zweiten Pixels ist. Die relative Häufigkeit ergibt sich somit zu

$$C_{ij} = \frac{P_{ij}}{R}. \tag{2.9}$$

Die Anzahl aller Pixelpaare ist R. Die Aussagekraft der einzelner Häufigkeiten C_{ij} ist gering, weswegen diese in der Co-occourence-Matrix aufgetragen werden.

$$M = \begin{pmatrix} C_{0,0} & C_{0,1} & \dots & C_{0,G-1} \\ C_{1,0} & C_{1,1} & \dots & C_{1,G-1} \\ \dots & \dots & \dots & \dots \\ C_{G-1,0} & c_{G-1,1} & \dots & C_{G-1,G-1} \end{pmatrix} \tag{2.10}$$

Um die in der Co-occournce-Matrix enthaltenen Informationen auswerten zu können, benötigt man definierte Texturmaße. Haralick [Haralick et al., 1973] hat 14 solcher Maße entwickelt, mit denen es möglich ist die Eigenschaften von Texturen ausreichend

Tabelle 2.1: Fünf wichtige Texturmaße nach [HARALICK et al., 1973].

Texture Feature	Formula
Energy	$\sum_i \sum_j C_{ij}^2$
Entropy	$\sum_i \sum_j C_{ij} \log_2 C_{ij}$
Contrast	$\sum_i \sum_j (i-j)^2 C_{ij}$
Homogeneity	$\sum_i \sum_j \frac{C_{ij}}{1+\lvert i-j \rvert}$
Correlation	$\sum_i \sum_j \frac{(i-\mu_x)(j-\mu_y)C_{ij}}{\sigma_x \sigma_y}$

zu beschreiben. In [HARALICK und SHANMUGAM, 1973] wird gezeigt, dass bereits anhand der fünf Maße *Energy*, *Entropy*, *Contrast*, *Homogenity* und *Correlation* eine gute Beschreibung von Texturen möglich ist. Die Texturmaße werden in Tabelle 2.1 aufgeführt. Die Reduzierung der Maße geht allerdings mit einem Informationsverlust einher, der in [CONNERS und HARLOW, 1980] beschrieben wird.

Die Parameter μ_x, μ_y sowie σ_x, σ_y sind der Mittelwert und die Standardabweichung von $C_x(i) = \sum_{j=1}^{G} M(i,j)$ und $C_y(j) = \sum_{i=1}^{G} M(i,j)$. Welche Texturmaße letztlich verwendet werden, um ein optimales Ergebnis zu erhalten, ist abhängig von den Anforderungen und muss im einzelnen entschieden werden. Zudem existiert keine etablierte Methode zur Bestimmung des Abstands d. Eine Auswertung anhand unterschiedlicher Abstände ist aufgrund des Rechenaufwands nicht praktikabel [TUCERYAN und JAIN, 1998].

2.2.2 Strukturelle Methoden der Texturanalyse

Die strukturellen Methoden zur Texturanalyse identifizieren elementare Texturelemente und schließen durch deren Auftreten im Bild auf Anordnungsregeln. Die Leistungsfähigkeit dieser Methoden hängt stark von den betrachteten Texturen ab. Um gute Ergebnisse zu erzielen zu können, sind möglichst regelmäßige Texturen nötig [TUCE-

RYAN und JAIN, 1998]. Strukturelle Methoden eignen sich besonders zur Beschreibung von Makrotexturen und beschreiben deren Eigenschaften häufig anhand der durchschnittlichen Intensität der Elemente, Orientierung, Elongation, Größe, Kompaktheit, Momente etc.[BAHEERATHAN et al., 1999][ZHANG und TAN, 2002].

Goyal [GOYAL, 1995] nutzt den Umfang und die Kompaktheit der Primitives zur Beschreibung von Textur. Der Umfang ist theoretisch invariant gegenüber Rotation und Translation. In der Praxis unterliegt dieser allerdings einem Quantisierungsfehler, dem Goyal durch eine neue Methode zur Berechnung des Umfangs in [GOYAL et al., 1994] begegnet. Die Kompaktheit, die auch als Bewertung der Kreisförmigkeit betrachtet werden kann, ist definiert durch:

$$r = \frac{P^2}{4\pi S}. \tag{2.11}$$

Dabei ist S die Fläche des Primitives und P der Umfang. Neben der Rotations- und Translationsinvarianz weist die Kompaktheit ebenfalls eine Skalierungsinvarianz auf [ZHANG und TAN, 2002].

Basierend auf der Hough-Transformation [HABERÄCKER, 1989] gewinnen Eichmann und Kasparis [EICHMANN und KASPARIS, 1988] invariante Texturmerkmale. Linien im Ausgangsbild werden im Ergebnisraum der Hough-Transformation als Punkte dargestellt. Dabei verschieben sich die dargestellten Linien im Ergebnisraum auf der Winkel-Achse bei Rotation des Eingangbildes und bei Größenveränderung auf der Achse die den Abstand zu dem Bildmittelpunkt beschreibt. Somit ist es möglich die Anzahl der unterschiedlichen Orientierungen der erkannten Linien, die relativen Winkel der Orientierung und den Abstand der Linien einer Orientierung zueinander zu bestimmen. Durch den Einsatz von Medianfilter auf den Ergebnisraum kann auftretendes Rauschen eliminiert werden. Die Hough-Transformation ist ursprünglich für Binärbilder ausgelegt, weshalb Eichmann und Kasparis die Radon-Transformation nutzen, um auch Grauwertbilder analysieren zu können. Diese Texturmerkmale beschränken sich auf die Analyse von Texturen, die sich hauptsächlich durch Linien auszeichnen [ZHANG und TAN, 2002].

Weitere sechs Merkmale, die an die menschliche Wahrnehmung angelehnt sind, werden in [TAMURA et al., 1978] vorgestellt. Die Entwicklung und Bewertung der Merkmale ist dabei an psychologische Messungen angelehnt. Ein grundlegendes Merkmal ist die Coarseness, welche die Granularität einer Textur beschreibt. Texturen mit wenigen großen Primitives entsprechen einer hohen Coarseness, während viele kleine Primitives mit einer niedrigen Coarseness bewertet werden. Ermittelt wird die Coarseness, in dem für jeden Punkt im Bild der Mittelwert der benachbarten Grauwerte gebildet wird. Anschließend werden die Differenzen zwischen sich nicht überlappenden, horizontalen und vertikalen Nachbarn gebildet. Dem Pixel wird der Wert der höchsten Differenz zugewiesen. Die Coarseness wird anhand des Mittelwertes über alle ermittelten maximalen Differenzen definiert.
Das zweite Merkmal ist der Contrast (Kontrast), der Intensitätsunterschiede zwischen benachbarten Pixel beschreibt. Hierzu wird die Standardabweichung σ und Varianz σ^2 um den Mittelwert der Wahrscheinlichkeitsverteilung der Grauwerte sowie das vierte Moment des Mittelwertes ermittelt. Anhand derer ist es möglich die Kurtosis α_4 und somit ein Maß der Polarisierung zu erhalten. Der Kontrast wird schließlich als $F_{con} = \frac{\sigma}{\alpha_4}$ bestimmt.
Die Directionality (Ausrichtung) ist ein Merkmal, das sich über eine definierte Region erstreckt. Dabei werden die Form und die Ausrichtung der Primitives in der betrachteten Region untersucht. Dies geschieht, indem alle Kanten, deren Intensitäten einen Schwellwert übersteigen, anhand ihrer Orientierung in einem Histogramm aufgetragen werden. Das Maß des Attributs wird durch die Anzahl und Stärke der Peaks bestimmt. Wenige große Peaks führen zu einem hohen Wert der Ausrichtung. Die Form der Primitives wird mit der Line-likeness beschrieben. Eine richtungsabhängige Cooccurencematrix wird dazu erstellt. Deren Einträge beinhalten die Frequenz mit der zwei Nachbarn, die über eine bestimmte Entfernung zueinander verfügen, mit jeweils einer bestimmten Ausrichtung auftreten. Alle Einträge, die zwei identische Richtungen besitzen werden mit +1 und Einträge mit orthogonal zueinander stehenden Richtungen mit -1 gewichtet.
Die Regularity ist ein Maß für Regelmäßigkeit der Anordnung der Primitives. Eine Textur ist nach [TAMURA et al., 1978] regelmäßig, wenn alle bereits beschriebenen

Merkmale über ein Bild nicht variieren. Dazu wird das Bild unterteilt und für jedes Teilbild die Merkmale berechnet. Eine geringe Differenz der Merkmale über alle Teilbilder führt zu einer hohen Regelmäßigkeit.
Das letzte Merkmal beschreibt die physikalischen Oberflächenrauigkeit der Textur und wird Roughness genannt. Aus der Summe der Coarseness und des Contrast wird der Wert des Merkmals gebildet.

Entsprechend der Arbeit von [Tamura et al., 1978] werden in [Amadasun und King, 1989] fünf Texturmerkmale vorgestellt, die an die menschliche Wahrnehmung angelehnt sind. Durch neue Methoden der Berechnung soll die Beschreibungsfähigkeit verbessert und der Berechnungsaufwand der Merkmale verringert werden.
Die Coarseness wird hier ebenfalls durch eine lokale Homogenität der Intensitätswerte dargestellt. Diese wird durch geringe Intensitätsänderungen über das Bild und somit durch kleine Differenzen zwischen der Intensität eines Pixels und dem Mittelwert der benachbarten Intensitäten deutlich.
Die neighborhood gray tone difference matrix (NGTDM) summiert Änderungen in den Intensitäten über das gesamte Bild auf und stellt somit ein inverses Maß für die Coarseness dar.
Ein weiteres Merkmal, das ebenfalls in [Tamura et al., 1978] vorgestellt wird, ist der Kontrast (enlg. Contrast). Bei Amadasun und King ist der Kontrast stark, wenn Regionen unterschiedlicher Intensitäten deutlich sichtbar sind. Dazu muss der Intensitätsunterschied der Regionen hoch sein. Erkannt werden kann dies durch weit auseinanderliegende Grauwerte oder sich häufig ändernde Intensitäten. Anhand der NGTDM können diese Eigenschaften einer Textur ermittelt und der Kontrast bestimmt werden.
Ein neu vorgestelltes Merkmal ist Busyness. Diese Merkmal beschreibt die Häufigkeit von starken Intensitätsschwankungen wodurch ein entsprechendes Bild unruhig wirkt. Der Quotient zwischen der Änderungshäufigkeit der Intensitäten über dem Bild und der Summe der Differenzen zwischen den Grauwerten dient als Wert des Merkmals.
Der Informationsgehalt einer Textur wird Complexity (Komplexität) genannt. Dieses Merkmal besitzt einen hohen Wert wenn viele Informationen in einer Textur enthalten sind. Ein hoher Informationsgehalt liegt vor, wenn viele Primitives im Bild vorhanden

sind und wird durch unterschiedliche mittlere Intensitäten der Primitives verstärkt. Somit weisen Texturen mit hoher Komplexität ebenfalls hohe Busyness und Contrast Merkmale auf. Erzeugt wird die Komplexität durch die Summe der normalisierten Differenzen zwischen jedem Punktepaar, welche anhand der Summe der entsprechenden Einträge in der NGTDM gewichtet werden.
Das letzte Merkmal ist die Texture Strength. Eine entsprechende Textur zeichnet sich durch Primitives aus, die leicht zu erkennen, sowie gut zu unterscheiden sind. Über die Größe und den Intensitätsunterschieden der Primitives können diese Eigenschaften bestimmt werden. Die Bewertung des Merkmals, findet anhand des Quotienten aus der Anzahl von unterschiedlichen Intensitätsniveaus, zwischen benachbarten Primitives und der Größe der Primitives der Textur, statt.

Local Binary Patterns

Der Local Binary Pattern (LBP) Operator kann als Kombination aus statistischen und strukturellen Methoden der Texturanalyse gewertet werden [Nava et al., 2011]. Der von [Ojala et al., 2002] vorgestellte Operator bietet eine gutes Maß für Texturen bei geringen Berechnungskosten. Dabei zeichnet sich der Operator durch seine Invarianz gegenüber monotonen Schwankungen der Grauwerte sowie der Invarianz gegenüber Rotation aus [Ojala et al., 2002].

Der LBP Operator untersucht in einer Nachbarschaft mit P Pixeln ($P > 1$) lokale Texturen in einem Grauwertbild. Diese werden mit

$$T = t(g_c, g_0, ..., g_{P-1}) \tag{2.12}$$

beschrieben. Der Grauwert des Zentrums der Textur ist g_c während die restlichen Einträge die Grauwerte der Nachbarn darstellen. Die Nachbarschaft ist kreisförmig definiert mit dem Radius $R > 0$. Die kleinste Nachbarschaft wird durch eine 3×3 Umgebung dargestellt (Abbildung 2.2(a)). Bei größeren Radien können die Nachbarn auf der Grenze zwischen mehreren Pixeln liegen. Der Wert des Nachbarn wird in diesem Fall aus den Grauwerten der überlagerten Pixel interpoliert. Im nächsten Schritt wird

der Wert des Zentrums als Schwelle definiert. Unterschreitet der Wert eines Nachbarn diese Schwelle wird er zu 0 gesetzt, ansonsten zu 1 (Abbildung 2.2(b)). Die erhaltene Folge wird als Binärwert interpretiert und im Uhrzeigersinn mit dem Wert 2^p multipliziert (Abbildung 2.2(c)). Die Summe der Produkte liefert einen für die lokale Textur eindeutigen dezimalen Wert(Abbildung 2.2(d)). Diese Berechnung des LBP Operators ist allerdings nicht invariant gegenüber Rotation. Diese Eigenschaft wird erhalten, indem die binäre Zahlenfolge bitweise rotiert wird, bis die kleinste dezimale Zahl erhalten wird.

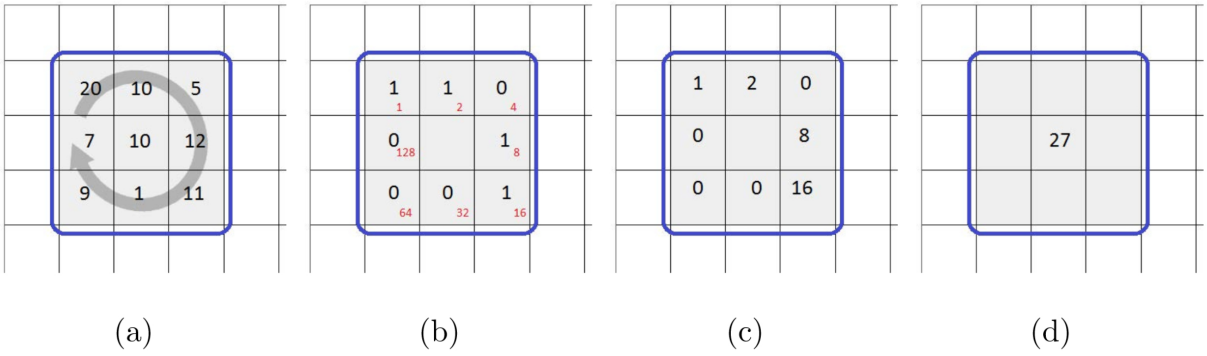


Abbildung 2.2: Berechnung des LBP Wertes auf einer 3×3 Umgebung [Nava et al., 2011].

In [Pietikäinen et al., 2000] wird gezeigt, dass Texturen nur mäßig gut unterschieden werden können, wenn alle auftretenden Texturmuster genutzt werden. Texturen bestehen teilweise zu über 90% aus Texturmustern die lediglich ein oder zwei 1/0 Bitübergänge aufweisen[Ojala et al., 2002]. Diese einheitlichen Texturmuster (engl. uniform pattern) stellen Mikrotexturen wie Punkte, Flächen, Linien, Kanten und Ecken, wie in Abbildung 2.3, dar [Ojala et al., 2002].

Um Texturen analysieren zu können, müssen die auftretenden Texturmuster in einem Histogramm aufgetragen werden. Die Verteilung der Muster im Bild kann anhand des Histogramms ermittelt werden, und sind für die jeweiligen Texturen charakteristisch [Nava et al., 2011]. Die Klassifizierung von Texturen kann anhand eines Vergleichs zwischen einem Beispiel und einem Modell Histogramm vorgenommen werden. Als Maß können Metriken wie Chi-Quadrat, L1 oder Bhattacharyya Distanz verwendet

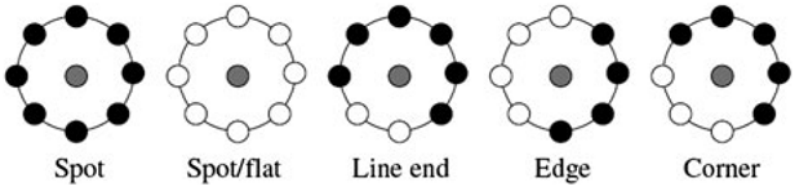


Abbildung 2.3: Texturmuster mit höchstens zwei 1/0 Übergängen [SHOYAIB et al., 2011].

werden [SCHAEFER und DOSHI, 2012]. Die zu klassifizierende Textur wird dem Modell zugeordnet, das die geringste Distanz des verwendeten Maßes aufweist [GUO und ZHANG, 2010].

2.2.3 Modellbasierte Methoden der Texturanalyse

Modellbasierte Methoden ermitteln eine bestimmte Anzahl an Parametern, die es ermöglichen eine Textur genau zu beschreiben oder auch künstlich darzustellen [TUCERYAN und JAIN, 1998]. Zu diesen Methoden gehören unter anderem Gaussian Markov Random Fields [COHEN et al., 1991], Gibbs Random Fields(GRF) [SIVAKUMAR und GOUTSIAS, 1999] und das Wold model [LIU und PICARD, 1996] ,[LIU und PICARD, 1994].

Markov Random Fields

Bei der Texturanalyse anhand von Markov Random Fields (MRF)[TUCERYAN und JAIN, 1998] werden die Pixel oder Gruppen von Pixeln eines Bildes als Zufallsvariablen betrachtet und entsprechenden Labels zugeordnet.

Bei Anwendungen mit a priori Wissen, wie bei der Analyse von Satellitenaufnahmen, können diesen Labels Objekte zugeordnet werden. Dabei können Siedlungen oder Vegetation repräsentiert werden oder bei fehlenden Vorwissen z.B. der Wert einer 8-bit

Graustufe. Während die Zufallsvariablen unabhängig voneinander sind, hängen die Wahrscheinlichkeiten der Label von den benachbarten Labeln sowie deren zugeordneter Zufallsvariablen ab. Somit stellen die MRFs eine Wahrscheinlichkeitsverteilung der Zufallsvariablen unter Einbeziehung der Nachbarschaften auf. Dadurch sind die Wahrscheinlichkeiten nicht ausschließlich durch die Werte im gegebenen Bildbereich bestimmt, sondern ziehen Informationen aus der Umgebung mit ein.

Das zu analysierende Bild kann durch ein $M \times N$ Gitter dargestellt werden:

$$L = \{(i, j) | 1 \leq i \leq M, 1 \leq j \leq N\}. \tag{2.13}$$

Zufallsvariablen werden durch $I(i, j)$ dargestellt und entsprechen dem Wert eines Pixels an der Stelle (i, j) im Gitter L. Für eine einfache Indizierung werden die Punkte des Gitters zeilenweise nummeriert so dass man I_t für

$$t = (i - 1)N + j \tag{2.14}$$

erhält. Es wird zwischen unterschiedlichen Ordnungen der Nachbarschaftsbeziehungen des Bereichs t unterschieden. Zum einen gibt es die 4er-Nachbarschaftsbeziehung, die als Nachbarschaft erster Ordnung und die 8er-Nachbarschaftsbeziehung, die als Nachbarschaft zweiter Ordnung bezeichnet wird. Diese Nachbarschaften können weiter unterteilt werden um nur bestimmte der angrenzenden Nachbarn in die Betrachtung mit einzubeziehen. Diese Sammlung von Bereichen wird Nachbarschaftscliquen genannt (Abbildung 2.4).

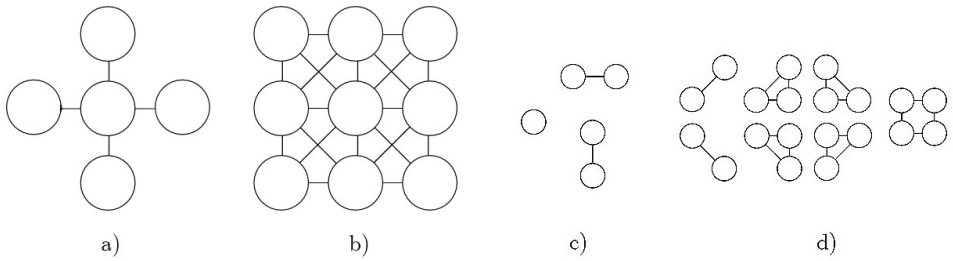


Abbildung 2.4: Nachbarschaftscliquen des Markov Random Fields 1.Ordnung a) + c). Nachbarschaftscliquen 2.Ordnung b) + c) + d).

Betrachtet man eine Nachbarschaftsclique 1. Ordnung, enthält diese drei unterschiedliche Konfigurationen. Die Häufigkeit jeder auftretenden Konfiguration, anhand der berechneten Wahrscheinlichkeiten der Label, bilden die MRF Parameter. Diese ermöglichen es ähnliche Texturen zu identifizieren.

Wold Model

Die Wold Theorie ermöglicht es ein 2-dimensionales homogenes random field $y(m, n)$ in drei gegenseitig orthogonale Komponenten aufzuteilen. Die Komponenten können als Zufälligkeit $w(m, n)$, Richtung $p(m, n)$ und Periodizität $g(m, n)$ der Textur beschrieben werden und entsprechen näherungsweise den von [Rao und Lohse, 1993] ermittelten wichtigsten Komponenten bei der Erkennung von Texturen durch den Menschen: "repetitiveness", "directionality" und "granularity and complexity". Diese Komponenten der Textur können durch die 2D Wold Dekomposition als

$$y(m, n) = w(m, n) + p(m, n) + g(m, n) \tag{2.15}$$

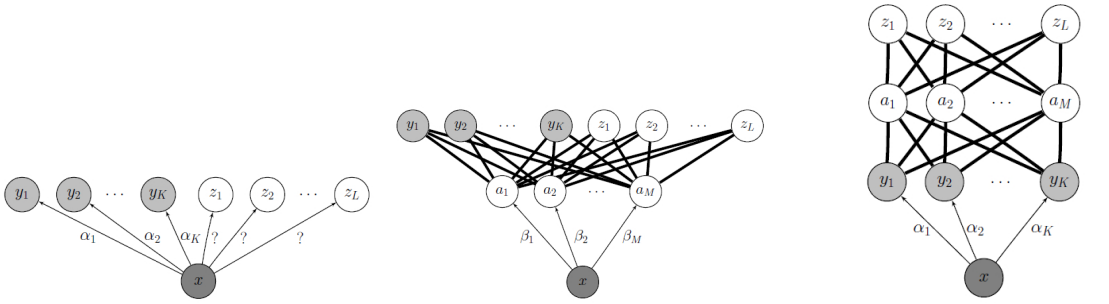
beschrieben werden. Zur Bestimmung der Parameter der Komponenten können unterschiedliche Ansätze wie die Maximum-likelihood Methode von [Francos et al., 1995] oder die Dekomposition der spektralen Dichtefunktion der Textur nach [Francos et al., 1993] genutzt werden.

2.2.4 Attributbasierte Texturklassifizierungsverfahren

Arbeiten in diesem Bereich stellen die Bedeutung von semantischen Attributen heraus. Diese ermöglichen die Beschreibung von Objekten anhand von Eigenschaften wie "ist gestreift", "hat zwei Räder" oder "hat vier Beine". Gleiche Objekte, mit einzelnen unterschiedlichen Attributen, können somit gut differenziert werden z.B. "Katze mit Punkten" und "Katze mit Streifen". Klassen können durch Attribute beschrieben werden und benötigen keine Trainingsbilder, wodurch eine einfache Erweiterung des Systems um Klassen durch den Anwender möglich wird.

Ein Ansatz um Attribute zu erlernen wurde von [Lampert et al., 2009] vorgestellt. Es werden Trainingsdaten aus unterschiedlichen Klassen zu Mengen mit identischen Attributen zusammengefügt. So kann z.B., abgeleitet aus einer Menge mit Bienen und Zebras, die Menge "gestreift" Bilder von Bienen und Zebras umfassen während die Menge "ist gelb" nur Bienen enthalten kann. Um dieses Vorgehen zu ermöglichen, wurden zwei Methoden zur attributbasierten Klassifikation eingeführt. Die Direct Attribute Prediction (DAP) trennt dabei in einem flachen Klassifizierungssystem die Bilddaten von der Ausgabeschicht durch einfügen einer Attributschicht (Abbildung 2.5(b)). Im Training erzeugen die Klassen der Bilddaten $(y_k)_{k=1,\dots,K}$ Werte in der Attributschicht $(a_m)_{m=1,\dots,M}$ anhand derer die Attributparameter β_m erlernt werden können. In der Testphase werden die Attributwerte zur Klassifizierung genutzt. Sind dabei keine Trainingsbilder vorhanden, können die entsprechenden Attribute händisch der Klasse zugeordnet werden, wodurch auf Trainingsbilder zum Erlernen neuer Objekte verzichtet werden kann. Die zweite Methode ist die Indirect Attribute Prediction (IAP), die wie die flachen Klassifizierungssysteme in der Trainingsphase für jede Klasse einen Parameter α_k lernt. Die Attributschicht verbindet hier die Schicht der trainierten Klassen mit der Schicht der unbekannten Klassen (Abbildung 2.5(c)). Die Ergebnisse der Vorhersagen der Trainingsphase gewichten die Attribute in der angrenzenden Schicht. Die Vorhersage der Testdaten kann aus der Attributschicht abgeleitet werden.

[Farhadi et al., 2009] beschränkt sich bei der Klassifizierung von Objekten nicht auf semantische Attribute sondern führt zusätzlich diskriminative Attribute und sogenannte "base features" ein. Semantische Attribute beschränken sich dabei auf die Beschreibung der geometrischen Form wie z.B. "2D Würfel", "3D Würfel" oder "Zylinder", die Beschreibung von Bestandteilen wie z.B. "besitzt Arm", "besitzt Rad" oder "besitzt Tür", sowie die Beschreibung des Materials wie z.B. "ist Fell", "ist glänzend" oder "ist Holz". Um Klassen, die durch eine ähnliche Menge von Attributen beschrieben werden, gut differenzieren zu können, werden diskriminative Attribute vorgestellt. Diese sind als Vergleich definiert und ermöglichen es, für Klassen fehlende typische Attribute zu erkennen oder atypische Attribute für eine Klasse zu identifizieren. Die Beschreibung der Attribute findet anhand der Base Features statt. Diese bestehen zum



(a)Flaches Klassifizierungssystem

(b) Direkt attribute prediction (DAP)

(c) Indirect attribute prediction (IAP)

Abbildung 2.5: Attributbasierte Klassifikation: Dunkelgraue Knoten werden ständig überwacht, hellgraue Knoten werden in der Trainingsphase überwacht, weiße Knoten werden abgeleitet. (a) Nur direkte Klassifizierung von gelernten Klassen. (b) Training von Parameter β_M zur Prädiktion der Attribute. Klassifizierung anhand Attributwerte. (c) Direkte Klassifizierung wie (a) für y_K. Schätzung der Attribute a_M anhand y_K über alle Testdaten. Unbekannte Klassen z_L werden in Abhängigkeit der Klassen-Attribut Beziehung ermittelt. [Lampert et al., 2009].

einen aus Farb- und Texturmerkmalen zur Beschreibung des Materials. Zum anderen werden Bestandteile durch Visual Words definiert und die Beschreibung der Geometrien findet anhand von Kanten statt. Ein Attributklassifizierer erzeugt den Attributwert aus dem Merkmalsraum eines Testbildes. Dieser Klassifizierer muss anhand von Merkmalen trainiert werden, die das Attribut gut repräsentieren. Zur Ermittlung solcher Merkmale werden Trainingsdaten gewählt, die identische Objekte aufweisen. Ein Teil dieser Trainingsdaten muss das gesuchte Attribut enthalten, während die restlichen Daten dieses nicht tun. Nur Merkmale, die Attribute erkennen welche auch im Trainingsbild vorhanden sind, werden anhand einer regulierten L1 Regression ermittelt. Dies verhindert, dass ein Attributklassifizierer typische Umgebungseigenschaften eines Attributs lernt, anstelle das Attribut selbst. Die ermittelten Attribute werden zur Klassifizierung genutzt (Abbildung 2.6).

Im Gegensatz zu den vorhergehenden Methoden wählt [Cimpoi et al., 2014] die Attribute anhand von Wörtern, die Menschen zur Beschreibung von Texturen nutzen.

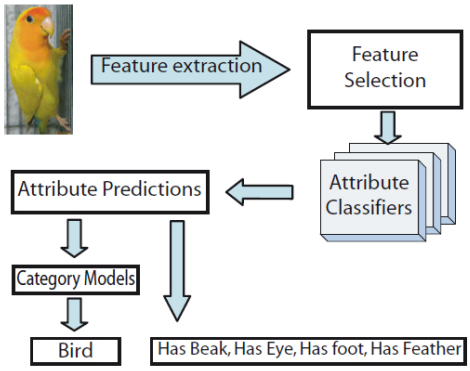


Abbildung 2.6: Klassifizierung anhand von Attributen nach [FARHADI et al., 2009].

Diese müssen Eigenschaften eines Objektes beschreiben, die durch Betrachtung erfasst werden können. Aus der Vielzahl von Wörtern, die Texturen beschrieben, werden in [BHUSHAN et al., 1997] 98 eindeutige und wiederholungsfreie Wörter identifiziert. Durch Ausschluss von Wörtern, die nicht für eine visuelle Beschreibung geeignet sind, werden 47 Attribute ausgewählt und für jedes 120 Trainingsbilder zusammengetragen. Für die Darstellung der Attribute kann der Improved Fisher Vector (IFV) oder DeCAF features genutzt werden. Dazu werden bei dem IFV lokale SIFT descriptors ermittelt die über ein Gausian Mixture Model quantisiert werden.

Die in dieser Arbeit verwendeten Texturmerkmale haben den Anspruch von Menschen interpretierbar zu sein. Dies hat das Ziel, ein System zu entwickeln, das dem Nutzer ermöglicht, neue Objekte intuitiv anhand ihrer Texturmerkmale dem System beizubringen. Außerdem soll bei mehreren, als ähnlich erkannten Objekten, eine Unterscheidung durch weitere Kriterien herbeigeführt werden können. Dieser Ansatz wird bereits von [LAMPERT et al., 2009] verfolgt, in dem zufällig nach wiederkehrenden Merkmalen gesucht wird. Weisen diese Merkmale einen ausreichenden Informationsgehalt auf, werden diese einer von Menschen interpretierbaren Eigenschaft zugeordnet. In Ergänzung zu den lernbasierten Verfahren zur Bestimmung von Texturattributen werden hier die Texturmerkmale, angelehnt an die menschlich Wahrnehmung, händisch ausgewählt. Leistungsfähige Verfahren wie der LBP Operator oder Markov Random Fields werden nicht mit einbezogen, da die ermittelten Histogramme und Wahrscheinlichkeiten nicht intuitiv interpretiert werden können. Die gewählten Texturmerkmale können in Farb-

und Strukturmerkmale unterteilt werden. Die Auswertung der Farbe wird durch Statistiken 1. Ordnung über die Kanäle des HSV-Farbraumes realisiert. Dies entspricht der lokalen Wahrnehmung von Menschen [Julesz, 1962]. Zur Analyse der Struktur werden die Primitives anhand von Konturen identifiziert und deren Eigenschaften wie Größe, Häufigkeit, Deutlichkeit, Gleichheit und Linienartigkeit bestimmt. Die Anordnung der Primitives wird als Linien- oder Gitterartig beschrieben und deren Ausrichtung festgestellt. Über die gesamte Textur wird zudem der Kontrast nach [Amadasun und King, 1989] ermittelt.

2.3 Klassifizierung

Die Aufgabe der Klassifizierung ist, eine Menge von Objekten anhand deren Merkmalen in definierte Klassen einzuordnen [Ferber, 2003]. Objekte können je nach Anwendung z.B. Bilder oder Bildsegmente wie Buchstaben bei der Texterkennung oder akustische Signale zur Spracherkennung sein. Anhand von Merkmalen können Eigenschaften von Objekten beschrieben werden und sind somit die eigentlichen Daten, die zur Klassifizierung dienen. Durch m Merkmale wird ein m-dimensionaler Merkmalsraum aufgespannt, in dem jedes Objekt durch dessen Merkmalsvektor einer genauen Position zugeordnet ist (Abbildung 2.7).

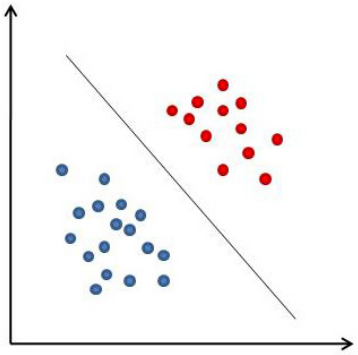


Abbildung 2.7: 2-dimensinaler Merkmalsraum in dem Klasse blau mit einer Geraden von Klasse rot getrennt werden kann.

Kompakte Agglomerationen von Merkmalsvektoren können jeweils als Klasse interpretiert werden, die durch die Merkmalsvektoren repräsentierte Objekte enthält. Eine Beschreibung der Klasse ist so durch verallgemeinerte Klassenvektoren oder die Klassen abgrenzenden Flächen möglich. Man erkennt, dass durch eine steigende Dimensionalität des Merkmalsraums der Aufwand für die Trennung der Merkmalsvektoren stark ansteigt. Die höhere Dimensionalität führt zu meist größeren Abständen der Agglomerationen, was eine Unterscheidung erleichtert. Methoden zur Berechnung der optimalen Anzahl und den am besten geeigneten Merkmale können in [Guyon et al., 2006] und [Niemann, 1983] gefunden werden. Eine optimale Klassifizierung ist selbst unter Vernachlässigung der Rechenzeit und der Annahme von [Jähne, 2005], dass eine maximale Anzahl von Merkmalen eine minimale Fehlerwahrscheinlichkeit erreicht, durch das Peaking-Phänomen nicht möglich. [Handels, 2009] Beschreibung für das Phänomen ist, dass bei ständiger Hinzunahme von Merkmalen die Güte der Klassifizierung bis zu einem maximalen Wert steigt und anschließend absinkt. Eine Erklärung dafür ist, dass bei steigender Anzahl von Merkmalen auch die Anzahl der Systemparameter steigt. Um diese weiter erfolgreich zu ermitteln, müsste die Anzahl der Trainingsdaten exponentiell mit der Dimensionalität der Merkmale steigen, da sonst die Parameter größeren Schwankungen ausgesetzt sind. Dieses Problem ist auch bekannt als “curse of dimensionality“.

Um möglichst gute Ergebnisse zu erhalten wird eine Dimensionsreduktion vorgenommen. Dabei sollen durch Vorverarbeitung der zu analysierenden Objekte, Transformation von Merkmalen und der Identifizierung von unabhängigen Merkmalen die Effizienz der Klassifizierung gesteigert und die Anzahl von Merkmalen gering gehalten werden [Handels, 2009]. Die Ermittlung geeigneter Merkmale ist die Merkmalextraktion. Einen wichtigen Beitrag hierzu liefert in der Bildverarbeitung bereits die Segmentierung. Durch das Betrachten einzelner Bereiche eines Bildes, die eine logische Einheit darstellen, ist es möglich, eine Vielzahl von Merkmalen zu definieren, die über das gesamte Bild keinen Informationsgewinn erzeugen würden. Wie solche Merkmale wie Farbe, Form oder Fläche eines Objekts interpretiert und dargestellt werden, ist abhängig von den vorhandenen Bilddaten und der Problemstellung.

Um vorhandene Merkmale auf ihre Aussagekraft zu untersuchen, wird die Methode der

Merkmalsselektion verwendet. Dabei sollen aus der Menge aller Merkmale all die Merkmale ermittelt werden, die den höchsten Informationsgewinn liefern und für Analyse und Klassifikation am besten geeignet sind. Dies ist der Fall, wenn sich alle Objekte in unterschiedlichen Klassen in mindestens einem Merkmal unterscheiden [Handels, 2009]. In der Literatur sind viele unterschiedliche Methoden zur Merkmalseselektion zu finden. Dazu gehören die Principal Component Analysis (PCA) [Jolliffe, 2002], die lineare Korrelationsanalyse [Moll und Bundesfo, 2003] und die Merkmalsauswahl mittels Fisher-Diskriminante [Fisher, 1936].

Die Klassifizierung von großen Datenmengen durch Menschen ist durch die niedrige Geschwindigkeit und abweichende Klassifizierung aufgrund der subjektiven Wahrnehmung der Personen nicht praktikabel. Eine automatische Klassifizierung, die anhand der Merkmale eine Kategorisierung von Objekten vornimmt, ist meist nicht nur schneller, sondern liefert auch bessere Klassifizierungsergebnisse [Niemann, 1983]. Um solche Klassifizierungen erfolgreich automatisiert durchzuführen, können maschinelle Lernverfahren genutzt werden. Diese sind Optimierungsverfahren oder Regressionsverfahren, die zu einer gegebenen Datenmenge die bestmögliche Klassenteilung erzeugen, mit der Absicht, eine gute Generalisierung jenseits der Trainingsdaten zu erhalten. Der Trainingsdatensatz muss dafür alle berechneten Merkmale aller Objekte, und für einige Lernverfahren auch eine entsprechende Zuweisung zu einer Klasse enthalten. Die Algorithmen, welche eine Datenanalyse auf den Trainingsdaten durchführen, können in bestärkende Verfahren, unüberwachte Verfahren und überwachte Verfahren eingeteilt werden.

Bei den bestärkenden Verfahren werden einem Agenten, der das System repräsentiert, durch positive und negative Rückmeldung auf sein Verhalten in bestimmten Situationen, Verhaltensregeln beigebracht, um bei erneutem Eintreten der Situation diese möglichst erfolgreich zu bewältigen [Sutton und Barto, 1998].

Unüberwachte Verfahren können auf Datensätze ohne Klassenzuweisung angewandt werden. Diese schließen aus der Verteilung der Objekte im Merkmalsraum auf die möglichen Klassen. Dabei ist es möglich, die Anzahl der Klassen vorzugeben oder diese direkt durch die Merkmale zu bestimmen. Unüberwachte Klassifizierungsverfahren werden unter anderem in [Niemann, 1983] beschrieben.

Die überwachten Verfahren gehen davon aus, dass die Trainingsdaten und somit die gelernten Klassen, die zu erwartenden Eingangsdaten vollständig repräsentieren. Objekte können während der Klassifizierung nur in Klassen eingeordnet werden, die in den Trainingsdaten enthalten waren. Der Ablauf eines überwachten Verfahren ist nach Handels [Handels, 2009] in drei Phasen aufgeteilt.

1. Vorbereitungsphase: Aufteilung des Datensatzes in Trainingsdaten und Testdaten
2. Lernphase: Training des Klassifikators des genutzten Verfahrens und Validierung der Güte anhand der Testdaten
3. Anwendungsphase: Anwendung der Klassifizierung auf das gegebene Problem

Ein entsprechendes überwachtes Verfahren ist die Suport Vector Machine. Dieses und das Neuronale Netz werden im Folgenden vorgestellt.

2.3.1 Support Vector Machine

Die Support Vector Machine (SVM) ist ein maschinelles Lernverfahren zur Klassifizierung und Regressionsanalyse. Ziel der SVM ist es, anhand von Trennflächen im Merkmalsraum, die auf Basis der Trainingsdaten ermittelt wurden, Klassen zu definieren. Der Algorithmus wurde ursprünglich zur Mustererkennung entwickelt und ordnet in seiner einfachsten Form Daten einem binären Problem zu. In einem 2-dimensionalen Merkmalsraum ist die einfachste Trennfläche eine Gerade.
Die Entscheidung zu welcher Klasse ein Objekt gehört, muss bei einer maschinellen Klassifizierung anhand eines Klassifikators stattfinden. Eine mathematische Beschreibung kann eine Trennfunktion (Diskriminantenfunktion) liefern. Dabei wird geprüft auf welcher Seite der Trennfunktion ein Datum liegt. Die Dimensionalität des Merkmalsraums gibt dabei die Komplexität der Trennfunktion vor. Anhand eines linear klassifizierbaren binären Problems wie in (Abb.2.8) kann die Diskriminantenfunktion,

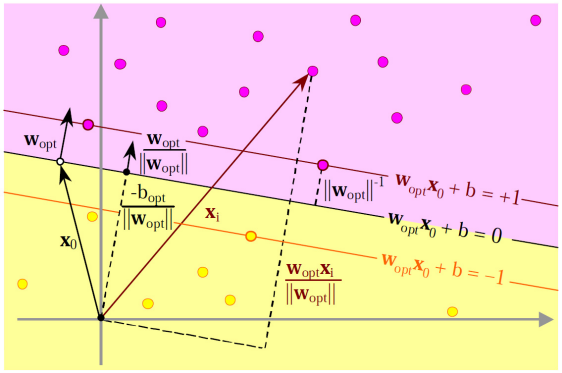


Abbildung 2.8: Linear separierbares Klassifizierungsproblem aus [HEINERT, 2010].

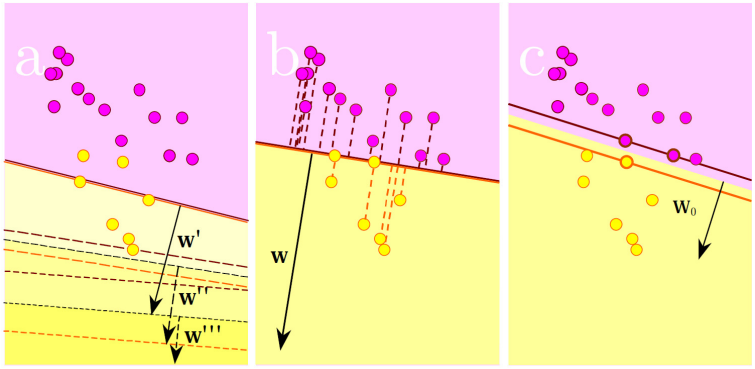


Abbildung 2.9: Einpassung der Hyperebene in den Merkmalsraum und Maximierung des Trennbereiches aus [HEINERT, 2010].

die als Hyperebene H definiert werden kann, als

$$H(x) = w^T x + b = \sum_{i=1}^{m} w_i x_i + b \quad \text{für m=2 im 2D-Fall} \tag{2.16}$$

beschrieben werden. Dabei ist w der Gewichtsvektor, b das Schwellwertgewicht oder bias und m die Anzahl der Merkmale. Befindet sich nun ein Datum ober- oder unterhalb der Hyperebene, wird es der entsprechenden Klasse zugeordnet. In der Trainingsphase der SVM wird nun mit Hilfe der Stützvektoren die Hyperebene so im Merkmalsraum angeordnet, dass ein möglichst großer Trennbereich entsteht, in dem kein Datum vorhanden ist (Abbildung 2.9).

Sind die Klassen im Merkmalsraum so verteilt, dass keine lineare Separierung möglich ist (Abb.2.10a), müssen die Objekte in einen höher dimensionalen Raum transformiert

werden (Abb.2.10b). Dadurch ist es möglich, die Separierung durch eine Hyperebene durchzuführen (Abb.2.10c). Eine Rücktransformation durch eine Kernelfunktion liefert nach dem Mercer-Theorem anhand der Hyperebene eine stetige, symmetrische und positiv definite Kernfunktion (Abb.2.10d-f). Der sogenannte Kernel-Trick ermöglicht es dadurch, einen linearen Klassifikator auf nicht linearklassifizierbare Daten anzuwenden [Heinert, 2010].

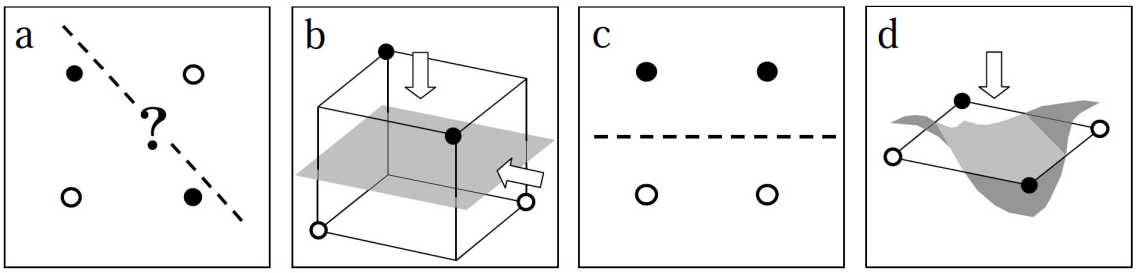


Abbildung 2.10: XOR-Problem a) im 2D- und b)3D-Merkmalsraums, c) ortogonale Darstellung des 3d-Merkmalsraum, d) Merkmalsraum mit Trennebene [Heinert, 2010].

2.3.2 Neuronale Netze

Künstliche Neuronale Netze (KNN) sind wie die SVM maschinelle Lernverfahren. KNN bilden in stark vereinfachter Form die Neuronalen Netze des menschlichen Gehirns und ihre Funktionsweise nach. Die Grundbausteine eines KNN sind Neuronen und Verbindungen. Neuronen sind einfache Recheneinheiten, die anhand deren Eingabe eine Ausgabe erzeugen (Abb.2.12). Verbindungen übertragen Informationen und können gewichtet sein, um verstärkend oder hemmend auf das Signal einzuwirken. Ein KNN ist eine je nach Anwendung festgelegte Struktur von Neuronen und Verbindungen. Dabei existiert eine Eingabeschicht, in der jedes Merkmal einer Klassifizierungsaufgabe als eine Verbindung mit einem Neuron verbunden ist. Die versteckte Schicht ist eine beliebig anzuordende Struktur deren Eingabeverbindung mit der Eingabeschicht und deren Ausgabeverbindung mit der Ausgabeschicht verbunden ist. Die möglichen Klassen der Klassifizierung werden in der Ausgabeschicht jeweils durch ein Neuron

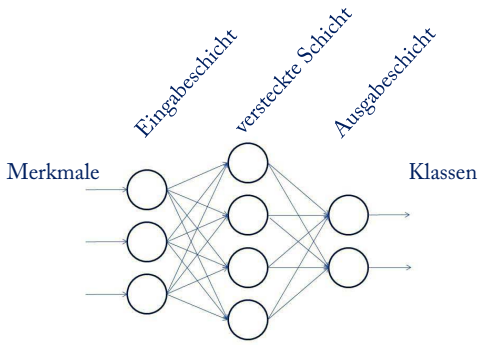


Abbildung 2.11: Beispielhafter Aufbau eines Neuronalen Netzes.

dargestellt (Abbildung 2.11).

Das Neuron erzeugt dessen Ausgabe durch die Propagierungsfunktion, der Aktivierungsfunktion und der Ausgabefunktion. Die Propagierungsfunktion erzeugt anhand aller eingehenden Daten der Verbindungen unter Berücksichtigung deren Verbindungsgewichte die Netzeingabe des Neurons. Ob ein Neuron aktiv ist und somit ein Signal aussendet, hängt davon ab ob der Schwellwert der Aktivierungsfunktion überschritten wird. Gängige Klassen der Aktivierungsfunktion sind lineare Aktivierungsfunktionen, Schwellwertfunktionen und sigmoide Funktionen. Ist ein Neuron aktiv, erzeugt die Ausgabefunktion den Wert, der an die verbundenen Neuronen weitergegeben wird. Häufig wird die Identitätsfunktion als Ausgabefunktion verwendet. Diese gibt den Wert der Aktivierungsfunktion weiter [Scherer, 1997].

In der Trainingsphase wird ein KNN entsprechend der zu lernenden Objekte verändert. Abhängig von der gewählten Trainingsmethode kann die Topologie des Netzes durch Hinzufügen oder Löschen von Verbindungen und Neuronen verändert werden. Zudem können die vorhandenen Parameter, wie die Gewichte der Verbindungen oder die Funktionen der Neuronen, angepasst werden. Eine Änderung der Neuronenfunktionen ist aufwendig zu implementieren und wird daher eher selten verwendet. Gute Ergebnisse können durch Änderung der Topologie erreicht werden. Durch die Anpassung der Verbindungsgewichte kann das Potential von KNN bei vergleichsweise geringem Aufwand ausgenutzt werden. KNN können sowohl unüberwacht, überwacht und durch bestär-

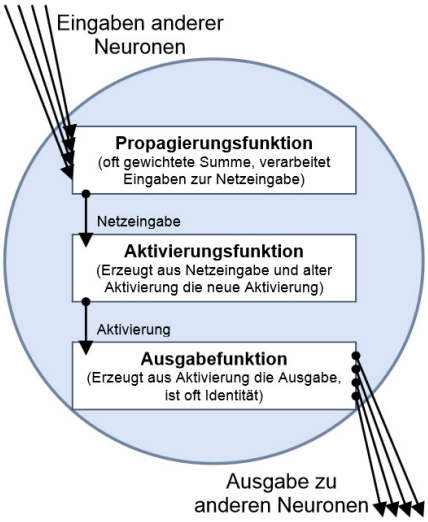


Abbildung 2.12: Aufbau eines Neurons aus [Kriesel, 2008].

kende Methoden trainiert werden. Dies ermöglicht es, KNN für viele unterschiedliche Anwendungen einzusetzen [Kriesel, 2008].

Ziel der Klassifizierung in dieser Arbeit ist es, festzustellen, wie gut die Texturmerkmale zur Identifizierung von bekannten Texturen geeignet sind. Eine Datenbank mit 1281 Bildern von Texturen aus 57 Klassen wurde dazu von einer SVM und einem Neuronalen Netz gelernt und getestet. Um weitere Funktionalitäten wie die Klassifizierung von unbekannten Klassen oder das nachträgliche Lernen von Klassen zu ermöglichen, wäre der Einsatz eines wissensbasierten Systems oder die Einführung einer Fuzzylogik nötig. Dies übersteigt jedoch den Umfang dieser Arbeit und wird daher nicht betrachtet.

Kapitel 3

Methoden

Die Aufgabe des erstellten Algorithmus ist es, anhand von RGB-D-Daten homogene Bereiche im erhaltenen Bild zu identifizieren und diese in eine Texturklasse einzuordnen.

3.1 Konzept

Zur Erfüllung der Anforderungen, die an den Algorithmus gestellt werden, durchläuft dieser die in Abbildung 3.1 aufgeführten Prozeduren. Zu Beginn wird die Umgebungsaufnahme anhand der Tiefendaten segmentiert und somit alle glatten Flächen identifiziert. Diese Segmente werden für eine einheitliche Verarbeitung in eine normalisierte Ansicht transformiert. Dazu wird mithilfe der Tiefendaten jeweils eine entsprechende Homographiematrix bestimmt. Die transformierten Segmente werden anhand ihrer Textureigenschaften weiter segmentiert, um unterschiedliche Texturen auf den Ebenen zu erkennen. Somit werden Segmente mit homogener Textur in einer normalisierten Ansicht gewonnen, auf denen die Texturattribute berechnet werden. Abschließend werden die Segmente anhand der Attribute klassifiziert. Ein maschinelles Lernverfahren, das mit den erzeugten Attributwerten über die Texturdatenbank A.3 trainiert wurde,

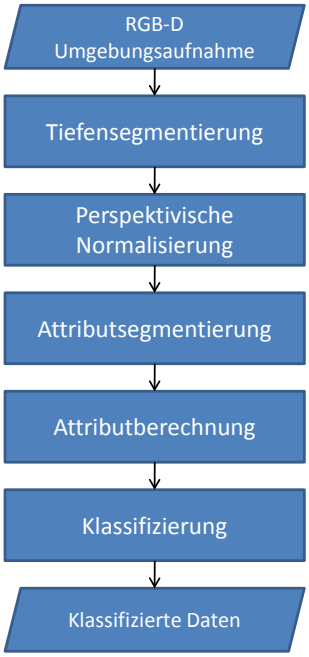


Abbildung 3.1: Ablauf des Algorithmus.

wird dazu eingesetzt.

3.2 Systemumgebung

Das hier entwickelte System soll auf dem Robotersystem Care-O-bot® 3 des Fraunhofer IPA eingesetzt werden. Der Roboter und die vorhandene Software werden im Folgenden kurz vorgestellt.

3.2.1 Care-o-bot® 3

Der Care-O-bot® (Abbildung 3.2) ist ein mobiler Serviceroboter, der bereits in der dritten Generation vom Fraunhofer IPA entwickelt wird. Die Entwicklungsplattform ist

Abbildung 3.2: Care-O-bot®3 [FRAUNHOFER, 2013].

für ein häusliches Umfeld konzipiert und bietet mittlerweile das Potential, in Alltagsumgebungen eingesetzt zu werden. Dabei soll der Care-O-bot® alltägliche Aufgaben übernehmen, um älteren und behinderten Menschen ein längeres und eigenständiges Leben in ihrem eigenen Zuhause zu ermöglichen. Der Roboter ist unter anderem in der Lage Hol- und Bringdienste auszuführen und kann in Notfällen Unterstützung bieten, in dem er als Schnittstelle zu einer Notfallzentrale genutzt wird [FRAUNHOFER, 2013].

Insgesamt wurden bisher acht Exemplare des Care-O-bot® gebaut. Alle Roboter sind ähnlich zu Abbildung 3.3 aufgebaut. Je nach Ausstattung können sich die Sensoren und Aktoren unterscheiden. Sensoren sind Systeme, die physikalische Größen in der Umgebung erfassen und in elektronische Signale umwandeln. Der Care-O-bot® verfügt über Laserscanner, Kameras, Mikrofon und ein Touchscreen. Aktoren dagegen wandeln elektrische Energie in andere Energieformen. Entsprechende Aktoren des Roboters sind die Plattform, Arm, Torso, Tablett, Lautsprecher und die Kopfachse.

Informationen über die Umgebung werden durch die Sensoren detektiert. Der Laserscanner erkennt Objekte im Raum und deren Entfernung und ermöglicht somit die Bahnplanung. Für Objektmanipulationen werden dagegen die Daten der Kamerasysteme genutzt. Für die Interaktion mit einer Person stehen das Mikrofon und der

Touchscreen zur Verfügung.

Die Aktoren ermöglichen eine Interaktion mit der Umgebung. Die omnidirektionale Plattform umfasst vier gelenkte und angetriebene Räder, wodurch flexibel im Raum navigiert werden kann. Durch den Roboterarm mit sieben Freiheitsgraden und einer 3-Finger-Hand ist es möglich, eine Vielzahl von Objektmanipulationen durchzuführen. Der Sensorkopf kann durch Bewegen des Torsos und der Kopfachse in eine für die Kamerasysteme optimale Position gebracht werden. Das Tablett ermöglicht eine Interaktion mit Personen, indem diese den verbauten Touchscreen bedienen oder darauf abgestellte Gegenstände entgegen nehmen.

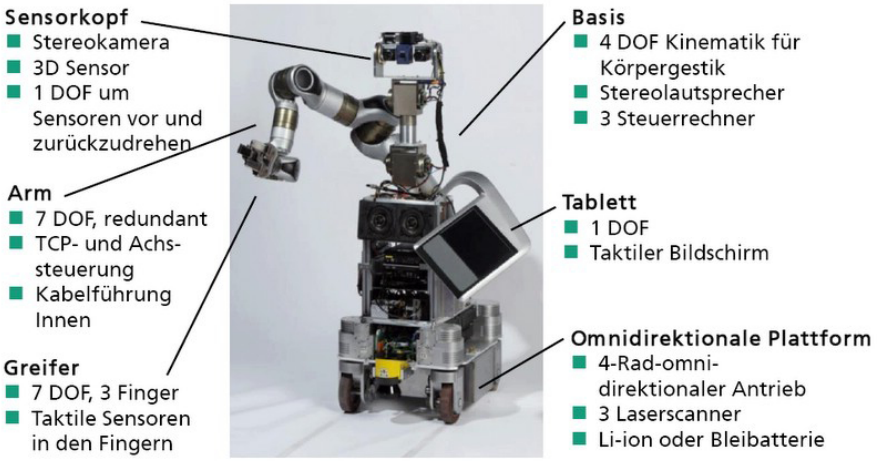


Abbildung 3.3: Care-O-bot® 3 Hardwarekomponenten [Fraunhofer, 2013].

3.2.2 ROS

Das Robot Operating System (ROS) ist ein Framework zur Entwicklung von Software für eine Vielzahl von Robotern. Entwickelt wird ROS von der Open Source Robotics Foundation und stellt unter anderem Bibliotheken, Werkzeuge und Gerätetreiber zur Verfügung [ROS, 2014a]. Die Software wird in Paketen (engl. Packages) verwaltet. Entsprechend der Funktionalität werden die Pakete in Stapel (engl. Metapackages) zusammengefasst. Programme werden in ROS als Knoten (engl. Nodes) dargestellt.

Knoten enthalten Algorithmen, die auf dem Robotersystem ausgeführt werden, wie die Steuerung von Hardwarekomponenten oder die Bahnplanung. Über peer-to-peer Verbindungen wird ein Prozessnetzwerk zwischen den Knoten aufgebaut, welches die Verteilung der Software über mehrere Computer ermöglicht. Das Netzwerk wird von einem zentralen Knoten, dem Meister (engl. Master), verwaltet. Um Daten zu veröffentlichen oder zu abonnieren meldet der Knoten einen Verbreiter (engl. Publisher) oder Abonnent (engl. Subscriber) am Meister an, welcher die Verwaltung der Verbindungen übernimmt. Für die Kommunikation zwischen den Knoten können ROS-Nachrichten oder ROS-Services genutzt werden. Die ROS-Nachrichten werden unabhängig von Abonnenten veröffentlicht. Ein Knoten kann gleichzeitig mehrere Nachrichten abonnieren und veröffentlichen. Der ROS-Service ist eine Verbindung von zwei Knoten, die als Dienst (engl. Service) und Nutzer (engl. Client) identifiziert werden. Die Kommunikation ist bidirektional und ermöglicht es dem Nutzer bestimmte Dienste vom Service-Knoten anzufordern [ROS, 2014b].

3.2.3 OpenCV

OpenCV (Open Source Computer Vision Library) ist eine freie Programmbibliothek für maschinelles Sehen und Lernen. Es werden eine Vielzahl von optimierten und etablierten Algorithmen bereitgestellt, sowie aktuelle Erkenntnisse aus der Forschung integriert [OpenCV, 2014a].

3.3 Einbindung in das System

Für eine Nutzung des Systems auf dem Serviceroboter Care-O-bot® muss die vorhandene Hardware in die Konzeption einbezogen werden und definierte Schnittstellen der Software vorhanden sein.

3.3.1 Eingabe

Der Care-O-bot® ist, je nach Ausstattung, mit einer Microsoft Kinect oder Asus Xtion PRO LIVE ausgestattet. Diese Kamerasysteme stellen RGB-D Daten in identischen Formaten zur Verfügung. Für die Verarbeitungsprozedur des Algorithmus ist es wichtig, dass die ermittelten 3D-Punkte den Pixeln im Farbbild zugeordnet werden können. Hierfür werden XYZRGB-Punktewolken verwendet. Diese sind Arrays, in denen die Pixel nach dem Bild-Raster angeordnet sind. Somit ist es möglich für jeden Punkt anhand dessen Position die u und v Koordinaten in der Bildmatrix zu bestimmen und gleichzeitig die metrischen Werte der Tiefenmessung sowie die Farbwerte zu erhalten.

3.3.2 Ablauf

Das System setzt sich aus den Knoten zur Steuerung des Kamerasystems und der Tiefensegmentierung zusammen sowie einem Knoten, der die restlichen Prozeduren aus Kapitel 3.1 beinhaltet. Ist die Kamerasteuerung gestartet, werden die aufgenommenen Punktwolken veröffentlicht. Die Tiefensegmentierung (Kapitel 3.4.1) abonniert die Punktewolke des Kamerasystems, segmentiert diese und veröffentlicht die Ergebnisse in der entsprechenden Nachricht. Der dritte Knoten abonniert wiederum die Nachricht der Tiefensegmentierung. Jedes erhaltene Segment wird perspektivisch normalisiert (Kapitel 3.5.3). Die aufbereiteten Segmente werden anhand von Textureigenschaften weiter segmentiert (Kapitel 3.4.2) und auf jedem erhaltenen Segment die Texturattribute berechnet (Kapitel 3.6). Nach Abschluss der Klassifizierung (Kapitel 3.7), werden die Segmente in die ursprüngliche Ansicht transformiert, um die entsprechenden Bereiche im Originalbild den Klassifizierungsergebnissen zuordnen zu können.

3.3.3 Ausgabe

Die Ergebnisse des Algorithmus werden dem ROS System zur Verfügung gestellt. Dazu wird eine Nachricht generiert, in der die ermittelten Segmente mit dem Ergebnis der Klassifizierung enthalten sind. Über den publisher werden die Nachrichten veröffentlicht. Benötigt ein Knoten die Informationen, kann er durch Abonnieren der Nachricht diese erhalten.

3.4 Segmentierung

3.4.1 Tiefensegmentierung

Die hier verwendete Tiefensegmentierung ist Teil eines bereits abgeschlossenen Projekts des Fraunhofer IPA zur Klassifizierung von Objektoberflächen in Punktwolkendaten und wird im Folgenden kurz beschrieben. Eine genaue Beschreibung ist in [Arbeiter et al., 2014] zu finden.

Für die Segmentierung werden Normalen auf jeden Punkt der Punktwolke berechnet. Diese liefern lokale Informationen zur Orientierung der Oberfläche. Führt man ähnliche Normalen in Segmente zusammen, ist es möglich, gleichartige Oberflächen zu detektieren.

Normalenberechnung

Die von einem Tiefensensor erhaltene Punktwolke speichert die Daten der Punkte in einer Struktur, die es ermöglicht, die Punktwolke in eine 2D-Matrix zu überführen. Anhand der erhaltenen Nachbarschaftsbeziehungen kann für jeden Punkt ein lokaler Normalenvektor der Form

$$n_q = (q_i - q_q) \times (q_j - q_q) \tag{3.1}$$

ermittelt werden. Hierbei ist n_q der Normalenvektor, der sich aus dem Kreuzprodukt der Vektoren von Punkt q_q zu dessen benachbarten Punkten q_i und q_j ergibt. Die Auswahl der Nachbarschaftspunkte findet anhand einer Maske statt, welche die Entfernung zu dem betrachteten Pixel von Punkt q_q, sowie die Entfernung unter den Nachbarschaftspixeln festlegt. Zudem darf der Tiefenabstand nicht einen bestimmten Schwellwert überschreiten. Die Berechnung der Normalen anhand der Umgebung führt zu fehlerhaften Ergebnissen an den Übergängen zwischen zwei sich berührenden Flächen, da auch Punkte mit einbezogen werden, die nicht mehr zu der betrachteten Fläche gehören. Dadurch entstehen anstelle von Kanten abgerundete Übergänge. Dies kann durch eine vorherige Detektierung der Konturen und einer angepassten Normalenberechnung in der Nähe dieser verhindert werden.

Segmentierung der Normalen

Die Normalen werden durch ein Region-Growing Verfahren (Kapitel 2.1.3) segmentiert. Dabei kommen zwei Ähnlichkeitskriterien zum Einsatz.

1. Geringer Tiefenabstand: Es sollen nur die betrachteten Punkte q hinzugefügt werden, deren Tiefenwert q_z einen maximalen Abstand d_{th} zum Tiefenwert p_z eines benachbarten Punktes p des Segments nicht unterschreitet

$$||q_z - p_z|| < d_{th} \tag{3.2}$$

2. Ähnliche Normalenrichtung: Die Normale des betrachteten Punktes n_q wird mit der gemittelten Normalen des Segments verglichen, die sich als

$$\bar{n} = \frac{1}{k} \sum_k n_k \tag{3.3}$$

berechnet und k die Punkte des Segments indiziert. Als Kriterium dient der Winkel zwischen $\bar{n}$ und n_q welcher den Schwellwert α_{th} nicht überschreiten darf.

$$\arccos(\bar{n} \cdot n_q) < \alpha_{th} \tag{3.4}$$

Ist die Segmentierung abgeschlossen, werden die einzelnen Segmente in eine normalisierte Bildansicht transformiert (Kapitel 3.5) um bei der weiteren Bearbeitung durch die Split and Merge Segmentierung (Kapitel 3.4.2) und der Berechnung der Texturfeatures (Kapitel 3.6) optimale Ergebnisse zu erhalten.

3.4.2 Split and Merge

Für die spätere Klassifizierung ist es wichtig, dass die Texturattribute möglichst korrekt berechnet werden. Voraussetzung dafür ist, dass die Segmente einheitliche Texturen aufweisen. Der Split and Merge Algorithmus (Kapitel 2.1.1) bietet durch die Überprüfung eines Homogenitätskriteriums über einer Region die Möglichkeit, anhand von Textureigenschaften eine Segmentierung vorzunehmen. Durch die Nutzung der Texturattribute als Homogenitätskriterium wird die Segmentierung auf die Anforderungen der Klassifizierung ausgelegt.

Split Phase

Die Split Phase hat das Ziel, das betrachtetet Bild so weit in kleine Bereiche zu unterteilen, bis alle entstandenen Regionen einem Homogenitätskriterium genügen. Dazu teilt das top-down Verfahren das Originalbild und die aus der Aufteilung entstehenden Regionen solange in vier gleich große Unterregionen auf, bis das Kriterium für jede Region erfüllt ist (Abbildung 3.4).

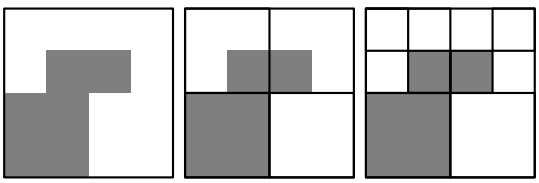


Abbildung 3.4: Split Prozess mit Aufteilung des gesamten Bildes sowie weitere Unterteilung zweier Unterregionen.

Zur Feststellung der Homogenität werden zwei Kriterien überprüft. Anhand der Standardabweichung über die Farbe wird ermittelt, ob der betrachtete Bereich eine einheitliche Farbe aufweist und somit homogen ist. Das zweite Kriterium überprüft, ob eine Textur vorhanden ist.

Die Standardabweichung wird für die Kanäle R, G und B des entsprechenden Farbraums ermittelt. Dabei wird die gesamte betrachtete Bildregion der Größe $N \times M$ in die Berechnung einbezogen.

$$\sigma = \sqrt{\frac{\sum_{i=1}^{N} \sum_{j=1}^{M} (x_{ij} - \mu)^2}{NM}} \tag{3.5}$$

Entsprechend dem Kanal ist x_{ij} der Rot, Grün oder Blau Wert an der Position (i, j) und μ der Mittelwert der Farbwerte. Übersteigt die Standardabweichung aller Kanäle nicht einen Schwellwert, wird von einer homogenen Bildregion ausgegangen.

Das zweite Kriterium ermittelt homogene Texturen mit Hilfe des LBP-Operators (Kapitel 2.2.2). Der betrachtete Bildbereich wird, wie bei einem Splitvorgang, in vier Unterbereiche aufgeteilt. Auf jedem Unterbereich wird das entsprechende LBP-Histogram ermittelt. Ähneln sich alle Histogramme weisen die Bereiche ähnliche Texturen auf und der gesamte Bildbereich ist homogen. Dies wird durch den Chi-Quadrat-Homogenitätstest festgestellt. Es werden die Stichproben in der Kreuztabelle 3.1 aufgetragen. Dabei sind die Einträge $H_n(m)$ die Werte an Position m im Histogramm des Bildbereichs mit der Nummer n. Die Summe aller Werte aus der Kreuztabelle werden als N dargestellt.

Die Kreuztabelle ermöglicht es, die erwartete Häufigkeit zu bestimmen

$$\hat{H}_{n,m} = \frac{h_{n,\bullet} \cdot h_{\bullet,m}}{h_{\bullet,\bullet}}. \tag{3.6}$$

Durch den Vergleich der erwarteten Häufigkeit mit der beobachteten Häufigkeit wird die Prüfgröße χ^2 ermittelt.

$$\chi^2 = \sum_n \sum_m \frac{(H_{n,m} - \hat{H}_{n,m})^2}{\hat{H}_{n,m}} \tag{3.7}$$

Tabelle 3.1: Kreuztabelle des Chi-Quadrat-Homogenitätstest.

	x_1	... x_i ...	x_{10}	
Reg 1	$H_1(1)$	... $H_1(i)$ ...	$H_1(10)$	$\sum_{j=1}^{10} H_1(j) = H_{1\bullet}$
Reg 2	$H_2(1)$	... $H_2(i)$ ...	$H_2(10)$	$\sum_{j=1}^{10} H_2(j) = H_{2\bullet}$
Reg 3	$H_3(1)$	... $H_3(i)$ ...	$H_3(10)$	$\sum_{j=1}^{10} H_3(j) = H_{3\bullet}$
Reg 4	$H_4(1)$	... $H_4(i)$ ...	$H_4(10)$	$\sum_{j=1}^{10} H_4(j) = H_{4\bullet}$
	$\sum_{j=1}^{4} H_j(1) = H_{\bullet 1}$	$\sum_{j=1}^{4} H_j(i) = H_{\bullet i}$	$\sum_{j=1}^{4} H_j(10) = H_{\bullet 10}$	N

Zur Bewertung der Hypothese wird die Prüfgröße mit dem kritischen Wert verglichen. Dieser wird anhand des Quantils der Chi-Quadrat-Verteilung bestimmt. Das Quantil ist abhängig von dem Freiheitsgrad $df = (n-1)(m-1)$ sowie dem Signifikanzniveau α. Übersteigt die Prüfgröße die kritische Größe wird die Hypothese verworfen und mindestens zwei Bildbereiche unterscheiden sich signifikant. Dies ist der Fall wenn

$$\chi^2 > \chi^2_{(n-1)(m-1),1-\alpha} \tag{3.8}$$

eintritt.

Die Beiden Kriterien bilden gemeinsam das Homogenitätskriterium

$$H = \begin{cases} \text{wahr} & \text{wenn } \sigma_R + \sigma_G + \sigma_B < \vartheta \quad \text{oder} \quad \chi^2 > \chi^2_{(n-1)(m-1),1-\alpha} \\ \text{falsch} & \text{sonst} \end{cases} \tag{3.9}$$

welches durch die Unterschreitung der Schwelle ϑ oder durch die Bestätigung des Chi-Quadrat-Test die Homogenität der Region belegt und den Splitvorgang für diese beendet.

Der Schwellwert des ersten Kriteriums sowie das Signifikanzniveau des Quantils des zweiten Kriteriums werden so gewählt, dass ein strenges Homogenitätskriterium entsteht. Dies ermöglicht es, diagonale Kanten im Bild durch stark aufgeteilte Regionen

annähernd genau darzustellen. Die größere Anzahl von Regionen nach der Split Phase wird durch den Merge Prozess auf Kosten einer erhöhten Laufzeit kompensiert.

Für den weiteren Algorithmus ist es wichtig, die Nachbarschaftsbeziehungen der Regionen zu kennen. Dazu werden diese als Blätter eines Quadtree gespeichert. Ist dieser balanciert, d.h. die Tiefe aller Blätter im Baum unterscheidet sich höchstens um eins, können Nachbarschaftsbeziehungen direkt aus dem Baum gewonnen werden. Tritt ein unbalancierter Baum auf, wurde mindestens eine homogene Region gefunden die wesentlich größer ist als die kleinsten Regionen. Um einen balancierten Baum zu erhalten, müssten die größeren Regionen trotz ihrer Homogenität weiter unterteilt werden. Um dies zu vermeiden, werden die Nachbarschaftsstrukturen aus dem Quadtree bestimmt und die entsprechenden Informationen den jeweiligen Knoten zugeordnet. Die Knoten des Quadtree besitzen eine eindeutige ID, deren Länge sich aus der Tiefe des Knotens ergibt. Wie in Abbildung 3.5 dargestellt, baut sich die ID, beginnend nach der Wurzel mit der ID "0", aus der Anordnung der vorherigen Regionen auf. Dabei wird, beginnend bei 1, von links nach rechts zeilenweise durchgezählt.

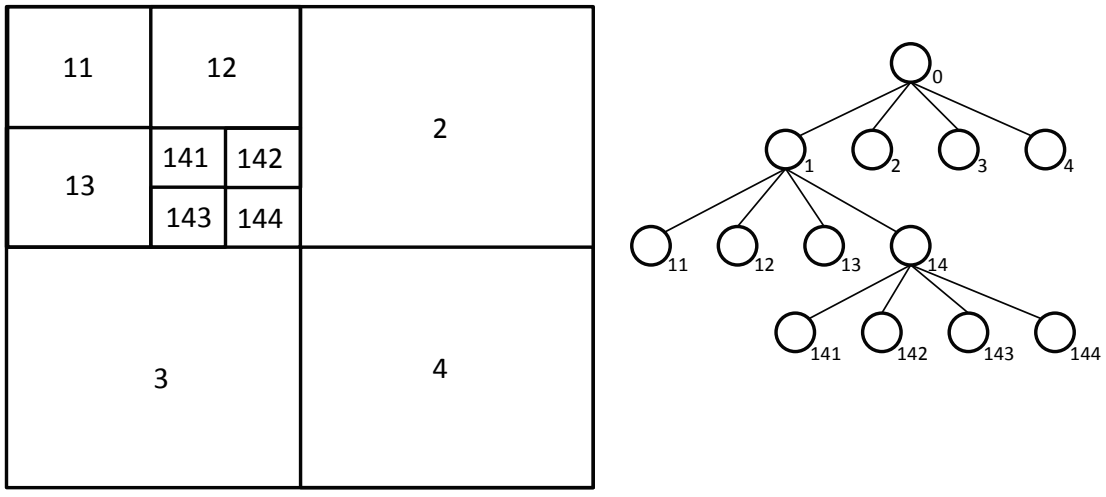


Abbildung 3.5: Links: Anordnung der ID´s im Bild. Rechts: Darstellung als Quadtree.

Durch die Unterteilung des Knotens 14 wird der Baum unbalanciert und Nachbarschaftsbeziehungen wie z.B. zwischen 2 und 142 sowie 144 können nicht mehr direkt

aus der ID des Knotens erkannt werden. Um diesem Problem zu begegnen, ist jeder Knoten eine Datenstruktur, in der Vater- und Kinderknoten angegeben sind, sowie die Möglichkeit existiert, für jede Seite die Nachbarn zu hinterlegen. Zudem wird die Position im Bild sowie die Größe der Region in der Struktur abgelegt. Nach Beendigung des Splitvorgangs durchläuft ein Algorithmus den Baum und aktualisiert für jeden Knoten die Informationen über die Nachbarschaftsbeziehungen. Der Algorithmus zur Bestimmung aller direkten Nachbarn baut darauf auf, dass bei Unterteilung der Region, vier Unterregionen entstehen, die jeweils mit zwei anderen Unterregionen benachbart sind (blaue Regionen in Abbildung 3.6). Die beiden anderen Seiten der Unterregion weisen dagegen Nachbarn auf, die bereits mit der ursprünglichen Region benachbart waren (gelbe und graue Regionen in Abbildung 3.6).

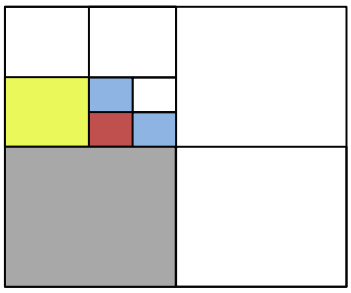


Abbildung 3.6: Nachbarn über drei ebenen des Quadtree. Rot: Betrachtete Region. Blau: Nachbarn der selben Ebene. Gelb und Grau: Nachbarn einer jeweils höheren Ebene.

Wie die Nachbarn dabei zugeordnet werden, hängt von der Position der Unterregion in der ursprünglichen Region ab. Der Quadtree wird rekursiv, beginnend bei der Wurzel, durchschritten. Für jeden Knoten wird der Algorithmus einmal durchlaufen. Übergeben werden dabei die ID des zu untersuchenden Knotens sowie die Nachbarn der jeweiligen Seite des Vaterknotens. Anhand der ID wird die Position in der Ursprungsregion ermittelt und die entsprechenden Nachbarn zugewiesen. Um sicherzustellen, dass die Regionen, die mit dem Vaterknoten benachbart waren, auch mit der Unterregion benachbart sind, wird geprüft, ob beide Regionen sich an einer Seite berühren. Dies ist nötig, da es sonst zu fehlerhaften Zuweisungen kommt, wenn die Nachbarregion stärker

unterteilt ist als die aktuell betrachtete Region. Ist somit sichergestellt, dass die korrekten Nachbarn eingetragen sind, werden deren Nachbarschaftsbeziehungen aktualisiert. Dazu wird die ID der Ursprungsregion entfernt und die neue ID der Unterregion eingefügt. Ist der Knoten kein Blatt des Quadtree, ruft sich der Algorithmus selbst vier mal auf und übergibt jeweils die ID eines Kindes sowie die Nachbarschaftsbeziehungen des Vaterknotens. Sobald alle Blätter erreicht wurden, ist der Algorithmus beendet. Der Ablauf wird in Algorithm 1 dargestellt.

Merge Phase

Nach Abschluss der Split Phase müssen gleichartige, benachbarte Bereiche zusammengefügt werden. Als Homogenitätskriterium dienen hier die verwendeten Texturfeatures aus Kapitel 3.6. Die teilweise sehr kleinen Regionen nach dem Split Vorgang werden in einem ersten Merge Durchlauf anhand der Farbmerkmale $F_{Merkmal_{in}}$ mit deren direkten Nachbarn zusammengefügt. Diese Merkmale werden durch Statistiken der 1. Ordnung ermittelt und sind somit bereits bei kleinen Regionen aussagekräftig. Durch die Verwendung des HSV-Farbraumes muss allerdings sichergestellt sein, dass die Dunkelstufe (V-Wert) einen minimalen Wert erreicht, um den Farbwert (H-Wert) nutzen zu können, da dieser sonst instabil ist. Das Homgenitätskriterium ist erfüllt, wenn die Summe der Beträge der Differenzen des Mittelwerts und der Varianz der S- und V-Werte, sowie des Wertes von colorfulness der beiden Regionen, nicht den Schwellwert übersteigen.

$$X_{Schwelle} > \sum_{n=1}^{5} ||F_{Merkmal_{in}} - F_{Merkmal_{jn}}|| \tag{3.10}$$

Hierbei ist $F_{Merkmal_{in}}$ das Farbmerkmal n der Region i. Stimmen die dominanten Farben in den verglichenen Regionen überein, wirkt sich das vermindernd auf die Summe der Differenzen aus. Aus den resultierenden Regionen ist es durch deren größere Fläche möglich, aussagekräftige Informationen anhand der Strukturmerkmale zu erhalten. Wie im vorhergehenden Merge Durchlauf werden zwei benachbarte Regionen verglichen. Das Homogenitätskriterium ist erfüllt, wenn die Summe der Beträge der

Algorithm 1 Nachbarschaftsbestimmung

1: **procedure** GETNEIGHBOR
2: $N \leftarrow$ Obere Nachbarn des Vaters
3: $E \leftarrow$ Rechte Nachbarn des Vaters
4: $S \leftarrow$ Untere Nachbarn des Vaters
5: $W \leftarrow$ Linke Nachbarn des Vaters
6: $id \leftarrow$ ID des zu bearbeitenden Knotens
7: $i \leftarrow$ Letzte Ziffer von id
8: $r \leftarrow$ Lade Struktur id
9: **if** $i = 1$ **then** r.E = ID + "2", r.S = ID + "3", r.N = N, r.W = W
10: **else if** $i = 2$ **then** r.E = ID + "1", r.S = ID + "4", r.N = N, r.W = E
11: **else if** $i = 3$ **then** r.E = ID + "1", r.S = ID + "4", r.N = S, r.W = W
12: **else if** $i = 4$ **then** r.E = ID + "2", r.S = ID + "3", r.N = S, r.W = E
13: **end if**
14: **for all** Nachbarn von Vaterknoten **do**
15: **if** Nachbar und Region kein gemeinsamer Rand **then** Lösche Nachbarn
16: **end if**
17: **end for**
18: **for all** Überprüften Nachbarn **do** Füge ID der Region als Nachbar bei Nachbarregion ein. Lösche ID von Vaterknoten.
19: **end for**
20: **if** Anzahl der Kinder von r = 4 **then**
21: Führe GETNEIGHBOR für jedes Kind des Knoten r aus und übergebe N, E, S und W.
22: **end if**
23: **end procedure**

Differenzen der einzelnen Strukturmerkmale $S_{Merkmal_{in}}$ nicht den Schwellwert überschreitet.

$$X_{Schwelle} > \sum_{n=1}^{12} ||S_{Merkmal_{in}} - S_{Merkmal_{jn}}|| \tag{3.11}$$

Die Verwendung der Texturmerkmale zur Identifizierung homogener Bereiche ist vorteilhaft für die spätere Klassifizierung, da somit sichergestellt ist, dass die Segmente nur aus einheitlichen Texturen im Sinne des gelernten Klassifizierers bestehen.

3.5 Bildtransformation

Texturen werden durch Linien, Muster und die Anordnung von Primitives charakterisiert. Ein veränderter Blickwinkel auf eine Textur kann die Wahrnehmung dieser Eigenschaften verfälschen. Durch unterschiedliche Entfernungen der Kamera zu einer Textur ist es ebenfalls möglich, dass die Größe und Häufigkeit von Primitives im dargestellten Bild sich verändert. Für einheitliche Klassifizierungsergebnisse ist es daher nötig den Blickwinkel und die Entfernung anzupassen und die betrachteten Texturen in eine normalisierte Ansicht zu überführen.

3.5.1 Prinzipal component analysis

Die Berechnung der Transformation einer durch die Tiefensegmentierung (Kapitel 3.4.1) gefundenen Oberfläche benötigt Informationen zu deren Lage. Anhand der Principal Component Analysis (PCA) ist es möglich, die orthogonal zueinander stehenden Hauptkomponenten dieser Oberfläche und somit deren Lage in dem Kamerakoordinatensystem zu ermitteln. Die PCA analysiert Datensätze und ist ein Verfahren der multivariaten Statistik. Dabei ist die PCA eine Projektion in einen Raum, dessen Dimensionen die ursprünglichen Daten mit maximaler Varianz darstellen. Die Daten mit der größte Varianz stellen dabei die erste Dimension dar während die kleinste Varianz für die letzte Dimension steht. Dies wird realisiert, indem korrelierte Variablen

des hochdimensionalen Datensatzes durch nicht korrelierte Variablen ersetzt werden. Dazu müssen neue, auf den Datensatz angepasste Basisvektoren ermittelt werden, auf die der Datensatz abgebildet wird. Durch die Basisvektoren wird ein neuer Eigenraum aufgespannt. Dadurch werden die Daten so in einem gedrehten Basiskoordinatensystem dargestellt, dass der neue Datensatz eine möglichst große Varianz aufweist. Diese beschreibt den Informationsgehalt der Variablen. Mit der Reduzierung der Variablen ist ein Informationsverlust verbunden. Eine sinnvolle Interpretation wird hierdurch nicht beeinträchtigt.

3.5.2 Koordinatentransformation

Anhand der durch die PCA ermittelten Hauptkomponenten ist es möglich eine Transformation der 3D Oberflächenpunkte zwischen dem Kamerakoordinatensystem und der Tangentialebene des 3D-Segmentes zu ermitteln. Hierfür kann die Translation und Rotation genutzt werden.

Translation

Eine Translation ist eine affine Abbildung eins Koordinatensystem p in ein Koordinatensystem p' um einen fest gewählten Translationsvektor t (Abbildung 3.7):

$$p' = p + t, \quad t \neq 0 \tag{3.12}$$

Dabei hat der Translationsvektor im $\Re^3$ die Form $t = (t_x, t_y, t_z)^T$. Die Translation kann auch als Verschiebung der abgebildeten Objekte innerhalb eines Koordinatensytems betrachtet werden.

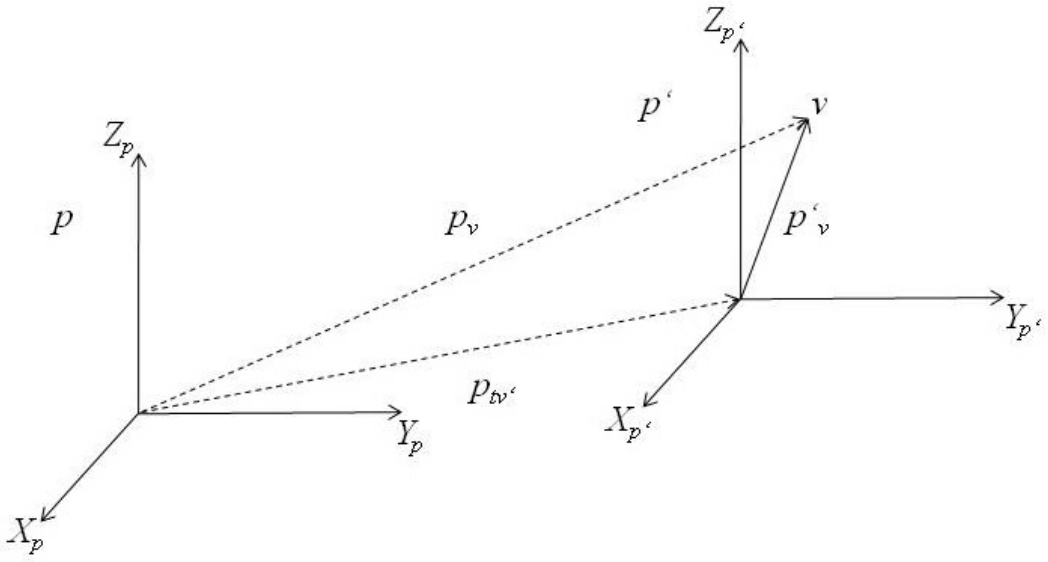


Abbildung 3.7: Translation eines Vektors v.

Rotation

Die Rotation oder Drehung ist eine lineare Abbildung die durch einen Winkel um einen Punkt oder eine Achse des Koordinatensystems gegeben ist.

$$p' = Rp \tag{3.13}$$

Dabei ist R die Abbildungsmatrix oder Rotationsmatrix. Diese ist eine orthonormale Matrix deren Spaltenvektoren Einheitsvektoren sind, die orthogonal zueinander stehen und die Eigenschaften $R^{-1} = R = R^T$ und $det(R) = 1$ aufweisen. Einfache Rotationen werden um eine feste Richtungsachse oder den Ursprung gedreht. Die Dimension des Koordinatensystems gibt die Anzahl der möglichen Richtungsachsen, um die gedreht werden kann, an. Somit kann im $\Re^3$ um die Achsen X, Y, Z gedreht werden. Die Rotationsmatrizen unterscheiden sich aufgrund der unterschiedlichen Abbildungsvorschriften.

Daher erhalten wir die Rotationsmatrizen R_x, R_y und R_z für den Winkel α:

$$R_{x,\alpha} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{pmatrix} \tag{3.14}$$

$$R_{y,\alpha} = \begin{pmatrix} \cos(\alpha) & 0 & \sin(\alpha) \\ 0 & 1 & 0 \\ -\sin(\alpha) & 0 & \cos(\alpha) \end{pmatrix} \tag{3.15}$$

$$R_{z,\alpha} = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.16}$$

Diese Rotationen können verkettet werden, um durch eine Rotationsmatrix mehrere Rotationen durchführen zu können. Dafür müssen die Rotationsmatrizen in der Reihenfolge der Ausführung von links multipliziert werden. In der Robotik ist es üblich die Rotationen in der Reihenfolge Rollen-Nicken-Gieren anzuwenden. Daraus ergibt sich die Rotationsmatrix

$$R_s = R_{z,\gamma} \cdot R_{y,\beta} \cdot R_{x,\alpha} \tag{3.17}$$

$$R_s = \begin{pmatrix} \mathrm{C}\alpha\mathrm{C}\beta & \mathrm{C}\alpha\mathrm{S}\beta\mathrm{S}\gamma - \mathrm{S}\alpha\mathrm{C}\gamma & \mathrm{C}\alpha\mathrm{S}\beta\mathrm{S}\gamma - \mathrm{S}\alpha\mathrm{C}\gamma \\ \mathrm{S}\alpha\mathrm{C}\beta & \mathrm{S}\alpha\mathrm{S}\beta\mathrm{S}\gamma - \mathrm{C}\alpha\mathrm{C}\gamma & \mathrm{S}\alpha\mathrm{S}\beta\mathrm{C}\gamma - \mathrm{C}\alpha\mathrm{S}\gamma \\ -\mathrm{S}\beta & \mathrm{C}\beta\mathrm{S}\gamma & \mathrm{C}\beta\mathrm{C}\gamma \end{pmatrix} \tag{3.18}$$

Hierbei steht C für $cos()$ sowie S für $sin()$ mit den Winkeln als Argumente der Funktionen. Eine Koordinatentransformation kann durch die Verkettung von Translation und Rotation durchgeführt werden. Dazu ist die Verwendung von homogenen Matrizen und Vektoren nötig. Einfacher können Rotation und Translation nacheinander durchgeführt werden um dasselbe Ergebnis zu erhalten.

$$p' = Rp - Rt \tag{3.19}$$

Es muss lediglich darauf geachtet werden, dass der Translationsvektor identisch rotiert wird.

3.5.3 Bildnormalisierung durch projektive Transformation

Häufig stehen bei der Bildverarbeitung Bilddaten mit fester oder sich ändernder perspektivischer Verzerrung zur Verfügung. Um der Realität entsprechende und vor allem gleichbleibende Ergebnisse bei der Berechnung von Attributen erzielen zu können, ist es von Vorteil die perspektivische Verzerrung zu entfernen und die Bilddaten in eine normierte Ansicht zu überführen. Um dies zu erreichen, können ebene Flächen in eine Frontalansicht gedreht oder die Oberfläche eines Zylinders in eine Ebene aufgerollt werden. Aufgrund der unterschiedlichen Oberflächen ist es nicht möglich eine optimale Abbildung zu definieren, die für jede Geometrie verzerrungsarm ist. Die im Einsatzbereich auftretenden haushaltsüblichen Oberflächen weisen in der Regel wenig Verzerrung auf weshalb die Kamera in einem definierten Abstand, senkrecht zur Tangentialebene positioniert wird. Die Tangentialebene kann wiederum aus allen 3D-Punkten des Segments berechnet werden. Damit ist die normalisierte Ansicht durch die Homographie H zwischen Abbildungsebene des Farbbildes über die Tangentialebene in die Abbildungsebene der virtuellen Kamera in normalisierter Perspektive definiert (Abbildung 3.8).

Homographie

Unter Verwendung von homogenen Koordinaten ist es möglich den betrachteten Punkt $\tilde{Q}$ auf den Punkt $\tilde{q}$ in der Kameraebene zu projizieren (Abbildung 3.8). Definiert man

$$\tilde{Q} = [X \quad Y \quad Z \quad 1]^T \tag{3.20}$$

$$\tilde{q} = [x \quad y \quad 1]^T \tag{3.21}$$

ist es mögliche die Homographie durch

$$\tilde{q} = sH\tilde{Q} \tag{3.22}$$

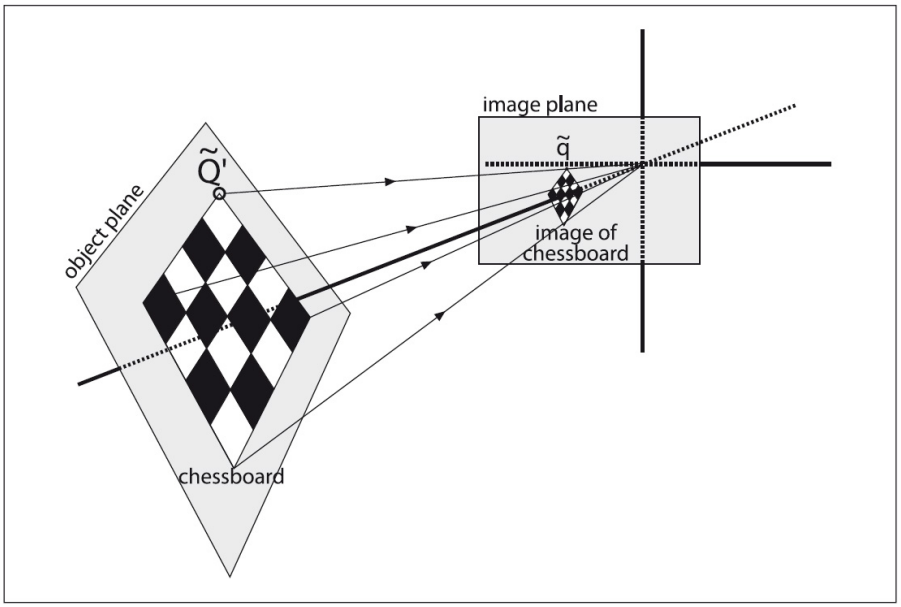


Abbildung 3.8: Ansicht eines ebenen Objekts das durch eine Homographie von der Abbildungsebene des Farbbildes auf die Abbildungsebene der virtuellen Kamera abgebildet wird [Bradski und Kaehler, 2008].

zu beschreiben mit s als Skalierungsfaktor. Die Homographiematrix H erfüllt zwei Aufgaben. Sie beschreibt zum einen die Translation und Rotation hinsichtlich der Kameraebene, sowie die Projektion auf diese. Unter Verwendung von homogenen Koordinaten kann die Transformationsmatrix zu

$$W = [R \quad t] \tag{3.23}$$

gewählt werden, was eine 3×4 Matrix darstellt, die in den ersten drei Zeilen die Einträge von R enthält. In der letzten Zeile befindet sich der Translationsvektor t. Anhand der Kameramatrix M, welche die Abbildung der Aufnahme der Kamera auf das Kamerabild beschreibt, ist es möglich die Transformation zu bestimmen:

$$\tilde{q} = sMW\tilde{Q} \tag{3.24}$$

mit

$$M = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.25}$$

Die Kameramatrix muss bereits vorhanden sein oder durch Kalibrierung der Kamera erstellt werden. Zur Ermittlung der transformierten Ebene in der Kameraebene, ist anstelle der Ermittlung eines allgemeingültigen $\tilde{Q}$, nur die Kenntnis über den Punkt $\tilde{Q}'$ auf der Objektebene nötig. Dies ermöglicht es in der Objektebene $Z = 0$ zu wählen, was die Rotationsmatrix um eine Zeile verringert, ohne Einfluss auf die Transformation zu nehmen.

$$\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = sM \begin{bmatrix} r_1 & r_2 & r_3 & t \end{bmatrix} \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} = sM \begin{bmatrix} r_1 & r_2 & t \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \tag{3.26}$$

Man erhält aus der vereinfachten Homographiematrix $H = sM \begin{bmatrix} r_1 & r_2 & t \end{bmatrix}$ die Transformation eines Punktes aus der Objektebene in die Kameraebene als

$$\tilde{q} = sH\tilde{Q}'. \tag{3.27}$$

Implementierung

Für die Ermittlung der normalisierten Bildansicht wird eine Punktewolke benötigt, die alle Punkte des zu transformierenden Segmentes enthält. Diese Punkte werden, entsprechend der Segmente der Tiefensegmentierung, der Punktewolke des Tiefensensors entnommen und in einer neuen Punktewolke zusammengefügt. Die PCA Implementierung der Point Cloud Library (PCL) ermittelt die Hauptkomponenten in dieser Punktewolke. Das Koordinatensystem P wird eingeführt, dessen Z_P Achse durch die Normalen des Segments angegeben ist. Die Normale $n_c = (a, b, c)^T$ entspricht der dritten Hauptkomponente und ermöglicht die Ermittlung der Koordinatengleichung der Ebene $ax_c + by_c + cz_c = d$, aus der wiederum die Punkte $p_{1,c}$ und $p_{2,c}$ bestimmt werden. Die Koordinaten mit dem Index C beziehen sich auf das Kamerakoordinatensystem. Die Koordinatenachsen von P werden anhand von $d_{1,c} = p_{2,c} - p_{1,c}$ und $d_{2,c} = n_c \times d_{1,c}$ schließlich durch

$$(x_p)_c = \frac{d_{1,c}}{||d_{1,c}||} \tag{3.28}$$

$$(y_p)_c = \frac{d_{2,c}}{||d_{2,c}||} \tag{3.29}$$

$$(z_p)_c = \frac{n_c}{||n_c||} \tag{3.30}$$

bestimmt. Der Punkt p_1, c wird zum Ursprung des Systems P gewählt. Dies ermöglicht die Transformation zwischen den Koordinatensystemen anhand von

$$p_c = R \cdot p_P + t \tag{3.31}$$

$$p_P = R^T p_c - R^T \cdot t. \tag{3.32}$$

Die Rotations und Translationsmatrizen sind dabei

$$R = \begin{bmatrix} (x_P)c,_x & (y_P)c,_x & (z_P)c,_x \\ (x_P)c,_y & (y_P)c,_y & (z_P)c,_y \\ (x_P)c,_z & (y_P)c,_z & (z_P)c,_z \end{bmatrix} \tag{3.33}$$

$$t = p_1, c. \tag{3.34}$$

Alle Punkte des Segments im System P werden anhand der Gleichung 3.32 in die Kameraebene transformiert und als Pixelpaar in einer Liste abgelegt. Dies wird von der OpenCV Funktion findHomography() benötigt, die aus beiden Listen die Homographiematrix, per Regression über mindestens vier Punktepaare, ermittelt. Eine bestimmte Entfernung des transformierten Segments zur Kameraebene sowie eine zentrierte Darstellung kann durch Hinzufügen eines Offsets zu den Zielpunkten erreicht werden. Der Offset wird durch

$$m_P = \begin{bmatrix} (\min p_{P,x} + \max p_{P,x})/2 - c/(2 * s) \\ (\min p_{P,y} + \max p_{P,y})/2 - r/(2 * s) \\ 0 \end{bmatrix} \tag{3.35}$$

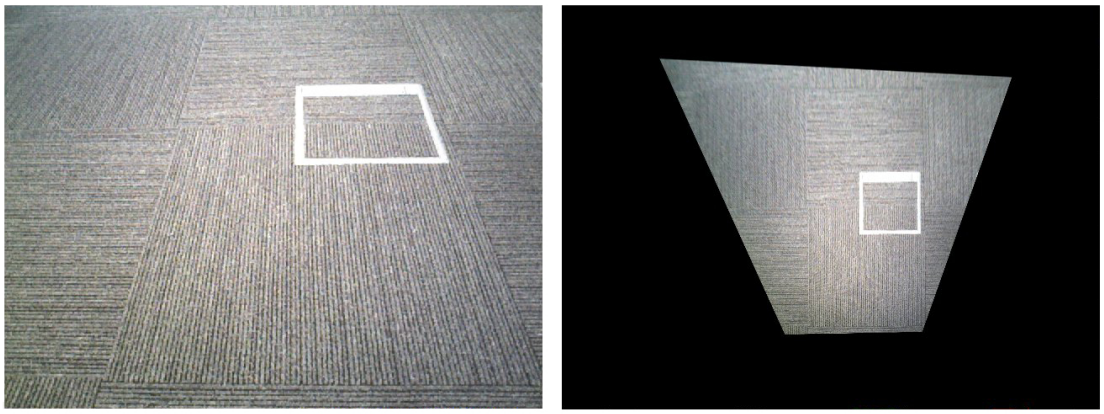


Abbildung 3.9: Links: Unbearbeitete Aufnahme. Rechts: Normalisierte Ansicht.

bestimmt wobei min und max jeweils die minimalen und maximalen x und y Koordinaten der transformierten Punkte P darstellt. c und r ist die Anzahl der Spalten und Zeilen des Bildes. Die Variable s stellt die Auflösung, mit der das transformierte Segment dargestellt werden soll, dar. Die angepassten Punkte werden durch

$$p_{P,n} = res(p_P - m_P) \tag{3.36}$$

berechnet und eine Untermenge von Punkten in gleichmäßigem Abstand ausgewählt, die zur Ermittlung der Homographie H genutzt werden. Die normalisierte Ansicht wird erzeugt, indem jeder Punkt im Segment durch $P_B = H \cdot P_C$ in die Kamerakoordinaten transformiert wird. Diese Aufgabe wird von der Funktion warpPerspective() übernommen, die perspektivische Transformationen anhand der übergebenen Transformationsmatrix vornimmt. Die unbearbeitete und eine normalisierte Ansicht eines Teppichmusters wird in Abbildung 3.9 dargestellt.

Das Kamerasystem des Care-O-bot® ist am oberen Ende des beweglichen Rumpfes angebracht und drehbar gelagert. Dies ermöglicht dem System, einen größeren Bereich mit der Kamera zu erfassen, und führt zu vielen unterschiedlichen Kameraperspektiven, wodurch eine Normalisierung der Bilddaten sinnvoll wird. Anhand der Tiefendaten ist es möglich, ein Koordinatensystem in die verzerrten Pixelwolken zu legen. Durch eine Koordinatentransformation in das Kamerakoordinatensystem erhält man die normalisierten Bildpunkte, die in Verbindung mit den verzerrten Bildpunkten die

Ermittlung der Transformationsmatrix möglich machen.

3.6 Textureigenschaften

Farbe ist neben Textur eine wichtige Eigenschaft für die menschliche Wahrnehmung sowie für die Bildverarbeitung [Palm, 2004]. Die Kombination der Farb- und Texturinformationen kann durch parallele, sequentielle und integrative Methoden durchgeführt werden. Dabei werden bei der Parallelen Methode statistische Informationen (Kapitel 2.2.1) der Farbe über dem gesamten Bild und lokale Textureigenschaften (Kapitel 2.2.2, 2.2.3) anhand der Grauwerte gesondert betrachtet. Die sequentiellen Methoden führen eine Segmentierung anhand der Farbe durch, um anschließend auf den erhaltenen Segmenten die Textureigenschaften über die Grauwerte zu ermitteln. Durch die Ermittlung von Textureigenschaften auf den einzelnen Kanälen eines Bildes versuchen die integrativen Methoden die Abhängigkeit von Farbe und Textur zur Beschreibung zu nutzen. Unterschieden wird die Betrachtung der einzelnen Känale (single-channel Strategie) und der Betrachtung von zwei oder mehreren Kanälen (multi-channel Strategie). Die hier verwendeten Textureigenschaften wurden bereits in [Esslinger, 2014] definiert und werden im Folgenden kurz beschrieben.

3.6.1 Farbattribute

Die Analyse von Farb- und Textureigenschaften wird in dieser Arbeit parallel durchgeführt. Daher werden Statistiken der 1. Ordnung als Farbattribute verwendet. Weitere Attribute sind die erste und zweite dominante Farbe sowie die Farbigkeit, welche die Menge unterschiedlicher Farben beschreibt. Bei der Wahl des Farbraumes muss darauf geachtet werden, dass Farben ähnlich dem menschlichen Empfinden beschrieben werden. Dies ermöglicht es Menschen die Farbattribute zu interpretieren. Der HSV-Farbraum erfüllt diese Anforderung.

HSV Farbraum

Der HSV Farbraum besteht aus den Kanälen Farbwert (Hue), Sättigung (Saturation) und Hellwert (Value) und kann als Kegel dargestellt werden (Abbildung 3.10 (a)). Der Farbort im Kegel wird durch den Farbwinkel des Farbwertes (H) auf dem Farbkreis, die Sättigung (S) als orthogonalen Abstand zu der vertikalen Achse in Prozent und der Dunkelstufe als Prozentsatz der vertikalen Achse bestimmt. Die Chrominanz wird durch den Farbwinkel bestimmt, welcher die Wellenlänge der dominanten Farbe angibt sowie deren Weißanteil der durch die Sättigung (S) beziffert wird. Der Hellwert (V) ist die Intensität der Farbe. Durch diese Aufteilung entspricht der HSV Farbraum der menschlichen Farbwahrnehmung besser als der RGB Farbraum [Arvis et al., 2011] und ist dadurch besonders geeignet zur Beschreibung von Menschen interpretierbaren Farbattributen.

$$
\begin{aligned}
H &= \begin{cases} 0 & \text{if } S = 0 \Leftrightarrow R = G = B \\ 60° \cdot \left(0 + \frac{(G-B)}{SV}\right) & \text{if } V = R \\ 60° \cdot \left(2 + \frac{(B-R)}{SV}\right) & \text{if } V = G \\ 60° \cdot \left(4 + \frac{(R-G)}{SV}\right) & \text{if } V = B \\ H + 360° & \text{if } H < 0 \end{cases} \\
S &= \begin{cases} 0 & \text{if } V = 0 \Leftrightarrow R = G = B = 0 \\ \frac{V - \min(R,G,B)}{V} & \text{if } V > 0 \end{cases} \\
V &= \max(R, G, B)
\end{aligned}
\tag{3.37}
$$

Statistische Farbattribute

Wie in Kapitel 2.2.1 beschrieben, ist es möglich mit Hilfe des Mittelwerts und der Varianz der Sättigung sowie des Hellwertes auf Eigenschaften von Texturen zu schließen.

(a) HSV-Farbkegel (b) HSV-Farbtonskala

Abbildung 3.10: Darstellung des HSV-Farbraumes [WIKIPEDIA, 2014].

Diese werden über alle Pixel einer Textur im HSV-Farbraum berechnet. Entsprechende Funktionen sind in der OpenCV Bibliothek vorhanden. In der Implementierung wurde die Funktion meanStdDev() genutzt. Diese berechnet den Mittelwert und die Standardabweichung über alle Kanäle eines Arrays welche in diesem Fall durch die Bildmatrix im HSV-Farbraum dargestellt ist. Es ist allerdings möglich, dass unterschiedliche Texturen ähnliche Mittelwerte und Varianzen über die S- und V-Kanäle besitzen. Für eine genaue Identifizierung sind somit weitere Attribute nötig.

Farbattribute anhand des Farbwerts

Aussagekräftige Informationen von Texturen können über die Farbe gewonnen werden. Diese ist im HSV-Farbraum im H-Kanal abgebildet. Drei Attribute, die zur Beschreibung von Texturen genutzt werden, sind erste und zweite dominante Farbe sowie die Farbigkeit, die die Menge unterschiedlicher Farben in der Textur wiedergibt. Eine Eigenschaft des HSV-Farbraumes ist, dass bei einer geringen Sättigung die Farbwerte instabil werden und durch die Beleuchtung in dem aufgenommenen Bild verändert werden können. Daher ist es nötig, dass für jedes Pixel die Sättigung über einem Schwellwert liegt. Ist dies nicht der Fall, wird der Farbwert des Pixels nicht betrachtet. Die restlichen Farbwerte werden in einem normalisierten Histogramm aufgetragen. Dazu wurde die Farbtonskala Abbildung 3.10 (b) des HSV-Raumes in zehn Unterbereiche

aufgeteilt, die jeweils den Wertebereich einer Klasse im Histogramm darstellt. Um die Attributwerte bestimmen zu können, werden relevante Klassen im Histogramm ermittelt. Diese müssen, um aussagekräftig zu sein, eine minimale Höhe übersteigen. Aus der Menge der relevanten Klassen wird das Attribut Farbigkeit bestimmt. Die Attribute dominante und zweite dominante Farbe werden anhand der Position der häufigsten und zweit häufigsten Klasse bestimmt. Eine dominante Farbe kann allerdings nur vorhanden sein, wenn mindestens eine Klasse als relevant eingestuft wurde wie auch eine zweite dominante Farbe erst bei mehreren relevanten Klassen auftreten kann.

3.6.2 Texturattribute

Ein wichtiger Aspekt einiger Texturattribute ist die Analyse der Primitives. Primitives sind einheitliche Mikrotexturen, die sich in einer festgelegten Weise wiederholen. Die Schwierigkeit bei der Detektion der Primitives ist dabei, zu unterscheiden, ob es sich um Textur bestehend aus großen Primitives, die feine Strukturen enthalten, handelt oder ob die Texturen aus kleinen Primitives bestehen. Die Primitives werden anhand ihrer Konturen detektiert. Somit ist es sinnvoll, über Verringerung des Kontrastes ("blurring") feine irrelevante Strukturen zu entfernen, die sonst als Konturen erkannt werden.
Konturen bestehen aus zusammenhängenden Kanten im Bild. Um diese zu detektieren, wurde die OpenCV Implementierung des Canny Edge Detectors verwendet. Der Algorithmus beginnt mit dem Entfernen von Rauschen mit Hilfe eines Gaußfilters K mit einer Kernelgröße von 5.

$$K = \frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} \tag{3.38}$$

Im nächsten Schritt werden anhand des Sobeloperators die partiellen Ableitungen er-

mittelt. Das Bild wird in x und y Richtung mithilfe der Faltungsmatrizen G_x und G_y gefaltet.

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \tag{3.39}$$

Anhand der Faltungen ist es möglich, die Polarkoordinaten in Form des Betrags des Gradienten G sowie dessen Richtung θ zu bestimmen.

$$G = \sqrt{G_x^2 + G_y^2} \qquad \theta = \arctan\left(\frac{G_y}{G_x}\right) \tag{3.40}$$

Die ermittelten Richtungen werden auf 0°, 45°, 90° und 135° gerundet. Die ermittelten Werte werden von der Non-maximum suppression Methode genutzt, um sicher zu stellen, dass Kanten eine Stärke von einem Pixel besitzen. Dazu werden für jedes Pixel die Beträge der benachbarten Pixel in Richtung des Gradienten verglichen. Ist der Betrag eines Nachbarn größer als der des betrachteten Pixels, gehört dieses nicht zu einer möglichen Kante. Im finalen Schritt werden die bisher ermittelten Kanten auf ihre Stärke überprüft. Ein Hystereseverfahren vergleicht dazu für jeden Pixel einer Kante die Beträge der Gradienten mit einem oberen und unteren Schwellwert (engl. Threshold). Übersteigt der Betrag den oberen Schwellwert, gehört das Pixel zu einer Kante. Wird der untere Schwellwert unterschritten wird das Pixel verworfen. Liegt der Betrag zwischen oberer und unterer Schwelle, gehört das Pixel zu einer Kante wenn dieser mit einem Pixel benachbart ist, das den oberen Schwellwert überschreitet. Nach Beendigung des Hystereseverfahrens ist die Kantendetektion abgeschlossen. Die ermittelten Kanten werden als 1 Pixel in einem Binärbild dargestellt. Dieses bildet die Datengrundlage zur Ermittlung des im Folgenden beschriebenen Attributs Kontrast sowie weiteren Texturattributen, die global das Binärbild untersuchen. Primitives werden lokal untersucht. Dazu ist es nötig, zusammenhängende Pixel im Binärbild zu erkennen, die eine Kontur darstellen. In der Implementierung wird die OpenCV Funktion findContours() verwendet, die einen Kantenverfolgungsalgorithmus nach [Suzuki et al., 1985] für die Detektion der Konturen verwendet. Als Ergebnis liefert die Funkti-

on ein Array, in dem die Konturen, dargestellt durch Angabe aller zugehörigen Pixel, nach der Größe absteigend sortiert sind.

Primitive size

Zur Ermittlung der Größe der Primitives wird die Anzahl der Pixel der größten Konturen betrachtet. Dabei wird von den drei größten Konturen, sowie von dem ersten Viertel des Arrays, das die ermittelten Konturen enthält, der arithmetische Mittelwert gebildet. Der Mittelwert dieser beiden Größen wird als Primitive size definiert.

Amount of primitives

Für die Ermittlung der Anzahl (Amount) der Primitives werden alle Konturen, die größer als 1/8 der größten Kontur sind, verwendet. Es wird der Abstand von jedem Bildpixel zu der nächstliegenden Kontur berechnet und über alle Distanzen der arithmetische Mittelwert gebildet. Fällt der Wert gering aus, sind viele Primitives im Bild vorhanden. Dieser Wert und die Anzahl der verwendeten Konturen werden auf ein Intervall skaliert und der größere Wert als Attributwert verwendet. Dadurch sollen fehlerhafte Berechnungen von Bildern mit hohem Rauschen vermieden werden.

Strength of primitives

Die “strength“ ist ein Maß, wie gut ein Primitive zu beschreiben ist und wie leicht es in einem Bild erkannt werden kann. Um dies zu untersuchen werden die größten Konturen betrachtet, die sich im vorderen Viertel des im vorherigen ermittelten Arrays befinden. Auf jedem Pixel, das zu einer der Konturen gehört, wird die direkte Nachbarschaft untersucht. Dazu wird im V-Kanal des Bildes im HSV-Farbraum an der Position der Pixel der Kontur die Standardabweichung zu den Nachbarn, die sich in einer 3×3 Umgebung befinden, ermittelt. Die Ergebnisse werden summiert und durch die Menge an Pixel dividiert. Wird der Quotient noch mit der mittleren Größe

der Konturen multipliziert, erhält man nach [AMADASUN und KING, 1989] ein Maß für die Genauigkeit der primitives.

Regularity of primitives

Für die Gleichmäßigkeit (regularity) werden die gleichen Konturen betrachtet, wie bei der Ermittlung der Anzahl der primitives. Zwei Kriterien werden dabei untersucht. Zum einen wird die Anzahl der Pixel aller Konturen ermittelt. Zum anderen wird die Ausdehnung (engl. Extent) der Konturen berechnet. Dazu wird ein möglichst kleines Rechteck um jede Kontur gelegt und das Verhältnis zwischen Pixeln, die zu der Kontur gehören, und den restlichen Pixeln ermittelt. Über die Werte der Größe und der Ausdehnung wird die Standardabweichung und der Mittelwert gebildet. Das Attribut Gleichmäßigkeit wird durch die Summen der Quotienten von Standardabweichung zum Mittelwert für die Größe und der Ausdehnung der Konturen bestimmt.

Contrast

Nach dem Ansatz von [AMADASUN und KING, 1989] ist ein hoher Kontrast (Contrast) vorhanden, wenn benachbarte Regionen hohe Intensitätsunterschiede aufweisen und unterschiedliche Intensitätsniveaus sichtbar sind. Nachgewiesen werden kann dies mit Hilfe einer neighborhood gray tone differenze matrix (NGTDM). Um diese ermitteln zu können, wird der Mittlere Grauwert $\bar{A}$ der Nachbarn jedes Pixels an der Position (x, y) aus dem V-Kanal des betrachteten Bildes durch

$$\bar{A}(x,y) = \frac{1}{W-1}\left(\sum_{k=-d}^{d}\sum_{l=-d}^{d} v(x+k, y+l)\right) \tag{3.41}$$

bestimmt, wobei $(k,l) \neq (0,0)$ gelten muss. Der Radius der Nachbarschaft wird durch d angegeben, und somit die Anzahl der Pixel die in der Region liegen anhand von

$W = (2d+1)^2$. Ist i der Grauwert an der Position (x, y) kann die NGTDM durch

$$z(i) = \begin{cases} \sum\limits_{(x,y)\in N_i} |i - \bar{A}(x,y)| & \text{wenn} N_i \neq \emptyset \\ 0 & \text{sonst} \end{cases} \tag{3.42}$$

mit $\{N_i\}$ als Menge aller Pixel mit dem Grauwert i, ermittelt werden. Umgesetzt wurde die Erzeugung der NGTDM durch die Faltung des Eingangsbildes mit dem Kernel

$$K = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{3.43}$$

für $d = 1$. Die Anzahl der Spalten und Zeilen des Kernels ergeben sich aus $2d + 1$. Während der Anker an der Position $(d, d) = 0$ ist werden die restlichen Stellen mit eins gefüllt. Die Werte des gefalteten Bildes an der Position (x, y) entsprechen $\bar{A}(x, y)$. Die oben beschrieben NGTDM kann somit durch eine pixelweise Subtraktion des gefalteten Bildes vom Originalbild erzeugt werden, während die Werte des Originalbildes als Index der NGTDM dienen.

Eine breites Spektrum an unterschiedlichen Grauwerten sowie eine hohe Änderungsrate der Grauwerte führt zu einem hohen Kontrast [Amadasun und King, 1989]. Die Berechnung des Kontrastes $cont_a$ kann durch

$$cont_a = \left(\frac{1}{N_g(N_g-1)} \sum_{i=0}^{G_h} \sum_{j=0}^{G_h} p_i p_j (i-j)^2 \right) \left(\frac{1}{mn} \sum_{i=0}^{G_h} z(i) \right) \tag{3.44}$$

vorgenommen werden [Amadasun und King, 1989]. Die Anzahl der unterschiedlichen, im Bild aufgetretenen Grauwerte wird durch N_g angegeben, während G_h der höchste Grauwert ist. In einem Bild der Maße $n \times m$ ist $p_i = \frac{N_i}{nm}$ die Wahrscheinlichkeit des Auftretens des Grauwertes i. Der erste Faktor bewertet das vorhandene Spektrum, indem die quadrierte Differenz aller Grauwertpaare durch die Wahrscheinlichkeit ihres Auftretens gewichtet summiert werden. Grauwertpaare die ein geringes Spektrum abdecken und selten auftretende Grauwerte haben durch ihre geringen Werte wenig Einfluss auf die Summe. Der zweite Faktor bezieht aus der NGTDM die durchschnittliche Differenz der Grauwerte zwischen Pixeln und deren Nachbarschaft.

Line-likeness

Zur Bestimmung der Linienartigkeit (Line-likeness) einer Textur werden zwei Kriterien auf der selben Auswahl von Konturen wie bei "Anzahl der primitives" überprüft. Beginnend mit der Circle Hough Transformation [Ballard, 1981] sollen alle Kreise im Bild detektiert werden. Die OpenCV Funktion HoughCircles() bietet die Möglichkeit, Kreise mit dem Hough Gradientenverfahren nach [Yuen et al., 1990] zu bestimmen, wie in [Bradski und Kaehler, 2008] detailliert beschrieben. Anhand der Parameter der Funktion ist es möglich, die Größe und Menge der erkannten Kreise einzustellen. Die Größe wird direkt über die Parameter maximaler und minimaler Kreisradius bestimmt. Eine Überlappung der Kreise kann über den minimalen Abstand der Kreismittelpunkte gesteuert werden. Ein zu klein gewählter Wert kann zur mehrfachen Detektion eines Kreises führen, während bei zu großen Werten nahe aneinander liegende Kreise nicht erkannt werden. Wie genau eine Kontur einem Kreis entsprechen muss, um detektiert zu werden, kann über die Gradientenverfahren spezifischen Parameter bestimmt werden. Für eine genaue Detektion aller Kreise wird die Funktion für vier unterschiedliche Größen mit angepassten Parametern durchgeführt. Viele erkannte Kreise lassen auf eine schlechte Linienartigkeit der Textur schließen. Das zweite Kriterium bewertet ob die Primitives einer Textur kreisförmig angeordnet sind. Dazu wird eine Ellipse um die Kontur gelegt und die Exzentrizität e derer ermittelt. Für eine Ellipse der Form $\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$ ist $e = \sqrt{a^2 - b^2}$. Das Verhältnis der Anzahl der Konturen, deren Exzentrizität einen Schwellwert übersteigt, zur Anzahl aller Konturen ist das Maß des zweiten Kriteriums. Die Funktion fitEllipse() der OpenCV Bibliothek bietet die geforderte Funktionalität und wurde bei der Implementierung verwendet. Unter der Annahme, dass von Menschen geschaffene Umgebungen, wie der häusliche Bereich, linienartig sind, wird der kleinere Wert der beiden Kriterien als Wert des Attributs übernommen.

Lined

Lined (liniert) beschreibt im Gegensatz zur Line-likeness nicht die Form der Primitives sondern deren Anordnung. Anhand der Gradienten in x und y Richtung wird die Ausrichtung der Pixel bestimmt. Dies wird realisiert durch die Anwendung des Sobeloperators wie zu Beginn des Kapitel 3.6.2 beschrieben. Die aus dem V-Kanal des Bildes im HSV-Farbraum erhaltenen Polarkoordinaten werden genutzt, um ein Histogramm $h(i)$ mit i Klassen zu erstellen. Ist der Gradient eines Pixels kleiner als 15% des maximalen Gradienten im Bild, werden die zugehörigen Polarkoordinaten nicht bei der Erstellung des Histogramms mit einbezogen. Dieses besteht aus 16 Klassen die sich über π verteilen. Die Winkel der gültigen Pixel werden durch π geteilt und anhand des Restbetrages im Histogramm eingeordnet. Nach [TAMURA et al., 1978] ist das Attribut Liniert stark vorhanden, wenn der höchste Peak an der Position ϕ_p eindeutig von seiner Nachbarschaft separiert ist und kann durch

$$l_{lined} = z_p \cdot \sum_{\phi \in w_p} |\phi - \phi_p| \tag{3.45}$$

dargestellt werden. Das Intervall zwischen den Minima um den höchsten Peak ist w_p während der Faktor

$$z_p = (\delta - \min(h(\phi_p), \epsilon)) \tag{3.46}$$

die absolute Höhe des größten Peaks zur Gewichtung des Attributwertes einbezieht. Die Parameter δ und ϵ können dabei den Einfluss von $h(\phi_p)$ steuern.

Checked

Eine Textur, die kariert (engl. checked) ist, verfügt über Primitives, die nach Attribut "lined" in zwei zueinander orthogonalen Orientierungen angeordnet sind. Über das Histogramm, das für das Attribut "lined" ermittelt wurde, kann die zweite Orientierung der Primitives bestimmt werden. Die Textur ist kariert, wenn ein zweiter Peak an der Position ϕ_q der außerhalb des Intervalls w_p liegt und einen Abstand von $\frac{\pi}{2}$ zum Peak an

der Position ϕ_p einhält, existiert. Bewertet wird die zweite Orientierung entsprechend dem vorherigen Attribut anhand der Gleichungen 3.45 und 3.46. Dies ist allerdings nur gültig, wenn der Attributwert von "lined" einen Schwellwert überschreitet und neben dem Peak q keine weiteren Peaks existieren.

Directionality

Die Richtungsabhängigkeit (engl. Direktionality) führt drei Kriterien ein. Für das erste Kriterium werden die Schwerpunkte der Konturen bestimmt und ein Kreis so eingepasst, dass die Abstände zwischen Kreis und Schwerpunkten minimal werden. Dies ermöglicht es, eine kreisförmige Anordnung der Primitives zu erkennen. Die Parameter a_i des Kreises der Form $x^2 + y^2 + a_1 \cdot x + a_2 \cdot y + a_3 = 0$ werden mit der solve() Funktion der OpvenCV Bibliothek anhand der Singulärwertzerlegung optimiert. Ist eine geringe Anzahl von Konturen vorhanden, wird der Wert des Kriteriums durch die Summe aller Abstände der Schwerpunkte zum ermittelten Kreis bestimmt. Ist die Anzahl der Konturen zu hoch, kann eine kreisförmige Anordnung nicht zuverlässig bestimmt werden. Das zweite Kriterium untersucht die räumliche Verteilung der Konturen in x und y Richtung. Die auftretenden x und y Koordinaten werden jeweils in einem Histogramm aufgetragen und die Standardabweichung derer bestimmt. Aus der Summe der Standardabweichungen wird der Wert des Kriteriums bestimmt. Das letzte Kriterium bildet die Orientierungen der Konturen in einem Histogramm ab. Die Orientierung wird aus dem Winkel zwischen der x-Achse und der Hauptachse der Ellipse gewonnen, die ebenfalls für das Attribut "Line-likness" bestimmt wird. In das Histogramm werden nur die Winkel aufgenommen, deren Kontur eine Exzentrizität aufweisen, die größer als ein Schwellwert ist. Der Wert des Kriteriums wird wiederum durch die Standardabweichung der auftretenden Winkel bestimmt. Das Attribut "Directionality" soll feststellen, ob die Primitives eine geordnete Orientierung haben. Welche Art der Orientierung ist dabei nicht von Bedeutung. Daher wird für den Attributwert der maximale Wert aus den drei vorherigen Kriterien sowie aus "lined" und "checked" gewählt.

3.6.3 3D Attribute

Die Tiefendaten ermöglichen es, Merkmale auf 3D-Segmenten zu gewinnen. Geometrische Eigenschaften der Oberflächen der Segmente wie z.B. kugelförmig, kegelförmig, konkav oder konvex als ein weiteres Attribut die Klassifizierung unterstützen. Dies wird in dieser Arbeit allerdings nicht weiter betrachtet.

3.7 Klassifizierung

Ziel der Klassifizierung ist es, Texturen, die aus der Segmentierung (Kapitel 3.4) gewonnen werden, in bekannte Klassen einzuordnen. Dabei stehen die Attribute in Tabelle 3.2 zur Verfügung, die bereits in Kapitel 3.6 beschrieben wurden. Durch die Datenbank A.3 sind die Klassen definiert und entsprechende Beispieldaten stehen zur Verfügung. Somit ist es möglich, die Attribute eines Beispielbildes zu berechnen und mit der entsprechenden Klasse zu verknüpfen. Die Datensätze bilden die Datenbasis für ein überwacht lernendes System.

Tabelle 3.2: 17 von Menschen interpretierbare Texturattribute.

1	colorfulness	7	variety of value	13	line-likeness
2	dominant color	8	average primitive size	14	3D roughness
3	secondary dominant color	9	number of primitives	15	directionality
4	saturation	10	strength of primitives	16	lined
5	variety of saturation	11	regularity of primitives	17	checked
6	value	12	contrast		

Entscheidend für die Qualität der Klassifizierung ist die Aussagekraft der Attribute. Weichen die Werte der Attribute bei gleichen Beispieldaten voneinander ab, oder erzeugen unterschiedliche Beispieldaten ähnliche Attributwerte, ist eine korrekte Klassifizierung kaum möglich. Zudem ist es wichtig, dass die Attribute nur Werte innerhalb eines Intervalls annehmen können. Treten bei einem Attribut größere numerische Wer-

te auf kann dieses dominant gegenüber Attributen mit kleinen Werten werden [Hsu et al., 2003]. Die hier verwendeten Attribute werden bereits bei deren Berechnung auf ein Intervall skaliert. Des Weiteren hat die Wahl der Parameter bei dem Lernprozess eines Systems großen Einfluss auf das Ergebnis. Welche Parameter dabei vorhanden sind,t abhängig von dem System. Zwei Systeme des überwachten Lernens sind die SVM und das Neuronale Netz deren mögliche Konfiguration zur Erfüllung der Anforderung im Folgenden beschrieben werden.

3.7.1 Support Vector Machine

Die SVM kann sowohl für Klassifizierung als auch für Regression und Verteilungsschätzung eingesetzt werden. Die verwendete Implementierung der OpenCV Bibliothek baut auf [Chang und Lin, 2011] auf. Für die Klassifizierung existieren mehrere Implementierungen weshalb eine entsprechende Formulierung der SVM gewählt werden muss. Zwei gängige Methoden sind die C-Support Vector Classification ([Boser et al., 1992], [Cortes und Vapnik, 1995]) und die υ-Support Vector Classification [Schölkopf et al., 2000]. Beide ermöglichen die Klassifizerung von zwei oder mehreren Klassen und erlauben eine unvollkommene Separierung. Der Unterschied der beiden Methoden liegt in den unterschiedlichen Parametern C und υ. Bei der C-Support Vector Classification ist der Parameter C ein Strafmaß (engl. Penalty Factor), das bei der Optimierung der Hyperebene auf nicht separierbare Punkte angewendet wird. Dadurch wird es möglich, die Komplexität der Hyperebene zu steuern. Ein hoher Wert für C führt zu einer starken Bestrafung der Ausreißer, was durch mehr Stützvektoren ausgeglichen wird, und somit zu einer Überanpassung (engl. Overfitting) führt. Hingegen werden bei zu kleinen Werten von C zu viele Ausreißer zugelassen, was zu Ungenauigkeiten (engl. Underfitting) führt. Die υ-Support Vector Classification dagegen nimmt direkten Einfluss auf die Anzahl der Stützvektoren anhand des Parameters υ, der den Anteil dieser angibt. Wie bei Parameter C führt ein zu klein oder groß gewähltes υ zu overfitting oder underfitting [Chang und Lin, 2011].

Nicht linear separierbare Probleme werden von der SVM mit dem Kernel-Trick gelöst. Dabei können unterschiedliche Kernel Funktionen gewählt werden:

- Die Lineare Kernel Funktion ist die simpelste und schnellste Kernelfunktion.

$$K(x_i, x_j) = x_i^T x_j \tag{3.47}$$

- Polynomial Kernel eignen sich besonders bei normalisierten Trainingsdaten.

$$K(x_i, x_j) = (\gamma x_i^T x_j + c)^d \tag{3.48}$$

  Dabei sind γ mit $\gamma > 0$ und c einstellbare Parameter und d der Grad des Polynoms.

- Radiale Basisfunktion(RBF) Kernel.

$$K(x_i, x_j) = e^{-\gamma ||x_i - x_j||^2} \tag{3.49}$$

  Wie bei Polynomial Kernel ist γ ein einstellbarer Parameter mit $\gamma > 0$.

- Sigmoid Kernel

$$K(x_i, x_j) = \tanh(\gamma x_i^T x_j + c) \tag{3.50}$$

  Dabei sind γ mit $\gamma > 0$ und c einstellbare Parameter.

Ein weiterer wichtiger Parameter ist das Abbruchkriterium. Die SVM löst das Optimierungsproblem iterativ und nähert sich somit immer weiter an eine optimale Lösung an. Es ist möglich, den Optimierungsprozess nach einer maximalen Anzahl von Iterationsschritten oder nach Einhaltung einer Fehlertoleranz ϵ zu beenden. Dabei muss die maximale Trennfläche und der minimale Fehler ϵ unterschreiten.

Die Einstellungen der SVM der Implementierung werden so gewählt, dass die Klassifizierung auf den Trainingsdaten der Texturdatenbank A.3 möglichst gute Ergebnisse liefert. Die Bewertung der Klassifizierung findet in Kapitel 4.3 statt. Die SVM Implementierung wird zur C-Support Vector Classification gewählt mit $C = 1,01$ und einem linearen Kernel. Das Abbruchkriterium wird zu n=1000 Iterationen oder einer Genauigkeit von $\epsilon = 10^{-8}$ gewählt.

3.7.2 Künstliche Neuronale Netze

Künstliche Neuronale Netze können gegenüber SVMs vielseitiger konfiguriert werden. Durch die frei modellierbare Hiddenschicht und die Möglichkeit jedem Neuron einzelne Aktivierungsfunktionen zu zuordnen, können die KNN nach Belieben modelliert werden. Lediglich die Inputschicht muss genau so viele Neuronen besitzen wie vorhandene Eingangsparameter, die hier die Texturattribute darstellen. Außerdem muss die Outputschicht so viele Neuronen enthalten wie die Anzahl der zu detektierenden Klassen. Die Wahl der Anzahl der verdeckten Neuronen und Schichten ist entscheidend für die Leistungsfähigkeit der Klassifizierung. Viele Neuronen und Schichten ermöglichen eine Flexibilität des Netzes, können allerdings zu overfitting führen. Im Gegenzug können KNN mit zu wenig Neuronen die Komplexität der Problemstellung nicht abbilden und neigen dadurch zu underfitting. Daher folgt bei steigender Anzahl von Eingangsparameter und Trainingsdaten eine steigende Anzahl von Neuronen durch die komplexere Problemstellung. Die Wahl, welche Modellierung der Hiddenschicht optimal ist, kann nur durch Experimente zuverlässig ermittelt werden.

Die in dieser Arbeit verwendete OpenCV Bibliothek bietet eine Implementierung von KNN´s als multi-layer perceptrons (MLP). MLPs bestehen wie bereits beschrieben jeweils aus einer Input-, Hidden- und Outputschicht. Die Propagierungsfunktion der Neuronen (Abbildung 3.11) dieser Implementierung ist festgelegt zu

$$u_i = \sum_j (w_{i,j} \cdot x_j) + w_{i,bias}. \tag{3.51}$$

Es werden die Ausgaben x_j der vorherigen Schicht n und das Bias gewichtet summiert. Die Verbindungsgewichte $w_{i,j}$ beschreiben den Faktor, der auf die Ausgabe des Neurons j aus der Schicht i wirkt, das mit dem betrachteten Neuron verbunden ist. Das Bias-Neuron ist ein zusätzliches Neuron in der Input- und Hiddenschicht. Es besitzt einen konstanten Wert, der durch das Verbindungsgewicht angepasst wird. Durch die Modifizierung des Verbindungsgewichts in der Lernphase entspricht das Bias einem variablen Schwellwert. Die Ausgabefunktion ist zu einer Identitätsfunktion festgelegt.

Somit ist die Ausgabe des Neurons

$$y_i = f(u_i) \tag{3.52}$$

mit $f(u)$ als Aktivierungsfunktion.

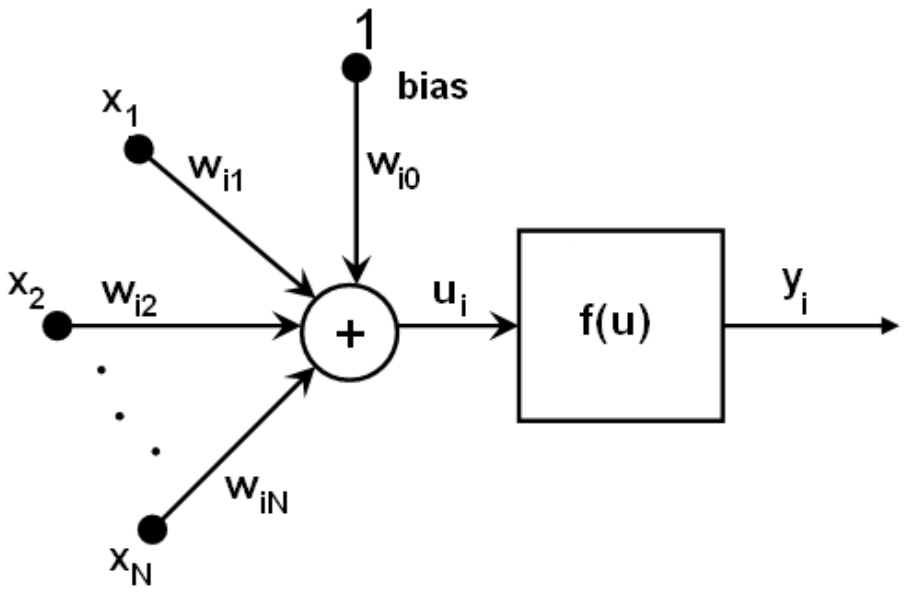


Abbildung 3.11: Modell eines Neurons der OpenCV Bibliothek [OpenCV, 2014c].

Die Wahl der Aktivierungsfunktion ist abhängig von dem zu lösenden Problem. Entsprechend den Eingangsparametern sollen die Aktivierungswerte y_i der Neuronen nur in sinnvollen Intervallen auftreten und somit ein Konvergieren des Netzes ermöglichen. Häufige Aktivierungsfunktionen sind:

- Lineare Funktionen

$$y_i = \sum_{i=1}^{n} w_{ij} x_j = net_i \tag{3.53}$$

  Hierbei sind w_{ij} die Verbindungsgewichte und x_j die Ausgaben der vorherigen Neuronen. Die Aktivierungswerte sind nicht beschränkt, weshalb häufig obere und untere Schranken verwendet werden.

- Schwellwert Funktionen

$$y_i = \begin{cases} x_1 & \text{für } net_i > \Theta_i \\ x_2 & \text{für } net_i \leq \Theta_i \end{cases} \tag{3.54}$$

  x_1 und x_2 sind zulässige Aktivierungswerte und Θ ein wählbarer Schwellwert.

- Sigmoid Funktionen

$$y_i = \frac{1}{1 + \exp(-net_i)} \tag{3.55}$$

- Gauss Funktionen

$$y_i = e^{-(1/2net_i)^2} \tag{3.56}$$

Die für die Implementierung gewählte Aktivierungsfunktion der Neuronen ist eine symmetrische Sigmoid Funktion der Form

$$f(x) = \beta \cdot (1 - e^{-\alpha x})/(1 + e^{-\alpha x}). \tag{3.57}$$

Die Parameter werden zu $\alpha = 1$ und $\beta = 1$ gewählt. Die Funktion entspricht Abbildung 3.12.

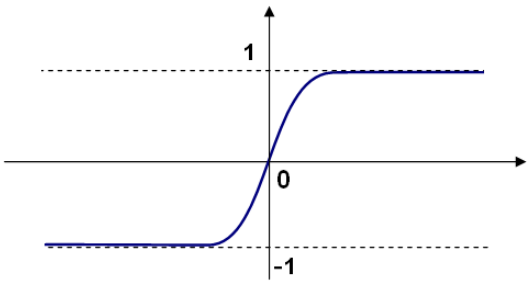


Abbildung 3.12: Sigmoidfunktion mit den Parametern $\alpha = 1$ und $\beta = 1$ **??**.

Dabei modelliert der Parameter α die Steigung der Funktion und β skaliert die Ähnlichkeit.

Wie die Verbindungsgewichte in einem KNN in der Lernphase verändert werden, ist durch Lernregeln bestimmt. Dabei unterscheiden diese sich in Lernverfahren für überwachtes und unüberwachtes Lernen, sowie in der Fähigkeit mit Hiddenschichten umzugehen.

- Die Hebb-Regel ist eine biologisch plausible Lernregel, die das Verbindungsgewicht zweier Neuronen verändert, wenn beide Neuronen aktiv sind. Die Änderung Δw_{ij} des Gewichtes wird durch den Aktivierungswert des sendenden Neurons y_j und den Aktivierungswert des empfangenden Neurons y_i sowie eines Trainingsparameters ϵ bestimmt:

  $$\Delta w_{ij} = \epsilon y_i y_j \tag{3.58}$$

  Die Vorteile dieser Lernregel sind ihre Einfachheit sowie die Anwendbarkeit bei überwachtem und unüberwachtem Lernen. Allerdings ist die Mächtigkeit des Systems beschränkt.

- Die Delta-Regel vergleicht die tatsächliche und die erwartete Ausgabe des Netzes.

  $$\delta = y_{i,erwartet} - y_{i,tastchlich} \tag{3.59}$$

  Dadurch ist diese Regel nur für überwachtes Lernen in Netzen ohne Hiddenschicht geeignet. Aus der Änderung des Verbindungsgewichts um

  $$\Delta w_{ij} = \epsilon \delta_i y_j \tag{3.60}$$

  erkennt man, dass das Verbindungsgewicht erhöht wird bei zu niedriger Ausgabe und verringert wird bei zu hoher Ausgabe. Stimmen tatsächlicher und erwarteter Wert überein, wird keine Änderung vorgenommen. Der Fehlerparameter ϵ gibt die Stärke der Gewichtsänderung an.

- Ein Lernverfahren für unüberwachtes Lernen ist das Competitive Learning. Dabei wird die Ausgabe der Outputschicht erzeugt und anschließend der größte

Aktivierungswert an einem Output-Neuron ermittelt. Alle Verbindungsgewichte die zu dem Neuron mit dem größten Aktivierungswert führen werden im letzten Schritt so angepasst, dass sie dem Input des Neurons ähnlicher werden.

- Die Backpropagation ermöglicht es KNN mit einer oder mehreren Hiddenschichten zu trainieren. Der Algorithmus ist in 3 Schritte aufgeteilt.

  1. Forward-Pass: Anhand von Trainingsdaten wird die Ausgabe der Output-Neuronen ermittelt.
  2. Fehlerbestimmung: Die Output-Werte aus dem vorherigen Schritt werden mit den erwarteten Werten verglichen. Überschreiten diese eine Fehlerschwelle, folgt der Backward-Pass. Sonst kann die Trainingsphase beendet werden.
  3. Backward-Pass: Beginnend bei der Ausgabeschicht werden Fehlerterme bis zur Inputschicht ermittelt. Anhand der Fehlerterme können die Verbindungsgewichte durch ein Gradientenabstiegsverfahren optimiert werden.

Als Lernregel des MLPs der Implementierung wurde die Backpropagation gewählt. Diese bietet die Möglichkeit, eine feste Lernrate und ein Momentum eines Trägheitsterms anzugeben. Die feste Lernrate η ist ein Faktor, der mit dem berechneten Gradientengewicht multipliziert wird und somit die Stärke der Änderung Δw_{ij} des Verbindungsgewichtes w_{ij} zwischen zwei Neuronen anpasst.

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}} = \eta \delta_j x_i \tag{3.61}$$

Dabei ist E die Fehlerfunktion und δ_j das Fehlersignal des Neurons j. Der Wert der Lernrate wurde zu $\eta = 0.1$ gewählt.

Der Trägheitsterm bietet die Möglichkeit, das Änderungsgewicht der vorherigen Iteration $\Delta w_{ij}(t)$ in die Ermittlung des aktuellen Gewichtes $\Delta w_{ij}(t+1)$ mit einzubeziehen.

$$\Delta w_{ij}(t+1) = \eta \delta_j x_i + \alpha \Delta w_{ij}(t) \tag{3.62}$$

Das Momentum α ist ein Faktor der den alten Wert $\Delta w_{ij}(t)$ gewichtet. Somit wird bei $\alpha = 0$ nur der Gradient zur Bestimmung der Änderung genutzt und bei steigendem α stärker $\Delta w_{ij}(t)$ mit einbezogen. Bei der Implementierung wurde $\alpha = 0.1$ gewählt.

Das Abbruchkriterium der Lernphase kann wie bei der SVM anhand der Anzahl der Iterationen des Lernverfahrens gewählt werden, oder wenn die Verbesserung des Fehlers eine gegebene Schwelle nicht übersteigt. Das KNN erreicht gute Ergebnisse bei der Klassifizierung bereits bei 400 Iterationen und einer Genauigkeit von $\epsilon = 10^{-4}$.

3.8 Automatische Attributwerterzeugung

Bei der Verwendung von maschinellen Lernverfahren (Kapitel 3.7) wird für jede neue Objektklasse, die klassifiziert werden soll, eine Menge an Trainingsbildern benötigt. Aus diesen Bildern werden Merkmale ermittelt, anhand derer der Klassifikator trainiert wird. Mit den hier gewählten Attributen ist es möglich, die Merkmale eines Objekts direkt von Personen bestimmen zu lassen. Anhand der Eingaben, die von einer kleinen Anzahl Personen gemacht wird, soll eine Datenbasis ermittelt werden, die es ermöglicht, den Klassifikator des lernenden Systems ohne Beispielbilder zu trainieren, um durch einen vergleichsweise geringen Aufwand die Klassifizierung neuer Objekte zu ermöglichen.

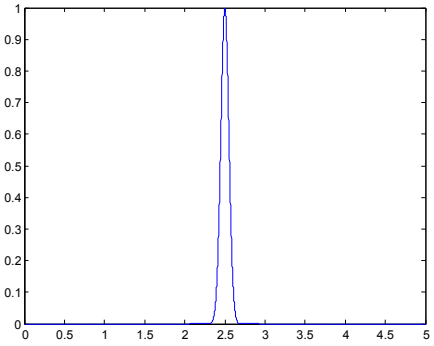
3.8.1 Prinzip

Anhand der Datenbank A.3 sowie der maschinell ermittelten Attributwerte erkennt man, dass identische Objekte bereits bei leicht unterschiedlichen Aufnahmen variierende Attributwerte aufweisen. Diese Varianz lässt sich unter anderem auf die subjektive Wahrnehmung der Personen zurückführen, die die Bewertung vorgenommen haben. Ist das Attribut gut bestimmbar, ist das Intervall in dem die Werte variieren klein. Ebenso steigt die Intervallgröße bei schwierig zu beschreibenden Attributen. Diese Ei-

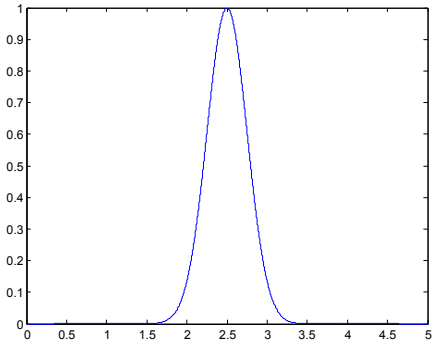
genschaft ermöglicht es, auf Basis der Eingabe weniger Personen eine Vielzahl von zufällig erzeugten Attributwerten zu erzeugen, die als Datenbasis für den Trainingsprozess des lernenden Systems dient. Um diese Streuung der Attributwerte möglichst genau darstellen zu können, ist es sinnvoll, dass mindestens zwei Personen die Bewertung vornehmen. Es wird zum einen für jedes Attribut der entsprechende Wert sowie die Eindeutigkeit bestimmt. Das als Eindeutigkeit benannte Maß dient dazu, die Streuung der Attributwerte zu bewerten. Die Einstufungen sind keine Streuung (no/low variety), wenig Streuung (moderate variety), viel Streuung (high variety) und willkürlich (arbitrary). Aus allen Bewertungen werden die Mittelwerte der Attribute und deren Eindeutigkeit bestimmt. Diese dienen als Parameter für die Erzeugung der neuen Attributwerte, die per Zufall in einer Gaußverteilung erzeugt werden. Der Mittelwert der bewerteten Attribute ist dabei der Erwartungswert μ. Die Varianz σ der Normalverteilung wird anhand des Mittelwerts der Bestimmbarkeit gewählt. Die vier Abstufungen weisen dabei in absteigender Strenge die Varianzen $\sigma = 0.05$, $\sigma = 0.25$, $\sigma = 1$ und für die schwächste Einstufung eine Gleichverteilung auf (Abbildung 3.13). Eine genau beschreibbares Attribut weist eine minimale Varianz auf (Abbildung 3.13 (a)), während in der letzten Abstufung (Abbildung 3.13 (d)) von einem Attribut ausgegangen wird, das nicht mehr genau eingeordnet werden kann, und somit Zufallswerte im gesamten Wertebereich des Attributs erzeugt werden. Treten Zufallszahlen auf, die außerhalb des Wertebereichs liegen, werden diese verworfen und neue generiert. So wird sichergestellt, dass Verteilungen mit Mittelwert in der Nähe der Intervallgrenzen ihre typische Form behalten (Abbildung 3.14).

3.8.2 Umsetzung

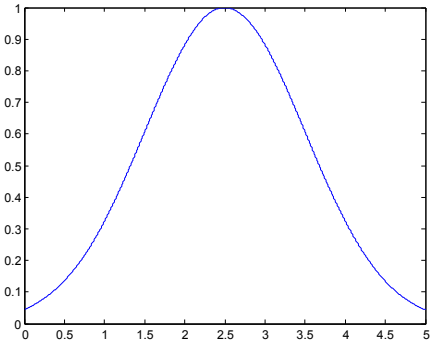
Die Erweiterung des Klassifizierers um ein neues Objekt soll vom Endbenutzer durchgeführt werden. Deshalb wird eine grafische Oberfläche genutzt, um mit wenig Vorkenntnissen und geringem Aufwand die benötigten Daten zu erzeugen. Implementiert ist die Anwendung in der Software MATLAB®. Zwei Funktionen, die jeweils über ein Grafisches User Interface (GUI) verfügen, ermöglichen zum einen die Eingabe durch den



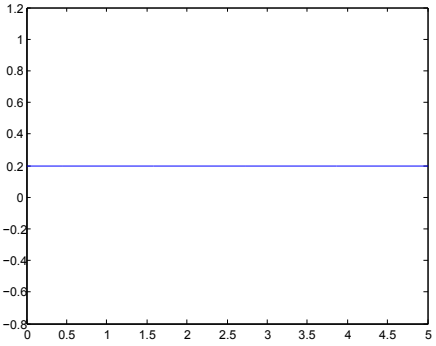
(a) low/no variety $\sigma = 0.05$



(b) modertate variety $\sigma = 0.25$



(c) high variety $\sigma = 1$



(d) arbitrary / Gleichverteilt

Abbildung 3.13: Verteilungen der auftretenden Zufallszahlen.

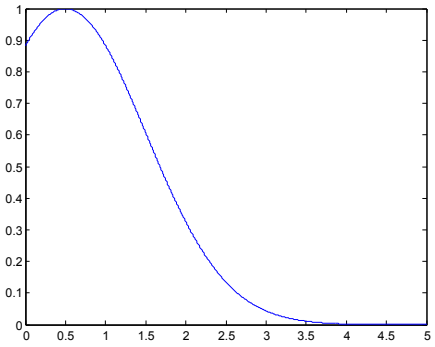


Abbildung 3.14: Verteilung mit Parameter $\sigma = 1$ und $\mu = 0.5$.

Benutzer, zum anderen die Erzeugung der benötigten Trainingsdaten im gewünschten Umfang und in einer definierten Formatierung. Bei der Eingabe (Abbildung 3.15) sind dem Benutzer diskrete Werte entsprechend den Attributen aus Kapitel 3.6 gegeben. Ebenfalls können die vier Stufen der Bestimmbarkeit des Attributs gewählt werden. Ist der Name des Objekts und ein Pfad zum Speichern der Daten angegeben, wird beim Betätigen der "Save values" Schaltfläche eine Datei mit den Eingaben angelegt oder eine vorhandene Datei mit den Daten erweitert. Neue Daten können über die GUI in Abbildung 3.16 erstellt werden. Dabei kann eine einzelne Datei mit den gespeicherten Bewertungen angegeben werden oder ein Ordner, der mehrere solche Dateien enthält. Benötigt wird auch die Datei mit den bisherigen Attributwerten, die um die neuen Klassen erweitert wird. Wie viele Objekte in einer Klasse erzeugt werden sollen, muss vor der Generierung der Daten angegeben werden. Sind alle Informationen vorhanden, kann über die jeweilige Schaltfläche "Compute new data" bzw. "Compute new data out of folder" der Algorithmus gestartet werden.

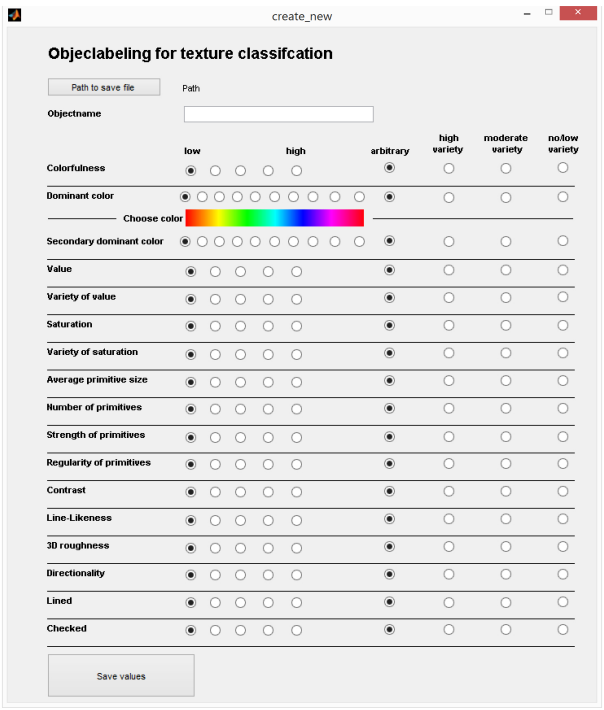


Abbildung 3.15: GUI zur Bewertung neuer Objektattribute.

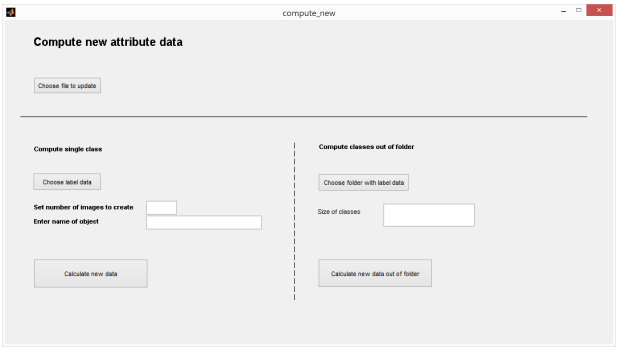


Abbildung 3.16: GUI zur Erzeugung der Attributwerte.

Kapitel 4

Evaluierung

4.1 Segmentierung

Nach [Zhang, 1996] ist es möglich, die Evaluation von Segmentierungsalgorithmen in drei Klassen einzuteilen. Diese sind “analytisch“, “empirische Güte“ und “empirische Diskrepanz“. Die Einteilung basiert auf den unterschiedlichen Vorraussetzungen der Methoden. Bei der analytischen Methode wird der Algorithmus auf dessen Prinzip, Anforderung und Komplexität untersucht. Eine Betrachtung des Ergebnisses des Algorithmus ist nicht nötig, weshalb eine Implementierung nicht vorhanden sein muss. Im Gegensatz dazu bewerten die empirischen Methoden das Ergebnis des Algorithmus. Dazu müssen eine Implementierung des Algorithmus und Testdaten vorhanden sein. Die empirische Diskrepanz Methode vergleicht die Ergebnisse mehrerer Segmentierungsalgorithmen und einer Referenzsegmentierung. Dadurch ist es möglich, Empfindlichkeit, Präzision und Laufzeit der Algorithmen bei unterschiedlichen Bedingungen und Parametereinstellungen zu vergleichen. Empirische Güte Methoden vergleichen dagegen nicht die Ergebnisse mit einer Referenzsegmentierung, sondern werden durch Experten oder anhand von ihnen aufgestellten Kriterien bewertet. Für die Bewertung der in dieser Arbeit vorliegenden Segmentierung wird die empirische Diskrepanz Methode gewählt, die auf einen dafür erstellten Testdatensatz angewendet wird. Die

Anforderung Texturen zu klassifizieren setzt eine Segmentierung voraus, die Bereiche mit homogener Textur segmentiert. Dazu werden die Testdaten nach diesem Kriterium händisch segmentiert und anschließend mit den Ergebnissen des Algorithmus verglichen.

Die erhaltenen Segmente werden Pixelweise mit den Referenzsegmenten verglichen. Dabei können vier unterschiedliche Ergebnisse auftreten:

- Richtig positiv (RP): Der Pixel ist Teil des Segments sowie Teil der Referenz.
- Falsch negativ (FN): Der Pixel ist nicht Teil des Segments aber Teil der Referenz.
- Richtig negativ (RN): Der Pixel ist nicht Teil des Segments und nicht Teil der Referenz.
- Falsch positiv (FP): Der Pixel ist Teil des Segments aber nicht Teil der Referenz.

Diese Häufigkeiten können dazu genutzt werden um Kenngrößen zu ermitteln, anhand derer die Qualität der Segmentierung bewertet werden kann. Die hier verwendeten Kenngrößen sind der positive Vorhersagewert, die Sensitivität, und das F-Maß.

Der positive Vorhersagewert oder die Genauigkeit (engl. Precision) gibt an, wie stark das ermittelte Segment das Referenzsegment überdeckt.

$$Precision = \frac{RP}{RP + FP} \tag{4.1}$$

Die Sensitivität oder Trefferquote (engl. Recall) gibt die Wahrscheinlichkeit an, dass ein richtig zugeordnetes Pixel Teil des Segments ist. Somit ist dies das Verhältnis zwischen der Fläche des korrekt segmentierten Segments zu der gesamten korrekt segmentierten Fläche.

$$Recall = \frac{RP}{RP + FN} \tag{4.2}$$

Um Korrelationseffekte zwischen Kenngrößen zu vermindern werden kombinierte Maße verwendet. Ein solches Maß ist das F-Maß (engl. F-score). Anhand des harmonischen

Mittels wird das Maß aus der Kombination von positivem Vorhersagewert und Sensitivität ermittelt:

$$F = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}. \tag{4.3}$$

4.1.1 RGB Segmentierung

Der Split and Merge Algorithmus besteht aus zwei unabhängigen Prozeduren. Während die Split Prozedur in diesem Kapitel einzeln betrachtet wird, untersucht Kapitel 4.1.2 unterschiedliche Merge Prozeduren in Kombination mit der Tiefensegmentierung und führt gleichzeitig ein Vergleich mit dieser durch.

Split Bedingung

Das Homogenitätskriterium des Split and Merge Algorithmus entscheidet ob ein Bildbereich homogen ist und somit nicht weiter unterteilt werden muss. Drei implementierte Kriterien werden auf ihre Fähigkeit untersucht möglichst große homogene Regionen zu identifizieren. Das erste Kriterium prüft, wie in Kapitel 3.4.2, ob die LBP-Histogramme der vier Teilregionen der betrachteten Region anhand des Chi-Quadrat-Homogenitätstest aussreichend ähnlich sind. Das zweite Kriterium ermittelt die Summe der Standardabweichungen der drei Farbkanäle R, G und B in der betrachteten Bildregion. Wird ein eingestellter Schwellwert nicht überschritten, ist die Bedingung erfüllt. Das dritte Kriterium ist eine Kombination der ersten beiden Kriterien und wird in Kapitel 3.4.2 beschrieben.

Für die Evaluierung der Kriterien wird die Split Prozedur jeweils mit einem der drei Kriterien auf zwei Bildern (Abbildung A.1 (a), Abbildung A.2 (a)) ausgeführt. Die erhaltenen Regionen werden anhand von Referenzsegmenten zusammengefügt. Dabei gehört die Region zu dem Segment, das den größten Anteil an der Region hält. Die Übereinstimmung der erhaltenen Segmente und der Referenzsegmente wird über die Precision, dem Recall und dem F-Maß bewertet.

Die Ergebnisse der Auswertung für Abbildung A.1 (a) und Abbildung A.2 (a) stehen in Tabelle A.1 sowie Tabelle A.2. Die Precision und der Recall für jedes Segment wird in einem Diagramm (PR-Diagramm) in Abbildung 4.1 und Abbildung 4.2 dargestellt. Die F-Maße der Segmente werden in Abbildung 4.3 und Abbildung 4.4 verglichen. Eine Visualisierung der Regionen und der daraus resultierenden Segmente ist in Abbildung A.1 und Abbildung A.2 zu finden.

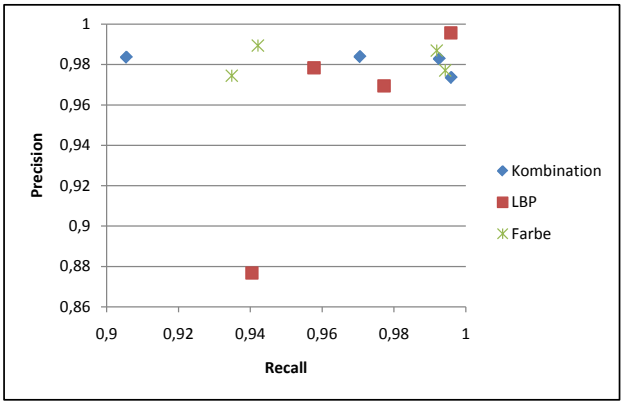


Abbildung 4.1: Segmente aus Bild 1 im PR Diagramm.

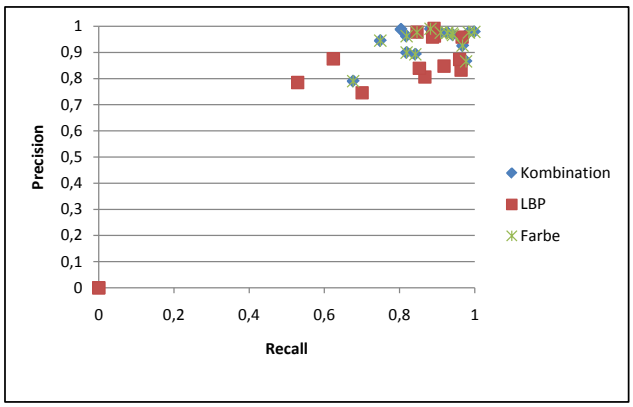


Abbildung 4.2: Segmente aus Bild 2 im PR Diagramm.

Betrachtet man die beiden PR-Diagramme erkennt man, dass die drei Kriterien hauptsächlich im oberen, rechten Bereich des Diagramms vertreten sind. Segmente die durch eine hohe Precision und Recall in diesem Bereich auftreten, weisen wenig falsch positiv und falsch negativ fälle auf. Aus Abbildung 4.1 ist eine Bewertung schwierig

da das kombinierte sowie das LBP Kriterium jeweils ein Segment aufweist, dass vergleichsweise schlecht abschneidet. In Abbildung 4.2 liefern das kombinierte Kriterium und das Farbkriterium annähernd gleiche Ergebnisse. Das LBP Kriterium Konnte hier zwei Segmente nicht erkennen und erreichte bei mehreren nur mäßige Ergebnisse. Eine genaue Bewertung anhand der PR-Diagramme ist hier schwierig.

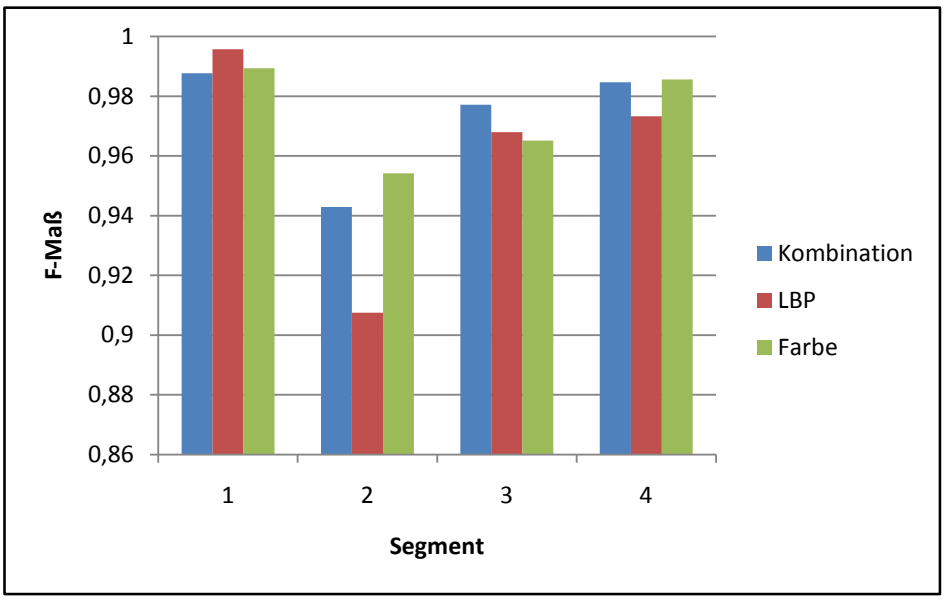


Abbildung 4.3: F-Maß der Segmente aus Bild 1.

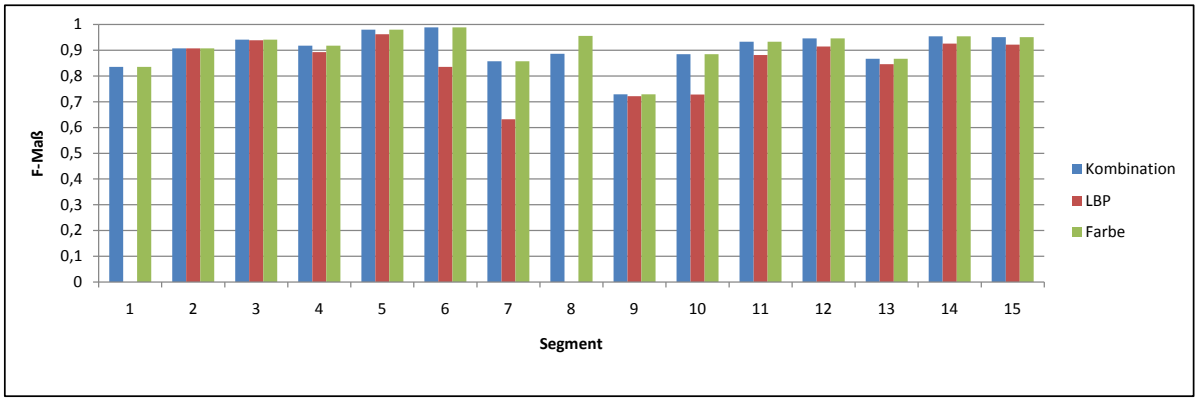


Abbildung 4.4: F-Maß der Segmente aus Bild 2.

Die F-Maße in Abbildung 4.3 und Abbildung 4.4 zeigen, dass das LBP Kriterium im

Vergleich zu den beiden anderen Kriterien schlechter abschneidet. Während in Abbildung 4.4 das kombinierte Kriterium und das Farbkriterium ähnliche Werte erreichen, schneidet in Abbildung 4.3 das Farbkriterium besser ab.

Tabelle 4.1: Anzahl der erzeugten Regionen.

Bild 1	Kombination	LBP	Farbe
homogen Regionen	922	183	4038
nicht homogene Regionen	68	52	8
Anzahl der entstandenen Regionen	990	235	4046
Bild 2			
homogen Regionen	413	318	444
nicht homogene Regionen	194	79	198
Anzahl der entstandenen Regionen	607	397	642

Aus Tabelle 4.1 kann ebenfalls geschlossen werden, dass das Farbkriterium bei Bild A.1(a) das beste Ergebnis liefert aufgrund der geringen Anzahl an nicht homogenen Regionen. Hier wird allerdings deutlich, welche Nachteile das Farbkriterium birgt. Szenen, in denen viele unterschiedliche Farben auftreten(Abbildung A.1(a)), werden sehr stark unterteilt, sodass viele kleinen Regionen entstehen(Abbildung A.1(g)). Informationen über Textur können in solch kleinen Regionen nicht ermittelt werden. Zudem entsteht ein enormer Rechenaufwand um alle Regionen in der Merge Phase einem Segment zu zu ordnen. Das LBP Kriterium dagegen ist in der Lage solche Strukturen zu erkennen, und als homogene Region einzustufen (Abbildung A.1(e)). Der LBP Operator kann allerdings nur Regionen mit Texturen zuverlässig bewerten. Auf einfarbigen Flächen liegen die Grauwerte dicht beieinander. Dies führt dazu, dass Texturmuster ermittelt werden, die nur aus Schwankungen der Helligkeit resultieren. Dadurch werden homogene Bereiche stärker unterteilt als es nötig wäre (Abbildung A.2(e)). Das kombinierte Kriterium ermöglicht es große Texturen und homogenen Flächen zu erkennen. Gleichzeitig werden Ränder von Segmenten anhand der Farbwerte genau bestimmt. Dies ist wichtig um in der Merge Phase Texturinformationen auf großen Flächen zu erhalten und gleichzeitig Segmenträndern genau darstellen zu können. Somit erfüllt das kombinierte Homogenitätskriterium die Anforderungen die an die Split Phase gestellt werden am besten.

Merge Bedingung

Die Merge Kriterien werden auf den Ergebnissen der Tiefensegmentierung ausgewertet. Dabei werden zwei Ansätze betrachtet. Die erste Methode bewertet mit Hilfe der absoluten Differenzen der Farbattribute aus Kapitel 3.6, wie stark die Regionen sich unterscheiden. Der zweite Ansatz ermittelt anhand aller Texturattribute die Homogenität wie in Kapitel 3.4.2 beschrieben.

Auf eine Untersuchung eines Homogenitätskriteriums basierend auf LBP-Histogrammen wie bei der Split Bedingung wurde verzichtet, da sich bereits gezeigt hat, dass einfarbige Bereiche nicht zuverlässig erkannt werden können.

4.1.2 RGB-D Segmentierung

Die Tiefensegmentierung wurde bereits in [Arbeiter et al., 2014] Bewertet und liefert gute Ergebnisse in der Segmentierung und der Rechenzeit. Ebenen und Flächen mit einheitliche Krümmung werden wie in Abbildung 4.5 zuverlässig und genau Segmentiert.

(a) Farbbild (b) Segmentiertes Bild

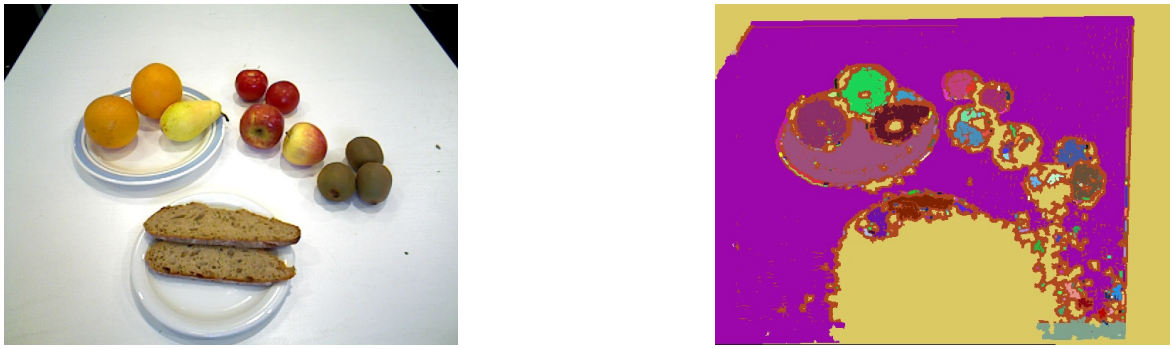
Abbildung 4.5: Links: Originalbild. Rechts: Erhaltene Segmente durch die Tiefensegmentierung nach [Arbeiter et al., 2014].

Weisen die Objekte variierende Krümmungen auf einer kleinen Fläche auf, werden

diese nicht erkannt oder in mehrere Segmente aufgeteilt. Dies tritt Beispielsweise bei Früchten auf, wie in Abbildung 4.6.

Ähnliche Probleme entstehen durch Objekte die nicht aus großen, zusammenhängenden Oberflächen bestehen wie z.B. Pflanzen. In Abbildung 4.5 wird die Topfpflanze in mehrere Segmente unterteilt. Das größte Segment beinhaltet den Topf sowie ein Teil der Pflanze und ist damit falsch segmentiert.

Befindet sich die Kamera zu nahe an der aufgenommenen Szene, ist es nicht mehr möglich Tiefendaten zu erfassen. In Abbildung 4.6(b) können, im vorderen mittleren Bereich, aufgrund der Nähe keine Tiefendaten ermittelt werden. Ebenfalls wurden im gleichfarbigen Bereich am Rand des Bildes keine Tiefeninformationen erfasst, da der Tiefensensor diese Bereiche nicht abdeckt.



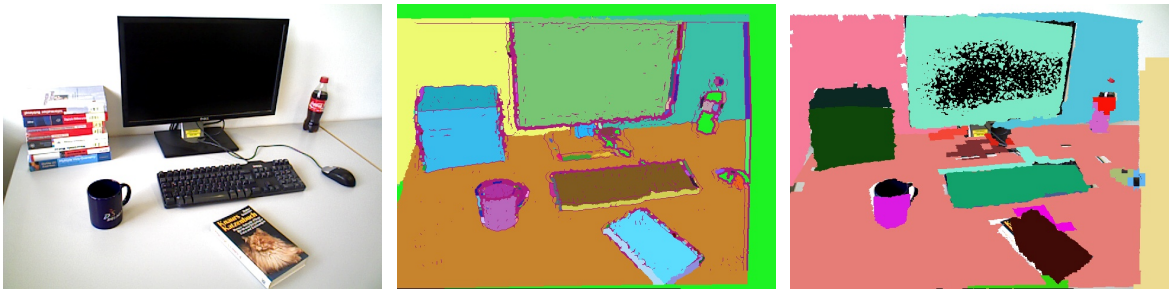
(a) Farbbild (b) Segmentiertes Bild

Abbildung 4.6: Links: Originalbild. Rechts: Erhaltene Segmente durch die Tiefensegmentierung nach [Arbeiter et al., 2014].

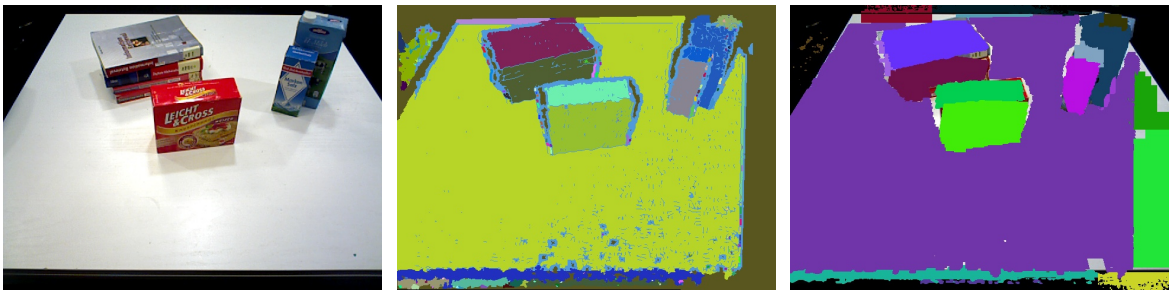
Fehler der Tiefensegmentierung die durch nicht auswertbare Tiefendaten entstehen, sollen mit Hilfe der Split and Merge Segmentierung vermieden werden. Dazu werden alle Bereiche in denen keine Tiefendaten vorhanden sind, sehr kleine Segmente und Segmente mit wenigen oder verrauschten Tiefendaten in einer Darstellung zusammengefasst. Diese wird anhand der Farb- und Texturmerkmale, durch den Split and Merge Algorithmus, segmentiert. Die restlichen, ermittelten Segmente werden nach Kapitel 3.5.3 in eine normalisierte Ansicht transformiert und ebenfalls segmentiert. Dadurch

sollen unterschiedliche Texturen auf einer Fläche erkannt werden.

In Szenen die aus glatten Oberflächen bestehen, erzielt die Tiefensegmentierung optimale Ergebnisse. Die weitere Bearbeitung der Segmente in einem solchen Fall, führt zu leichten Qualitätsverlusten an den Segmenträndern. Dies erkennt man in Abbildung 4.7 an den nicht zugewiesenen Bereichen zwischen einzelnen Segmenten.



(a) Farbbild (b) Tiefensegmentierung (c) Tiefen- und Textursegmentierung



(d) Farbbild (e) Tiefensegmentierung (f) Tiefen- und Textursegmentierung

Abbildung 4.7: Links: Originalbild. Mitte: Erhaltene Segmente anhand Tiefensegmentierung. Rechts: Erhaltene Segmente anhand RGB-D Segmentierung.

Die Vorteile der zweistufigen Segmentierung wird deutlich, wenn in den Szenen die vorherig genannten Problematiken der Tiefendaten auftreten.

Bei einigen Früchten in beiden Szenen der Abbildung 4.8 ist die Tiefensegmentierung nicht in der Lage, auf Grund der Krümmung, eine korrekte Segmentierung vorzunehmen. Speziell an den äußeren Bereichen der Früchte entstehen häufig kleine Segmente die durch unterschiedliche Färbung gekennzeichnet sind. Der Split and Merge Algo-

rithmus erreicht auf diesen kleinen Ausschnitten sehr gute Ergebnisse. Lediglich in Abbildung 4.8 (f) werden die roten Äpfel und die braunen Kiwis identisch eingestuft. Dies ist zurückzuführen auf die identische Form und Anordnung der Früchte sowie den nur geringen Größenunterschied. Ebenfalls wird hier deutlich, dass durch die Farbtonskala im HSV-Farbraum Farben mit ähnlichen Wellenlängen nur schwer unterschieden werden können.

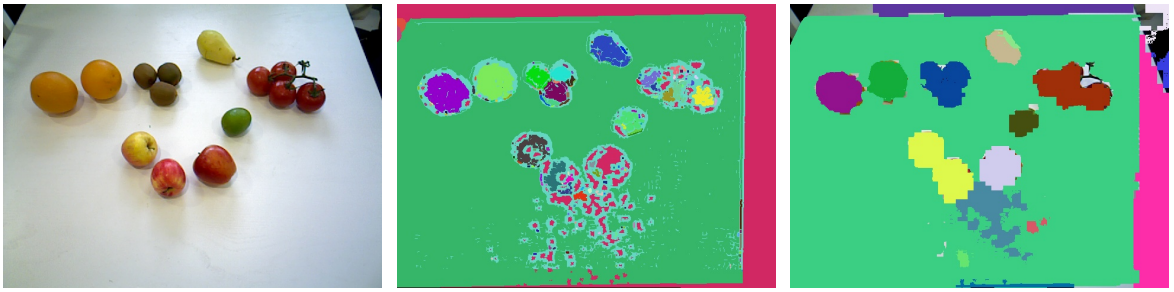
Die Trauben in Abbildung 4.8(d) erzeugen das gleiche Problem wie die Topfpflanze in Abbildung 4.5. Hier kann durch den zweiten Segmentierungsschritt eine genaue Abgrenzung zwischen Trauben und Teller erreicht werden.

In Abbildung 4.8(b) kommt es durch die Nähe der Szene im mittleren vorderen Bereich des Bildes zu fehlenden Tiefendaten. Durch die weitere Segmentierung werden in Abbildung 4.8(c) die angrenzenden Früchte erkannt. Der Bereich des Tisches wird auf Grund der großen Menge der fehlenden Tiefendaten in dem kleinen Bereich zu einem eigenen Segment zusammengefasst. Die fehlenden Tiefendaten am Rand beider Szenen werden ebenfalls erfolgreich durch den zweiten Segmentierungsschritt abgefangen und in Tisch und Hintergrund segmentiert.

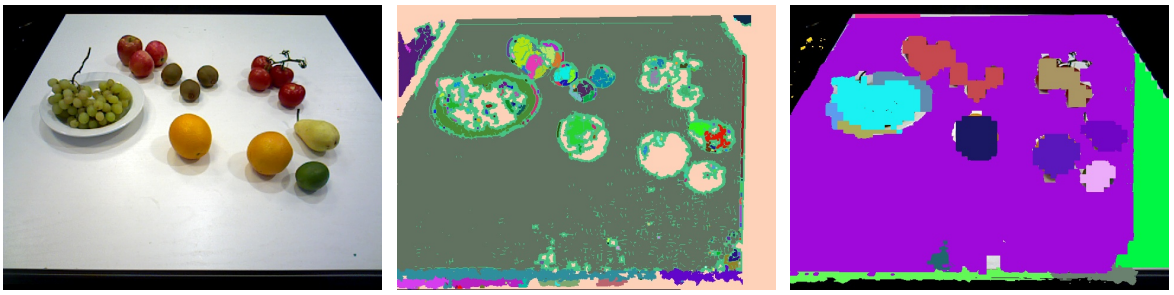
Bewertung

Die Bewertung der RGB-D Segmentierung wird mit dem kombinierten Split Kriterium durchgeführt. Dieses wurde bereits als stärkstes Kriterium für den Splitvorgang ermittelt und stellt damit den unterschiedlichen Mergekriterien die selben Daten zur Verfügung. Als Homogenitätskriterien für den Mergeprozess werden die Farbattribute sowie die Texturattribute gewählt. Die beiden RGB-D Segmentierungen werden zudem mit der einfachen Tiefensegmentierung verglichen um eine Leistungssteigerung darlegen zu können.

Die drei Algorithmen segmentieren jeweils sieben Szenen. Diese sind in Abbildung A.3 von oben nach unten angeordnet. Von links ist das Originalbild, die Farbseg-



(a) Farbbild (b) Tiefensegmentierung (c) Tiefen- und Textursegmentierung



(d) Farbbild (e) Tiefensegmentierung (f) Tiefen- und Textursegmentierung

Abbildung 4.8: Links: Originalbild. Mitte: Erhaltene Segmente anhand Tiefensegmentierung. Rechts: Erhaltene Segmente anhand RGB-D Segmentierung.

mentierung, die Textursegmentierung und die Tiefensegmentierung dargestellt. Die Ergebnisse werden wie bereits in Abschnitt 4.1.1 ausgewertet und in Tabelle A.3 aufgetragen. Zusammenfassend werden die Mittelwerte der Kenngrößen Precision, Recall und F-Maß über die Segmente der Bilder in Tabelle 4.2 dargestellt.

Die besten Ergebnisse bei der Precision erreicht die Tiefesegmentierungen. Bei insgesamt vier Bilder wird mit dieser Methode der beste Wert erreicht.

Die Kennzahl Recall und das F-Maß werden wesentlich besser durch die zweistufigen Segmentierung dargestellt. Das Texturkriterium erreicht fünf mal die Spitzenwerte während das Farbkriterium sich zwei mal durchsetzt (Abbildung 4.9).

Für die Berechnung der Texturattribute ist es wichtig, dass das Segment nur eine

Textur enthält. Eine nicht vollständige Erkennung des Segments hat dagegen weniger Einfluss auf das Ergebnis der Berechnung, da lediglich die Anzahl der vorhandenen Primitives geringer ist und nicht unterschiedliche vorhanden sind.

Anhand der Tabelle 4.2 erkennt man, dass das kombinierte Homogenitätskriterium am besten für die gestellte Aufgabe geeignet ist. Die hohen Recall Werte verdeutlichen, dass die durch dieses Kriterium ermittelten Segmente die größte Homogenität aufweisen und somit die genauste Berechnung der Attributwerte möglich ist. Im Gegensatz zur Tiefensegmentierung werden die Segmente weniger gut erkannt, was aus den niedrigeren Precision Werten geschlossen werden kann. Diese weisen allerdings noch gute Ergebnisse auf wodurch das kombinierte Kriterium bei fünf Bildern das beste F-Maße erreicht. Es ist sinnvoll, im zweiten Segmentierungsschritt einen geringe Verringerung der Precision in kauf zu nehmen. Über einen verbesserten Recall kann damit ein, für diese Aufgabe, optimiertes Gesamtergebnis erhalten werden.

Tabelle 4.2: Mittlere Werte der Kenngrößen Precision, Recall und F-Maß für die drei untersuchten Methoden.

Durschnittswerte		Bild 1	Bild 2	Bild 3	Bild 4	Bild 5	Bild 6	Bild 7
Farbe	Precision	0,8304	0,8014	0,8637	0,7282	0,7848	0,6897	0,7961
	Recall	0,7457	0,7909	0,6406	0,5878	0,7593	0,5275	0,7059
	F-Maß	0,7682	0,7717	0,7162	0,6282	0,7697	0,5774	0,7378
Kombination	Precision	0,8932	0,7939	0,8610	0,7846	0,8047	0,6890	0,6954
	Recall	0,7767	0,8065	0,6660	0,6928	0,7417	0,5322	0,6288
	F-Maß	0,8089	0,7763	0,7341	0,6965	0,7579	0,5833	0,6430
Tiefe	Precision	0,7472	0,9086	0,7823	0,7965	0,8620	0,7546	0,6639
	Recall	0,6858	0,6062	0,5876	0,4715	0,5920	0,4814	0,5586
	F-Maß	0,6934	0,6442	0,6416	0,5461	0,6326	0,5527	0,5877

4.2 Texturattribute

Die maschinell erzeugten Werte der Texturattribute sollen möglichst dem subjektiven Empfinden von Menschen entsprechen. Inwieweit diese Anforderung erfüllt ist, kann

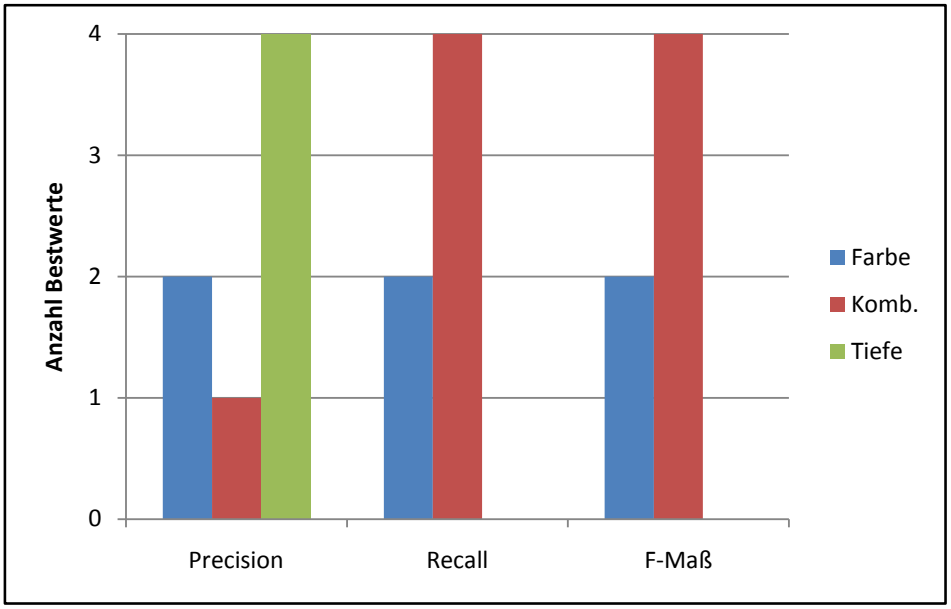


Abbildung 4.9: Anzahl der erreichten Bestwerte auf den Testbildern für die untersuchten Kenngrößen.

durch den Vergleich von maschinell erzeugten Attributwerten und händisch bewerteten Attributen einer Textur ermittelt werden. Eine Datenbank mit Bildern von Texturen auf denen die Attributwerte, anhand der in C++ implementierten Texturattribute ausgewertet werden, und den entsprechenden händisch erzeugten Attributwerten des Fraunhofer IPA dienen hierfür als Grundlage.

Texturdatenbank

Die für die Evaluierung verwendete Datenbank A.3 wurde in [Esslinger, 2014] beschrieben und besteht aus 1281 Bildern von Texturen aus dem häuslichen Umfeld, die in 57 Klassen unterteilt sind (Abbildung 4.10). Die Klassen bestehen wiederum aus mehreren Objekten denen mehrere Ansichten zugeordnet sind, die unter ähnlichen Bedingungen wie Entfernung zur Oberfläche, Blickwinkel oder Anzahl der sichtbaren Objekte aufgenommen sind. Die im JPEG Format gespeicherten Bilder entsprechen der Auflösung 2592x1944 Pixel. Für jedes Bild wurden die Attribute von zwei Personen

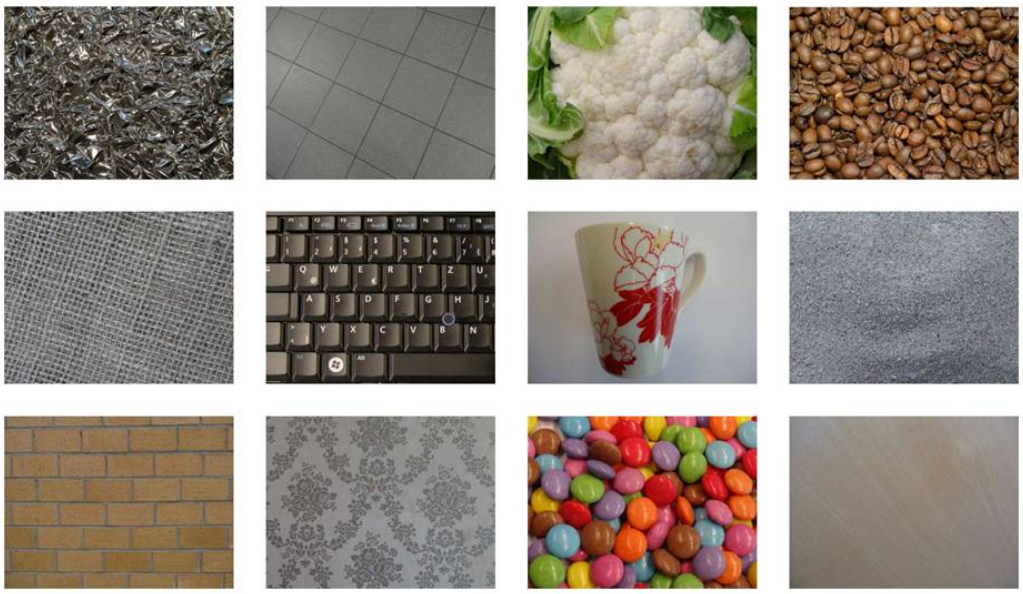


Abbildung 4.10: Unterschiedliche Klassen der Datenbank.

bestimmt und der Mittelwert der manuellen Bewertung als Attributwert bestimmt.

Auswertung

Als Maß für die Bewertung der Attribute $a_i = [1, ..., 17]$ dient die absolute Differenz d zwischen dem maschinell erzeugten Attributwert a_m und dem händisch bewerteten Attributwert a_h.

$$d = ||a_m - a_h|| \tag{4.4}$$

Für jedes der 17 Attribute wird der Mittelwert der absoluten Differenz über die 1281 Bilder der Datenbank gebildet und in Tabelle 4.3 aufgetragen.

$$\bar{x}_{a_i} = \frac{d_0 + d_1 + ... + d_{1280}}{1281} \tag{4.5}$$

Die Attribute sind auf den Wertebereich $W =: [1, ..., 5]$ skaliert. Dadurch ist es möglich, anhand von $\bar{x}_{Attribut}$ die Attribute untereinander auf deren Fähigkeit, das menschliche Empfinden nachzustellen, zu vergleichen. Außerdem kann bei $\bar{x}_{Attribut} < 0.5$ von sehr guten und bei $\bar{x}_{Attribut} < 1.0$ von guten Nachstellungsfähigkeiten ausgegangen werden.

Tabelle 4.3: Absoluter mittlerer Fehler zwischen den berechneten und den manuellen Attributwerten.

Attribut	**mittlere absolute Differenz**
colorfulness	0.53
dominant color	0.47
sec. dominant color	0.68
saturation	0.51
variety of saturation	0.43
value	0.54
variety of value	0.73
average primitive size	0.76
number of primitives	0.81
strength of primitives	0.97
regularity of primitives	1.09
contrast	0.51
line-likeness	1.01
3D roughness	-
directionality	1.10
lined	0.68
checked	0.42

Diese Grenzen entsprechen einer Abweichung der ermittelten Werte von der Vorgabe um 10% sowie um 20%. Entsprechende Abweichungen treten auch bei der händischen Bewertung der Attribute auf.

Die Attribute dominante Farbe, Varianz der Sättigung sowie die Karriertheit liefern sehr gute Ergebnisse. Eine Abweichung von über 20% weisen die Attribute regelmäßigkeit der Primitives, Linienartigkeit und die Orientierung auf. Die restlichen Attribute liegen im Bereich der guten Nachstellung des menschlichen Empfindens (Tabelle 4.3).

In [Esslinger, 2014] wird diese Auswertung bereits, auf in MATLAB® implementierten Texturattributen, durchgeführt. Diese Arbeit verwendet eine C++ Implemen-

tierung der Texturattribute.

Um festzustellen wie gut die hier verwendeten Texturattribute Texturen nachstellen können, werden diese mit Methoden aus Kapitel 2.2.4 verglichen. Entsprechende Implementierungen der Methoden von [Cimpoi et al., 2014] und [Farhadi et al., 2009] werden von diesen zur Verfügung gestellt. Ausserdem werden gelernte händisch bewertete Attribute gelernt in dem jedes Attribut durch eine SVM repräsentiert wird.

Für eine zuverlässige Bewertung der Methoden wird die Leave-One-Out-Kreuzvalidierung durchgeführt. Hierbei wird aus einer oder mehreren Klassen alle Ansichten eines zufälligen Objekts entnommen und zu dem Testdatenset hinzugefügt. Die Methoden werden mit dem übrigen Trainingsdatenset trainiert und erzeugen anschließend die Texturattribute für den Testdatensatz. Dieser Vorgang wird mehrmals wiederholt und die absoluten Differenzen zwischen den erzeugten Attributwerten und den händischen Attributwerten bestimmt. Über dieses Verfahren wird sichergestellt, dass einzelne besonders gut oder schlecht darstellbare Attribute das Ergebnis nicht verfälschen.

Alle Methoden werden mit einer 20-fachen leave out one object per class cross-validation und einer 57-fachen leave out one class cross-validation ausgewertet. Somit werden bei der ersten Kreuzvalidierung alle Texturbeispiele eines Objektes dem Testdatenset hinzugefügt während in der zweiten Validierung eine Klasse aus den Trainingsdaten entnommen wird. Eine genaue Auflistung der Ergebnisse der beiden Kreuzvalidierungen findet in Tabelle A.5 und Tabelle A.6 statt. In beiden Tabellen wird im oberen Tabellenabschnitt die absolute mittlere Differenz zwischen ermittelten und bewerteten Attributwert dargestellt. Die beiden unteren Bereiche geben an, wie viel Prozent der ermittelten Attributwerte eine absolute Differenz <0.5 und <1.0 gegenüber den Testdaten aufweisen. Die besten Ergebnisse werden in Tabelle 4.4 dargestellt. Diese werden mit der Kreuzvalidierung, unter herauslassen von Objekten, erreicht. Die einfachen Texturattribute erreichen vereinzelt ähnlich gute Ergebnisse wie die anderen Methoden, sind diesen allerdings auf der Mehrzahl der Attribute unterlegen. Die attributbasierten Texturklassifizierungsverfahren weisen auf einigen Attributen sehr

Tabelle 4.4: Vergleich der Texturattribute mit attributbasierten Texturklassifizierungsverfahren.

	Händisch	*Händisch* und lernend	*Farhadi*	*Cimpoi*
	<1.0	<1.0	<1.0	<1.0
Attribute 1:	80,0622%	86,1275%	90,9176%	92,3484%
Attribute 2:	64,6034%	72,9705%	75,9565%	79,2224%
Attribute 3:	77,4184%	85,6299%	87,1851%	84,3546%
Attribute 4:	87,5583%	93,2504%	94,1213%	93,8414%
Attribute 5:	81,2753%	79,6890%	83,7947%	83,0793%
Attribute 6:	92,3173%	93,7170%	93,9658%	94,6812%
Attribute 7:	94,2768%	96,8896%	96,0809%	96,3297%
Attribute 8:	69,6112%	74,0280%	79,5645%	75,7387%
Attribute 9:	68,7714%	73,4370%	81,3064%	78,1337%
Attribute 10:	75,1166%	80,0933%	83,9191%	80,9331%
Attribute 11:	64,4168%	72,5350%	82,8305%	79,5956%
Attribute 12:	94,4635%	94,1524%	95,6454%	96,4541%
Attribute 13:	75,4277%	84,1058%	86,3453%	82,6128%
Attribute 14:			93,4681%	89,8911%
Attribute 15:	60,1244%	67,3717%	69,1446%	67,4339%
Attribute 16:	66,6563%	72,6905%	69,5490%	68,2737%
Attribute 17:	73,0949%	82,4883%	85,3188%	84,8212%

ähnliche Ergebnisse auf. Die Methode nach [Farhadi et al., 2009] stellt elf Attribute am besten dar und liefert somit das beste Ergebnis.

Für die Auswertung werden die optimalen Parameter der Texturklassifizierungsverfahren gewählt. Diese beinhalten eine SVM als maschinelles Lernverfahren für alle Methoden. Der SVM Typ ist die ν-Support Vector Regression mit einem Abbruchkriterium von 1000 Iterationen oder der Genauigkeit von $\epsilon = 10^{-6}$. Der Parameter der Kernelfunktion wird zu $\gamma = 0.1$ und für das Optimierungsproblem der SVM wird $C = 1.0$ und $\nu = 0.4$ gewählt. Lediglich bei der Kernelfunktion wird für [Farhadi et al., 2009] und der händisch gelernte Methode die Radiale Basisfunktion gewählt während für [Cimpoi et al., 2014] eine linearer Kernel zum Einsatz kommt.

Tabelle 4.5: Klassifizierungsergebnisse auf 57-fold leave one object out Kreuzvalidierung.

Klassifizierungen	Ergebnis
Neuronales Netz	77.1%
SVM	75%
Decision Tree	59%
Random Tree	55%
Bayes Classifier	33%

4.3 Klassifizierung

Die Evaluierung der Klassifizierung untersucht die Fähigkeit der in Kapitel 3.7 beschriebenen maschinellen Lernverfahren(ML), Texturen in eine Klasse einzuordnen. Das ML wird mit den Attributen der Texturen aus der in Kapitel 4.2 vorgestellten Texturdatenbank trainiert. Der Testdatensatz wird vor dem Training des Netzes aus der Texturdatenbank entnommen und nach Beendigung des Trainings Klassifiziert. Dieses vorgehen entspricht der in Kapitel 4.2 beschrieben Leave-One-Out-Kreuzvalidierung. Trainingsdatensatz und Testdatensatz wurden händisch bewertet. Somit erhält man durch diese Auswertung den theoretischen Grenzwert der erreichbaren Klassifizierungsgüte, da die berechneten Attribute nicht perfekt mit den gelabelten übereinstimmen. In einem Validierungsvorgang wird aus einer Klasse alle Ansichten eines Objektes als Testdatensatz bestimmt. Die restlichen Daten trainieren als Trainingsdatensatz den Klassifizierer der im Anschluss den Testdatensatz klassifiziert. Dies wird für jede Klasse wiederholt wodurch die Auswertung der 57-fold leave one object out cross-validation entspricht. Anhand des Verhältnisses zwischen richtig klassifizierten Trainingsdaten zu allen Trainingsdaten (entspricht der Sensitivität aus Kapitel 4.1) wird die Klassifizierung bewertet. Die Kreuzvalidierung wird auf fünf ML Verfahren durchgeführt und die Ergebnisse in Tabelle 4.5 angeführt.

Das Neuronale Netz und die SVM können deutlich besser die Testdaten Klassifizieren als die restlichen Verfahren. Das Neuronale Netz besteht aus drei Schichten. Die

Tabelle 4.6: Klassifizierungsergebnisse auf 20-fold leave one object out Kreuzvalidierung.

looo 20-fold:	Händisch	Händ. und lernend	Farhadi	Cimpoi
Trainiert mit bewerteten Att.	6.24512%	27.0607%	44.1991%	44.6656%
Trainiert mit berechneten Att.	40.0311%	35.832%	50.9176%	53.0638%

Eingabeschicht enthält 16 oder 17 Neuronen, da das Attribut "3D roughness" nicht implementiert allerdings auf der Datenbank bewertet wurde. Die versteckte Schicht enthält 400 Neuronen und die Ausgabeschicht beinhaltet 57 Neuronen. Als Aktivierungsfunktion wurde die Sigmoidfunktion mit den Paramtern $\alpha = 0.4$ und $\beta = 1.2$ gewählt. Die Vorhersage des Neuronalen Netzes ist das Neuron mit dem maximalen Ausgabe ist.

Die SVM ist eine C-Support Vector Classification mit linearem Kernel.

Für beide Verfahren ist das Abbruchkriterium 1000 Iterationen oder eine Genauigkeit von $\epsilon = 10^{-6}$

In einer weiteren Auswertung wird die Vorhersagekraft anhand berechneter Attribute bewertet. Dazu wird eine 20-fold leave one object out Kreuzvalidierung auf bewerteten und berechneten Attributen durchgeführt. Die Ergebnisse werden in Tabelle 4.6 dargestellt. Eine Vorhersage ist mit allen Methoden auf den berechneten Attributen erfolgreicher. Die beste Vorhersage ist somit durch Cimpoi realisierbar. Farhadi weist ein ähnlich gutes Ergebnis auf während die händischen Methoden etwas schlechter abschneiden.

Wird das System auf realen Kameradaten angewendet werden bestimmte Objekte wie z.B. Holz gut erkannt. Insgesamt fällt das Ergebnis allerdings schlechter aus als auf der Datenbank. Dies ist auf die unterschiedlichen Auflösungen der Bilddaten beim Training und der Klassifizierung zurückzuführen. Ebenfalls kann durch eine unterschiedliche Beleuchtung der Szene Abweichungen zwischen den Trainierten und den zu Klassifizerenden Daten entstehen was wiederum zur Verminderung des Klassifizie-

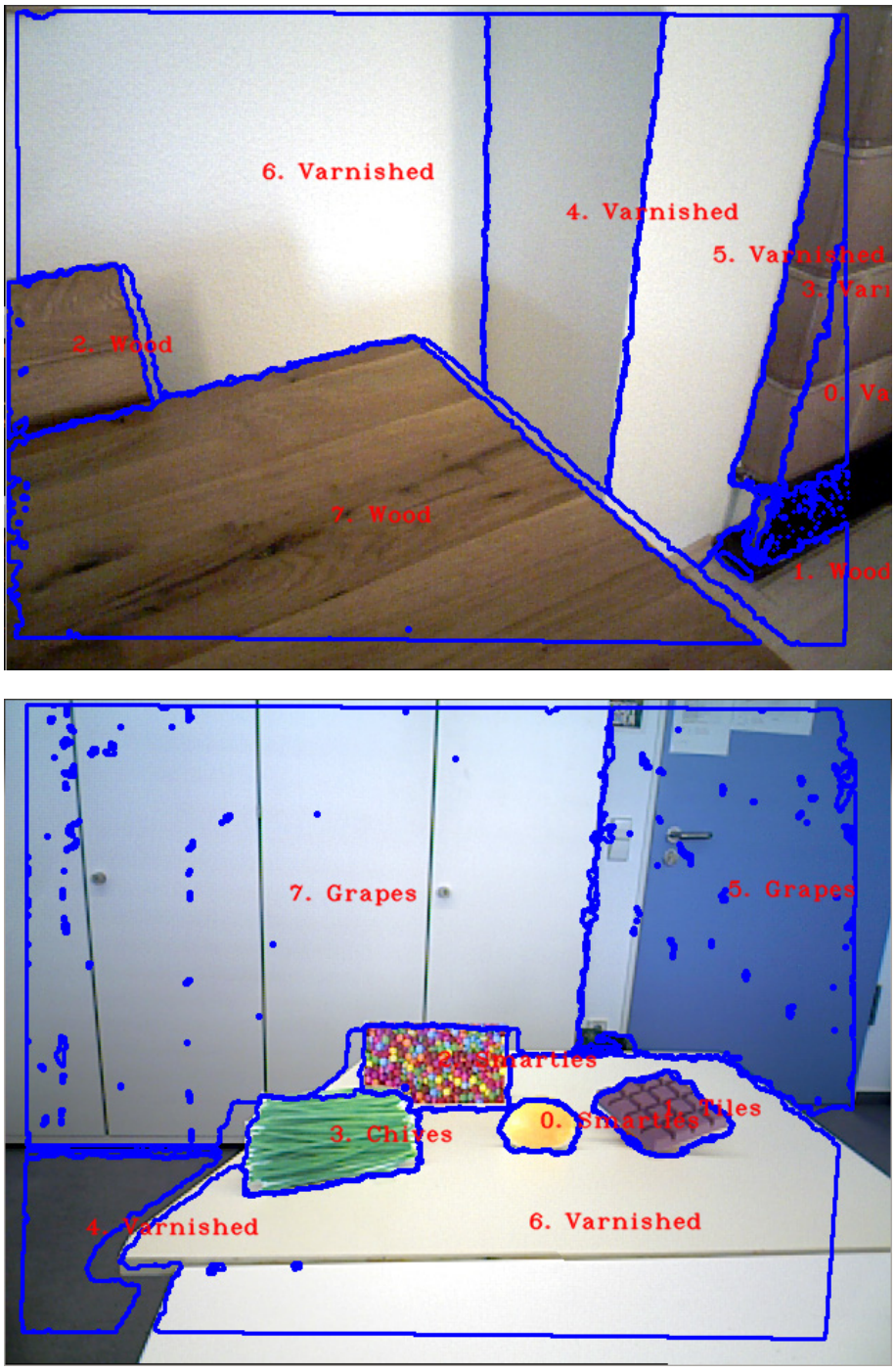


Abbildung 4.11: Visualisierung der Ergebnisse des Systems.

rungsrate führt. Zwei Beispielaufnahmen werden in Abbildung 4.11 dargestellt.

4.4 Automatische Attributwerterzeugung

Das Ziel der automatischen Attributwerterzeugung ist, das bereits durch wenige Beschreibungen einer Textur genügend Trainingsdaten erzeugt werden können, um einen Klassifikator zu trainieren, der die beschriebene Textur klassifizieren kann.

Für die Evaluierung werden aus jeder Klasse der Texturdatenbank A.3 zwei Objekte gewählt und entsprechend Kapitel 3.8 bewertet. Diese Daten werden genutzt, um Trainingsdaten im Umfang der Datenbank zu erzeugen. Um die Ähnlichkeit der beiden Datensätze zu bestimmen, werden die absoluten Differenzen zwischen den Attributwerten der bewerteten Trainingsdaten und den erzeugten Trainingsdaten für jede Klasse gebildet. Die Mittelwerte der absoluten Differenzen für jedes Attribut über alle Klassen ist in Tabelle 4.7 dargestellt. Ausgenommen der dominanten Farben bewegen sich alle Differenzen zwischen 0.32 und 0.54 was einer gewünschten, zufälligen Varianz entspricht. Diese Varianz ist ebenfalls in den händisch erhobenen Daten vorhanden. Das in der Abbildung 4.12 dargestellte Diagramm, zeigt wie häufig sich die Attributwerte der Attribute (1-17 entspricht der Nummerierung in Tabelle 4.7) wenig (blaue Balken), leicht (rote Balken) und stark (grüne Balken) unterscheiden. Die größeren Differenzen lassen sich auf Attribute zurückführen, die für bestimmte Texturen nicht aussagekräftig sind und somit stark variieren können. Die Differenzen für die dominanten Farben und der Verlauf im Diagramm an Position 2 und 3 lässt auf eine schlechte Farbvorhersage im Vergleich zu den restlichen Attributen schließen. Die schlechten Ergebnisse sind auf die Skalierung der Farbtonskala auf einen diskreten Wertebereich zurückzuführen. Dieser berücksichtigt nicht, dass leichte subjektive Farbunterschiede bereits zu hohen Unterschieden im Wertebereich führen können und somit einen starken Einfluss auf die Auswertung haben.

Der Vergleich der Trainingsdatensätze lässt erkennen wie ähnlich die einzelnen Attributwerte sind. Wie gut Texturen durch maschinelle Lernverfahren, die mit den erzeugten Daten trainiert wurden, klassifiziert werden können muss wie in Kapitel 4.3 festgestellt werden.

Tabelle 4.7: Absoluter mittlerer Fehler zwischen den händisch bewerteten Attributwerten und den erzeugten Attributwerten.

Attribut	**mittlere absolute Differenz**
colorfulness (1)	0.48
dominant color (2)	1.29
sec. dominant color (3)	0.80
saturation (4)	0.51
variety of saturation (5)	0.54
value (6)	0.45
variety of value (7)	0.40
average primitive size (8)	0.45
number of primitives (9)	0.38
strength of primitives (10)	0.51
regularity of primitives (11)	0.53
contrast (12)	0.50
line-likeness (13)	0.45
3D roughness (14)	0.38
directionality (15)	0.52
lined (16)	0.38
checked (17)	0.32

Dazu werden die automatisch erzeugten Attribute sowie die durch nach [Farhadi et al., 2009], [Cimpoi et al., 2014] und der händisch gelernten Methode erzeugten Attributwerte über eine 57-fold leave-one-out cross-validation ausgewertet und die Ergebnisse in Tabelle 4.8 dargestellt.

Das Beste Ergebnis erreicht Cimpoi während die automatisch erzeugten Attributwerte ähnlich gut abschneiden. Farhadi und Lampert sind im Vergleich deutlich schlechter als die beiden anderen Verfahren. Absolut kann allerdings kein Verfahren gute Klassifizierungsergebnisse aufweisen. Die schlechte Klassifizierung ist wie bereits in Kapitel 4.3 beschrieben auf das schlechte Verhältnis zwischen Anzahl der Klassen zur Anzahl der Attribute, und der daraus schlechten Trennbarkeit der Klassen, zurückzuführen.

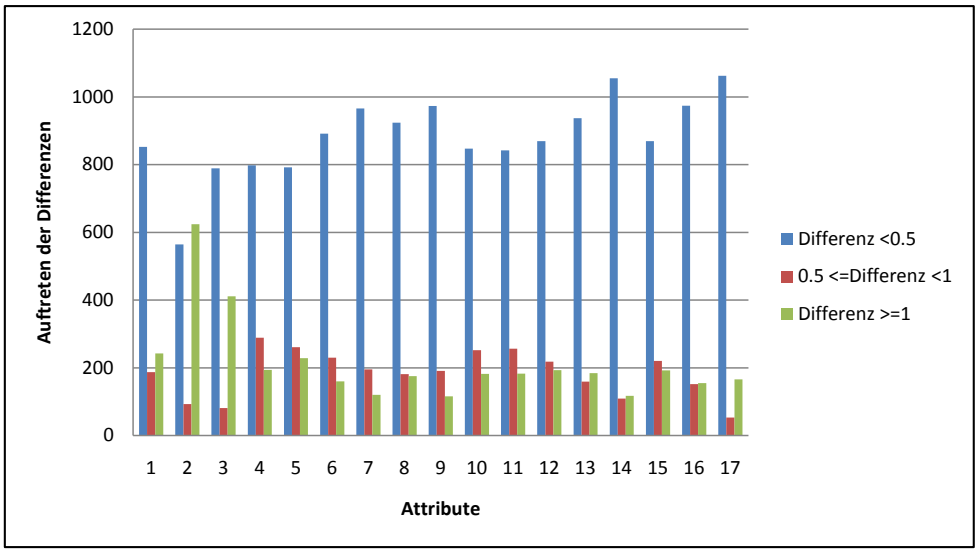


Abbildung 4.12: Häufigkeiten der auftretenden geringen, mittleren und hohen Differenzen Zwischen der Texturdatenbank A.3 und einem Zufällig erzeugten Datensatz.

Tabelle 4.8: Klassifizierungsergebnisse der erzeugten Attribute.

Klassifizierungen	**Ergebnis**
Automatische Attributwerterzeugung	8.19672%
Farhadi	5.30835%
Cimpoi	8.43091%
händisch gelernt	4.60578%

Außerdem wird hier die ungenaue Farbvorhersage durch die gewollte Varianz negativ verstärkt. Der hier verwendetet Klassifikator ist zudem nicht in de Lage, Schlussfolgerungen zu Treffen wie, “Zitronen sind nicht blau oder rot“ was zu einer Verbesserung der Klassifizierung führen würde.

Kapitel 5

Zusammenfassung und Ausblick

Das Ziel dieser Arbeit war, ein System zu entwickeln, dass eine Klassifizierung von Objekten, anhand von neuartigen, menschenverständlichen Texturattributen, vornimmt. Durch den entwickelten Algorithmus sollen die Möglichkeiten der menschenverständlichen Attribute aufgezeigt werden können. Das entstandene System ist in der Lage, eine aufgenommene Szene in homogene Bereiche zu segmentieren, deren Attribute zu bestimmen und eine Klassifizierung vorzunehmen. Die Segmentierung nutzt Tiefendaten, um große Flächen zu erkennen und unterteilt diese in Bereiche, die in ihren Texturattributen gleich sind. Durch Normalisierung der Ansicht der Segmente werden gleichbleibende Bedingungen für die Berechnung der Attribute geschaffen. Die auf einer Texturdatenbank ausgewerteten Attributwerte werden zum Training von maschinellen Lernverfahren genutzt, um weitere Texturen klassifizieren zu können.

Das gewählte zweistufige Segmentierungsverfahren nutzt alle Daten, die dem System zur Verfügung stehen und erfüllt die Anforderung, Segmente zu erzeugen die nach den menschenverständlichen Texturattributen homogen sind. Die Segmente weisen eine hohe Homogenität auf, was optimal für die nachfolgenden Verarbeitungsschritte ist. An den Segmenträndern können allerdings kleine unsegmentierte Bereiche entstehen, die aufgrund des Prinzips des gewählten Split and Merge Verfahrens auftreten. Während die Auflösung des Kamerasystems für die Segmentierung ausreichend ist,

erweist sich die Attributberechnung auf niedrig aufgelösten Texturen als schwierig. Feine Details, die charakteristisch für viele Texturen sind, können oft nur schlecht ausgewertet werden. Die Klassifizierung von Kameradaten erreicht dadurch, im Gegensatz zu hochauflösenden Bilddaten aus einer Datenbank, nur mäßige Ergebnisse. Für die Erzeugung von Trainingsdaten aus wenigen händischen Eingaben wurden experimentell Datensätze erzeugt, die von Menschen erstellte Datensätzen sehr ähnlich sind. Bei einem Versuch zur Klassifizierung schnitten die erstellten Daten schlechter ab als echte Trainingsdaten wodurch Rückschlüsse auf Attribute möglich waren, deren Formulierung bei der Erzeugung von Datensätzen angepasst werden sollte.

Für die Optimierung des Systems kann die Segmentierung um eine Pixelbetrachtung an den Segmentgrenzen erweitert werden, so dass unsichere Bereiche erfolgreich einem entsprechendem Segment zugeordnet werden können. Eine Verbesserung der Attributberechung auf den Kameradaten kann durch die Anpassung der Attribute auf gröbere Texturen erzielt werden. Bessere Klassifizierungsergebnisse können erreicht werden, wenn Trainings- und Testdaten über eine annähernd einheitliche Auflösung verfügen. Dies kann durch Downsampling oder durch die Reduzierung der Trainingsdaten auf kleinere Bildausschnitte realisiert werden. Um eine Klassifizierung von vielen Klassen zu ermögliche, müssen mehr Attribute eingeführt werden. Diese ermöglichen es spezielle Texturen wie z.B. Spiralen zu repräsentiert und erhöhen die Dimensionalität des Merkmalsraums wodurch unterschiedliche Klassen besser getrennt werden können. Die Nutzung eines Klassifizierers der Schlussfolgerungen treffen kann, führt ebenfalls zu einer Verbesserung der Klassifizierung. Denn oft kann bereits anhand weniger Attribute die Menge an in frage kommender Klassen stark reduziert werden.

Anhang A

Ergänzende Unterlagen

A.1 Evaluierung Split

A.2 Auswertung Segmentierung

A.3 Texturdatenbank

A.4 Klassifizierung Attributwerte

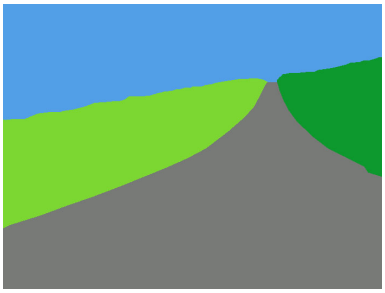
Tabelle A.1: Auswertung der Splitkriterien auf Bild 1.

Kombination	Seg. 1	Seg. 2	Seg. 3	Seg. 4	Mittel
Precision	0,98277823	0,98354892	0,98383172	0,97355096	0,98092746
Recall	0,9926132	0,90552095	0,97052899	0,99592904	0,96614804
F-Maß	0,98767123	0,94292346	0,97713508	0,98461287	0,97308566
LBP	Seg. 1	Seg. 2	Seg. 3	Seg. 4	Mittel.
Precision	0,99564293	0,87673651	0,97832368	0,96940099	0,95502603
Recall	0,99586035	0,94045438	0,9577389	0,97724565	0,96782482
F-Maß	0,99575163	0,90747834	0,96792186	0,97330751	0,96111484
Farbe	Seg. 1	Seg. 2	Seg. 3	Seg. 4	Mittel.
Precision	0,98687877	0,97440957	0,98930729	0,97701439	0,98190251
Recall	0,9919011	0,93483572	0,94209277	0,9942659	0,96577387
F-score	0,98938356	0,95421251	0,96512294	0,98556466	0,97357092

Tabelle A.2: Auswertung der Splitkriterien auf Bild 2.

Kombination	Seg. 1	Seg. 2	Seg. 3	Seg. 4	Seg. 5	Seg. 6	Seg. 7	Seg. 8
Precision	0,94492	0,97705	0,97563	0,86578	0,97612	0,97833	0,89939	0,98821
Recall	0,74868	0,84615	0,90909	0,97657	0,98420	0,99877	0,81859	0,80367
F-Maß	0,83543	0,90690	0,94118	0,91784	0,98014	0,98844	0,85709	0,88644
	Seg. 9	Seg. 10	Seg. 11	Seg. 12	Seg. 13	Seg. 14	Seg. 15	Mittel.
Precision	0,79033848	0,96233357	0,99026217	0,92548759	0,89293542	0,96701088	0,97498427	0,94059
Recall	0,67651195	0,81838498	0,88231368	0,96785593	0,84197138	0,94172646	0,92705841	0,87610
F-Maß	0,72900879	0,88454106	0,93317647	0,94619771	0,86670485	0,95420121	0,95041755	0,90518
Farbe	Seg. 1	Seg. 2	Seg. 3	Seg. 4	Seg. 5	Seg. 6	Seg. 7	Seg. 8
Precision	0,94492	0,97705	0,97563	0,86578	0,97612	0,97833	0,89939	0,97280
Recall	0,74868	0,84615	0,90909	0,97657	0,98420	0,99877	0,81859	0,93914
F-Maß	0,83543	0,90690	0,94118	0,91784	0,98014	0,98844	0,85709	0,95567
	Seg. 9	Seg. 10	Seg. 11	Seg. 12	Seg. 13	Seg. 14	Seg. 15	Mittel.
Precision	0,79033848	0,96233357	0,99026217	0,92635467	0,89293542	0,96701088	0,97498427	0,93962
Recall	0,67651195	0,81838498	0,88231368	0,9671782	0,84197138	0,94172646	0,92705841	0,88509
F	0,72900879	0,88454106	0,93317647	0,94632637	0,86670485	0,95420121	0,95041755	0,90981
LBP	Seg. 1	Seg. 2	Seg. 3	Seg. 4	Seg. 5	Seg. 6	Seg. 7	Seg. 8
Precision	0,00000	0,97705	0,99141	0,83178	0,95823	0,80537	0,78447	0,00000
Recall	0,00000	0,84615	0,89166	0,96345	0,96605	0,86724	0,52874	0,00000
F-Maß	0,00000	0,90690	0,93889	0,89279	0,96212	0,83516	0,63171	0,00000
	Seg. 9	Seg. 10	Seg. 11	Seg. 12	Seg. 13	Seg. 14	Seg. 15	Mittel.
Precision	0,74498653	0,87518294	0,84743326	0,87273634	0,83899233	0,96241571	0,95817346	0,76322
Recall	0,70014065	0,62365912	0,91813126	0,95964241	0,85246423	0,89225919	0,88747361	0,72647
F-Maß	0,72186775	0,72831666	0,88136679	0,91412849	0,84567463	0,92601055	0,92146941	0,74043

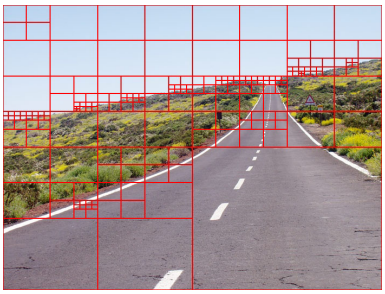
(a) Originalbild



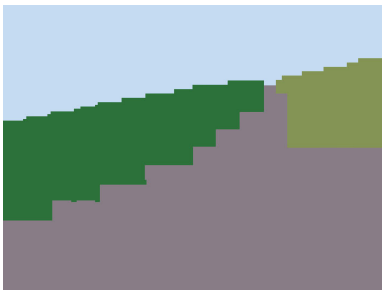
(b) Homogenen Regionen

(c) Kombination LBP-Farbe

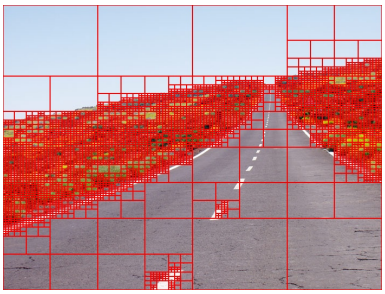
(d) Kombination LBP-Farbe



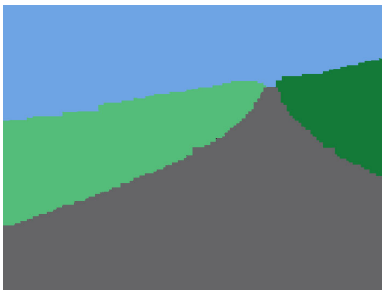
(e) LBP-Kriterium



(f) LBP-Kriterium

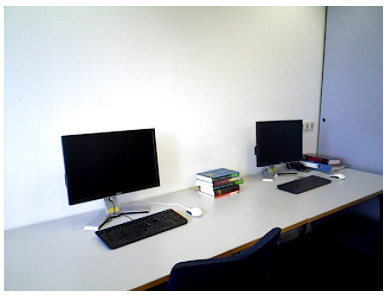


(g) Farbkriterium

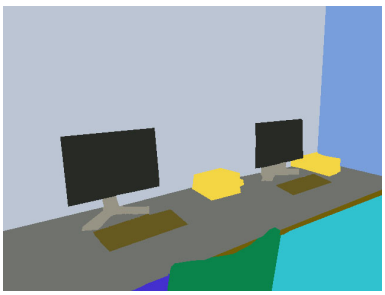


(h) Farbkriterium

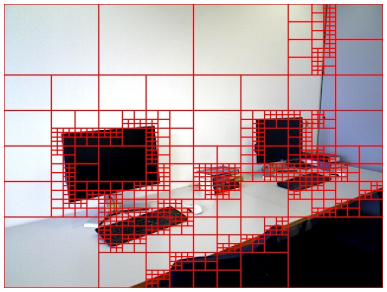
Abbildung A.1: Ergebnisse Splitvorgang auf Bild 1. Links: Erhaltene Regionen. Rechts: Regionen zugeordnet zu homogenen Bereichen.



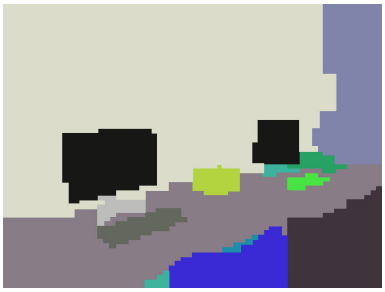
(a) Originalbild



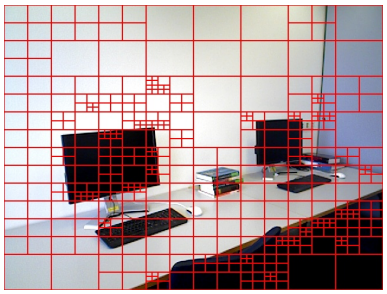
(b) Homogene Regionen



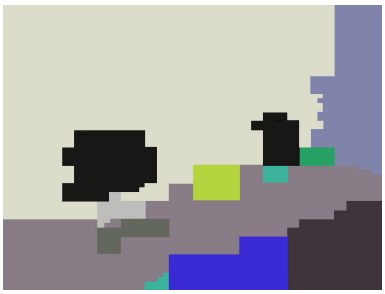
(c) Kombination LBP-Farbe



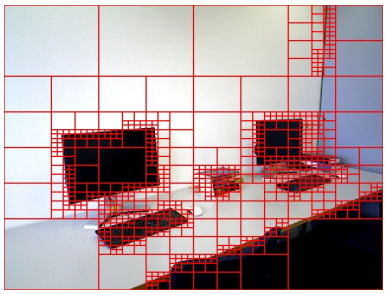
(d) Kombination LBP-Farbe



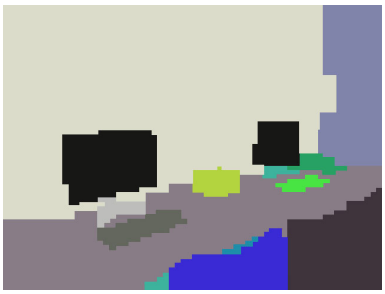
(e) LBP-Kriterium



(f) LBP-Kriterium



(g) Farbkriterium



(h) Farbkriterium

Abbildung A.2: Ergebnisse Splitvorgang auf Bild 2. Links: Erhaltene Regionen. Rechts: Regionen zugeordnet zu homogenen Bereichen.

Tabelle A.3: Auswertung der Segmentierungsergebnisse Bild 1-5.

Bild1		Seg. 1	Seg. 2	Seg. 3	Seg. 4	Seg. 5	Seg. 6	Seg. 7	Seg. 8	Seg. 9	Seg. 10	Seg. 11	Mittel.
Farbe	Precision	0,2023	0,5081	0,9973	0,9644	0,9526	0,9658	0,9402	0,9401	0,9569	0,8440	0,8633	0,8304
	Recall	0,1601	0,0656	0,8040	0,9633	0,9685	0,9296	0,8811	0,6401	0,8982	0,9444	0,9480	0,7457
	F-Maß	0,1787	0,1162	0,8903	0,9638	0,9605	0,9473	0,9097	0,7616	0,9266	0,8914	0,9037	0,7682
Komb.	Precision	0,8604	0,5081	0,9882	0,9644	0,9526	0,9658	0,9402	0,9618	0,9569	0,8632	0,8633	0,8932
	Recall	0,4176	0,0656	0,8767	0,9633	0,9685	0,9296	0,8811	0,6519	0,8982	0,9437	0,9480	0,7767
	F-Maß	0,5623	0,1162	0,9291	0,9638	0,9605	0,9473	0,9097	0,7771	0,9266	0,9017	0,9037	0,8089
Tiefe	Precision	0,9612	0,4542	0,9960	0,9667	0,9729	0,9777	0,9806	0	0,9796	0	0,9303	0,7472
	Recall	0,3125	0,9110	0,8653	0,9026	0,9605	0,9370	0,8878	0	0,8711	0	0,8958	0,6858
	F-Maß	0,4716	0,6062	0,9261	0,9336	0,9667	0,9569	0,9319	0	0,9222	0	0,9127	0,6934

Bild2		Seg. 1	Seg. 2	Seg. 3	Seg. 4	Seg. 5	Seg. 6	Seg. 7	Seg. 8	Seg. 9	Seg. 10	Seg. 11	Seg. 12	Mittel.
Farbe	Precision	0	0,9903	0,4781	0,9960	0,9933	0,9650	0,4679	0,9165	0,9910	0,8997	0,9715	0,9469	0,8014
	Recall	0	0,9268	0,9924	0,8420	0,8251	0,4523	0,9810	0,9209	0,9207	0,8474	0,9130	0,8690	0,7909
	F-Maß	0	0,9575	0,6453	0,9126	0,9014	0,6159	0,6336	0,9187	0,9545	0,8727	0,9413	0,9063	0,7717
Komb.	Precision	0	0,9917	0,4648	0,9932	0,9878	0,9500	0,4556	0,9137	0,9496	0,9069	0,9719	0,9414	0,7939
	Recall	0	0,9289	0,9932	0,8502	0,8709	0,4728	0,9847	0,9389	0,9848	0,8540	0,9210	0,8781	0,8065
	F-Maß	0	0,9593	0,6332	0,9162	0,9256	0,6314	0,6230	0,9261	0,9669	0,8796	0,9458	0,9086	0,7763
Tiefe	Precision	0,9775	0,9899	0,9719	1,0000	0,9948	0,0947	0,9928	0,9995	0,9851	0,9269	0,9818	0,9883	0,9086
	Recall	0,2994	0,8474	0,4221	0,1492	0,8490	0,9992	0,6696	0,4431	0,2895	0,7987	0,8387	0,6686	0,6062
	F-Maß	0,4584	0,9131	0,5885	0,2596	0,9161	0,1730	0,7998	0,6140	0,4474	0,8580	0,9046	0,7976	0,6442

Bild3		Seg. 1	Seg. 2	Seg. 3	Seg. 4	Seg. 5	Seg. 6	Seg. 7	Seg. 8	Seg. 9	Seg. 10	Seg. 11	Seg. 12	Mittel.
Farbe	Precision	0	0,9259	0,9312	0,9656	0,8789	0,8986	0,9746	0,9427	0,9418	0,9896	0,9273	0,9878	0,8637
	Recall	0	0,3054	0,8260	0,3520	0,5058	0,9258	0,9709	0,6144	0,9293	0,8418	0,6630	0,7530	0,6406
	F-Maß	0	0,4593	0,8754	0,5159	0,6421	0,9120	0,9728	0,7440	0,9355	0,9097	0,7732	0,8546	0,7162
Komb.	Precision	0	0,9671	0,9312	0,9246	0,8795	0,8986	0,9746	0,9432	0,9418	0,9896	0,9273	0,9545	0,8610
	Recall	0	0,4105	0,8260	0,3191	0,5181	0,9258	0,9709	0,7416	0,9293	0,8418	0,6630	0,8463	0,6660
	F-Maß	0	0,5764	0,8754	0,4744	0,6520	0,9120	0,9728	0,8303	0,9355	0,9097	0,7732	0,8972	0,7341
Tiefe	Precision	0	0,9856	0,6526	0	0,9688	0,9498	0,9919	0,9234	0,9860	0,9921	0,9742	0,9632	0,7823
	Recall	0	0,4038	0,9389	0	0,3412	0,8964	0,9345	0,2688	0,9230	0,8176	0,6692	0,8577	0,5876
	F-Maß	0	0,5728	0,7700	0	0,5047	0,9224	0,9624	0,4164	0,9534	0,8964	0,7934	0,9074	0,6416

Bild4		Seg. 1	Seg. 2	Seg. 3	Seg. 4	Seg. 5	Seg. 6	Seg. 7	Seg. 8	Seg. 9	Seg. 10	Seg. 11	Seg. 12	Seg. 13	Seg. 14	Mittel.
Farbe	Precision	0,0419	0,9319	0,9494	0,9853	0,7269	0,7127	0,9868	0	0,8945	0,9906	0	0,9912	0,9931	0,9902	0,7282
	Recall	0,0034	0,3674	0,8750	0,8736	0,9550	0,1695	0,8001	0	0,9901	0,9289	0	0,9423	0,7792	0,5451	0,5878
	F-Maß	0,0063	0,5270	0,9107	0,9261	0,8255	0,2739	0,8837	0	0,9399	0,9588	0	0,9662	0,8733	0,7031	0,6282
Komb.	Precision	0,4385	0,9319	0,9267	0,9846	0,6132	0,8973	0,9756	0,9009	0,8813	0,9702	0	0,9760	0,9926	0,4951	0,7846
	Recall	0,0211	0,3674	0,9633	0,8809	0,9757	0,6250	0,8845	0,3477	0,9915	0,9572	0	0,9669	0,7791	0,9381	0,6928
	F-Maß	0,0403	0,5270	0,9447	0,9299	0,7531	0,7368	0,9278	0,5018	0,9332	0,9637	0	0,9714	0,8730	0,6481	0,6965
Tiefe	Precision	0,4683	0,9713	0,9255	1,0000	0,9736	0	0,9915	0,9666	0,9769	0,9995	0	0,9620	0,9758	0,9403	0,7965
	Recall	0,9097	0,4364	0,9028	0,3350	0,2202	0	0,8659	0,7544	0,3709	0,3815	0	0,8469	0,2944	0,2826	0,4715
	F-Maß	0,6183	0,6023	0,9140	0,5019	0,3592	0	0,9244	0,8474	0,5377	0,5522	0	0,9008	0,4523	0,4346	0,5461

Bild5		Seg. 1	Seg. 2	Seg. 3	Seg. 4	Seg. 5	Seg. 6	Seg. 7	Seg. 8	Seg. 9	Seg. 10	Seg. 11	Seg. 12	Seg. 13	Seg. 14	Seg. 15	Mittel.
Farbe	Precision	0	0	0,9644	0,9478	0,9816	0,8877	0,9912	0,9682	0,8977	0,9576	0,9939	0,9902	0,4595	0,8249	0,9066	0,7848
	Recall	0	0	0,9370	0,9386	0,9227	0,8512	0,8285	0,8770	0,9024	0,9529	0,8573	0,8384	0,5964	0,8978	0,9898	0,7593
	F-Maß	0	0	0,9505	0,9432	0,9512	0,8691	0,9026	0,9204	0,9000	0,9552	0,9206	0,9080	0,5190	0,8598	0,9464	0,7697
Komb.	Precision	0	0	0,9355	0,9478	0,9798	0,8877	0,9697	0,9662	0,8977	0,9576	0,9924	0,9837	0,9276	0,7362	0,8893	0,8047
	Recall	0	0	0,9128	0,9386	0,9282	0,8512	0,8783	0,8746	0,9024	0,9529	0,8608	0,8752	0,2382	0,9187	0,9943	0,7417
	F-Maß	0	0	0,9240	0,9432	0,9533	0,8691	0,9217	0,9181	0,9000	0,9552	0,9219	0,9263	0,3791	0,8174	0,9388	0,7579
Tiefe	Precision	1,0000	0,1927	0,9822	0,9734	0,9973	0,9422	0,9909	0,9829	0,9679	0,9980	0,9954	0,9947	0,9878	0	0,9242	0,8620
	Recall	0,2362	0,9967	0,8165	0,7791	0,2466	0,7223	0,6944	0,7270	0,7807	0,7978	0,1880	0,8677	0,1558	0	0,8718	0,5920
	F-Maß	0,3822	0,3229	0,8917	0,8655	0,3954	0,8177	0,8166	0,8358	0,8643	0,8867	0,3162	0,9269	0,2691	0	0,8972	0,6326

Tabelle A.4: Auswertung der Segmentierungsergebnisse Bild 6 und Bild 7.

Bild6		Seg. 1	Seg. 2	Seg. 3	Seg. 4	Seg. 5	Seg. 6	Seg. 7	Seg. 8	Seg. 9	Seg. 10	Seg. 11	Seg. 12	Seg. 13	Seg. 14	Seg. 15	Seg. 16	Mittel.
Farbe	Precision	0,7102	0,7313	0	1,0000	0	0,8140	0,9559	0,9368	0,9697	0,9740	0,9854	0,8781	0,9921	0,7673	0,3203	0	0,6897
	Recall	0,8777	0,5312	0	0,2680	0	0,8687	0,8382	0,3792	0,6523	0,8930	0,7487	0,5635	0,8479	0,8573	0,1146	0	0,5275
	F-Maß	0,7851	0,6154	0	0,4227	0	0,8405	0,8932	0,5399	0,7799	0,9318	0,8509	0,6865	0,9144	0,8098	0,1688	0	0,5774
Komb.	Precision	0,7102	0,7313	0	1,0000	0	0,8140	0,9430	0,9368	0,9697	0,9734	0,9854	0,8781	0,9921	0,7689	0,3203	0	0,6890
	Recall	0,8777	0,5312	0	0,3627	0	0,8687	0,8271	0,3792	0,6523	0,8938	0,7487	0,5635	0,8479	0,8475	0,1146	0	0,5322
	F-Maß	0,7851	0,6154	0	0,5323	0	0,8405	0,8812	0,5399	0,7799	0,9319	0,8509	0,6865	0,9144	0,8063	0,1688	0	0,5833
Tiefe	Precision	0,7665	0,8697	0	1,0000	0	0,8097	0,9550	0,9611	0,9861	0,9663	0,9922	0,9786	0,9978	0,7900	1,0000	0	0,7546
	Recall	0,8769	0,3135	0	0,2816	0	0,9304	0,8036	0,3461	0,5431	0,7882	0,6862	0,2372	0,8479	0,7924	0,2559	0	0,4814
	F-Maß	0,8180	0,4608	0	0,4394	0	0,8659	0,8728	0,5089	0,7004	0,8682	0,8113	0,3819	0,9168	0,7912	0,4075	0	0,5527

Bild7		Seg. 1	Seg. 2	Seg. 3	Seg. 4	Seg. 5	Seg. 6	Seg. 7	Seg. 8	Seg. 9	Seg. 10	Seg. 11	Seg. 12	Seg. 13	Seg. 14	Seg. 15	Seg. 16	Seg. 17	Seg. 18	Seg. 19	Mittel.
Farbe	Precision	0,9650	0,7579	0,8495	0	0,8496	0,9570	0,9965	0,9652	0,7290	0,9733	0,4782	0,9435	0	0,9867	0,9700	0,9002	0,9889	0,9773	0,8391	0,7961
	Recall	0,6908	0,9213	0,7395	0	0,8304	0,7441	0,8156	0,9638	0,9756	0,8285	0,3568	0,8807	0	0,6995	0,3767	0,8750	0,8769	0,9262	0,9110	0,7059
	F-Maß	0,8052	0,8317	0,7907	0	0,8399	0,8372	0,8970	0,9645	0,8344	0,8951	0,4086	0,9110	0	0,8187	0,5426	0,8874	0,9295	0,9511	0,8735	0,7378
Komb.	Precision	0	0,7579	0,8495	0	0,8797	0,8174	0,9965	0,9652	0,7290	0,9525	0	0,8948	0	0,9694	0,8859	0,7102	0,9889	0,9773	0,8391	0,6954
	Recall	0	0,9213	0,7395	0	0,8144	0,2093	0,8156	0,9638	0,9756	0,8131	0	0,9682	0	0,7525	0,3680	0,8927	0,8769	0,9262	0,9110	0,6288
	F-Maß	0	0,8317	0,7907	0	0,8458	0,3333	0,8970	0,9645	0,8344	0,8773	0	0,9301	0	0,8473	0,5200	0,7910	0,9295	0,9511	0,8735	0,6430
Tiefe	Precision	0	0,8226	0,9705	0	0	0	0,9991	0,9944	0,9502	0,9682	0	0,9126	0,2314	0,9846	0,9760	0,9390	0,9917	0,9867	0,8864	0,6639
	Recall	0	0,5066	0,6992	0	0	0	0,7648	0,9387	0,9611	0,4365	0	0,9610	0,6688	0,7465	0,3639	0,8351	0,9648	0,8727	0,8933	0,5586
	F-Maß	0	0,6270	0,8128	0	0	0	0,8664	0,9658	0,9557	0,6017	0	0,9361	0,3439	0,8492	0,5302	0,8840	0,9781	0,9262	0,8898	0,5877

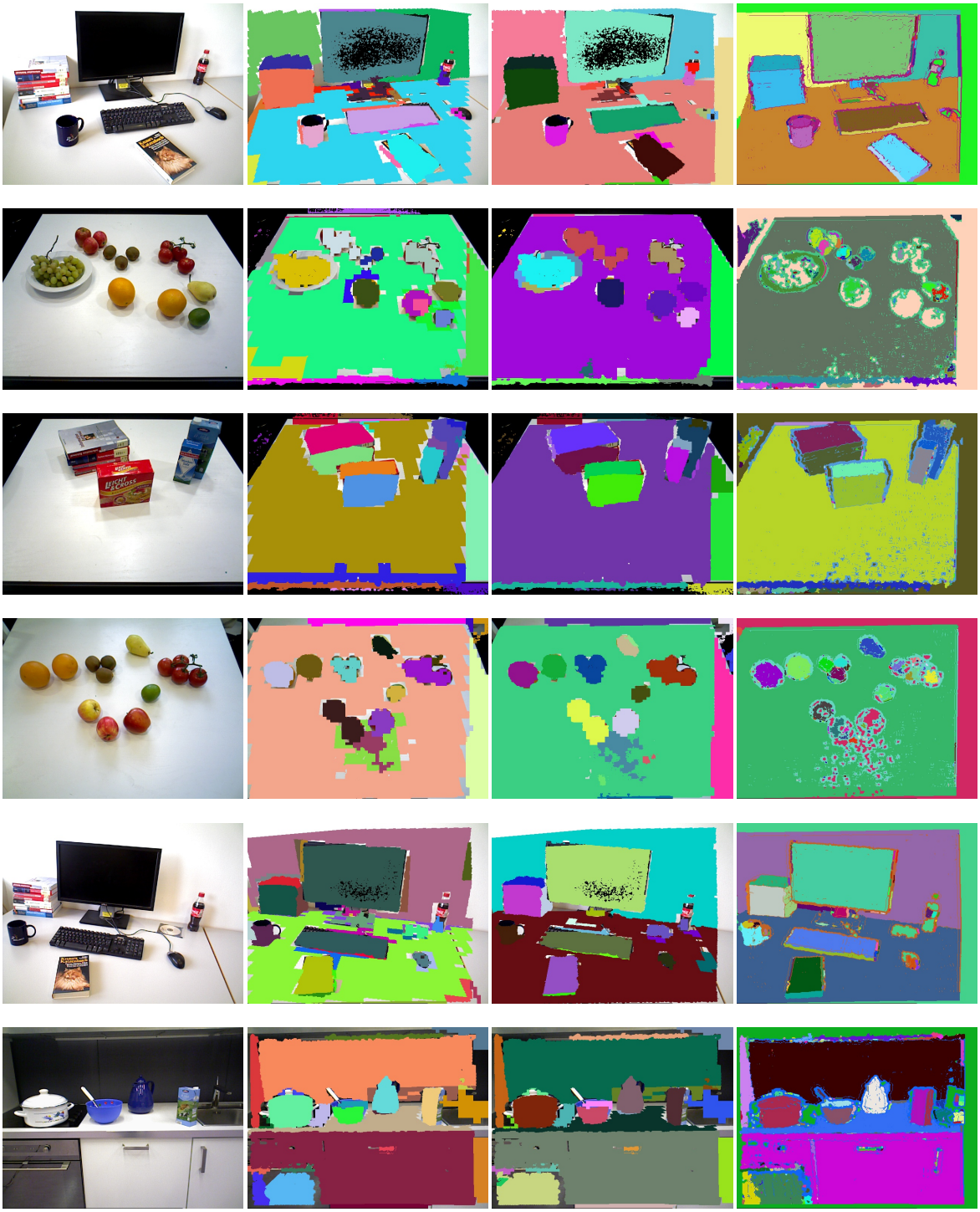


Abbildung A.3: Ergebnisse der Segmentierung. Von links: Originalbild, Tiefen- und Farbsegmentierung, Tiefen- und Textursegmentierung, Tiefensegmentierung.

Tabelle A.4: Klassen der Texturdatenbank mit Anzahl der Texturen sowie deren Einordnung in Objekte.

	Anzahl Texturen	Anzahl Objekte		Anzahl Texturen	Anzahl Objekte		Anzahl Texturen	Anzahl Objekte
Aluminum foil	48	10	Flakes	28	8	Parsley	7	7
Asphalt	28	10	Flour	17	7	Pasta	23	6
Book shelf	17	11	Foam	17	5	Paving stone	20	7
Bread	44	6	Football	7	2	PC keyboard	43	9
Brick	16	7	Fork	16	8	Pineapple	13	7
Broccoli	36	10	Fur	30	12	Plate	33	12
Carpet	29	3	Granite floor	17	9	Rice	27	9
Cauliflower	9	8	Grapes	22	6	Sand	11	3
CD	17	10	Ingrain wallpaper	23	10	Smarties	25	11
Chocolate bar	32	8	Jalousie	21	7	Sponge	28	11
Clock	17	10	Kiwi	17	11	Spoon	17	8
Coconut	16	2	Knife	18	8	Styrofoam	25	8
Coffee beans	15	1	Leather	30	12	Telephone	26	9
Concrete	17	6	Lemon	13	11	Textured wallpaper	31	11
Corduroy	18	4	Lime	12	9	Tiled floor	39	13
Cork floor	23	7	Linen	17	6	Tomato	17	16
Cotton/Wool	26	10	Marble	20	9	Varnished homogen.	33	10
Cracker	11	6	(Computer) Mouse	18	5	Washing machine	18	8
Cup	38	10	Orange	10	9	Wooden texture	34	13

Tabelle A.5: 57-fold leave out one class cross-validation

	Händisch mean abs error	*Händisch und* lernend mean abs error	*Farhadi* mean abs error	*Cimpoi* mean abs error
Attribut 1:	0,638165	0,756687	0,698583	0,833766
Attribut 2:	0,927686	0,813027	0,826324	0,969795
Attribut 3:	0,764898	0,696053	0,723918	0,768060
Attribut 4:	0,489813	0,567467	0,521272	0,605544
Attribut 5:	0,662986	0,766166	0,701336	0,821697
Attribut 6:	0,472430	0,556218	0,562320	0,768330
Attribut 7:	0,426308	0,456097	0,470736	0,530168
Attribut 8:	0,814734	0,864653	0,810345	0,814617
Attribut 9:	0,813942	0,874635	0,770154	0,803739
Attribut 10:	0,685184	0,853968	0,717670	0,676821
Attribut 11:	0,805720	0,977025	0,839264	0,792650
Attribut 12:	0,398693	0,527006	0,521669	0,521143
Attribut 13:	0,722782	0,793073	0,755610	0,694601
Attribut 14:			0,598053	0,677765
Attribut 15:	0,915689	1,112390	1,034030	0,863733
Attribut 16:	0,887516	1,013410	1,085590	0,887162
Attribut 17:	0,743421	0,765630	0,771496	0,629973
	<0.5	<0.5	<0.5	<0.5
Attribut 1:	45,2877%	41,0617%	48,9461%	38,0172%
Attribut 2:	39,7201%	55,6596%	50,8197%	35,2069%
Attribut 3:	57,1384%	61,8267%	57,7674%	57,2209%
Attribut 4:	60,5288%	53,9422%	59,4848%	50,2732%
Attribut 5:	44,1680%	42,2326%	45,2771%	36,4559%
Attribut 6:	62,7683%	54,4887%	54,4887%	37,8610%
Attribut 7:	68,8336%	66,1983%	63,0757%	56,1280%
Attribut 8:	35,8009%	39,7346%	38,4856%	38,9539%
Attribut 9:	37,1384%	36,0656%	40,3591%	38,3294%
Attribut 10:	43,0793%	33,2553%	41,9204%	43,4817%
Attribut 11:	35,2411%	30,4450%	34,6604%	37,5488%
Attribut 12:	68,6470%	55,0351%	53,0835%	55,2693%
Attribut 13:	48,3048%	50,5855%	44,3404%	56,2061%
Attribut 14:			60,7338%	51,0539%
Attribut 15:	32,3173%	26,1514%	28,1811%	35,9094%
Attribut 16:	40,7154%	45,1210%	20,6870%	36,2217%
Attribut 17:	59,0669%	67,1351%	57,0648%	70,7260%
	<1.0	<1.0	<1.0	<1.0
Attribut 1:	80,0622%	74,1608%	77,9079%	69,0086%
Attribut 2:	64,6034%	75,0195%	73,6924%	72,4434%
Attribut 3:	77,4184%	82,5917%	84,6995%	82,8259%
Attribut 4:	87,5583%	84,3091%	85,1678%	81,6550%
Attribut 5:	81,2753%	71,3505%	76,1124%	65,7299%
Attribut 6:	92,3173%	85,2459%	81,3427%	73,7705%
Attribut 7:	94,2768%	92,6620%	93,5207%	88,6807%
Attribut 8:	69,6112%	67,4473%	67,1351%	66,8228%
Attribut 9:	68,7714%	66,4325%	69,7112%	66,8228%
Attribut 10:	75,1166%	62,9196%	71,5066%	74,3169%
Attribut 11:	64,4168%	57,6112%	66,1202%	71,4286%
Attribut 12:	94,4635%	86,2607%	88,6807%	88,6027%
Attribut 13:	75,4277%	76,6589%	76,3466%	77,1272%
Attribut 14:			83,6846%	83,2162%
Attribut 15:	60,1244%	50,1171%	52,0687%	65,4957%
Attribut 16:	66,6563%	64,7151%	55,1913%	69,3989%
Attribut 17:	73,0949%	75,6440%	79,9375%	81,2646%

Tabelle A.6: 20-fold leave out one object cross-validation

	Händisch mean abs error	*Händisch und* lernend mean abs error	*Darhadi* mean abs error	*Cimpoi* mean abs error
Attribut 1:	0,638165	0,562576	0,473034	0,432697
Attribut 2:	0,927686	0,812877	0,728206	0,648843
Attribut 3:	0,764898	0,625930	0,597618	0,603323
Attribut 4:	0,489813	0,404726	0,378477	0,383148
Attribut 5:	0,662986	0,639878	0,544661	0,511013
Attribut 6:	0,472430	0,407787	0,379973	0,358369
Attribut 7:	0,426308	0,335594	0,325031	0,309832
Attribut 8:	0,814734	0,741633	0,633224	0,675029
Attribut 9:	0,813942	0,756136	0,590142	0,633506
Attribut 10:	0,685184	0,640193	0,529364	0,578609
Attribut 11:	0,805720	0,721490	0,607523	0,643452
Attribut 12:	0,398693	0,387336	0,362978	0,350850
Attribut 13:	0,722782	0,623259	0,558363	0,613867
Attribut 14:			0,386019	0,415940
Attribut 15:	0,915689	0,854374	0,792145	0,806662
Attribut 16:	0,887516	0,852615	0,873533	0,883361
Attribut 17:	0,743421	0,563456	0,592386	0,596284
	<0.5	<0.5	<0.5	<0.5
Attribut 1:	45,2877%	54,9922%	64,3235%	65,3499%
Attribut 2:	39,7201%	58,1960%	58,7869%	61,1198%
Attribut 3:	57,1384%	67,4339%	72,7838%	68,3670%
Attribut 4:	60,5288%	70,5443%	71,7885%	70,2644%
Attribut 5:	44,1680%	51,5086%	59,4090%	64,0435%
Attribut 6:	62,7683%	69,8911%	72,4106%	77,3872%
Attribut 7:	68,8336%	79,3468%	81,1820%	82,4261%
Attribut 8:	35,8009%	45,0078%	50,6998%	49,4246%
Attribut 9:	37,1384%	41,7418%	54,3701%	51,4774%
Attribut 10:	43,0793%	46,6563%	59,3779%	54,6190%
Attribut 11:	35,2411%	40,6221%	50,7309%	48,9580%
Attribut 12:	68,6470%	72,0373%	75,1477%	75,3655%
Attribut 13:	48,3048%	61,5863%	65,3188%	60,9953%
Attribut 14:			73,5925%	72,1928%
Attribut 15:	32,3173%	34,7745%	41,2131%	40,9020%
Attribut 16:	40,7154%	42,8616%	35,2411%	43,0793%
Attribut 17:	59,0669%	71,8196%	70,6376%	67,9005%
	<1.0	<1.0	<1.0	<1.0
Attribut 1:	80,0622%	86,1275%	90,9176%	92,3484%
Attribut 2:	64,6034%	72,9705%	75,9565%	79,2224%
Attribut 3:	77,4184%	85,6299%	87,1851%	84,3546%
Attribut 4:	87,5583%	93,2504%	94,1213%	93,8414%
Attribut 5:	81,2753%	79,6890%	83,7947%	83,0793%
Attribut 6:	92,3173%	93,7170%	93,9658%	94,6812%
Attribut 7:	94,2768%	96,8896%	96,0809%	96,3297%
Attribut 8:	69,6112%	74,0280%	79,5645%	75,7387%
Attribut 9:	68,7714%	73,4370%	81,3064%	78,1337%
Attribut 10:	75,1166%	80,0933%	83,9191%	80,9331%
Attribut 11:	64,4168%	72,5350%	82,8305%	79,5956%
Attribut 12:	94,4635%	94,1524%	95,6454%	96,4541%
Attribut 13:	75,4277%	84,1058%	86,3453%	82,6128%
Attribut 14:			93,4681%	89,8911%
Attribut 15:	60,1244%	67,3717%	69,1446%	67,4339%
Attribut 16:	66,6563%	72,6905%	69,5490%	68,2737%
Attribut 17:	73,0949%	82,4883%	85,3188%	84,8212%

Abbildungsverzeichnis

Literaturverzeichnis

[Adams und Bischof, 1994] Adams, Rolf und L. Bischof (1994). *Seeding Region Growing*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 16. Jg.(6):641–647.

[Amadasun und King, 1989] Amadasun, Moses und R. King (1989). *Textural features corresponding to textural properties*. IEEE Transactions onSystems, Man and Cybernetics, 19(5):1264–1274.

[Anderson et al., 1989] Anderson, John Robert, A. Albert, J. Grabowski-Gellert, S. Granzow und U. Fehr (1989). *Kognitive Psychologie: Eine Einführung*. Spektrum der Wissenschaft Verlagsgesellschaft.

[Arbeiter et al., 2014] Arbeiter, Georg, S. Fuchs, J. Hampp und R. Bormann (2014). *Efficient segmentation and surface classification of range images*. In: *IEEE International Conference on Robotics and Automation (ICRA)*, S. 5502–5509.

[Arvis et al., 2011] Arvis, Vincent, C. Debain, M. Berducat und A. Benassi (2011). *Generalization of the cooccurrence matrix for colour images: application to colour texture classification*. Image Analysis & Stereology, 23(1):63–72.

[Baheerathan et al., 1999] Baheerathan, S., F. Albregtsen und H. Danielsen (1999). *New texture features based on the complexity curve*. Pattern Recognition, 32:605–618.

[Ballard, 1981] Ballard, Dana H (1981). *Generalizing the Hough transform to detect arbitrary shapes*. Pattern recognition, 13(2):111–122.

[Bhushan et al., 1997] Bhushan, Nalini, A. R. Rao und G. L. Lohse (1997). *The texture lexicon: Understanding the categorization of visual texture terms and their relationship to texture images*. Cognitive Science, 21(2):219–246.

[Bleiweiss und Werman, 2009] Bleiweiss, Amit und M. Werman (2009). *Fusing Time-of-Flight Depth and Color for Real-Time Segmentation and Tracking*. In: Kolb, Andreas, Hrsg.: *Dynamik 3D Imaging*, S. 306–317. Springer.

[Boser et al., 1992] Boser, Bernhard E, I. M. Guyon und V. N. Vapnik (1992). *A training algorithm for optimal margin classifiers*. In: *Proceedings of the fifth annual workshop on Computational learning theory*, S. 144–152.

[Bradski und Kaehler, 2008] Bradski, Gary und A. Kaehler (2008). *Learning OpenCV: Computer vision with the OpenCV library*. OŔeilly Media, Inc.

[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 (TIST), 2(3):27.

[Cheng, 1995] Cheng, Yizong (1995). *Mean shift, mode seeking, and clustering*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 17(8):790–799.

[Cimpoi et al., 2014] Cimpoi, Mircea, S. Maji, I. Kokkinos, S. Mohamed und A. Vedaldi (2014). *Describing textures in the wild*. arXiv preprint arXiv:1311.3618.

[Coggins, 1983] Coggins, James Michael (1983). *A framework for texture analysis based on spatial filtering*.

[Cohen et al., 1991] Cohen, Fernand S., Z. Fang und M. a. Patel (1991). *Classification of rotated and scaled texture images using Gaussian Markov random field models*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 13:192–202.

[Comaniciu und Meer, 2002] Comaniciu, Dorin und P. Meer (2002). *Mean shift: A robust approach toward feature space analysis*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 24(5):603–619.

[Conners und Harlow, 1980] Conners, R.W. und C. Harlow (1980). *A theoretical comparison of texture algorithms*. IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-2(3):204–222.

[Cortes und Vapnik, 1995] Cortes, Corinna und V. Vapnik (1995). *Support-vector networks*. Machine learning, 20(3):273–297.

[Davies, 2008] Davies, E.R. (2008). *Introduction to texture analysis*. In: Mirmedi, Majid, X. Xie und J. Suri, Hrsg.: *Handbook of Texture Analysis*, Kap. 1, S. 1–31. Imperial Colege Press.

[DeSouza und Kak, 2002] DeSouza, Guilherme N und A. C. Kak (2002). *Vision for mobile robot navigation: A survey*. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 24(2):237–267.

[Eichmann und Kasparis, 1988] Eichmann, G. und T. Kasparis (1988). *Topologically invariant texture descriptors*. Computer Vision, Graphiscs, and Image Processing, 41:267–281.

[Esslinger, 2014] Esslinger, Dominik (2014). *Texture Classification for Service Robotics. Unveröffentlichte Studienarbeit , Universität Stuttgart*.

[Farhadi et al., 2009] Farhadi, Ali, I. Endres, D. Hoiem und D. Forsyth (2009). *Describing objects by their attributes*. In: *IEEE Conference on Computer Vision and Pattern Recognition*, S. 1778–1785.

[Ferber, 2003] Ferber, Reginald (2003). *Information Retrieval - Suchmodelle und Data-Mining-Verfahren für Textsammlungen und das Web*. dpunkt.verlag, Heidelberg.

[Fisher, 1936] Fisher, Ronald A (1936). *The use of multiple measurements in taxonomic problems*. Annals of eugenics, 7(2):179–188.

[Francos et al., 1993] Francos, Joseph M, A. Meiri und B. Porat (1993). *A unified texture model based on a 2-D Wold-like decomposition*. IEEE Transactions on Signal Processing, 41(8):2665–2678.

[Francos et al., 1995] Francos, Joseph M, A. Narasimhan und J. W. Woods (1995). *Maximum likelihood parameter estimation of textures using a Wold-decomposition based model.* IEEE Transactions on Image Processing, 4(12):1655–1666.

[Fraunhofer, 2013] Fraunhofer, IPA (2013). *Care-O-bot®3 PRODUKT-VISION EINES INTERAKTIVEN HAUSHALTSASSISTENTEN (besucht am 09.09.2014).* `http://www.care-o-bot.de/content/dam/careobot/de/documents/Produktblaetter/PB_300_309_Produktblatt_Care-o-bot.pdf`.

[van Gool et al., 1985] Gool, Luc van, P. Dewaele und A. Oosterlinck (1985). *Texture analysis anno 1983.* Computer Vision, Graphiscs, and Image Processing, 29. Jg.(3):336–357.

[Goyal, 1995] Goyal, R.K. (1995). *Scale and rotation invariant texture analysis based on structural property.* In: *Industrial Electonics, Control, and Insturmentation*, Bd. 2, S. 1290–1294.

[Goyal et al., 1994] Goyal, RK, W. L. Goh, D. P. Mital und K. L. Chan (1994). *Invariant element compactness for texture classification.* In: *Proceedings of the Third International Conference on Automation, Robotics and Computer Vision, Singapore*, S. 1902–1096.

[Guo und Zhang, 2010] Guo, Zhenhua und D. Zhang (2010). *A completed modeling of local binary pattern operator for texture classification.* IEEE Transactions on Image Processing, 19(6):1657–1663.

[Guyon et al., 2006] Guyon, Isabelle, S. Gunn, M. Nikravesh und L. A. Zadeh (2006). *Feature Extraction - Foundations and Applications.* Springer, Berlin, Heidelberg.

[Haberäcker, 1989] Haberäcker, Peter (1989). *Digitale Bildverarbeitung: Grundlagen und Anwendungen.* Hanser Studienbücher, München, Wien, 3. überarb. Aufl.

[Handels, 2009] Handels, Heinz (2009). *Medizinische Bildverarbeitung - Bildanalyse, Mustererkennung und Visualisierung für die computergestützte ärztliche Diagnostik und Therapie*. Springer-Verlag, Berlin Heidelberg New York, 2. überarb. u. erw. Aufl.

[Haralick, 1979] Haralick, R.M (1979). *Statistical and structural approaches to texture*. Proceedings of the IEEE, 67:786–804.

[Haralick und Shanmugam, 1973] Haralick, RM und K. Shanmugam (1973). *Computer classification of reservoir sandstones*. IEEE Transactions on Geoscience Electronics, 11(4):171–177.

[Haralick et al., 1973] Haralick, R.M, K. Shanmugam und I. Dinstein (1973). *Textural features for image classification*. IEEE Transactions on Systems, Man, and Cybernetics, SMC-3:610–621.

[Heinert, 2010] Heinert, Michael (2010). *Support Vector Machines?Teil 1: Ein theoretischer Überblick*. Zeitschrift für Geodäsie, Geoinformation und Landmanagement, 3. Jg.:179–189.

[Hermes, 1999] Hermes, Thorsten (1999). *Texturen: Analyse, Beschreibung und Synthese*. Infix.

[Holz et al., 2012] Holz, Dirk, S. Holzer, R. B. Rusu und S. Behnke (2012). *Realtime Plane Segmentation using RGB-D Cameras*. In: Röfer, Thomas, Hrsg.: *RoboCup 2011: Robot Soccer World Cup XV*, S. 306–317. Springer.

[Hsu et al., 2003] Hsu, Chih-Wei, C.-C. Chang, C.-J. Lin et al. (2003). *A practical guide to support vector classification*.

[Jähne, 2005] Jähne, Bernd (2005). *Digitale Bildverarbeitung*. Springer, Berlin, Heidelberg, 6. überarb. u. erw. Aufl.

[Jolliffe, 2002] Jolliffe, I.T. (2002). *Principal Component Analysis* -. Springer, Berlin, Heidelberg, 2nd ed. 2002 Aufl.

[Julesz, 1962] Julesz, Bela (1962). *Visual pattern discrimination*. IRE Transaction on Information Theory, 8(2):84–92.

[Julesz, 1981] Julesz, Bela (1981). *Textons, the elements of texture perception, and rheir interactions*. Nature, 290. Jg.(5802):91–97.

[Kriesel, 2008] Kriesel, David (2008). *Ein kleiner überblick über Neuronale Netze (besucht am 12.08.2014*. `http://www.dkriesel.com/science/neural_networks`.

[Lampert et al., 2009] Lampert, Christoph H, H. Nickisch und S. Harmeling (2009). *Learning to detect unseen object classes by between-class attribute transfer*. In: *IEEE Conference on Computer Vision and Pattern Recognition*, S. 951–958.

[Liu und Picard, 1994] Liu, Fang und R. W. Picard (1994). *Periodicity, directionality, and randomness: Wold features for perceptual pattern recognition*. In: *Pattern Recognition, 1994. Vol. 2-Conference B: Computer Vision & Image Processing., Proceedings of the 12th IAPR International. Conference on*, Bd. 2, S. 184–189.

[Liu und Picard, 1996] Liu, Fang und R. Picard (1996). *Periodicity, directionality, and randomness: wold features of image and modeling and retrieval*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 18:722–733.

[Moll und Bundesfo, 2003] Moll, Eckard und D. . S. d. Bundesfo (2003). *Einführung in die Biometrie. 4. Grundlagen der Korrelationsanalyse und der Regressionsanalyse* -. Saphir-Verlag.

[Nava et al., 2011] Nava, Rodrigo, G. Cristóbal und B. Escalante-Ramírez (2011). *Invariant texture analysis through Local Binary Patterns*. arXiv preprint arXiv:1111.7271.

[Niemann, 1983] Niemann, H. (1983). *Klassifikation Von Mustern*. Springer-Verlag New York Incorporated, Berlin Heidelberg.

[Ojala et al., 2002] Ojala, Timo, M. Pietikainen und T. Maenpaa (2002). *Multiresolution gray-scale and rotation invariant texture classification with local bina-*

ry patterns. IEEE Transactions on Pattern Analysis and Machine Intelligence, 24(7):971–987.

[OpenCV, 2014a] OpenCV (2014a). *About OpenCV (besucht am 7.10.2014)*. `http://opencv.org/about.html`.

[OpenCV, 2014b] OpenCV (2014b). *Mean Shift (besucht am 17.10.2014)*. `http://docs.opencv.org/trunk/doc/py_tutorials/py_video/py_meanshift/py_meanshift.html`.

[OpenCV, 2014c] OpenCV (2014c). *Neural Networks (besucht am 19.10.2014)*. `http://docs.opencv.org/modules/ml/doc/neural_networks.html`.

[Palm, 2004] Palm, Christoph (2004). *Color texture classification by integrative co-occurrence matrices*. Pattern Recognition, 37(5):965–976.

[Pietikäinen et al., 2000] Pietikäinen, Matti, T. Ojala und Z. Xu (2000). *Rotation-invariant texture classification using feature distributions*. Pattern Recognition, 33(1):43–52.

[Rao und Lohse, 1993] Rao, A Ravishankar und G. L. Lohse (1993). *Towards a texture naming system: identifying relevant dimensions of texture*. In: *IEEE Conference on Visualization, 1993. Visualization'93, Proceedings*, S. 220–227.

[ROS, 2014a] ROS (2014a). *About ROS (besucht am 07.10.2014)*. `http://www.ros.org/about-ros/`.

[ROS, 2014b] ROS (2014b). *ROS Concepts (besucht am 07.10.2014)*. `http://wiki.ros.org/ROS/Concepts`.

[Schaefer und Doshi, 2012] Schaefer, Gerald und N. P. Doshi (2012). *Multi-dimensional local binary pattern descriptors for improved texture analysis*. In: *21st International Conference on Pattern Recognition (ICPR)*, S. 2500–2503.

[Scherer, 1997] Scherer, Andreas (1997). *Neuronale Netze Grundlagen Und Anwendungen*. Vieweg Verlag, Friedr, und Sohn Verlagsgesellschaft mbH, Braunschweig, Wiesbaden, 1997 Aufl.

[Schölkopf et al., 2000] Schölkopf, Bernhard, A. J. Smola, R. C. Williamson und P. L. Bartlett (2000). *New support vector algorithms*. Neural computation, 12(5):1207–1245.

[Shoyaib et al., 2011] Shoyaib, Mohammad, M. Abdullah-Al-Wadud und O. Chae (2011). *A Noise-Aware Coding Scheme for Texture Classification*. Sensors, 11(8):8028–8044.

[Sivakumar und Goutsias, 1999] Sivakumar, K. und J. Goutsias (1999). *Morphologically constrained GRFs: applications to texture synthesis and analysis*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 21:148–153.

[Sutton und Barto, 1998] Sutton, Richard S. und A. G. Barto (1998). *Reinforcement Learning - An Introduction*. MIT Press, Cambridge.

[Suzuki et al., 1985] Suzuki, Satoshi et al. (1985). *Topological structural analysis of digitized binary images by border following*. Computer Vision, Graphics, and Image Processing, 30(1):32–46.

[Tamura et al., 1978] Tamura, H., S. Mori und Y. Yamawaki (1978). *Textural Features Corresponding to Visual Perception*. IEEE Transactions on Systems, Man, and Cybernetics, SMC-8:460–473.

[Thrun et al., 2002] Thrun, Sebastian et al. (2002). *Robotic mapping: A survey*. Exploring artificial intelligence in the new millennium, S. 1–35.

[Tuceryan und Jain, 1998] Tuceryan, Mihran und A. K. Jain (1998). *Texture Analysis*. In: Chen, C.H. und W. P. Pau, L.F., Hrsg.: *The Handbook of Pattern Recognition and Computer Vision*, Kap. 2.1, S. 207–248. World Scientific Publishing Co.

[Voss und Süsse, 1991] Voss, Klaus und H. Süsse (1991). *Praktische Bildverarbeitung*. Hanser, München.

[Wang, 2010] Wang, Yu-Hsiang (2010). *Tutorial: Image Segmentation*. National Taiwan University, Taipei, S. 1–36.

[Weis, 2000] Weis, Martin (2000). *Verwendung von Texturparametern bei der Klassifizierung von hochauflösenden Satellitendaten.*

[Whelan und Ghita, 2008] Whelan, Paul F. und O. Ghita (2008). *Color Texture Analysis.* In: Mirmedi, Majid, X. Xie und J. Suri, Hrsg.: *Handbook of Texture Analysis*, Kap. 5, S. 129–163. Imperial Colege Press.

[Wikipedia, 2014] Wikipedia (2014). *HSV-Farbraum (besucht am 25.10.2014).* `http://de.wikipedia.org/wiki/HSV-Farbraum`.

[Yuen et al., 1990] Yuen, HK, J. Princen, J. Illingworth und J. Kittler (1990). *Comparative study of Hough transform methods for circle finding.* Image and Vision Computing, 8(1):71–77.

[Zhang und Tan, 2002] Zhang, Jianguo und T. Tan (2002). *Brief review of invariant texture analysis methods.* Pattern Recognition, 35:735–747.

[Zhang, 1996] Zhang, Yu Jin (1996). *A survey on evaluation methods for image segmentation.* Pattern recognition, 29(8):1335–1346.