



Rheinische
Friedrich-Wilhelms-
Universität Bonn



Institute for Computer Science
Department VI
Autonomous Intelligent Systems

RHEINISCHE
FRIEDRICH-WILHELMS-UNIVERSITÄT BONN

BACHELORARBEIT

Objektklassifikation, Identifizierung und
Posenschätzung mit Hilfe von
vortrainierten Konvolutionsnetzen

Autor:

Max SCHWARZ

Erstgutachter:

Prof. Dr. Sven BEHNKE

Zweitgutachter:

Priv.-Doz. Dr. Volker STEINHAGE

Betreuer:

Hannes SCHULZ

Abgabe: 8. Dezember 2014

Eidesstattliche Erklärung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Bachelorarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen verwendet habe. Die Stellen der Arbeit sowie evtl. beigefügte Abbildungen, Zeichnungen oder Grafiken, die anderen Werken dem Wortlaut oder Sinn nach entnommen wurden, habe ich unter Angabe der Quelle kenntlich gemacht.

Ort, Datum

Unterschrift

Zusammenfassung

Objekterkennung und Posenschätzung sind wichtige Fähigkeiten für Manipulationsroboter. Ohne diese sind sie nicht in der Lage, in natürlichen Umgebungen Manipulationsaufgaben zu erfüllen. Durch die neuerdings gute Verfügbarkeit von RGB-D-Sensorik bietet es sich an, RGB-D-Daten für diese Zwecke zu nutzen.

Diese Arbeit basiert auf neuronalen Konvolutionsnetzen, die für Bildklassifizierung vortrainiert wurden. Es wird eine neue Vorverarbeitung für Tiefendaten vorgestellt, die es erlaubt, ein auf RGB-Daten trainiertes Netz auch für Tiefendaten zu benutzen.

Die mit Hilfe des vortrainierten Netzes extrahierten Merkmalsvektoren werden dann zum überwachten Lernen einer Hierarchie von SVM-Klassifikatoren und -Regressoren verwendet. Die Hierarchie erlaubt auch die Schätzung von Posen ohne korrekte Instanzklassifizierung – eine wichtige Fähigkeit in vielen Bereichen.

Der Ansatz wird schließlich auf dem Washington-RGB-D-Objects-Datensatz evaluiert und demonstriert dort die semantische Ausdrucksstärke der berechneten Merkmale. Aktuelle Ergebnisse in der Objektklassifikation und der Posenschätzung werden verbessert. Die Performanz degradiert nur langsam, wenn die Anzahl der Trainingsbeispiele reduziert wird.

Inhaltsverzeichnis

| | |
|---|-----------|
| 1. Einleitung | 1 |
| 1.1. Methodik und Zielsetzung | 1 |
| 1.2. Gliederung | 2 |
| 2. Verwandte Arbeiten | 5 |
| 3. Grundlagen | 7 |
| 3.1. Neuronale Konvolutionsnetze | 7 |
| 3.2. Supportvektor-Klassifikation und -Regression | 8 |
| 4. Merkmalsextraktion mit CNNs | 11 |
| 4.1. RGB-Bilder | 11 |
| 4.2. Tiefenbilder | 12 |
| 4.3. Merkmalsextraktion | 16 |
| 5. Lernverfahren | 19 |
| 5.1. Klassifizierung | 19 |
| 5.2. Posenschätzung | 19 |
| 6. Evaluation | 21 |
| 6.1. Evaluationsverfahren | 21 |
| 6.2. Ergebnisse | 22 |
| 7. Zusammenfassung und Ausblick | 31 |
| Anhang | 33 |
| A. t-SNE-Einbettungen | 33 |

Abbildungsverzeichnis

| | |
|---|----|
| 3.1. CNN-Architektur | 8 |
| 3.2. SVM und SVR | 9 |
| 4.1. Überblick über die RGB-Vorverarbeitung | 12 |
| 4.2. Überblick über die Vorverarbeitung für Tiefendaten | 13 |
| 4.3. Koordinatensystem für Einfärbung | 15 |
| 4.4. CNN-Aktivitätsbilder | 17 |
| 5.1. Hierarchische Klassifikation und Posenschätzung | 20 |
| 6.1. RGB-Beispielsequenzen aus dem Washington RGB-D Objects Dataset | 22 |
| 6.2. Visualisierung des Merkmalsraumes | 24 |
| 6.3. Visualisierung des Merkmalsraumes pro Kategorie | 25 |
| 6.4. Beispiele für Verwechslungen | 26 |
| 6.5. Konfusionsmatrix für Kategorie-Erkennung | 26 |
| 6.6. Posenfehler pro Kategorie | 27 |
| 6.7. Lernkurven für Klassifikation und Posenschätzung | 28 |
| A.1. CNN: Kategorien | 33 |
| A.2. CNN: Instanzen | 34 |
| A.3. PHOW: Kategorien | 35 |
| A.4. PHOW: Instanzen | 36 |

Tabellenverzeichnis

| | |
|---|----|
| 6.1. Vergleich der Korrektklassifikationsrate | 25 |
| 6.2. Vergleich der Posenschätzungsfehler | 25 |
| 6.3. Farbpaletten-Evaluation | 29 |
| 6.4. Laufzeiten | 29 |

1. Einleitung

In vielen Forschungs- und Anwendungsbereichen spielt das maschinelle Sehen eine sehr wichtige Rolle. Insbesondere die Objektklassifikation ist oft elementar, um eine bestimmte Aufgabe lösen zu können. Hier sei als Beispiel ein Haushaltsroboter genannt, der oft bestimmte Objekte finden und manipulieren soll. Dazu muss er die Objekte in seinem Umfeld voneinander unterscheiden können. Als diskriminatives Merkmal eignet sich hier in einer menschlichen Umgebung vor allem die optische Erscheinung der Objekte – sie wurden für Menschen konstruiert, die Objekte hauptsächlich mit dem Auge wahrnehmen. Also bieten sich optische Sensoren besonders an, um solche Objekte zu klassifizieren. Weiterhin ist in diesem Szenario oft auch die Pose des Objektes zu schätzen – sobald der Roboter das Zielobjekt identifiziert hat, soll es gegriffen werden. Dazu muss bei vielen nicht-symmetrischen Objekten die Orientierung im Raum bekannt sein, sonst besteht die Gefahr eines Fehlgriffs.

1.1. Methodik und Zielsetzung

Methodisch haben sich in den letzten Jahren im Feld der Objektklassifikation die neuronalen Konvolutionsnetze aus dem Tiefenlernen durchgesetzt. Ihr anhaltende Erfolg im maschinellen Sehen ist auf die enormen Größen der Trainingsdatenbanken und die großen Fortschritte in der Verarbeitungsgeschwindigkeit im Trainieren der nicht-linearen Modelle zurückzuführen. Allerdings ist die verfügbare Datenmenge abhängig von dem konkreten Problem. Gerade in Robotik-Anwendungen müssen die verwendeten Ansätze mit sehr wenig Daten zurecht kommen, da das Erzeugen und Annotieren von Datensätzen sehr spezifisch im Hinblick auf Roboter und Aufgabe (z.B. Greifen) ist und somit der Aufwand größere Datensätze unmöglich macht. Diese Arbeit versucht, dieses Problem zu lösen, indem einmal von einem Konvolutionsnetz auf einem großen Datensatz gelernte Merkmale auf einem vergleichsweise kleinen Datensatz von Haushaltsobjekten wiederverwendet werden. Arbeiten aus anderen Bereichen (Donahue u. a., 2014; Girshick u. a., 2014; Razavian u. a., 2014) suggerieren, dass dieses Transfer-Lernen eine vielversprechende Alternative zu manuellem Merkmalsdesign ist.

1. Einleitung

Um ein Konvolutionsnetz (im Folgenden auch CNN nach der englischen Bezeichnung „Convolutional Neural Network“ genannt) einsetzen zu können, müssen die Eingabedaten vorverarbeitet werden. Der in dieser Arbeit entwickelte Algorithmus segmentiert Objekte auf flachen Untergründen, entfernt evtl. ablenkende Hintergrundinformationen und gleicht die Daten der Eingabeverteilung aus dem Training des CNNs an. Die CNN-berechneten Merkmale werden dann mit Hilfe von Supportvektormaschinen (SVM) klassifiziert um Objektkategorie und -instanz festzustellen. Weiterhin wird unter Verwendung von Supportvektorregression (SVR) auch eine Pose des Objektes geschätzt. Während dieser Ansatz bereits vergleichbar mit anderen aktuellen Arbeiten ist, kann er jedoch keine Tiefeninformationen benutzen.

Tiefensensoren sind in der heutigen Robotik weit verbreitet, es gibt jedoch keine frei verfügbaren und großen Datensätze, die zum Training eines CNNs geeignet wären¹. Diese Arbeit schlägt vor, die Tiefendaten so zu transformieren, dass das auf Farbbildern trainierte CNN diese einfach interpretieren kann. Nachdem die Ebene detektiert wurde, auf der das Objekt steht, wird das Objekt aus einer kanonischen Pose gerendert und abhängig von der Distanz zum Objektzentrum eingefärbt. Kombiniert mit Farb-Merkmalen schlägt dieses Verfahren die aktuell besten Algorithmen auf dem Washington RGB-D-Objects-Datensatz in vielen Problemstellungen, welche die Kategorisierung von Haushaltsobjekten, Erkennung von einzelnen Instanzen und das Schätzen der jeweiligen Objektpose umfassen.

1.2. Gliederung

Im folgenden Kapitel 2 werden zunächst verwandte Arbeiten im Bereich von Konvolutionsnetzen und Arbeiten auf dem betrachteten Datensatz vorgestellt.

Es folgt eine kurze Einführung in die verwendeten Grundlagen von Konvolutionsnetzen und Supportvektormaschinen (Kapitel 3).

Anschließend wird im Kapitel 4 die Vorverarbeitung der Eingabebilder und die Verwendung eines CNNs zur Merkmalsextraktion präsentiert. Insbesondere wird eine Methode zur Vorverarbeitung der Bilder entwickelt, die die Verarbeitung von beliebigen RGB- und Tiefenbildern erlaubt.

Das Kapitel 5 beschreibt das verwendete hierarchische Lernverfahren zur Objektklassifikation und Posenschätzung.

Im Kapitel 6 wird dann das entwickelte System auf einem bekannten annotierten Datensatz von RGB- und Tiefenbildern evaluiert. Dazu wird der entstehende Merkmalsraum analysiert, und quantitative Vergleiche mit aktuellen Arbeiten auf

¹ Ausnahme: Synthetische Datensätze, z.B. für Microsoft Kinect (Shotton u. a., 2013)

dem Datensatz angestellt. Auch die Laufzeit der Methode wird untersucht.

Schließlich bietet Kapitel 7 einen abschließenden Überblick über die Ergebnisse der Arbeit und erlaubt einen Ausblick auf Möglichkeiten anschließender Forschung.

2. Verwandte Arbeiten

Konvolutionsnetze (CNN) aus dem Tiefenlernen (LeCun, Bottou u. a., 1998) sind die dominante Methode in der ImageNet Large Scale Image Classification Challenge (Russakovsky u. a., 2014) seit der bahnbrechenden Arbeit von Krizhevsky u. a. (2012). Ihr Erfolg kann auf die Verfügbarkeit von großen Datenmengen zum Trainieren und schnellen GPU-basierten Implementierungen zurückgeführt werden, welche es möglich machten, große nicht-lineare Modelle zu benutzen. Um die Aufgabe zu lösen, 1000 (manchmal sehr ähnliche) Objektkategorien zu unterscheiden, berechnen die Netzwerke neue Repräsentationen der Eingabebilder durch wiederholte Faltungen, aggregieren Aktivitäten durch lokale Maximalwertbildung und benutzen stark nicht-lineare Operationen (Krizhevsky u. a., 2012). Die semantische Ausdrucksstärke der Merkmalsräume steigt mit der Höhe im Konvolutionsnetz, daher sind die höheren Repräsentationen von besonderem Interesse (Zeiler und Fergus, 2014). Diese Beobachtung wird gestützt durch die bemerkenswerte Performanz von CNN-Merkmalen, die nur auf Klassifizierungs-Problemstellungen trainiert wurden und nun auf neuen Aufgaben im maschinellen Sehen zum Einsatz kommen. Zu nennen sind hier Objekterkennung (Girshick u. a., 2014; Razavian u. a., 2014), Sub-Kategorisierung, Wechsel der Domäne, Szenenerkennung (Donahue u. a., 2014), Attributserkennung und Informationsrückgewinnung (Razavian u. a., 2014). In vielen Fällen sind die Ergebnisse, die von CNNs erzielt werden, auf Augenhöhe oder besser als aktuelle Algorithmen aus den jeweiligen Problemfeldern.

Während Girshick u. a. (2014) von Verbesserungen durch Feinabstimmung von CNNs¹ berichten, hat dieses Vorgehen Probleme, wenn nur wenig Trainingsbeispiele vorliegen. Dann kann es schnell zur Überanpassung kommen. In dieser Arbeit wird stattdessen dem Vorgehen von Donahue u. a. (2014) gefolgt, und lediglich die Ausgaben des Netzes neu interpretiert. Im Unterschied zu den Untersuchungen von Donahue u. a. (2014) und Razavian u. a. (2014) konzentriert sich diese Arbeit auf einen Datensatz aus der Robotik mit relativ wenigen annotierten Instanzen. Es wird auch gezeigt, wie das vortrainierte CNN zusammen mit Tiefendaten benutzt werden kann, was von anderen Arbeiten nicht angesprochen wird.

¹Hier wird also ausgehend von dem vortrainierten CNN mit Hilfe neuer Trainingsbeispiele weiter trainiert, oft mit stärkerem Fokus auf höhere Ebenen im Netzwerk

2. Verwandte Arbeiten

Die Arbeiten auf dem untersuchten RGB-D Objects Dataset beginnen mit der Datensatz-Publikation von Lai u. a. (2011a), die eine Kombination verschiedener manuell entworfener Merkmale (SIFT, Texton, Farbhistogramm, Drehbilder, minimale umschließende Rechtecke) benutzen und verschiedene Klassifizierer vergleichen (lineare SVM, Gaußkern-SVM, Entscheidungsbäume und Random Forests). Diese grundlegenden Ergebnisse wurden in späteren Arbeiten signifikant verbessert.

Lai u. a. (2011b) stellen ein sehr effizientes hierarchisches Verfahren zur Klassifizierung vor, welches Klassifikations- und Posenschätzungsfehler gleichzeitig auf allen Hierarchieebenen minimiert. Dazu kommt der stochastische Gradientenabstieg (engl. Stochastic Gradient Descent, SGD) zum Einsatz. Dadurch ist das Verfahren sehr effizient und kann neue Objekte leicht zur Hierarchie hinzufügen, ohne das Training von neuem zu beginnen (sog. warm start). Obwohl die Trainingsmethode sehr interessant ist und auch in dieser Arbeit verwendet werden könnte, bleiben die Ergebnisse signifikant hinter dem aktuellen Stand zurück.

Letztlich demonstrieren Bo u. a. (2013) eine signifikante Verbesserung in der Genauigkeit der Klassifizierung und eine große Reduktion des Posenschätzungsfehlers. Ihre Methode lernt hierarchische Merkmalsrepräsentationen von RGB-D-Daten ohne Supervision mit Hilfe von Hierarchical Matching Pursuit (HMP, zu deutsch etwa "Hierarchische Verfolgung von Matchings"). Sie zeigt die Möglichkeiten des Merkmallernens von rohen Daten und ist die aktuell beste Arbeit auf dem RGB-D-Objects-Datensatz. Allerdings muss die Anzahl der Trainingsbeispiele ausreichend hoch sein um robuste Merkmale zu lernen – im Gegensatz zu der hier vorgestellten Methode, die von „vorgelernten“ Merkmalen ausgeht und so mit wenigen Trainingsbeispielen auskommt. Dadurch fällt auch der Prozess des Lernens von Merkmalen weg – ein großer Geschwindigkeitsvorteil für die vorgestellte Methode. Schließlich erzeugt der CNN-basierte Ansatz auch weniger Merkmalsdimensionen (10 192 vs. 188 300) und ist so sowohl schneller im Trainieren von Klassifikatoren, als auch in der Vorhersage.

3. Grundlagen

3.1. Neuronale Konvolutionsnetze

Neuronale Konvolutionsnetze (engl. convolutional neural network, CNN) besitzen drei Merkmale, die sie von klassischen neuronalen Netzen unterscheiden: Einschränkung der Konnektivität zwischen den Ebenen, Neuronengruppen mit gemeinsamen Gewichten, und räumliche Aggregation von Aktivitäten (*spatial pooling*). Konvolutionsnetze gehen auf die Arbeit von Fukushima (1980) zurück und wurden u.a. von LeCun, B. Boser u. a. (1989) verbessert und zur Erkennung von Handschrift eingesetzt.

Genauer besteht eine CNN-Ebene aus Neuronen, die in 2D-Gittern, sog. *Feature Maps* gruppiert sind (siehe Abbildung 3.1). Eine solche Map stellt jeweils eine Repräsentation des Eingabebildes dar. Die Neuronen jeder Map sind auf der Eingabeseite nur mit Neuronen der vorherigen Ebene verbunden, die räumlich gesehen in der Gitterstruktur benachbart sind. Durch diese Einschränkung lernen die Neuronen verstärkt, lokale Merkmale zu detektieren.

Weiterhin besitzen alle Neuronen einer Feature Map die selben Gewichte. Dieses *weight-sharing* genannte Verfahren reduziert die Anzahl der Parameter deutlich. Einerseits wird hierdurch das Training größerer Netze auf aktueller Hardware überhaupt erst möglich, andererseits reduziert sich auch die Überanpassung (engl. Overfitting) durch zu wenig Trainingsbeispiele. Durch die Struktur wird sogar eine gewisse Invarianz gegenüber Translationen und Skalierungen der Eingabe garantiert¹.

Um die Anzahl der Merkmale in späteren Ebenen zu reduzieren, benutzen CNNs Aggregationen (engl. *pooling*). Hierbei werden die Aktivitäten einer Map abgetastet bzw. aggregiert, sodass eine Repräsentation von niedrigerer Auflösung entsteht. Hier kommt meist das sog. *max-pooling* zum Einsatz, welches lokale Maxima der Aktivitäten bildet, sodass starke Merkmale besonders propagiert werden.

Zum Training der Netzwerke (also zum Finden der Gewichte, die den Vorhersagefehler minimieren) kommt bei aktuellen Arbeiten der SGD-Algorithmus (Stocha-

¹Diese Invarianz gilt natürlich nur, wenn um eine ganzzahlige Anzahl Pixel verschoben bzw. skaliert wird.

3. Grundlagen

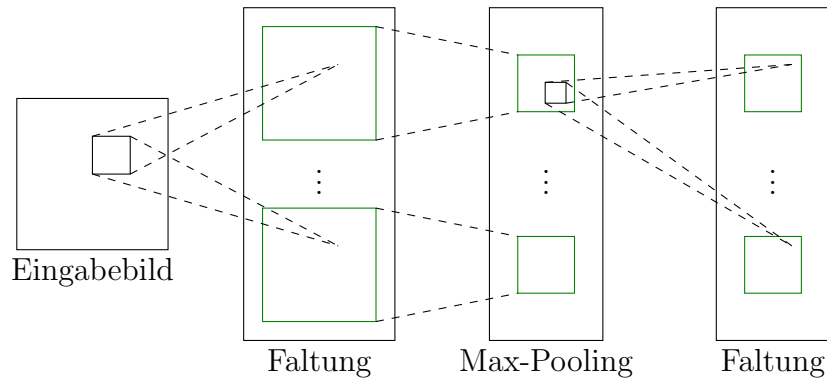


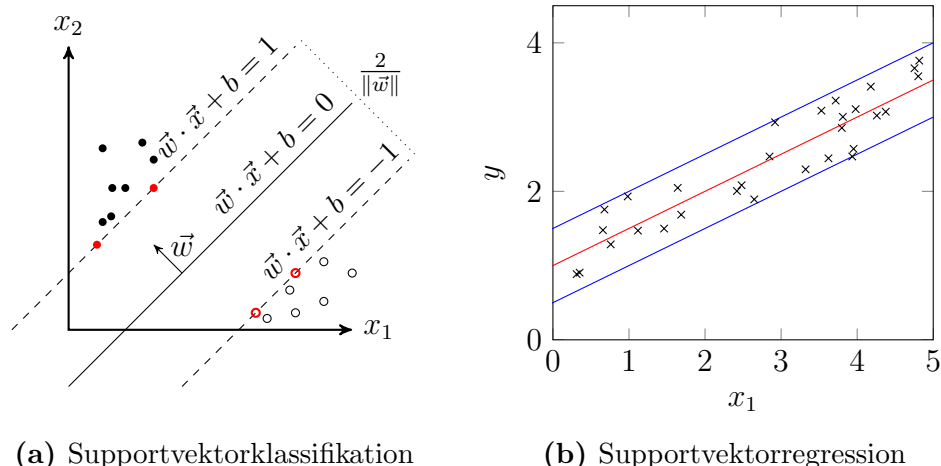
Abbildung 3.1.: Ausschnitt aus einer beispielhaften CNN-Architektur. Zu sehen ist eine Faltung gefolgt von einer Max-Pooling-Operation und einer erneuten Faltung. Die Feature Maps jeder Ebene sind grün umrandet.

stic Gradient Descent) zum Einsatz. Dies ist eine Variante des Gradientenabstiegs, die den Gradienten iterativ jeweils auf einem Trainingsbeispiel schätzt und sich in Richtung des (lokalen) Optimums bewegt. Durch die iterative Gestalt kann der Algorithmus sehr effizient implementiert werden. Durch die Verwendung von GPUs kann die benötigte Vorwärts- und Rückwärtspropagierung innerhalb des Netzwerkes deutlich beschleunigt werden. Dennoch liegen die Trainingszeiten für größere Netze auf großen Bilddatenbanken im Bereich von mehreren Tagen, für das später betrachtete *CaffeNet* bei mehr als einer Woche.

Meistens werden CNNs darauf trainiert, für ein Eingabebild Klassenwahrscheinlichkeiten $P(C = c_i)$ zu schätzen. Jedoch kann ein einmal trainiertes Netz auch dazu benutzt werden, Merkmalsvektoren für ein Eingabebild zu berechnen, in dem Ausgaben in beliebigen Ebenen abgegriffen werden. So können die gelernten Merkmalsdeskriptoren aus den Faltungsebenen für neue Aufgaben benutzt werden.

3.2. Supportvektor-Klassifikation und -Regression

Um Objekte zu klassifizieren, werden in dieser Arbeit lineare Supportvektormaschinen (engl. Support Vector Machine, SVM) benutzt. Diese gehen zu großen Teilen auf V. Vapnik zurück (B. E. Boser u. a., 1992; Cortes und Vapnik, 1995). Die Idee ist, den Eingaberaum durch Hyperebenen zu trennen, die maximalen Abstand zu den Trainingsbeispielen haben (siehe Abbildung 3.2a). Diese Idee führt für das einfache binäre Problem mit einer Hyperebene der Gleichung $\mathbf{w} \cdot \mathbf{x} - b = 0$



(a) Supportvektorklassifikation

(b) Supportvektorregression

Abbildung 3.2.: Supportvektor-Methoden zur Klassifikation und Regression

zum folgenden Optimierungsproblem:

$$\begin{aligned} \text{Minimiere bzgl. } \mathbf{w} : & \quad \|\mathbf{w}\|^2 \\ \text{sodass } \forall i = 1, \dots, n : & \quad y_i(\mathbf{w} \cdot \mathbf{x}_i - b) \geq 1, \end{aligned} \quad (3.1)$$

wobei x_i das i -te der n Trainingsbeispiele meint und $y_i \in -1, 1$ das zugehörige Label.

Da die Daten oft nicht perfekt linear separierbar sind, führt man Schlupfvariablen ξ_i ein, die Verletzungen von (3.1) erlauben:

$$\begin{aligned} \text{Minimiere bzgl. } \mathbf{w}, \xi_i : & \quad \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n \xi_i \\ \text{sodass } \forall i = 1, \dots, n : & \quad y_i(\mathbf{w} \cdot \mathbf{x}_i - b) \geq 1 - \xi_i \end{aligned} \quad (3.2)$$

Die Konstante C bestimmt hierbei, wie wichtig es ist, keine Fehler zu machen (großes C) oder möglichst großen Abstand zu den Beispielen zu erreichen (kleines C). Da C ein Hyperparameter der SVM ist, muss es durch Kreuzvalidierung auf dem Datensatz bestimmt werden (siehe Kapitel 6).

Zur Lösung des Optimierungsproblems stehen verschiedene Ansätze zur Verfügung, die das primale Problem oder das korrespondierende duale Problem lösen. In dieser Arbeit werden die Algorithmen aus der LIBLINEAR-Bibliothek² benutzt. Die Wahl der primalen bzw. dualen Methode hängt von dem Verhältnis von Trainingsbeispielen zu Merkmalsdimensionen ab, da der duale Algorithmus effizienter ist, falls weniger Beispiele als Dimensionen vorliegen.

²<http://www.csie.ntu.edu.tw/~cjlin/liblinear/>

3. Grundlagen

Für den Fall von mehr als zwei Klassen kann die Methode auf zwei Arten erweitert werden: Entweder wird für jede Klasse eine SVM trainiert, die diese Klasse entscheidet („1-vs-all“), oder eine SVM für jede mögliche Kombination („1-vs-1“). Im letzteren Fall muss zusätzlich ein Punkteschema implementiert werden, um die Klasse mit den meisten „gewonnenen“ Zweikämpfen zu bestimmen. In dieser Arbeit wird ausschließlich die „1-vs-all“-Strategie verfolgt. Es gibt Arbeiten, die suggerieren, dass die Wahl der Strategie keinen großen Einfluss auf die Performanz hat (Hsu und Lin, 2002).

Eine ähnliche Methodik kann zur Regression von kontinuierlichen Funktionen benutzt werden. Hier wird eine lineare Funktion gesucht, für die alle Trainingsbeispiele innerhalb eines gewissen Korridors liegen (siehe Abbildung 3.2b). Für Details sei hier auf die exzellente Einführung von Smola und Schölkopf (2004) verwiesen.

Schließlich können beide Methoden (Klassifikation und Regression) erweitert werden, in dem die Beispiele durch eine nicht-lineare Funktion in einen höherdimensionalen Raum transformiert werden, und dann dort mit Hilfe der bekannten Methoden gelernt wird. So können auch nicht-linear-separierbare Daten getrennt werden. Dieser Trick findet sich unter dem Namen „Kernel-Trick“ in der Literatur (u.a. Smola und Schölkopf (2004)). Zur Klassifikation kommen in dieser Arbeit nur lineare SVMs zum Einsatz. Zur Posenregression wird der Trick jedoch angewandt, wie in Abschnitt 5.2 ausgeführt.

4. Merkmalsextraktion mit CNNs

Bevor ein vortrainiertes Konvolutionsnetzwerk zur Merkmalsextraktion benutzt werden kann, müssen die Eingabedaten in ein Format gebracht werden, welches dem Format der Trainingsdaten entspricht. Das hier benutzte *CaffeNet*-Netzwerk¹ basiert auf der Arbeit von Krizhevsky u. a. (2012) und erwartet quadratische 227×227 -Farbbilder, die ein dominantes Objekt darstellen. Dies entspricht den Vorgaben der ImageNet Large Scale Visual Recognition Challenge (Russakovsky u. a., 2014).

4.1. RGB-Bilder

Da das untersuchte Konvolutionsnetz auf RGB-Bildern trainiert wurde, wird keine komplexe Vorverarbeitung benötigt um robuste Merkmale aus den Farbbildern zu extrahieren. Beispielbilder aus der Vorverarbeitung sind in Abbildung 4.1 zu sehen.

Zuerst wird das Eingabebild auf ein minimales, das Objekt umschließendes Quadrat zugeschnitten (siehe Abbildung 4.1a). In einer Live-Situation wird dieses Quadrat aus den Objektpunkten bestimmt, die aus der Tischebenen-Segmentierung stammen (Abschnitt 4.2). In der Evaluation auf dem RGB-D-Objects-Datensatz (Kapitel 6), wird die vom Datensatz bereitgestellte Segmentierungsmaske benutzt um das Quadrat zu bestimmen.

Die extrahierte Region wird dann skaliert, um die CNN-Eingabegröße zu erreichen, in diesem Fall 227×227 Pixel. Um die Reaktionen des CNNs auf Hintergrundmerkmale zu vermindern, wird eine Ausblendungsoperation verwendet (Abbildung 4.1c). Jeder Pixel wird zwischen seinem RGB-Farbwert $\mathbf{c}_0 = (r_0, g_0, b_0)$ und dem korrespondierendem Pixel $\mathbf{c}_m = (r_m, g_m, b_m)$ im Mittelwertsbild des ILS-RVC 2011 interpoliert, basierend auf seiner Pixeldistanz r zu dem nächsten Objektpixel (Abbildung 4.1b):

$$\mathbf{c} := \alpha \cdot \mathbf{c}_0 + (1 - \alpha) \cdot \mathbf{c}_m, \quad (4.1)$$

¹<http://caffe.berkeleyvision.org/>

4. Merkmalsextraktion mit CNNs

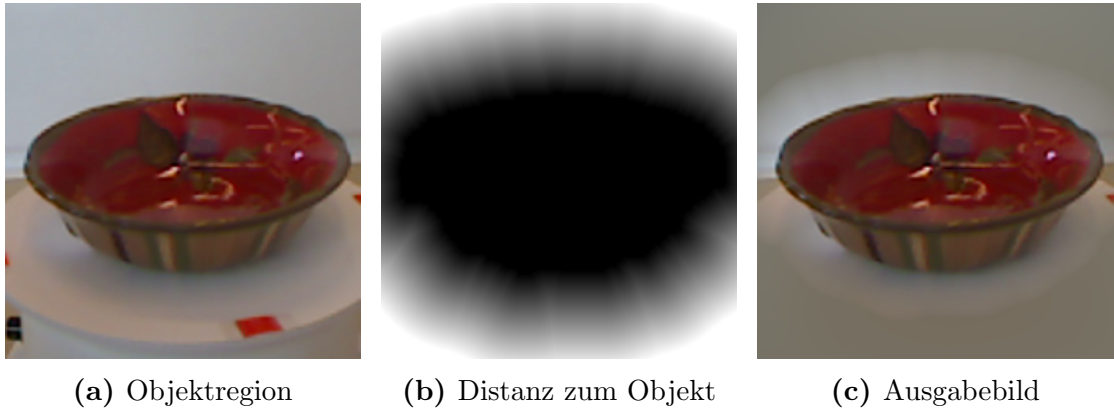


Abbildung 4.1.: Überblick der RGB-Vorverarbeitung, mit ROI aus der Objektsegmentierung, Distanztransformation zur Objektmaske und schließlich dem Ausgabebild ohne Hintergrund, welches zur CNN-Merkmalsextraktion benutzt wird.

mit

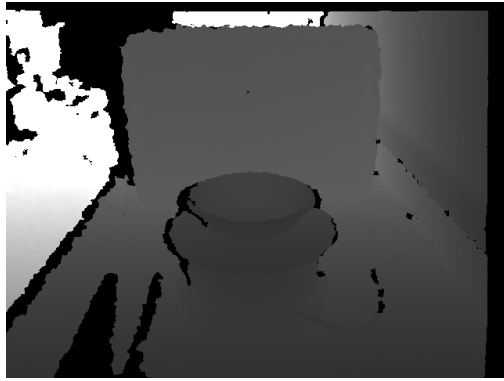
$$\alpha := \begin{cases} 1 & \text{falls } r = 0 \\ 0 & \text{falls } r > R \\ (R - r)^\beta & \text{sonst.} \end{cases} \quad (4.2)$$

Der Ausblendungsradius $R = 30$ wurde später manuell eingestellt, um den Hintergrund möglichst weitgehend auszublenden, ohne Bilder von Objekten mit fehlerhafter Segmentierung zu beschädigen. Der Exponent $\beta = 0,75$ wurde so bestimmt, dass er den Kreuzvalidierungsfehler in der Kategorie-Klassifizierung minimiert (siehe Kapitel 6).

4.2. Tiefenbilder

Tiefenbilder sind weitaus problematischer als RGB-Bilder. Von dem untersuchten CNN kann nicht erwartet werden, dass es robuste Merkmale aus Tiefenbildern extrahieren kann, da es nur auf Farbbildern trainiert wurde. Um dieses Problem zu beheben, werden bildähnliche Ansichten des Objektes aus den Tiefendaten mit Hilfe einer fünfschrittigen Vorverarbeitung erzeugt, die im Folgenden erklärt wird und in Abbildung 4.2 mit Beispielen zu sehen ist.

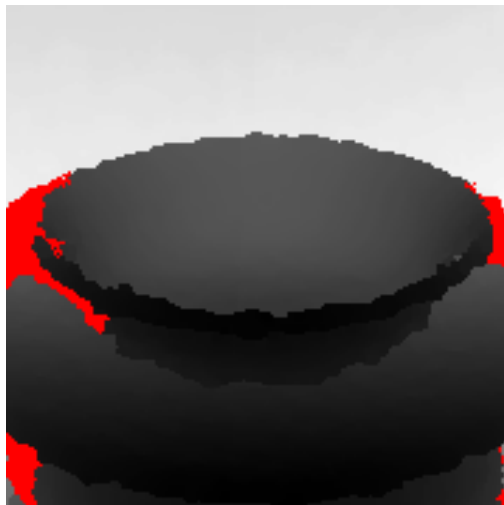
In einem ersten Schritt wird eine einfache Segmentierung vorgenommen, um die horizontale Oberfläche zu finden, auf der das Objekt steht. Analog zu dem Ansatz von Holz u. a. (2012) werden Oberflächennormalen direkt aus den Tiefenbildern geschätzt, Punkte von nicht-horizontalen Flächen verworfen und ein Ebenenmo-



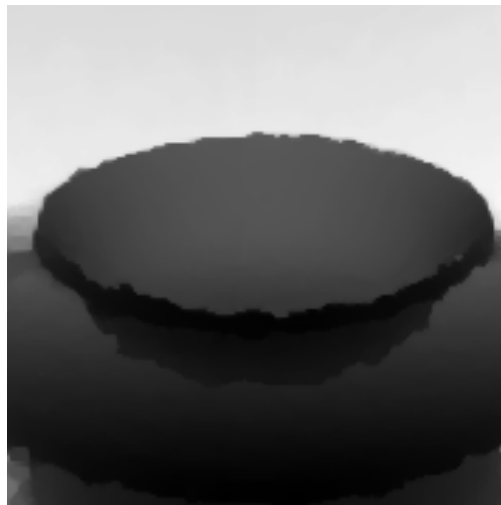
(a) Tiefenbild



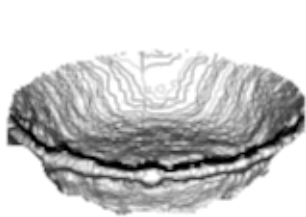
(b) Objektsegmentierung



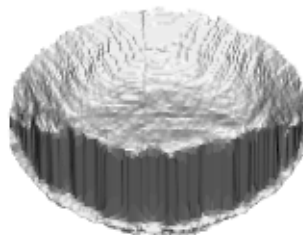
(c) Objektregion



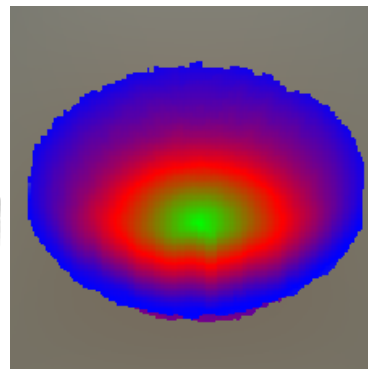
(d) Aufgefüllte Tiefe



(e) Berechneter Mesh



(f) Kanonische Ansicht



(g) Eingefärbte Ausgabe

Abbildung 4.2.: Überblick über die Vorverarbeitung für Tiefendaten, beginnend bei der Eingabe, über Reprojektion von einer kanonischen Kamerapose bis zu dem letztlich für die CNN-Merkmalsextraktion benutzten eingefärbten Bild. (c): nicht verfügbare Tiefenwerte rot eingefärbt.

4. Merkmalsextraktion mit CNNs

dell mit Hilfe von Random Sample Consensus (RANSAC) registriert. Das Ziel ist hier, ein lokales Referenzsystem am Objekt zu finden, welches in allen Dimensionen fixiert ist, außer in der Rotation um die Normale der Tischoberfläche durch das Objektzentrum. Das Ebenenmodell erlaubt es auch, Punkte oberhalb der Ebene zu finden und Objektcluster mit Hilfe von euklidischer Cluster-Analyse zu extrahieren (siehe Abbildung 4.2b).

Im nächsten Schritt werden Löcher im Tiefenbild aufgefüllt. Löcher entstehen meist durch Materialien, die durch die Tiefenkamera generell nicht messbar sind (etwa ungenügend reflektierende Materialien bei Kameras mit Infrarotprojektor), oder durch einen ungünstigen Winkel der Oberfläche zur Kamera. Diese Löcher sind einerseits problematisch, da sie eventuell hoch-frequente Messwerte an vielleicht uninteressanten Stellen erzeugen, andererseits, da sie die Geometrie der Szene an dieser Stelle zerstören. Dies macht eine Reprojektion (s.u.) sehr schwierig. Zur Auffüllung von Löchern bietet sich eine Methode basierend auf der Arbeit von Levin u. a. (2004) an, die das Einfärben von Grauwert-Bildern mit Hilfe von wenigen Farbpixeln untersuchten. Die Einfärbung wird durch das Grauwert-Bild gesteuert, sodass Bereiche mit gleicher Intensität ähnliche Farben zugewiesen bekommen. In der hier betrachteten Anwendung wird stattdessen das Tiefenbild aufgefüllt, gesteuert durch eine Grauwert-Kopie des korrespondierenden RGB-Bildes. Der Vorteil gegenüber den weit verbreiteten Median-Filtern ist, dass RGB-Informationen genutzt werden können, um Mehrdeutigkeiten zwischen verschiedenen Tiefenebenen aufzulösen.

Im Detail ist das Ziel des Auffüllens die Minimierung der quadratischen Abweichung zwischen dem Tiefenwert $D(\mathbf{p})$ mit $\mathbf{p} = (u, v)$ und einem gewichteten Mittelwert der Tiefe von benachbarten Pixeln,

$$J(D) = \sum_{\mathbf{p}} \left(D(\mathbf{p}) - \sum_{\mathbf{s} \in N(\mathbf{p})} w_{\mathbf{ps}} D(\mathbf{s}) \right)^2, \quad (4.3)$$

mit

$$w_{\mathbf{ps}} = \exp \left[-(G(\mathbf{p}) - G(\mathbf{s}))^2 / 2\sigma_{\mathbf{p}}^2 \right], \quad (4.4)$$

wobei $G(\mathbf{p})$ der Grauwert-Intensität von \mathbf{p} und $\sigma_{\mathbf{p}}$ der Varianz der Intensität in einem N -Fenster um \mathbf{p} entspricht. Die Minimierung von $J(D)$ führt auf ein dünn besetztes lineares Gleichungssystem, welches mit dem BiCGSTAB-Gleichungslöser aus der Eigen-Bibliothek für lineare Algebra² gelöst wird. Ein beispielhaftes Er-

²<http://eigen.tuxfamily.org/>

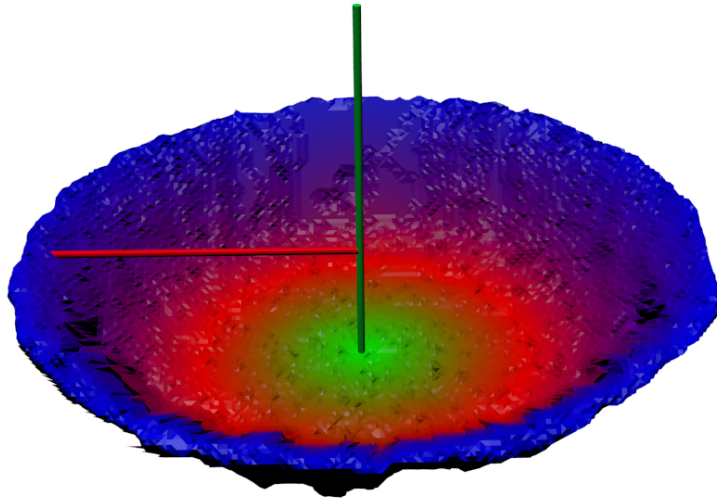


Abbildung 4.3.: Das zur Einfärbung verwendete Koordinatensystem. Die geschätzte Oberflächennormale \mathbf{n} durch das Objektzentrum \mathbf{p} , ist in grün eingezeichnet. Die Distanz zu einem Punkt auf dem Objekt ist rot markiert.

gebnis der Auffülloperation ist in Abbildung 4.2d zu sehen.

Nach der Auffülloperation werden die Tiefendaten mit einem Schattenfilter gefiltert, der Punkte, deren Normale nahezu im rechten Winkel zur Blickrichtung liegen, verworfen werden. Diese Operation wird nur an den Außenkanten des Objektes ausgeführt, um die gerade gewonnene Kontinuität der Tiefendaten innerhalb des Objekts nicht zu zerstören.

Um die Invarianz der generierten Bilder gegenüber Änderungen des Kamera-Neigewinkels weiter zu verstärken, kann ein optionaler Reprojektions-Schritt angewendet werden. Das Ziel ist es hier, eine Ansicht des Objektes aus einem kanonischen Neigewinkel zu erzeugen. Um die Reprojektion zu ermöglichen, wird zunächst ein Netz (Mesh) des Objektes aus einer naiven Triangularisierung aus dem aufgefüllten Tiefenbild erzeugt (Abbildung 4.2e). Dann wird das Objektnetz aus einer kanonischen Pose gerendert, um ein neues Tiefenbild zu erhalten. Dieser naive Ansatz zur Berechnung des Netzes erzeugt eine lineare Interpolation für vorher unsichtbare Oberflächen des Objektes (siehe Abbildung 4.2f). Dies stellt eine plausible Schätzung der unbekanntenen Geometrie dar, die ohne weitere Annahmen über das Objekt (z.B. Symmetrien) auskommt.

Um die Reprojektion abzuschließen, wird nun noch eine Farbe C für jeden Pixel $\mathbf{p} = (u,v)$ im Tiefenbild mit der 3D-Projektion $\mathbf{p}_3 = (x,y,z)$ bestimmt. Zunächst wird ein Zentrum \mathbf{q} des Objektes in 3D aus dem umgebenden Hyperrechteck der Objektpunktwolke geschätzt. Die Punkte werden dann abhängig von der Distanz r

4. Merkmalsextraktion mit CNNs

von der Geraden g durch \mathbf{q} in Richtung der Tischnormale \mathbf{n} aus der Tischebenen-Segmentierung eingefärbt (Abbildung 4.3):

$$C(\mathbf{p}) = P(\text{dist}_g(\mathbf{p})) \quad (4.5)$$

Hier fiel die Wahl auf eine RGB-Interpolation von Grün über Rot und Blau hin zu Gelb für die Palettenfunktion P . Da die Einfärbung nicht normalisiert ist, erlaubt sie dem Netzwerk skalierte Versionen derselben Objektgestalt zu unterscheiden. Die Wahl der Palettenfunktion wird in Kapitel 6 evaluiert.

Mit der Motivation, zuverlässig zusätzliche Ansichten des Objektes generieren zu können, wurde auch der Ansatz verfolgt, aus den Trainingsbildern ein komplettes Mesh des Objektes zu erzeugen. Durch die teilweise stark symmetrischen Objekte und ungenaue Posenannotierungen waren die erzeugten Meshes jedoch nicht zum Training geeignet.

4.3. Merkmalsextraktion

In dieser Arbeit wird das Gewinner-CNN der ILSVRC 2011 von Krizhevsky u. a. (2012) untersucht. Das quelloffene *Caffe*-Framework³ stellt eine vortrainierte Version dieses Netzwerk bereit.

Es werden die Merkmale aus den vorletzten und letzten voll-verbundenen Ebenen (genannt *fc7* und *fc8* in *Caffe*) extrahiert. Das heißt, dass $4\,096 + 1\,000 = 5\,096$ Merkmale jeweils pro RGB- und Tiefenbild entstehen, insgesamt also $10\,192$ Merkmale pro RGB-D-Bild.

Der Reprojektionsschritt und die Einfärbung werden nur für die Klassifikation auf Instanzebene und die Posenschätzung benutzt, da die Objektkategorisierung unter dem gegebenen Evaluationsschema auf dem RGB-D-Objects-Datensatz nicht von der kanonischen Perspektive profitieren kann (siehe Kapitel 6).

Abbildung 4.4 zeigt Antworten des Netzwerkes auf RGB- und Tiefendaten nach der ersten konvolutionellen Ebene. Die selben Filter zeigen verschiedene Verhalten in den RGB- und Tiefenkanälen. Wie gewünscht zeigen die RGB-Aktivitätsbilder wenig Aktivität im Hintergrund, ein Resultat der Hintergrundausbildung in der Vorverarbeitung.

³<http://caffe.berkeleyvision.org/>

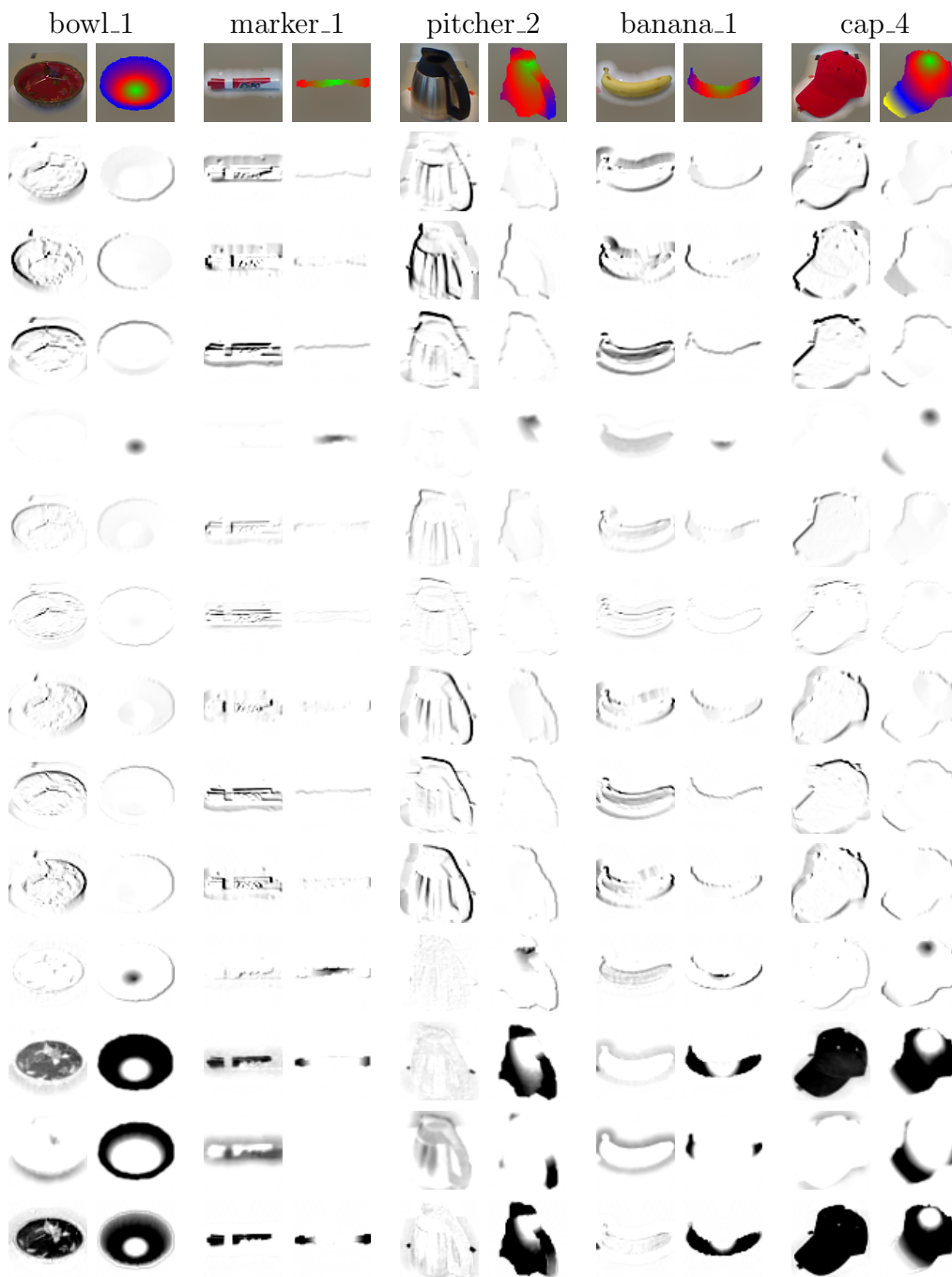


Abbildung 4.4.: Ausgewählte CNN-Aktivierungen für Beispielbilder aus dem RGB-D-Datensatz. Die erste Zeile zeigt das CNN-Eingabebild (Farbe und Tiefe pro Objekt), alle anderen Zeilen zeigen ausgewählte korrespondierende Aktivitäten des Netzwerkes (schwarz: hohe Aktivität) nach der ersten konvolutionellen Ebene. Man beachte, dass jede Zeile das Ergebnis des selben Filters (angewandt auf Farbe und vorverarbeitete Tiefe) ist.

5. Lernverfahren

5.1. Klassifizierung

Zur Objektklassifizierung wird analog zu Lai u. a. (2011b) eine Hierarchie konstruiert: Auf der ersten Ebene sagt eine lineare Multiklassen-SVM die Objektkategorie voraus. Die nächste Ebene enthält SVMs, die die Instanz in der jeweiligen Kategorie voraussagen. Abbildung 5.1 visualisiert die Hierarchie von Klassifikatoren.

5.2. Posenschätzung

Der RGB-D-Objects-Datensatz macht die Annahme, dass die Orientierung eines Objektes durch einen einzigen Winkel α um den Normalenvektor der Ebene definiert ist, auf der das Objekt steht. Dieser Winkel ist innerhalb jeder Kategorie konsistent annotiert. Allerdings gibt es keine Konsistenz der Annotationen über Kategoriegrenzen hinweg.

Statt α direkt zu regressieren, wird eine Hierarchie für die Posenschätzung (siehe Abbildung 5.1) konstruiert, um die Diskontinuität bei $\alpha = 0^\circ = 360^\circ$ zu vermeiden, die bei der Regression Probleme bereitet. Zuerst sagt eine lineare SVM ein Intervall voraus, in dem α liegt. In den Experimenten lieferte eine Aufteilung in vier Winkelintervalle die besten Resultate. Für jedes Intervall wird dann ein RBF-Kernel Supportvektor-Regressor trainiert, um α vorherzusagen. Durch eine geeignete Verschiebung des Intervalls kann so die Diskontinuität vermieden werden. Während dem Training werden auch Beispiele aus den benachbarten Winkelintervallen verwendet, um die Regression robust gegenüber Fehlklassifizierungen auf der Intervallebene zu machen.

Dieser zweistufige Regressor wird für jede Instanz trainiert. Außerdem wird noch ein Regressor für jede Kategorie trainiert, um auch ohne Instanzidentifikation eine Posenschätzung durchführen zu können. Diese Möglichkeit wird zwar durch den Datensatz unterstützt, wird aber von anderen Arbeiten nicht genutzt. Ungeachtet dessen ist sie aber sehr wichtig z.B. in Robotik-Anwendungen in Haushaltsumgebungen, da hier jederzeit unbekannte Instanzen vorkommen können.

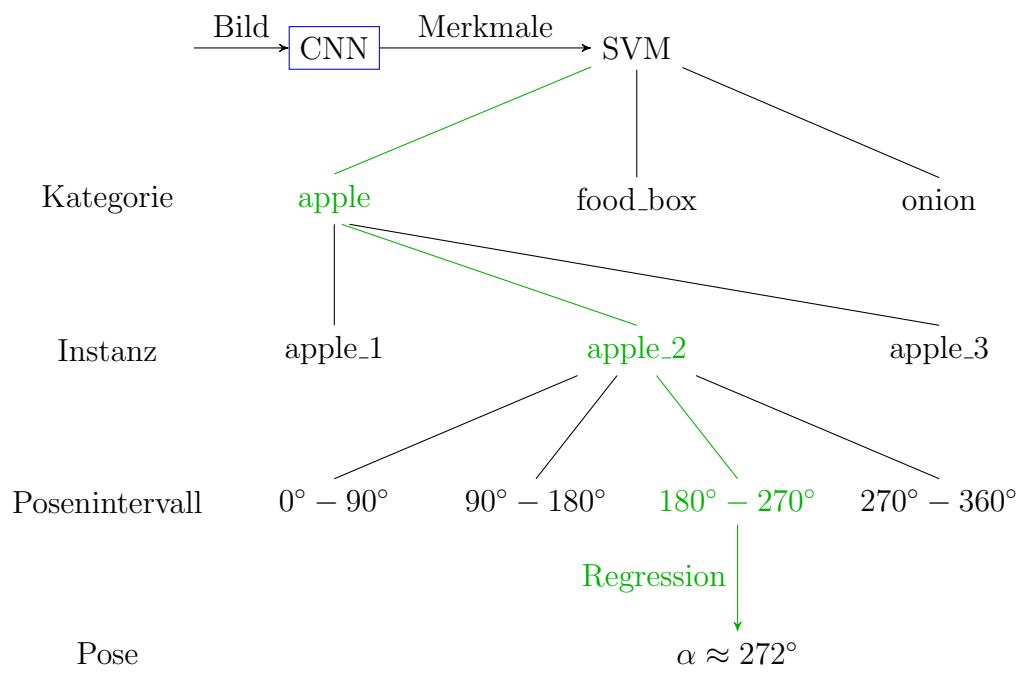


Abbildung 5.1.: Hierarchische Klassifikation mit Hilfe von SVMs und Posen-schätzung auf Instanzebene durch Supportvektorregression.

6. Evaluation

Der vorgestellte Ansatz wird auf dem Washington RGB-D Objects Dataset (Lai u. a., 2011a) evaluiert. Der Datensatz enthält 300 Objekte, die in 51 Kategorien organisiert sind. Für jedes Objekt gibt es drei Drehtellervideos, die in verschiedenen Winkeln zum Horizont aufgenommen wurden (30° , 45° und 60°). Die Sequenzen wurden mit einer *ASUS Xtion Pro Live*-Kamera mit 640×480 Pixeln Auflösung in RGB- und Tiefenkanal aufgenommen. Der Datensatz enthält auch ungefähre Annotationen für den Rotationswinkel des Drehtellers. Abbildung 6.1 zeigt beispielhaft die RGB-Sequenzen des Datensatzes für ein Objekt.

Weiterhin stellt der Datensatz auch eine Objektsegmentierung basierend auf Tiefe und Farbe bereit. Diese Segmentierungsmaske wird in der Vorverarbeitung benutzt (siehe Abschnitt 4.1). Da die RGB-Vorverarbeitung Hintergrundpixel für das glatte Ausblenden des Hintergrundes benötigt, konnten allerdings nicht die bereitgestellten vormaskierten Evaluationsbilder benutzt werden. Stattdessen wurden die korrespondierenden Bilder aus dem vollen Datensatz entnommen. Da die Methode fast den gesamten Hintergrund entfernt, sind eventuell verbliebene Merkmale nah am Objekt. Dies kann die Drehtelleroberfläche sein, die für die Klassifizierung nicht interessant ist, da sie in allen Bildern vorkommt. Die andere Möglichkeit sind die Markierungen auf dem Drehteller (vgl. Abbildung 6.1), die allerdings zur Posenschätzung auch nicht hilfreich sind, da die Objekte in jeder Aufnahme zufällig auf dem Drehteller platziert werden. Insbesondere wird in der Evaluation der Posenschätzung die 45° -Sequenz gänzlich zum Testen zurück gehalten (s.u.), sodass die Markerposition zur Testzeit völlig neu für das System ist. Daher ist der vorgestellte Ansatz mit den anderen Arbeiten auf dem Datensatz vergleichbar.

6.1. Evaluationsverfahren

Die Evaluation folgt dem von Bo u. a. (2013) und Lai u. a. (2011a) etablierten Protokoll. Es wird jedes fünfte Bild für Training und Evaluation genutzt. Für die Kategorie-Erkennung wird die durchschnittliche Korrektklassifikationsrate über zehn vordefinierte Training/Test-Aufteilungen auf dem Datensatz berechnet. Dabei sind die Test-Instanzen in jeder Aufteilung dem System komplett unbekannt.

6. Evaluation



Abbildung 6.1.: RGB-Beispielsequenzen aus dem Washington RGB-D Objects Dataset. Von oben nach unten: 60°-Sequenz, 45°-Sequenz, 30°-Sequenz. Die Spalten zeigen von links nach rechts Bilder mit aufsteigenden Winkeln α .

Für die Instanzerkennung und Posenschätzung wird das *Leave-Sequence-Out*-Schema von Bo u. a. (2013) benutzt, bei dem das System auf den 30°- und 60°-Aufnahmen trainiert wird und die 45°-Aufnahme zur Evaluation verwendet wird.

6.2. Ergebnisse

Als Vergleich der Ergebnisse auf dem Washington-RGB-D-Objects-Datensatz werden zwei anderen Arbeiten (Bo u. a., 2013; Lai u. a., 2011a) herangezogen. Leider gibt es für das aktuell beste Verfahren aus Bo u. a. (2013) keine Lernkurven für verkleinerte Datensatzgrößen. Da es weder gelang, mit Hilfe des bereitgestellten Quellcodes die Ergebnisse zu reproduzieren, noch die Autoren zu kontaktieren, wird noch ein Standard-Ansatz betrachtet, der auf der Bag-Of-Visual-Words-Technik basiert: PHOW-Merkmale (Dense-SIFT-Merkmale auf verschiedenen Skalen, siehe Bosch u. a. (2007)) werden aus den vorverarbeiteten Bildern extrahiert und ein Wörterbuch mit Hilfe von K-Means-Clusteranalyse gelernt. Die Wörter eines Bildes werden in 2×2 - und 4×4 -Gittern in Histogrammen aggregiert und bilden so Merkmalsvektoren. Es kommt die PHOW- und K-Means-Implementierung aus der vlfeat-Bibliothek¹ zum Einsatz. Alle Hyperparameter des Verfahrens (Gittergrößen, SIFT-Skalen) wurden auf den Standardeinstellungen belassen, die sehr gute Ergebnisse auf dem Caltech-101-Datensatz² liefern. Als Lernverfahren kommt

¹<http://www.vlfeat.org/>

²http://www.vision.caltech.edu/Image_Datasets/Caltech101/

das beschriebene hierarchische SVM-Verfahren zum Einsatz.

Ohne überwachtes Lernen zu verwenden, können die Merkmale, die vom CNN erzeugt wurden, in \mathbb{R}^2 mit Hilfe einer t-SNE-Einbettung (Van der Maaten und Hinton, 2008) visualisiert werden. Diese Einbettung versucht, lokale und globale Strukturen des hochdimensionalen Merkmalsraumes im \mathbb{R}^2 zu erhalten und erlaubt so, die Güte der Struktur im Merkmalsraumes zu beurteilen. Während Abbildung 6.2 (oben) zeigt, dass die Objektklassen im Merkmalsraum gut separiert sind, demonstriert Abbildung 6.3, dass Objektinstanzen aus der selben Kategorie ebenfalls gut getrennt werden. Weiterhin bleiben ähnliche Posen des selben Objektes nah beieinander im Merkmalsraum und stellen eine niederdimensionale Mannigfaltigkeit dar. Diese Eigenschaften der unüberwacht gelernten Merkmalsabbildung sind sehr günstig und erlauben das Lernen bereits mit wenigen Trainingsbeispielen.

Abbildung 6.2 (unten) zeigt eine Visualisierung der PHOW-Merkmale. Im direkten Vergleich fällt auf, dass die CNN-Merkmale die Kategorien im Merkmalsraum deutlich besser separieren. Im Anhang A sind auch Einbettungen für einzelne Kategorien beigefügt, die zeigen, dass die Instanzen in t-SNE-Einbettungen des PHOW-Merkmalsraumes oft nicht gut trennbar sind. Diese qualitativen Eigenschaften der Merkmalsräume werden sich auch in quantitativ besserer Performanz des CNN-Ansatzes im Vergleich zum PHOW-Ansatzes niederschlagen. Im Anhang A sind größere Ansichten und t-SNE-Einbettungen für weitere Kategorien beigefügt.

Tabelle 6.1 zeigt die quantitativen Ergebnisse des CNN-Ansatzes im Vergleich mit den anderen Ansätzen. Die aktuellen Ergebnisse in der Kategorie- und Instanzschätzung für RGB- und RGB-D-Daten werden verbessert. Die Ausnahme bildet die RGB-basierte Instanzerkennung, in der der HMP-Ansatz von Bo u. a. mit 0,1% Vorsprung gewinnt. Weiterhin kann ein Vorteil der Reprojektion und Einfärbung der Tiefendaten für die Instanzklassifizierung nachgewiesen werden.

Die Analyse der Konfusionsmatrix (Abbildung 6.5) ergibt, dass die Kategorie-Klassifizierung nur wenige systematische Fehler macht. Manche Objektkategorien erweisen sich als sehr schwer klassifizierbar, da sie entweder stark variierende Instanzen mit nur wenig Beispielen (z.B. *mushroom*) enthalten, oder Objekte enthalten, die eine hohe Ähnlichkeit zu Instanzen anderer Kategorien (z.B. *pitcher* und *coffe_mug*) aufweisen (vgl. Abbildung 6.4).

Wenn das Evaluationsprotokoll für Instanzerkennung und Posenschätzung benutzt wird (d.h. alle Instanzen bekannt sind, aber die 45°-Sequenz gänzlich unbekannt ist), erreicht die Kategorieerkennung basierend auf CNN-Merkmalen eine nahezu perfekte Genauigkeit von 99,6%. Dies legt den Schluss nahe, dass das Verfahren sehr robust gegenüber Perspektivwechseln ist, und die Schwierigkeit der Instanz-Klassifikation hauptsächlich in der Varianz der Instanzen innerhalb einer Kategorie begründet ist.

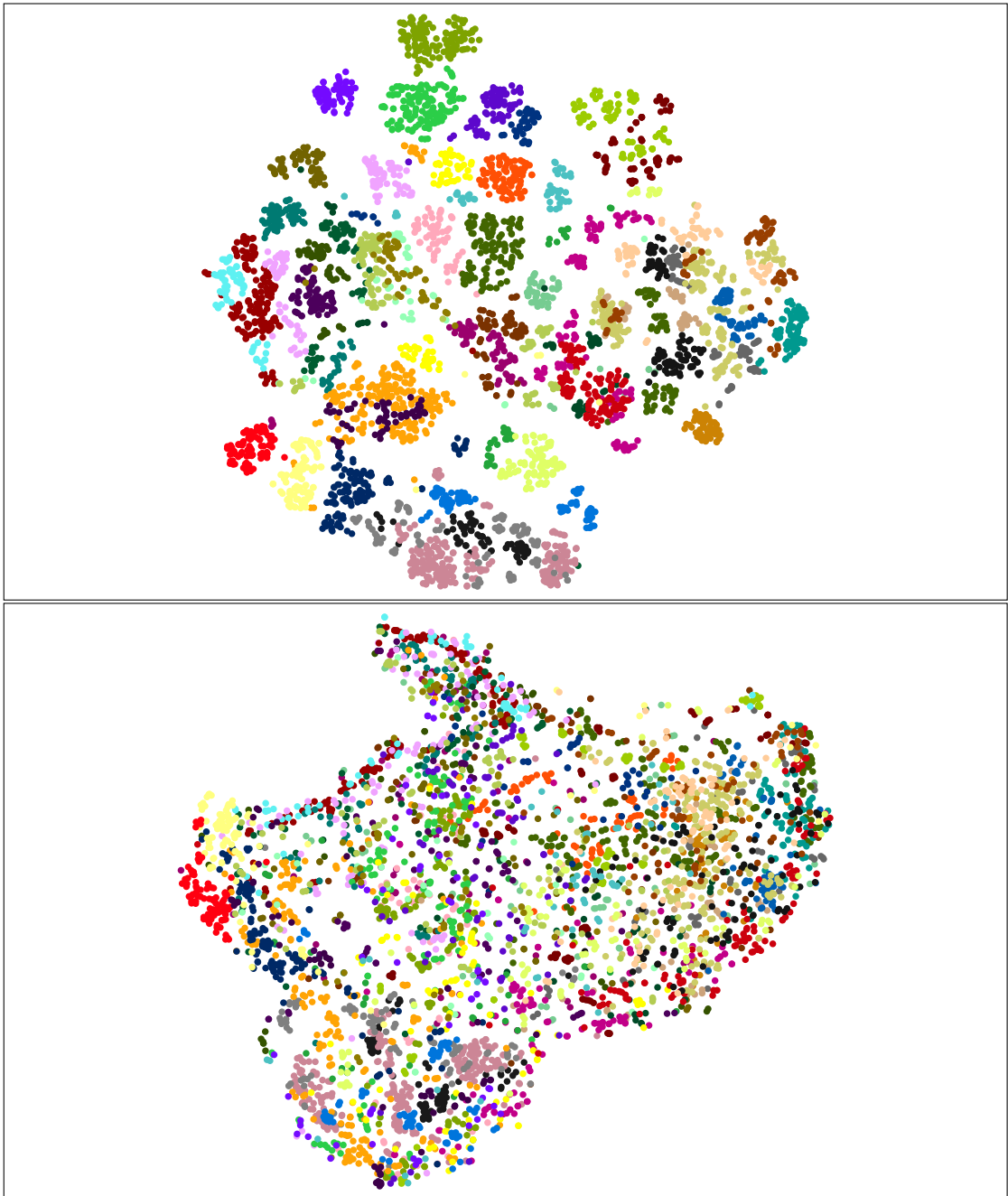


Abbildung 6.2.: t-SNE-Einbettung von 1/10 der Bilder des Washington-RGB-D-Datensatzes, eingefärbt nach Objektkategorie. Oben: CNN-Merkmale. Unten: Merkmale des PHOW-basierten Vergleichsansatzes. Für größere Versionen und weitere Klassen siehe Anhang A.

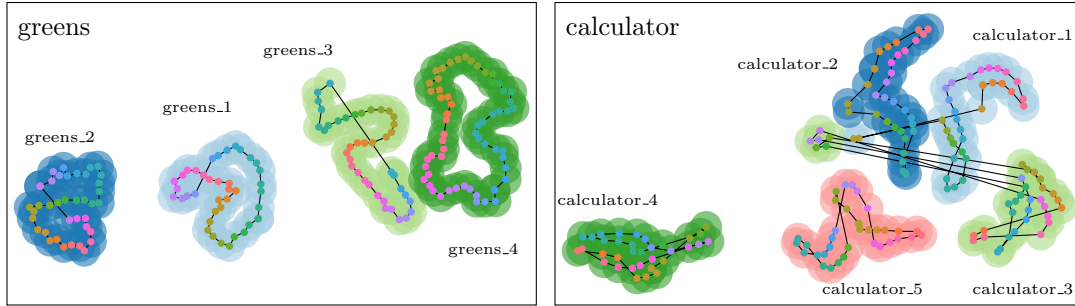


Abbildung 6.3.: Separate t-SNE-Einbettung der CNN-Merkmale (nur 45°-Sequenzen) für die Kategorien *greens* und *calculator*, eingefärbt und verbunden nach Objektpose. Die Punkte einer Instanz besitzen die gleiche Hintergrundfarbe.

Tabelle 6.1.: Vergleich der Korrektklassifikationsrate auf Kategorie- und Instanzebene auf dem Washington RGB-D-Objects-Datensatz. Für die Tiefenvorverarbeitung werden verschiedene Varianten untersucht (keine Vorverarbeitung, Reprojektion, Reprojektion und Einfärbung). Die Zeile **CNN** resultiert aus den besten Ergebnissen über die Varianten.

| Methode | Kategorie (%) | | Instanz (%) | |
|-------------------|-------------------|-------------------|-------------|-------------|
| | RGB | RGB-D | RGB | RGB-D |
| Lai u. a. (2011a) | 74,3 ± 3,3 | 81,9 ± 2,8 | 59,3 | 73,9 |
| Bo u. a. (2013) | 82,4 ± 3,1 | 87,5 ± 2,9 | 92,1 | 92,8 |
| PHOW | 80,2 ± 1,8 | — | 62,8 | — |
| CNN | 83,1 ± 2,0 | 89,4 ± 1,3 | 92,0 | 94,1 |
| keine Vorv. | — | 89,4 ± 1,3 | — | 92,6 |
| Einfärbung | — | 88,9 ± 1,1 | — | 93,2 |
| Reproj. + Einf. | — | 88,3 ± 1,5 | — | 94,1 |

Tabelle 6.2.: Median- (Med) und Durchschnittsfehler (Ave) der Posenschätzung auf dem RGB-D Objects Dataset. Falsche Klassifizierungen werden mit 180° Fehler belegt. (C) und (I) sind Teilmengen des Datensatzes, auf denen das System die Kategorie bzw. Instanz korrekt vorhersagt. Für den CNN-Ansatz ist die Posenschätzung auf Instanz- und Kategorie-Ebene einzeln aufgeführt.

| Methode | Winkelfehler (°) | | | | | |
|-------------------|------------------|-------------|-------------|-------------|-------------|-------------|
| | Med | Med(C) | Med(I) | Ave | Ave(C) | Ave(I) |
| Lai u. a. (2011b) | 62,6 | 51,5 | 30,2 | 83,7 | 77,7 | 57,1 |
| Bo u. a. (2013) | 20,0 | 18,7 | 18,0 | 53,6 | 47,5 | 44,8 |
| CNN – Instanz | 20,4 | 20,4 | 18,7 | 51,0 | 50,4 | 42,8 |
| CNN – Kategorie | 19,2 | 19,1 | 18,9 | 45,0 | 44,5 | 43,7 |

6. Evaluation



Abbildung 6.4.: Beispiele für leicht verwechselbare Kategorien. Gezeigt werden Objekte, die große Ähnlichkeit zu Objekten anderer Kategorien aufweisen.

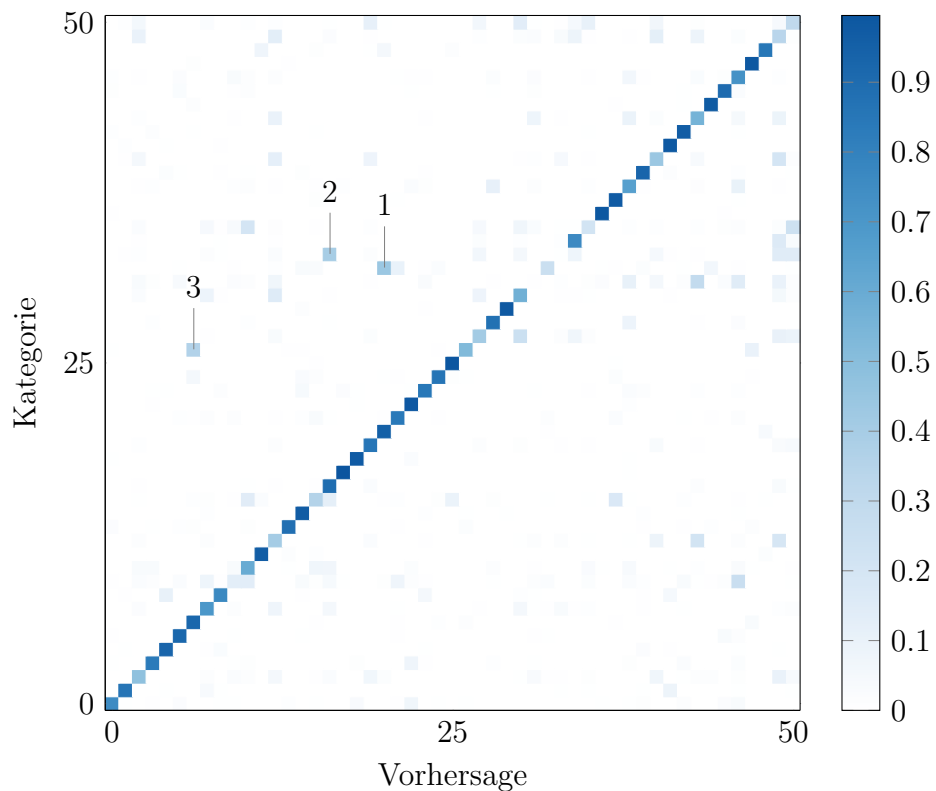


Abbildung 6.5.: Konfusionsmatrix für die Kategorie-Erkennung, normalisiert durch die Anzahl Beispiele für jede Annotation. Markierte Ausreißer: 1) *pitcher* erkannt als *coffee_mug*, 2) *peach* als *sponge*, 3) *keyboard* als *food_bag*.

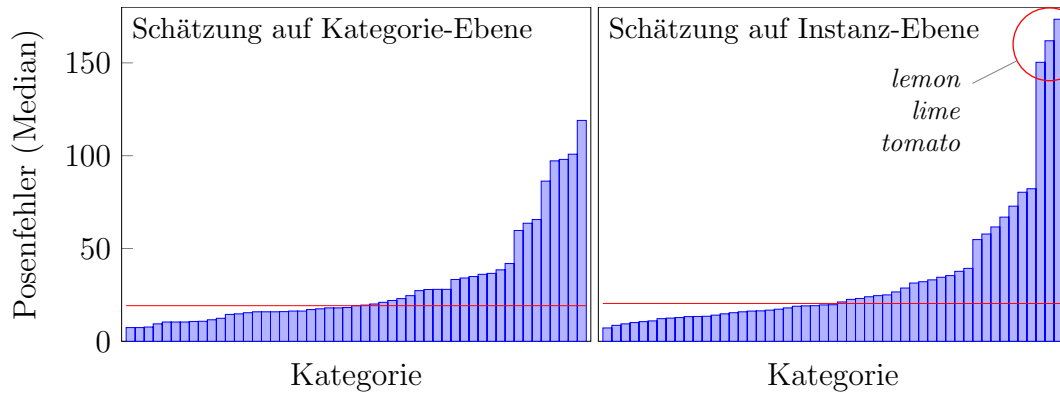


Abbildung 6.6.: Verteilung der Medianfehler in der Posenschätzung über die Kategorien. Die Posenschätzung wurde links auf Kategorie-Ebene durchgeführt, rechts auf Instanz-Ebene. Die Fehler wurden jeweils nach ihrem Betrag sortiert. Der Median über alle Kategorien ist als horizontale Linie (rot) eingezeichnet.

Auch die Ergebnisse in der Posenschätzung werden durch den vorgestellten Ansatz leicht verbessert (siehe Tabelle 6.2). Der Fehler in der Posenschätzung wird auf Instanz- und Kategorieebene gemessen. „MedPose“ beschreibt nach Bo u. a. (2013) den Median des Posenfählers über alle Testbilder, mit 180° Fehler falls die Klasse oder Instanz des Objektes nicht korrekt identifiziert wurde. „MedPose(C)“ ist der Median des Posenfählers für alle Beispiele mit korrekter Klassifizierung auf Kategorieebene. Schließlich umfasst „MedPose(I)“ nur die Fälle, in denen Klasse und Instanz korrekt bestimmt wurden. Analog beschreiben „AvePose“, „AvePose(C)“ und „AvePose(I)“ den durchschnittlichen Fehler in jedem Fall. Man beachte, dass der durchschnittliche Fehler signifikant kleiner ist als der der anderen Arbeiten.

Überraschenderweise zeigt die CNN-basierte Posenschätzung auf Kategorieebene noch bessere Ergebnisse als die Posenschätzung auf Instanzebene und erzielt sogar niedrigere Fehlerraten als der Ansatz von Bo u. a. (2013). Die Posenschätzung auf Kategorieebene muss sich nur in den „MedPose(I)“- und „AvePose(I)“-Sparten geschlagen geben, in denen ihr breiteres Wissen nicht so nützlich ist wie genaue Anpassung an die jeweilige Instanz. Die guten Ergebnisse zeigen, dass die Posenschätzung stark von der Generalisierung über Instanzen hinweg profitieren kann, die das Training auf Kategorieebene erlaubt.

Die Verteilung des Posenfählers über die Kategorien (Abbildung 6.6) zeigt, dass es einige Kategorien gibt, die besonders schwierig zu schätzen sind, zum Beispiel *lemon*, *lime* und *tomato*, die alle hochgradig rotations-symmetrisch sind und somit wenig Anhaltspunkte für eine Posenschätzung bieten.

Die Performanz des CNN-Ansatzes sinkt nur langsam, wenn die Anzahl der Trainingsbeispiele verringert wird (siehe Abbildung 6.7). In diesem Experiment wird

6. Evaluation

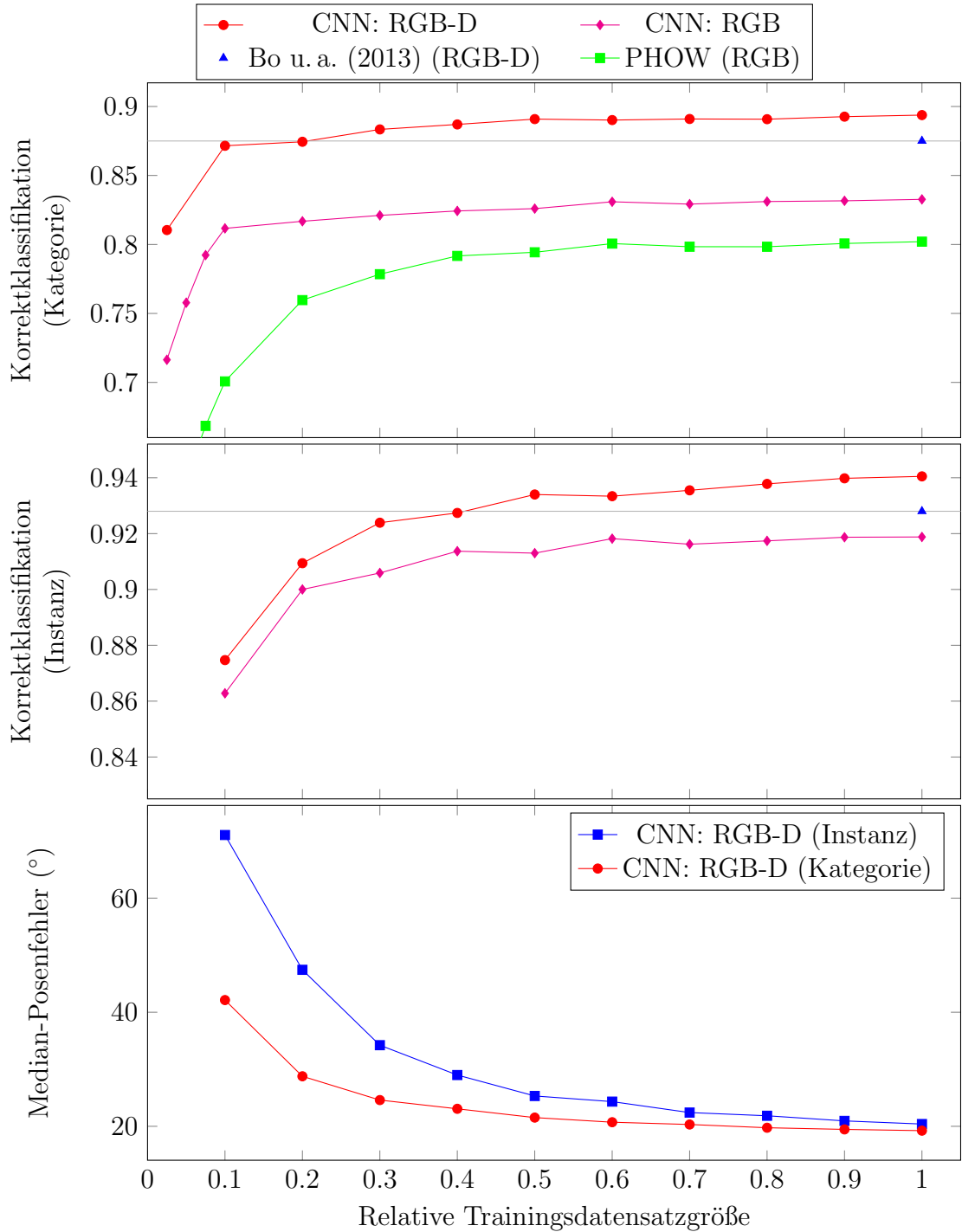


Abbildung 6.7.: Lernkurven für die Klassifikation (oben und Mitte) und Posen-schätzung (unten). Wie zuvor wird die durchschnittliche Genauigkeit aus der Kreuzvalidierung für die Kategorieerkennung und Genauigkeit auf der 45° -Sequenz für die Instanzerkennung angegeben.

Tabelle 6.3.: Evaluation verschiedener Farbpaletten für die Einfärbung der Tiefendaten mit Korrektorklassifikationsrate in der Instanzerkennung.




| Palette | Farben | Korrektklassifikation (%) | |
|--------------------|---|---------------------------|-------------|
| | | Tiefe | RGB-D |
| Grauwert |  | 41,8 | 93,1 |
| Grün |  | 38,8 | 93,3 |
| Grün-Rot-Blau-Gelb |  | 45,5 | 94,1 |

Tabelle 6.4.: Laufzeiten verschiedener Algorithmus-Teile pro Eingabebild in Sekunden. Die Laufzeiten wurden alle auf einem Intel Core i7-4800MQ @ 2,7 GHz und einer Mobil-Grafikkarte (NVidia GeForce GT 730M) für CUDA-Berechnungen gemessen. Sie beinhalten alle Vorverarbeitungsschritte.

| Schritt | CNN | PHOW | Bo u. a. (2013) |
|----------------------------|-------|--------------|-----------------|
| Merkmalsextraktion (RGB) | 0,013 | 0,161 | 0,294 |
| Merkmalsextraktion (Tiefe) | 0,173 | — | 0,859 |
| Insgesamt | 0,186 | 0,161 | 1,153 |

der Datensatz durch schichtenweise Stichprobenentnahme auf Kategorie- bzw. Instanzebene reduziert. Dadurch wird sichergestellt, dass sich die Verteilung der Beispiele über die Kategorien bzw. Instanzen nicht verändert. Mit nur 30% der Trainingsdaten verringert sich die Korrektorklassifikationsrate der Kategorisierung nur ungefähr um einen Prozentpunkt, während die Rate der Instanzerkennung nur um 2% sinkt. Diese Ergebnisse stützen die Annahme, dass die CNN-Merkmale die Kategorien und Instanzen des RGB-D-Objects-Datensatzes auf eine semantisch nützliche Art separieren. Im Vergleich mit der PHOW-Implementierung sinkt die Kategorie-Korrektklassifikationsrate langsamer bei verkleinertem Datensatz. Die PHOW-Implementierung zeigt sehr schlechte Ergebnisse (unter 70%) bei der Instanzerkennung, sodass man sie ohne weitere Optimierung hier nicht zum Vergleich heranziehen sollte.

Die verwendete Farbpalette (Grün, Rot, Blau, Gelb) wurde ebenfalls evaluiert. Tabelle 6.3 zeigt Korrektorklassifikationsraten für einige ausgewählte Paletten. Die Ergebnisse zeigen, dass das CNN besser auf farbige Kodierungen reagiert als auf monochromatische Kodierungen.

Da die Rechenleistung in Robotikanwendungen typischerweise sehr beschränkt ist, wurden die Laufzeiten für die Merkmalsextraktion auf einem Notebook gemessen. Dieses besitzt einen Mobilprozessor (Intel Core i7-4800MQ @ 2,7 Ghz) und

6. Evaluation

eine Mobilgrafikkarte (NVidia GeForce GT 730M), die für CUDA-Berechnungen benutzt wird. Wie in Tabelle 6.4 ersichtlich ist, wird die Laufzeit des Ansatzes dominiert von der Vorverarbeitung für Tiefendaten, die allerdings noch Optimierungspotential besitzt. Trotzdem sind die Laufzeiten klein genug, um Bildraten von bis zu 5 Hz in einer Echtzeitanwendung zu erlauben. Die angegebenen Laufzeiten des Ansatzes von Bo u. a. (2013) beziehen sich auf den veröffentlichten Quellcode, mit dem die in ihrer Arbeit berichteten Ergebnisse nicht reproduzierbar waren.

7. Zusammenfassung und Ausblick

In dieser Arbeit wurde ein Ansatz vorgestellt, der es erlaubt, Objekte auf planaren Oberflächen zu kategorisieren, Instanzen zu identifizieren und ihre Pose zu schätzen. Statt Merkmale zu lernen oder manuell zu entwerfen, wurde auf ein vortrainiertes Konvolutionsnetz (CNN) gesetzt, welches auf einem großen Bildkategorisierungs-Datensatz trainiert wurde. Es wurden Tiefenmerkmale benutzt, in dem Objekte aus einer kanonischen Pose gerendert wurden und ein Einfärbungsverfahren entwickelt wurde, welches metrische Distanz vom Objektzentrum als Farbe kodiert. Der vorgestellte Ansatz wurde auf dem anspruchsvollen Washington-*RGB-D*-Datensatz evaluiert und die Trennung von Kategorien und Instanzen im Merkmalsraum jeweils gezeigt. Überwachtes Lernen auf den CNN-Merkmalen verbesserte dann die bisherigen Ergebnisse auf dem Datensatz sowohl in der Klassifikation, als auch in der Posenschätzung. Die Performanz des Ansatzes verschlechtert sich nur geringfügig, wenn der Datensatz verkleinert wird, insbesondere langsamer als der implementierte Vergleichsansatz mit *PHOW*-Merkmalen. Weiterhin erlaubt die Methode die Posenschätzung ohne Identifizierung der Instanz.

Für weitere Arbeiten bietet sich eine Verbesserung der Einfärbungsmethode an, mit dem Ziel, auch in der Kategorie-Klassifikation Vorteile gegenüber den Rohdaten zu erreichen. Dazu könnten normalisierte Farbpaletten zum Einsatz kommen, die eine Invarianz gegenüber Skalierungen besitzen, die hier (im Gegensatz zur Instanzerkennung) vorteilhaft sein könnte. Auch die Schätzung des Objektzentrums könnte in diesem Kontext verbessert werden, da sie in der aktuellen Implementierung durch Verdeckungen verfälscht werden kann. Ein größerer Schritt wäre es, den *Mesh*-Ansatz (siehe Abschnitt 4.2) weiter zu verfolgen und Meshes der Objekte aufzunehmen, um synthetisch mehr Trainingsbeispiele aus neuen Perspektiven erzeugen zu können.

Weiterhin kann der Ansatz zur Zeit nicht mit Objekten unbekannter Kategorie umgehen. Dies ist aber in vielen Kontexten notwendig, da oft nicht garantiert werden kann, dass nur bekannte Objekte präsentiert werden. Zur Detektion eines unbekanntes Objektes könnte die *SVM* auf Kategorieebene kalibriert werden, um Wahrscheinlichkeiten für die Vorhersagen zu erhalten. Bei zu kleiner Wahrscheinlichkeit könnte dann das Objekt als unbekannt deklariert werden.

A. t-SNE-Einbettungen

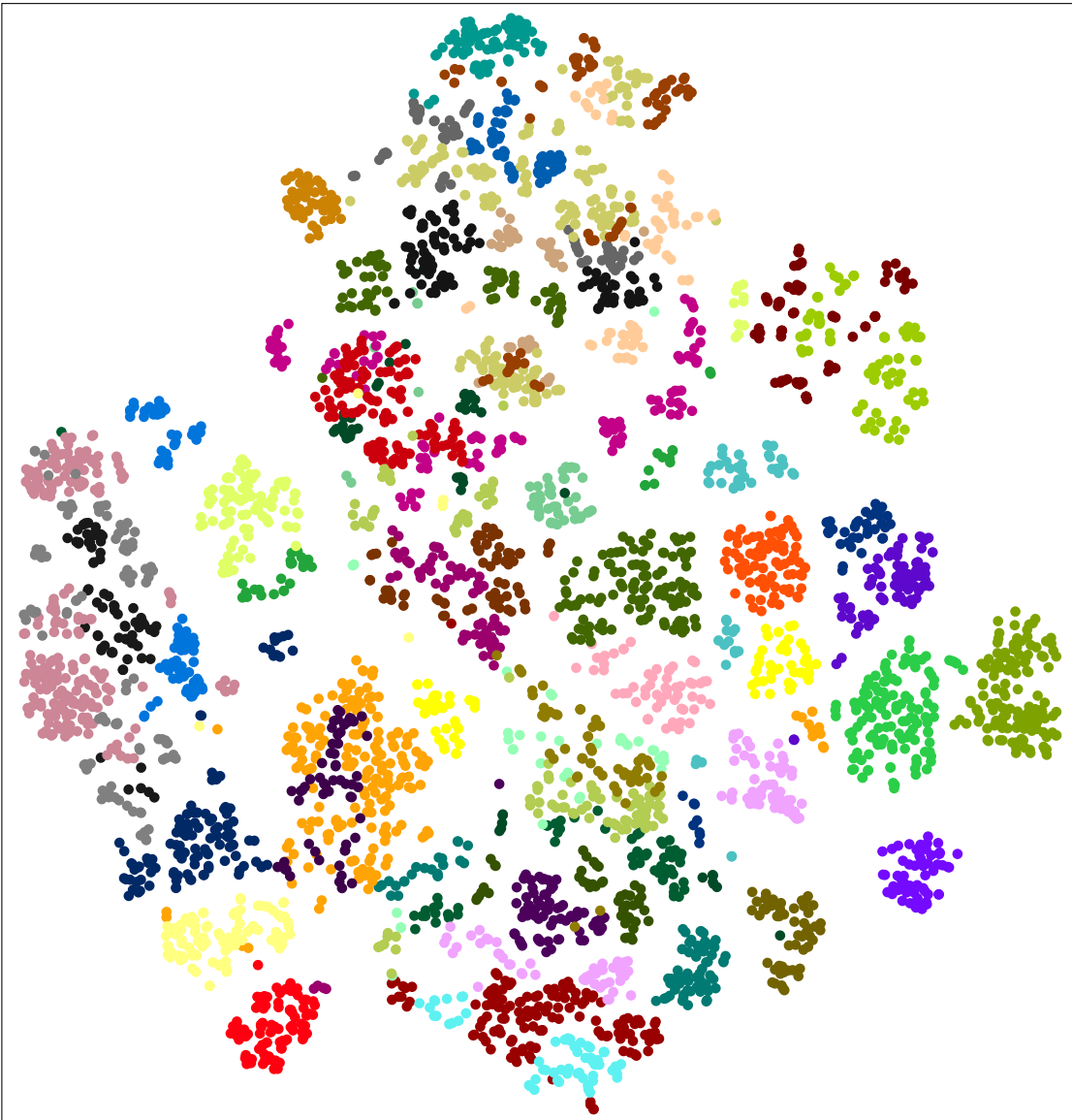


Abbildung A.1.: t-SNE-Einbettung der CNN-Merkmalvektoren von 1/10 des Datensatzes. Jeder Punkt entspricht einem Eingabebild und ist nach der Objektkategorie eingefärbt.

A. *t*-SNE-Einbettungen

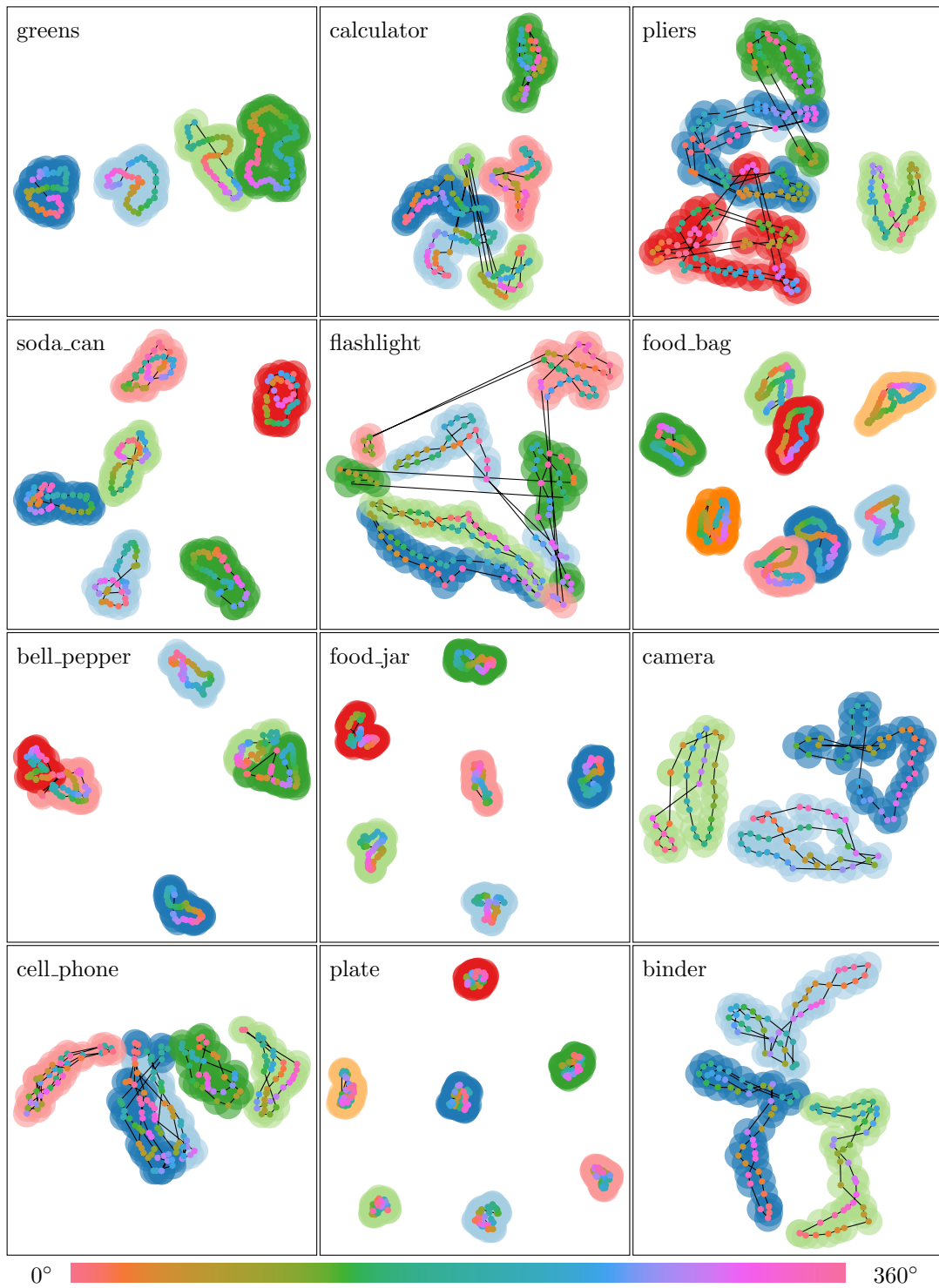


Abbildung A.2.: *t*-SNE-Einbettungen von 12 beispielhaften Kategorien im CNN-Merkmalraum. Jeder Punkt entspricht einem Eingabebild und ist nach der Objektpose eingefärbt. Weiterhin ist jeder Punkt mit einer Scheibe hinterlegt, die nach der Instanz eingefärbt ist. Die Punkte sind nach dem Posenwinkel verbunden.



Abbildung A.3.: t-SNE-Einbettung der PHOW-Merkmalvektoren von 1/10 des Datensatzes. Jeder Punkt entspricht einem Eingabebild und ist nach der Objektkategorie eingefärbt. Für eine detailliertere Beschreibung der Methode siehe Abschnitt 6.2.

A. *t*-SNE-Einbettungen

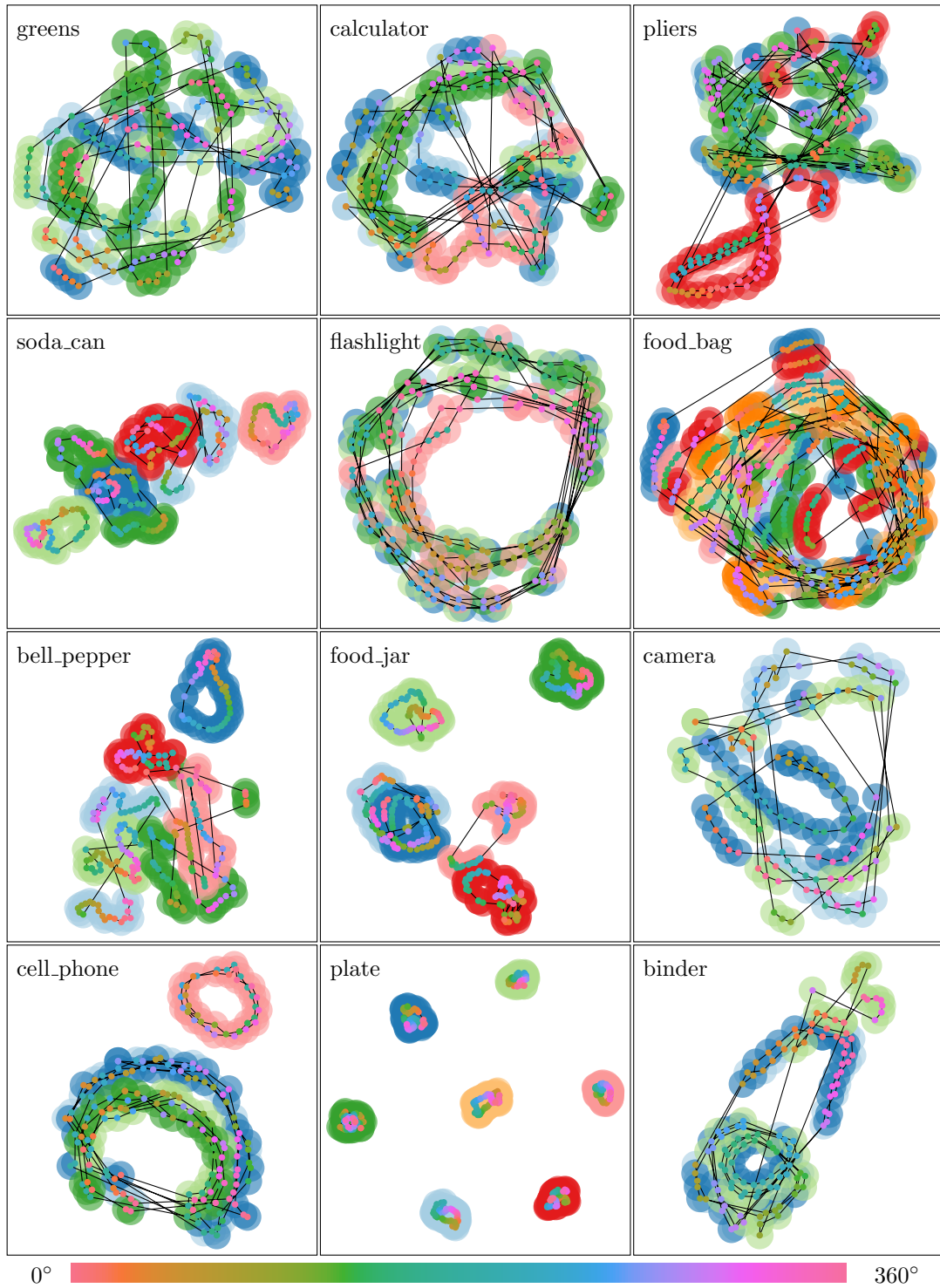


Abbildung A.4.: *t*-SNE-Einbettungen der Kategorien aus Abbildung A.2 im PHOW-Merkmalsraum. Jeder Punkt entspricht einem Eingabebild und ist nach der Objektpose eingefärbt. Jeder Punkt ist mit einer Scheibe hinterlegt, die nach der Instanz eingefärbt ist. Die Punkte sind nach dem Posenwinkel verbunden.

Literatur

- Bo, L., X. Ren und D. Fox (2013). „Unsupervised feature learning for RGB-D based object recognition“. In: *International Symposium on Experimental Robotics*, S. 387–402.
- Bosch, A., A. Zisserman und X. Munoz (2007). „Image classification using random forests and ferns“. In: *International Conference on Computer Vision*.
- Boser, B. 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*. ACM, S. 144–152.
- Cortes, C. und V. Vapnik (1995). „Support-vector networks“. In: *Machine learning* 20.3, S. 273–297.
- Donahue, J., Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng und T. Darrell (2014). „DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition“. In: *Proc. Int. Conf. on Machine Learning (ICML)*, S. 647–655.
- Fukushima, K. (1980). „Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position“. In: *Biological cybernetics* 36.4, S. 193–202.
- Girshick, R., J. Donahue, T. Darrell und J. Malik (2014). *Rich feature hierarchies for accurate object detection and semantic segmentation*. arXiv: 1311.2524.
- Holz, D., S. Holzer, R. B. Rusu und S. Behnke (2012). „Real-time plane segmentation using RGB-D cameras“. In: *RoboCup 2011: Robot Soccer World Cup XV*, S. 306–317.
- Hsu, C.-W. und C.-J. Lin (2002). „A comparison of methods for multiclass support vector machines“. In: *Neural Networks, IEEE Transactions on* 13.2, S. 415–425.
- Krizhevsky, A., I. Sutskever und G. E. Hinton (2012). „Imagenet classification with deep convolutional neural networks“. In: *Advances in Neural Information Processing Systems (NIPS)*, S. 1097–1105.
- Lai, K., L. Bo, X. Ren und D. Fox (2011a). „A large-scale hierarchical multi-view RGB-D object dataset“. In: *International Conference on Robotics and Automation (ICRA)*, S. 1817–1824.
- (2011b). „A Scalable Tree-Based Approach for Joint Object and Pose Recognition“. In: *Proceedings of the Conference on Artificial Intelligence (AAAI)*.
- LeCun, Y., B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard und L. D. Jackel (1989). „Backpropagation applied to handwritten zip code recognition“. In: *Neural computation* 1.4, S. 541–551.

Literatur

- LeCun, Y., L. Bottou, Y. Bengio und P. Haffner (1998). „Gradient-based learning applied to document recognition“. In: *Proceedings of the IEEE* 86.11, S. 2278–2324.
- Levin, A., D. Lischinski und Y. Weiss (2004). „Colorization using optimization“. In: *Transactions on Graphics (TOG)*. Bd. 23. 3, S. 689–694.
- Razavian, A. S., H. Azizpour, J. Sullivan und S. Carlsson (2014). „CNN Features off-the-shelf: an Astounding Baseline for Recognition“. In: *CVPR DeepVision Workshop*.
- Russakovsky, O., J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein u. a. (2014). *ImageNet Large Scale Visual Recognition Challenge*. arXiv: 1311.2524.
- Shotton, J., T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook und R. Moore (2013). „Real-time human pose recognition in parts from single depth images“. In: *Communications of the ACM* 56.1, S. 116–124.
- Smola, A. J. und B. Schölkopf (2004). „A tutorial on support vector regression“. In: *Statistics and computing* 14.3, S. 199–222.
- Van der Maaten, L. und G. Hinton (2008). „Visualizing data using t-SNE“. In: *Journal of Machine Learning Research* 9, S. 2579–2605.
- Zeiler, M. D. und R. Fergus (2014). „Visualizing and understanding convolutional networks“. In: *European Conference on Computer Vision (ECCV)*, S. 818–833.