04: Decomposing a Scene into Geometric and Semantically Consistent Regions

Benito van der Zander

9. September 2010

Zusammenfassung

Diese Arbeit behandelt das Problem des algorithmischen Verstehens eines einzelnen gegebenen Bildes, bei dem jeder Pixel einer semantischen und geometrischen Klasse zugewiesen werden soll, die erklärt, was dieser Pixel darstellt. Dazu wird ein Verfahren von Gould et al. [6] beschrieben und diskutiert, das mittels einer gegebenen Datenbank von Trainingsbildern die Beziehungen zwischen Objektregionen erlernt und das Bild in entsprechend klassifizierte Regionen unterteilt. Das gelernte Modell wird dabei als Energiefunktion beschrieben, welche die Ergebnisse mehrere Klassifikatoren zusammenfasst und bei der Segmentierung durch verschiedene Änderung der Zuordnung zwischen Pixeln und Regionen optimiert wird. Zudem wird erklärt, wie aus dieser Segmentierung ein scheinbar realistisches 3D-Modell der Szene erzeugt werden kann.

Inhaltsverzeichnis

1 Einführung

Die Segmentierung in konsistente und sinnvolle Regionen behandelt ein Problem, dessen umgangssprachliche Formulierung für die meisten Leute trivial klingt. Es ist ein bestimmtes Bild gegeben, und man stellt die einfache Frage: Was ist darauf zu sehen?

Ein Mensch, der es betrachtet, erkennt sofort, wo im Bild, welche Objekte dargestellt sind; ein Computer dagegen ist ratlos. Frühere Versuche, diese Frage zu beantworten, haben zu den Gebieten der Objekterkennung und Segmentierung geführt. Aber eine reine Objekterkennung verläuft üblicherweise lokal: Ein bestimmtes Objekt oder eine bestimmte Objektklasse wird im Bild gesucht, und alle Stellen markiert, an denen es sich befinden könnte. Dies führt jedoch prinzipiell nur zu löchrigen Informationen über ein Bild und kann nicht als wirkliches Verstehen betrachten werden, da der Großteil des Bildes, der das gesuchte Objekt nicht enthält, als unbekannt oder ≫da ist nichts≪-markiert wird, obgleich natürlich auf jeden Pixel etwas abgebildet wurde. Auch die Kombination mehrerer, separater Objektdetektoren löst das Problem nicht wirklich, da stets nichtklassifizierte Pixel übrigbleiben. In gewisser Weise verschärft eine solche Kombination das Problem sogar, da ein einzelner Pixel dabei zu mehreren Objekten gezählt werden kann. Zudem ist es häufig nicht möglich Objekte lokal zu erkennen, ohne bereits Wissen über die dargestellte Szene zu besitzen. Zum Beispiel kann eine runder Fleck weißer Pixel auf einem Tisch durchaus eine liegende Untertasse sein, am Himmel dagegen handelt es sich mit größerer Wahrscheinlichkeit um eine Wolke.

Dagegen verstehen traditionelle Segmentierungsverfahren, obgleich sie das gesamte Bild betrachten und keine Pixel ignorieren, überhaupt nichts von der Szene. da sie nur ähnlich aussehende Pixel zusammenfassen. Trotzdem sind diese Verfahren nützlich, denn um die gesamte Semantik eines Bildes zu verstehen, benötigt man größere Regionen, in denen man Objekte erkennen kann, da einzelne Pixel mehrdeutig sind.

Ein Algorithmus, der die eingangs gestellte Frage beantworten will, muss also eine Kombination von Segmentierung und Objekterkennung darstellen, die das Bild in Regionen unterteilt und für jede Region erkennt, welches Objekt sie enthält.

Eine solche, rein semantische Klassifizierung reicht jedoch im Grunde noch nicht aus, um eine Szene wirklich zu verstehen, da man dadurch nichts über die Anordnung der Objekte gelernt hat. Es wäre daher wünschenswert, wenn ein solcher Algorithmus jeder Region auch eine geometrische Klasse zuweist, die angibt, wie das Objekt geformt oder ausgerichtet ist. Es ist zu erwarten, dass ein solches geometrisches Wissen auch die Genauigkeit der semantischen Klassifizierung verbessert, da beispielsweise zwei vertikal ausgerichtete, sich im Bild überlappende Objekte wahrscheinlich hintereinander stehen, während eine Überlappung bei einem horizontalen und einem vertikalen Objekt eher bedeutet, das letzteres auf ersterem steht.

Es ist aber schwer zu entscheiden, welche Region von Pixel ein einzelnes Objekt darstellt. Ein mögliches Objekt kann nämlich aus vielen kleinen Einzelheiten mit sehr unterschiedlichem Aussehen bestehen, andererseits kann eine kleine Gruppe von Pixeln auch ein weit entferntes Objekt darstellen. Auch Licht und Schatten können das Aussehen eines Objektes verfälscht haben, oder ein Objekt kann durch Überdeckungen nur teilweise sichtbar sein.

Es ergibt sich also eine weitere Frage: Woher kann der Algorithmus wissen, wie (bestimmte) Objekte aussehen?

Eine präzise mathematische Beschreibung eines Objekts, die für einige gegebene Pixel die exakte Wahrscheinlichkeit, ein solches Objekt zu sehen, berechnet, scheint nicht möglich zu sein; und wenn doch, ist sie eher unpraktisch, vor allem wenn man weitere Objektklassen hinzufügen will. Stattdessen muss das Verfahren in der Lage sein, aus mehreren Trainingsbilder, in denen die entsprechenden Objektklassen markiert sind, selbstständig zu lernen, wie die Objekte aussehen müssen.

Das Problem lässt sich also folgendermaßen zusammenfassen: Es ist eine Menge von Bildern gegeben, in denen jeder Pixel gemäß dem menschlichen Szenenverständnis mit einer semantischen und geometrischen Klasse markiert ist. Diese Trainingsbilder müssen nun von einem Algorithmus analysieren werden, um zu lernen, wie die Objekte einer Klasse aussehen. Anschließend muss der Algorithmus in einem neuen Testbild für jeden Pixel ermitteln, zu welchen beiden Klassen dieser

am wahrscheinlichsten gehört.

Die restliche Arbeit, stellt nun das von Gould et al. entwickelte Verfahren[6] zur Lösung dieses Problems vor. Im nächsten Abschnitt werden kurz ähnliche Paper aufgezählt; in Abschnitt 3 wird beschrieben, wie Gould et al. das Problem formalisieren; in Abschnitt 4, wie ihr Algorithmus die nötigen Daten erlernen kann und in Abschnitt 5, wie die gelernten Daten verwendet werden, um ein Bild zu segmentieren und als 3D-Objekt darzustellen. Schließlich wird das Verfahren in Abschnitt 6 mit anderen Verfahren verglichen und in Abschnitt 7 eine Schlussfolgerung gezogen.

2 Verwandte Literatur

Es gibt bereits einige Paper, in denen Verfahren vorgestellt werden, die einzelne Bilder global in semantisch oder geometrisch klassifizierte Regionen segmentieren. Die bisherigen Algorithmen verwenden aber üblicherweise entweder nur semantische oder nur geometrische Klassen, so dass Gould et al. die ersten sind, deren Algorithmus Zusammenhänge zwischen der Geometrie und der Semantik eines Bildes ausnutzen kann.

Eine der wichtigsten Arbeiten zur Zerlegung von Bildern in semantische Klassen stammt von Jamie Shotton [18]. Dort wird zum einen ein Textonfeaturevektor, welcher das Aussehen eines Pixels beschreibt, und zum anderen ein CRF-Modell, welches daraus für jeden Pixel eine semantische Klasse berechnet, vorgestellt. Von ihm übernimmt Gould den Textonfeaturevektor.

Christian Wojek kombiniert ein klassisches CRF-Modell, das jedem Pixel eine semantische Klasse zuweist, mit einem HOG-Objektdetektor[20], wodurch sich die Ergebnisse beider Verfahren verbessern. Damit beschäftigt er sich allerdings mit einer leicht anderen Forschungsrichtung als Gould, da sich sein Verfahren auf das praktische Erkennen bestimmter Objektklassen konzentriert, anstatt zu versuchen, alle Klassen gleich gut zu erkennen. Er arbeitet zudem bevorzugt mit Videoaufnahmen, obgleich sein Algorithmus auch auf Einzelbilder angewandt werden kann.

Eine weitere Arbeit, die semantische Klassifizierung behandelt, stammt von Gould selber[5]. Dort werden die Pixel noch nicht in geometrische Klassen unterteilt, sondern für jede semantische Klasse wird die zweidimensionale, relative Position zu anderen Klassen gelernt, wie beispielsweise dass Autos über der Straße, anstatt unter ihr fahren. Sein damaliges Verfahren zerlegt das Bild in Superpixel, erstellt dann eine erste Hypothese über die vorhandenen Klassen aus dem Aussehen der Pixel, berechnet daraus die wahrscheinlichsten Positionen für die Klassen, und daraus wiederum eine neue Klassifizierung der Superpixel.

Der geometrische Aspekt der Segmentierung wurde vor allem von Derek Hoiem [8][9][10] untersucht. Sein Algorithmus zerlegt das Bild ebenfalls in Superpixel, welche dann zu Regionen zusammengefasst werden und nach einer von drei möglichen geometrischen Klassen klassifiziert werden. Zusätzlich zu den drei Hauptklassen wird jeder Region eine von fünf geometrischen Unterklassen zugewiesen, welche, obgleich geometrische Klassen, sehr den von Gould verwendeten semantischen Klassen ähneln.

Hoiem et al. beschreiben auch ein Verfahren zur automatischen 3D-Rekonstruktion einer Szene aus der dazugehörigen geometrischen Segmentierung[7]. Dabei wird jede Region als flache Ebene im 3D-Raum betrachtet, deren Position und Orientierung aus dem im Bild sichtbaren Schnitt der Regionsebene mit einer festgelegten Grundebene berechnet werden kann. Dieses Verfahren wird mit minimalen Änderungen von Gould et al. übernommen, allerdings integrieren sie den Horizont, der der Fluchtlinie der Grundebene entspricht, in ihr Segmentierunsmodell, anstatt wie Hoiem die Horizontposition nachträglich aus parallelen Linien zu ermitteln.

3 Formalisierung des Problem

3.1 Energiefunktion

Das Ziel einer Segmentierung ist es, das Bild, eine Menge von Pixeln, in mehrere disjunkte Regionen zu zerlegen, also jeden Pixel p einer Region R_p zuzuordnen.

Um die Qualität einer solchen Segmentierung bewerten zu können, wird in [6] eine Energiefunktion konstruiert, die gegebenen Pixel- und Regionseigenschaften eine Energie zuordnet, und die ihr Minimum bei einer als optimal gelernten Segmentierung annimmt. Als einen zusätzlichen Parameter fügen sie die Position des Horizonts hinzu, der das Erkennen der dreidimensionalen Position eines Objektes im Bild vereinfachen soll.

Da die berechnete Segmentierung eine geometrisch sinnvolle Bedeutung besitzen soll, muss für jede Region ermittelt werden, was sie eigentlich darstellt. Dazu wird jeder Region r eine semantische Objektkategorie S_r und geometrische Klasse G_r zugewiesen. In [6] haben sie sich entschieden, zwischen den achten Objektkategorien Gras, Berg, Wasser, Himmel, Straße, Baum, Gebäude und Vordergrund zu unterscheiden, sowie als geometrische Klasse G_r die drei Ausrichtungen horizontal, vertikal oder Himmel zu erlauben[1].

Sie wählen diese Klassen, da sie sich nur mit Landschaft-/Außenfotos beschäftigen, wo vor allem Objekte dieser Klassen auftreten. Die Klasse Vordergrund sollte dabei allerdings besser »sonstiges« genannt werden, da sie im Grunde alle Objekte zusammenfasst, die zu keiner der anderen Klassen gehören.

Beim Horizont gehen sie davon aus, dass die Kamera nicht um die Blickachse rotiert ist, so dass der Horizont parallel zur x-Achse des Bildes ist und vollständig durch seine y-Koordinate v^{hz} beschrieben wird.

Mit den oben erwähnten Parametern, dem Bild I, dessen genauer analysierten Aussehen A, den gelernten Parametern θ und der Anzahl der Regionen K, definieren sie nun die Energiefunktion folgendermaßen:

$$
\begin{aligned}
&E(R, S, G, A, v^{hz}, K | I, \theta) = && (1)\\
&+ \theta^{horizon} \psi^{horizon}(v^{hz}) && (2)\\
&+ \theta^{region} \sum_{r} \psi_{r}^{region}(S_r, G_r, v^{hz}; A_r, P_r) && (3)\\
&+ \theta^{pair} \sum_{rs} \psi_{rs}^{pair}(S_r, G_r, S_s, G_s; A_r, P_r, A_s, P_s) && (4)\\
&+ \theta^{boundary} \sum_{pq} \psi_{pq}^{boundary}(R_p, R_q; \alpha_p, \alpha_q) && (5)
\end{aligned}
$$

Dabei sind die ψ gelernte Klassifikatoren, die θ den Klassifikatoren zugeordnete Gewichte, r, s die Indizes einer Summe über Regionen, p, q Indizes einer Summe über benachbarte Pixel und $P_r = \{p | R_p = r\}$ alle Pixel einer Region r.

3.2 Beschreibung des Aussehen

Das Aussehen A besteht aus dem Aussehen der Pixel α_p und dem Aussehen der Regionen A_r.

Für jeden Pixel p ist α_p ein Vektor, der aus der Verkettung eines 17-dimensionalen Texturvektors mit einem 11-dimensionalen Klassifizierungsergebnisvektor entsteht.

Der Farb/Texturvektor wurde von [18] übernommen, die ihn im Prinzip wiederum von [19] übernommen haben, und beschreibt die durchschnittliche Farbe um den Pixel herum sowie die Stärke vorhandener Kanten und Ecken. Konkret werden dabei drei Gaußfilter, vier Laplacefilter und vier Ableitungen des Gaußfilters auf das Bild im CIE Lab Farbmodell angewandt, und die Ergebnisse für jeden Pixel im dazugehörigen Vektor gespeichert. Jeder der drei Gaußfilter mit $\sigma \in \{1, 2, 4\}$ wird auf jeden der drei Lab-Kanäle angewandt, während die Laplacefilter mit $\sigma \in \{1, 2, 4, 8\}$ und die Ableitungen nach x und y mit $\sigma \in \{2, 4\}$ nur für den Luminanzkanal verwendet werden (siehe Abbildung 1).

Für jeden Pixel wird zudem gespeichert, wie groß aus rein lokaler Sicht die Wahrscheinlichkeit ist, dass er zu einer bestimmten Klasse gehört. Dazu ergänzen sie den Texturvektor, um die Log-

[1] Himmel ist sowohl eine semantische Klasse als auch eine geometrische Klasse, da der Himmel ausgedehnt ist und keine feste Richtung wie horizontal oder vertikal hat.

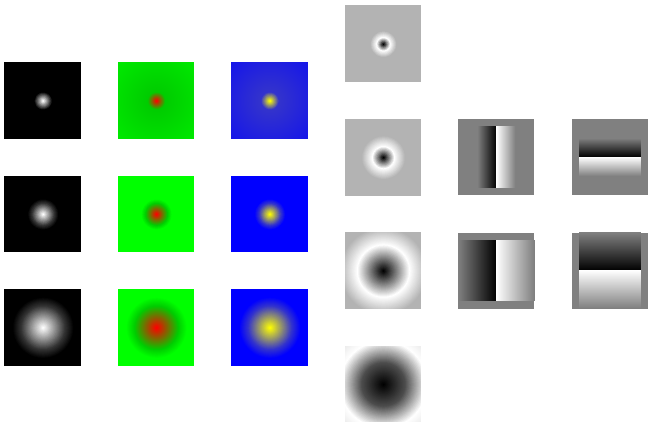


Abbildung 1: (ungefähre) Filterbank für den 17-dimensionalen Texturvektor [17]

itergebnisse von Klassifikatoren, die zuvor separat für jede der acht semantischen und drei geometrischen Klassen gelernt wurden, angewandt auf die Texturvektoren in der Umgebung von Pixel p. Konkret wird für jedes Feature im Texturvektor der Durchschnitt und die Varianz dieses Features in je einem 5x5-Fenster um den Pixel und um die acht an diesen Pixel angrenzenden Pixel zusammen mit dem Texturvektor und der y-Koordinate von Pixel p zu einem insgesamt 324-dimensionalen Vektor kombiniert, auf den dann die Klassifikatoren angewandt werden (siehe Abbildung 2). Gelernt werden diese Klassifikatoren auf den entsprechenden 324-dimensionalen Vektoren aller Pixel in allen Bilder mit dem GentleBoost-Verfahren mit zweimal verzweigenden Entscheidungsbäumen als schwachen Lernern.

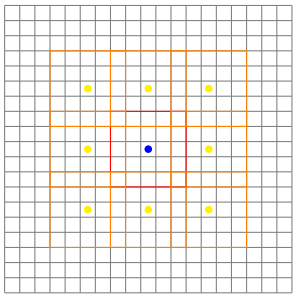


Abbildung 2: Beispiel mit übertriebenen Abständen für die Platzierung der Fenster beim Ermitteln der lokalen Fenster um einen Pixel herum.

Das elementare Aussehen einer Region modellieren sie als Gaußverteilung des Aussehen der Pixel, das heißt mathematisch: Für jede Region r ist $A_r = (\mu_r^A, \sigma_r^A)$ der Mittelwert $\mu_r^A \in \mathbb{R}^{28}$ und die Varianz $\sigma_r^A \in \mathbb{R}^{28x28}$ der mehrdimensionalen Gaußverteilung über alle Pixel P_r dieser Region.

Diese statistische Beschreibung ist robuster als das Betrachten einzelner Pixel, da sie globale Überlegungen über die Region ermöglicht. Durch das Einbinden lokaler Klassifizierungsergebnisse in den Featurevektor, verlieren sie zudem keine/kaum lokale Informationen, wenn sie die globale Gaußverteilung bilden.

3.3 Potential des Horizont

Bei der Modellierung des Horizonts gehen sie davon aus, dass dessen relative Position unabhängig vom eigentlichen Inhalt des Bildes ist, und sie direkt dessen a-priori Wahrscheinlichkeit verwenden können.

Die relative Horizontposition wird nun durch eine Zahl $v^{hz} \in [0, 1]$ dargestellt, deren Potential in der Energiefunktion die Loglikelihood $\psi^{horizon}(v^{hz}) = -\frac{1}{2}(\frac{v^{hz}-\mu^{hz}}{\sigma^{hz}})^2 - \log(\sigma^{hz}\sqrt{2\pi})$ einer durch

das Training vorgegebenen Gaußverteilung (μ^{hz}, σ^{hz}) beträgt.

3.4 Potential einer Region

Das Potential ψ^{region} entspricht der log-Wahrscheinlichkeit, dass die in der Region enthaltenen Pixel tatsächlich eine einzelne Region mit der gegebenen Klassifizierung bilden werden, unabhängig von allen anderen Regionen. Dieser Wert wird ermittelt, indem das Tupel aus semantischer und geometrischer Klasse zusammen mit den visuellen Eigenschaften der Region von einem erlernten Klassifikator bewertet wird.

Für die visuellen Eigenschaften extrahieren sie im Grunde drei unterschiedliche Gruppen von Werten: Als erstes das oben bereits beschriebenen Aussehen $A_r = (\mu_r^A, \Sigma_r^A)$ als Gaußverteilung des Aussehens der enthaltenen Pixel und den Normalisierungsfaktor $\log\det\Sigma_r^A$ der Gaußverteilung. Diese Werte liefern die direktesten Informationen über das Aussehen der Region, wie Farbe oder Textur.

Als zweite Gruppe von Parametern verwenden sie die globale Gestalt einer Region, das heißt, ob sie eher klein oder groß, kompakt oder gestreckt, rund oder eckig ist. Konkret berechnen sie dazu die normalisierte Regionsgröße, den Umfang, jeweils das erste und zweite x und y-Moment[11], sowie Fehler bei der Approximation der oberen/unteren Regionsgrenze durch eine Gerade.

Und schließlich fügen sie den durchschnittlichen Kontrast an der Regionsgrenze und innerhalb der Region hinzu, da sich eine Region bei einer guten Segmentierung normalerweise von den benachbarten Region unterscheidet, aber im Inneren homogen ist.

Welche dieser Werte tatsächlich wichtig sind, entscheidet der Klassifikator, während der Trainingsphase selbständig. Es ist durchaus möglich, dass manche Werte beinahe ignoriert werden, oder auch nur für bestimmte Klassenunterscheidungen wichtig sind.

Schreibt man diese Parameter als $\phi_r(A_r, P_r)$, die Größe der Region als N_r und den gelernten Klassifikator als σ^{region}, so ist

$$\psi_r^{region}(S_r, G_r, v^{hz}; A_r, P_r) = -N_r \log \sigma^{region}(S_r \times G_r | \phi_r(A_r, P_r), v^{hz})$$

Der Faktor N_r bewirkt, dass größere Regionen auch größeren Einfluss auf die Energiefunktion haben und diese annähernd unabhängig von der Zahl der Regionen ist[2].

3.5 Potential benachbarter Pixel

Für das Potential $\psi_{pq}^{boundary}$ benachbarter Pixel verwenden sie das übliche Kontrastpotential, das bewirkt, dass ähnliche, benachbarte Pixel bevorzugt zu derselben Region gehören. Falls die beiden Pixel nicht in derselben Region sind, soll sich die Gesamtenergie um einen Wert vergrößern, der mit zunehmender Distanz ihrer Aussehensvektoren exponentiell abfällt, also:

$$\psi_{pq}^{boundary}(R_p, R_q; \alpha_p, \alpha_q) = \begin{cases} \exp(-\beta^{-1}||\alpha_p - \alpha_q||^2) & R_p \neq R_q \\ 0 & \text{sonst} \end{cases}$$

Dabei erfüllt β den Zweck einer Normalisierungskonstante und ist der durchschnittliche Kontrast zwischen allen benachbarten Pixeln. Da dieses Potential zudem über alle benachbarten Pixel in unterschiedlichen Regionen summiert wird, ist es insgesamt proportional zur Gesamtlänge aller Regionsgrenzen, wodurch dieses Potential die Vereinigung unterschiedlicher Regionen fördert.

3.6 Potential benachbarter Regionen

$\psi_{rs}^{pair}(S_r, G_r, S_s, G_s; Ar, P_r, A_s, P_s)$ ist das Potential zwischen zwei benachbarten Regionen r und s, also die log-Wahrscheinlichkeit, dass zwei solche Regionen r und s benachbart sind.

Hierbei gehen sie davon aus, dass die semantische und geometrische Klasse weitgehend unabhängig voneinander sind, und somit separate logistische Klassifikatoren für $S_r \times S_s$ und $G_r \times G_s$ erlernt werden können, was das Training natürlich vereinfacht. Wie im Potential ψ^{region} werden

[2]Dies sieht man daran, dass bei konstantem σ^{region} das Potential ohne N_r die Zahl der Regionen wäre und mit N_r die konstante Zahl der Pixel.

auch hier zusätzliche Eigenschaften ϕ_{rs} aus den Pixeln der Regionen entnommen. Diese sind die Differenz zwischen den Schwerpunkten der beiden Regionen, der Anteil der Grenzpixel, bei denen die Region r über s liegt, Länge und Ausrichtung der Grenze, die Abweichung bei einer Approximation der Grenze durch eine Gerade und der Unterschied der Aussehensvektoren normalisiert mit der Varianz innerhalb der jeweiligen Region.

Die Normalisierung des letzten Parameters soll die Tatsache widerspiegeln, dass im Hintergrund die Unterschiede zwischen den in Regionen enthaltenen Pixeln und die Unterschiede zwischen benachbarten Regionen grundsätzlich geringer sind, als bei Vordergrundobjekten.

Die anderen Eigenschaften beschreiben, die relative Lage der Regionen zueinander, beispielsweise, dass Straßen unter Gebäuden liegen und die Grenze zwischen ihnen gerade ist. Dadurch, dass der Anteil der Pixel von r, die über denen von s liegen, zu den Parametern gehört, ändern sich diese durch das Vertauschen von r und s, weshalb das Potential asymmetrisch ist.

Nennt man die gelernten logistischen Klassifikatoren σ^{pair_S} und σ^{pair_G}, so ist der Wert von $\psi_{rs}^{pair}(S_r, G_r, S_s, G_s; Ar, P_r, A_s, P_s)$ also

$$-\left(\frac{N_r}{|nbrs(r)|}+\frac{N_s}{|nbrs(s)|}\right)\left(\log\left(\sigma^{pair_S}(S_r \times S_s|\phi_{rs})\right)+\log\left(\sigma^{pair_G}(G_r \times G_s|\phi_{rs})\right)\right)$$

Der Vorfaktor $\left(\frac{N_r}{|nbrs(r)|}+\frac{N_s}{|nbrs(s)|}\right)$ soll größeren Regionen mehr Gewicht geben, ohne dabei die Anzahl der Nachbarn $nbrs(r)$ zu berücksichtigen. Die entsprechende Division ist nötig, da größere Regionen mehr Nachbarn haben, und für jede Region das obige Potential für jeden Nachbarn einmal zur Energiefunktion addiert wird.

4 Trainingsphase

Um die oben definierte Energiefunktion tatsächlich berechnen zu können, müssen also folgende Klassifikatoren gelernt werden:

- $\psi^{horizon}(v^{hz})$
- $\sigma^{region}(S_r \times G_r|\phi_r(A_r, P_r), v^{hz})$
- $\sigma^{pair_S}(S_r \times S_s|\phi_{rs})$
- $\sigma^{pair_G}(G_r \times G_s|\phi_{rs})$

Dazu lassen sie 715 Bilder von Hand segmentieren, indem sie einen entsprechenden Auftrag auf Amazon Mechanical Turk für unter 250$ anbieten, und wenden dann auf dem dadurch gewonnen Datensatz Standardlernverfahren an:

4.1 Position des Horizonts

$\psi^{horizon}$ ist die einfachste der vier Funktionen, da die Position des Horizonts als eindimensionale Gaußverteilung modelliert wird. Mit dieser Annahme wird sie, wie in Abschnitt A.1 beschrieben, vollständig durch den Mittelwert und die Standardabweichung aller Horizontpositionen in den Trainingsdaten bestimmt.

Mit den Trainingsdaten von [6] soll sich ein Mittelwert von ungefähr 0.5 und eine Standardabweichung von 0.15 (relativ zur Bildhöhe) ergeben, es entspricht also der Annahme, dass der Horizont genau in der Mitte des Bildes liege.

4.2 Regionenklassifikatoren

Die anderen drei Klassifikatoren werden nach einem logistischen Regressionsmodell gelernt, wie im Anhang A.3 beschrieben.

Hierbei haben sie festgestellt, dass es nicht ausreicht, den σ^{region} Klassifikator darauf zu trainieren, die Regionen aus den Trainingssegmentierungen korrekt den jeweiligen Klassentupeln

zuzuordnen. Die Trainingsdaten enthalten nämlich nur gültige Regionen, während viele Regionen, die während der Berechnung der Segmentierung entstehen, ungültig sind. Deshalb fügen sie zusätzliche, inkorrekte Regionen und eine inkorrekt-Klasse zu den Trainingsdaten hinzu. Eine gültige Segmentierung darf dann keine solche Region der inkorrekt-Klasse enthalten.

Sie generieren diese inkorrekten Regionen auf zwei Weisen: Als erstes vereinigen sie in den Trainingssegmentierungen alle Paare von benachbarten Regionen zu einer einzigen Region, falls diese beiden Regionen ausreichend unterschiedlich sind. Als zweites, lassen sie den Algorithmus die Trainingsbilder segmentieren und extrahieren alle Regionen, die falsch segmentiert wurden. Diese werden dann als inkorrekt zu den Trainingsdaten hinzugefügt, und die Trainingsbildersegmentierung wird wiederholt, bis – vermutlich – keine oder nur wenige Fehler auftreten.

5 Anwendung

5.1 Segmentierung

Mit der nun berechenbaren Energie vereinfacht sich das Problem der Bildsegmentierung auf das Finden des Minimums der Energiefunktion. Da sie natürlich zu kompliziert für ein analytisches Vorgehen ist, wird in [6] ein Optimierungsverfahren vorgestellt, bei dem abwechselnd die Zuordnung von Pixeln zu Regionen und die Zuordnung der Regionen zu Klassen zusammen mit der Horizontposition optimiert wird.

Für den ersten Schritt stellt sich die Frage, welche Pixel neu welcher Region zugeordnet werden sollen. Dafür wählen sie eine dieser beiden Möglichkeiten[3]: Im einfachsten Fall vereinigen sie zwei benachbarte Regionen r und s, das heißt, sie setzen $R_p = s$ für alle p mit $R_p = r$.

Im anderen Fall wählen sie eine Menge ähnlicher Pixel ω und weisen diese Pixel einer Region r zu, indem sie $R_p = r$ für alle $p \in \omega$ setzen. Für r wird dabei entweder eine neue, noch nicht in der Segmentierung vorkommende Region, oder eine Region, die in der Nachbarschaft der Pixel von ω liegt, gewählt.

Welche Pixel als ähnlich gelten, wird vor der Optimierung der Energiefunktion berechnet, indem für ein gegebenes Bild ein Wörterbuch Ω aller ähnlichen Mengen berechnet wird. Damit kann dann für das ω irgendein $\omega \in \Omega$ gewählt werden. Das Wörterbuch selbst konstruieren sie aus den Segmenten, die von traditionellen Segmentierungsverfahren gefunden werden. Sie wenden dazu mehrmals Meanshift mit unterschiedlichen Fenstergrößen für jeweils den Positions- und Farbvektor auf das Eingabebild an (siehe Anhang A.5) und fügen alle dabei gefundenen Segmente zu Ω hinzu. Nach jeder Meanshiftwiederholung führen sie zudem hierarchisches agglomeratives Clustern, wie im Anhang A.4 beschrieben, durch – das heißt, sie vereinigen benachbarte und ähnlich aussehende Segmente, bis jedes vereinigte Segment aus einer maximalen Anzahl ursprünglicher Segmente besteht – und fügen die dabei entstehenden Segmente ebenfalls zum Wörterbuch Ω hinzu. Ein Beispiel für eine solche Übersegmentierung zeigt Abbildung 3.

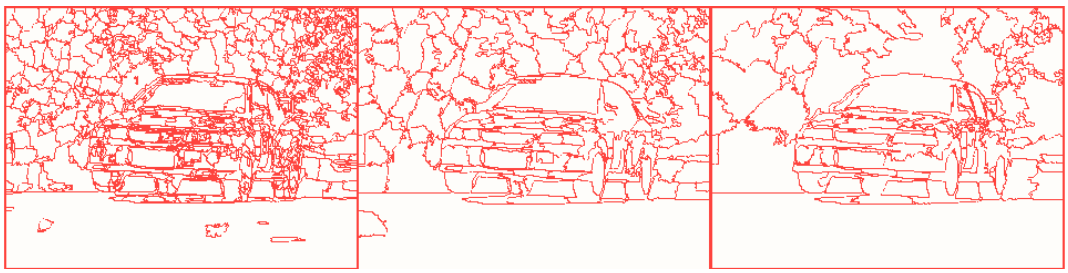


Abbildung 3: Beispiel für eine Übersegmentierung[6]

Außerdem sortieren sie das Wörterbuch nach der Entropie der Aussehensvektoren der Pixel in einem Segment, damit zuerst möglichst homogene Regionen entstehen. Damit kann man vermuten,

[3]Sie verraten nicht wie, aber es ist zu vermuten, dass sie entweder zufällig wählen, oder beide Möglichkeiten ausprobieren, und dann die verwerfen, die im zweiten Schritt zu einer höheren Energie führt.

dass sie das $\omega \in \Omega$ auch nicht zufällig wählen, sondern gemäß der Reihenfolge im Wörterbuch, und jedes ω genau einmal testen. Damit ist es im Grunde ein Greedyverfahren, und es wäre interessant zu wissen, wie gut die Ergebnisse mit einem anderen Auswahlverfahren wären.

Im zweiten Schritt ist die Aufteilung des Bildes in – noch nicht optimale – Regionen bekannt und es müssen nur noch die Klassen der Regionen und die Horizontposition berechnet werden. Hierfür verwenden sie ebenfalls Standardalgorithmen, indem sie die semantischen und geometrischen Klassen der durch den vorherigen Schritt beeinflussten Regionen mit max-product Belief propagation, wie im Anhang A.8 beschrieben, aktualisieren. Nachdem die Klassen feststehen, berechnen sie die Horizontposition mit ICM aus Anhang A.7.

Diese beiden Schritte werden nun mehrmals wiederholt, bis das Verfahren zu einer Segmentierung konvergiert ist. Falls sich bei einer der Wiederholungen der beiden Schritte die Energie vergrößert, anstatt sich zu verringern, verwerfen sie die in dieser Wiederholung aufgetretenen Änderungen, damit die Energie monoton sinkt.

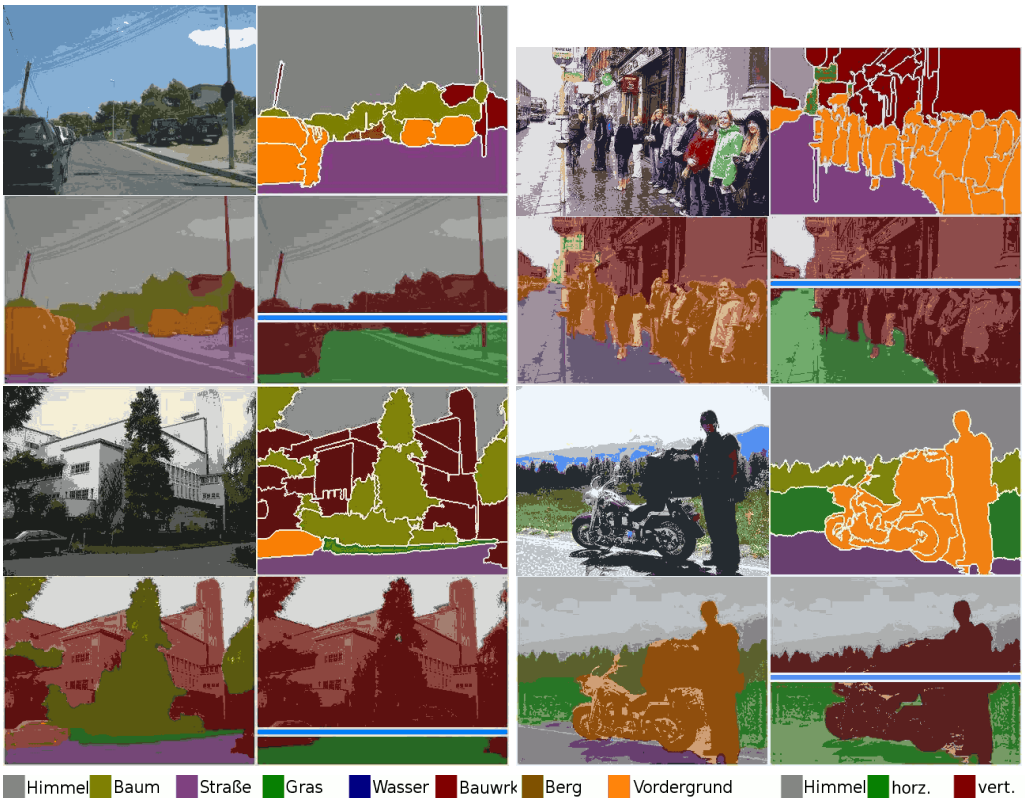


Abbildung 4: Segmentierung einiger Bilder. Im Uhrzeigersinn sieht man Originalbild, Segmentierung, geometrische Klassen und semantische Klassen

Abbildung 4 zeigt vier von Gould et al. gegebene Beispielbilder.

5.2 3D-Rekonstruktion

Für die 3D-Konstruktion setzen sie voraus, dass die interne Kamerakalibrierung bekannt ist, dass die Kamera mit ihrer Bildebene einen bestimmten Abstand y (konkret $y = 1.8$) von einer festen Bodenebene hat und dass der rechte-Richtungsvektor der Kamera parallel zur Ebene ist.

Unter diesen Annahmen gibt es für die Kamera nur einen weiteren Freiheitsgrad: die Rotation um den rechten-Richtungsvektor. Die anderen drei theoretisch möglichen Freiheitsgrade – Verschiebung der Kamera parallel zur Ebene und Rotation um die Ebenennormale – sind nämlich identisch zur Verschiebung und Rotation der gesamten Welt von Ebene und Kamera, weshalb

sich darin unterscheidende Parameter zur gleichen 3D-Rekonstruktion, in einem jeweils anderen Koordinatensystem, führen.

Damit ergibt sich aus dem Tangenssatz als Rotation der Kamera der Wert $\theta = \arctan(\frac{1}{f}(v^{hz} - v_0))$, wobei f die Brennweite der Kamera ist und v_0 die Bildmitte.

Ein Punkt im Weltkoordinatensystem wird dann mit der Projektionsmatrix $P = KR(\theta)$ auf die Bildebene projiziert und somit gilt umgekehrt für einen Punkt auf der Bildebene mit projektiven Koordinaten $[u\ v\ 1]^T$, dass der entsprechende projizierte Weltpunkt auf dem Strahl $R(\theta)^{-1}K^{-1}[u\ v\ 1]^T$ liegt.

Um nun aus der Segmentierung eine 3D-Rekonstruktion zu erhalten, teilen sie die Regionen nach ihrer semantischen Klasse in Bodenregionen und »vertikale« Regionen auf. Zu ersten zählen sie Straße, Gras und Wasser, für letztere nehmen sie die Klassen Baum, Gebäude, Berg und Vordergrundobjekt.

Für alle Pixel von Bodenregionen wählen sie dann einfach als zugehörigen projizierten Weltpunkt, den Schnittpunkt des Strahles mit der festgelegten Grundebene, unabhängig davon zu welcher der Bodenregionen der Pixel gehört.

Für die restlichen Regionen setzen sie voraus, dass alle Weltpunkte einer Region auf derselben Ebene liegen, so dass sie nur deren Orientierung und Abstand von der Kamera berechnen müssen.

Falls eine Region eine Bodenregion berührt, gehen sie davon aus, dass Pixel, die im Bild benachbart sind, auch von der Projektion benachbarter Weltpunkte stammen. Da die 3D-Position aller Bodenpixel aus dem vorherigen Schritt bereits bekannt ist, können sie dann die zu den Pixeln der Regionsgrenze gehörenden Weltpunkte durch die Punkte der Bodenregion approximieren. Da diese Punkte nach den getroffenen Annahmen durch den Schnitt zweier Ebenen bestimmt sind, müssen sie auf einer Geraden liegen, welche sich wiederum aus den Punktpositionen robust (z.B.: mit einer Art RANSAC) approximieren lässt. Damit ist der Abstand und ein Freiheitsgrad der Orientierung der Regionsebene bekannt.

Falls eine Region r keine Bodenregion berührt, berechnen sie die Orientierung der Regionsebene »mit den Pixeln am unteren Rand dieser Region« und setzen den Abstand auf den Mittelwert zwischen dem Abstand der Ebene der darunterliegenden Region s und dem maximal möglichen Abstand. Der maximal mögliche Abstand, ist entweder die Distanz zum Horizont oder der Abstand, bei dem die Grundebene hinter der Region s sichtbar würde, das heißt, der minimale Abstand zu einem Schnittpunkt zwischen der Grundebene und einem Strahl durch die Pixel an der Grenze zwischen den Regionen r und s.

Leider wird das Paper an dieser Stelle etwas vage, das heißt, sie beschreiben nicht, wie sie den noch fehlenden Freiheitsgrad der Regionsebene berechnen, oder die Orientierung einer Region aus den Pixeln an ihrem unteren Rand. Wahrscheinlich setzen sie zusätzlich noch voraus, dass jede Regionsebene senkrecht zur Grundebene ist, wie es auch Hoiem et al. tun[7]. Für die Approximation der Orientierung einer Region r, die keine Bodenregion berührt, gehen sie vermutlich davon aus, dass die Grenze zwischen der Region r und der darunterliegenden Region s eine Gerade ist und approximieren dann diese Gerade aus den bekannten 3D-Position der Pixel der Region s, um einen Freiheitsgrad der Orientierung der Region r zu erhalten. Dazu müssen sie die Regionen von unten nach oben verarbeiten und zudem ist unklar, was passiert, wenn die unterste Region keine Bodenregion ist.

Außerdem sagen sie, dass sie davon ausgehen, dass der Horizont unendlich weit von der Kamera entfernt ist, wodurch es aber unmöglich wird, eine Region an einem endlichen Abstand auf halbem Weg zum Horizont zu platzieren, wie sie am Ende des Abschnitts schreiben. Also arbeiten sie vermutlich doch mit einem endlichen Horizontabstand, sobald sie die Kameraneigung berechnet haben.

Zum Schluss platzieren sie Regionen der Klasse Himmel hinter der letzten vertikalen Region und setzen die 3D-Position jedes Pixels einer Region auf den Schnittpunkt des Strahles durch diesen Pixel mit der entsprechenden Regionebene.

Nach ihrer Beschreibung ignorieren sie völlig die zuvor ermittelten geometrischen Klassen, und berücksichtigen bei der 3D-Rekonstruktion nur die semantischen, obwohl es eigentlich sinnvoller erscheint, alle als horizontal klassifizierten Regionen als Bodenregionen und alle vertikal klassifizierten als vertikale Regionen zu verwenden. Allerdings gibt es eine hohe Korrelation zwischen

semantischer und geometrischer Klasse (siehe Tabelle 3), so dass es in den meisten Fällen keinen Unterschied zwischen diesen beiden Zuordnungsmöglichkeiten gibt.

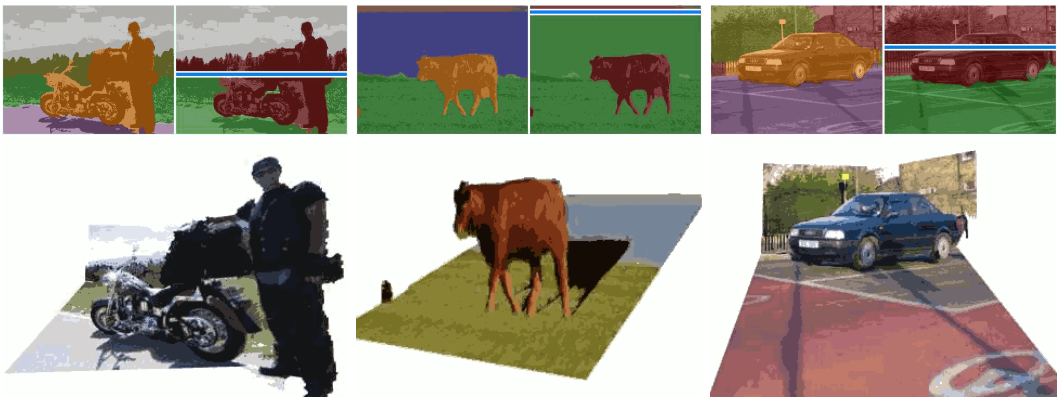


Abbildung 5: 3D-Rekonstruktion einiger Bilder

Abbildung 5 zeigt drei ihrer 3D-Rekonstruktionen. Sie sehen gut aus, obgleich sie nicht realistisch sind.

6 Vergleich

6.1 Datensätze und Vergleichsmodelle

Am praktischen lässt sich die Zuverlässigkeit eines Segmentierungsalgorithmus einschätzen, indem man ihn mit den momentanen state-of-the-art Verfahren auf den gleichen Eingabedaten vergleicht. Dies ist hier jedoch nicht direkt möglich, da die bisherigen Verfahren ein Bild entweder in semantische oder geometrische Klassen zerlegen, aber nicht in beide zugleich. Daher vergleichen Gould et al. in [6] nur eingeschränkte Varianten ihres Algorithmus mit den üblichen Verfahren, und den vollständigen Algorithmus nur mit einem einfachen CRF-Modell auf einem neuen Datensatz.

Diesen stellen sie aus den populären Datensätzen LabelMe [15], MSRC [2], PASCAL [3] und GC [9] zusammen, indem sie die Bilder wählen, die ungefähr 320×240 groß sind, ein Vordergrundobjekt enthalten und in denen der Horizont entweder sichtbar oder durch ein anderes Objekt verdeckt ist. Damit erhalten sie die 715 Bilder, von denen sie über Amazon Mechanical Turk eine Segmentierung in semantische und geometrische Klassen ermitteln ließen. Außerdem erstellen sie fünf zufällige Zerlegungen in 572 Testbilder und 143 Trainingsbilder, so dass ein Bild in vier Wiederholungen als Testbild und einmal als Trainingsbild verwendet wird.

Die MSRC und GC-Datensätze werden zudem vollständig verwendet, um zu untersuchen, wie gut ihr Verfahren auch ohne geometrische oder ohne semantische Klassen funktioniert. MSRC ist ein Datensatz von Microsoft Research Cambridge und enthält 591 Bilder mit 23 semantischen Klassen, von denen die Klassen Pferd und Gebirge üblicherweise ignoriert werden[2][18], so dass man effektiv mit 21 Klassen arbeitet. In den Bildern wurde jeder Pixel klassifiziert, allerdings sind manche Pixel als leer markiert, und sollten ignoriert werden. Außerdem gibt es keine vorgegebene Zerlegung in Test und Trainingsbilder, was zu leichten Abweichungen in den später angegebenen Prozentzahlen führen kann. GC (Geometric Context) ist ein Datensatz von Hoiem et al. [8] mit drei geometrischen Klassen Boden, vertikal und Himmel. Allerdings wurden dabei nur Superpixel klassifiziert.

Das einfache CRF-Vergleichsmodell erstellen Gould et al. nun, indem sie den Horizont und den Zusammenhang zwischen benachbarten Regionen aus ihrem Modell entfernen, und zusätzlich voraussetzen, dass jeder Pixel eine eigene Region ist und geometrische und semantische Klassen voneinander unabhängig sind. Damit entfernen sie zwei Potentiale aus ihrer Energiefunktion und vereinfachen das innere Regionspotential. Außerdem zerfällt die Energiefunktion durch die Annahme der Unabhängigkeit in zwei unabhängige Funktionen, die separat maximiert werden können:

$$E(S|I) = \sum_p \psi_p^S(S_p; \alpha_p) + \theta \sum_{pq} \psi_{pq}^{boundary}(S_p, S_q; \alpha_p, \alpha_q)$$

$$E(G|I) = \sum_p \psi_p^G(G_p; \alpha_p) + \theta \sum_{pq} \psi_{pq}^{boundary}(G_p, G_q; \alpha_p, \alpha_q)$$

Dabei sind ψ^S und ψ^G wieder logistische Klassifikatoren, die für dieses Modell neu gelernt werden müssen, θ ein Potentialgewicht wie in der ursprünglichen Energiefunktion und $\psi^{boundary}$ ist das in Abschnitt 3.5 definierte Kontrastpotential von benachbarten Pixeln, wobei es nun 0 für gleiche Klassen statt für gleiche Regionen ist. In Anhang A.6 ist eine graphische Darstellung des Modells gegeben. Mit einem solchen CRF-Modell lassen sich die optimalen Klassenzuordnungen beispielsweise durch Graphcut-Verfahren oder Belief propagation berechnen[11].

Es ist allerdings seltsam, dass sie kein CRF-Modell verwenden, das wie ihr Verfahren gleichzeitig semantische und geometrische Klassen zuweist, wie zum Beispiel:

$$E(S \times G|I) = \sum_p \psi_p^{SG}(S_p \times G_p; \alpha_p) + \theta \sum_{pq} \psi_{pq}^{boundary}(S_p \times G_p, S_q \times G_p; \alpha_p, \alpha_q)$$

Hätten sie auch die Ergebnisse eines solchen Modells angegeben, könnte man besser untersuchen, wie viele neue Pixelfehler durch den Verzicht auf Regionen und wie viele Fehler durch den Verzicht auf die gleichzeitige semantische und geometrische Klassifizierung entstehen.

Außerdem hätten sie, wenn sie die Qualität der Segmentierung in semantische und geometrische Klassen bei ihren kombinierten Daten sowieso separat vergleichen, auch jeweils ein bekanntes Verfahren verwenden können. Das gleiche Modell separat für beide Klassenarten zu verwenden, erlaubt es zwar prinzipiell Fehler zu erkennen, die nur bei semantischer oder nur bei geometrischer Klassifizierung auftreten. Dies ist aber in diesem Fall eigentlich sinnlos, da die gelernten logistischen Klassifikatoren und damit die gesamte Energiefunktion nach einem neuen Training symmetrisch bezüglich dem Vertauschen von semantischer und geometrischer Klasse sind (da ein gelernter Klassifikator nicht »weiß«, für welche Klassenart er lernen soll), so dass man damit nur Implementationsfehler erkennen könnte.

6.2 Vergleich der Segmentierung

Tabelle 1 zeigt die Ergebnisse des Vergleiches des regionbasierten Segmentierungverfahren von Gould et al. [6] mit ihrem einfachen CRF-Modell. Für jede Klassenart ist angegeben, wie viele Pixel in Durchschnitt korrekt klassifiziert wurden, und wie groß die entsprechende Standardabweichung ist:

	semantische Klasse		geometrische Klasse	
	Akkuranz	Abweichung	Akkuranz	Abweichung
Pixel CRF	74.3%	0.80%	89.1%	0.73%
Gould Regionen	76.4%	1.22%	91.0%	0.56%

Tabelle 1: Vergleich mit CRF auf kombinierten Daten

Man sieht dabei, dass die regionbasierte Segmentierung bei semantischen Klassen das Ergebnis um 2.1% verbessert, und bei geometrischen um 1.9%. Gould et al. sagen, diese Verbesserung sei signifikant. Allerdings ist dies bei den semantischen Klassen nur bei einem geringen Signifikanzniveau der Fall, da gilt $2.1\% < 2.39\% = 1.96 \cdot 1.22\%$ und damit die Bedingung für ein Signifikanzniveau von 5% nicht erfüllt ist.

Da sie aber auch berichten, dass ihr Algorithmus in jeder der fünf Aufteilungen der Kreuzvalidierung bessere Ergebnisse als das CRF-Modell liefert, kann man wohl trotzdem schließen, dass es

besser funktioniert. Sie berichten außerdem, dass die vom Algorithmus gefundene Horizontposition durchschnittlich um nur 6.9% der Bildhöhe vom tatsächlichen Wert abweicht.

Für den zweiten Vergleich schränken sie ihren regionbasierten Algorithmus auf nur semantische oder nur geometrische Klassen ein, und wenden ihn zusammen mit dem CRF-Pixel-Modell auf die MRSC- und GC-Datensätze an. Der Anteil der dabei korrekt klassifizierten Pixel ist in Tabelle 2, zusammen mit den Ergebnissen anderer bekannter Algorithmen, gezeigt:

MSRC		GC	
TextonBoost [18]	72.2%	Hoeim Pixel	82.1%
Yang et al. [21]	75.1%	Hoeim Alt [9]	86 %
Gould et al. [5]	76.5%	Hoeim Neu [8]	88.1%
Gould CRF	75.3%	Gould CRF	86.5%
Gould Regionen	76.4%	Gould Regionen	86.9%

Tabelle 2: Vergleich mit state-of-the-art Verfahren auf den MSRC[2] und GC-Datensätzen[8]

.

Sowohl bei der Beschränkung auf nur semantische als auch bei der Beschränkung auf nur geometrische Klassen führt ihr Algorithmus zu leicht schlechteren Ergebnissen, als die bisherigen state-of-the-art Verfahren. Leider ist unklar, ob dies daran liegt, dass ihr Algorithmus mit der Beschränkung auf eine Klasse per Region nicht sinnvoll arbeitet, oder ob er generell nicht die Leistungsfähigkeit der bisher bekannten Verfahren erreicht, also ob diese alten Verfahren auf den kombinierten Daten auch bessere Ergebnisse als ihr neuer Algorithmus erreichen würden.

Um zu überprüfen, welcher Fall zutrifft, sollte man alle Algorithmen jeweils auf alle Daten anwenden, was jedoch ohne veröffentlichte Implementierungen relativ umständlich ist. Alternativ kann man die Ergebnisse des Pixel CRF Verfahren, welches auf alle vier Daten angewandt wurde, vergleichen. Man sieht, dass es auf den MSRC-Daten 75.3% der Pixel korrekt klassifiziert und auf dem semantischen Teil der kombinierten Daten nur 74.3%, also kann man vermuten, dass die MSRC-Daten »leichter« sind, und alle Algorithmen auf den kombinierten Daten schlechter arbeiten würden. Dagegen werden auf den GC-Daten nur 86.5% korrekt klassifiziert, während es auf dem geometrischen Teil der kombinierten Daten 89.1% sind. Dies könnte wiederum darauf hindeuten, dass die GC-Daten »schwerer« sind, als die kombinierten, und das Verfahren von Hoiem et al. dort ebenfalls bessere Ergebnisse liefern würde.

Andererseits stammt der GC-Datensatz von Hoiem et al. und, da sowohl ihr Verfahren wie auch die Testsegmentierung von GC auf Superpixeln basiert, könnte es sein, dass ihr Verfahren auf den GC-Daten besonders gute Ergebnisse liefert, so dass es selbst mit den (wie sie behaupten, genaueren) Trainingsdaten von Gould, nur geringfügig besser wäre.

Es wäre auch interessant gewesen, in jedem Bild die Pixel zu markieren, die von ihrem Algorithmus korrekt und von den bisherigen state-of-the-art Verfahren falsch, beziehungsweise umgekehrt, klassifiziert wurden.

Zusätzlich zu ihren drei geometrischen Klassen klassifizieren Hoeim et al. übrigens ihre Regionen nach Unterklassen, bei der jeder vertikalen Region noch eine weitere Klasse von fest, porös, planar links, planar recht oder planar mittig zugewiesen wird. Diese zeigen eine leichte Übereinstimmung zu den semantischen Klassen von Gould. So sind Berge in erster Linie fest, Bäume größtenteils porös und Häuser planar[4].

Dies kann man so interpretieren, dass Hoiem et al. im Grund schon eine Art semantischer Klasse verwenden. Allerdings erreichen sie für ihre Unterklassen nur eine Genauigkeit von 61.5%, während Gould et al. 76.4% der Pixel mit den korrekten semantischen Klassen klassifiziert, so dass das Verfahren von Gould besser zu arbeiten scheint. Eventuell könnte dies aber auch durch die Qualität der Trainingsdaten begründet sein, da die Unterscheidung von semantischen Klassen wie Baum oder Haus, sehr viel intuitiver ist, als die Klassifizierung als fest oder planar links, und es somit weniger Fehler in den Trainingsdaten gibt. Hoeim et al. berichten dabei von Abweichungen von bis zu 15%, die durch Unklarheiten über die korrekte Segmentierung entstehen.

Leider berichten Gould et al. nicht, wie akkurat ihre semantische Segmentierung mit den Daten von [9] arbeitet, wenn man die Subklassen als semantische Klassen verwendet.

Zudem scheint das Verfahren von Hoeim et al. schneller zu sein, so berichten sie in [8], dass die Segmentierung eines 640×480 Bild dreißig Sekunden dauert, während der Algorithmus von Gould et al. für ein viertel so großes 320×240-Bild zwischen dreißig Sekunden und zehn Minuten benötigt. Allerdings behandelt [8] eine ältere Version ihres Verfahrens, das nur eine Genauigkeit von 86% für die geometrischen Klassen und 52% für die Subklassen erreicht, und für die neuere Variante scheinen sie keine Laufzeit anzugeben.

Desweiteren stellt sich die Frage, ob die Unterscheidung von geometrischen Klassen wirklich hilfreich ist. Wie man in den Tabellen 1 und 2 sieht, funktioniert das Verfahren mit und ohne Berücksichtigung von geometrischen Klassen gleich gut. Vergleicht man semantische und geometrische Klassen in den kombinierten Daten, so stellt man in Tabelle 3 sogar fest, dass Straße, Gras und Wasser fast immer horizontal sind, während Baum, Gebäude, Berg und Vordergrund fast immer vertikal sind. Man sollte somit fast dieselben Ergebnisse erreichen, wenn man die geometrische Klasse in der Energiefunktion ignoriert, und entsprechend der semantischen wählt.

	Himmel	horizont	vertikal
Himmel	0.139	0.000	0.006
Baum	0.007	0.004	0.130
Straße	0.000	0.218	0.008
Gras	0.000	0.064	0.005
Wasser	0.001	0.038	0.001
Gebäude	0.002	0.002	0.222
Berg	0.000	0.001	0.012
Vordergrund	0.002	0.008	0.127

Tabelle 3: Anteil der Pixel mit dem jeweiligem Klassenpaar

6.3 Vergleich der Rekonstruktion

Die 3D-Rekonstruktion aus einer Segmentierung und gegebenem Horizont ist im Grund eine vereinfachte Version des Popupverfahrens von Hoiem et al. [7]. Dort treffen sie dieselben Annahmen bezüglich der Kamera und teilen das Bild ebenfalls in Bodenregionen, vertikale Regionen und Himmelsregionen auf, welche jeweils im 3D-Raum als eine Ebene dargestellt und auf einer Grundebene platziert werden. Auch die Orientierung und Entfernung einer solchen Regionsebene wird im Prinzip gleich berechnet, indem eine Gerade an die untere Regionsgrenze angepasst wird. Dazu verwenden sie eine Houghtransformation und vereinigen benachbarte Linien. Außerdem verbinden sie sich schneidende Geraden zu Polylinien und modellieren alle Regionen, die zu einem der Segmente der Polylinie gehören, als ein einziges Objekt im 3D-Raum, indem sie die Ebenen der Regionen an den Rändern verbinden.

Gould et al. platzieren dagegen alle Regionsebenen unabhängig im Raum, so dass es in der 3D-Rekonstruktion Löcher zwischen benachbarten Regionen geben könnte[4]. Daher erzeugt das Verfahren von Hoiem et al. vermutlich bessere Rekonstruktionen von größeren Objekten wie Gebäuden oder Fahrzeugen. Dafür haben sie Probleme mit Überlappungen, die sie einfach ignorieren, und freistehenden Vordergrundobjekten, die von ihrem Verfahren häufig auf die Grundebene gelegt werden.

Ob die Überlappungen von Gould et al. wirklich gut behandelt werden, ist allerdings auf Grund ihrer vagen Erklärung nicht klar, vor allem da die Testbilder keine Überlappungen zeigen; außerdem entsteht bei Hoiem et al. das Problem der Vordergrundobjekten nicht durch ihren Rekonstruktionsalgorithmus, sondern durch ihr Segmentierungsverfahren.

[4] In beiden Fällen kann es keine Überlappungen aus Sicht der Kamera geben, da auf jedem Strahl durch einen Pixel des Bildes nur ein 3D-Punkt liegt.

6.4 Bilder

Abbildung 6 zeigt die Bilder von [6], bei denen ihr Verfahren gut funktioniert, und Abbildung 7 die Bilder, bei denen es schlecht funktioniert.

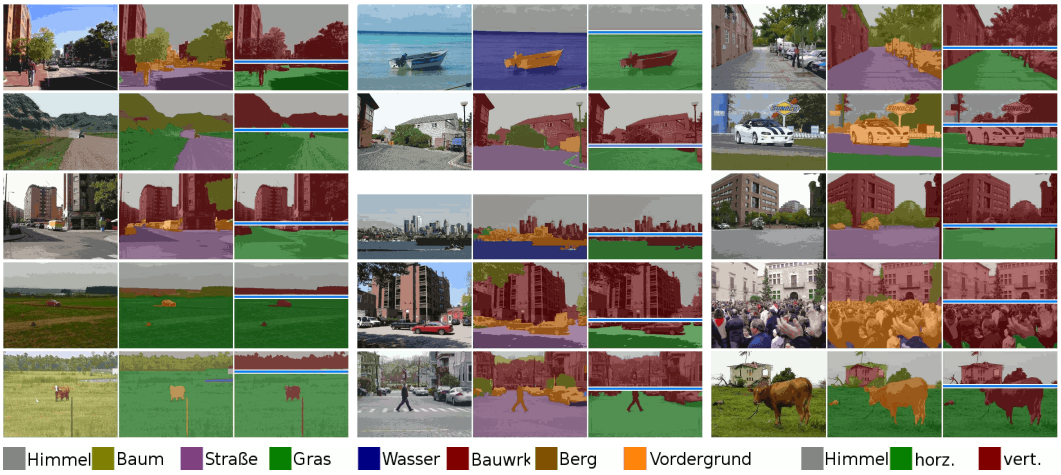


Abbildung 6: Bilder, bei denen das Verfahren gut funktioniert. Die drei Spalten zeigt jeweils, Orginalbild, semantische Klassifizierung und geometrische Klassifizierung mit Horizont

In den ersten beiden Bildern in Abbildung 7 wird Straße und Wasser verwechselt, und im zweiten Bild zusätzlich noch Himmel und Wasser, alles drei semantische Klassen mit sehr ähnlichem Aussehen. Eventuell hätte hier geholfen, zusätzliche semantische Klassen für Auto und Boot zu lernen, Im ersten Bild sieht man auch, dass in dem Modell eine Art Fluchtpunkt fehlt, mit dem es leicht zu erkennen wäre, dass die entfernten Häuser in der Bildmitte kein Vordergrundobjekt darstellen.

Bild 2 und Bild 3 zeigen zudem, dass das Verfahren von starken Schatten und Lichter verwirrt werden kann, da sie deutliche sichtbare, aber nicht existierende Grenzen bilden. In manchen Fällen gelingt die Segmentierung aber trotzdem, wie Bild 11 in Abbildung 6 zeigt.

Bild 4 zeigt, dass überlappende Objekte gleicher Farbe Probleme verursachen. In diesem Fall hätte wohl nur ein Objektdetektor, oder eine semantische Klasse für Fahrrad-/Motorradfahrer geholfen, da die Variationen innerhalb des Hauses auch nicht kleiner sind, als die zwischen Haus und Mensch.

Bild 5 wird wahrscheinlich von fast allen Verfahren falsch kategorisiert, da man durch die Fenster tatsächlich den Busch sehen kann. Eventuell hätte ein Auto-Objektdetektor geholfen, aber mit dem Erkennen von Scheiben und Reflexion haben ja selbst Tiere, wie beispielsweise die vereinzelt gegen Fenster fliegenden Vögel, Probleme.

In Bild 6 sind die Wolken zu grau für den Himmel, aber es ist schwer zu beurteilen, wie man es verbessern könnte. Eine höhere Affinität für Himmel würde wohl dazu führen, dass alles – inklusive die Masten – als Himmel klassifiziert wird. Es gibt auch keine Klasse, wie Boot, die den Masten sinnvoll zugewiesen werden könnte, am ehesten passt Vordergrund, obwohl die Schiffe im Hintergrund sind.

Außerdem neigt das Verfahren dazu, Büsche als Bäume zu bezeichnen, was aber wohl eher auf eine schlecht benannte Klasse hindeutet.

7 Fazit

Die Segmentierung in semantische und geometrisch konsistente Regionen führt zu einem holistischen Verständnis des gesamten Bildes, in dem jeder Pixel von einer bekannten Objektklasse stammt und bei dem globale Beziehungen zwischen den vorhandenen Objekten berücksichtigt werden.

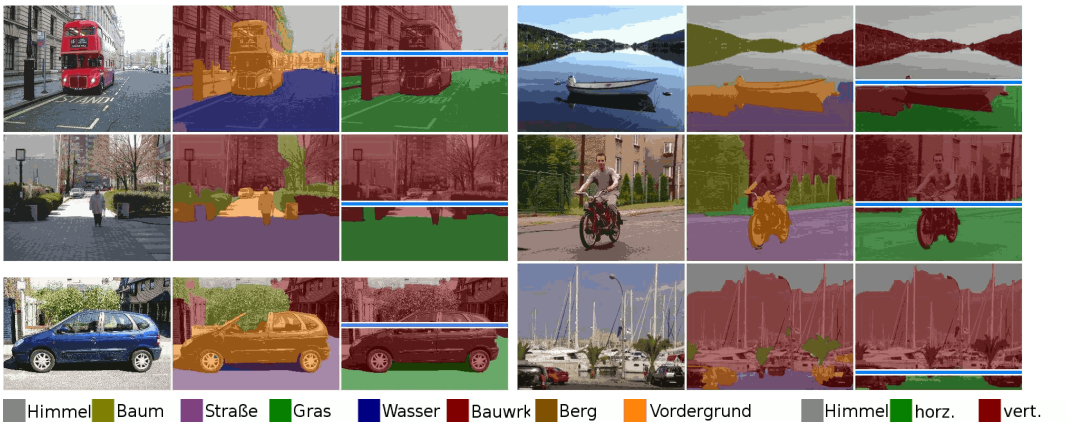


Abbildung 7: Bilder, bei denen das Verfahren schlecht funktioniert. Die drei Spalten zeigt jeweils, Orginalbild, semantische Klassifizierung und geometrische Klassifizierung mit Horizont

Die Qualität der Ergebnisse des Algorithmus von Gould et al. ist gut und vergleichbar mit den der momentanen state-of-the-art Verfahren, aber es ist noch unklar, in welchen Fällen er besser und in welchen er schlechter funktioniert.

Es ist zudem das erste Verfahren, das gleichzeitig eine semantische und geometrische Klassifizierung generiert, jedoch ist es zweifelhaft, ob die geometrische Klassifizierung tatsächlich sinnvoll ist.

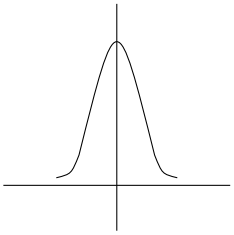
Außerdem scheinen sie als erste für die Segmentierung ein closed-loop Training zu verwenden, dass automatisch negative Trainingsdaten aus positiven erzeugt. Allerdings benötigen sie wie die meisten entsprechenden Segmentierungsverfahren eine große Anzahl von positiven Trainingsbildern, welche mühsam, pixelweise per Hand klassifiziert werden müssen. Durch die gleichzeitige Verwendung von geometrischen und semantischen Klassen, verdoppelt sich sogar die Menge der nötigen Trainingslabel.

Es wäre auch interessant verschiedene Erweiterungen des Verfahrens auszuprobieren, beispielsweise, ob sich in Kombination mit einem Objektdetektor die Klassifizierungsgenauigkeit von Objekten ohne homogenes Aussehen verbessert. Oder, ob es möglich wäre, präzisere geometrische Informationen zu ermitteln, indem auch nach Überdeckungen und Schatten gesucht wird.

A Verwendete Methoden

A.1 Gaußscher Maximum Likelihood

Bei der Verwendung des Gaußschen Maximum Likelihoods, wird die tatsächliche Wahrscheinlichkeitsverteilung durch eine Gaußfunktion approximiert:



Im eindimensionalen Fall ist die Gaußfunktion durch

$$\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}$$

gegeben. Für gegebene Datenpunkte $x_i \in \mathbb{R}$ mit $1 \leq i \leq n$, sind die Parameter (μ, σ), welche die Wahrscheinlichkeit für das Auftreten dieser Datenpunkte maximieren und gelernt werden sollten:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$$

$$\sigma = \sqrt{\frac{1}{n}\sum (x_i - \mu)^2}$$

Im mehrdimensionalen Fall kann man sich die Gaußverteilung als eine Art Ellipse oder Hyperkugel vorstellen, und sie ist durch

$$\frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)$$

gegeben, mit Daten $x \in \mathbb{R}^d$, dem Mittelwert $\mu \in \mathbb{R}^d$ und der Covarianzmatrix $\Sigma \in \mathbb{R}^{d\times d}$.

Die Parameter für die entsprechende ML sind dann:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$$

$$\Sigma = \frac{1}{n}\sum_{i=1}^{n} (x_i - \mu)(x_i - \mu)^T$$

[12]

A.2 Entropie

Die von Shannon[16] eingeführte Entropie ist ein Maß für den Informationsgehalt einer Wahrscheinlichkeitsverteilung.

Im Falle von diskreten Variablen, bei denen Möglichkeit i mit Wahrscheinlichkeit p_i auftritt, ist es definiert als $-\sum_i p_i \log p_i$.

Für kontinuierliche Variablen ist es entsprechend $-\int_{-\infty}^{\infty} p(i) \log p(i) di$.

Gould et al. sagen nicht, wie sie die Entropie für ihre Segmente berechnen, eine Möglichkeit wäre es, wenn sie, da sie sowieso eine Gaußverteilung des Aussehens ihrer Regionen berechnen, die Entropie dieser Gaußverteilung nehmen würden. Setzt man die kontinuierliche Gaußverteilung in die obige Gleichung ein, ergibt sich eine Entropie von $\frac{1}{2}\ln((2\pi e)^d|\Sigma|)$.

A.3 Multiclass logistic regression classifier

Bei der logistischen Klassifizierung modelliert man die Wahrscheinlichkeitsverteilung mittels der logistischen Funktion $\frac{e^z}{1+e^z}$ mit $z \in \mathbb{R}$. Sie hat gegenüber linearen Klassifizierung den Vorteil, dass sie alle Werte auf eine Wahrscheinlichkeit im Interval $[0, 1]$ abbildet.

Sind die Daten mehrdimensional, so setzt man $z = \beta^T \cdot [x\ 1]$ mit $x \in \mathbb{R}^d, \beta \in \mathbb{R}^{d+1}$. Dabei ist β der Vektor der Regressionskoeffizienten, der angibt, wie wichtig jedes Element des Datenvektors x ist. $\beta \cdot [0^d\ 1]$ ist eine Art Bias.

Für mehrere Klassen m verwendet man mehrere Klassifikatoren mit jeweils eigenen Koeffizienten β_i, deren Ausgabe zusammen einen m-dimensionalen Vektor bildet, in dem jedes Element einer Art Wahrscheinlichkeit entspricht, dass diese Klasse korrekt ist. Für Klasse j ist der entsprechende Wert $\frac{e^{\beta_j^T \cdot [x\ 1]}}{\sum_i e^{\beta_i^T \cdot [x\ 1]}}$. Dabei verschwindet die 1 im Nenner, setzt man aber ein β_j auf 0^{d+1}, erkennt man wieder die logistische Funktion.

Um die Parameter β_j für eine Maximum Likelihood-Verteilung zu bestimmen, verwendet man üblicherweise numerische Verfahren, wie z.B.: das Newton-Raphson-Verfahren.[14]

A.4 Agglomerierendes hierarchisches Clustern

Beim (agglomerierenden hierarchischen) Clustern arbeitet, man mit Mengen $M_1, \dots M_m \subseteq V$, auf die alle Datenpunkte aufgeteilt werden.

Die M_i sollen paarweise disjunkt sein und sich zur Menge V vereinigen:

$$M_i \cap M_j = \emptyset \; \forall i \neq j$$

$$\bigcup_i M_i = V$$

Beim agglomerierendem hierarchischen Clustern [13] sind initiale Mengen gegeben und man vereinigt iterativ ähnliche Mengen, bis man eine maximale Anzahl von Vereinigungen oder eine maximale Anzahl von Untermengen in einer vereinigten Menge erreicht. Es werden also die folgenden beiden Schritte wiederholt:

Ermitteln der beiden ähnlichsten Mengen mit einer gegebenen Distanzfunktion d:

$$(i, j) \leftarrow \operatorname*{argmin}_{i,j} d(M_i, M_j)$$

Vereinigen dieser beiden Mengen und Ersetzen dieser beiden Mengen durch die Vereinigung:

$$\{M_k | 1 \leq k \leq m\} \mapsto (\{M_k | 1 \leq k \leq m\} \setminus \{M_i, M_j\}) \cup \{M_i \cup M_j\}$$

Gould et al. verwendeten in [6] dieses Clusterverfahren, um verschiedene Übersegmentierungen zu erstellen. Die Mengen M_i entsprechen also einem Segment von ähnlichen Pixeln und die Funktion d bewertet die Ähnlichkeit des Aussehens der Segmente (z.B.: Euklidischer Abstand zwischen den Aussehensvektoren).

A.5 Mean-Shift

Mean-Shift ist ein parameterfreies Verfahren zum Clustern von Datenpunkten, bei dem man ein Fenster um einige Datenpunkte legt und dieses solange, zum Schwerpunkt der im Fenster enthalten Punkte schiebt, bis es zu einem Clusterzentrumpunkt konvergiert ist. Alle Punkte, von denen das Fenster aus zum gleichen Clusterzentrumpunkt konvergiert, werden zum gleichen Cluster gezählt.

Hat man eine Menge von Datenpunkten $V \subset \mathbb{R}^d$ und ein Fenster mit Radius $r \in \mathbb{R}$ an Position $p \in \mathbb{R}^d$, so erhält man das entsprechende Clusterzenter durch folgende Iteration, die bis zur Konvergenz wiederholt werden sollte:

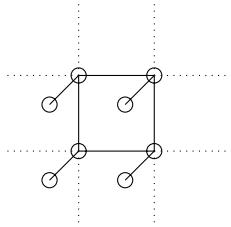
$$p \leftarrow \frac{1}{|w|} \sum_{s \in w} s \text{ mit } w = \{s \in V | d(s, p) \leq r\}$$

Startet man diese Iteration je einmal von jedem Datenpunkt aus, indem man jeweils $p = s$ für alle $s \in V$ setzt, erhält man (langsam) alle Cluster.

[11]

A.6 Conditional Random Field (CRF)

Ein CRF ist ein graphisches Modell zur Beschreibung der Wahrscheinlichkeit einer Variablenbelegung. Das für Bildanalysen üblicherweise und auch hier verwendete Modell sieht folgendermaßen aus:



Jeder Knoten symbolisiert eine Variable und jede Kante eine Funktion, die von diesen Variablen abhängt. Das obige Modell besteht aus zwei Schichten, bei denen die Variableknoten in der vorderen Schichten dem Aussehen der Pixel eines Bild entsprechen, und die Knoten der hinteren Schicht einer unbekannten Klassifizierung. Die Kanten zwischen Knoten von unterschiedlichen Schichten entsprechen dem Klassen-Pixel-Potential, und die Kanten zwischen den Knoten der hinteren Schicht entsprechen dem Potential zwischen benachbarten Klassen.

Es handelt sich dabei im Prinzip nur um eine andere Darstellung einer Energiefunktion, aber es wurden viele Algorithmen entwickelt, die für eine in dieser Form gegebene Funktion eine maximierende Variablenbelegung berechnen können, wie z.B. Graphcut-Verfahren, ICM, oder Belief propagation [11][1].

A.7 Iterated Conditional Modes (ICM)

ICM ist ein Greedy-Verfahren zur Bestimmung der Variablenwerte, welche die Wahrscheinlichkeit eines graphischen Modells (wie CRF) maximieren.

Dazu wird einfach ein Knoten beziehungsweise eine Variable des Modells ausgewählt und der Wert für diese Variable berechnet, der die vom Modell berechnete Funktion maximiert, wenn alle anderen Variablen unverändert bleiben. Bei einer diskreter Variable kann man dies zum Beispiel durch Ausprobieren aller Werte realisieren, bei stetigen Variablen durch eine Art Differenzenquotient.

Hat man dann eine Variable optimiert, wählt man eine andere und optimiert diese ebenfalls, bis keine optimierende Änderung mehr möglich ist.

Es ist simple und recht schnell, liefert aber keine optimalen Ergebnisse[1].

A.8 Belief propagation

Hierfür wird das allgemeine graphische Modell in einen Faktorgraphen, umgewandelt und dann dynamische Programmierung angewandt.

Ein Faktorgraph ist ein bipartites, graphisches Modell, bei dem berechnende Funktionen und Variablen getrennt sind. Es berechnet eine Funktion $p(X) = \prod_i f_i(X_i)$, wobei f_i die Funktionen, X_i für jede Funktion eine Menge von Variablen ist, von denen sie abhängt, und X die Menge aller Variablen(-belegungen) ist. Diese Funktion wird als Graph dargestellt, bei dem ein Knoten für jede Funktion f_i und ein Knoten für jede Variable $x \in \bigcup_i X_i$ existiert. Zwischen dem Funktionsknoten f_i und der Variable x existiert genau dann eine (ungerichtete) Kante, wenn $x \in X_i$ ist. Es gibt keine Kanten zwischen zwei Funktionsknoten oder zwei Variablenknoten. Jedes graphische Modell lässt sich in einen Faktorgraphen überführen[1].

Um nun die wahrscheinlichsten Variablenbelegungen zu berechnen, muss man die Funktion $\operatorname{argmax}_X p(X) = \operatorname{argmax}_X \prod_i f_i(X_i)$ optimieren. Wären die X_i disjunkt, könnte man dies einfach Ausmultiplizieren und separat $\prod_i \operatorname{argmax}_{X_i} f_i(X_i)$ maximieren. Dies ist zwar normalerweise nicht der Fall, aber man kann es meistens in zwei Produkte $\operatorname{argmax}_{X'} \prod_i f_i'(X_i') \operatorname{argmax}_{X''} \prod_i f_i''(X_i'')$ zerlegen, die sich dann separat rekursiv maximieren lassen. Dies führt letztendlich, nach mehreren Umformungen [1] zu diesen Gleichungen:

$$\mu_{f_s \to x}(x) = \max_{x_1,\ldots,x_M} \left(f(x, x_1, \ldots, x_M) \prod_{m \in N(f_s) \setminus x} \mu_{x_m \to f_s}(x_m) \right)$$

$$\mu_{x \to f_s}(x) = \prod_{l \in N(x) \setminus f_s} \mu_{f_l \to x}(x)$$

Dabei bezeichnet f Funktionsknoten, x einen Variablenknoten und N die Nachbarn eines Knotens. Jede Nachricht ist eine Funktion und kann im Falle von endlichen möglichen Variablenwerten als Vektor dargestellt werden.

Sendet man diese Nachrichten nun von jedem Knoten zu seinen Nachbarn, erhält man die maximale Wahrscheinlichkeit von $p(X)$. Um auch die entsprechende Variablenbelegung zu erhal-

ten, muss man noch in jedem Knoten speichern, welche Wahl der Variable zu dem übersendeten Maximum führt.

Belief propagation ergibt allerdings nur dann die optimale Belegung und lässt sich in einem Durchlauf berechnen, wenn es keine Zyklen gibt. Trotzdem kann man es auch auf Graphen mit Zyklen anwenden, indem man so tut, als ob es keine gäbe, und einfach die Nachrichten versendet. In den meisten Fällen konvergiert es dann zu einer guten, approximativen Lösung[1] [12].

Im Falle von Gould et al. sind die Variablenwerte die möglichen Klassen und die Funktionen sind die Potentiale der Energiefunktion.

Literatur

[1] Christopher M. Bishop. *Pattern recognition and machine learning.* Springer, 2006.

[2] A. Criminisi. Microsoft research cambridge object recognition image database. `http://research.microsoft.com/vision/cambridge/recognition`, 2004.

[3] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge, 2007.

[4] S. Gould, B. Liu, , and D. Koller. Single image depth estimation from predicted semantic labels. In *CVPR*, 2010.

[5] S. Gould, J. Rodgers, D. Cohen, G. Elidan, and D. Koller. Multi-class segmentation with relative location prior. In *IJCV*, 2008.

[6] Stephen Gould, Richard Fulton, and Daphne Koller. Decomposing a scene into geometric and semantically consistent regions. In *ICCV*, 2009.

[7] D. Hoiem, A. A. Efros, and M. Hebert. Automatic photo pop-up. In *ACM SIGGRAPH*, 2005.

[8] D. Hoiem, A. A. Efros, and M. Hebert. Geometric context from a single image. In *ICCV*, 2005.

[9] D. Hoiem, A. A. Efros, and M. Hebert. Recovering surface layout from an image. In *IJCV*, 2007.

[10] D. Hoiem, A. N. Stein, A. A. Efros, and M. Hebert. Recovering occlusion boundaries. In *ICCV*, 2007.

[11] Bastian Leibe. Lecture computer vision. 2009.

[12] Bastian Leibe. Lecture machine learning. 2010.

[13] C. D. Manning, P. Raghavan, and H. Schütze.

[14] Germán Rodríguez. Lecture notes: Generalized linear models. `http://data.princeton.edu/wws509/notes/c6s2.html`.

[15] B. C. Russell, A. B. Torralba, K. P. Murphy, and W. T. Freman. Labelme: A database and web-based tool for image annotation. In *IJCV*, 2008.

[16] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 1948.

[17] J. Shotton, J. Winn, C. Rother, and A. Criminisi. Texton-boost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *ECCV*, 2006.

[18] J. Shotton, J. Winn, C. Rother, and A. Criminisi. Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context. In *IJCV*, 2007.

[19] J. Winn, A. Criminisi, and T. Minka. Object categorization by learned universal visual dictionary. In *ICCV*, pages 1800–1807. IEEE Computer Society, 2005.

[20] C. Wojek and B. Schiele. A dynamic conditional random field model for joint labeling of object and scene classes. In *ECCV*, 2008.

[21] L. Yang, P. Meer, and D. J. Foran. Multiple class segmentation using a unified framework over mean-shift patches. In *CVPR*, 2007.