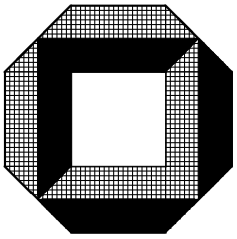
Proseminar

Künstliche Intelligenz



Universität Karlsruhe (TH)
Fakultät für Informatik
Institut für Algorithmen und Kognitive Systeme

Prof. Dr. J. Calmet
Dipl.-Inform. A. Daemi

Wintersemester 2003/2004

Institut für Algorithmen und Kognitive Systeme
Fakultät für Informatik
Universität Karlsruhe
Am Fasanengarten 5
76 128 Karlsruhe

Bildverarbeitung und Robotik

22. Februar 2004

Matthias Tandler - Korbinian Molitorisz

Bildverarbeitung

1 Einleitung

Wenn man heutzutage von Bildverarbeitung spricht, denken viele zunächst an mächtige Grafik-Bearbeitungsprogramme. Diese Programme sind unter Anderem in der Lage Photos zu retuschieren, eigene Bilder von Benutzern erstellen zu lassen, oder gar komplexe 3D-Szenen in Photoqualität zu erstellen. Was jedoch haben diese Eigenschaften mit künstlicher Intelligenz zu tun? Vorgefertigte Algorithmen, die eine 3D-Szene rendern, kann man nicht gerade als intelligent bezeichnen.

Eine ursprüngliche Vision war „Augen und Ohren für den Rechner“[GG00, Seite 816] zu entwickeln. Jedoch ahnte damals wohl noch niemand wie umfangreich die Implementierung dieser recht einfach klingenden Aussage sein möge. Naiv könnte man behaupten, dass es bereits Augen und Ohren in Form von CCD-Kameras und Mikrofonen gibt. Gemeint war aber nicht Bilder und Töne in digitale Daten umzuwandeln sondern diese auszuwerten oder gar zu interpretieren. Zunächst forschte man in Richtung der „Bild-zu-Bild-Transformation“(z.B. Entzerrung von Luftbildern für militärische Zwecke).

Die Probleme der Objekterkennung wurden zunächst an der Zeichenerkennung (wie z.B. Handgeschriebene Druckbuchstaben) untersucht. Hier versuchte man Objekte anhand verschiedener Erkennungsmerkmale zu klassifizieren. Die Erfolge dieser Forschung sind heute (2003) mehr als deutlich in Form der neuen Tablet-PC's und Handheld Computer, die in der Lage sind Handschrift zu erkennen, zu spüren. Die Entwicklung der Mustererkennung findet man heute jedoch nicht nur im Bereich der Schrifterkennung sondern auch in Bereichen wie Erkennung von Werkstücken auf einem Fliesband und Auswertung von seismographischen Messungen etc. Man stellte fest, dass diese spezielle Art vom Bildverstehen für allgemeine Probleme ungeeignet war. Man brauchte Programme, die in der Lage waren Objekte dreidimensional zu erkennen. Hier spielt nicht nur die Form eines Objektes eine Rolle sondern ebenso die Beleuchtung des Objekts und somit entstehende Schatten, die Farbe bzw. Spiegelungen auf der Oberfläche, Transparenzeigenschaften, räumliche Tiefe und Größenverhältnisse, etc.

Hierbei kommt eine weitere Schwierigkeit ins Spiel. Was ist wenn das Bild, das uns vorliegt, durch äußerliche Faktoren verfälscht wurde? Wie effektiv arbeiten unsere Algorithmen wenn beispielsweise ein Teil des Bildes durch Nebel beeinflusst ist? Ein weiteres Ziel ist es dreidimensionale Objekte aus zweidimensionalen Bildern zu rekonstruieren. Also Tiefenhinweise aus Einzelbildern zu erkennen und auszuwerten. Beispiele hierzu sind u.A. Interpretation von Stichzeichnungen als dreidimensionale Drahtobjekte, Schatten und Schattierung, Verdeckungsrelationen, etc.

Obwohl Computer inzwischen in der Lage sind, Handschrift mit befriedigendem Erfolg zu erkennen, bereitet uns die allgemeinere Objekterkennung sehr viel mehr Schwierigkeiten. Das hängt unter Anderem damit zusammen, dass niemand genau weiß, an welchem Punkt man anfangen könnte zu forschen. Somit existieren bis heute einige viel versprechende Ansätze zu diesem Thema. Die

Behauptung, dass die allgemeine Objekterkennung die „Kinderschuhe“ weitestgehend verlassen hat, muss jedoch zum gegenwärtigen Zeitpunkt stark bestritten werden.

2 Kantendetektion

Warum sind Kanten für die Bildverarbeitung so wichtig? Was sind Kanten überhaupt und wie lassen sie sich finden? H.A.Mallot beschreibt Kanten als „Orte im Bild, an denen sich die Intensität abrupt oder wenigstens „schnell“ ändert. [...] Die höchste örtliche Dichte von Informationen in Bildern tritt bei schneller Änderung der Intensität, d.h. an Kanten auf [...].“[Mal98, Seite 75]

Wir sind in erster Linie nicht daran interessiert ob ein Objekt besonders glänzt oder ob es rot oder blau ist, sonder vielmehr worum es sich bei diesem Objekt überhaupt handelt. Daher sehen viele Forscher die Kantendetektion nicht als ein in sich abgeschlossenen Prozess an, sondern vielmehr als grundlegende Basis für verschiedene Folgeprozesse an. Man sieht so leicht ein, dass ein schlecht arbeitender Kantendetektor zu großen Folgefehlern in weiteren Bildverarbeitungsprozessen führen kann. Wo genau sollen also die Stärken eines Kantendetektors liegen? Was soll er überhaupt können? Laut Canny (1986)[Mal98, Vgl Seite 79] sollte er folgenden Anforderungen genügen:

1. **Detektionsgüte:** Geringe Fehlerwahrscheinlichkeit für irrtümliche Detektion und Übersehene Kanten. Dies ist äquivalent zu der Forderung nach einem hohen Signal-zu-Rausch Verhältnis am Ausgang des Detektors.
2. **Lokalisation:** Das Maximum des Detektorausgangs soll an der Position der Kante auftreten.
3. **Eindeutigkeit:** Es soll nur eine Antwort pro tatsächlich vorhandener Kante erzeugt werden. Dies folgt z.T. aus der Detektionsgüte.

Wie erkennt nun ein Kantendetektor wo genau eine Kante vorliegt? Jeder Kantendetektor gibt ein unbrauchbares Ergebnis aus, wenn die Vorlage nicht in angemessener Qualität vorliegt. Wenn wir also einen Kantendetektor auf ein fehlerhaftes Bild (verursacht z.B. durch Kratzer auf dem Objektiv einer Kamera) anwenden, können wir auch nur mit einem semioptimalen Ergebnis rechnen. Daher muss das Bild vorher bearbeitet werden um so Fehler zu minimieren. Dies erreicht man indem man zunächst einen Filter auf das Objekt (Bild) anwendet.

2.1 Filter

2.1.1 Unschärfe-Filter

Mit Hilfe eines Unschärfe-Filters lässt sich ein Bild „glätten“ bzw. „weich zeichnen“. Es entsteht dadurch der Eindruck der Unschärfe. Was für den menschlichen Betrachter eher als störend empfunden wird ist eine sehr nützliche Voroperation für den Kantendetektor da so störende Rauschmuster unterdrückt werden und Kanten als Maxima eindeutiger zu identifizieren sind.

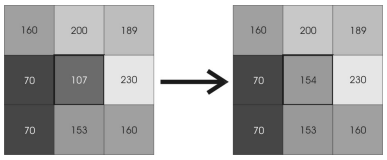


Abbildung 1: Grauwertbestimmung

Das Prinzip ist sehr einfach. Man wählt nacheinander jeden Pixel aus und ersetzt seinen Grauwert durch das Grauwertmittel seiner Nachbarpixel. In Abbildung 1 ersetzten wir den Grauwert 107 des mittleren Pixels durch den Grauwert 154. Durch dieses Verfahren gehen natürlich Informationen im Detail verloren. Wir bekommen jedoch wesendlich exaktere Kanten da sich relevante Grauwertmaxima eindeutiger abzeichnen. Man kann sich leicht vorstellen, dass bei steigender Auflösung dieses Verfahren weiter optimiert werden muss. Immerhin arbeitet der soeben vorgestellte Algorithmus auf Pixelebene. Bei steigender Auflösung kann jedoch schnell ein Areal von 2x2 Pixeln Fehler aufweisen. Somit sollte man bei der Neuberechnung eines Pixel mehr als nur die direkt umliegenden mit einbeziehen.

2.1.2 Gauss-Filter

Der Gauss-Filter ist ein Standardhilfsmittel der Bildverarbeitung. Er erlaubt es jeden einzelnen Pixel mit Hilfe aller Pixel eines beliebig großen Umfeldes zu mitteln. Je größer der Abstand zwischen dem zu ermittelnden Pixel und dem betrachteten dabei ist, desto weniger beeinflusst dieser den endgültigen Grauwert. Da wir diese Berechnung für jeden einzelnen Pixel des Bildes unverändert anwenden, nennt man eine solche Operation „ortsinvariant“ oder „translationsinvariant“.

Diesem Verfahren liegt eine dreidimensionale Gauss'sche-Glockenkurve zugrunde. Das Prinzip dieses Filters ist sehr einfach. Man wählt ein Pixel aus und ersetzt ihn durch die gewichtete Summe der umliegenden Werte. Die dabei benutzte Gauss-Glocke beschreibt man wie folgt:

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-x^2+y^2}{2\sigma^2}}$$

Die Konstante „σ“ gibt hierbei die Streckung der Kurve an und legt somit die Gewichtung der umliegenden Pixel fest. Somit wird der neue Pixel $I(x_0, y_0)$ in Abhängigkeit der umliegenden Pixel $I(x, y)$ mit $G_\sigma(d)$ bestimmt. d gibt hierbei die Distanz des Pixels $I(x, y)$ zum Pixel $I(x_0, y_0)$ an. Diese beiden Einzelfunktionen fassen wir nun in einer neuen Funktion H zusammen.

$$H = I * G_0$$

Und speziell für unser zu berechnendes Bild:

$$H(x, y) = \sum_{u=-\infty}^{+\infty} \sum_{v=-\infty}^{+\infty} (f(u, v) * g(x - u, y - v))$$

In der Praxis hat sich eine Summe von $+ - 3\sigma$ bewährt.

2.2 Schematische Kantendetektion

Stellen wir uns vor, uns liegt ein Bild in Graustufen vor. In diesem Bild möchten wir Kanten erkennen. Zunächst minimieren wir das zweidimensionale Kantendetektionsproblem auf eine Dimension. Dies erreichen wir, indem wir uns nur eine einzelne Pixel-Zeile anschauen. Die Grauwerte dieser Pixel können wir nun

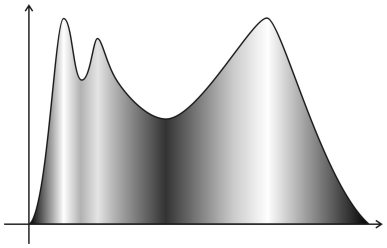
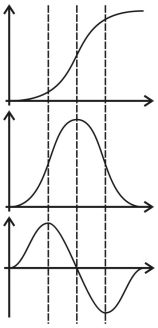


Abbildung 2: Grauwertverlauf



in einem Diagramm abtragen. Ein schwarzer Pixel in einem 8-Bit-Grauwertbild weist den Wert 0, ein weißer dagegen den Wert 255 auf. Der Abbildung 2 dargestellte Grauwertverlauf könnte so eine solche Pixelreihe repräsentieren. An der Höhe des Graphen lässt sich nun der jeweilige Grauwert des entsprechenden Pixels ablesen. Wie bereits weiter oben definiert, ist eine Kante eine abrupte Intensitätsänderung. In diesem Graph erkennt man nun sehr deutlich vorhandene Kanten. Zur genauen Bestimmung ihrer Positionen nehmen wir ein elementares Hilfsmittel der Analysis zu Hilfe. Prinzipiell führen wir nun eine klassische Kurvendiskussion durch. Kanten können wir nun mathematisch als Wendepunkte des Grauwertgraphen auffassen. Diese Wendepunkte sind nun die von uns gesuchte Intensitätsänderung. Nach zweifacher Ableitung werden sie zu exakt berechenbaren Nullstellen. Eine „reine“ Konturkante ist durch zwei nahe beieinander liegende Sprungkanten unterschiedlicher Vorzeichen charakterisiert. In einigen Fällen kommt es vor, dass die zweite Ableitung zwar Nullstellen aufweist, jedoch einige keine Kanten sind. Hierbei handelt es sich um eine minimale bzw. keine Grauwertänderung im Ausgangsgraphen. Man findet diese falschen Kanten durch die Untersuchung der dritten Ableitung. Diese muss an den Positionen der Nullstellen der zweiten Ableitung einen von „0“ verschiedenen Wert ergeben. Dieses Verfahren kann man nun auf jede einzelne Zeile des zweidimensionalen Bildes anwenden.

Natürlich ist diese Möglichkeit der Kantendetektion noch recht primitiv. Daher werden wir im Folgenden einige Verfahren kennen lernen, die es uns ermöglichen, Kanten eines Bildes wesentlich geschickter und weniger Fehlerbehaftet zu erkennen.

2.3 Sobel-, Roberts-, Prewitt-Operatoren

Neben dem Sobel-Operator existieren noch weitere Operatoren wie z.B. der Roberts-Cross- oder der Prewitt-Operator. Die Funktionsweise der einzelnen

Operatoren ist sehr Ähnlich. Sie unterscheiden sich im Wesentlichen lediglich in der Wahl der Matrix, die auf das Bild angewendet wird.

Das Ziel ist es den Gradienten, also die Steigung des oben anschaulich dargestellten Grauwertverlaufs festzustellen (Schematische Kantendetektion). Da uns im Allgemeinen ein zweidimensionales Bild vorliegt, nutzen wir zusätzlich zu dem eindimensionalen Grauwertverlauf in x-Richtung einen weiteren (eindimensionalen) Grauwertverlauf in y-Richtung. Solch ein Grauwertverlauf wird nun wie folgt beschrieben:

$$G(f(x,y)) = \begin{pmatrix} G_x \\ G_y \end{pmatrix} = \begin{pmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{pmatrix} = \begin{pmatrix} \lim\limits_{\triangle x \to 0} \frac{f(x+\triangle x,y)-f(x,y)}{\triangle x} \\ \lim\limits_{\triangle y \to 0} \frac{f(x,y+\triangle y)-f(x,y)}{\triangle y} \end{pmatrix}$$

Eine relativ einfache Möglichkeit ist es das $\triangle x$ sowie $\triangle y = 1$ zu setzen. Da dies auch recht sinnvoll ist, wird durch die Überlegung klar, dass das kleinste Element eines Bildes genau ein Pixel ist. Schritte wir 0.5 sind daher wenig sinnvoll um Gradienten zu berechnen. Durch das Setzen zu 1 gelangen wir zu folgender Formel:

$$G(f(x,y)) = \begin{pmatrix} f(x+1,y) - f(x,y) \\ f(x,y+1) - f(x,y) \end{pmatrix}$$

Vereinfacht können wir sie durch folgende Matrizen ersetzen:

$$G_x = \begin{pmatrix} -1 & -1 \end{pmatrix} \quad G_y = \begin{pmatrix} -1 \\ 1 \end{pmatrix}$$

Da es kritisch sein kann, dass beide Operatoren (sowohl in x- als auch in y-Richtung) auf der exakt gleichen Bildposition arbeiten wird diese Matrix durch eine 2x2-Matrix ersetzt.

$$G_x = \begin{pmatrix} -1 & 1 \\ -1 & 1 \end{pmatrix} \quad G_y = \begin{pmatrix} 1 & 1 \\ -1 & -1 \end{pmatrix}$$

Der zu berechnende Gradient liegt nun exakt in der Mitte der 2x2-Pixelmatrix. Noch wesendlich besser ist hier eine 3x3-Matrix. Eine derartige Matrix verwenden auch die zu Anfang erwähnten Sobel- und Prewitt-Operatoren.

Prewitt-Operator:

$$P_x = \begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix} \quad P_x = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{pmatrix}$$

Sobel-Operator:

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

2.4 Nonmaxima Suppression

Unterdrückung von „Nicht-Maxima“
Da wir lediglich an Kanten eines Bildes interessiert sind, enthält unser zu verarbeitendes Bild noch viele nutzlose Informationen (z.B. große Flächen eines Grauwertes, etc.). Diese Informationen sind für die Kantenerkennung nicht von Bedeutung, da es uns nur auf starke Änderungen im Grauwert ankommt. Ein Nonmaxima-Supression-Algorithmus hat nun zwei Aufgaben. Die eine, eben genannte, und eine Verstärkung des Grauwertgradienten (unterschied zweier benachbarter Grauwerte) einer Kante.

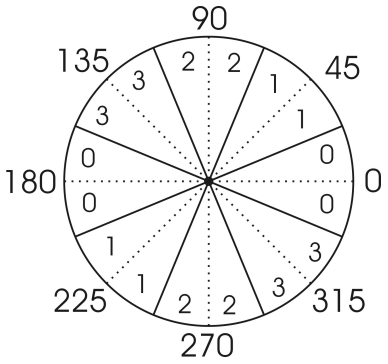


Abbildung 3: Kantenorientierung

Zunächst untersucht dieser Algorithmus die Orientierung einer Kante. Eine Kante kann, wie in Abbildung 3 ersichtlich, lediglich in vier verschiedene Richtungen orientiert sein. Die Orientierung bekommt man aus gleichen Grauwertgradienten bei Untersuchung einer 3x3-Matrix. Falls der Gradient des mittleren Pixels (der 3x3-Matrix) jedoch kleiner ist, als die beiden gegenüberliegenden, wird der mittlere Pixel nicht mehr als Kantenzugehörig betrachtet und somit zu 0 gesetzt.

Mit Hilfe dieses Verfahrens erhalten wir Kanten eines Pixels breite. Dieser Algorithmus ist zur Kantendetektion allein jedoch noch nicht zu gebrauchen, da in dem Bild immer noch überflüssige Informationen vorhanden sind. So werden z.B. leichte Texturen (also Färbungen geringer Grauwertänderung) als Kanten erkannt. Diese müssen wir abschließend noch versuchen herauszufiltern.

2.5 Thresholding

Ein Threshold-Algorithmus wird benutzt um die Anzahl der falschen Kanten zu minimieren. Einfach gesagt werden alle Pixel, die unterhalb eines gewissen Wert liegen zu 0 gesetzt und somit aus dem Bild eliminiert. Das größte Problem hierbei ist es diesen „Grenzwert“ zu finden. Setzten wir ihn zu niedrig, bleiben weiterhin falsche Kanten im Bild. Setzten wir ihn jedoch zu hoch, so fallen gewollte Kanten aus dem Ergebnis heraus. Man sieht, dass die Wahl eines geeigneten Granzwertes nicht trivial ist. Um trotzdem recht genaue Ergebnisse zu erhalten, wendet man den Algorithmus doppelt an. Man wählt nicht einen,

sondern zwei Grenzwerte. Ungefähr der Art:

$\tau_1 \approx 2\tau_2$

Daraus ergeben sich zwei neue Bilder. Das Erste besitzt zu viele und das Zweite zu wenig oder nur unvollständige Kanten. Nun betrachtet der Algorithmus das zweite (unvollständige) Bild und untersucht jeweils anhand einer 3x3-Matrix eine Kantenunterbrechung. Wenn eine solche gefunden wurde, wird das erste Bild betrachtet. Wird in diesem an gleicher Position die Kante Festgestellt, so wird sie im unvollständigen ergänzt. Dieser doppelt angewandte Algorithmus verringert das Problem nach einer geeigneten Suche eines einzelnen Grenzwertes enorm, da ein leichter Fehler durch das Kombinieren zweier Bilder wieder leicht ausgemerzt wird.

2.6 Kantendetektion nach J.F. Canny

Die Kantendetektion nach Canny läuft nun mit Hilfe der vorgestellten Algorithmen ab. Dieser Kantendetektor ist in der Bildverarbeitung einer der meist beschriebenen. Er vereinigt im Wesentlichen folgende vier Algorithmen:

- Glättung des vorliegenden Bildes mit Hilfe eines Gauss-Filters
- Berechnung der Gradienten in x- und y-Orientierung (z.B. mit dem Sobel-Operator)
- Anwendung von Nonmaximal Suppression
- Benutzung des Thresholding-Algorithmus zur Bestimmung der Kanten und ihrer Verknüpfung untereinander

2.7 Farbe und Textur

Zu Farbe und Textur könnte man ganze Bücher schreiben. Dies Thema allein wäre in unseren Augen eine eigene Abhandlung wert, deshalb möchten wir in Bezug auf Kantendetektion nicht näher eingehen. Folgende Überlegung sei jedoch gesagt. Wenn wir in der Lage sind Kanten aus einem Grauwertbild erkennen, dann sind wir ebenfalls in der Lage Kanten in einem Bild zu erkennen, das aus verschiedenen Rot-Tönen besteht. Da sich jedes Bild in seine Grundfarben zerlegen lässt (im RGB-Format wird es in dieser Weise gespeichert), kann man die Kantendetektion dreifach (d.h. einmal auf jeder Grundfarbe) ausführen und die Ergebnisse vergleichen. Befindet sich an einer bestimmten Position aller drei Einzelbilder eine Kante, so kann man davon ausgehen, dass es sich hierbei um eine echte Kante handelt.

3 Extraktion von 3D-Informationen

„Es bleibt ein Geheimnis, wie wir Menschen die Bilder, denen wir Tag für Tag gegenüberstehen, erfassen, analysieren und klassifizieren.“[Dew95, Seite 130] Wie sollen wir nun unsere, nicht verstandene Fähigkeit auf eine Maschine übertragen? Es existieren einige viel versprechende Ansätze hierzu. Jedoch gibt es bis heute keine Patentlösung. Verschiedenen Kantendetektoren gelingt es zwar recht beeindruckende Ergebnisse zu präsentieren, jedoch müssen die Informationen im Weiteren gedeutet werden. Ein Ansatz kommt aus dem Jahr 1975.

3.1 Waltz-Algorithmus

Waltz beschäftigte sich 1975 bereits damit dreidimensionale Informationen aus zweidimensionalen Bildern zu erkennen. Er untersuchte die Eigenschaften der Kanten in einem Würfel und klassifizierte sie als konvex bzw. als konkav (hierbei bezeichnet „konkav“ eine vertiefte, konvex eine spitze Kante).

Das Ziel dieses Algorithmus ist es nun alle Kanten eines Objekts als konvex/konkav zu erkennen und somit zu bestimmen ob eine entsprechende Kante innen oder außen liegt. Um die Kante als innen bzw. Außenkante zu erkennen, bestimmte Waltz zunächst alle möglichen Ecken (Abbildung 4).

Abbildung 4: Mögliche Ecken

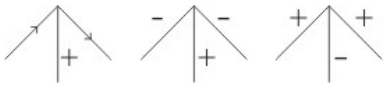
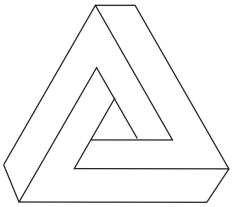


Abbildung 5: Kantenorientierung für ’c’

Anschließend bildete er alle dazugehörigen Kantenmöglichkeiten (illustriert an Kante „c“ Abbildung 5). Eine Außenkante ist hier mit einem „+“ versehen. Eine Innenkante dementsprechend mit einem „-“. Die Pfeile bedeuten, dass dies eine Objektbegrenzungskante ist. Für alle oben abgebildeten Kanten (a-d) gibt es insgesamt achtzehn mögliche Kantenbelegungen. Anhand dieser Informationen kann man nun, durch logische Kombination, die Orientierung einzelner Kanten bestimmen (siehe Abbildung 6). Leider ist dieser recht einfache und zugleich mächtige Algorithmus ungeeignet, wenn es sich um Objekte mit Bögen oder Kreisen handelt. Um dies trotzdem erledigen zu können, muss er erweitert werden. Beispielsweise kann man durch eine geeignete Approximation eine runde Kante durch mehrere Ecke ersetzen. Ebenfalls ungeeignet ist er, wenn das zu rekonstruierende Objekt Ecken vom Grad >3 hat oder gar eine Form, die in Wirklichkeit nicht existieren könnte (siehe optische Täuschung rechts). Ein weiteres, generelles Problem der Bilderkennung ist die Bilddeutung. Es gibt mehrere Körper, die ebenfalls im Gebiet der Optischen Täuschung zu finden sind, die in der Realität zwar existieren, bei denen wir, als dreidimensional sehende Menschen, uns nicht sicher sein können, ob es sich nun um eine Kante handelt, die vorne oder eher hinten im Bild liegt. Eine Möglichkeit sich diesem Problem zu stellen, ist den Waltz Algorithmus in so fern zu erweitern, dass er einen eventuell vorhandenen Schatten mit in seine Berechnung mit einbezieht. Dadurch vergrößert sich die Anzahl der möglichen Kantenausrichtungen. Man kann sich dies leicht an den drei oben Illustrierten Kantenausrichtungen klar machen. Die Anzahl dieser drei vergrößert sich um die möglichen Schatten-Permutationen.



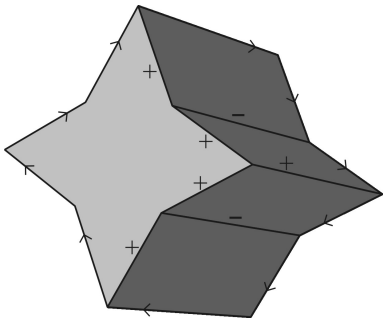


Abbildung 6: Festgestellte Orientierung

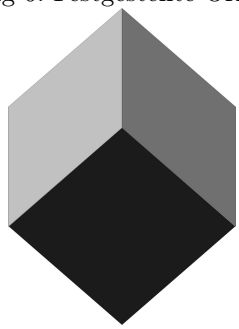


Abbildung 7: Objekt mit Schatteninformation

3.2 $2\frac{1}{2}$-Dimensionale Bilder

Unter $2\frac{1}{2}$-Dimensionalen Bildern versteht man zweidimensionale (also handelsübliche) Bilder, die dreidimensionale Informationen enthalten. Diese Informationen können zum Beispiel aus Schattierungen und Perspektive bestehen.

3.2.1 Schattierung

Mit einer Erweiterung des Waltz-Algorithmus ist es nun möglich diese Informationen ebenfalls mit in die Bewertung von Kantentypen einzubringen, und so mögliche Kantenarten zu begrenzen. Weiterhin ist es auch möglich mit Hilfe von Schatten Rundungen zu erkennen wie in folgender Abbildung 8 illustriert.

3.2.2 Perspektive

Eine weitere Information kann man aus perspektivischen Bildeigenschaften extrahieren. Diese Information ist jedoch nicht immer verlässlich wie man sich leicht vorstellen kann. Hierbei sind Runde Körper recht problematisch. Eine Kugel besitzt nicht wie beispielsweise ein Würfel einen eindeutig zu erkennenden Bezugspunkt. Ein flaches rundes Objekt (wie zum Beispiel eine CD) bildet bei perspektivischer Darstellung eine elliptische Form. Jedoch lässt sich aufgrund dieser Form allein kein eindeutiger Bezugspunkt feststellen wie das bei einem Würfel möglich wäre. Aufgrund dieses Schönheitsfehlers spielt die Perspektive eher eine untergeordnete Rolle.

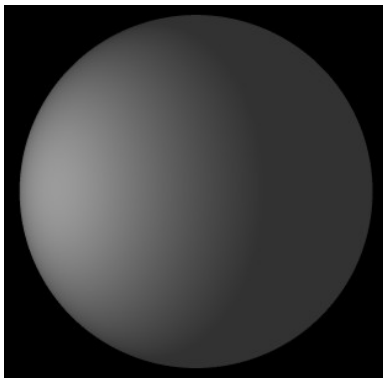


Abbildung 8: 2 $\frac{1}{2}$ -Dimensionale Kugel

3.2.3 Textur

Auf Textur möchte ich hier auch weniger eingehen. Jedoch ist sie zweifellos ein untrennbarer Teil eines fast jeden Bildes. Wenn man von Textur spricht meint man im Allgemeinen die Färbung und (meistens) ein Muster einer Fläche, die durch ihre Kanten begrenzt ist. Genau hier fangen die Schwierigkeiten jedoch schon an. Angenommen wir haben eine sehr strukturstarke Textur. Daraus Folgt unter Umständen, dass der normalerweise recht gut funktionierende Kantendetektor fälschlicherweise ein Teil der Textur als Kante erkennt. Somit liefert der auf Kanten basierende Waltz-Algorithmus ebenfalls ein fehlerhaftes Ergebnis und somit ein falsch geformtes Objekt. Wie an diesem Beispiel ersichtlich, ist das Thema Textur, bezogen auf Kantendetektion ein weiteres, zu tief gehendes Kapitel.

3.3 Orientierung von Objekten

Wir wollen nun ein Objekt im dreidimensionalen Raum betrachten. Jeder Pixel kann bei mathematischer Betrachtung durch seine drei Koordinaten eindeutig einer Position im Raum zugeordnet werden. Durch eine (handelsübliche) Aufnahme, ganz gleich ob durch eine Kamera oder ein Foto, bekommen wir jedoch lediglich eine zweidimensionale Abbildung des Objektes. Jedes Objekt hat genau genommen zwei Orientierungen.

1. die Orientierung des Objekts selbst
2. die Orientierung der Oberfläche

Angenommen wir fotografieren ein Objekt aus einem bestimmten Winkel. Anschließend machen wie ein zweites Bild von dem Objekt, dass wir jedoch um wenige Grad gedreht haben. Wie soll ein Programm nun entscheiden, ob es sich hierbei um das gleiche Objekt wie in Bild eins handelt, jedoch um ein paar Grad gedreht? Was ist, wenn wir nicht das Objekt, sondern die Kamera um das Objekt bewegen? Unter Umständen erhalten wir eine vollkommen

andere Umgebung und somit ein komplett anderes Foto. Angenommen wir haben nicht zwei vollkommen verschiedene Fotos, sondern ein Magazin von leicht unterschiedlichen Aufnahmen eines Objekts (z.B. ein Film in dem das Objekt mit konstanter Geschwindigkeit gedreht wird). Um daraus die Tiefe einzelner Bildpunkte zu bestimmen markiert man nun gewisse Punkte im Bild (Konkav-, Konvex-Ecken sowie Kantenverbindungen im Objekt).

Wendet man nun dieses Verfahren auf das nächste Bild an, so kann man daraus die relative Bewegung der markierten Punkte erkennen. Schließlich lässt sich aus dieser Bewegung die Tiefe bestimmen (in der Literatur bekannt unter „Courtesy of Carlo Tomasi“). Um diese doch aufwendige Berechnung etwas zu minimieren, existieren weitere Verfahren um an dreidimensionale Eigenschaften von Objekten zu gelangen. Man kann beispielsweise mittels einer gezielt projizierten geraden „Linie“, etwa eines Laser-Strahls, die räumliche Position der Pixel bestimmen. Durch die unterschiedlichen Tiefen des Objektes wird der Strahl nun in unterschiedlicher Krümmung reflektiert. Die aufnehmende Kamera muss hierbei natürlich in einem gewissen Abstand zum Laser stehen, um die Krümmung überhaupt als solche wahr zu nehmen (Anmerkung: Auf diese Weise arbeitet ein Laser basierter 3D-Scanner; siehe Abbildung 9). Eine weitere Möglichkeit besteht darin, statt eines Lasers eine zweite Kamera zu nehmen. Somit ist es aufgrund des unterschiedlichen Aufnahmewinkels die voneinander abweichenden Pixelpositionen und somit den Tiefenunterschied derer zu bestimmen.

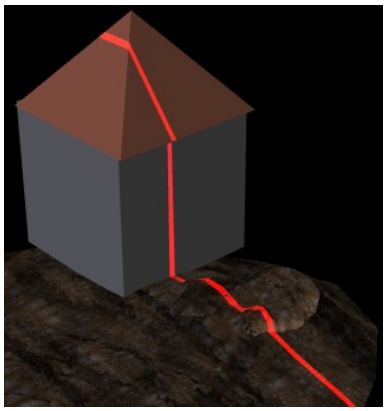


Abbildung 9: 3D-Extraktion mittels Laser

4 Objekterkennung

Die Objekterkennung gehört in ihrer Anwendung mit Sicherheit zu den Interessantesten Forschungsbereichen der Künstlichen Intelligenz - *Bildverarbeitung*. Zugleich ist sie eines der schwierigsten Teilbereiche der Künstlichen Intelligenz. Von der Objekterkennung wird nicht nur gefordert, dass Objekte als solche erkannt werden (Katze, Mensch, ein Glas Wasser, eine Banane, etc.), sondern beschrieben werden (eine weiße junge Katze läuft über eine Straße, etc.). Zum

Teil hat die Objekterkennung schon in unser Leben Einzug erhalten. So finden wir heutzutage zum Beispiel handschrifterkennende (Schreibschrift) Computer (sog. PDA's) oder Maschinen, die die handgeschriebene Anschrift eines Briefes erkennen und automatisch in das entsprechende Fach sortieren.

Ein weiteres Einsatzgebiet wird in sehr naher Zukunft auf Flughäfen eingesetzt. Hier versucht man mit Hilfe der Videoüberwachung ein- und ausreisende Personen mit einer Verbrecherkartei zu vergleichen, diese automatisch zu erkennen und, durch die Polizei, zu überführen. Ein erster Pilotversuch läuft seit Januar 2003 auf dem Flughafen Berlin-Tegel. In den Vereinigten Staaten wird ab dem 01.01.2004 jeder Einreisende fotografiert. Dieses Foto wird anschließend mit einer Verbrecherdatenbank verglichen. Dies geschieht ebenfalls mit den Fingerabdrücken der einreisenden Person. Ein großer Forschungsbereich ist die Entwicklung von sog. „Software-Agenten". Diese Agenten sind u. A. in der Lage das Internet intelligent zu durchsuchen. Bei einem Suchbegriff wie „Albert Einstein" werden zurzeit lediglich Texte mit diesem Inhalt gefunden. Zukünftig sollen ebenfalls Bilder/Videos mit den abgebildeten Objekten (in diesem Fall „Albert Einstein") gefunden werden. Das Prinzip eines Agenten werden wir im Teil Robotik erläutern.

4.1 Zur Geschichte...

Man begann damit Objekte anhand festgesetzter Merkmalsvektoren zu beschreiben. Diese Vektoren kann man mit Hilfe der Kantendetektion finden. Diese verglich man nun mit einer Sammlung von Merkmalsvektoren verschiedener Objekte und konnte so mit gewisser Wahrscheinlichkeit feststellen um was es sich handelte. Die Versuche waren zunächst lediglich auf Druckschrift-Erkennung beschränkt. Nach einiger Zeit wurde klar, dass sich so zwar Buchstaben und zweidimensionale Objekte über Vektoren erkennen ließen, aber der Versuch viele und Komplexe Objekte zu erkennen war nicht von Erfolg gekrönt. 1965 begann L.G. Roberts mit der Erforschung seiner sog. Blockswelt (siehe Abbildung 10). Mit der Analyse solcher Polyederszenen versuchte er nicht Objekte zu klassifizieren, sondern zu beschreiben. Eine entscheidende Rolle spielte hierbei auch der Waltz-Algorithmus von 1975 (siehe weiter oben: „Waltz-Algorithmus").

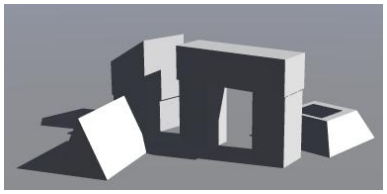


Abbildung 10: Blockswelt-Szene

4.2 Was mach Objekterkennung so schwierig?

Objekterkennung ist kein einzelner Teilbereich der künstlichen Intelligenz. Objekterkennung ist ein stark bereichsübergreifendes Gebiet der Forschung. Man

kann sich leicht vorstellen, dass ein Algorithmus, der nicht nur eine Mandarine von einer Orange unterscheiden soll, sondern auch herausfinden soll wie die schöne Frau auf dem Bild aus dem Internet heißt, kein triviales Unterfangen sein kann.

4.3 Algorithmenansatz

Es gibt noch keine entdeckte Patentlösung zur Objekterkennung. Man nähert sich der Lösung dieses Gesamtproblems mit verschiedenen Ansätzen. Zunächst sollten wir das Objekt soweit wie möglich beschreiben. Die Hilfsmittel, die uns bisher zur Verfügung stehen, bieten dabei eine recht fundierte Basis. Wir könnten zunächst einen Kantendetektor (z.B. nach Canny) auf das entsprechende Objekt anwenden. Anschließend lässt sich mit einem Waltz-Algorithmus bzw. über Methoden der Extraktion von 3D-Informationen recht gut bestimmen wie das entsprechende Objekt einzuordnen ist. Bevor man versucht 3D-Informationen zu erhalten, sollte jedoch geklärt werden, ob die entsprechende Vorlage diese Informationen überhaupt hergibt bzw. ob sie benötigt werden denn um eine Handschrift zu erkennen braucht man keinen zusätzlichen Algorithmus zur Bestimmung der Tiefe des Objekts. Wenn wir nun das Ergebnis des Kantendetektors analysieren, können wir einen Markierungs-Algorithmus darauf anwenden (siehe hierzu nähere Erläuterung in „Orientierung von Objekten“).

Dieser Algorithmus sucht nun konkave bzw. konvexe Ecken und markiert sie (z.B. mit einem kleinen Kästchen). Anhand der relativen markierten Punkte des Bildes kann man nun, durch entsprechende Skalierung, die Größe des Objektes für die Objekterkennung außer Acht lassen. Ein zusätzlicher Vorteil dieser Methode ist die relative Unabhängigkeit der Beleuchtung (natürlich ist dieses Verfahren auch nicht in der Lage aufgrund einer mangelhaft beleuchteten Vorlage exakte Ergebnisse zu präsentieren, jedoch ist es nicht so dramatisch wenn das Objekt nicht von weißem Licht, sondern eher von gelbem beleuchtet wird). Der große Nachteil liegt hier aber ebenfalls auf der Hand.

Die Textur bzw. Färbung des Objektes wird zunächst nicht betrachtet. Somit kann es natürlich passieren, dass eine Mandarine als Tischtennisball erkannt wird und somit das Objekt bei Klassifizierung, nicht unter Nahrungsmittel sondern unter Sport einsortiert wird. Und somit kommen wir zum nächsten Teil des gedachten Algorithmus.

Wir versuchen nun, basierend auf unserer Objekt-Datenbank ein entsprechendes Objekt zu finden, das unserem, zu erkennenden, recht nahe kommt. Auf derartiger Basis wird versucht zu brauchbaren Ergebnissen zu kommen. Es ist ersichtlich, dass diese Methode (wie auch im Text erwähnt) Nachteile besitzt. Jedoch ist bisher, wie bereits erwähnt, noch kein wirklich zuverlässiger Algorithmus bekannt, mit dem man das gesamte Gebiet der Objekterkennung abdecken könnte. Unser gedachter Algorithmus besitzt (abgesehen von einigen kleinen Feinheiten) große Ähnlichkeit mit dem sog. ALIGN-Algorithmus, wie er in der Literatur beschrieben wird.

Robotik

5 Einleitung

Die Robotik entstand aus der Motivation heraus, dem Menschen anstrengende, eintönige oder auch gefährliche Arbeit abzunehmen. Heutzutage findet man in allen möglichen Bereichen des täglichen Lebens Roboter, die man allgemein in drei Hauptgruppen unterteilen kann. Zum Einen gibt es Roboter als Programme, die nach gewissen Algorithmen Aufgaben erledigen, zum Zweiten sind das Industrie und Serviceroboter, die hauptsächlich in der Industrie zum Einsatz kommen und mechanische Arbeitsabläufe verrichten und schließlich die Androiden als Krönung des derzeitigen Entwicklungsstandes.

Als aktuellstes Beispiel sei hier verwiesen auf die Marsroboter, die für den Menschen selbstätig Eindrücke sammeln, und dies in einer Umgebung, die kein Mensch zuvor betreten hat. Dieser letzte Teilbereich der Androiden ist wohl der Aufsehenerregenste, weil hier versucht wird den Menschen in seiner Gesamtheit aufgrund mathematischer sowie logischer Grundsätze zu imitieren.

„Robotik ist die umfassende Disziplin, die wesentliche Teilbereiche der Gebiete KI, Kognitionswissenschaft und Aktor-Design zusammenführt und in einem technischen System operationalisiert, welches komplexe Handlungen in unterschiedlichen Umwelten autonom ausführen kann, um damit zielgerichtet Manipulationsaufgaben ausführen kann“. Handbuch der künstlichen Intelligenz, 3. Auflage, Oldenbourg.
In den folgenden Abschnitten wird versucht, auf das Problem der Wahrnehmung von Robotern sowie dem Planen und Bewegen einzugehen. Der erste Bereich setzt sich mit der Wahrnehmung auseinander und differenziert hierbei zwischen stationären Robotern wie sie in industriellen Fertigungsanlagen gefunden werden können und mobilen Robotern, da die Fähigkeit der Wahrnehmung hier relativ unterschiedlich ausfallen. Eine besondere Beachtung genießen Androiden, da menschenähnliche Roboter ohne Zweifel die komplexesten Roboter sind. Speziell wird hierbei die Funktion von Agenten beleuchtet und dadurch übergeleitet auf den zweiten Teil, der Planung und Bewegung von Robotern.

6 Wahrnehmung

6.1 stationäre Roboter

Bei stationären Robotern ist in erster Linie der fehlerfreie Bewegungsapparat, die Fähigkeit Gegenstände genau zu lokalisieren und sie zu verfolgen von Bedeutung. Von besonderem Interesse im Hinblick auf die Zukunft wird hierbei die erleichterte Benutzersteuerung sein, also der Programmierung durch Mustererkennung, sowie das Vermögen des Roboters sich eigenständig an ungewohnte Situationen zu gewöhnen, und zu lernen. Die Wahrnehmung stationärer Roboter ist relativ beschränkt, da Ihre Bewegungsabfolgen so geplant wurden, daß es zu keinerler ungewollter Berührungen kommen kann. Daher gehört zur Wahrnehmung eines stationären Roboters zunächst die Planung der Bewegungen,

genannt Aktorik. Das Planen der Aktorik lässt sich gliedern in die Montagesequenzplanung sowie die Bewegungs- und Griffplanung.
Die Montagesequenzplanung beschäftigt sich mit dem Problem, zu welcher Zeit welches Objekt zum Einsatz kommt. Hierzu ist es notwendig, dass der Roboter dynamische Umgebungen erkennen kann sowie die Objekte selbst anhand ihrer geometrischen Struktur oder physikalischer Größen eindeutig erkennen kann.
Die Bewegungs- und Griffplanung beschäftigt sich mit der Thematik hoher Geschwindigkeit, Bahntreue und gleichzeitig geringe Belastung der mechanischen Abläufe sowie natürlich Kollisionen.
Die Sensoreinsatzplanung beschäftigt sich mit der Aktivierung und der Festlegung von Parametern der wahrnehmenden Sensoren um als Ziel die Erkennung eines Objektes zu haben. Eine solche Planung ist eine erste Voraussetzung, redundante Informationen zu erkennen und zu eliminieren.
Die Tatsache, daß redundante Sensoraufnahmen verarbeitet werden ist keineswegs Zufall. Sollte nämlich einem Sensor die Sicht versperrt sein, kann die Systemsteuerung auf andere Sensoren zurückgreifen. Die optimale Sicht wird dabei sehr oft probabilistisch über die Fuzzy-Regelbasis bestimmt. Dies führt zur Datenfusion um aufgrund verschiedenster Sensoraufnahmen ein Gesamtbild zu erzeugen, Fehler zu minimieren und redundante Informationen zu kompensieren.
Ein relativ aktuelles Beispiel dieser Technik ist die sogenannte „Bullet-Time“ aus der Matrixreihe, wo aus einer Vielzahl von Einzelbildern eine Abfolge von Bildern erzeugt wird. Die Schwierigkeit besteht hierbei beim Filtern der sachdienlichen Informationen aus den Einzelaufnahmen. Dies geschieht über die Mustererkennung von Objekten der Einzelbilder.
Ein sehr wichtiger Aspekt der maschinellen Wahrnehmung ist der Erwerb und die Generalisierung von Fertigkeiten eines Roboters. Diese Idee beruht auf dem Wunsch, dass stationäre Fertigungsroboter nicht programmiert werden müssen um eine Bewegungsabfolge auszuüben, sondern daß diese aufgrund Ihrer Beobachtung diese Schritte lernen. Der Roboter beobachtet über Kameras einen menschlichen Instrukteur, der ihm die Fertigung eines Objektes aus mehreren Einzelkomponenten vormacht. Der Roboter beobachtet speziell die notwendigen Einzelteile und speichert, in welcher Weise diese Komponenten zusammengebaut werden müssen. Für (relativ) einfache Produkte ist dieses Verfahren schon gezeigt worden. Für komplexe Fertigungsprozesse allerdings scheitert dieses Verfahren bisher aber daran, dass der Roboter die innere Struktur der Einzelteile nicht erkennen kann. Er kennt nicht die physikalischen Eigenschaften und er erkennt auch nicht die versteckten Eigenschaften des Instrukteurs, der manche Fertigungsschritte aus seiner Erfahrung heraus macht. Auf dieses Wissensrepertoire kann die Maschine nur aus der Beobachtung heraus nicht zugreifen.

In der Praxis steuert eine Dialogkomponente den gesamten Fertigungsablauf. Sie überwacht die verarbeiteten Sensoraufnahmen, Roboterzustände sowie Eingaben des Instrukteurs. Um hierbei ein Nachvollziehen seitens des Instrukteurs zu ermöglichen sollten diese Komponenten nach den Prinzipien Inkrementalität sowie Interaktivität aufgebaut sein. Diese beiden Prinzipien bedeuten, daß der Roboter mit seinem Instrukteur in Kontakt steht und sein Lernprozeß linear fortschreitet. Die Inkrementalität bedeutet, dass Daten seitens der Dialogkomponente sofort interpretiert werden, ohne auf das Ende des Daten zu warten. Bei der sprachlichen Eingabe bedeutet dies konkret, dass nicht auf das Satzende gewartet wird.

Die Interaktivität gewährleistet eine technische Implementierung der Datenfusion. Das bedeutet, dass die Teildaten verschiedener Quellen zusammengeführt werden, um ein Gesamtergebnis zu erhalten. Hierbei besteht das Gesamtsystem aus mehreren Agenten die ihrerseits der Dialogkomponente Ergebnisse zurückliefern. Der Datenaustausch innerhalb des Systems erfolgt durch ein dynamisches Netzprotokoll. Die sensorischen und motorischen Komponenten des Robotersystems dienen zur Erkennung von (sprachliche) Eingaben, oder des aktuellen Zustandes der Verarbeitung, insgesamt also zur Abtastung der Umwelt und physikalischer Kräfte. Die Steuerung der Sensoren wird vom Sensoragenten geregelt. Agenten liefern also jeden exklusiven Zugriff auf die an Ihnen angeschlossene Systeme und besitzen ferner die Funktionalität, diese Rohdaten vorzubearbeiten.
Der Vorteil für den Anwender dieses sogenannten dynamischen Multi-Agent-Systems ist klar: Da das System eigenständig Anfragen auf seine Agenten machen kann, Agenten aber ebensogut eigenständig Rückmeldungen machen können oder temporäre Agentenverbände schließen kann muß dem Anwender nicht bekannt sein, welche Agenten nun interagieren, welche im Ruhezustand sind etc. Der Roboter konfiguriert sich selbst. Die zukünftigen Roboter werden vermehrt über verschiedenste Sensoren wie Kameras, Laser etc. verfügen, was dadurch begründet werden kann, daß die Rechenleistung moderner Chips die Verarbeitung komplexer Informationen erlaubt und der Preisverfall bei Sensorsystemen derartige Roboter für Betriebe möglich macht. Dadurch werden Robotersystem Einzug in Fertigungshallen erhalten, die auf Gestik, Mimik und Sprache reagieren können.

6.2 Mobile, verhaltenstreue Roboter

Bei der „Verhaltensbasierten KI “ geht es um den Versuch, Intelligenz als solches in unserer Umwelt einzuordnen und auf den Computer zu übertragen. Dies wirft eine sich über die Informatik hinausreichende, philosophische Frage auf. In welcher Form existiert Intelligenz und unter welchen Vorraussetzungen kann Intelligenz überhaupt existieren? Durch Roboter wird die Frage der Existenz von Intelligenz jedoch zum ersten Mal in der Geschichte der Menschheit in einem Atemzug mit Maschinen erwähnt. Alles in allem kann man feststellen dass Intelligenz in der Natur nie ohne einen Körper existieren kann.

Geht es also darum, ein künstlich intelligentes System zu erschaffen geht das nur auf der Grundlage von Robotern. Als Ziel des fehlerfreien Betriebs eines Roboters in seiner Umwelt geht man weg von der Idee die Natur soweit zu abstrahieren bis sie für einen Computer sensorisch erfassbar ist. Es geht darum, die sensorischen Fähigkeiten so zu verfeinern, dass sie sich auf die Umwelt und die ständig auftretenden Änderungen einstellen. Somit interagieren dann Maschine und Umwelt wobei beide ständig selbständig ihre Zustände ändern. In diesem Zusammenhang wurden Roboter erschaffen, die sich in Ihrer Umwelt ähnlich einem tierischen Vorbild bewegen können. Hierbei setzt sich ein solches System aus kleineren Einheiten zusammen die durch sensomotorische Rückkopplungsschleifen zielgerichtet einfachste Bewegungsabläufe realisieren.
Das Problem hierbei ist einmal mehr die Komplexität dieses Gesamtsystems. Man weiß, dass bei einem derartigen System 20 bis 40 dieser einzelnen Prozessstränge, auch Threads genannt, auftreten können, die reibungslos miteinander

interagieren müssen. Steigt die Anzahl derer in realisierten Systemen allerdings, so steigt auch die Komplexität und der Aufwand der Gesamtheit unkontrollierbar für den kontrollierenden Menschen. In der verhaltensbasierten KI wird ein Roboter also dadurch programmiert, dass Einzelmechanismen gebaut und implementiert werden, die zusammen Verhaltenssysteme ergeben.

6.3 Menschenähnliche Roboter

Menschenähnliche Roboter, auch Androiden genannt sind zweifelsohne die allgemeinhin bekannteste Materie der Künstlichen Intelligenz und stellen eine Spezialisierung autonomer mobiler Roboter dar. Dies ist zum einen dadurch zu erklären, dass Androiden auch für materiefremde Menschen den Begriff der künstlichen Intelligenz fassbar machen. Zum anderen muss ein Androide eine Vielzahl von Funktionen haben um dem Menschen so ähnlich erscheinen zu können wie möglich. Um dieses gewaltige Gesamtsystem aus Prozessoren, Sensoren und Algorithmen beherrschbar zu machen, wird ein autonomer mobiler Roboter dezentral gesteuert.
Eine Zentrale Steuereinheit verwaltet nur das nötigste selbst und verteilt die Aufgaben weiter. Die einzelnen Komponenten des Roboters verfügen über so genannte Agenten, welche die an Sie angeschlossenen Geräte verwaltet und gegebenenfalls Informationen von anderen Komponenten aufnehmen aber genauso gut Informationen an andere Komponenten senden kann. Auf diese Weise ist auch eine direkte Kommunikation zwischen verschiedenen Komponenten möglich und entlastet die zentrale Steuereinheit.
Der momentane technische Entwicklungsstand in diesem Teilbereich der künstlichen Intelligenz geht vor allen in die Richtung der Unvorhersehbarkeit, der Unsicherheit und der Veränderung. Ein Androide muss in der Lage sein sich auf ein schnelles Ändern von räumlichen Situationen einstellen können und darüber hinaus in gewissen Situationen aufgrund früherer Erfahrungen Vorhersagen zu machen. Genau diese Themen werden aktuell diskutiert.

7 Planen und Bewegen

7.1 Einleitung

Das Bewegen ist die herausfordernste Teildisziplin, da in Ihr viele andere Komponenten der künstlichen Intelligenz zum Tragen kommen: Das Erkennen, das Handeln, das Planen, die Architektur, die Hardware und die Problemlösung. Die zwei Prinzipien, nach denen ein beweglicher Roboter den Weg zu seinem Ziel plant heissen Topologisches navigieren sowie metrisches Navigieren. Diese beiden Prinzipien leiten über zu der Frage, wie ein Roboter daraus eine Art Landkarte für sich erstellt und sich darin zurechtfindet. Die iterative Abfolge beim der Bewegung lautet :

- Wo will ich hin?
- Was ist der beste Weg dorthin?

- Wo war ich schon?
- Wo bin ich?

„Wo will ich hin?“. Diese Aufgabe wird meist von einem Menschen eingegeben oder aber dem Roboter in einer gewissen Form eingespeichert. „Was ist der beste Weg dorthin?“ Dieser Iterationsschritt bezieht sich auf die Pfadplanung. Um die Pfadplanung so schnell und so zuverlässig wie möglich zu machen müssen einige Sachen beachtet werden. Zunächst einmal kostet die Suche Speicher sowie Rechenzeit. Somit wird eine geringe Komplexität der Suche angestrebt. Ein weiterer Aspekt eines robusten Algorithmusses ist, ob er bei der Planung auf die Beschaffenheit des Terrains eingeht. Es muß beachtet werden, dass manche Wege aufgrund ihrer Bodenstruktur nicht zum gesuchten Ziel führt. Ein aktuelles Beispiel hierfür sind die Marsroboter, die sich Ihren Weg durch zerklüftetes Gebiet bahnen können müssen. Eine einfache binäre Entscheidung, ob ein Weg passierbar ist oder nicht ist nicht ausreichend. Der Algorithmus muß imstande sein, die Passierbarkeit einzelner Terrainstrukturen mehr zu differenzieren.
Hier kommt nun eine weiter Entscheidung zum Tragen, nämlich die der Beweglichkeit des Roboters selbt. Im Bezug auf Pfadplanung ist die größte Frage, ob der Roboter auf der Stelle wenden kann. Verfügt der Roboter nicht über diese Fähigkeit muß in die Pfadplanung einberechnet werden, in welcher Richtung er gerade sieht und ob das wenden oder das Drehen in eine gewisse Richtung möglich ist. Da diese Information zusätzlichen Aufwand verursacht, werden bewegliche Roboter meist rund gebaut um beim Drehen um die eigene Achse nicht irgendwo anzustoßen.
Schließlich muß der Algorithmus zur Planung noch über die Funktion verfügen, die anfangs übergebene Landkarte zu verifizieren und gegebenenfalls zu überarbeiten. Aus Aufwandsgründen wird hierbei eine Aktualisierung der bestehenden Karte vorgezogen, als die Karte bei Veränderung der Umwelt komplett neu zu erarbeiten.

7.2 Topologisches Navigieren

Das Topologische Navigieren kann gegliedert werden in einen relationalen und einen assoziativen Lösungsansatz. Der relationale Ansatz wird allgemein als einfacher angesehen. Der Algorithmus sucht hier einen Pfad in der ihm bekannten Landkarte.

Über relationale Techniken erkennt der Roboter seinen Weg in Form eines Pfades durch die Karte. Bei den assoziativen Techniken fließen vor allem die Sensoren stark mit ein. Hierbei werden zusätzlich „Snapshots “ verwendet, weswegen diese Technik eher dazu geeignet ist, schon bekannte Wege wieder zu finden und relationale Techniken eher dem Finden eines Weges dienen.
Die wichtigsten Punkte im topologischen Navigieren sind die „Landmarks “ und die „Gateways“.
Landmarks sind spezielle Gegenstände oder Orte.
Gateways bezeichnen Weggabelungen. Zusammengenommen kann man auf diese Weise Graphen beschrieben wobei die Landmarks / Gateways die Knoten darstellen und die Kanten einen Weg zwischen den Orten bezeichnen. Die Landmarks müssen hierbei gut gewählt sein. Zum einen muß die Landmark vom

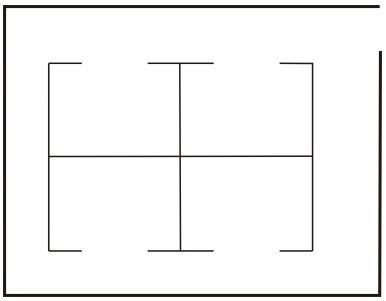


Abbildung 11: Einfacher Flurplan

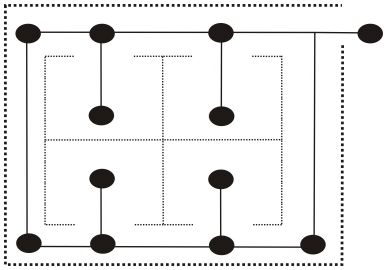
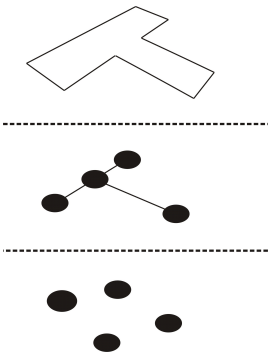


Abbildung 12: Relationaler Graph

Roboter klar gefunden werden können und auch als solche erkannt werden können. Zum anderen muss das Merkmal, welches einen Ort zu einem Landmark macht von allen Seiten aus erkennbar sein, so dass sich der Roboter von allen Richtungen darauf zu bewegen kann und das Objekt dennoch als Landmark entdeckt. In der realen Welt müssen für eine fehlerlose Wegfindung die gewählten Landmarks natürlich noch weitere Bedingungen erfüllen

7.2.1 Relationale Methoden

Hierbei wird die von den Sensoren erfasste Außenwelt vom Agenten als Graph dargestellt, wobei es verschiedene Abstraktionsebenen gibt. Die einfachste ist, die Landmarks als Knoten zu erfassen. Die nächsthöhere Stufe erstellt Verbindungen zwischen diesen Marken als Kante. Schließlich kann der Agent mithilfe eines Hill-Climbing Algorithmus, der den Roboter über die Kanten laufen lässt Entfernungen feststellen.



Somit entsteht eine räumliche Repräsentation von Mauern und Gegenständen in einem metrischen System. Trivialerweise verursacht die jeweils nächsthöhere Abstraktionsebene auch einen höheren Aufwand bei der Berechnung. Der Hill-Climbing Algorithmus steuert den Roboter jeweils in die Richtung, in der sich die Werte der Entfernungsfunktion am meisten verändern.
Der Vorteil an diesem System ist, dass der Roboter selbständig eventuell auftretende Fehler in der Navigationsanweisung beheben kann. Sollte die Anweisung ihn zu einer Koordinate außerhalb des gültigen Bereiches schicken, erkennt er dies anhand der Entfernungsfunktion, korrigiert seine Bewegungsrichtung und lokalisiert sich neu in seiner Umgebung. Ein weiterer Aspekt ist, dass der Roboter unerforschtes Gebiet katalogisieren kann.

Ein Nachteil ist die Voraussetzung der Konsistenz des Graphen. Um in diesem Modell sicherzustellen, dass sich der Roboter korrekt bewegt müssen sämtliche Ecken und Wände in dem Graphen existieren.

7.2.2 Assoziatives Navigieren

Der Ansatz für assoziative Methoden besteht in der Extraktion der Informationen der Sensoren. Das bedeutet, dass der Roboter speziell aufgrund von Sensoraufnahmen eine gewisse Landmark erreicht und nicht anhand eines Graphen.

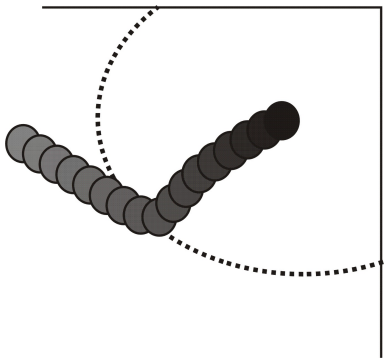


Abbildung 13: Reaktives Navigieren

Hierbei hat die Landmark im Allgemeinen zwei Eigenschaften. Sensoraufnahmen, in denen sich der Roboter um die eigene Achse dreht, sehen ähnlich aus und Aufnahmen einer Landmark, bei denen sich der Roboter bewegt, unterscheiden sich. Es wird ein Muster einer Landmark erstellt und im Folgenden nach diesem Muster gesucht. Die Motivation zu dieser Art der Navigation richtete sich nach der Sichtweise einer Biene. Entscheidend ist hierbei, dass der Roboter nicht gezwungenermaßen Landmarks übergeben bekommt um sich zurechtzufinden. Er kann sich Muster gewisser Bereiche auch selbst suchen.

Es erscheint klar, dass relationale Methoden ihre Vorzüge in der Navigation innerhalb eines Gebäudes haben, da hier eine Art Landkarte anhand der Begrenzungen angelegt wird. Assoziatives Navigieren dagegen hat aufgrund der Spezialisierung auf Bildmustererkennung eher Vorteile im freien Gelände. Dies führte zu einem Zusammenschluß beider Prinzipien mit dem Ziel einen Roboter im freien Gelände herumwandern zu lassen und spezielle Objekte zu erkennen, und nach Ihnen Ausschau zu halten ohne ähnlich aussehende Objekte aber damit zu verwechseln.

Dieses Prinzip „QualNav “ erstellt einen Graph und nimmt typische Objekte anhand ihrer Muster auf. Das entscheidende ist nun, die Bereiche die von den Objekten und dem Verbindungskanten eingekreist werden als Bereiche zu definieren. Dreht sich der Roboter um die eigene Achse so ist das Abfolge dieser Objekte immer die gleiche. Daraus erkennt der Roboter nun, in welchem dieser Bereich er sich befindet und ferner, ob neue Objekte wie zum Beispiel „Eindringlinge “ aufgetaucht sind. Die Distanz zu den Objekten ist hierfür nicht mehr von Interesse. Anstatt sich drehen zu müssen ist es auch denkbar, dass der Roboter in Form eines Wagens Sensoren an allen Seiten hat.
Als Nachteil dieses Systems sei zu nennen, daß ein großer Speicherplatz benötigt wird um die Objekte und Muster zu sichern und daß dieses Modell in einer dynamischen Welt, in der sich die als Landmark gesetzten Objekte bewegen, zu Problemen führen kann.

7.3 Metrische Pfadplanung

Die metrische Pfadplanung ist ein komplett anderer Ansatz im Vergleich zur topologischen Planung obwohl das Ziel, nämlich einen Pfad zum Ziel zu finden,

das gleiche ist. Metrische Pfadplanung konzentriert sich auf die Messung von Weglängen und erachtet den kürzesten Pfad als den besten, wohingegen die topologische Pfadplanung diesen Weg über Landmarks und Gateways zu finden versucht. Ein weiterer Unterschied ist, daß metrische Systeme die Gesamtstrecke in Koordinatenpaare, genannt Waypoint zerlegen. Diese Wegpunkte sind meist mathematischer Natur und bezeichnen keine Objekte in der Umgebung.

Metrische Systeme betrachten immer alle Möglichkeiten, was zwar Rechenaufwendig sein kann, allerdings auf diese Weise auch immer einen optimalen Weg findet. Die Eigenschaft, daß diese Systeme vergleichen und entscheiden können setzt als Grundlage eine schon existente Landkarte vorraus, die als korrekt und aktuell angesehen wird. Hieraus ist nun ersichtlich, daß metrisches Navigieren bei der Pfadsuche abwägen können wohingegen topologische Systeme besser als reaktive Systeme dienen. Ein metrisches System setzt sich aus zwei grundlegenden Komponenten zusammen. Zum einen wird eine Struktur benötigt, die die Außenwelt gemäß einer gewissen Rasterstruktur widerspiegelt, zum anderen den Algorithmus, der darin den optimalen Weg sucht.

7.3.1 Die Rasterstruktur

Sie erlaubt es dem Roboter, sich in seiner Umgebung eindeutigt zu lokalisieren. Um ein Objekt korrekt darstellen zu können werden insgesamt 6 Dimensionen benötigt. Da diese überflüssige Information zum Teil nicht benötigt wird, ist die Rasterstruktur die Abbildung der realen Welt in einen Unterraum. Die Schwierigkeit hierbei ist es, die überflüssigen Informationen herauszufiltern. Es existieren eine Vielzahl verschiedener Repräsentationsschemata, allen gemein ist allerdings die Idee, daß sich der Roboter in allen unmarkierten Bereichen frei bewegen kann.

7.3.2 Die Meadow Map

Die „meadow map “ ist ein Repräsentationsschema, in dem die a-priori bekannte Karte in konvexe Polygone verwandelt wird. Dieses Prinzip ist schon sehr alt und geht davon aus, daß dem Roboter von Anfang an eine sehr umfangreiche Karte eingegeben wurde. Der Vorteil den freien Raum in Konvex-polygone zu verwandeln ist der, daß der Roboter von einem Punkt auf dem Umfang in einer geraden Linie zu jedem anderen Punkt des Polygons bewegen kann ohne es zu verlassen. Zwar ist diese Technik in der Robotik nicht besonders häufig zu finden, illustriert aber sehr gut die Idee der metrischen Pfadplanung.

Oftmals werden die Kanten in der Karte bei der Unterteilung in Polygone zunächst um die Größe des Roboters gedehnt, um auszuschließen daß der Roboter an Kanten stößt. Ist die Unterteilung abgeschlossen sammelt der Suchalgorithmus alle Polygonkanten, die nicht zur Mauer oder zu Wänden gehören, denn der optimale Weg führt nur über die Kanten der konvexen Polygone. Im nächsten Schritt muß ein Waypoint auf den Kanten gefunden werden, den der Roboter ansteuert. Welcher Punkt das ist kommt auf die jeweilige Optimierung an. Es könnte zunächst die Kantenmitte gewählt werden und danach Waypoint-Tripel auf Transitivität hin überprüft werden. Die Probleme dieses Prinzips sind zum einen der hohe Rechenaufwand zur Findung der konvexen Polygone, zum

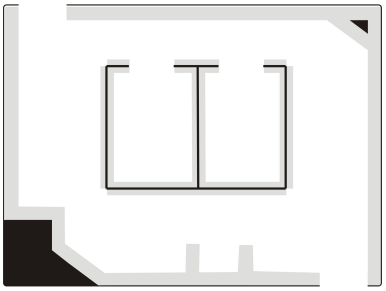


Abbildung 14: MeadowMap 1

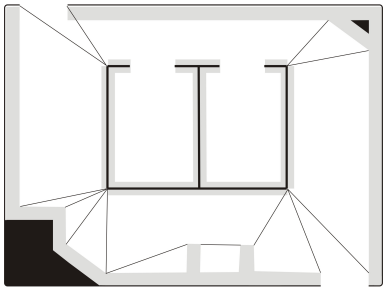


Abbildung 15: MeadowMap 2

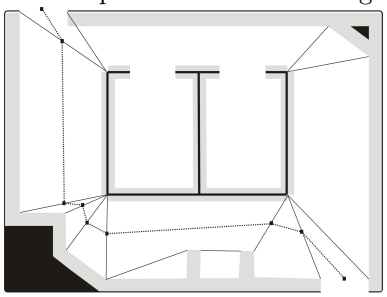


Abbildung 16: MeadowMap 3

anderen ist es die Vorraussetzung einer statischen Karte und zum dritten die Unfähigkeit, Inkonsistenzen oder schlichtweg Fehler an der Karte zu korrigieren.

7.3.3 Generalized Voronoi Graphen

Die sogenannten „Generalized Voronoi Graphen “ (GLV) sind um einiges beliebter, weil Roboter eher auf ungewohnte Umgebungen reagieren können und weit weniger Rechenlast benötigen. Der Ansatz ist hier Knoten zu finden, die von zwei Ecken in der Karte gleichweit entfernt liegen. Der Roboter fährt auf der Verbindungslinie dieser Knoten entlang und kann sicher sein, nirgends anzustoßen, weil die Knoten über die er fährt von den Ecken der Wände und Objekte gleichweit entfernt liegen.

7.4 Kartenerstellung

Bisher wurde die Frage geklärt, wie mobile Roboter Ihren Weg planen und suchen. Es wurde bisher allerdings nicht geklärt, wie der Roboter seinen aktuellen Standpunkt findet und auf welche Weise karten erstellt werden. Diese beiden Punkte liegen sehr nahe beisammen, da ein Roboter nicht imstande ist, eine korrekte Karte seiner Umgebung zu erstellen ohne zu wissen wo er sich befindet. Bei der Kartenerstellung bewegt sich das Steuerprogramm in einer iterativen Schleife, in der aktuelle Sensoraufnahmen analysiert und in die Map eingefügt werden. Hierbei berechnet das System die zurückgelegte Distanz. Bei diesem Prozess kommt ein systematisches Problem zum Tragen. Die zurückgelegte Distanz, respektive die Rollenumdrehung, ist stark abhängig von der Oberfläche

auf der sich der Roboter bewegt. Technologies wie GPS versprechen hier Abhilfe. Zwar wurde dieses Problem auf vielfache Weise angegangen, zeigt allerdings auch wie umfangreich eine korrekte Einsatzplanung eines mobilen Roboters ist.

Inhaltsverzeichnis

Bildverarbeitung **2**

1 Einleitung **3**

2 Kantendetektion **4**

2.1 Filter 4

2.1.1 Unschärfe-Filter 4

2.1.2 Gauss-Filter 5

2.2 Schematische Kantendetektion 6

2.3 Sobel-, Roberts-, Prewitt-Operatoren 6

2.4 Nonmaxima Suppression 8

2.5 Thresholding 8

2.6 Kantendetektion nach J.F. Canny 9

2.7 Farbe und Textur 9

3 Extraktion von 3D-Informationen **10**

3.1 Waltz-Algorithmus 10

3.2 $2\frac{1}{2}$-Dimensionale Bilder 11

3.2.1 Schattierung 11

3.2.2 Perspektive 11

3.2.3 Textur 12

3.3 Orientierung von Objekten 12

4 Objekterkennung **13**

4.1 Zur Geschichte... 14

4.2 Was mach Objekterkennung so schwierig? 14

4.3 Algorithmenansatz 15

Robotik **16**

5 Einleitung **16**

6 Wahrnehmung **16**

6.1 stationäre Roboter 16

6.2 Mobile, verhaltenstreue Roboter 18

6.3 Menschenähnliche Roboter 19

7 Planen und Bewegen **19**

7.1 Einleitung 19

7.2 Topologisches Navigieren 20

7.2.1 Relationale Methoden 21

7.2.2 Assoziatives Navigieren 21

7.3 Metrische Pfadplanung 22

7.3.1 Die Rasterstruktur 23

7.3.2 Die Meadow Map 23

7.3.3 Generalized Voronoi Graphen 24

7.4 Kartenerstellung 24

Literatur

[Caw03] Alison Cawsey. *Künstliche Intelligenz im Klartext*. Pearson Education Inc, first edition edition, 2003.

[Dew95] A.K. Dewdney. *Der Turing Omnibus*. Springer Verlag, 1995.

[GG00] J.Schneeberger G. Görz, C.-R. Rollinger. *Handbuch der Künstlichen Intelligenz*. Oldenburg Verlag, 3. vollständig überarbeitete auflage edition, 2000.

[Gin93] Matt Ginsberg. *Essentials Of Artificial Intelligence*. Morgan Kaufman Publishers Inc, 1993.

[Mal98] Hanspeter A. Mallot. *Sehen und die Verarbeitung visueller Informationen*. Vieweg Verlag, 1998.

[Mur00] Robin R. Murphy. *Introduction to AI Robotics*. The MIT Press, first edition edition, 2000.

[RJ95] Brian G. Schunck Ramesh Jain, Rangachar Kasturi. *Machine Vision*. McGraw-Hill Inc, international edition 1995 edition, 1995.

[SR95] Peter Norvig Stuart Russell. *Artificial Intelligence - A Modern Approach*. Pearson Education Inc, second edition edition, 1995.