Visuelles Reinforcement-Lernen zur Feinpositionierung eines Roboterarms über kompakte Zustandskodierung

Published in *Tagungsband „15. Fachgespräch Autonome Mobile Systeme", München, 1999. Springer Verlag.*

J. Zhang, G. Brinkschröder und A. Knoll

Technische Fakultät, Universität Bielefeld, 33501 Bielefeld

Zusammenfassung Diese Arbeit beschreibt eine hybride Methode zur Zustandsrepräsentation bei visuell geführter Feinpositionierung eines Roboterarms. Mit dieser Repräsentation kann ein Regler zum Greifen eines bestimmten Objektes, das sich an einer beliebigen Position befindet, über Reinforcement-Lernen automatisch erzeugt werden. Eine Positionierungsaufgabe mit drei Freiheitsgraden wird in mehrere Teilaufgaben gegliedert, die jeweils von einem darauf spezialisierten Lerner gelöst wird. Jeder dieser Lerner erhält eine eigene, speziell auf seine Aufgabe zugeschnittene Zustandskodierung. Basierend auf einer Hauptkomponentenanalyse (PCA) kann die Orientierung eines beliebigen Objektes korrekt kodiert werden.

1 Einführung

Die klassischen Methoden zum visuell geführten Greifen basieren auf Hand-Auge-Kalibration. Dies ist in der Regel sehr aufwendig und fehleranfällig. Hinzu kommt, daß die Kalibrierung jedesmal neu durchgeführt werden muß, wenn sich an der Versuchsanordnung etwas ändert. Eine Alternative sind lernende Verfahren. Überwachtes Lernen basiert auf adaptiven *universellen Funktionsapproximatoren* [7]. Um sich zu konfigurieren, benutzen sie eine Reihe von *Trainingsbeispielen*, auf die eine *Lernregel* angewendet wird. Trainingsbeispiele sind in diesem konkreten Fall Kamerabilder und die zugehörigen, gewünschten Roboterbewegungen. Diese Verfahren sind nur während der Trainingsphase adaptiv. Das bedeutet, daß das System in der Anwendungsphase nicht mehr selbständig auf Veränderungen reagieren kann. Beispielsweise könnte irgendwann gewünscht werden, daß das Objekt auf eine andere Weise gegriffen wird. Es muß dann mit neuen Beispielen wieder trainiert werden. Beim *Reinforcement-Lernen* [2] wird nicht zwischen Training und Anwendung unterschieden. Es wird lediglich eine *Belohnung* für eine erfolgreiche Positionierung vergeben. Aus diesen Belohnungen lernt das System. Da die Belohnungen ständig vergeben werden, kann das System ständig dazulernen und sich so auch an Veränderungen anpassen.

Obwohl das Reinforcement-Lernen in einigen mobilen Robotersystemen eingesetzt wird [5], findet man nur wenige Anwendungen in sensorbasierten Manipulationsaufgaben. Die Arbeit [4] beschreibt ein experimentelles Verfahren

zum Lernen von Greifen mit Zweifinger-Greifern. Ein modulares neuronales Netz wird trainiert, um den Sensor-Eingang auf eine Roboteraktion abzubilden. Ein Reinforcement-Signal wird benutzt, um die Anzahl der Fehlversuche zu reduzieren. Die Arbeit wird nur in einer simulierten Umgebung durchgeführt. Die Aspekte der Merkmalsextraktion und Auswertung der Reinforcement-Signale werden so vereinfacht, daß das Verfahren nicht auf reale Gegebenheiten übertragen werden kann.

Die Arbeit [3] stellt ein generelles Schema für das Lernen sensomotorischer Aufgaben. Greifen von unbekannten Objekten wird über Reinforcement-Lernen on-line durchgeführt. Eine Untermenge von greifrelevanten Merkmalen wird von einem Menschen aus den Kamerabildern ausgesucht. Ein Roboterarm mit vier Freiheitsgraden realisiert das eigentliche Greifen. Weil dieses Verfahren initiales Wissen über die Aufgaben braucht, werden offline heuristische Strategien für die Explorationsrichtung selektiert.

Die Arbeit [6] zeigt die Positionierung einer Roboterhand über einem auf dem Tisch liegenden Objekt mittels Reinforcement-Lernens. Die Aktionsauswahl erfolgt ungerichtet, d.h. Aktionen werden zufällig, aber gemäß einer bestimmten Wahrscheinlichkeitsverteilung ausgewählt, die die schon gelernten Reinforcement-Werte mitberücksichtigt. Die Probleme mit unvollständigen Zustandsinformationen wurden untersucht. *Experience Replay* bietet sich an, um die Lernschritte abzuspeichern und intern zu wiederholen. Als Zielzustand wird ein Bildpunkt vorgegeben, weswegen die Stabilität und die Qualität des Greifens nicht über externe Sensoren automatisch überwacht wird.

Eine der Schwierigkeiten der praktischen Anwendungen von Reinforcement-Lernen in Manipulationsaufgaben liegt im Finden einer kompakten und eindeutigen Zustandskodierung, womit das Problem des „Fluchs der Dimensionalität“ behandelt werden kann und der gelernte Controller robust funktioniert. Der wesentliche Teil dieser Arbeit liegt in einem automatischen Verfahren zur Lösung dieses Problems. Im nachfolgenden Abschnitt 2 werden die Hardware und Bildverabeitungssoftware unseres experimentellen Aufbaus beschrieben. Die detaillierten Vorgänge der Zustandskodierung werden in Abschnitt 3 erläutert. Das implementierte Reinforcement-Lernen für einen dreidimensionalen Bewegungsraum wird in Abschnitt 4 beschrieben. Der letzte Abschnitt faßt die Vorteile des Verfahrens zusammen und diskutiert einige Erweiterungsmöglichkeiten.

2 Umgebung für die Experimente

2.1 Kameragestützter Roboterarm

Zur Demonstration wird ein Aufbau aus Standard-Komponenten verwendet: ein auf dem Manipulator (Puma 260) montierter Zweifinger-Greifer und eine Handkamera, Abb. 1. Der Greifer soll stets senkrecht zur Tischebene positioniert werden und außerdem eine feste Höhe haben. Dann bleiben ihm noch noch drei Freiheitsgrade: Bewegungen parallel zur Tischebene (x/y) und Rotationen um eine Achse senkrecht zur Tischebene (θ).

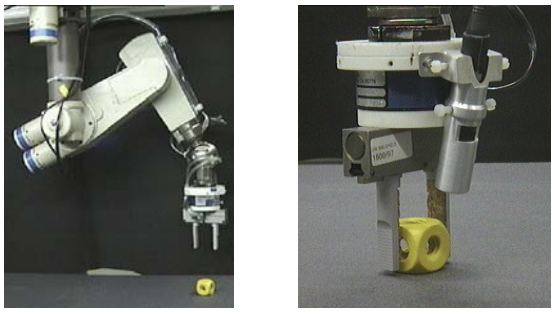

Abbildung1. Der PUMA-260-Roboter (links) und der Greifer mit anmontierter Kamera (rechts).

2.2 Vorverarbeitung der Kamerabilder

Die Kamera nimmt die RGB-Bilder auf. Diese Bilder sind als Zustandsvektoren ungeeignet weil sie erstens viele irrelevanten Informationen beinhalten, und zweitens aus sehr vielen Pixeln bestehen. Daher werden die Kamerabilder wie folgt weiterverarbeitet: Zunächst werden aus den Bildern alle irrelevanten Informationen herausgefiltert. Die übrigbleibende, relevante Information liegt in Form eines Binärbildes vor. Auf dieses Binärbild werden dann Methoden zur Dimensionsreduktion angewendet: die Berechnung von Geometrieparametern und Hauptkomponentenanalyse (Abb. 2).

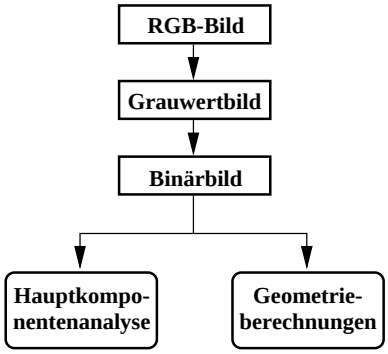

Abbildung2. Weiterverarbeitung der Kamerabilder.

Für die meisten Greifaufgaben ist es ausreichend, wenn das Binärbild nur noch die *Silhouette* des zu greifenden Objektes enthält. Die RGB-Bilder aus der Hand-Kamera werden über eine Reihe von Vorverarbeitungsschritten in bereinigte binäre Bilder umgewandelt, Abb. 3.

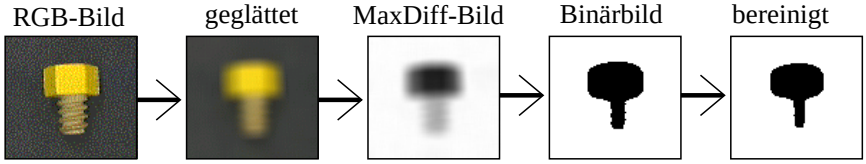

Abbildung3. Alle Schritte der Bildvorverarbeitung im Überblick.

2.3 Geometrieparameter

Hat man ein Binärbild erzeugt, das nur noch die Silhouette des Objekts enthält, so lassen sich durch einfache Integrationen grund-

legende Geometrieparameter wie Flächeninhalt, Lage des Schwerpunktes und Orientierung berechnen [1]. Während also die Schwerpunktberechnung für beliebige Objekte funktioniert, ist die Berechnung der Orientierung auf längliche Objekte beschränkt, eignet sich also nur bedingt als Zustandskodierung. Um eine wesentlich größere Klasse von Objekten greifen zu können, wird ein automatisches Verfahren zum Auffinden von Zustandsvektoren entwickelt.

3 Automatische Orientierungskodierung

3.1 PCA-Ansatz

Um die Orientierung eines beliebigen Objektes in einem Kamerabild zu kodieren wird eine Serie von Bildern des Objektes aufgenommen. Das Objekt hat dabei in jedem Bild ungefähr dieselbe Position, aber verschiedene Orientierungen. Die Bildzeilen werden konkateniert und die Bilder als Vektoren betrachtet. Man berechnet nun die ersten k Hauptkomponenten $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_k$ dieser Vektoren. Da der wesentliche Unterschied zwischen den Bildern in der Orientierung des Objektes liegt, beinhalten auch die ersten k Hauptkomponenten im wesentlichen Orientierungsinformation. In den *Eigenraum*, den diese k Vektoren aufspannen, werden nun zukünftig alle Bilder projiziert. Diese Projektionen dienen als Kodierung der Orientierung des Objektes.

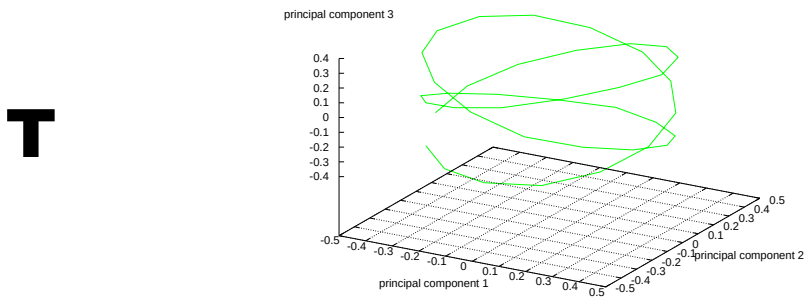

Abbildung4. Beispiel für eine Mannigfaltigkeit im von den ersten drei Hauptkomponenten aufgespannten Eigenraum. Links das Objekt, von dem eine Bilderserie verwendet wurde.

Nimmt man eine Bilderserie von einem Objekt auf, wobei das Objekt bei jedem Bild ein Stück weiter gedreht wird, so daß es sich am Schluß wieder in seiner Ausgangslage befindet, so sollten die Projektionen dieser Bilder im von den Hauptkomponenten aufgespannten Eigenraum eine relativ glatte, geschlossene Mannigfaltigkeit durchlaufen (Abb. 4). Nur dann kann man von einer sinnvollen Kodierung ausgehen. Die verschlungene, spiralenartige Struktur wirft

aber gleichzeitig die Frage auf, ob drei Hauptkomponenten für eine eindeutige Kodierung ausreichen. Da dieses Verfahren auf beliebige Objekte angewendet werden soll, stellt sich ganz allgemein die Frage: Wie viele und welche Hauptkomponenten sind für eine eindeutige Kodierung der Orientierung eines Objektes notwendig? Wichtig ist insbesondere, daß für die Kodierung möglichst wenige Hauptkomponenten verwendet werden, da mit steigender Dimension des Zustandsraumes auch der Lernaufwand zunimmt.

3.2 Kombination von Hauptkomponenten

Zunächst muß eine Methode gefunden werden, wie die beiden Komponenten eines Datenpunktes in einen einzigen Parameter umgerechnet werden sollen. Da sich die Punkte auf einem Kreis um den Ursprung des Koordinatensystems befinden, bietet es sich, wie schon oben erwähnt, an, den Winkel zwischen der gedachten Geraden durch den Punkt sowie den Ursprung und einer Referenzachse zu berechnen. Diese Abbildung hat folgende, wichtige Eigenschaften:

- Sie ist hinreichend stetig und glatt. Ähnliche Datenpunkte werden also auf ähnliche Werte abgebildet.
- Sie ist eindeutig. Es gibt keine zwei Datenpunkte, die auf denselben Wert abgebildet werden.

Somit bleiben die wesentlichen Eigenschaften der zweidimensionalen Datenpunkte erhalten. Eine einfache Möglichkeit, diese Winkelberechnung vorzunehmen, bietet die atan2-Funktion. Sie ist wie folgt definiert:

$$\operatorname{atan2}(y,x) = \begin{cases} \arctan\left(\frac{y}{x}\right) & \text{falls } x > 0 \\ \frac{\pi}{2} & \text{falls } x = 0 \ \wedge \ y > 0 \\ -\frac{\pi}{2} & \text{falls } x = 0 \ \wedge \ y < 0 \\ \arctan\left(\frac{y}{x}\right) + \pi & \text{falls } x < 0 \ \wedge \ y \geq 0 \\ \arctan\left(\frac{y}{x}\right) - \pi & \text{falls } x < 0 \ \wedge \ y < 0 \end{cases} \tag{1}$$

Die atan2-Funktion erweitert den Wertebereich der atan-Funktion von $[-\frac{\pi}{2}; \frac{\pi}{2}]$ auf $[-\pi; \pi]$. Sie weist jedem Punkt (x, y) der Ebene den Winkel zwischen der Geraden durch den Punkt und den Ursprung und der positiven x-Achse zu.

3.3 Auffinden von Hauptkomponentenpaaren

Es wird ein Verfahren benötigt, das automatisch ermittelt, ob eine Menge von Datenpunkten eine Kreisstruktur aufweist und um wieviele Kreise es sich gegebenenfalls handelt, wieviele Perioden der atan2-Graph also durchläuft.

Zunächst besitzt der Graph eine bevorzugte *Monotonierichtung*: Weist die Mehrzahl der Schritte von einem berechneten Wert zum nächsten Schritt nach oben, so ist die Monotonierichtung steigend, sonst fallend. Man betrachtet nun drei Arten von *markanten Stellen* im Graphen (Abb. 5):

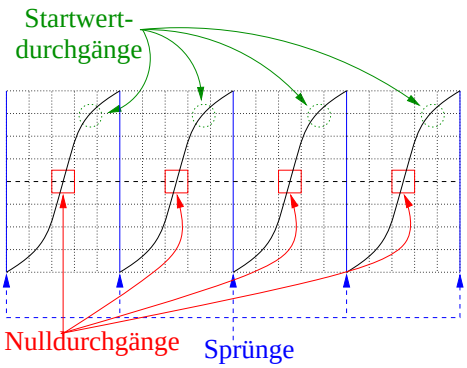

Abbildung5. Markante Stellen in einem atan2-Graphen.

Nulldurchgänge sind Stellen, an denen die die berechneten Funktionswerte interpolierende Kurve die x-Achse in Monotonierichtung schneidet.

Sprünge sind Schritte entgegen der Monotonierichtung. Bei einem Graphen sind dies die Sprünge vom Maximum zum Minimum (oder umgekehrt).

Startwertdurchgänge sind Stellen, an denen der Graph den atan2-Wert des ersten Datenpunktes in Monotonierichtung passiert.

Indem man nun den gesamten Graphen nach diesen markanten Stellen absucht, bildet man dementsprechend drei **Kennzahlen**,

- die Anzahl n der Nulldurchgänge,
- die Anzahl s der Sprünge, und
- die Anzahl st der Startwertdurchgänge.

Aus diesen drei Kennzahlen läßt sich nun ermitteln, ob eine Kreisstruktur vorliegt, und auch, um wieviele Kreise es sich handelt. Bei einem beliebigen Graphen sind die Kennzahlen unkorreliert. Bei einem Graphen wie in Abb. 5 hingegen kann ihre Differenz, wie man sich leicht überlegen kann, höchstens Eins betragen. Ist die maximale Differenz zwischen n, s und st also größer als Eins, so kann man davon ausgehen, daß keine Kreisstruktur vorliegt. Andernfalls ist die Wahrscheinlichkeit einer Kreisstruktur sehr groß. Wie läßt sich in diesem Fall die Anzahl der Perioden, die der Graph durchläuft, aus den Kennzahlen ermitteln? Dazu betrachte man zunächst Tabelle 1.

n	0	0	1	1	1	2	2	2	2	3	...
s	0	1	0	1	2	1	2	3	2	3	...
p	1	1	1	1 oder 2	2	2	2 oder 3	3	3	3 oder 4	...

Tabelle1. Kombinationen von n und s und die sich daraus ergebende Periodenanzahl p.

Sie zeigt die möglichen Wertekombinationen für die Kennzahlen n und s und die zugehörige Anzahl Perioden p. Wie sich diese Anzahlen ergeben, macht man

sich am besten anhand von Abb. 5 klar. Man sieht, daß die Periodenanzahl schon aus n und s bestimmt werden kann, wenn diese nicht den gleichen Wert haben. Falls dies doch der Fall ist, stehen zwei Werte zur Auswahl, und für die Entscheidung muß st herangezogen werden. Aus Abb. 5 geht außerdem hervor, daß die Anzahl der Startwertdurchgänge immer um Eins kleiner ist als die Anzahl der Perioden, falls es gleich viele Nulldurchgänge und Sprünge gibt. Die Anzahl der Perioden ist in dem Fall also $st + 1$.

3.4 Auswahl von Werten für die Zustandskodierung

Es bleibt die Frage, wann man genügend Informationen hat, um die Orientierung eines Objektes eindeutig kodieren zu können. Sicherlich ist dies der Fall, wenn man ein Hauptkomponentenpaar gefunden hat, dessen atan2-Funktion nur eine einzige Periode umfaßt, wie etwa in Abb. 6.

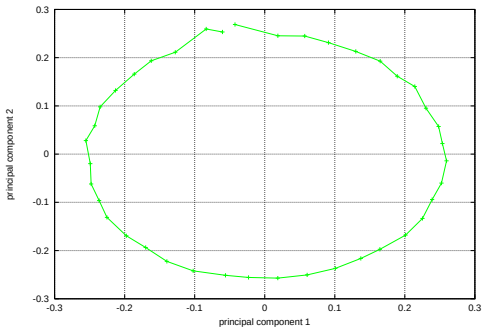

Abbildung6. Der von der ersten und zweiten Hauptkomponente aufgespannte Untervektorraum der Bilderserie eines quadratischen Objektes.

Dann genügt ein einziger Parameter, um diese an sich auch eindimensionale Information zu kodieren. Hat man jedoch Funktionen mit mehreren Perioden, so ist die Abbildung mehrdeutig. Mehrere Orientierungen des Objektes werden mit demselben Zustand kodiert. Man könnte allerdings versuchen, mehrere dieser atan2-Funktionen zu kombinieren. Es stellt sich heraus, daß stets zwei solcher Funktionen genügen (wenn sie für das betreffende Objekt existieren), um eine eindeutige Kodierung zu erreichen.

Um das einzusehen, betrachte man folgende Analogie: Die Funktion $\sin(x)$ hat die Periode 2π. Die Funktion $\sin(kx)$ hat dementsprechend die Periode $\frac{2\pi}{k}$. Nun nehme man zwei Funktionen $\sin(k_1 x)$ und $\sin(k_2 x)$ mit den Perioden $\frac{2\pi}{k_1}$ und $\frac{2\pi}{k_2}$. Wenn man die beiden Funktionen nun kombiniert (z.B. addiert), so hat das Ergebnis als Periode das kleinste gemeinsame Vielfache der Einzelperioden

$$\text{kgV}\left(\frac{2\pi}{k_1}, \frac{2\pi}{k_2}\right) = 2\pi \cdot \text{kgV}\left(\frac{1}{k_1}, \frac{1}{k_2}\right) = 2\pi \text{ , falls } k_1 \text{ und } k_2 \text{ teilerfremd sind} \tag{2}$$

Obwohl also die Periodenlänge bei den beiden einzelnen Funktionen durch den Faktor k_1 bzw. k_2 dividiert wurde, besitzt die Kombination der beiden Funktionen wieder die volle Periodenlänge, sofern k_1 und k_2 teilerfremd sind.

Kombiniert man drei Funktionen, so daß die Kombination die Periode 2π hat, so müssen mindestens zwei der drei k_i teilerfremd gewesen sein. Dann hat aber bereits die Kombination dieser beiden Funktionen die Periode 2π, und die dritte Funktion ist reduziert.

Analog gilt für die atan2-Funktionen: Hat man zwei Hauptkomponentenpaare mit den Periodenanzahlen p_1 und p_2, und p_1 und p_2 sind teilerfremd, so ist die Kodierung bereits eindeutig. Kodiert man die Orientierung des Objektes mit mehr als zwei Hauptkomponentenpaaren, so sind alle bis auf zwei überflüssig.

Zusammengefaßt sollte man also folgende Strategie bei der Suche nach einer geeigneten Kodierung für die Orientierung eines Objektes anwenden:

1. Suche nach einem Hauptkomponentenpaar, dessen atan2-Funktion eine einzige Periode umfaßt.
2. Falls das fehlschlägt: Suche nach zwei Hauptkomponentenpaaren, deren Periodenanzahlen teilerfremd sind.
3. Schlägt auch das fehl: Nimm eine weitere Hauptkomponente zur Zustandskodierung hinzu.

4 Implementierung von Reinforcement-Lernen

4.1 Dekomposition von Translation und Rotation

Um kleinere Zustandsräume und eine bessere Zustandskodierung zu erzielen, wird die Lernaufgabe auf zwei Lerner aufgeteilt, einen für die Translationsbewegungen und einen für die Rotationsbewegungen. Der *Translations-Lerner* hat dann vier Aktionen zur Auswahl (zwei in x- und zwei in y-Richtung). Als Zustandskodierung wird der aus dem Kamerabild berechnete Objektschwerpunkt verwendet. Der *Rotations-Lerner* hat zwei Aktionen zur Auswahl (Drehung im und entgegen dem Uhrzeigersinn). Diese Aufteilung bietet gegenüber einem einzigen Lerner folgende Vorteile:

- Die Zustandsräume sind wesentlich kleiner. Im obigen Beispiel ergeben sich $20^2 = 400$ Zustände für den Translations-Lerner und 30 für den Rotations-Lerner.

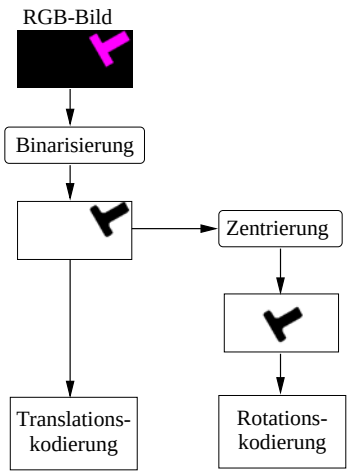

Abbildung7. Für die Rotationskodierung werden die vorverarbeiteten Bilder zusätzlich zentriert.

– Die Zustandskodierung ist so gestaltet, daß die Zustandsvektoren die für den jeweiligen Lerner relevante Informationen enthalten.

Die beiden Lerner wechseln sich nun ab. Zuerst führt der Translations-Lerner so lange Lernschritte aus, bis er sich bezüglich seiner Zustandskodierung im Zielzustand befindet. Dann wird er abgelöst durch den Rotations-Lerner, der ebenfalls so lange Schritte ausführt, bis er sich bezüglich seiner Kodierung im Zielzustand befindet. Nun kann es sein, daß durch die Rotationen der Translations-Zielzustand wieder verlassen wurde. Daher wird der Translations-Lerner wieder aktiviert usw., bis beide Lerner melden, daß sie sich im Zielzustand befinden. Dies ist definiert als der Gesamt-Zielzustand.

4.2 Variable Schrittweite

Der Lerner befindet sich in einem permanenten Zyklus aus Zustandsbestimmung und Aktionsausführung. Der aktuelle Zustand wird bestimmt, indem ein Kamerabild aufgenommen, vorverarbeitet und in einen Zustandsvektor umgewandelt wird. Das erfordert einen relativ großen Zeitaufwand, so daß es wünschenswert ist, mit möglichst wenigen Kamerabildern das Ziel zu erreichen. Dazu bietet es sich an, den Roboter, wenn er weit vom Ziel entfernt ist, große Schritte machen zu lassen, und in der Nähe des Ziels die Schrittweite entsprechend klein zu halten.

Man bedenke, daß ein realer Roboter bei jedem Schritt eine bestimmte Entfernung zurücklegt, die normalerweise nichts mit den Distanzen zwischen den Zuständen im Zustandsraum zu tun hat. So kann man nicht voraussagen, ob die Welt nach dem Schritt in einen benachbarten oder einen „weiter entfernten“ Zustand übergeht, oder ob sie sogar im selben Zustand bleibt. Daher wird das Konzept der **variablen Schrittweite** eingeführt. Man legt eine minimale und eine maximale Schrittweite fest und schätzt die maximale euklidische Distanz eines beliebigen Zustandes zum Zielzustand ab. Die Schrittweite wird dann entsprechend proportional zur Distanz des aktuellen Zustandes zum Zielzustand gewählt.

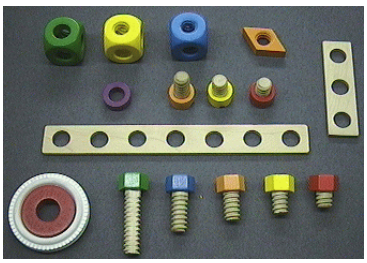

Abbildung8. Diese Baufix-Teile dienen als Testobjekte.

Das oben beschriebene Lernverfahren ist in Form einer Objekthierarchie implementiert. Greifexperimente mit realen Objekten (Abb. 8) wurden erfolgreich durchgeführt.

5 Diskussion

Die Innovation dieses Ansatzes liegt im automatischen Verfahren zur Kodierung von Roboter-Zuständen und seiner Anwendung im visuell geführten Greifvor-

gang. Während die auf geometrischen Merkmalen basierte Darstellung nur bei länglichen Objekten funktioniert, ist unserer Ansatz für Objekte beliebiger Form einsetzbar, auch wenn keine eindeutigen geometrischen Merkmale extrahiert werden können. Dadurch werden die Zustände bei der visuell geführten Bewegung auf die kompakteste Weise dargestellt. Das Reinforcement-Lernen kann für eine breite Reihe von Greif-Operationen in Praxis umgesetzt werden.

In Zukunft soll die gesamte Qualität eines Greifversuchs automatisch über *aktive Kameras* und *aktive Testbewegungen* evaluiert werden. Nach dem Greifen eines Objektes muß die andere Hand-Kamera einen oder mehrere gute Ansichtspunkte finden, um von dort Bilder aufzunehmen. Diese Bilder werden mit Hilfe von heuristischem Wissen ausgewertet, um die Greifqualität zu ermitteln. Wie beim Greifen mit einer menschlichen Hand garantiert eine optisch korrekte Positionierung nicht direkt ein stabiles Greifen. Einige Testbewegungen können für die greifende Hand programmiert werden, z.B. entlang der Normalen-, Schließ-, Annäherungsrichtung. Die Kameras überwachen durchgehend die Handbacken und das gegriffene Aggregat. Unter der visuellen Führung und Kraftregelung, soll die Greifhand das gegriffene Aggregat aktiv auf die Tischfläche absetzen. Die Rutschbewegungen können über die Auswertung des Kraftsensors festgestellt werden. Die möglicherweise große Positionsverschiebung des Aggregates kann wiederum über die Kameraüberwachung erfaßt werden.

Literatur

1. Berthold Klaus Paul Horn. Binary images: Geometrical properties. In *Robot Vision*, chapter 3, pages 46–61. McGraw-Hill, 1986.
2. Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research*, 4:237–285, Mai 1996.
3. I. Kamon, T. Flash, and S. Edelman. Learning visually guided grasping: A test case in sensorimotor learning. *IEEE Transactions on System, Man and Cybernetics*, 28(3):266–276, May 1998.
4. M. A. Moussa and M. S. Kamel. An experimental approach to robotic grasping using a connectionist architecture and generic grasping functions. *IEEE Transactions on System, Man and Cybernetics*, 28(2):239–253, May 1998.
5. Sebastian Thrun and Anton Schwartz. Finding Structure in Reinforcement Learning. In *Advances in Neural Information Processing Systems 7*, pages 385–392, 1995.
6. Thomas Wengerek. *Reinforcement-Lernen in der Robotik*. PhD thesis, Universität Bielefeld - Technische Fakultät, Dezember 1995.
7. J. Zhang, R. Schmidt, and A. Knoll. Appearence-based visual learning in a neuro-fuzzy model for fine-positioning of manipulators. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Detroit, 1999.