Learning and imitation in heterogeneous robot groups

Wilhelm Richert, Universität Paderborn

C-LAB Report

Vol. 9 (2010) No. 04

Cooperative Computing & Communication Laboratory

ISSN 1619-7879

C-LAB ist eine Kooperation
der Universität Paderborn und der Siemens AG
www.c-lab.de
info@c-lab.de

C-LAB Report

Herausgegeben von
Published by

Dr. Wolfgang Kern, Siemens AG
Prof. Dr. Franz-Josef Rammig, Universität Paderborn

Das C-LAB - Cooperative Computing & Communication Laboratory - leistet Forschungs- und Entwicklungsarbeiten und gewährleistet deren Transfer an den Markt. Es wurde 1985 von den Partnern Nixdorf Computer AG (nun Siemens AG) und der Universität Paderborn im Einvernehmen mit dem Land Nordrhein-Westfalen gegründet.
Die Vision, die dem C-LAB zugrunde liegt, geht davon aus, dass die gewaltigen Herausforderungen beim Übergang in die kommende Informationsgesellschaft nur durch globale Kooperation und in tiefer Verzahnung von Theorie und Praxis gelöst werden können. Im C-LAB arbeiten deshalb Mitarbeiter von Hochschule und Industrie unter einem Dach in einer gemeinsamen Organisation an gemeinsamen Projekten mit internationalen Partnern eng zusammen.

C-LAB - the Cooperative Computing & Cooperation Laboratory - works in the area of research and development and safeguards its transfer into the market. It was founded in 1985 by Nixdorf Computer AG (now Siemens AG) and the University of Paderborn under the auspices of the State of North-Rhine Westphalia.
C-LAB's vision is based on the fundamental premise that the gargantuan challenges thrown up by the transition to a future information society can only be met through global cooperation and deep interworking of theory and practice. This is why, under one roof, staff from the university and from industry cooperate closely on joint projects within a common research and development organization together with international partners. In doing so, C-LAB concentrates on those innovative subject areas in which cooperation is expected to bear particular fruit for the partners and their general well-being.

ISSN 1619-7879

C-LAB
Fürstenallee 11
33102 Paderborn
fon: +49 5251 60 60 60
fax: +49 5251 60 60 66
email: info@c-lab.de
Internet: www.c-lab.de

C-LAB-TR-2010-04

Learning and imitation in heterogeneous robot groups

Wilhelm Richert

Fakultät für Elektrotechnik, Informatik und Mathematik,
Universität Paderborn
richert@c-lab.de

Abstract: Mit wachsenden technischen Fähigkeiten und gestiegenen Rechenkapazitäten dringen autonome Roboter immer weiter in bislang undenkbare Anwendungsfelder vor. Die effiziente Programmierung des gewünschten Roboterverhaltens zählt dabei zu einem der herausforderndsten Themen in der Robotik. Verfahren, die ausschließlich auf Lernen basieren, haben den Nachteil einer langen Lerndauer. Imitation ist hier als mächtiges Werkzeug bekannt, mit dem Roboter das Verhalten voneinander übernehmen können und dadurch Lernabkürzungen nehmen können. Dazu muss der imitierende Roboter die sogenannten fünf wichtigen Fragen der Imitation beantworten: wann soll welches Verhalten von welchem Roboter wie imitiert, und wie soll der Imitationserfolg schließlich gemessen werden. Die Komplexität des Imitationsprozesses hat im Roboterumfeld bisher zu Lösungen geführt, die sich nur auf einen Ausschnitt dieser Fragen konzentrieren und den Rest als gegeben betrachten oder ignorieren.

Die hier zusammengefasste Dissertation [Ric09] leistet dabei einen wichtigen Beitrag zum Stand der Forschung, indem sie diese Fragen der Imitation in der Robotik zum ersten Mal geschlossen beantwortet. Mit der vorgestellten Roboterarchitektur und den darauf aufbauenden Verfahren kann Imitation nun auch in vollkommen autonomen Robotergruppen angewandt werden.

1 Einführung

Mit Imitation kann ein Roboter sowohl seinen Lernprozess beschleunigen, als auch seine Einsatzdauer erhöhen, da er besonders riskante Explorationen auslässt. Imitation ist dabei ein Sammelbegriff, unter dem Themenfelder wie z.B. *Programming By Example* [Lie01], *Apprenticeship Learning* [CAN09] oder *Behavioral Cloning* [KSS06] zusammengefasst werden. Damit ein Roboter überhaupt imitieren kann, müssen die folgenden Fragen geklärt sein [DN02][1]:

Wer soll imitiert werden? In einer heterogenen Robotergruppe sind nicht alle Roboter gleich gute Demonstratoren.

Wann soll imitiert werden? Der Imitator muss die aktuelle Situation des potentiellen Demonstrators berücksichtigen und ihn nur dann imitieren, wenn er gerade etwas imitierungswürdiges ausführt.

[1] Im Folgenden bezeichnet *Imitator* den Roboter, der imitieren möchte, und *Demonstrator* einen Roboter, der potentiell imitiert werden kann.

Was soll imitiert werden? Das Endergebnis des beobachteten Verhaltens, die einzelnen Aktionen, die zu dem Ergebnis geführt haben, oder das dahinter liegende Ziel, das der Demonstrator aller Wahrscheinlichkeit nach gehabt hat?

Wie soll das beobachtete Verhalten in das bereits gelernte Verhalten und die Aktionsfähigkeiten des Imitators integriert werden (das sog. *Korrespondenzproblem*)?

Wie bewerten? Was sollte als erfolgreicher und was als eher erfolgloser Imitationsversuch gezählt werden?

In der Literatur wird fast ausschließlich das "wie" und "was" thematisiert. Die Frage, "wer" imitiert werden soll, wird hingegen meistens ignoriert – oft dadurch, dass ein Roboter einen zuvor spezifizierten menschlichen Experten imitieren soll oder dadurch, dass in Multiroboterszenarien nur exakt baugleiche Roboter eingesetzt werden. Weiter wird die Frage, "wann" imitiert werden soll, in vielen Ansätzen dadurch gelöst, dass Start und Ende des zu imitierenden Verhaltens fest vorgegeben werden. Die Bewertung des Imitationserfolges wird dann über die Ähnlichkeit von demonstrierter und imitierter Handlung bestimmt. In Robotergruppen sollte jedoch nicht die Ähnlichkeit der Verhaltensreproduktion, sondern der Nutzen des imitierten Verhaltens für den Imitator ausschlaggebend sein. Schlussendlich wird meistens der Tabula Rasa Ansatz gewählt: Die Imitation startet ohne Vorwissen. Ein vorheriges oder anschließendes Weiterlernen von Verhalten ist nicht möglich.

Aus diesen Gründen war Imitation in autonomen Robotergruppen ohne manuelle Intervention bislang nicht einsatzfähig. Dabei ist gerade hier Imitation besonders sinnvoll: Einmal gelerntes Verhalten kann in der Gruppe durch Imitation schnell an weitere Gruppenmitglieder propagiert werden, die den zeitaufwendigen Explorationsprozess für das jeweilige Verhalten dann nicht mehr selber durchführen müssen.

Die Herausforderungen wurden in der Dissertation mit Hilfe einer speziellen Architektur und darauf basierenden Verfahren gelöst, die Imitation mit eigenem Lernen kombinieren (**Abschnitt 2**). Damit kann ein Roboter seine Gruppenmitglieder sogar dann imitieren, wenn diese mit einer anderen Hard- oder Software laufen, solange sie ihren allgemeinen Befindlichkeitszustand nach außen signalisieren – ähnlich dem menschlichen Emotionszustand, welcher uns oft in der Frage leitet, ob wir jemanden imitieren oder nicht. Dies ist auch die einzige Anforderung, die in dieser Arbeit an zu imitierende Roboter gestellt wird. Das "wann", also das Zeitintervall der Imitation, wird durch den eben erwähnten beobachteten Befindlichkeitszustand des Demonstrators bestimmt. Das "was" und "wie" wird dadurch gelöst, dass der Imitator selber Aktionen gegen eine Zielfunktion lernen kann. Dies kann einerseits genutzt werden, um den Fortschritt der eigenen Aktion zu messen, andererseits um diese Aktion in dem beobachteten Verhalten eines anderen Roboters zu erkennen (**Abschnitt 3**). Die Frage, "wer" imitiert werden soll, verlangt nach einer Metrik, die die Unterschiede in den Aktionsfähigkeiten der einzelnen Robotern messen kann. Auch in diesem Bereich gibt es zurzeit nur sehr begrenzt Beiträge in der Literatur. Während einige Ansätze detaillierte Informationen über die zugrunde liegenden Zustands- und Aktionsräume der beiden zu vergleichenden Roboter benötigen, verlangen andere, dass das Korrespondenzproblem manuell gelöst wird. Der in dieser Dissertation vorgestellte Ansatz beantwortet die Frage in Form eines neuen Ansatzes zur Messung von Verhaltensdifferenzen, der benutzt werden kann, ohne vorher auf die Imitationsdaten zuzugreifen (**Abschnitt 4**).

2 Architektur

Es stellte sich relativ früh heraus, dass die fünf Fragen der Imitation mit einem einzelnen Algorithmus nicht zufriedenstellend beantwortet werden könnten. Nur durch das Zusammenspiel von übergeordneten Zielen und den darunter liegenden Aktionen können komplexe Aufgaben wie das Erkennen von bekannten Verhaltensweisen in dem Verhalten unbekannter beobachteter Roboter bewältigt werden. Dies beinhaltet das Integrieren dieser Verhaltensweisen in die eigene Verhaltenswissensbasis oder die Entscheidung, welcher Roboter überhaupt imitiert werden kann. Die Architektur in Abb. 1 ist dazu in der Lage. Über die Sensoren erhält der Roboter seine Perzeption, die er an die beteiligte Motivations-, Strategie- und Skillschicht weiterleitet. Die Motivationsschicht berechnet dann das unmittelbar nächste Ziel auf Basis seines aktuellen Motivationszustandes. Die Strategieschicht verarbeitet diese Information zusammen mit der aktuellen Perzeption, um die nächste Aktion für den aktuellen abstrahierten Zustand auszuwählen. Die Aktion gibt sie an die Skillschicht weiter, die wiederum dafür sorgt, die abstrakte Aktion durch eine Sequenz von Aktuatorkommandos auszuführen.

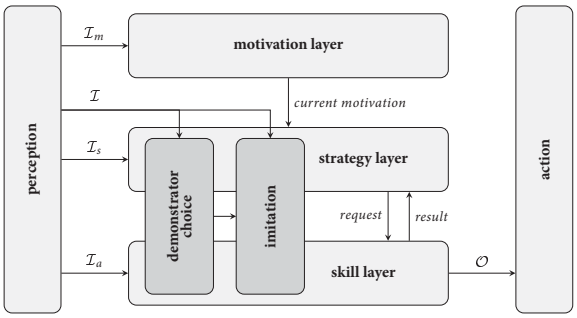

Abbildung 1: Die Architektur eines imitierenden Roboters

2.1 Motivationsschicht

Die Motivationsschicht definiert die verschiedenen Ziele des Roboters. Im Idealfall muss ein Programmierer nur diese Schicht festlegen – der Roboter erlernt die dazugehörigen Strategien und Aktionen selbstständig. Die einzelnen Teilziele werden somit in Form von biologisch inspirierten Bewertungsmethoden gemessen und der Strategieschicht als Motivationsvektor $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_n)^T$ bereitgestellt ($\mu_i \in \mathbb{R}^+$). Der aktuelle Wert für μ_i wird dabei kontinuierlich durch die Bewertungsfunktion $\hat{\mu}_i : \mathcal{I}_m \rightarrow \mathbb{R}^+$ berechnet. Der Programmierer legt deshalb für jedes Ziel i die entsprechende Bewertungsfunktion fest. Die Strategieschicht sorgt dafür, dass die Ziele erreicht werden. Dies geschieht dadurch, dass sie versucht, für jeden Zustand eine Aktion auszuwählen, die zur Folge hat, dass der Betrag des Vektors $\boldsymbol{\mu}$ minimiert wird.

Eine weitere Aufgabe der Motivationsschicht ist es nach außen zu signalisieren, wie zufrieden der Roboter mit seiner aktuellen Situation ist. Dies ist notwendig, damit andere Roboter ausreichend Informationen während des Informationsprozesses erhalten. Man kann es mit dem Ausdruck von Emotionen vergleichen: Ein beobachtender Mensch kann leichter entscheiden, ob er jemand anderes imitiert, wenn er dessen Emotionsausdruck wahrnehmen kann.

2.2 Strategieschicht

Diese Schicht sucht nun nach Strategien, die das Ziel haben, sämtliche Motivationen permanent zufrieden zu stellen. Der zugrunde liegende Ansatz ist in Abb. 2 dargestellt. Er basiert auf Reinforcement Learning, das in großen Zustandsräumen dafür bekannt ist viele Lernzyklen zu benötigen, um zu einer brauchbaren Strategie zu gelangen. Deshalb wird die vorgefilterte Perzeption in einem Erfahrungsspeicher vorgehalten. Der wird genutzt um, angelehnt an das AMPS-System [Koco6], in einem Zusammenspiel aus Zustandsabstraktion und diversen Heuristiken zur Laufzeit den Perzeptionszustand zu einem symbolischen Zustand zu abstrahieren. Das Ergebnis ist ein sowohl hinreichend kleiner als auch hinreichend expressiver Zustandsraum ($\xi : \mathbb{R}^n \to \mathbb{N}^+$), der zusammen mit den Übergangswahrscheinlichkeiten $T(s, a, s')$ für (Zustand, Aktion, Folgezustand)-Tupel, dem geschätzten diskontierten Reward $R(s, a)$ und der Diskontierungsfunktion γ, das Modell bilden. Für dieses Modell generiert der Reinforcement Learning-Algorithmus auf Basis von Semi-Markov-Entscheidungsprozessen (SMDP) eine Strategie für jedes Element im Motivationsvektor.

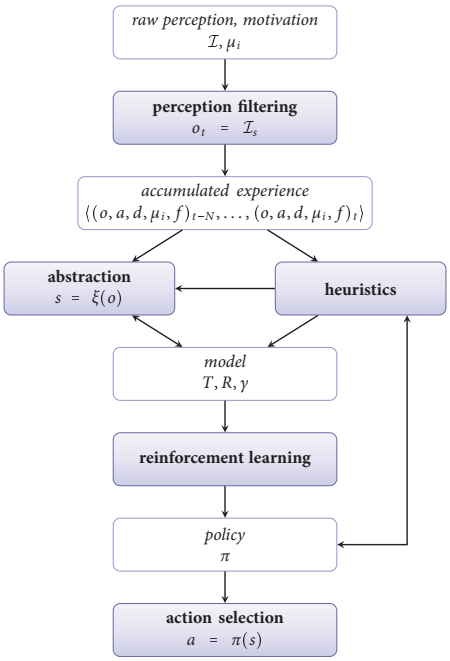

Abbildung 2: Strategieschicht: Von der Perzeption zur Aktionsauswahl

2.3 Skillschicht

Die Aktionen in der Strategieschicht sind abstrakt und nicht direkt in der Umwelt des Roboters ausführbar. Deshalb wurde im Rahmen der Arbeit eine Komponente entwickelt, mit der ein Roboter selbstständig seine eigenen Verhaltensmöglichkeiten erforschen und lernen kann [RTo9]. Diese Verhaltensmöglichkeiten werden in Form von Skills zur Laufzeit permanent optimiert. Das Ziel dieser Komponente ist es, der Strategieschicht diese Skills zur Verfügung zu stellen. Die Strategieschicht braucht nur noch mit abstrakten Symbolen als Aktionen zu operieren. Die werden dann von der Skillschicht in die jeweilige Folge von reaktiven Aktuatorkommandos ($\mathcal{O}$ in Abb. 1) abgebildet.

Ein Skill ist definiert als Tupel $a = (f_e^1, \ldots, f_e^N)$, wobei $f_e^i : \mathcal{I}_a \times \mathcal{I}_a \to \mathbb{R}^+$ eine Fehlerfunktion ist, die einem Perzeptionspaar $\big(I(t_i), I(t_j)\big)$ einen Fehlerwert zuweist[2]. Dieser Fehlerwert beschreibt, wie sehr der Skill seinem Ziel im Zeitintervall $[t_i, t_j]$ nähergekommen ist. Ein Beispielskill mit der Aufgabe "fahre zum Ball und richte dich zu ihm aus" könnte damit folgendermaßen definiert werden:

$f_e^1(I(t_i), I(t_j)) = d_{\text{ball}}(I(t_j))$ *minimiere die Balldistanz*
$f_e^2(I(t_i), I(t_j)) = |\alpha_{\text{ball}}(I(t_j))|$ *minimiere den Winkel zum Ball*
$a = (f_e^1, f_e^2)$ *zum Ball fahren und ausrichten*

Zusammen mit den Modellen, die zur Ausführung eines Skills notwendig sind, wird auch

[2] Der Skill a in der Skillschicht ist die Realisierung der abstrakten Aktion a in der Strategieschicht.

eine Fortschrittsfunktion $f_p : \mathcal{I}_a \times \mathcal{I}_a \rightarrow [0,1]$ gelernt, mit der der Roboter zu jedem Zeitpunkt ermitteln kann, wie weit der Skill mit der Erfüllung seiner Aufgabe vorangeschritten ist. Für einen Skill $a = (f_e^1, \ldots, f_e^N)$ ist sie definiert als

$$f_p(I(t_i), I(t_j)) = \begin{cases} 0 & \text{if } C_1 \leq W(I(t_i), I(t_j)) \\ \frac{C_1 - W(I(t_i), I(t_j))}{C_1 - C_2} & \text{if } C_2 < W(I(t_i), I(t_j)) < C_1 \\ 1 & \text{if } W(I(t_i), I(t_j)) \leq C_2 \end{cases}$$

Die dafür notwendigen Abbruch- und Erfolgsschwellwerte $C_1 \in \mathbb{R}^+$ und $C_2 \in \mathbb{R}^+$ ($C_2 < C_1$) werden vom Roboter selbstständig gelernt. $I(t_i)$ ist die Perzeption beim Starten des Skills, $I(t_j)$ die jeweils aktuelle Perzeption und $W(I(t_i), I(t_j)) = \sum_{k=1}^{N} f_e^k(I(t_i), I(t_j))$ der aktuelle Gesamtfehler des Skills.

Aufbauend auf diese Architektur bestehend aus Motivations-, Strategie- und Skillschicht, kann der in dieser Dissertation entwickelte Imitationsansatz Verhalten in der Beobachtungssequenz beliebiger Roboter erkennen und in das Verhalten des Roboters integrieren (Abschnitt 3). Die Frage, welcher Roboter imitiert werden kann, wird in Abschnitt 4 behandelt.

3 Imitation

Das Hauptziel der Dissertation war die Entwicklung eines Imitationsverfahrens, das ohne menschliche Intervention in autonomen Robotergruppen anwendbar ist. Der entstandene Imitationsansatz [RNK08] ist in Abb. 3 dargestellt. Zum Imitieren nimmt der Imitator (I) bereits im Vorfeld sämtliche beobachtbare Informationen über den entsprechenden Demonstrator (D) als Episode von (Perzeption o^I, Motivationszustand e)-Tupel auf. e ist dabei der vom Imitator wahrgenommene Motivationszustand des Demonstrators. Hieran kann der Imitator erkennen, ob die jüngsten Aktionen des Demonstrators eher positiv oder negativ zu bewerten sind. Die Perzeptionsdaten o^I werden anschließend in das egozentrische Koordinatensystem des Demonstrators transformiert (o^D). Dadurch kann der Imitator "mit den Augen des Demonstrators" die Daten sehen. In dem Interpretationsschritt erkennt der Imitator nun, welche Teilsequenzen der Beobachtung mit welchen eigenen Skills reproduziert werden können. Anschließend wird anhand des Motivationszustandes der Reward abgeschätzt. Nun liegen sämtliche Daten vor, um der eigenen Strategieschicht die in den Beobachtungsdaten erkannte Verhaltenssequenz in Form von Erfahrung zu übergeben. Nach einem Standard-Reinforcement Learning-Schritt ist die Beobachtung nun in die eigene Strategie integriert und wird beim nächsten Ausführungsschritt mitbenutzt. Nachfolgender Abschnitt konzentriert sich wegen des verfügbaren Umfanges auf den Interpretationsschritt.

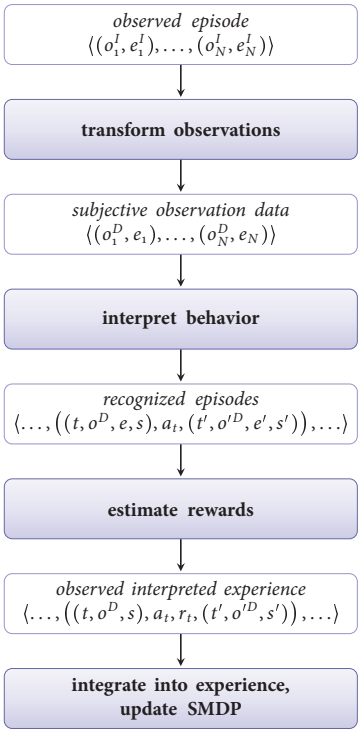

Abbildung 3: Imitationsprozess

Der Interpretationsschritt ist vom Viterbi-Algorithmus inspiriert [Rab89]. Dieser wird bei Hidden Markov Modellen zur effizienten Herleitung wahrscheinlicher Zustandsübergänge aus Beobachtungsdaten genutzt. Im Gegensatz zu Viterbi liegt in dem Interpretationsschritt jedoch die Herausforderung darin, nicht nur die Zustandsübergänge, sondern auch die sie hervorrufenden Aktionen zu erkennen (Abb. 4). Dazu wird für jedes Paar von Beobachtungsperzeptionen (o_{t-1}, o_t) die Differenz Δo_t berechnet. Diese kann mit Hilfe der in der Skillschicht gelernten Fortschrittsfunktionen f_p^a für jeden Skill a daraufhin untersucht werden, wie wahrscheinlich a die beobachtete Perzeptionsveränderung selbst hätte realisieren können. $f_p(I(t_i), I(t_j))$ emuliert dadurch das sogenannte Spiegelneuronensystem [RC04], das bei Menschen wichtige Funktionen bei der Imitation übernimmt: Die gleichen Neuronen, die feuern, wenn ein Mensch z.B. eine Tasse in die Hand nimmt, feuern auch, wenn der Mensch jemand anderen dabei beobachtet, wie er eine Tasse in die Hand nimmt. Wenn f_p nicht nur zur Bewertung seiner eigenen Skills benutzt wird, sondern wenn der Imitator für $I(t_i)$ und $I(t_j)$ die Perzeptionen eines Demonstrators abschätzen kann, so ist der imitierende Roboter in der Lage, das beobachtbare Verhalten eines anderen Roboters zu inferieren. Genau dieser Trick wird hier benutzt, um zusammen mit der selbst gelernten Strategie das Verhalten anderer Roboter zu erkennen und in die eigene Strategie einzubinden. Angenommen der Imitator hat zwei Skills: a_0 ist in der Lage auf ein Objekt zu zu fahren, und a_1 kann das Objekt anheben. Wenn der Imitator nun als Perzeptionsveränderung wahrnimmt, dass der Demonstrator seine Distanz zu einem Objekt verringert hat, wird wahrscheinlich $f_p^{a_0}$ für die entsprechende Veränderung einen höheren Wert berechnen als $f_p^{a_1}$.

Mittels $P_a(o_t \mid o_{t-1})$ kann darauf basierend die Wahrscheinlichkeit berechnet werden, mit der Skill a die Perzeptionsänderung Δo_t von o_{t-1} nach o_t realisieren kann:

$$P_a(o_t \mid o_{t-1}) = \begin{cases} \min\left(\max\left(\frac{f_p^a(o_t) - f_p^a(o_{t-1})}{1 - f_p^a(o_t)}, 0\right), 1\right), & 1 - f_p^a(o_t) < \epsilon \\ 0, & \text{sonst} \end{cases} \tag{1}$$

Formel (1) sorgt auf diese Weise dafür, dass Skills, die nach f_p^a weiter fortgeschritten sind, bevorzugt werden bzw. ignoriert werden, wenn sie ihr Ziel bereits erreicht haben. Es kann natürlich vorkommen, dass der Imitator für das beobachtete Verhalten keine Entsprechung in seinen eigenen Fähigkeiten findet – wenn er z.B. beobachtet, wie ein anderer Roboter ein Objekt hebt, während er selbst keinen Greifer besitzt. In diesem Fall wird der Interpretationsalgorithmus für das entsprechende Intervall in Abhängigkeit von $\epsilon \in (0, 1)$ signalisieren, dass hier ein nicht interpretierbares Verhalten vorliegt.

Die wahrscheinlichste Aktion a_{ml} zwischen zwei Zeitpunkten t_1 und t_2 ergibt sich als

$$a_{ml} = \arg\max_a \frac{\sum_{t=t_1}^{t_2} P_a(o_t \mid o_{t-1})}{t_2 - t_1} . \tag{2}$$

Mit ihr können dann die Transitionswahrscheinlichkeiten der Zustände berechnet werden:

$$P(s_{t_2} \mid s_{t_1}) = T(s_{t_1}, a_{ml}, s_{t_2}) \tag{3}$$

Der in dieser Dissertation entwickelte Algorithmus ermittelt mit Hilfe von Formeln (1)–(3) eine Sequenz von (Zustand, Aktion)-Tupeln. Diese beschreiben, welche Aktionen der

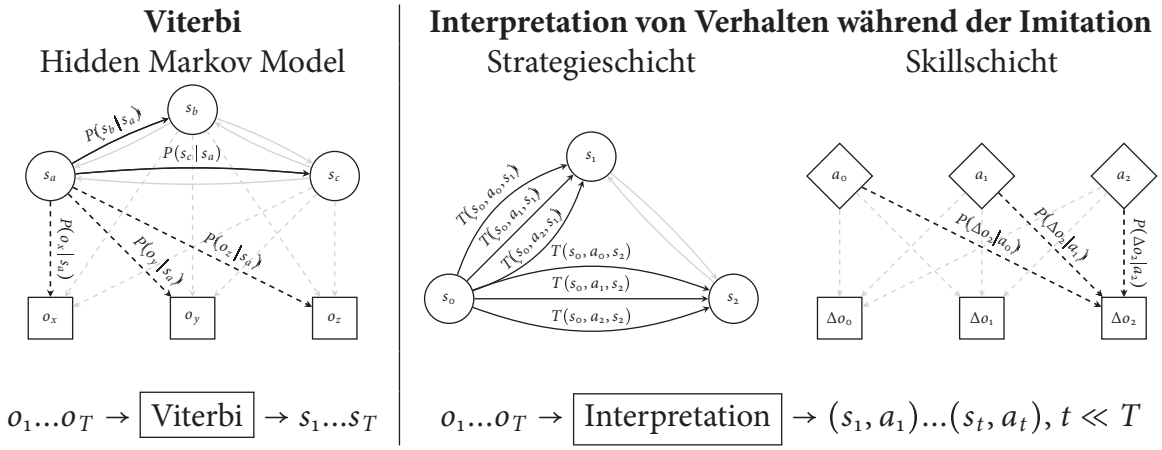

Abbildung 4: Vergleich von Viterbi und dem Interpretationsschritt in der Imitation: Im Gegensatz zu Viterbi müssen hier nicht nur Zustandsfolgen, sondern auch die sie verursachenden Aktionen bestimmt werden.

Imitator in welchen Zuständen ausführen muss, um das beobachtete Verhalten des Demonstrators zu reproduzieren. Diese Sequenz besteht dabei nur aus Informationen aus dem Verhaltensrepertoire des Imitators – keinerlei Zugriff auf die internen Datenstrukturen des Demonstrators ist dabei notwendig.

4 Auswahl des zu imitierenden Roboters

Zusätzlich zu den von der im vorigen Abschnitt beschriebenen Imitationskomponente, benötigt der Imitator jedoch noch die Information, *wer* überhaupt imitiert werden soll. Dies wird mit der Demonstratorauswahlkomponente realisiert. Die Schwierigkeit liegt darin, dass die Auswahl des zu imitierenden Roboters getroffen werden muss *bevor* die eigentliche Imitation stattfindet. D.h., dass der Roboter weder das Wissen über seine Strategie noch über seine Skills benutzen kann. In der Literatur sind deshalb keine Ansätze vorhanden, die zu dieser Entscheidung praktisch in der Lage sind.

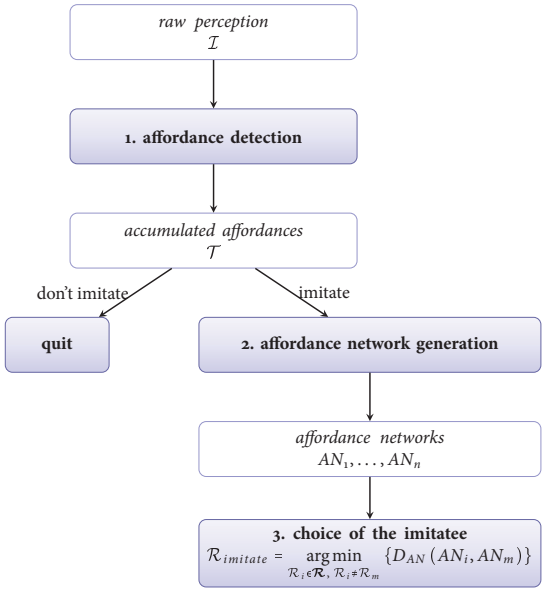

Abbildung 5: Demonstratorauswahlprozess

Der in dieser Dissertation entwickelte Ansatz (Abb. 4) sammelt und verdichtet hierzu Information bezüglich potentiell zu imitierender Roboter, die vor der Imitation selbst verfügbar sind. Diese Informationen sind sogenannte Affordanzen. Sie beschreiben Interaktionsmöglichkeiten zwischen einem Subjekt und einem Objekt. Ein Stuhl bietet einem Menschen z.B. die Affordanz *besitzbar*, während er einem Roboter diese nicht bietet.

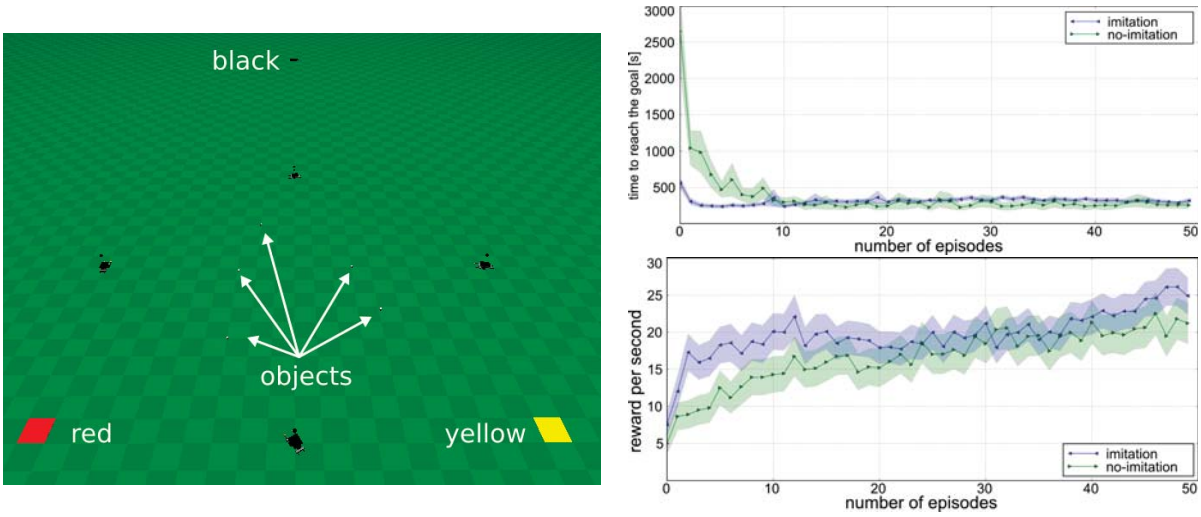

Abbildung 6: Links: Imitationszenario mit 3 Plattformen, auf die 4 Roboter 5 Objekte transportieren müssen. Graph oben rechts: Durchschnittlich benötigte Zeit, um ein Objekt zur Plattform zu transportieren mit 95% Konfidenzintervall. Graph unten rechts: Belohnung pro Zeiteinheit.

Erkennungsdetektoren von Affordanzen sind anwendungsneutral und können im Vorfeld dem Roboter vorgegeben werden. Der Imitator beobachtet in diesem Ansatz nun jeden potentiellen Demonstrator und speichert, welche Objekte dem jeweiligen Demonstrator welche Affordanzen anbieten. In dem Fall, dass ein Imitator imitieren möchte, generiert er sogenannte Affordanznetze. Dies sind Bayessche Netzwerke, die aus den Affordanzdaten mit dem *Alternating Model Selection EM*-Algorithmus [Fri97] gelernt worden sind. Für jeden Roboter (einschließlich dem Imitator selbst) liegt ihm nun ein Affordanznetzwerk vor. Mit einer in dieser Arbeit entwickelten Graphdistanzmetrik auf Affordanznetzwerken kann der Imitator nun feststellen, welcher Demonstrator ihm verhaltensmäßig am ähnlichsten ist: je kleiner die Graphdistanz zwischen dem Imitator und einem Demonstrator, desto ähnlicher sind sie sich. Der Demonstrator mit der geringsten Distanz wird dann imitiert.

5 Evaluation

Um eine ausreichende Signifikanz in den Ergebnissen zu erhalten, wurde der Imitationsansatz in einem physikalisch realistischen Simulator auf Basis der Open Dynamics Engine evaluiert. Der Simulator ermöglicht dabei das Generieren morphologisch unterschiedlicher Robotertypen, die damit in einer heterogenen Robotergruppe den Ansatz unter Beweis stellen können. In diesem Abschnitt wird auf zwei Experimente der Dissertation eingegangen, die jeweils den Imitationsalgorithmus und den Demonstratorauswahlalgorithmus untersuchen. Beide Experimente wurden jeweils in zwei Modi durchgeführt: Im ersten Modus durften die Roboter nur ihre Lernkomponenten benutzen, während im zweiten Lernen und Imitation verzahnt verwendet wurden.

Im ersten Experiment haben vier Roboter die Aufgabe, verschiedene in der Umgebung platzierte Objekte aufzusammeln und zu einer der drei Plattformen zu transportieren (Abb. 6). Hierbei ist die schwarze Plattform weiter entfernt, führt aber zu einer größeren Belohnung. Die Roboter sollen einzeln lernen, wie sie zu einem Objekt hinfahren und dieses dann zu

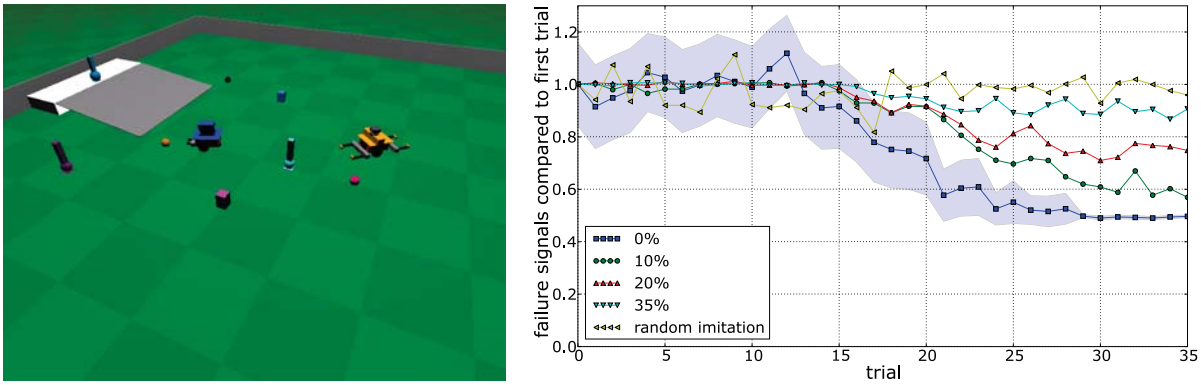

Abbildung 7: Demonstratorauswahl: Morphologisch unterschiedliche Roboter müssen entscheiden, welchen potentiellen Demonstrator sie jeweils imitieren. Der Einfluss des Demonstratorauswahlansatzes ist rechts zu sehen: Für verschiedene Rauschanteile in der Affordanzerkennung ist die normierte Fehlerrate der Imitation zu sehen (95% Konfidenzintervall im 0% Fall)

einer der Plattformen transportieren. Die Ergebnisse in diesem Fall sind in Abb. 6 als "no-imitation" gekennzeichnet. Sie können jedoch zusätzlich zum eigenen Lernprozess einander imitieren ("imitation"). In diesem Fall erzielen die Roboter gerade zu Beginn einen deutlichen Lernvorsprung – nicht nur in der Zeit, die sie benötigen, um ein Objekt zu einer der Plattformen zu transportieren (oberer Graph), sondern auch in der Belohnung pro Zeiteinheit (unterer Graph).

In dem zweiten Experiment müssen morphologisch unterschiedliche Roboter entscheiden wen sie imitieren möchten (Abb. 7). Dabei dürfen sie verschiedene Aktionen wie *heben*, *schieben* oder *ziehen* in einer sehr heterogenen Umwelt ausführen. Dafür standen Objekte mit unterschiedlicher Größe, Form und Gewicht zur Verfügung. Jeder Versuch ("trial") beginnt mit der Beobachtung einer Affordanz eines zufällig ausgewählten Demonstrators, die zur Affordanzdatenbasis hinzugefügt wird ("accumulated affordances" in Abb. 4). Aus dieser bildet der Imitator für den Demonstrator das Affordanznetzwerk neu. Anschließend berechnet der Imitator basierend auf den Distanzen zwischen seinem Affordanznetzwerk und denen der potentiellen Demonstratoren, welcher von ihnen ihm am ähnlichsten ist und imitiert ihn dann. Während der Imitation wird in der Skillschicht (Abschnitt 2.3) in Form von Fehlersignalen gemessen wie erfolgreich der Roboter gewesen ist. Dieser Wert ist normiert auf den ersten Versuch in der rechten Abbildung dargestellt. Um die Auswirkung von Ungenauigkeiten in der Perzeption auf die Affordanzerkennung zu berücksichtigen, wurde das Experiment für verschiedene Rauschanteile durchgeführt. Es zeigt sich, dass die intelligente Demonstratorauswahl gegenüber der zufälligen Auswahl deutliche Vorteile aufweist. Damit ist es zum ersten Mal möglich, Imitation in Robotergruppen einzusetzen. Basierend auf den erkannten Affordanzen kann jeder Roboter vollautomatisch den jeweils ihm passenden Demonstrator auswählen.

6 Zusammenfassung

Die hier zusammengefassten Beiträge der Dissertation erweitern das Feld der Robotik dadurch, dass nun erstmals Imitation in realistischen Robotergruppen anwendbar ist. Bei Verwendung der vorgestellten Architektur mit den darauf basierenden Methoden zur Imitation

sowie der Auswahl des zu imitierenden Roboters ist keine menschliche Intervention mehr notwendig. Dadurch können Roboter sogar in heterogenen Robotergruppen schneller ihre jeweiligen Aufgaben lernen. Dabei stellen sie nur minimale Anforderungen aneinander. Gerade in praxisrelevanten Einsatzszenarien, in denen die manuelle Programmierung zu aufwendig ist oder ein Lernprozess zu lange dauern würde, ermöglichen die Beiträge der vorgestellten Dissertation neue Anwendungsmöglichkeiten.

Literatur

[CAN09] Adam Coates, Pieter Abbeel und Andrew Y. Ng. Apprenticeship learning for helicopter control. *Communications of the ACM*, 52(7):97–105, 2009.

[DN02] K. Dautenhahn und C. Nehaniv. *Imitation in Animals and Artifacts*, Kapitel "An agent-based perspective on imitation". MIT Press, 2002.

[Fri97] N. Friedman. Learning belief networks in the presence of missing values and hidden variables. In *14th International Conference on Machine Learning*, 1997.

[Koc06] M.J. Kochenderfer. Adaptive Modelling and Planning for Learning Intelligent Behaviour. 2006.

[KSS06] W. Kadous, C. Sammut und R. Sheh. Autonomous Traversal of Rough Terrain Using Behavioural Cloning. In *The 3rd International Conference on Autonomous Robots and Agents*, 2006.

[Lie01] Henry Lieberman. *Your wish is my command: programming by example*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2001.

[Rab89] L.R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[RC04] G. Rizzolatti und L. Craighero. The Mirror-Neuron System. *Annual Review of Neuroscience*, 27(1):169–192, 2004.

[Ric09] W. Richert. *Learning and imitation in heterogeneous robot groups*. Dissertation, Universität Paderborn, 2009.

[RNK08] W. Richert, O. Niehörster und M. Koch. Layered understanding for sporadic imitation in a multi-robot scenario. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'08)*, 2008.

[RT09] W. Richert und R. Tornese. ESLAS – a robust layered learning framework. *International Journal On Advances in Intelligent Systems*, 2(1):241–253, May 2009.

Wilhelm Richert schloss 2009 seine Promotion mit dem Titel "Learning and imitation in heterogeneous robot groups" am Lehrstuhl von Prof. Dr. Franz J. Rammig an der Universität Paderborn mit Auszeichnung ab. Im Laufe seiner Dissertation verfasste er über 20 Publikationen (darunter Journalartikel und Buchbeiträge). Zur Zeit ist Wilhelm Richert wissenschaftlicher Mitarbeiter im C-Lab, der Innovationswerkstatt von Siemens und der Universität Paderborn.