Lernen komplexer Aufgaben aus Demonstration und eigener Erfahrung

Dissertation

zur

Erlangung des Doktorgrades (Dr. rer. nat.)

der

Mathematisch-Naturwissenschaftlichen Fakultät

der

Rheinischen Friedrich-Wilhelms-Universität Bonn

vorgelegt von

Kathrin Gräve

aus

Olpe

Bonn, 05.01.2015

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Rheinischen Friedrich-Wilhelms-Universität Bonn.

Erstgutachter: Prof. Dr. Sven Behnke, Bonn
Zweitgutachter: Prof. Dr. Maren Bennewitz, Bonn
Tag der Promotion: 22.10.2015
Erscheinungsjahr: 2015

Kurzfassung

Heutige Industrieproduktionen wären nicht möglich ohne die Erfindung von Robotern, die effizient und präzise sich ständig wiederholende Aufgaben ausführen. Gleichzeitig stellt die industrielle Fertigung das bisher einzige Gebiet dar, in dem Roboter in großem Maßstab eingesetzt werden. Dabei gibt es auch in anderen Bereichen des Alltags Aufgaben, bei denen Roboter Menschen sinnvoll unterstützen können, indem sie ihnen beispielsweise monotone oder lästige Aufgaben abnehmen. Diese Einsatzgebiete unterscheiden sich jedoch fundamental von den kontrollierten, sorgfältig abgeschirmten Umgebungen, in denen Roboter heutzutage in der Regel arbeiten. Auch sind die Aufgabenstellungen vielfältiger und komplexer als die meisten Industrieabläufe, so dass sich neue Herausforderungen für Serviceroboter ergeben. Aufgrund dieser Vielfältigkeit ist eine Programmierung der benötigten Fähigkeiten, die alle Ausprägungen der Aufgabe und Rahmenbedingungen berücksichtigt, nicht mehr praktikabel. Eine mögliche Alternative stellen Lernmethoden dar, die Robotern erlauben, die benötigten Fähigkeiten auf eine intuitive Art und Weise zu erlernen und diese bei Bedarf an neue Situationen anzupassen und zu ergänzen.

Das Ziel dieser Dissertation ist die Entwicklung von Methoden, die es Robotern ermöglichen, komplexe Fähigkeiten zur Erfüllung alltäglicher Aufgaben zu erlernen. Im ersten Teil dieser Arbeit wird ein Verfahren vorgestellt, welches anhand eines gemeinsamen probabilistischen Modells sowohl Bewegungstrajektorien segmentieren und klassifizieren als auch neue Trajektorien für Bewegungen zwischen beliebigen Endpunkten generieren kann. Als zentraler Bestandteil dieses Ansatzes wird ein inkrementeller Algorithmus zur simultanen Segmentierung und Klassifizierung entwickelt, der mit Hilfe einer Vorausschau sowohl das jeweils aktuelle als auch das darauffolgende Segment in die Bestimmung von Segmentgrenzen einbezieht. Auf diese Weise werden systematische Segmentierungsfehler vermieden. Das Verfahren wird zum Lernen aus von Menschen demonstrierten Bewegungsfolgen eingesetzt.

Der zweite Teil der Dissertation beschreibt Ansätze, um Robotern das Lösen komplexer Aufgaben durch eine gezielte Kombination einfacher Fähigkeiten beizubringen. Zunächst wird ein Lernverfahren vorgestellt, in dem Bestärkendes Lernen und Lernen aus Demonstrationen kombiniert werden, um einem Roboter auf intuitive Weise Bewegungssequenzen zum Lösen einer Aufgabe beizubringen. Die heterogene Zusammensetzung des Zustands- und Aktionsraumes sowie die variable Länge der Aktionssequenzen, die verschiedene Strategien zum Lösen einer Aufgabe repräsentieren, stellen hierbei besondere Herausforderungen dar. Diesen begegnet der hier vorgestellte Ansatz durch eine probabilistische Approximation der Nutzenfunktion über Zustands- und Aktionspaare mit einem speziell entwickelten, kombinierten Kernel. Diese Approximation liefert die Grundlage für eine Bayessche Explorationsstrategie, die auf der Optimierung der Erwarteten Veränderung basiert und ein effizientes Bestärkendes Lernen ermöglicht. Um eine bestmögliche Integration des Bestärkenden Lernens mit Expertenwissen aus Demonstrationen zu erreichen, wird ein mehrstufiges Entscheidungssystem genutzt, das in jeder Situation bestimmt, welches der beiden Lernmodule das geeignetere ist. Als Ergebnis wird ein Verfahren

erzielt, das ein sicheres, aber gleichzeitig auch effizientes Lernen von Bewegungssequenzen ermöglicht.

Mit zunehmender Länge der benötigten Aktionssequenzen und beim Einsatz in realistischen Umgebungen stoßen viele der heutigen Systeme schnell an ihre Leistungsgrenzen. Für praktische Anwendungen werden daher skalierbare Lernalgorithmen benötigt, die in Aufgabenstellungen vorhandene Strukturen berücksichtigen und Abstraktionsmöglichkeiten nutzen. Dazu wird in dieser Dissertation eine Erweiterung der MAXQ-Methode für hierarchisches Bestärkendes Lernen vorgestellt, um deren Nutzung in praktischen Anwendungen mit kontinuierlichen Zustandsräumen zu ermöglichen. Dabei wird eine Gauß-Prozess-Approximation der MAXQ-Zerlegung für jede Teilaufgabe eingesetzt, anhand derer das System rekursiv probabilistische Schätzungen der Q-Werte entlang der Aufgabenhierarchie berechnet. Diese erlaubt es, das bereits erfolgreich zum Lernen von Aktionssequenzen eingesetzte Bayessche Explorationskriterium auch zum effizienten Lernen von MAXQ-Hierarchien anzuwenden. Die so erweiterte MAXQ-Methode wird in jeder Teilaufgabe der MAXQ-Hierarchie mit dem Lernen aus Demonstrationen ergänzt und das System setzt für jede Teilaufgabe ein Entscheidungsmodul ein, um das jeweils am besten geeignete Verfahren zu ermitteln. Auch beim Lernen aus Demonstrationen nutzt das System die Vorteile der hierarchischen Aufgabenstruktur, indem gezielt Demonstrationen nur für Aufgabenteile anfordern werden, in denen diese tatsächlich benötigt werden. Somit werden unnötige redundante Demonstrationen vermieden und der Aufwand für den Experten verringert.

Die dazugehörigen Experimente, die sowohl in einer simulierten Umgebung als auch auf einem humanoiden Roboter durchführen werden, zeigen, dass die in dieser Arbeit vorgestellten Verfahren ein effizientes und sicheres Lernen komplexer Aufgaben ermöglichen und somit einen Beitrag für einen möglichen zukünftigen Einsatz von Robotern für alltägliche Aufgaben darstellen.

Abstract

Today's industrial mass production would not be possible without the invention of robots that are able to carry out repetitive tasks efficiently and precisely. Industrial manufacturing tasks, in fact, remain the only field to date where robots have been successfully deployed on a large scale. There are various tasks in every day life however, where robots could assist humans with monotonous or tedious tasks. The requirements for service robots in human environments differ substantially from those for industrial robots that typically operate in well-controlled and carefully isolated environments. Tasks in everyday life typically require a broader range of skills, and are more complex than in a fabrication environment. The complexity of the tasks and the environment today is one of the major obstacles for the development of robots that can be used in everyday life. Manually programming a robot with the required skills in a way that takes into account all possible manifestations of a task and its environment is simply not feasible. Instead, learning methods that enable robots to acquire the necessary skills in an interactive and intuitive manner, and to adapt and extend them as needed, offer a promising alternative.

This dissertation investigates learning methods that enable robots to learn complex skills that are needed to solve tasks in everyday life. First, a new method for the segmentation and classification of action sequences and the reproduction of generalized movements between arbitrary locations based on a shared probabilistic model of action classes is presented. At the core of this method is an incremental algorithm for the simultaneous segmentation and classification of trajectory data. The algorithm delineates segments using a look-ahead on the sequential stream of trajectory data and following a probabilistic approach that takes into account the current segment as well as the subsequent one. In this way, the presented algorithm avoids a systematic bias, and achieves reliable segmentation and classification results. These are a prerequisite for the recognition of actions in human demonstrations throughout the remainder of this dissertation.

In the second part of this dissertation, new approaches to teach robots to solve complex tasks by combining elementary skills in a suitable manner are developed. First, the problem of learning sequential tasks is considered, which leads to the development of a system that combines reinforcement learning with learning from demonstrations to create an intuitive way of teaching robots. The tasks considered are characterized by heterogeneous state and action spaces and variable-length action sequences representing different strategies for a task. These issues pose a significant challenge to learning methods. They are addressed by approximating the state-action value function using a probabilistic Gaussian Process model with a custom-developed, combined kernel. Based on this approximation, a Bayesian exploration strategy is devised that seeks to optimize the Expected Deviation. This allows for efficient and safe action selection during reinforcement learning. In order to introduce expert knowledge into the reinforcement learning process as effectively as possible, a multi-level decision module is introduced that determines whether reinforcement learning or learning by asking an expert is more effective in a given situation. The presented work results in a system that enables a robot

to safely, yet efficiently learn variable-length action sequences for solving manipulation tasks.

In real world settings, the requirements for solving tasks that require action sequences of increasing length quickly exceed the capabilities of many contemporary systems for robot learning. Therefore, new learning algorithms need to be developed that take the inherent structure of tasks into account and harness opportunities for abstraction to achieve scalability. Hierarchical methods for robot learning accomplish this goal by decomposing tasks recursively into a hierarchy of subtasks with reduced complexity. This dissertation proposes an extension to the MAXQ method for hierarchical reinforcement learning to allow for its application to practical tasks exhibiting continuous state spaces. For every subtask, a Gaussian Process approximation is applied to the components of the MAXQ decomposition. Based on these approximations, probabilistic estimates of state-action values are computed recursively along the subtask hierarchy and the previously introduced Bayesian exploration strategy is applied to efficiently learn subtask policies. On every subtask of the MAXQ hierarchy, the presented method complements the enhanced MAXQ learning algorithm with learning from demonstrations and implements a decision module that determines the most appropriate learning method independently for every subtask in every iteration. When learning from demonstrations, the presented method again takes advantage of the hierarchical task structure by specifically asking for demonstrations only for subtasks where a reliable solution cannot be determined autonomously based on the available data. This strategy greatly reduces the number of unnecessary and redundant demonstrations that a human expert needs to give the system, and thereby renders learning more feasible.

Experimental results from a simulated environment and on a humanoid robot indicate that the approaches presented in this dissertation facilitate efficient and safe learning of complex tasks. In this way, the presented work provides a step towards the goal of applying robots to service tasks in everyday life.

Danksagung

Während dieser Arbeit habe ich viele Menschen kennengelernt, die mich in den letzten Jahren begleitet haben und ohne die diese Arbeit nicht möglich gewesen wäre, und daher möchte ich an dieser Stelle ein kurzes Danke aussprechen. Ich möchte Prof. Dr. Sven Behnke danken, dass er mich die letzten Jahre betreut hat und ich Teil seiner Arbeitsgruppe sein durfte, in der ich sehr viel gelernt habe. Zudem danke ich der B-IT-Research School für mein Stipendium, das mir den Freiraum gegeben hat, mich auf mein Forschungsgebiet zu konzentrieren.

Ich danke meinen Arbeitskollegen und vor allem meinen Büromitbewohnern für ihre Unterstützung, die Diskussionen, die tolle Atmosphäre, dafür dass wir immer ein super Team waren und die vielen schönen RoboCup-Ereignisse. Ich werde euch vermissen. Zudem ein dickes Danke an Dr. Nils Goerke, Wilfried Kunze, Philipp Allgeuer, Deborah Busch, David Dröschel, Benedikt Lemmen, Malte Lohmeyer und Henning Schal, die Teile meiner Arbeit Korrektur gelesen haben. Zusätzlich ein Danke an Michael Schreiber, der immer eine helfende Hand hatte, wenn diese für Experimente gebraucht wurde.

Vor allem aber danke ich meinem Freund Christoph Kunze für das jahrelange Verständnis, die Ermutigungen, seine Unterstützung und die vielen Iterationen des Korrekturlesens dieser Arbeit und meiner Veröffentlichungen. Abschließend möchte ich meiner Familie und meinen Freunden danken, die immer an meiner Seite standen und für mich da waren.

Inhaltsverzeichnis

1 Einleitung

Kapitel

Industrieroboter sind schon seit vielen Jahren Bestandteil der Gesellschaft und prägen die industrielle Massenproduktion, aus der sie gar nicht mehr wegzudenken sind. Sie nehmen Menschen Aufgaben ab, die schwer, schmutzig oder monoton sind und in manchen Fällen sogar eine Gefahr für Menschen darstellen. In industriellen Anwendungen sind vor allem ihre hohe Präzision, Stärke und Schnelligkeit wichtige Faktoren, um hohe Stückzahlen und Qualitätsstandards zu erreichen. Roboter arbeiten dort typischerweise in sehr kontrollierten Umgebungen und werden dafür eingesetzt, vordefinierte und hochspezialisierte Aufgaben zu bewältigen, die sich unter stets gleichbleibenden Bedingungen ständig wiederholen. Zudem verrichten sie ihre Arbeit meistens in abgegrenzten Bereichen, da Menschen sich aus Sicherheitsgründen während des laufenden Einsatzes nicht in ihrer Nähe befinden dürfen.

Durch den technologischen Fortschritt der vergangenen Jahre werden Roboter zunehmend auch für Anwendungen außerhalb der Industrie interessant. Die Entwicklung autonomer Roboter stellt daher heute einen der großen Forschungsschwerpunkte im Bereich der Robotik dar. Die möglichen Einsatzgebiete decken dabei ein breites Anwendungsspektrum ab. Sie reichen von der Untersuchung unzugänglicher Gebiete, wie beispielsweise der des Weltraumes oder des Meeresgrundes, über die Unterstützung bei der Bewältigung von Gefahrensituationen, wie etwa nach Erdbeben oder anderen Naturkatastrophen, bis hin zu alltäglichen Aufgaben im privaten Lebensbereich. Auch hier fallen viele unangenehme, zeitraubende und lästige Arbeiten an, bei denen Roboter Menschen im Alltag unterstützen können, wie zum Beispiel im Haushalt. Daneben gibt es weitere Servicebereiche, in denen Roboter zum Beispiel Aufgaben übernehmen, die ältere oder kranke Menschen alleine nicht mehr bewältigen können, so dass diese länger eigenständig in ihrem eigenen Haushalt leben können.

Die Anforderungen, die für diese Art von Aufgaben benötigt werden, unterscheiden sich allerdings stark von denen, die an Industrieroboter gestellt werden, so dass sich bei der Entwicklung von Servicerobotern viele neue Herausforderungen ergeben. Eine solche stellt beispielsweise der Schritt aus der kontrollierten Welt der Industrieroboter in unvorhersehbare, sich ständig ändernde Umgebungen dar, die deutlich weniger strukturiert sind als typische Produktionsstätten, in denen Roboter derzeit eingesetzt werden. Dazu kommt, dass das Aussehen dieser Umgebungen im Servicebereich stark variieren kann. Eine weitere Herausforderung besteht darin, dass Roboter nicht länger isoliert arbeiten

sollen, sondern in ihrer Umgebung gemeinsam mit Menschen agieren und teilweise sogar direkt mit ihnen zusammenarbeiten müssen. Für alltägliche Aufgaben muss ein Roboter somit in der Lage sein, eine große Anzahl verschiedener Situationen zu verarbeiten und sich in sich ständig dynamisch verändernden Umgebungen zurechtzufinden. Des Weiteren muss der Roboter die Anwesenheit von Menschen in seinem Arbeitsbereich berücksichtigen und es muss dabei garantiert werden, dass der Roboter weder in den Haushalten, in denen er arbeitet, Schaden anrichtet, noch bei der Kooperation mit Menschen ein Risiko für diese darstellt.

Anhand der Statistiken über den Verkauf von Servicerobotern in der heutigen Gesellschaft [1] lässt sich erkennen, dass die Anzahl der Roboter für Aufgaben im privaten Bereich in den letzten Jahren sprunghaft angestiegen ist. In den meisten Fällen sind diese Roboter allerdings spezialisierte Maschinen, die für vergleichsweise einfache Aufgaben ausgelegt sind und auch nicht auf andere Aufgaben erweitert werden können. Bekannte Beispiele sind Roboter, die Menschen das Staubsaugen oder das Rasenmähen abnehmen. Allerdings muss auch für diese Roboter zumeist eine wenigstens einigermaßen kontrollierte Umgebung geschaffen werden, damit ihr Einsatz problemlos möglich ist. Beispielsweise muss für einige Rasenmäher-Modelle eine Drahtschleife im Boden verlegt werden, um den zu mähenden Bereich zu markieren. Um Roboter in großem Maßstab für Aufgaben im Alltag einsetzen zu können, sind statt solch spezialisierter Maschinen flexible Systeme erforderlich, die für unterschiedliche Aufgaben eingesetzt werden können. Diese müssen darüber hinaus in der Lage sein, sich in eine für Menschen geschaffene Umgebung zu integrieren, ohne dass diese dafür angepasst werden muss.

Erste Ansätze für die Entwicklung von Robotern, die in der Lage sind, auch komplexere Aufgaben in einer realitätsnahen Umgebung zu lösen, sind bereits in der Forschung zu finden. Bis heute ist es allerdings nicht gelungen, ein System zu entwickeln, das in der Lage ist, autonom in einer komplexen, sich ständig verändernden Umgebung, wie beispielsweise einem Haushalt, zu arbeiten und dort eine Vielzahl von Aufgaben zu erledigen. Ansätze der Programmierung, wie sie zumeist in der Industrierobotik verwendet werden, stoßen hier aufgrund der großen Anzahl an Aufgaben und der zu erwartenden Vielfalt der Ausgangssituationen an ihre Grenzen. In der Praxis ist es außerdem schwierig, die konkreten Einsatzbedingungen eines Roboters im Voraus zur Entwicklungszeit des Systems abzusehen, wenn die Umgebung des Roboters nicht extra für den Roboter angepasst werden soll. Hinzu kommt, dass die Umwelt ständigen Veränderungen unterworfen ist und sich somit auch die Einsatzbedingungen für Roboter und die von ihnen zu erledigenden Aufgaben mit der Zeit verändern und immer wieder neue, vom Programmierer nicht vorhergesehene Situationen hinzukommen. Um Roboter sinnvoll im alltäglichen Leben einsetzen zu können, werden daher in Zukunft hochgradig flexible Robotersysteme benötigt, die es ihren Anwendern ermöglichen, die Fähigkeiten des Roboters entsprechend ihrer individuellen Bedürfnisse zu gestalten und sie im Laufe der Zeit an veränderte Bedingungen anzupassen und zu erweitern.

Als Alternative zur klassischen Programmierung werden zu diesem Zweck maschinelle Lernverfahren als effizientes Mittel des Wissenstransfers intensiv erforscht. Diese ermöglichen es auf der einen Seite, Robotern die benötigten Fähigkeiten inkrementell zu vermitteln, und stellen darüber hinaus einen Weg für das System dar, seine Fähigkeiten im Laufe der Zeit autonom zu verbessern und an veränderte Rahmenbedingungen

anzupassen. In der Robotikforschung werden dabei Prinzipien untersucht, die aus der menschlichen Lernpsychologie bekannt sind und Menschen einen effizienten Erwerb von Fähigkeiten ermöglichen.

Eine besondere Rolle beim menschlichen Lernen spielen das Lernen am Modell und das Lernen aus eigener Erfahrung. Beim Lernen am Modell eignet sich der Mensch Fähigkeiten durch Beobachtung und Nachahmung eines Vorbildes an. Lernt er anhand von Erfahrungen, so entwickelt und erprobt der Mensch eigene Lösungsstrategien und erkennt durch Beobachtung der Auswirkungen seiner Aktionen, ob diese zur Lösung einer Aufgabe geeignet sind. Um sich effizient Fähigkeiten anzueignen, nutzen Menschen mehrere Arten des Lernens gleichzeitig. Das Lernen am Modell und das Lernen aus eigener Erfahrung stellen dabei eine besonders effiziente Kombination dar. Die Nachahmung eines Vorbildes ermöglicht es dem Menschen, vorhandenes Wissen über geeignete Aktionen zum Lösen einer Aufgabe zu nutzen. Auf diese Weise können Menschen sich auch komplexe Fähigkeiten schnell aneignen. Gleichzeitig stellt das Demonstrieren von Fähigkeiten eine für Menschen intuitive und effiziente Art und Weise dar, einander Wissen zu vermitteln. Durch Nachahmung eines Vorbildes allein lässt sich eine Fähigkeit allerdings oft nur bis zu einem gewissen Grad erwerben, und es erfordert selbstständige Übung, um die vorgeführte Aufgabe anschließend selbst lösen zu können. Dabei wird der Lernprozess durch das anhand der Nachahmung erlangte Wissen beschleunigt. Die Vorteile des Lernens am Modell und des autonomen Lernens aus eigener Erfahrung sind somit komplementär, und ihre Kombination ermöglicht durch den Einsatz des geeigneten Verfahrens in den entsprechenden Situationen ein effizientes Lernen.

Damit Roboter Fähigkeiten ebenfalls durch Lernen erwerben können, werden diese Arten des Lernens auch bei der Entwicklung maschineller Lernverfahren in der Robotik eingesetzt und dort als Lernen aus Demonstrationen und Bestärkendes Lernen bezeichnet. Mit Hilfe von Demonstrationen könnten einem Roboter Fähigkeiten auf eine ähnliche Art und Weise wie einem Menschen beigebracht werden. Somit stellt dies eine intuitive Art des Wissenstransfers für einen Experten dar, die keinerlei technisches Vorwissen erfordert und die Akzeptanz des Systems bei potenziellen Nutzern fördert.

Um effiziente Lernverfahren für Roboter zu entwickeln, ist es auch in diesem Bereich ein vielversprechender Ansatz, die Vorteile eines kombinierten Ansatzes zu nutzen, indem Bestärkendes Lernen mit Lernen aus Expertenwissen, zum Beispiel in Form von Demonstrationen, ergänzt wird. Dabei kann das Wissen des Experten genutzt werden, um die Suche nach geeigneten Aktionen für das Bestärkende Lernen zu fokussieren und somit zu beschleunigen. Auf der anderen Seite werden im Bestärkenden Lernen gesammelte Erfahrungen generalisiert, um Fähigkeiten auch unter veränderten oder gänzlich unbekannten Voraussetzungen anwenden zu können. Somit kann ein Roboter anhand nur weniger Beispiele, die von einem Experten gegeben werden, Fähigkeiten erlernen und der Arbeitsaufwand für den Experten wird gering gehalten.

Neben einem effizienten Lernen ist Sicherheit ein wichtiger Aspekt für die Entwicklung von Robotern, die in häuslichen Umgebungen und im direkten Kontakt mit Menschen eingesetzt werden sollen. Dazu ist es unter anderem erforderlich, dass ein Roboter risikolos lernt, was bedeutet, dass auf der Suche nach geeigneten Aktionen zum Lösen einer Aufgabe kein Risiko eingegangen werden darf, dass ein Schaden für die Umgebung, die Menschen darin oder den Roboter selbst entstehen kann.

Trotz der geschickten Kombination von Lernverfahren stellen reale Aufgaben aus dem alltäglichen Leben heute immer noch eine Herausforderung für lernende Systeme dar. Die Komplexität der häuslichen Umgebung und die situationsabhängigen Strategien, die zur Lösung dieser Art von Aufgaben benötigt werden, verursachen einen enormen Lernaufwand. Daher ist es wichtig, dass Lernverfahren mit diesen Herausforderungen skalieren können und die zur Verfügung stehenden Daten effizient nutzen. In Anlehnung an die Art und Weise, wie Menschen komplexe Aufgaben lösen, indem sie diese rekursiv in leichter zu lösende Teilaufgaben gliedern, stellen hierarchische Ansätze einen vielversprechenden Ansatz zur Entwicklung skalierbarer Lernverfahren dar. Sie nutzen dabei die Struktur der Aufgabenstellung und bilden diese auf die Struktur des Lernverfahrens ab. Dies ermöglicht die Nutzung verschiedener Arten von Abstraktionen, durch die das Lernen von Teilaufgaben auf den unterschiedlichen Komplexitätsebenen der Aufgabe vereinfacht wird. Zudem ergibt sich der Vorteil, dass gelernte Aufgabenteile in unterschiedlichen Kontexten wiederverwendet werden können, was bei einem flachen Lernansatz nicht möglich wäre.

Die Kombination eines hierarchischen Lernverfahrens mit der Verwendung verschiedener Lernverfahren in Form von Bestärkendem Lernen und Lernen aus Demonstrationen verspricht somit, dass Roboter effizient und sicher komplexe Fähigkeiten erlernen können. Diese Dissertation beschäftigt sich mit diesen Ansätzen, um effiziente Lernverfahren zu entwickeln, die es möglich machen, Roboter für praktische Aufgaben einzusetzen.

1.1 Wissenschaftlicher Beitrag

In dieser Arbeit werden verschiedene Ansätze und Algorithmen vorgestellt, die es ermöglichen, dass Roboter komplexe Aufgaben durch Kombination geeigneter Aktionen lernen können, und somit dazu beitragen, dass Roboter im alltäglichen Leben eingesetzt werden können. Die Arbeiten sind allgemein in die Bereiche der Robotik und des Maschinellen Lernens einzuordnen und leisten Beiträge zur Erkennung und Generierung von Aktionssequenzen sowie zum Lernen sequentieller und hierarchischer Aufgaben. Die vorgestellten Verfahren werden dabei sowohl in einer physikbasierten Simulation als auch auf einem humanoiden Roboter evaluiert.

Aktionserkennung und Generalisierung von Bewegungen

Das Lösen komplexer Aufgaben erfordert die Kombination mehrerer einfacher Bewegungen, um einen gewünschten Effekt in der Umwelt zu erzielen. In dieser Arbeit werden dazu Ansätze vorgestellt, in denen Roboter diese Fähigkeit unter anderem durch Lernen aus Demonstrationen erwerben. Eine Voraussetzung dazu ist, die Struktur von demonstrierten Bewegungssequenzen zu erkennen und diese in ihre Bestandteile zu zerlegen.

Hierzu wird in dieser Arbeit ein Verfahren entwickelt, das dieser Herausforderung mit einer probabilistischen Repräsentation von Aktionsklassen begegnet. Diese ermöglicht die Verwendung eines gemeinsamen Modells, das sowohl zur Segmentierung von Bewegungssequenzen als auch für die Zuordnung der einzelnen Segmente zu Aktionsklassen und die Generierung von Bewegungen zwischen beliebigen Endpunkten eingesetzt wird. Da sich die Segmentierung einer Bewegungssequenz und die Klassifizierung der einzelnen Teilstücke gegenseitig beeinflussen, werden in dem hier vorgestelltem Ansatz

beide Aufgaben mit einem gemeinsamen Algorithmus gelöst. Eine Besonderheit dieses Algorithmus ist, dass er eine nicht-gierige, inkrementelle Strategie zur Klassifizierung und Segmentierung beliebig langer Eingabesequenzen verwendet. Dafür wird zur Bestimmung einer Segmentgrenze, welche auf der einen Seite das Ende, andererseits aber auch gleichzeitig den Anfang eines Segments darstellt, sowohl das vorangehende als auch das nachfolgende Segment berücksichtigt. Somit erreicht der hier vorgestellte Ansatz zuverlässige Segmentierungsergebnisse und vermeidet systematische Fehler.

Zur internen Repräsentation von Trajektorien transformiert der Ansatz diese in eine in dieser Arbeit entwickelte Merkmalsrepräsentation. Diese hat die Eigenschaft, dass sie invertierbar ist und sich besonders zur Kodierung gerichteter Bewegungen eignet, da sie die charakteristischen Eigenschaften der Bewegungen erhält, während sie von den Endpunkten der Bewegungen abstrahiert. Somit erlaubt es diese Repräsentation, mit dem präsentierten Ansatz gelernte Bewegungen zu generalisieren und Trajektorien zwischen beliebigen Endpunkten zu erzeugen, auch wenn für diese keine Trainingsdaten beobachtet wurden. Die Wahl der Merkmale macht es möglich, das Verfahren auf ein breites Spektrum gerichteter Bewegungen anzuwenden.

Die durchgeführten Experimente zeigen, dass die Qualität der Segmentierung und Klassifizierung von Bewegungssequenzen durch den vorausschauenden Ansatz unter Einbeziehung des nachfolgenden Segments steigt und dass mit diesem sowohl lange Sequenzen als auch unterschiedliche Arten von Bewegungen korrekt segmentiert und klassifiziert werden können. Des Weiteren führt der vorgestellte Algorithmus zur Generierung von Bewegungen zu glatten Trajektorien zwischen beliebigen Endpunkten.

Die in diesem Kapitel vorgestellten Arbeiten basieren auf der eigenen Veröffentlichung

Incremental Action Recognition and Generalizing Motion Generation based on Goal-Directed Features
Kathrin Gräve und Sven Behnke
IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vilamoura, Portugal, Oktober 2012.

Lernen sequentieller Aufgaben

Eine häufige Annahme bei der Suche nach Lösungen für komplexe Aufgaben ist, dass sich diese als Sequenzen elementarer Bewegungsprimitive auffassen lassen. In dieser Arbeit wird daher ein Verfahren entwickelt, mit dem Roboter die zum Lösen einer Aufgabe benötigten Aktionssequenzen auf eine intuitive Art lernen können. Dabei wird zum Lernen eine geschickte Kombination aus Bestärkendem Lernen und Lernen aus Demonstrationen eingesetzt, bei der sich der Roboter in jeder Episode aktiv für eine Art des Lernens entscheidet. Auf diese Weise werden in diesem Verfahren die gegensätzlichen Vorteile beider Arten des Lernens vereint, so dass es auch für komplexe Aufgaben möglich ist, diese effizient und sicher zu lernen. Dieses wäre mit einem dieser Verfahren alleine oftmals nicht möglich. Durch Integration einer interaktiven Komponente in Form eines Lösungsvorschlages kann die benötigte Anzahl an Demonstrationen eines Experten in diesem Lernverfahren zusätzlich erheblich reduziert werden.

Eine besondere Herausforderung beim Lernen von Aktionssequenzen stellt die Kombination aus einem zumeist kontinuierlichen Zustandsraum mit einem diskretwertigen

Raum der Aktionssequenzen dar. Dazu kommt, dass die Lösung einer Aufgabe in verschiedenen Situationen unterschiedliche Lösungsstrategien mit einer unterschiedlichen Anzahl an Aktionen erfordern kann. Daher wird zur Generalisierung über den kombinierten Raum der Zustände und Aktionssequenzen eine Approximation der Nutzenfunktion durch Gauß-Prozesse in Kombination mit einem speziell in dieser Arbeit entwickelten Kernel genutzt, der es zum einen erlaubt, mit verschiedenen Repräsentationen zu arbeiten, und gleichzeitig den Umgang mit unterschiedlich langen Aktionssequenzen ermöglicht. Aufbauend auf dieser Approximation ermöglicht der Ansatz ein effizientes und gleichzeitig sicheres Bestärkendes Lernen, indem zur Auswahl von Aktionen ein Bayessches Explorationskriterium eingesetzt wird, dessen Ziel die Optimierung des Wertes der Erwarteten Veränderung ist. Hierdurch werden vielversprechende Aktionssequenzen in jeder Episode ausgewählt, während gleichzeitig darauf geachtet wird, dass die gewählten Sequenzen kein Risiko für den Roboter oder seine Umgebung darstellen.

Anhand einer Beispielaufgabe wird demonstriert, dass die hier vorgestellten Lernverfahren effizient und sicher Aktionssequenzen für verschiedene Strategien lernen können, die in unterschiedlichen Situationen zur Lösung der Aufgabe benötigt werden. Dabei profitieren die Verfahren von den Vorteilen, die sich durch den vorgeschlagenen kombinierten Lernansatz ergeben.

Die in diesem Kapitel vorgestellten Arbeiten basieren auf der eigenen Veröffentlichung

Learning Sequential Tasks Interactively from Demonstrations and Own Experience
Kathrin Gräve und Sven Behnke
IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),
Tokyo, Japan, November 2013
Finalist des CoTeSys Cognitive Robotics Best Paper Award.

Lernen hierarchischer Aufgaben

Viele alltägliche Aufgaben zeichnen sich durch sehr große, kontinuierliche Situationsräume und komplexe Aufgabenstellungen aus. Der Einsatz von flachen Lernverfahren ist aufgrund ihrer begrenzten Skalierungsfähigkeit unter diesen Bedingungen nicht praktikabel. Daher wird in dieser Arbeit, inspiriert durch das Lernen von Menschen, ein hierarchisches Lernverfahren entwickelt, das sich die Struktur einer Aufgabe zunutze macht, um Redundanzen beim Lernen zu vermeiden und die Aufgabe durch Nutzung von Abstraktionen in eine Hierarchie einfacherer Teilaufgaben zu zerlegen. Dazu wird die MAXQ-Methode für hierarchisches Bestärkendes Lernen für den Einsatz in kontinuierlichen Zustandsräumen erweitert. Die Nutzenfunktionen der einzelnen Teilaufgaben werden dazu auf den unterschiedlichen Hierarchieebenen durch unabhängige Gauß-Prozesse approximiert. Indem in dem hier vorgestellten Ansatz die Gauß-Prozess-Schätzungen der Komponenten der MAXQ-Zerlegung und die mit ihnen verbundenen Unsicherheiten rekursiv über die verschiedenen Ebenen der MAXQ-Hierarchie aggregiert werden, erhält das System eine probabilistische Schätzung der Nutzenwerte für beliebige Teilaufgaben. Dies ermöglicht es, auch im hierarchischen Bestärkenden Lernen Aktionen anhand eines Bayesschen Explorationskriteriums basierend auf der Optimierung der Erwarteten Veränderung auszuwählen. Somit wird auch in diesem Verfahren nicht nur für eine effiziente Aktionsauswahl gesorgt, sondern auch die Sicherheit des Roboters und seiner Umwelt gewährleistet. Das Bestärken-

de Lernverfahren ist allerdings nur einer von zwei alternativen Kontrollflüssen in diesem Ansatz. Wie schon beim Lernen von Aktionssequenzen, wird das Bestärkende Lernen mit dem Lernen aus Demonstrationen in einem gemeinsamen System ergänzt, um von den Vorteilen beider Verfahren zu profitieren. Dazu wird in jeder Situation und für jede Teilaufgabe auf jeder Ebene der Aufgabenhierarchie individuell die Entscheidung für das besser geeignete Lernverfahren getroffen. Durch diese individuelle Entscheidung können Demonstrationen eines Experten gezielt eingesetzt werden. Dabei muss nur die angefragte Teilaufgabe auf der gewünschten Ebene demonstriert werden, wodurch Wiederholungen von Aufgaben in diesem Ansatz vermieden werden und die Arbeitslast des Experten reduziert wird. Durch den modularen Aufbau des Systems und durch die Nutzung des gleichen Lernverfahrens auf jeder der Ebenen ist das System einfach und nur mit wenigen Parameteranpassungen um neue Ebenen erweiterbar. Somit ist das System für Aufgaben verschiedener Komplexitätsstufen nutzbar.

Die hier vorgestellten Experimente demonstrieren, dass der vorgeschlagene hierarchische Lernansatz mit einer Kombination aus Bestärkendem Lernen und Lernen aus Demonstrationen in der Lage ist, effizient und sicher verschiedene situationsabhängige Strategien zum Lösen einer Aufgabe zu erlernen. Dabei profitiert das System von der hierarchischen Struktur der Aufgabe, indem die Gesamtaufgabe in einfachere Teilaufgaben gegliedert wird und diese von verschiedenen Elternaufgaben gemeinsam gelernt und genutzt werden können.

Die in diesem Kapitel vorgestellten Arbeiten basieren auf der eigenen Veröffentlichung

Bayesian Exploration and Interactive Demonstration in Continuous State MAXQ-Learning
Kathrin Gräve und Sven Behnke
IEEE International Conference on Robotics and Automation (ICRA),
Hong Kong, Mai 2014.

Die in dieser Dissertation vorgestellten Verfahren demonstrieren verschiedene, teilweise aufeinander aufbauende Ansätze, um die Anwendung von Lernverfahren für Roboter in praktischen Anwendungen im alltäglichen Leben zu ermöglichen.

1.2 Gliederung

Diese Arbeit ist in sechs thematisch zusammenhängende und in sich abgeschlossene Kapitel unterteilt, so dass jedes Kapitel bei Bedarf unabhängig von den anderen gelesen werden kann. Trotzdem bauen die Kapitel aufeinander auf, indem Konzepte aus vorangegangenen Kapiteln in späteren Kapiteln bei der Entwicklung neuer Ansätze aufgegriffen werden. An diesen Stellen werden die zum Verständnis benötigten Grundlagen erwähnt, während für Detailfragen auf die entsprechenden Kapitel verwiesen wird.

In *Kapitel 2* wird ein Ansatz zur Segmentierung, Klassifizierung und Generierung von Bewegungen vorgestellt. Kernbestandteile dieses Ansatzes sind ein gemeinsames probabilistischen Modell von Aktionsklassen, ein Algorithmus zur simultanen Segmentierung und Klassifizierung, eine Merkmalsrepräsentation für gerichtete Bewegungen und eine Methode zur Erzeugung generalisierter Bewegungen zwischen beliebigen Punkten. Dieser Ansatz wird in den nachfolgenden Kapiteln zum Lernen komplexer Aufgaben aus

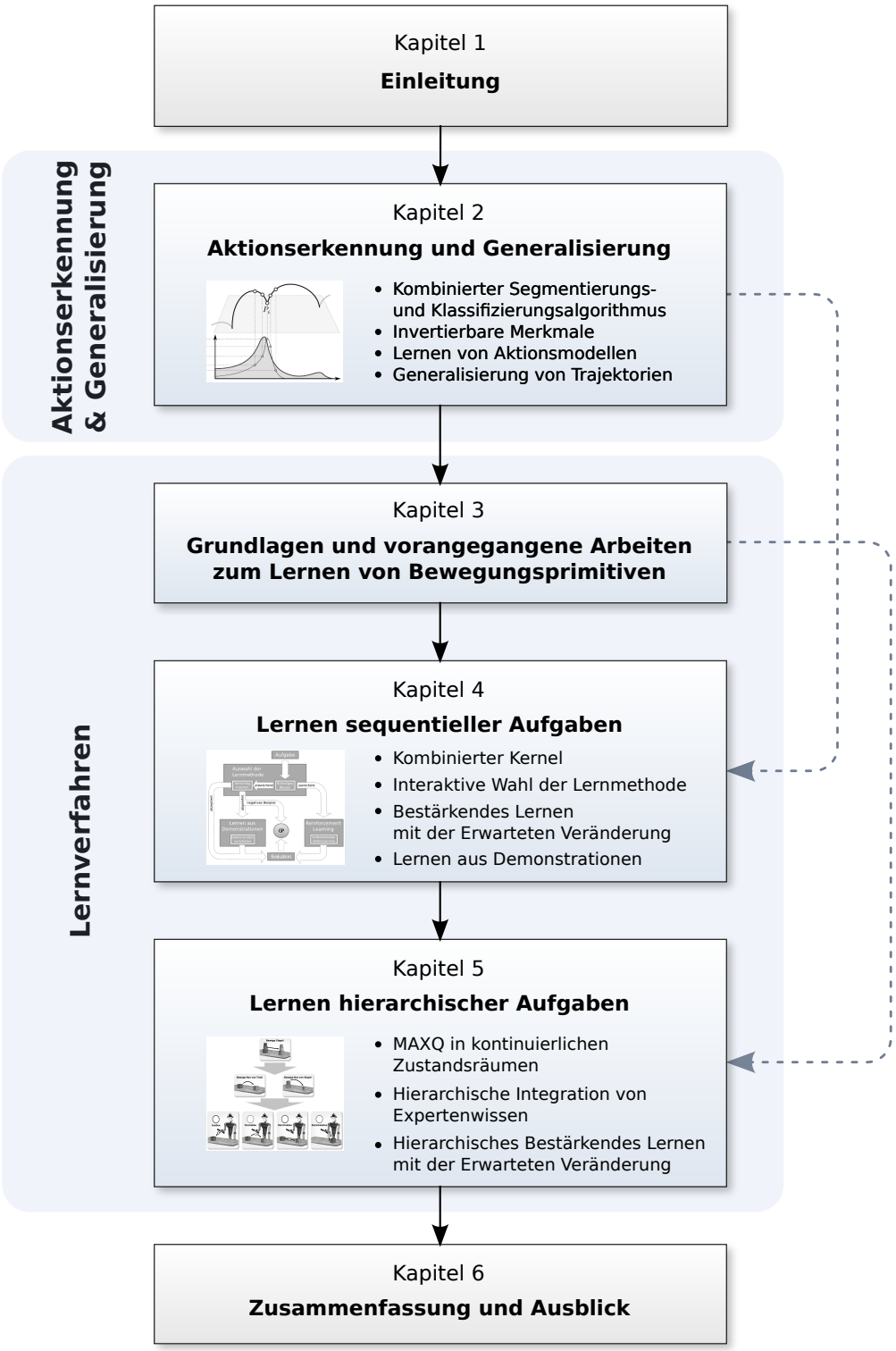


Abbildung 1.1: *Übersicht über die Zusammenhänge der Kapitel der Arbeit. Allgemein können die Kapitel getrennt voneinander gelesen werden. Die dargestellten Pfeile in der Übersicht deuten zusätzlich auf aufeinander aufbauende Arbeiten beziehungsweise Arbeiten, die die Verfahren der vorangegangenen Kapitel nutzen, hin.*

Demonstrationen eines Experten eingesetzt und ist Bestandteil der dort vorgestellten Algorithmen. Methodisch steht dieses Kapitel getrennt von den nachfolgenden Kapiteln.

Die *Kapitel 3*, *4* und *5* beschäftigen sich mit verschiedenen Ansätzen zum Lernen mit einer Kombination aus Bestärkendem Lernen und Lernen aus Demonstrationen. In *Kapitel 3* werden zunächst Vorarbeiten vorgestellt, die für das Lernen von Bewegungsprimitiven entwickelt wurden und die in den *Kapiteln 4* und *5* in einer ähnlichen Weise verwendet und erweitert werden. Die späteren Kapitel beschäftigen sich im Gegensatz zu Kapitel 3 mit dem Lernen von komplexen Aufgaben für praktische Anwendungen.

In *Kapitel 4* wird ein Lernverfahren zum Lösen sequentieller Aufgaben durch eine Kombination von Bestärkendem Lernen und Lernen aus Demonstrationen vorgestellt. In jeder Episode wird dabei durch ein mehrstufiges Entscheidungsmodul der am besten geeignete Lernansatz ausgewählt, um die Vorteile beider Verfahren zu kombinieren. Durch einen speziell entwickelten Kernel können mit dem hier entwickelten Lernansatz diskrete Aktionssequenzen variabler Länge in kontinuierlichen Zustandsräumen effizient gelernt werden.

In *Kapitel 5* wird auf Kapitel 4 aufbauend ein hierarchisches Lernverfahren entwickelt, um mit den Herausforderungen realer Aufgaben skalieren zu können. Dazu wird die MAXQ-Methode für hierarchisches Bestärkendes Lernen für kontinuierliche Zustandsräume erweitert. Um in diesen ein effizientes Lernen zu ermöglichen, setzt der hier vorgestellte Ansatz eine Bayessche Explorationsstrategie ein, die rekursiv anhand der MAXQ-Zerlegung bestimmt wird. Das so erweiterte Bestärkende Lernverfahren wird in dieser Arbeit mit dem Lernen aus Demonstrationen in einem hierarchischen System kombiniert.

Abschließend werden in *Kapitel 6* die Ansätze, Algorithmen und Beiträge zusammengefasst und eine Übersicht über potenzielle zukünftige Arbeiten gegeben. Die Zusammenhänge zwischen den einzelnen Kapiteln werden in Abbildung 1.1 dargestellt.

2 Kapitel Aktionserkennung und Generalisierung von Bewegungen

Die Fähigkeit, menschliche Aktionen zu erkennen und sie sich anzueignen, ist ein grundlegendes Problem in vielen Bereichen der Robotik. Hierzu zählen Anwendungen, in denen die Interaktion zwischen Robotern und Menschen eine Rolle spielt oder in denen Roboter anhand von demonstrierten Bewegungen lernen sollen. Um Menschen beispielsweise sinnvoll in ihrem Alltag unterstützen zu können, müssen Roboter komplexe Aufgaben beherrschen, die oftmals durch eine Kombination elementarer, einfacher Fähigkeiten gelöst werden können, wie es in den folgenden Kapiteln beschrieben wird. Diese Aktionen müssen für das Lernen aus demonstrierten Bewegungssequenzen segmentiert, erkannt und später von einem Roboter zum selbstständigen Lösen einer Aufgabe auch wieder generiert werden können.

Um diesen Herausforderungen zu begegnen, wird in diesem Kapitel ein neuer, integrierter Ansatz zur Segmentierung und Klassifizierung von Aktionen in menschlichen Bewegungssequenzen und zu deren Reproduktion und Generalisierung auf unbekannte Situationen vorgestellt. Für alle diese Aufgaben wird ein gemeinsames probabilistisches Modell eingesetzt. Dieses beschreibt Aktionsklassen durch eine Reihe von im Rahmen dieser Arbeit entwickelten invertierbaren Merkmalen für gerichtete Bewegungen. Die Merkmale erhalten die Charakteristik der Bewegungen und abstrahieren gleichzeitig von den verschiedenen Endpunkten dieser. Basierend auf dieser Repräsentation wird in dieser Arbeit ein kombinierter Segmentierungs- und Klassifizierungsalgorithmus vorgeschlagen, der die Eingabedaten nicht gierig verarbeitet, sondern eine inkrementelle Vorausschau über das aktuelle Segment hinaus nutzt. Somit wird bei der Bestimmung von Segmentgrenzen nicht nur die Aktion betrachtet, welche an der gesuchten Segmentgrenze endet, sondern auch die nachfolgende Aktion, die an diesem Punkt beginnt. Dadurch können verlässlich Übergänge zwischen einzelnen Aktionen bestimmt und systematische Fehler vermieden werden. In einem Szenario, das Programmieren oder Lernen durch Demonstration nutzt, ermöglichen die genutzten Aktionsmodelle, gelernte Bewegungen zu reproduzieren und zu vorher unbekannten Situationen zu generalisieren, wobei glatte Bewegungen generiert werden.

Um die Leistungsfähigkeit des hier vorgestellten Ansatzes zu evaluieren, werden typische Manipulationsaufgaben auf einer Tischoberfläche betrachtet. Aus Sequenzen von menschlichen Demonstrationen extrahiert der Ansatz erfolgreich die einzelnen Segmente und ordnet diese der passenden Aktionsklasse zu. Dabei zeigen die Experimente, dass der

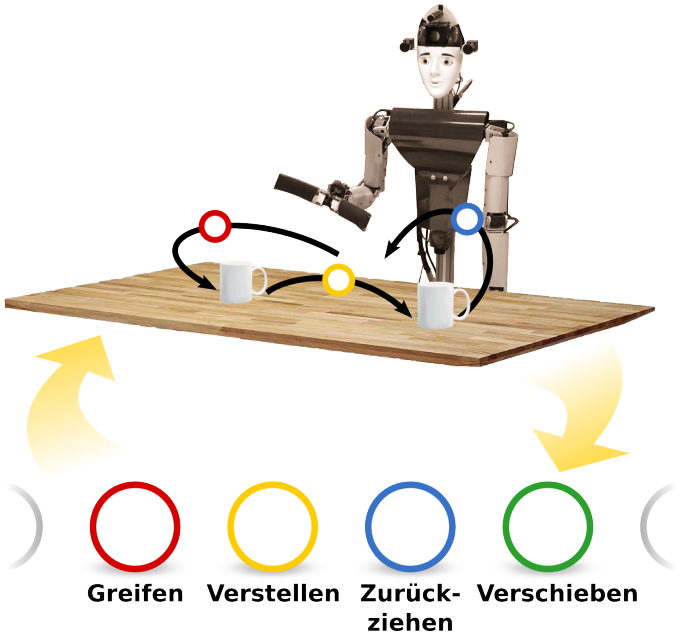


Abbildung 2.1: *Darstellung von typischen Aktionen in einem Manipulationsszenario, die mit dem hier vorgestellten Ansatz aus menschlichen Demonstrationen extrahiert werden.*

Ansatz von der inkrementellen Vorausschau bei der Bestimmung von Segmentgrenzen profitiert. Zusätzlich wird die Fähigkeit des Systems vorgestellt, gelernte Bewegungen zu reproduzieren und auf unbekannte Situationen zu generalisieren. Das folgende Kapitel basiert auf der eigenen Arbeit [2], die auf der IROS 2012 vorgestellt wurde.

2.1 Einleitung

Damit Roboter sinnvoll für Aufgaben des alltäglichen Lebens eingesetzt werden können, müssen sie in der Lage sein, eng mit Menschen zusammenzuarbeiten. Dies erfordert die Fähigkeit menschliche Aktionen wahrzunehmen und zu erkennen, um geeignet auf sie reagieren zu können. Einen weiteren Bereich, in dem diese Fähigkeit von zentraler Bedeutung ist, stellen Verfahren des maschinellen Lernens dar, die menschliche Demonstrationen als Wissensquelle nutzen. Sie sind ein vielversprechender Ansatz, um das Wissen von Experten einem künstlichen System effizient zugänglich zu machen und auf diese Weise den für praktische Aufgaben benötigten manuellen Programmieraufwand zu reduzieren.

Ebenso wie die direkte Interaktion mit Menschen, erfordert auch die Verarbeitung von Demonstrationen in einem Lernverfahren die Fähigkeit, die in der Sequenz enthaltenen Aktionen zu erkennen. Da solche Aktionen als Teilsequenzen in den verschiedensten Aufgaben wiederholt vorkommen, können Redundanzen erkannt und zur kompakten Repräsentation von Bewegungen genutzt werden. Dies erlaubt die Entwicklung effizienter Lernverfahren, denn anstatt für jede Aufgabe eine vollständige Trajektorie zu lernen, kann diese durch wenige Aktionen beschrieben werden. Die Lernaufgabe wird somit erheblich vereinfacht, da nur die Auswahl und die Reihenfolge der einzelnen Aktionen gelernt werden

müssen. Die einzelnen Aktionen lassen sich wiederum effizienter lernen als Trajektorien für die gesamte Aufgabe, da diese weniger komplex sind und Demonstrationen für alle Aufgaben, in denen sie vorkommen, gemeinsam genutzt werden können.

Wichtig für das Lernen aus Demonstrationen ist nicht nur das Erkennen der einzelnen Bewegungen, sondern auch, dass das System anschließend wieder Trajektorien generieren kann, um gelernte Aktionen anwenden zu können. Das System soll allerdings nicht nur die ursprünglichen Bewegungen reproduzieren können, sondern diese auch generalisieren, d.h. gelernte Bewegungen auf unterschiedliche Situationen übertragen können. Solch eine Anforderung setzt eine Repräsentation der Trajektorien voraus, die auf der einen Seite auf verschiedene Situationen flexibel anwendbar ist, auf der anderen Seite aber auch spezifisch genug ist, um die charakteristischen Merkmale der Bewegungen zu erhalten.

In diesem Kapitel wird das Problem der automatischen Erkennung und Extraktion einzelner Aktionen aus einer Sequenz menschlicher Bewegungen sowie der Erzeugung generalisierter Trajektorien betrachtet. Dazu wird eine gemeinsame Aktionsrepräsentation entwickelt, die sowohl die Erkennung und Klassifizierung als auch die Generierung von Aktionen erlaubt.

Ein zentrales Ziel des hier vorgestellten Ansatzes ist die Extraktion sinnvoller Aktionen, die in sich abgeschlossen sind und einen eindeutig definierten Effekt haben, so dass sie als Grundelemente in der Programmierung durch Demonstration oder anderen interaktiven Anwendungen verwendet werden können (siehe Abbildung 2.1). Dies ist notwendig, um generierte Aktionen einzeln oder zusammengesetzt zum Lösen komplexer Aufgaben wiederverwenden zu können. Zusätzlich bietet dieser Grad der Strukturierung einen guten Kompromiss zwischen Flexibilität im Bezug auf verschiedene zu lösende Aufgaben und Dateneffizienz. Eine große Herausforderung besteht dabei allerdings darin, die intuitive Vorstellung von sinnvollen Aktionen für einen Lernalgorithmus zu formalisieren. In diesem Ansatz werden daher Modelle bekannter Aktionsklassen als Hintergrundinformation benutzt. Des Weiteren gehen Aktionen in der Regel nahtlos in benachbarte Bewegungen über, so dass es schwierig ist, einzelne Grundelemente exakt zu begrenzen. Häufig verwendete Merkmale zur Abgrenzung von Bewegungssegmenten, wie beispielsweise Ruhephasen in der Bewegung, sind unter realen Bedingungen in der Regel nicht eindeutig erkennbar.

Eine weitere Schwierigkeit bei der Erkennung von Aktionen ist, dass Aktionen typischerweise in speziellen Kontexten ausgeführt werden. Zum Beispiel weisen einige Aktionsklassen sehr ähnliche Trajektorien auf, obwohl sie auf semantischer Ebene sehr unterschiedlich sind. Diese Aktionsklassen unterscheiden sich hauptsächlich durch den Kontext, in dem sie ausgeführt werden. Diese Art von relevanten Informationen sind oftmals für den Roboter nicht vorhanden. Werden beispielsweise nur die Bewegungen der Finger beim Spielen eines Klaviers und beim Tippen auf einer Computertastatur betrachtet, so lassen sich diese schwer unterscheiden, wenn keinerlei Kontextinformationen über die verwendeten Objekte vorliegen. So bieten die Daten, die zum Beispiel durch Motion Capture-Systeme oder kinestatisches Training aufgenommen werden können, in der Regel nur rein geometrische Informationen, die alleine oftmals die Essenz der Handlung schwer erkennen lassen.

Offensichtlich sind die Segmentierung und die Klassifizierung von Bewegungsdaten zwei eng verwandte Probleme, deren Lösungen voneinander abhängen. Ein ungenau

segmentiertes Bewegungsprimitiv erschwert die richtige Zuordnung zu einer Bewegungsklasse. Umgekehrt können Segmentgrenzen nur verlässlich bestimmt werden, wenn das betrachtete Teilsegment der richtigen Aktionsklasse zugeordnet werden kann.

In dieser Arbeit wird ein Algorithmus vorgeschlagen, der beide Aufgaben gemeinsam löst. Die hier vorgestellte Methode beruht auf Hidden Markov Modellen (HMM) zur Kodierung von Aktionsklassen. Dies erlaubt es, in unbekannten Bewegungssequenzen iterativ sinnvolle Segmente im Bezug auf zuvor gelernte Einzelaktionen mittels eines Maximum Likelihood-Kriteriums und einer Vorausschau auf das nachfolgende Segment zu bestimmen. Die Aktionen werden durch aufgabenspezifische invertierbare Merkmale für gerichtete Bewegungen repräsentiert, die es dem System erlauben, das Aktionsmodell generativ zu nutzen, um gelernte Bewegungen zu reproduzieren und diese zusätzlich auf neue Situationen zu generalisieren.

2.1.1 Verwandte Arbeiten

Die grundlegende Bedeutung von Verfahren zur Segmentierung, Klassifizierung und Generierung von Aktionen für die Robotik wird deutlich, wenn die vielen Arbeiten betrachtet werden, die diesen Aufgaben und ihren verschiedenen Aspekten in den letzten Jahren gewidmet wurden. Obwohl die drei Probleme, wie zuvor erläutert, stark zusammenhängen, ist die Zahl der Ansätze, die diese gemeinsam in einer integrierten Art und Weise zu lösen versuchen, vergleichsweise gering. Hidden Markov Modelle [3] stellen in diesem Zusammenhang einen soliden probabilistischen Ansatz zur Modellierung von Aktionen dar. Entsprechend sind sie ein beliebtes Werkzeug in diesem Bereich, das bereits in einigen Ansätzen erfolgreich zur Segmentierung und Klassifizierung von menschlichen Bewegungen angewandt wurde. Bashir et al. [4] nutzen zum Beispiel HMM zur Erkennung von Handgesten aus der australischen Gebärdensprache. Die Trajektorien werden in diesem Ansatz anhand ihrer Krümmung in eine Reihe von Teiltrajektorien unterteilt. Mittels Hauptkomponentenanalyse (PCA) [5] werden diese Teiltrajektorien zu Merkmalsvektoren mit einer festgelegten Größe reduziert, welche anschließend gruppiert und in HMM kodiert werden. Die Klassifizierung neuer Daten erfolgt, indem für jede Gruppe die Wahrscheinlichkeit berechnet wird, mit der das zugehörige HMM die zu klassifizierenden Daten erzeugt haben könnte, und anschließend die Gruppe ausgewählt wird, deren HMM diese Beobachtungswahrscheinlichkeit maximiert.

Verschiedene Autoren haben geeignete Merkmale für die HMM-basierte Klassifikation untersucht. Al Mansur et al. [6] betrachten beispielsweise dynamische Merkmale zur Charakterisierung von Bewegungen. Dazu berechnen sie anhand von 3D-Motion Capture-Daten und eines Modells des menschlichen Körpers die Drehmomente der einzelnen Gelenke. Von diesen verwenden sie die Daten zweier Kniegelenke und zweier Hüftgelenke, um Hidden Markov Modelle für sieben aufgenommene Ganzkörperbewegungen zu trainieren. Diese nutzen sie anschließend zur Klassifizierung neuer Bewegungen. De Schutter et al. [7] entwickelten aus der Darstellung von Starrkörperbewegungen als Schraubenbewegungen eine koordinatenfreie Repräsentation mit sechs Freiheitsgraden. Diese ist invariant gegenüber dem Referenzsystem des Beobachters, der bei der Aufnahme der Bewegung beobachteten Position auf einem Objekt, dem Geschwindigkeitsverlauf der Bewegung sowie gegenüber einer Skalierung der Bewegung in Raum oder Zeit. Diese

Eigenschaft nutzen die Autoren, um Hidden Markov Modelle für Aktionen aus Motion Capture-Aufnahmen der entsprechenden Bewegungen zu lernen und gute Ergebnisse bei der Klassifikation zu erreichen. Dies wird an Manipulationsaktionen gezeigt, die anhand der Bewegung des manipulierten Objekts klassifiziert werden.

Als Alternative zur Identifikation von Aktionsgrenzen durch lokale Extrema von Merkmalen, verfolgen einige Forscher probabilistische Ansätze, in denen sie Hidden Markov Modelle nicht nur zur Klassifikation, sondern auch zur Segmentierung von Bewegungen einsetzen. Elmezain et al. [8] schlagen ein System zur Extraktion von Gesten aus einer Folge von Bewegungsdaten vor. Als Merkmale verwenden sie die Orientierung der Bewegung in der Ebene. Auf Basis der Beobachtungswahrscheinlichkeiten der Aktionsklassen wird der Start- beziehungsweise Endpunkt einer Bewegung bestimmt. Dabei wird der Unterschied der Likelihood zwischen der wahrscheinlichsten Aktionsklasse, gegeben die Daten, und einer Restklasse [9] berechnet. Ein Vorzeichenwechsel dieses Maßes wird als Segmentgrenze aufgefasst. Die verschiedenen Bewegungsklassen sowie die Restklasse werden dabei durch einzelne HMM repräsentiert. Mit diesem Ansatz können die Autoren Armbewegungen erkennen, die arabische Zahlen darstellen. Kohlmorgen und Lemm [10] entwickelten einen Online-Algorithmus zur unüberwachten Segmentierung und Klassifizierung von Bewegungsdaten mit HMM. Die Zustände des HMM repräsentieren dabei die Wahrscheinlichkeitsverteilung der Daten über die Zeit. Diese wird innerhalb eines Fensters aus den Eingabedaten geschätzt. Somit lässt sich durch den Viterbi-Algorithmus die wahrscheinlichste Folge von Verteilungen, gegeben die Daten, berechnen. Dabei wird die Anzahl der Zustandsübergänge durch eine entsprechende Wahl der Übergangswahrscheinlichkeiten begrenzt. Die Segmentierung der Eingabedaten ergibt sich in diesem Ansatz implizit durch die ermittelten Zustandsübergänge, die Veränderungen in der Verteilung der Daten markieren.

Um die Komplexität der Segmentierung zu reduzieren, verwenden Li et al. [11] einen zweistufigen Ansatz, inspiriert durch Ansätze aus der Spracherkennung. Dabei wird auf der unteren Ebene für jedes relevante Objekt eine Menge von Bewegungsprimitiven verwaltet, die durch HMM beschrieben werden. Die Autoren verwenden dabei zum Training der Modelle relative Merkmale, welche sich auf das jeweilige Objekt beziehen. Auf der oberen Ebene werden komplexere Aufgaben durch aufgabenspezifische Markov-Ketten modelliert, die Übergangswahrscheinlichkeiten zwischen den Klassen der unteren Ebene beinhalten. Anhand der auf der oberen Ebene erkannten aktiven Aufgaben kann die Anzahl relevanter Objekte reduziert werden, wodurch die Segmentierung erleichtert wird. Sugiura et al. [12] beschäftigen sich mit dem Lernen von Manipulationsbewegungen, bei denen das manipulierte Objekt in Bezug zu einem anderen Objekt bewegt werden muss. Dabei schlagen sie ein System vor, das während des Trainings anhand von Trajektoriendaten sowohl das Bezugsobjekt als auch das intrinsische Koordinatensystem einer Bewegung aus einer vorgegebenen Liste möglicher Bezugssysteme anhand eines Maximum Likelihood-Kriteriums schätzt. Bewegungen werden dabei durch eine spezielle Form von HMM, die das Bezugsobjekt berücksichtigt, anhand statischer und dynamischer Merkmale kodiert. Diese Modelle ermöglichen die Klassifikation und Reproduktion von vorsegmentierten Trajektorien.

Billard et al. [13] beschäftigen sich mit dem Problem des Erzeugens von Bewegungen, die die Essenz der gelernten Aufgabe wiedergeben. Sie nutzen dazu die Variabilität

der Merkmale im Gelenk- und im kartesischen Raum über mehrere Demonstrationen einer Aufgabe, um ihre Relevanz für die Bewegungsreproduktion durch eine Kostenfunktion zu beurteilen. Hierfür werden zunächst in einem Vorverarbeitungsschritt die Wendepunkte von demonstrierten Trajektorien als Merkmale extrahiert und in einem HMM pro Dimension kodiert, um eine probabilistische Beschreibung der Aufgabe zu erhalten. Dieses Modell kann anschließend verwendet werden, um weitere Demonstrationen zu klassifizieren oder um eine repräsentative Trajektorie aus den gesammelten Demonstrationen zu erzeugen. Um Trajektorien für einen Roboter zu generieren, die die demonstrierte Lösung reproduzieren, muss die für die Aufgabe ermittelte Kostenfunktion unter Berücksichtigung der Kinematik des Roboters optimiert werden. In einer jüngeren Arbeit nutzen Calinon et al. [14] HMM, um robuste Modelle von Bewegungen aus Demonstrationen zu lernen. Die HMM kodieren dabei die Trajektorien, die durch eine Folge von Positionen und Geschwindigkeiten im Raum gegeben sind. Diese stellen in ihrem Ansatz die Grundlage zur flexiblen Reproduktion von Bewegungen mit Hilfe der sogenannten Gaussian Mixture-Regression (GMR) [15] und einem dynamischen System zur Generierung von Trajektorien dar. Der auf HMM und GMR basierende Ansatz extrahiert die Redundanzen über verschiedene Demonstrationen und generiert aus diesen Informationen ein zeitunabhängiges Modell der demonstrierten Bewegungen. In ihren Experimenten können die Autoren zeigen, dass auch während generalisierter Bewegungen verschiedene Randbedingungen durch die Kombination von mehreren, für verschiedene Referenzsysteme gelernten HMM beachtet werden können. Im Gegensatz zu den oben diskutierten HMM-basierten Ansätzen ist das Ziel der in diesem Kapitel vorgestellten Ansätze, ein einheitliches System zu entwerfen, das sowohl für die Klassifizierung und Segmentierung als auch für die Erzeugung von neuen Bewegungen geeignet ist.

Mit einem ähnlichen Anliegen haben kürzlich Kulić et al. [16] eine Integration verschiedener Ansätze vorgeschlagen, um inkrementell Bewegungen anhand von Demonstrationen eines Experten zu erlernen. Die Trajektorien werden online mit einer verbesserten Variante des Algorithmus von Kohlmorgan und Lemm [10] segmentiert und zu abstrakten Bewegungsprimitiven gruppiert. Die Kodierung der so ermittelten Primitive durch HMM erlaubt es, neue Bewegungen einer bekannten Klasse zuzuordnen und zu einer bekannten Bewegungsklasse passende Trajektorien zu generieren. Sequenzen aus primitiven Bewegungen können durch einen Graphen, der aus beobachteten Übergangswahrscheinlichkeiten gelernt wird, erzeugt werden. Während Kulić et al. effizient mehrere Modelle kombinieren, ist das Ziel der vorliegenden Arbeit, ein gemeinsames Modell zu entwickeln, auf dessen Grundlage alle drei Aufgaben gelöst werden können.

Ein aktueller Ansatz, der nicht auf HMM für die Segmentierung und Klassifizierung von Bewegungsdaten setzt, wird von Meier et al. [17] vorgeschlagen. Diese reduzieren mit Hilfe ihres Ansatzes das Segmentierungsproblem von Bewegungssequenzen auf das Problem der sequentiellen Erkennung von Bewegungen in einem Online-Verfahren. Dazu trainieren die Autoren sogenannte Dynamic Movement Primitives (DMP) [18], die die Generalisierung von Aktionen auf beliebige Positionen erlauben. Durch eine Umformulierung des ursprünglichen DMP-Formalismus in ein lineares dynamisches System mit zusätzlichen Kontrolleingaben erhalten sie eine Schätzung der Zielposition und der Dauer einer teilweise beobachteten Trajektorie. Basierend auf diesen Schätzungen wird ein Algorithmus zur Online-Segmentierung von Trajektorien entwickelt, der Schnittpunkte

durch inkrementelle Erweiterung des aktuellen Segments in Kombination mit einem Maximum Likelihood-Kriterium bestimmt.

Im Gegensatz zu diesen Ansätzen basiert die hier vorgestellte Arbeit auf zielgerichteten, invertierbaren Merkmalen, die die Charakteristik der Aktionsklassen erfassen. Des Weiteren wird ein Ansatz verwendet, der die Daten für das aktuelle Segment nicht rein sequentiell verarbeitet. Stattdessen wird sowohl das an einer Segmentgrenze endende Segment als auch das darauffolgende, dort beginnende Segment betrachtet, um die Segmentierungsergebnisse zu verbessern.

2.1.2 Wissenschaftlicher Beitrag

Anstatt die eng verwandten Aufgaben der Segmentierung von Aktionen aus Bewegungssequenzen, der überwachten Klassifizierung von Aktionen und der Erzeugung generalisierter Trajektorien für gelernte Aktionsklassen durch unterschiedliche Verfahren getrennt voneinander zu lösen, ist der zentrale Beitrag der in diesem Kapitel vorgestellten Arbeit die Entwicklung eines homogenen Systems, das das Lösen dieser Aufgaben auf Basis eines gemeinsamen probabilistischen Modells der Bewegungsklassen ermöglicht. Um die Darstellung von Bewegungsklassen durch HMM zu ermöglichen, wird in diesem Kapitel eine Merkmalsrepräsentation für gerichtete Bewegungen entwickelt. Diese führt zu einer geringen Intra-Klassen-Varianz, während sie gleichzeitig die charakteristische Form einer Bewegungsklasse erhält. Zur Segmentierung und Klassifizierung der auf diese Weise beschriebenen Bewegungen wird ein kombinierter Algorithmus entwickelt, der systematische Fehler in der Wahl der Segmentgrenzen während der sequentiellen Verarbeitung der Eingabedaten vermeidet. Dazu wird sowohl die Wahrscheinlichkeit, dass die vorherige Aktion an einer möglichen Segmentgrenze endet, als auch die Wahrscheinlichkeit, dass der darauffolgende Bereich den Anfang einer neuen Bewegung darstellt, berücksichtigt. Dieser Ansatz vermeidet die typischen Probleme von sogenannten gierigen Verfahren, welche bei der Segmentierung nur das Ende eines Segments berücksichtigen. Eine weitere wichtige Eigenschaft der hier vorgestellten Merkmalsrepräsentation von Bewegungen ist ihre Umkehrbarkeit. Diese erlaubt es, aus einer Folge von Merkmalen eine Trajektorie zwischen beliebigen Punkten zu berechnen, die durch einen Roboter ausgeführt werden kann. Dazu generiert der hier vorgestellte Ansatz eine Folge von Stützvektoren im Merkmalsraum aus dem HMM einer Aktionsklasse. Diese werden mittels Gauß-Prozessen approximiert, so dass die generierten Bewegungen auf der einen Seite die Charakteristika der Bewegungsklasse wiedergeben und auf der anderen Seite einen glatten Verlauf aufweisen. Der vorgestellte Ansatz verfügt über keine freien Parameter und kann somit leicht auf ein breites Spektrum von gerichteten Bewegungsklassen angewendet werden.

Das Kapitel ist wie folgt aufgebaut: In Abschnitt 2.2 wird der integrierte Ansatz zur Segmentierung und Klassifizierung von Bewegungssequenzen und zur Generierung neuer Bewegungen präsentiert. Dabei werden (i) die in dieser Arbeit entwickelte Merkmalsrepräsentation für gerichtete Bewegungen, (ii) das zur Beschreibung von Aktionsklassen genutzte grafische Modell und wie dieses gelernt wird, (iii) der kombinierte Segmentierungs- und Klassifikationsalgorithmus und (iv) die Generierung neuer Bewegungen vorgestellt. In Abschnitt 2.3 wird dieser Ansatz an typischen Manipulationsbewegungen evaluiert. Dazu werden zunächst die Datenaufnahme und -vorverarbeitung sowie der Versuchsaufbau

beschrieben. Anschließend wird die Evaluation sowohl im Bezug auf den Segmentierungs- und Klassifizierungsalgorithmus, als auch auf die Generalisierungsfähigkeit gelernter Bewegungen beschrieben. Letztlich wird in Abschnitt 2.4 der vorgestellte Ansatz zusammengefasst und die entstandenen Ergebnisse werden diskutiert.

2.2 Ansatz zur Segmentierung, Klassifizierung und Generalisierung von Aktionen

In diesem Abschnitt wird der integrierte Ansatz zur Segmentierung, Klassifizierung, Reproduktion und Generalisierung von Aktionen im Detail vorgestellt. Insbesondere wird dabei auf den hier entwickelten, kombinierten Segmentierungs- und Klassifizierungsalgorithmus zur Erkennung von Aktionen in Bewegungssequenzen und zur Zuordnung von Aktionen zu bekannten Bewegungsklassen eingegangen. Ein besonderes Merkmal dieses Algorithmus ist, dass er beide Segmente, die die gesuchte Segmentgrenze berühren, in einem Maximum Likelihood-Kriterium berücksichtigt. Anstatt nur sequenziell in den Eingabedaten nach dem Ende der gerade beobachteten Aktion zu suchen, nutzt der Algorithmus zusätzlich Informationen über mögliche Startpunkte der nachfolgenden Aktion zur Bestimmung von Segmentgrenzen aus, um die Genauigkeit der Segmentierung zu verbessern. Die Berechnung dieses Kriteriums beruht auf einem gemeinsamen probabilistischen Aktionsmodell, das sowohl zur Segmentierung und Erkennung, als auch zur Generierung von Bewegungen einsetzen wird. Dies ist ein wichtiger Vorteil dieses Ansatzes, da diese drei Aufgaben besonders in praktischen Anwendungen eng miteinander verknüpft sind und daher häufig gemeinsam gelöst werden müssen. So ist eine Segmentierung oftmals nicht ohne eine verlässliche Klassifikation und anders herum die Klassifikation nicht ohne eine gute Segmentierung möglich. Zur probabilistischen Modellierung von Aktionsklassen werden Hidden Markov Modelle eingesetzt, in denen diese unter Berücksichtigung ihrer räumlichen und zeitlichen Charakteristika kodiert werden. Die Leistungsfähigkeit dieses Modells und die Fähigkeit, generalisierte Trajektorien anhand der gelernten Aktionsklassen zu generieren, hängt dabei von der Wahl einer geeigneten Merkmalsrepräsentation der Eingabedaten ab.

2.2.1 Invertierbare Merkmale zur Erkennung gerichteter Bewegungen

Um eine exakte Segmentierung und eine zuverlässige Klassifizierung von Aktionen zu ermöglichen, müssen Merkmale zur Bewegungsrepräsentation eine Reihe von Kriterien erfüllen. Aus der Modellierungsperspektive müssen sie Ähnlichkeiten von Aktionsklassen wiedergeben, was bedeutet, dass die Varianz der Merkmale, die zu Aktionen der gleichen Kategorie gehören, gering sein sollte. Gleichzeitig sollten die Merkmale verschiedener Klassen möglichst unterschiedlich sein, um eine robuste Kategorisierung zu erlauben. Aus der Anwendungsperspektive sollte die Aktionsrepräsentation genügend Flexibilität aufweisen, um möglichst viele verschiedene Aktionsklassen beschreiben zu können. Zudem sollten die Merkmale möglichst einfach und schnell berechenbar sein.

In diesem Kapitel werden gerichtete Aktionen betrachtet, die durch eine Folge von Positionen des Endeffektors beschrieben werden. Die Darstellung dieser Positionen im kartesischen Raum erfüllt jedoch nicht die aufgeführten Anforderungen. Bereits eine kleine

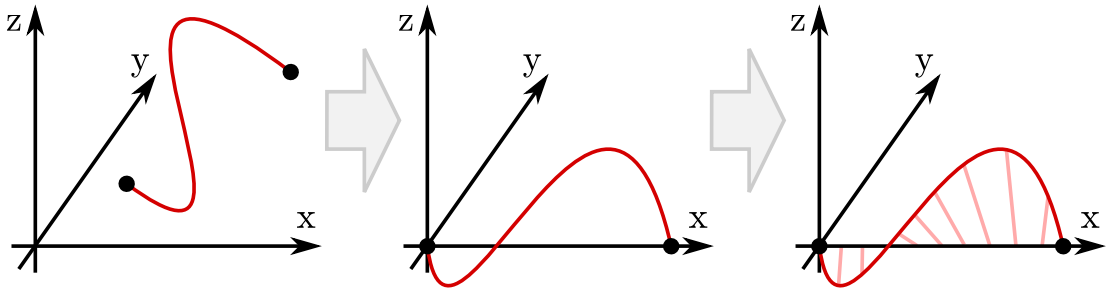


Abbildung 2.2: *Illustration des Merkmalsextraktionsschemas. Links: Originaltrajektorie. Mitte: Trajektorie nach der linearen Transformation. Der Startpunkt der Trajektorie liegt im Koordinatenursprung und der Endpunkt auf der x-Achse. Rechts: Finale Merkmale nach der Distanztransformation. Die Abstände der Trajektorienpunkte zu den ermittelten Stützstellen sind hellrot eingezeichnet.*

Änderung in der Ausführung einer Aktion oder in der Situation, in der sie ausgeführt wird, kann zu großen Änderungen im Koordinatenraum führen. Dies vergrößert die Varianz innerhalb einer Klasse und kann dazu führen, dass sich Bewegungen unterschiedlicher Klassen, für die die gleichen Start- und Zielsituationen gewählt werden, mehr ähneln, als Bewegungen derselben Klasse mit unterschiedlichen Endpunkten. Dadurch wird die Separierbarkeit zwischen den einzelnen Klassen reduziert und das Klassifizierungsproblem erschwert. In dieser Arbeit transformiert der Algorithmus daher die kartesischen Koordinaten in einem Vorverarbeitungsschritt in eine Merkmalsrepräsentation, bevor diese zum Training probabilistischer Modelle von Aktionsklassen genutzt werden. Diese Transformation erfolgt in drei Schritten:

Zentrierung und Rotation auf die x-Achse: Um die Trajektorie zu zentrieren und ihre Ausrichtung zu normalisieren, wird eine lineare Transformation auf die Eingabetrajektorien angewandt, so dass die Bewegungen alle im Ursprung starten und an einem fest definierten Punkt auf der x-Achse enden.

Wahl äquidistanter Stützstellen: Im zweiten Schritt betrachtet der Algorithmus die Strecke zwischen dem Ursprung und dem transformierten Endpunkt der Trajektorie auf der x-Achse und wählt entlang dieser eine Reihe äquidistanter Stützstellen. Die Anzahl der Stützstellen wird durch die Anzahl der Punkte in der ursprünglichen Trajektorie bestimmt.

Distanztransformation: Schließlich ermittelt der Algorithmus die Position jedes Punktes der transformierten Trajektorie relativ zu der entsprechenden Stützstelle. Diese Abstände stellen die Merkmalsvektoren dar.

Dieser Prozess ist in Abbildung 2.2 veranschaulicht. Das Ergebnis sind Merkmale, die die Charakteristiken der originalen Bewegungen bewahren und gleichzeitig den Vorteil haben, invariant gegenüber räumlicher Rotation, Translation und verschiedenen Start-

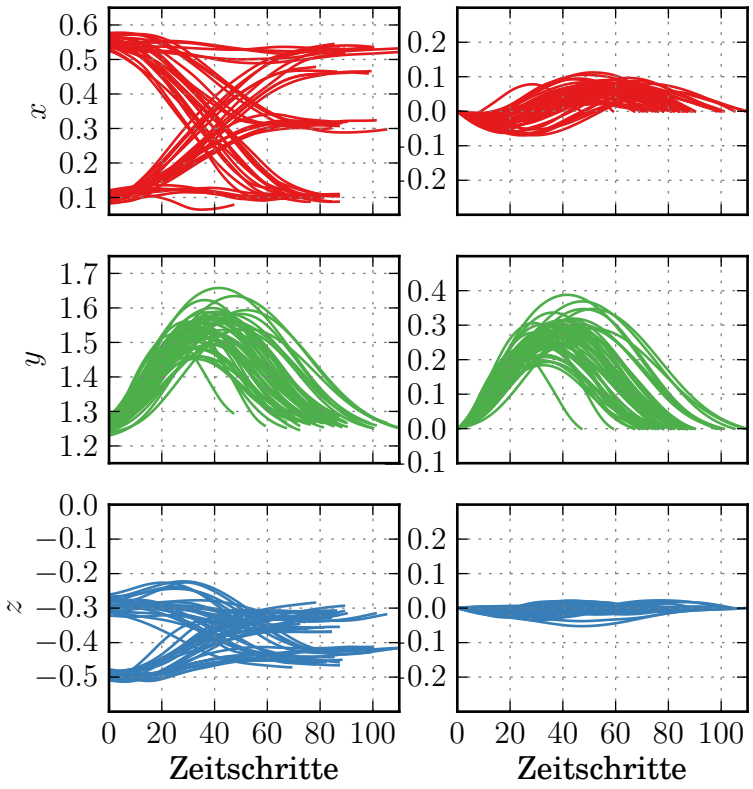


Abbildung 2.3: *Illustration des Effektes der hier vorgestellten Transformationen. Dazu wurden 50 Verstellbewegungen auf einem Tisch mittels eines Motion Capture-Systems aufgenommen. Dabei wurden jeweils fünf Bewegungen von zwei festgelegten Punkten zu fünf anderen Punkten innerhalb des natürlichen Arbeitsbereiches eines Menschen ausgeführt. Links sind die originalen Trajektoriendaten der aufgenommen Bewegungen zerlegt in die einzelnen kartesischen Komponenten dargestellt. Rechts sind die entsprechenden transformierten Merkmale zu sehen.*

oder Zielpunkten zu sein. Illustriert wird dies an einem Beispiel in Abbildung 2.3. Die Ähnlichkeit der Trajektorien innerhalb der Klasse erhöht sich, wenn die Trajektorien in die hier vorgeschlagene Merkmalsrepräsentation umwandelt werden. Als Ergänzung zu den transformierten kartesischen Koordinaten werden Distanzen zu relevanten Objekten zu dem hier gewählten Merkmalsvektor hinzugefügt, um aufgabenspezifische Informationen bei der Segmentierung und Klassifizierung von Aktionen berücksichtigen zu können. Dies ist durch die Tatsache motiviert, dass Aktionen abhängig von der Situation sehr unterschiedliche Bedeutungen haben können, auch wenn sich ihre Trajektorien im kartesischen Raum stark ähneln. Um diese Bewegungen korrekt zu erkennen, ist es daher oftmals hilfreich, Informationen über Relationen zu Objekten, die von der Aktion betroffen sind, zu berücksichtigen.

Ein weiterer Vorteil der hier vorgestellten Merkmalswahl ist, dass die Merkmale unabhängig von konkreten Endpunkten der Bewegung sind und ihre Berechnung leicht

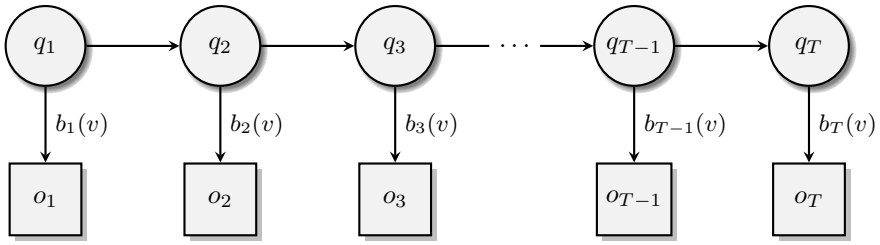


Abbildung 2.4: *Darstellung der Zufallsprozesse aus denen sich ein HMM zusammensetzt. Die Kreise repräsentieren die verdeckten Zustände q_t eines Markov-Prozesses. In Abhängigkeit von diesen Werten werden durch einen zweiten Zufallsprozess zu jedem Zeitpunkt t Beobachtungen o_t generiert. Die Pfeile deuten an, dass jeder Zustand q_t aufgrund der Markov-Eigenschaft, gegeben den vorangegangenen Zustand q_{t-1}, bedingt unabhängig von allen anderen Zuständen ist. Beobachtungen o_t sind in diesem Modell ebenfalls bedingt unabhängig von der Historie des Systems, wenn der Zustand q_t bekannt ist.*

invertierbar ist, so dass Bewegungen zwischen beliebigen Start- und Zielpunkten aus einer gegebenen Folge von Merkmalsvektoren erzeugt werden können.

2.2.2 Stochastisches Aktionsmodell

Um die gelernten Aktionsklassen kompakt für die Segmentierung und Klassifizierung von Bewegungssequenzen darzustellen, werden *Hidden Markov Modelle (HMM)* [3] auf der in Abschnitt 2.2.1 beschriebenen Merkmalsrepräsentation trainiert. Bei HMM handelt es sich um ein generatives stochastisches Modell, das eine Beobachtungsfolge durch zwei gekoppelte Zufallsprozesse beschreibt, wie in Abbildung 2.4 dargestellt. Der erste Prozess ist ein Markov-Prozess, dessen Zustand q_t zum Zeitpunkt $t \in [1, \ldots, T]$ nicht direkt beobachtbar ist. Dieser steuert einen zweiten Zufallsprozess, der gemäß zustandsabhängiger Wahrscheinlichkeitsverteilungen zu jedem Zeitpunkt t eine Beobachtung o_t generiert. HMM definieren somit Wahrscheinlichkeitsverteilungen über Beobachtungssequenzen, die die Segmentierung und Klassifizierung von Bewegungssequenzen mit einem probabilistischen Ansatz ermöglichen. Sie sind ein beliebtes Werkzeug zur Analyse von Zeitreihen, da sie eine flexible und transparente Modellierung auf Grundlage der Wahrscheinlichkeitstheorie ermöglichen. Darüber hinaus bieten sie die Möglichkeit, Eingabesequenzen variabler Länge zu modellieren und zu verarbeiten. Bekannt geworden sind HMM durch Anwendungen in der Spracherkennung [3]. Durch ihren großen Erfolg auf diesem Gebiet wurde ihre Nutzung auf viele weitere Bereiche der Mustererkennung, der Bioinformatik und der Zeitreihenanalyse ausgeweitet. Beispiele hierfür finden sich in der automatischen Handschrifterkennung [19], der Analyse von DNA-Sequenzen in der Biologie [20] oder der Erkennung von Kreditkartenbetrug [21].

Formal werden HMM durch eine Menge von Zuständen $\mathcal{S} = \{s_1, \ldots, s_n\}$, ein Alphabet $\mathcal{V}$ der möglichen Beobachtungen und ein Tripel $\lambda = (A, B, \pi)$ definiert. Die Bestandteile des Tripels λ sind dabei:

- Die Matrix $A = \{a_{i,j}\} \in [0, 1]^{n \times n}$ beschreibt die Zustandsübergangswahrschein-

lichkeiten $a_{i,j} = P(q_t = s_j \mid q_{t-1} = s_i)$ des Systemzustands q vom Zustand s_i zum Zeitpunkt $t-1$ in den Zustand s_j zum Zeitpunkt t. Aufgrund der Markov-Eigenschaft des verdeckten Zufallsprozesses hängt die Wahrscheinlichkeitsverteilung eines Zustandes nur von dessen unmittelbarem Vorgänger ab und ist von der übrigen Zustandshistorie bedingt unabhängig:

$$P(q_t \mid q_{t-1}, \ldots, q_1) = P(q_t \mid q_{t-1})$$

- Die Beobachtungsmodelle $B = \{b_i\}$ ordnen den Symbolen $v \in \mathcal{V}$ die Wahrscheinlichkeit $b_i(v) = P(o_t = v \mid q_t = s_i)$ zu, vom Modell zum Zeitpunkt t als Beobachtung o_t emittiert zu werden, wenn sich das System zu diesem Zeitpunkt im Zustand s_i befindet. Gegeben den aktuellen Systemzustand ist die aktuelle Beobachtung bedingt unabhängig von allen vorherigen Systemzuständen und Beobachtungen:

  $$P(o_t \mid q_t, q_{t-1}, \ldots, q_1, o_{t-1}, \ldots, o_1) = P(o_t \mid q_t)$$

  Jeder Zustand besitzt ein eigenes Beobachtungsmodell, welches von denen der anderen Zustände unabhängig ist. Für kontinuierliche Beobachtungen werden häufig Modelle mit normalverteilten Beobachtungswahrscheinlichkeiten verwendet.

- Der Vektor $\pi = \{\pi_i\}, \pi_i \in [0,1]^n$ enthält für alle Zustände die Wahrscheinlichkeiten $\pi_i = P(q_1 = s_i)$, dass es sich bei dem Zustand s_i um den Startzustand q_1 handelt.

Die Matrix A der Zustandsübergangswahrscheinlichkeiten, die Beobachtungsmodelle B und die Verteilung π des Startzustands sind dabei freie Parameter des Modells. Sie können mit Hilfe des auf Expectation Maximization [22] basierenden Baum-Welch-Algorithmus so bestimmt werden, dass die Emissionswahrscheinlichkeit $P(O \mid \lambda)$ einer gegebenen Beobachtungsfolge $O = (o_1, \ldots, o_T)$ lokal maximiert wird. Zur Verwendung von HMM zur Repräsentation von Daten ist es also nicht notwendig, die Struktur der Daten vorab zu kennen und die verdeckten Zustände des HMM entsprechend zu wählen.

Die anhand von Beispielbewegungen mit dem Baum-Welch-Algorithmus gelernten Modelle für Aktionsklassen lassen sich nutzen, um neue Bewegungen zu klassifizieren. Dabei wird für jede Klasse die höchste Wahrscheinlichkeit ermittelt, dass die beobachtete Bewegung durch eine bestimmte Folge von Zustandsübergängen des zugehörigen Modells erzeugt worden ist. Die Klassifizierung erfolgt durch Auswahl der Klasse mit der maximalen Wahrscheinlichkeit. Zur Berechnung der wahrscheinlichsten Zustandsfolge zu einer gegebenen Bewegung gemäß

$$\max_{q_1,\ldots,q_T} P(q_1, \ldots, q_T \mid o_1, \ldots o_T, \lambda)$$

kann der Viterbi-Algorithmus eingesetzt werden.

Zur Segmentierung und Klassifizierung von Aktionen in Bewegungssequenzen mit HMM wird angenommen, dass ein Trainingssatz mit vorsegmentierten Trajektorien aller Aktionsklassen zur Verfügung steht. Jede Aktionsklasse wird durch ein dediziertes Modell repräsentiert, dessen Parameter iterativ mit dem Baum-Welch-Algorithmus geschätzt werden. Die initiale Zustandsverteilung und die Zustandsübergangsmatrix werden dabei so eingeschränkt, dass das Modell alle Zustände in einer linearen Reihenfolge durchlaufen muss. Dies bedeutet, dass die einzigen möglichen Zustandsübergänge die zum

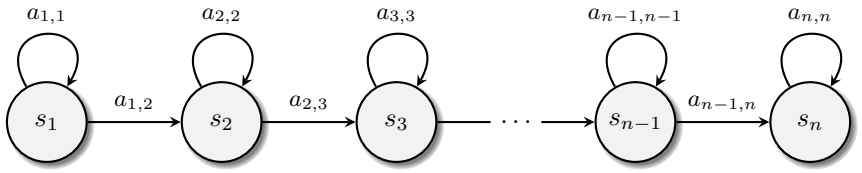


Abbildung 2.5: *Schematische Darstellung eines linearen HMM mit Zuständen s_i und Zustandsübergangswahrscheinlichkeiten $a_{i,j}$, die angeben, mit welcher Wahrscheinlichkeit ein Übergang in den aktuellen beziehungsweise in den nachfolgenden Zustand möglich ist. Zustandsübergänge, die nicht explizit als Pfeile eingezeichnet sind, sind in einem linearen HMM nicht zulässig und haben entsprechend die Wahrscheinlichkeit null.*

nachfolgenden oder zum aktuellen Zustand sind und keine Zustände ausgelassen werden dürfen. Entsprechend ergibt sich in der Matrix der Übergangswahrscheinlichkeiten eine Bandstruktur, in der nur die Einträge $a_{i,j} \neq 0$ sind, für die $i = j$ oder $i + 1 = j$ gilt. Die möglichen Übergänge eines linearen HMM, sowie die zugehörigen Übergangswahrscheinlichkeiten sind in Abbildung 2.5 dargestellt. Diese Wahl spiegelt die sequentielle Struktur der zu klassifizieren Bewegungsdaten wider. Die zeitlichen Charakteristiken der Daten werden dabei implizit in die Zustandsübergangswahrscheinlichkeiten des Modells kodiert.

2.2.3 Kombinierter Segmentierungs- und Klassifizierungsalgorithmus

Basierend auf der in Abschnitt 2.2.1 entwickelten Merkmalsrepräsentation und der in Abschnitt 2.2.2 beschriebenen probabilistischen Repräsentation von Aktionsklassen durch HMM wird im Folgenden ein Algorithmus zur simultanen Segmentierung und Klassifizierung von Aktionen in Bewegungssequenzen vorgestellt. Da zur Berechnung der Merkmalsrepräsentation eines Bewegungssegments auf alle Punkte des Segments eine Transformation angewandt wird, die von den Endpunkten des Segments abhängt, haben Änderungen der Endpunkte nicht nur lokale Auswirkungen auf die Merkmalsrepräsentation, sondern verändern potenziell die Darstellung der gesamten Sequenz. Ändern sich im Rahmen einer Segmentierung die Segmentgrenzen, so ändern sich damit die Merkmale des gesamten Segments. Aus diesem Grund können bestehende Lösungen zur HMM-basierten Segmentierung [19] nicht ohne weiteres angewandt werden.

In diesem Kapitel wird ein iterativer Ansatz zur Abgrenzung einer Aktion aus einer kontinuierlichen Bewegungssequenz verwendet. Dabei werden durch diesen Algorithmus die Grenzen der einzelnen Bewegungen so gewählt, dass sie die Likelihood der beobachteten Bewegungssequenz, gegeben die Bewegungsklassen, maximieren. Die Likelihood-Werte lassen sich aus den Aktionsmodellen berechnen, die in dem vorgestellten Ansatz als HMM kodiert werden. Da zunächst auch die Zuordnung zwischen den gegebenen Aktionsklassen und Teilen der Bewegungssequenz unbekannt ist, wird eine simultane Optimierung sowohl der Segmentgrenzen als auch der Wahl der zugrunde liegenden Aktionsklassen durchgeführt. Auf diese Weise liefert der hier vorgestellte Algorithmus sowohl die Segmentierung einer Bewegungssequenz als auch die Klassifizierung ihrer Segmente. Im Verlauf der Optimierung wird ein minimales Segment schrittweise erweitert und nach der Transformation in die Merkmalsrepräsentation die zugehörige Likelihood berechnet. Anstatt nur

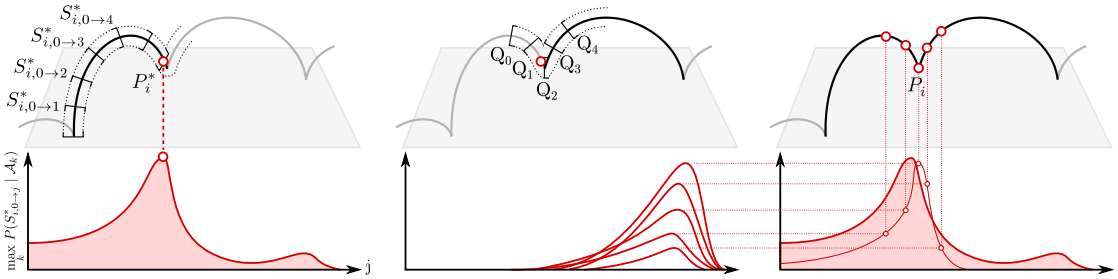


Abbildung 2.6: *Schematische Übersicht über den kombinierten Segmentierungs- und Klassifizierungsalgorithmus. In der oberen Reihe werden eine Beispielbewegung auf dem Tisch und die relevanten Segmente gezeigt. Die dazugehörigen Likelihood-Werte werden in der unteren Reihe dargestellt. In Schritt 1 (links) werden die Likelihood-Werte von Segmentkandidaten $S^*_{i,0\to j}$ berechnet und der Endpunkt P^*_i des wahrscheinlichsten Kandidaten bestimmt. In Schritt 2 (mittig) wird die Wahrscheinlichkeit der nachfolgenden Segmente berechnet, deren Ursprung Q_i in der Nachbarschaft von P^*_i liegt. In Schritt 3 (rechts) werden für alle Punkte Q_i die Likelihood-Werte, dass das erste Segment an diesem Punkt endet, mit denen, dass das zweite Segment an diesem Punkt beginnt, verrechnet. Der Punkt mit dem maximalen Ergebnis beschreibt die Segmentgrenze P_i.*

die Zeitschritte, die bis zu einer potenziellen Segmentgrenze ausgeführt werden, für die Segmentierung zu berücksichtigen, nutzt der Algorithmus zusätzlich eine Vorausschau auf das nachfolgende Segment, das an diesem Punkt startet, um dessen Likelihood mit einzubeziehen. Dies ist wichtig zum Finden des richtigen Übergangspunktes, da dieser nicht nur als Endpunkt großen Einfluss auf die dort endende Bewegung hat, sondern auch als Startpunkt auf das nachfolgende Segment. Diese Grenze wird in dieser Arbeit durch die Maximierung der kombinierten Likelihood beider Segmente bestimmt. Der Algorithmus arbeitet mit den folgenden drei Schritten, um die einzelnen Aktionen zu segmentieren und die so bestimmten Segmente zu klassifizieren. Diese sind grafisch in Abbildung 2.6 dargestellt.

Schritt 1:
In diesem Schritt wird ein Kandidat P^*_i für die Segmentgrenze P_i berechnet, die das aktuell gesuchte Segment S_i von seinem Nachfolger S_{i+1} trennt. In diesem Schritt beachtet der Algorithmus nur das aktuelle Segment S_i. Den Anfang des gesuchten Segments S_i stellt entweder der Startpunkt einer neuen Bewegungssequenz oder der Endpunkt des zuletzt bestimmten Segments dar. Der Algorithmus beginnt an dieser Stelle mit einem leeren vorläufigen Segment $S^*_{i,0\to0}$ und fügt inkrementell Trajektorienpunkte hinzu, um neue vorläufige Segmente $S^*_{i,0\to1}, \ldots, S^*_{i,0\to n}$ zu bilden. In jeder Iteration j werden die Merkmale für das aktuelle vorläufige Segment $S^*_{i,0\to j}$ nach dem in Abschnitt 2.2.1 beschriebenen Schema berechnet. Dies ist wichtig, da die Merkmale nicht nur von dem Startpunkt des jeweiligen vorläufigen Segments $S^*_{i,0\to j}$ abhängen, sondern auch von dessen Zielpunkt. Somit können verschiedene Endpunkte zu sehr unterschiedlichen Merkmalsrepräsentationen einer Bewegung führen. Mit diesen Merkmalen werden die Wahrscheinlichkeiten

$$P(S^*_{i,0\to j} \mid \mathcal{A}_k) \qquad \forall j,k \tag{2.1}$$

der Segmente $S^*_{i,0\to j}$ in Bezug auf die Aktionsmodelle $\mathcal{A}_k$ berechnet [3]. Der Index k nimmt dabei Werte zwischen eins und der Anzahl der gegebenen Aktionsmodelle an. Das wahrscheinlichste Modell $\mathcal{A}^{best}_j$ aller vorhandenen Aktionsmodelle wird für jedes der möglichen Segmente $S^*_{i,0\to 0}, \ldots, S^*_{i,0\to n}$ zusammen mit seiner Likelihood gespeichert. Aus allen diesen Segmenten und den dazugehörigen Aktionsklassen wird die Kombination mit der maximalen Likelihood bestimmt und der Endpunkt des Segments wird als potenzielle Segmentgrenze P^*_i festgehalten.

Schritt 2:
Die Wahl der Segmentgrenze beeinflusst allerdings nicht nur die vorhergehende Bewegung, die an der gesuchten Segmentgrenze endet, sondern definiert gleichzeitig den Startpunkt der nachfolgenden Aktion. Um diesen Einfluss zu berücksichtigen, werden Punkte Q_l in der lokalen Nachbarschaft der potenziellen Segmentgrenze P^*_i ausgewählt und als potenzielle Ausgangspunkte für die nachfolge Aktion S_{i+1} berücksichtigt, um die in Schritt 1 bestimmte Segmentgrenze zu verbessern. Dazu wird erneut der in Schritt 1 beschriebene Algorithmus ausgeführt, diesmal beginnend mit diesen neuen potenziellen Ausgangspunkten für das nachfolgende Segment. Dabei wird für jeden dieser Punkte eine Likelihood nach Gleichung (2.1) für die nachfolgende Aktion, die an diesem Punkt startet, und der dazugehörige Endpunkt mit der maximalen Likelihood berechnet.

Schritt 3:
Für jeden Ausgangspunkt, der in Schritt 2 untersucht wurde, wird eine kombinierte Likelihood aus den Ergebnissen von Schritt 1 und Schritt 2 berechnet. Somit wird die Likelihood des Punktes, der der Endpunkt des vorangehenden Segments S_i ist, genauso berücksichtigt wie die Likelihood, dass dieser der Startpunkt des nachfolgenden Segments S_{i+1} ist. Der Punkt P_i entspricht dem Endpunkt des Segments mit der Länge l^{best}, welches diese kombinierte Likelihood maximiert:

$$\begin{aligned} l^{best} = \arg\max_l \{ &\log P(S^*_{i,0\to l} \quad | \mathcal{A}^{best}_l) \\ &+ \log P(S^*_{i+1,l+1\to m} | \mathcal{A}^{best}_m) \} \end{aligned} \tag{2.2}$$

Dabei ist $\mathcal{A}^{best}_l$ das wahrscheinlichste Modell für das in Schritt 1 bestimmte $S^*_{i,0\to l}$ und entsprechend $\mathcal{A}^{best}_m$ das wahrscheinlichste Modell für $S^*_{i+1,l+1\to m}$ aus Schritt 2. Das an dem Punkt P_i endende Segment S_i wird entsprechend der Likelihood des Modells $\mathcal{A}^{best}_l$ klassifiziert.

Um die nächste Segmentgrenze zu bestimmen, wird der Prozess wiederholt. Dazu wird der Endpunkt $S^*_{i+1,l+1\to m}$, der in Schritt 2 im vorangegangenen Durchlauf gefunden worden ist, als vorläufige Segmentgrenze P^*_{i+1} des Schrittes 1 im darauffolgenden Durchlauf wiederverwendet. Der Prozess wird solange wiederholt, bis das Ende einer Bewegungssequenz erreicht ist und somit die komplette Bewegung segmentiert und ihre einzelnen Teilsegmente klassifiziert sind.

2.2.4 Generierungsalgorithmus für Bewegungen

Der Einsatz von HMM zur Beschreibung von Aktionsklassen ermöglicht es, gelernte Aktionen nicht nur segmentieren und klassifizieren zu können, sondern auch Bewegungen

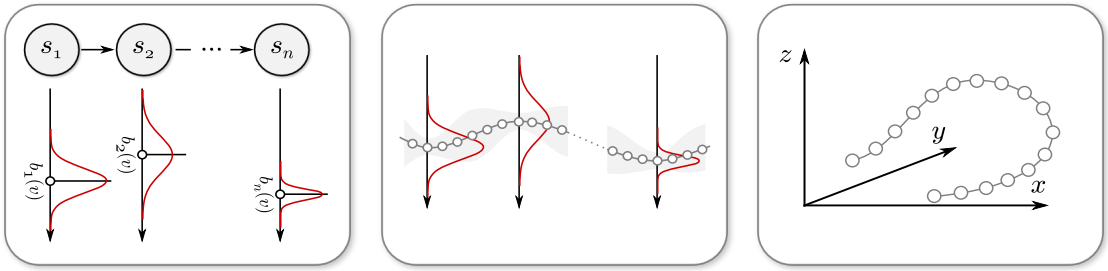
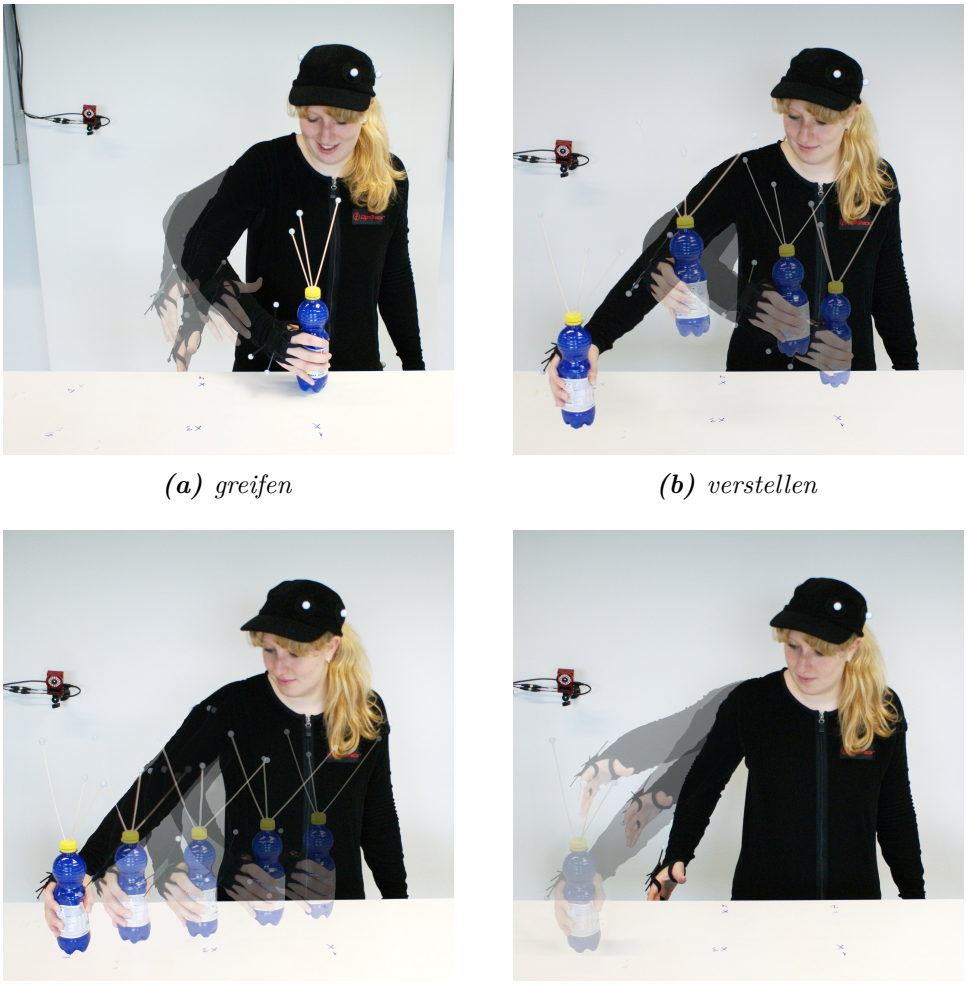


Abbildung 2.7: *Schematische Darstellung der Generierung von Trajektorien anhand des probabilistischen Modells von Aktionsklassen. In der linken Grafik sind die Zustände s_i einer Aktionsklasse mit den dazugehörigen Beobachtungsmodellen $b_i(v)$ dargestellt, aus denen Stützstellen extrahiert werden. Die mittlere Abbildung zeigt deren Interpolation mittels Gauß-Prozess-Regression. In der rechten Grafik wird aus den interpolierten Daten eine Trajektorie mit Hilfe der inversen Merkmalstransformation erzeugt, die anschließend auf einem Roboter abgespielt werden kann.*

wieder zu reproduzieren und diese auf neue Situationen zu verallgemeinern. Um eine Bewegung einer gelernten Aktionsklasse zwischen einem gewünschten Start- und Zielpunkt zu generieren, extrahiert der hier vorgestellte Algorithmus zunächst eine Folge von Stützstellen aus dem dazugehörigen Aktionsmodell. Durch Interpolation aus der generierten Folge von Stützstellen wird eine Sequenz von Merkmalen bestimmt, aus der die gewünschte Trajektorie berechnet wird.

Die Kodierung von Merkmalen in HMM ermöglicht es, ein einfaches Schema zu verwenden, um eine Sequenz von Merkmalen für eine gegebene Aktion zu generieren. Dieses wurde durch eine Arbeit von Calinon et al. [23] inspiriert. Aufgrund der verwendeten, linearen Modelltopologie werden die verdeckten Zustände der HMM sequenziell durchlaufen, wobei in jedem Zustand Emissionen gemäß normalverteilter Beobachtungswahrscheinlichkeiten generiert werden. Die Erwartungswerte der Beobachtungsverteilungen der verdeckten Zustände des HMM werden daher als Stützstellen der gewünschten Merkmalssequenz genutzt. Um eine glatte Bewegung zur Ausführung auf dem Roboter zu generieren, wird eine Gauß-Prozess-Regression [24] mit einem RBF-Kernel zur Interpolation der Merkmale zwischen den einzelnen Stützstellen verwendet. Die Gauß-Prozess-Regression modelliert implizit die Statistik der Daten und stellt einen guten Kompromiss zwischen einer genauen Anpassung an diese und einer glatten Approximation dar. Die einzige Eingabe, die das System neben den Stützstellen benötigt, ist dabei die gewünschte Anzahl an Punkten in der zu berechnenden Trajektorie. Durch eine geeignete Wahl dieses Parameters lassen sich Trajektorien mit beliebiger räumlicher Auflösung erzeugen.

Nachdem auf diese Weise eine Sequenz von Merkmalen ermittelt worden ist, kann die endgültige Trajektorie zwischen beliebigen Start- und Zielpunkten generiert werden. Dazu wird das in Abschnitt 2.2.1 beschriebene Verfahren invertiert, um durch Rücktransformation der Merkmale in den kartesischen Raum eine Trajektorie zu berechnen. Diese umgekehrte Transformation benötigt als Parameter lediglich die gewünschten Start- und Zielpunkte der Trajektorie. Da die Merkmale selbst unabhängig von diesen Punkten sind und nur die Form der Trajektorie kodieren, können mit diesem Verfahren



(a) greifen *(b) verstellen*

(c) verschieben *(d) zurückziehen*

Abbildung 2.8: *Beispiele für die vier in den Experimenten verwendeten Aktionsklassen, für die Bewegungen mit einer Motion Capture-Anlage aufgenommen wurden. Für jede Aktion werden mehrere Posen überlagert, um den zeitlichen Ablauf darzustellen. Die Zielpose jeder Aktion wird nicht-transparent angezeigt.*

auch Bewegungen für bisher unbekannte Situationen, dargestellt durch zuvor nicht trainierte Start- und Zielpunkte, generiert werden. Dieses Verfahren zur Generierung von Bewegungstrajektorien wird in Abbildung 2.7 zusammengefasst.

2.3 Experimente und Evaluation

Um die Fähigkeit des hier vorgestellten Ansatzes, Aktionen in Bewegungssequenzen zu segmentieren und zu klassifizieren, zu validieren, wurde eine Reihe von Experimenten mit Hilfe eines Motion Capture-Systems durchgeführt. Im Folgenden wird gezeigt, dass der auf Likelihood-Maximierung basierende Ansatz Segmente korrekt identifiziert und

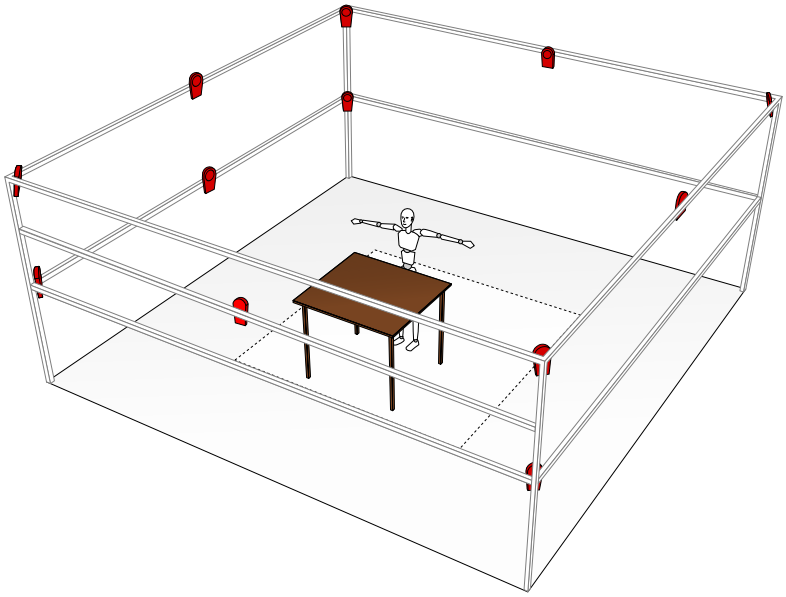


Abbildung 2.9: *Kameraanordnung der Motion Capture-Anlage für die durchgeführten Experimente.*

klassifiziert und sowohl mit komplexen Bewegungssequenzen als auch mit lückenhaften Trajektorien umgehen kann. Zudem demonstrieren die Experimente die Fähigkeit dieses Ansatzes, bekannte Aktionen zu generalisieren und glatte Trajektorien zwischen zuvor unbeobachteten Start- und Zielpunkten zu generieren. Diese werden sowohl in einer simulierten Umgebung als auch auf dem realen Roboter Baxter ausgeführt.

Als Beispielaufgabe für die Experimente wird die Manipulation eines Objekts auf einem Tisch betrachtet. Derartige Bewegungen sind in vielen realen Anwendungen für Haushaltsroboter wichtig. Dort können sie in verschiedenen Kombinationen zu Sequenzen zusammengefügt werden, um Effekte auf verschiedene Objekte auszuüben und damit komplexe Aufgaben zu lösen. Wie in Abbildung 2.8 zu sehen, konzentrieren sich die Experimente auf vier unterschiedliche Klassen von typischen Manipulationsaktionen. Dazu gehören die Bewegungen ein Objekt zu greifen, es an eine Zielposition zu verstellen, es dorthin zu verschieben, ohne es anzuheben, und die Hand auf eine für den Menschen oder später auch für den Roboter komfortable Ruheposition zurückzuführen. Die Aufgabe des Segmentierens und der Klassifizierung dieser Bewegungen wird dadurch erschwert, dass sie durch ähnliche Bewegungen des Endeffektors im kartesischen Raum beschrieben sind und zudem nahtlos ineinander übergehen.

2.3.1 Datenaufnahme

Zur Aufnahme der in diesen Experimenten verwendeten Trajektorien wird eine optische Motion Capture-Anlage der Firma OptiTrack [25] verwendet. Zu dem von OptiTrack angebotenen System gehören 12 Kameras vom Typ Flex:V100. Die Kameras liefern Bilder mit einer Auflösung von 640×480 Pixeln und einer Bildwiederholfrequenz von 100 Hz. Mit ihnen werden 2,5 cm große Marker aus mehreren Richtungen aufgenommen, so dass

ihre dreidimensionalen Positionen im Raum über die Zeit vom System bestimmt werden können. Die Marker können sowohl an speziellen Motion Capture-Anzügen befestigt werden, um die Bewegungen menschlicher Skelette aufzunehmen, als auch an Objekten in der Szene. Sie können durch die Kameras des Systems bis in eine Entfernung von ungefähr 6 m erfasst werden. Die Anordnung der Kameras in den hier durchgeführten Experimenten ist in Abbildung 2.9 schematisch dargestellt.

Mit der zum System gehörenden Software Arena [26] können sowohl menschliche Ganzkörperbewegungen, von denen für diese Experimente nur die dreidimensionale Trajektorie der Hand benötigt wird, als auch Bewegungen einzelner Objekte aufgenommen werden. Ebenso ist es möglich, die Bewegungen mehrerer menschlicher Skelette sowie eine Kombination aus Menschen und Objekten gleichzeitig zu verfolgen. Die Software berechnet anhand der aufgenommenen Daten die Zuordnung der einzelnen Marker und bestimmt die Positionen der aufgenommenen Objekte und Körperteile aus den ihnen zugeordneten Markern. Außerdem führt die Software eine automatische Nachbearbeitung und Fehlerkorrektur der Daten durch. Als Ergebnis liefert sie die kartesischen Koordinaten der einzelnen menschlichen Körperteile und die der mit Markern versehenen Objekte.

Die aufgenommenen und verarbeiteten Trajektorien können in Echtzeit über ein lokales Netz an andere Anwendungen versandt werden. Im Rahmen dieser Arbeit wird zum Empfang der Daten eine eigene, unter Verwendung des NatNet-SDK [27] von OptiTrack entwickelte Client-Anwendung [28] eingesetzt, die als ROS-Modul zur freien Verwendung veröffentlicht worden ist. Diese läuft, im Gegensatz zu der Arena Software, die für Windows entwickelt ist, unter Linux und stellt die Daten über die Robotermiddleware ROS [29] zur Weiterverarbeitung zur Verfügung.

2.3.2 Datenvorverarbeitung

Die hier durchgeführten Experimente demonstrieren die Generalisierungsfähigkeit und die Praxistauglichkeit des Ansatzes, indem der vorgestellte Algorithmus Trajektorien für gelernte Bewegungsklassen zwischen verschiedenen Punkten generiert und diese nach erfolgreicher Simulation von einem humanoiden Roboter ausgeführt werden.

Da die in den Experimenten verwendete Motion Capture-Anlage und der verwendete Roboter Baxter verschiedene Koordinatensysteme verwenden, müssen aufgenommene Bewegungsdaten im Rahmen einer Vorverarbeitung in ein gewähltes Koordinatensystem für den Roboter transformiert werden, bevor sie von dem System verarbeitet werden können. Der Ursprung des von der Motion Capture-Anlage verwendeten Koordinatensystems liegt an einem im Motion Capture-Raum festgelegten Punkt auf dem Boden. Dieser Punkt wird mittels eines speziellen Objekts, das mit der Hardware der Anlage mitgeliefert wird, während der Kalibrierung des Systems bestimmt. Als neutrale Orientierung verwendet die Motion Capture-Anlage die sogenannte T-Pose, in der die aufgenommene Person beide Arme seitlich ausgestreckt hält. Der Roboter hingegen verwendet zur Ausführung von Bewegungen ein Referenzsystem, dessen Bezugspunkt sich an der Basiseinheit des Roboters befindet. Auch die Koordinatenachsen sind in den beiden Systemen unterschiedlich gelagert. Während die Motion Capture-Anlage Bewegungsdaten in einem linkshändischen System liefert, dessen x-Achse nach rechts und dessen z-Achse nach vorne zeigt, verwendet der Roboter ein rechtshändisches System, in dem die horizon-

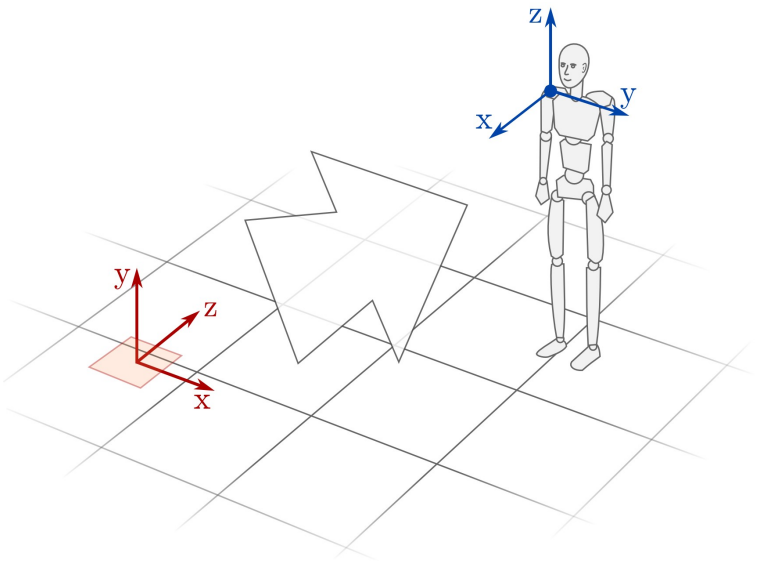


Abbildung 2.10: *Transformation des Koordinatensystems der Motion Capture-Anlage, welches in Rot dargestellt ist, in die Schulter der Versuchsperson beziehungsweise für das Generieren von Bewegungen in die des Roboters, dargestellt in Blau.*

tale Ebene durch die x-Achse (nach vorne) und die y-Achse (nach links) definiert wird. Ein weiterer Unterschied ist die verwendete räumliche Einheit. Während die Motion Capture-Anlage Positionen in Millimetern angibt, verwendet der Roboter als Maßeinheit Meter. Zur Überführung aufgenommener Bewegungsdaten in eine Repräsentation, die die obigen Unterschiede kompensiert und kompatibel mit dem Roboter ist, werden in den Experimenten einfache räumliche Transformationen verwendet. Dazu kommt, dass die Empfangszeit der angekommenen Daten gespeichert wird, da keine Zeitangaben in den Daten von der Motion Capture-Anlage enthalten sind.

Um den Effekt der in diesem Kapitel gewählten Merkmalstransformation zu untersuchen und zu veranschaulichen, ist es für die Experimente von Vorteil, wenn die demonstrierten Bewegungen unabhängig von der aktuellen Position der Versuchsperson im Motion Capture-Volumen sind. Zu diesem Zweck verschiebt der verwendete Algorithmus den Nullpunkt des Koordinatensystems in die rechte Schulter der Versuchsperson, wie in Abbildung 2.10 dargestellt. Dieses Koordinatensystem ist besonders gut geeignet, da es sich bei den in den Experimenten untersuchten Bewegungen um Manipulationsbewegungen handelt, die allesamt mit dem rechten Arm ausgeführt werden und für diese die Bewegung des Oberkörpers nicht relevant sind. Diese Normalisierung bringt den Vorteil, dass der Punkt, an dem die Bewegung aufgenommen wird, jedes Mal frei gewählt werden kann. Dadurch unterliegt er keinen Einschränkungen und es können verschiedene Situationen auf gleiche Art und Weise verarbeitet werden und müssen nicht nachträglich noch einmal nachbearbeitet werden. Hinzu kommt, dass Fehler aufgrund von Ungenauigkeiten bei der Demonstration von Bewegungen vermieden werden. Obwohl während der untersuchten Bewegungen nur der Arm bewegt wird, kommt es bei Demonstrationen häufig auch zu kleinen Bewegungen der Schulter. Diese werden vernachlässigt und es werden in

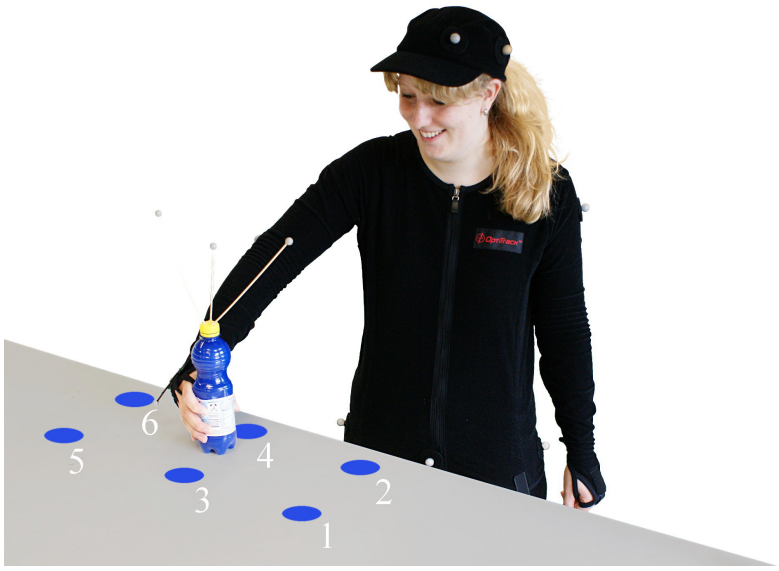


Abbildung 2.11: *Versuchsanordnung zur Aufnahme von Motion Capture-Daten. Dazu steht die Versuchsperson parallel zur Kante des Tisches, auf dem sich ein mit Markern ausgestattetes Objekt befindet. Um Verdeckungen durch die Hand zu vermeiden, befinden sich die Marker an 20 cm langen Stäben. Für die Bewegungsaufnahmen wird der Bereich des Tisches zwischen den sechs in blau eingezeichneten Positionen verwendet. Position 3 und 4 liegen dabei vor der Schulter der Versuchsperson, während 1 und 2 sich jeweils links und 5 und 6 rechts davon befinden.*

den Experimenten die über die gesamte Bewegung gemittelte Position der Schulter zur Normalisierung verwendet.

Um Effekte wie Verdeckungen und Rauschen für die beschnittenen Trainingsdaten zu reduzieren und somit bessere Eingabedaten für die probabilistischen Aktionsmodelle zu erhalten und so deren Leistungsfähigkeit zu steigern, werden eine zeitliche Abtastung und eine räumliche Glättung durch Approximation der demonstrierten Trajektorien durch Gauß-Prozesse verwendet.

2.3.3 Versuchsaufbau und Initialisierung

Für die Versuche wird im Aufnahmebereich der Motion Capture-Anlage ein Tisch platziert, auf dem sich ein Objekt befindet. Die Versuchsperson steht in den Experimenten hinter dem Tisch und manipuliert das Objekt durch die vorgegebene Auswahl möglicher Aktionen. Die Bewegungen und ihre Auswirkungen auf das Objekt werden durch die Motion Capture-Anlage aufgezeichnet und dienen als Eingabe für den Algorithmus.

Zur Vorbereitung der Experimente werden zunächst mehrere Demonstrationen jeder Aktionsklasse aufgezeichnet. Mit diesen Daten wird anschließend, wie in Abschnitt 2.2.2 beschrieben, jeweils ein HMM trainiert, um ein probabilistisches Modell der Aktionsklasse zu erzeugen. Die Start- und Endpunkte der demonstrierten Bewegungen werden durch sechs verschiedene Positionen auf dem Tisch und eine zusätzliche Ruheposition für die

Hand beschrieben. Die Positionen auf dem Tisch sind gleichmäßig in zwei Dreierreihen im Abstand von 26 cm und 40 cm zum menschlichen Oberkörper parallel zur Tischkante angeordnet. Das jeweils mittlere Element befindet sich zentral vor der Ruheposition der Hand, die anderen beiden Positionen befinden sich im Abstand von 25 cm links beziehungsweise rechts davon. Die Ruheposition der Hand befindet sich neben der menschlichen Hüfte, so dass der Arm, wie in einer typischen bequemen Haltung eines stehenden Menschen, leicht angewinkelt herab hängt. Diese Anordnung ist in Abbildung 2.11 dargestellt. Die Punkte sind so gewählt, dass sie den bequemen Aktionsradius des Menschen für Manipulationen auf einem Tisch abdecken.

Für jede dieser sechs Tischpositionen wurden 10 Demonstrationen, wie ein Objekt gegriffen beziehungsweise wie die Hand vom Objekt in die Ruheposition zurückgeführt wird, aufgezeichnet. Des Weiteren wurden von Position 1 und 6 auf dem Tisch aus jeweils fünf Verstell- und Verschiebebewegungen zu den fünf anderen vorgegebenen Positionen aufgenommen. Die verschiedenen Kombinationen von Start- und Endpunkten führen zu Unterschieden in der Form der demonstrierten Bewegung und sorgen somit für Varianz in den Trainingsdaten. Insgesamt wurden 60 Greif- und Zurückführbewegungen und 50 Verstell- und Verschiebebewegungen zu Trainingszwecken aufgenommen.

Die Referenzobjekte, die während der Experimente zusätzliche Informationen zur Berechnung der Merkmale liefern, sind die Tischoberfläche und das Objekt, das manipuliert wird. Somit ist die Eingabe für das verwendete Modell ein fünfdimensionaler Zustandsvektor $\vec{x} = [x_M, y_M, z_M, h, d]$, der neben den in Abschnitt 2.2.1 beschriebenen 3D-Merkmalen $[x_M, y_M, z_M]$, die aus der Handtrajektorie berechnet werden, den vertikalen Abstand h des Endeffektors zur Tischoberfläche und den euklidischen Abstand d zu dem relevanten Objekt enthält.

Für die Experimente wird eine Implementierung des hier vorgestellten Algorithmus in Python verwendet. Diese beruht auf der unter der LGPL verfügbaren GHMM Bibliothek [30], die eine Implementierung von Hidden Markov Modellen und häufig verwendeten Algorithmen, wie zum Beispiel des Viterbi- oder des Baum-Welch-Algorithmus, zur Verfügung stellt. Wie in Abschnitt 2.2.2 beschrieben, wird zur Kodierung von Bewegungen eine lineare Modelltopologie mit 10 Zuständen verwendet. Die Anzahl der Zustände ist dabei empirisch anhand der aufgenommenen Daten bestimmt. Die Verteilung π der Startzustände ist so gewählt, dass jede Trajektorie im ersten Zustand s_0 beginnt. Die fünfdimensionalen Beobachtungsvektoren entstammen in den Experimenten einem kontinuierlichen Alphabet $\mathcal{A} \subseteq \mathbb{R}^5$. Dabei wird davon ausgegangen, dass sich durch die gewählte Merkmalstransformation die Bewegungen einer Aktionsklasse als Folgen normalverteilter Positionen darstellen lassen, und entsprechend wird eine mehrdimensionale Normalverteilung als Beobachtungsmodell gewählt.

2.3.4 Kombinierte Segmentierung und Klassifizierung

In diesem Abschnitt werden die Ergebnisse der Experimente vorgestellt, mit denen die Leistungsfähigkeit des hier vorgestellten Segmentierungs- und Klassifizierungsansatzes anhand verschiedener Fragestellungen untersucht wird. Es wird gezeigt, dass der Ansatz durch die Maximierung der Likelihood gute Segmentierungs- und Klassifizierungsergebnisse erreicht und dass diese durch Berücksichtigung der Likelihood des nachfolgenden

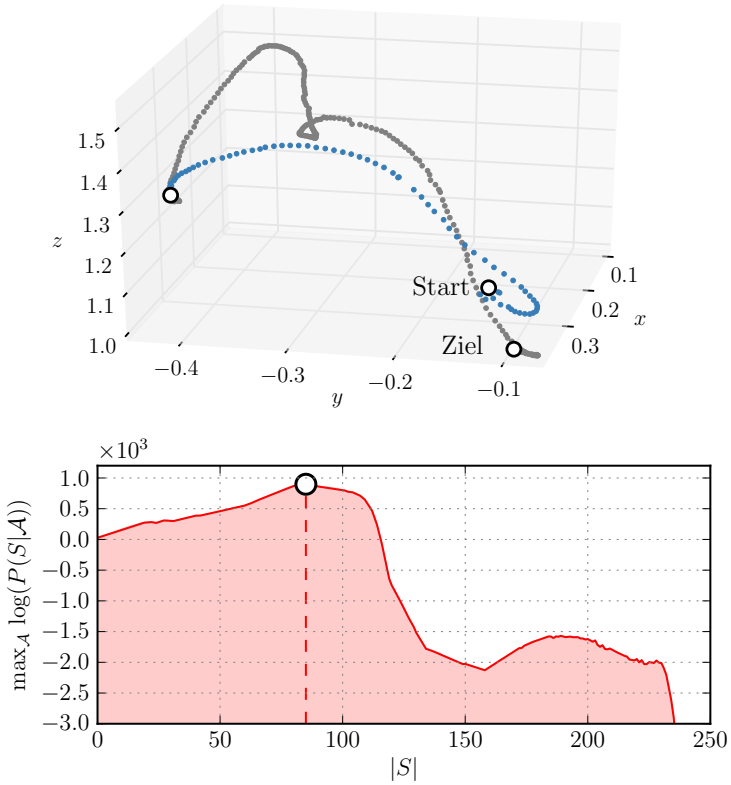
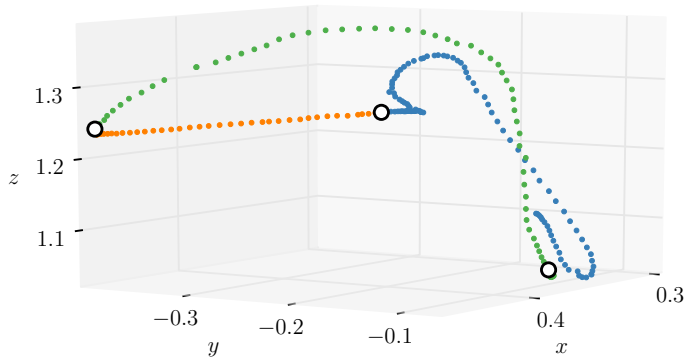


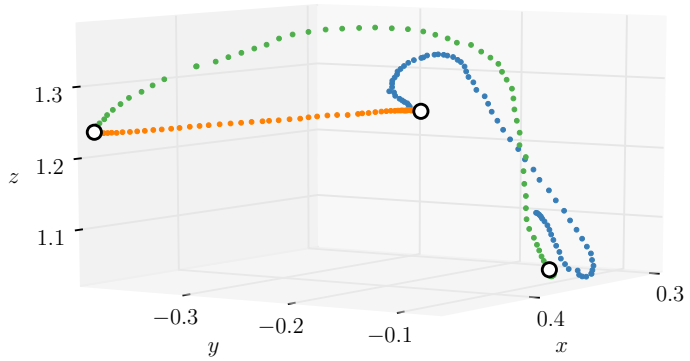
Abbildung 2.12: *Illustration der Segmentierung einer Bewegung durch Maximierung der Likelihood. Oben: Beispieltrajektorie, in der das, durch den Algorithmus korrekt segmentierte, erste Segment blau hervorgehoben ist. Unten: Log-Likelihood für verschiedene mögliche Segmentlängen $|S|$. Das durch den Algorithmus gefundene globale Maximum, das die Grenze zwischen dem ersten und zweiten Segment darstellt, ist durch einen weißen Kreis hervorgehoben.*

Segments noch verbessert werden können. Zudem wird die Zuverlässigkeit des Algorithmus anhand der Fehlerrate aus 107 untersuchten Bewegungssequenzen und sein Verhalten bei komplexen und langen Bewegungen beziehungsweise lückenhaften Daten untersucht. Für diese Experimente wurde eine Reihe von Bewegungssequenzen aufgenommen, die eine variierende Anzahl an Einzelaktionen enthalten, mit denen ein Objekt zwischen verschiedenen Start- und Zielpunkten manipuliert wurde.

In dem vorgestellten Ansatz werden Aktionsgrenzen durch das Maximieren der Likelihood der Eingabedaten im Bezug auf die gelernten Aktionsklassen bestimmt. Abbildung 2.12 zeigt eine Beispieltrajektorie und die Grenze des ersten Segments, die durch den Algorithmus gefunden wird. Dazu ist die Entwicklung der Log-Likelihood der möglichen Segmente dargestellt, die im Rahmen des Suchprozesses stetig erweitert werden. Es ist zu erkennen, dass die Likelihood zunächst, mit Ausnahme einiger Ausreißer in den Eingabedaten, monoton wächst und an dem Endpunkt des ersten Segments ein



(a) *Ergebnis ohne Berücksichtigung des nachfolgenden Segments*



(b) *Ergebnis unter Berücksichtigung des nachfolgenden Segments*

Abbildung 2.13: *Vorteile des vorausschauenden Ansatzes. Die Segmentierung, die nur das vorangehende Segment bei der Suche nach einer Segmentgrenzen miteinbezieht (oben) führt zu einem suboptimalen Ergebnis, bei dem die Greifbewegung (blau) in die Verschiebebewegung (orange) hineinreicht. Die Berücksichtigung des aktuellen und des nachfolgenden Segments (unten) führt hier zu einer verbesserten Segmentierung.*

ausgeprägtes globales Maximum erreicht. Bei der Betrachtung größerer Segmente wird bei einer Segmentgröße von ungefähr 185 Posen ein weiteres lokales Maximum in der Likelihood gefunden. Dieses stellt die Grenze zwischen dem zweiten und dritten Segment der Bewegung dar und ist dadurch zu erklären, dass die Form der kombinierten Bewegung aus dem ersten und zweiten Segment im Raum der Merkmale grob einer Greifbewegung ähnelt.

In vielen Fällen kann die Grenze zwischen zwei Segmenten bereits korrekt ermittelt werden, indem ein Algorithmus, wie oben gezeigt, nur den Endpunkt des vorausgehenden Segments durch Optimierung von dessen Log-Likelihood sucht. In einigen Fällen führt diese einseitige Betrachtung jedoch zu einer systematisch fehlerhaften Segmentierung. Abbildung 2.13 zeigt den Unterschied zwischen der Wahl der Segmentgrenze, wenn nur das vorangehende Segment in Betracht gezogen wird, und die Grenze, die gewählt wird, wenn zusätzlich das nachfolgende Segment mit einbezogen wird. Dabei führt die erste Wahl zu

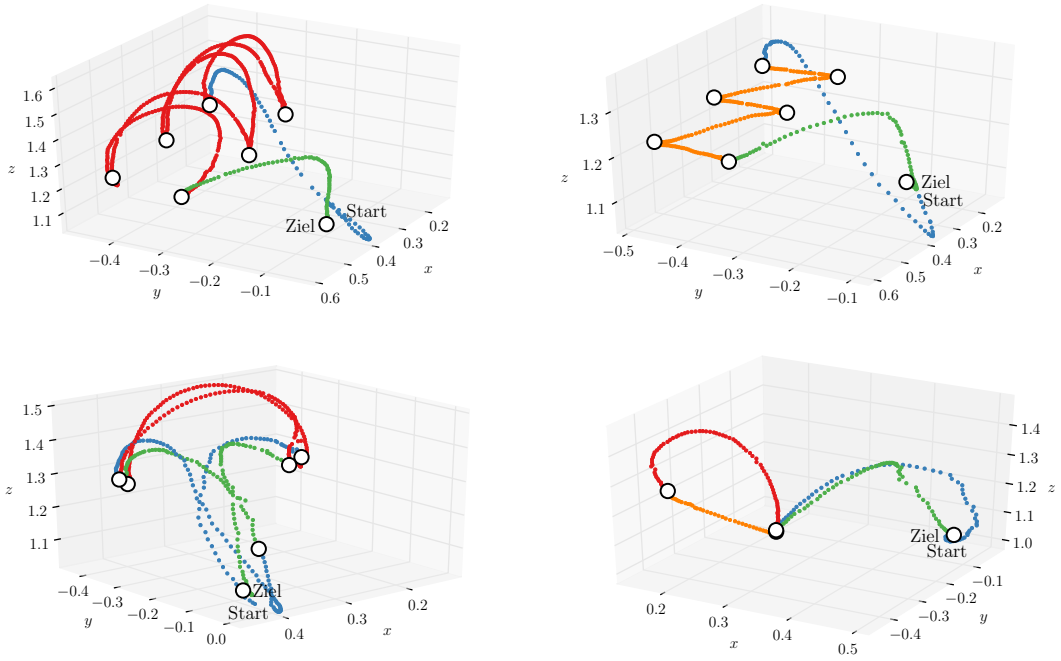


Abbildung 2.14: *Beispiele von komplexen Bewegungssequenzen, die mit dem hier vorgestellten Algorithmus erfolgreich segmentiert und die Segmente den richtigen Aktionsklassen zugeordnet werden. Die verschiedenen Aktionsklassen werden durch unterschiedliche Farben gekennzeichnet: Greifen eines Objektes (blau), verstellen des Objektes (rot), verschieben des Objektes (orange) und das Zurücknehmen der Hand in eine bequeme Ruheposition (grün). Die gefundenen Segmentgrenzen werden durch weiße Kreise dargestellt.*

einem Segment, das über die tatsächliche Grenze der ersten Teilbewegung hinausgeht und sich auf einen Teil der nachfolgenden Bewegung ausdehnt. Durch zusätzliche Betrachtung der Log-Likelihood des nachfolgenden Segments wird, wie im unteren Teil der Abbildung gezeigt, die korrekte Segmentgrenze gefunden.

Um auch quantitativ die Performanz des präsentierten Ansatzes zu beurteilen, wurden 107 Sequenzen aufgenommen, die aus jeweils drei typisch aufeinander folgenden Aktionen bestehen. Am Anfang jeder Bewegungssequenz greift die Versuchsperson, die die Bewegungen vorführt, das Objekt auf dem Tisch. Danach verstellt sie das Objekt entweder oder sie verschiebt es, ohne es anzuheben, zu der gewünschten Zielposition. Am Ende jeder Aktionsfolge wird die Hand, wie bei menschlichen Bewegungen typisch, in eine bequeme Ruheposition zurückgeführt. Die Bewegungssequenzen wurden in dem gleichen Bereich des Tischs ausgeführt, in dem auch die Modelle der Aktionsklassen trainiert wurden. Allerdings wurde die Position des Objekts in der Testphase nicht auf die sechs im Training verwendeten Positionen limitiert, sondern Testdaten in dem gesamten durch die Trainingspunkte umfassten Bereich aufgenommen und somit auch Positionen gewählt, die zwischen den gelernten Punkten lagen. Der in diesem Kapitel beschriebene Algorithmus zur Segmentierung und Klassifizierung wird auf die aufgenommenen Sequenzen angewendet und die fehlerhaften Segmentierungen und Klassifizierungen gezählt.

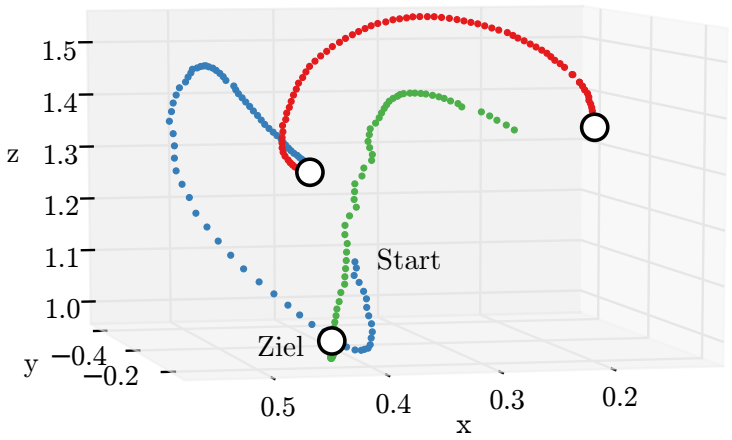


Abbildung 2.15: *Performanz des hier vorgestellten Ansatzes bei unvollständigen Daten. Die Beispielbewegungen werden korrekt erkannt, obwohl die Daten eine größere Lücke zwischen der Verstellbewegung (rot) und dem Zurückführen der Hand zu der Ruheposition (grün) aufweisen.*

In den 321 aufgenommenen Segmenten gibt es 13 falsch segmentierte Aktionen. Dies entspricht einer Fehlerfreiheit von 96%. Die Fehler entstehen meistens durch besonders ausschweifende Greifbewegungen, die von dem Algorithmus in zwei separate Aktionen unterteilt werden. Dieser Fehler hat allerdings keinen Einfluss auf nachfolgende Segmente. Der Klassifikationsalgorithmus erreicht 100% Fehlerfreiheit bei richtig segmentierten Aktionen und klassifiziert somit keine einzelne Aktion unzutreffend.

Das vorherige Experiment zeigt die Leistungsfähigkeit des hier vorgestellten Ansatzes an exemplarischen Sequenzen, bestehend aus jeweils drei Segmenten. Die Fähigkeit dieses Ansatzes, Sequenzen, die aus einer größeren Anzahl von Aktionen zusammengesetzt sind, zu segmentieren und die Teilstücke richtig zu klassifizieren, wird in Abbildung 2.14 demonstriert. Alle gezeigten Sequenzen beginnen, wie typische menschliche Aktionsfolgen, mit einer Greifbewegung, gefolgt von einer Reihe von Verstell- und Verschiebebewegungen mit dem gegriffenen Objekt. Am Ende jeder Sequenz wird die Hand in eine bequeme Ruheposition zurückgeführt. Die gezeigten Beispiele sind repräsentativ aus den untersuchten Bewegungen gewählt und zeigen, dass alle Einzelaktionen korrekt segmentiert und klassifiziert werden.

Für die Experimente wurden die zu verarbeitenden Bewegungssequenzen mit einer Motion Capture-Anlage aufgenommen. Durch verschiedene Effekte, wie zum Beispiel Ungenauigkeit der Kameras oder Verdeckungen der vom System verfolgten Marker, kann es zu Rauschen, Ausreißern und Lücken in den Eingabedaten kommen, die sich in einigen Fällen über mehrere Zeitschritte erstrecken. Der hier vorgestellte Algorithmus ist robust gegenüber solchen Fehlern und benötigt daher keine spezielle Vorverarbeitung oder Fehlerbereinigung der Eingabedaten. Dies wird in Abbildung 2.15 dargestellt, in der an einem Beispiel gezeigt wird, dass trotz einer lückenhaften Trajektorie die Segmentierung und Klassifizierung korrekt durchgeführt wird.

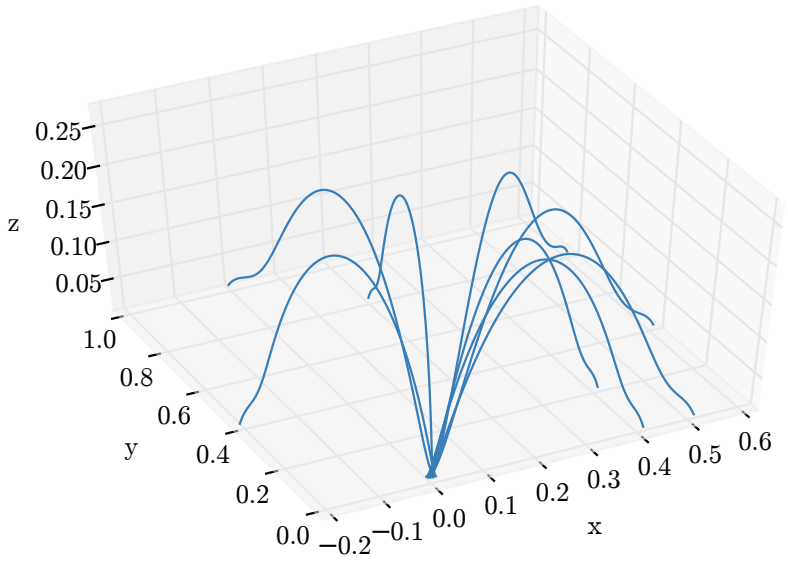


Abbildung 2.16: *Generierte Verstellbewegungen von derselben Startposition zu verschiedenen Zielpositionen auf der Tischoberfläche.*

Die beschriebenen Experimente zeigen, dass der kombinierte Segmentierungs- und Klassifizierungsalgorithmus beliebige Kombinationen aus verschiedenen Bewegungen aus aufgenommenen Motion Capture-Daten zuverlässig extrahiert und den entsprechenden Aktionsklassen zuordnet. Dies schließt lückenhafte Eingabedaten ein, die durch den vorgestellten Algorithmus ebenfalls korrekt segmentiert und klassifiziert werden. Dabei profitiert der Algorithmus von der beidseitigen Optimierung der Segmentgrenzen, wodurch die Genauigkeit der Segmentierung verbessert wird.

2.3.5 Reproduktion und Generalisierung von Bewegungen

Das Ziel von Bewegungsgeneralisierung ist es, für beliebige Situationen Trajektorien zu generieren, welche auf der einen Seite die charakteristischen Merkmale der gelernten Aktionen aufweisen, während sie auf der anderen Seite von den Besonderheiten der einzelnen Trainingsbewegungen abstrahieren. Die Generalisierung auf beliebige Start- und Zielpunkte von Bewegungen wird in dem hier vorgestellten Ansatz durch die Transformation von Trajektorien in eine Merkmalsrepräsentation erreicht, die Bewegungen unabhängig von diesen beiden Punkten darstellt. Zur Erzeugung glatter Merkmalstrajektorien aus den Zustandssequenzen und den Beobachtungsmodellen des HMM wird die Gauß-Prozess-Regression zur Interpolation der anhand der Beobachtungsmodelle gewonnenen Stützstellen angewendet. Die Generalisierungsfähigkeit dieses Ansatzes, die es erlaubt, Trajektorien zwischen beliebigen Start- und Zielpunkten zu generieren, wird in Abbildung 2.16 demonstriert. Sie zeigt mehrere Verstellbewegungen, die von einer gemeinsamen Startposition aus zu verschiedenen Positionen auf dem Tisch in unterschiedlichen Richtungen und Entfernungen generiert wurden. Nur die gewünschte Start- und Zielposition im kartesischen Raum werden als Eingabe für den Generierungsalgorithmus

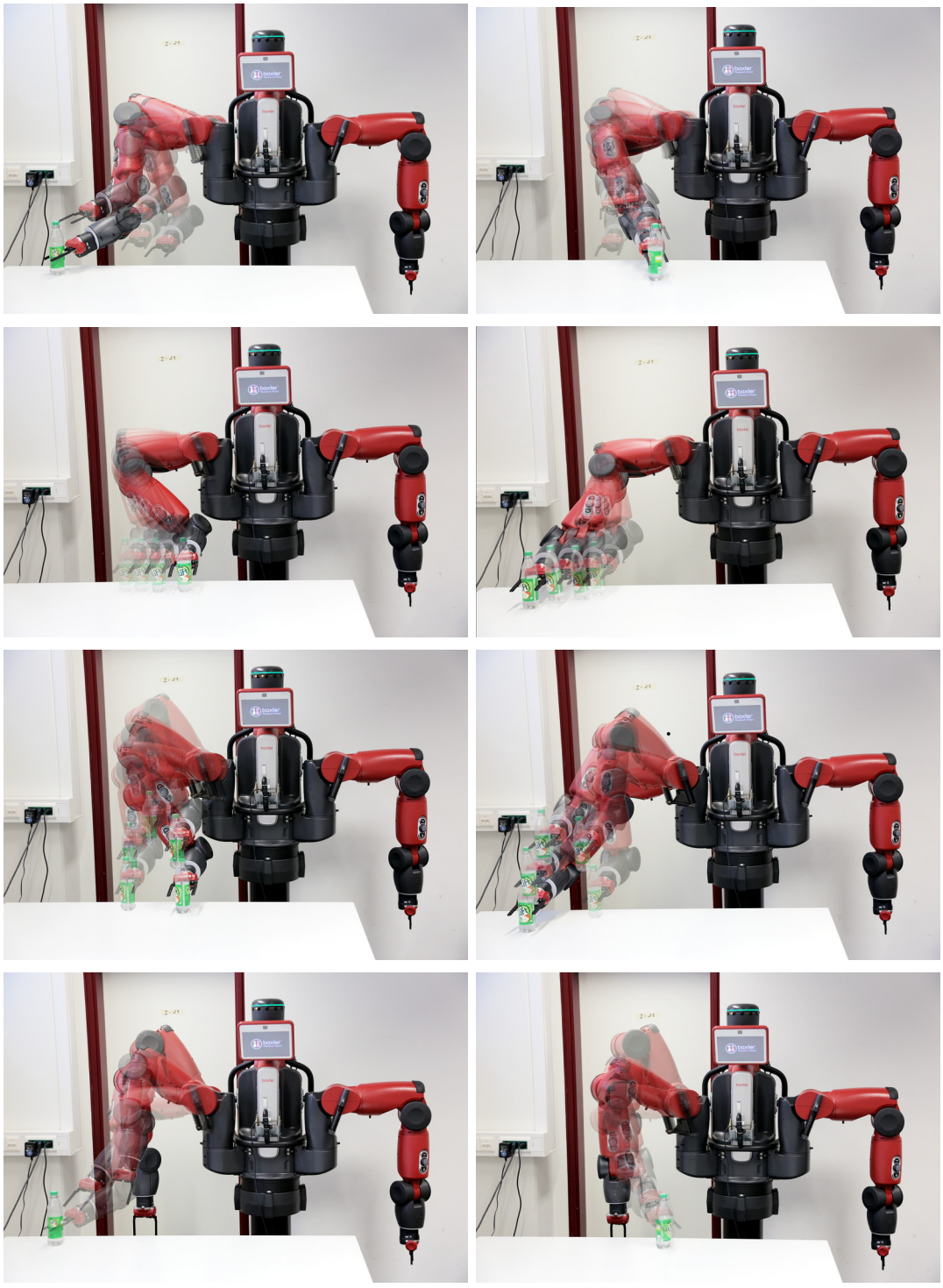


Abbildung 2.17: *Beispiele für generierte Bewegungen, die auf dem Roboter Baxter ausgeführt werden. Dabei werden die vier verschiedenen Aktionsklassen für jeweils zwei unterschiedliche Start- und Zielpositionen in einer Zeile dargestellt. Die verschiedenen Zeilen stellen von oben nach unten die folgenden Bewegungsklassen dar: Greifen eines Objektes, das Verstellen dieses, das Verschieben eines Objektes und das Zurückziehen der Hand auf die Ruheposition. In jedem Bild werden mehrere Posen überlagert, um den zeitlichen Ablauf darzustellen. Die Zielpose jeder Aktion wird nicht-transparent angezeigt.*

benötigt. Die Form der generierten Bewegungen wird ausschließlich aus den gelernten Modellen berechnet. Die Abbildung zeigt, dass die charakteristische Form der Bewegung über die verschiedenen Generalisierungen erhalten bleibt.

Um die generierten Trajektorien auf dem Roboter auszuführen, ist es wichtig, dass die Bewegungen den kinematischen Fähigkeiten des Roboters entsprechen. In den hier vorgestellten Experimenten wird der Roboter Baxter [31] der Firma Rethink Robotics [32] verwendet. Baxter ist ein humanoider, anthropomorpher Roboter, der mit zwei Armen ausgestattet ist. Jeder der beiden Arme verfügt über sieben Freiheitsgrade. Baxter wurde im Jahr 2012 vorgestellt und soll in industriellen Anwendungen eingesetzt werden, in denen Kooperation mit Menschen wünschenswert ist. Zusätzlich ist Baxter auch als wissenschaftliche Plattform zu erwerben, die im Gegensatz zu der Industrieversion ein SDK [33] beinhaltet und es somit erlaubt, eigene Programme auf dem Roboter auszuführen. Diese Software baut auf der Robotikmiddleware ROS auf und ermöglicht somit eine einfache Anbindung eigener Programme sowie eine einfache Integration mit anderen ROS-kompatiblen Softwareprodukten. Baxter kann auf einem Fuß montiert werden, der separat erworben werden kann. Auf diesem erreicht der Roboter eine Gesamthöhe von 1,91 m. Dadurch ist er in der Lage, bequem Objekte auf einem normalen Tisch greifen zu können. Wird der Roboter ohne diesen Fuß verwendet, erreicht er eine Größe von 0,94 m. Das Gewicht des Roboters zusammen mit dem Fuß ist 139 kg. Die Länge eines Arms ohne einen Greifer beträgt dabei 1,04 m. Die für die Experimente verwendete Version von Baxter ist mit elektrischen parallelen Greifern, ebenfalls von der Firma Rethink Robotics, ausgestattet, mit denen Objekte mit einer Größe von 6 cm bis 10 cm gegriffen werden können. Die zu manipulierenden Objekte sollten dabei ein Gewicht von bis zu 2,3 kg nicht überschreiten.

Ebenfalls Teil des SDK ist eine auf Gazebo [34] aufbauende physikbasierte Simulation des Roboters sowie die Konfiguration für die Anbindung des Roboters an die Software MoveIt! [35] zur Planung von Bewegungen. Da die Trajektorien, die durch die in diesem Kapitel vorgestellte Methode generiert werden, nur die Positionen des Endeffektors des Roboters beinhalten, wird die Software MoveIt! zur Planung der Gelenkwinkel des gesamten Roboterarms, ausgehend von der durch das System generierten Trajektorie, genutzt. Die Abbildung 2.17 zeigt den Roboter Baxter bei der Ausführung von Bewegungen, die mit dem hier vorgestellten Algorithmus anhand des probabilistischen Aktionsmodells erzeugt werden. Für jede der vier Aktionsklassen, die in den Experimenten betrachtet werden, sind jeweils zwei exemplarische Bewegungen zwischen unterschiedlichen Start- und Zielpositionen dargestellt. Diese liegen innerhalb des Arbeitsbereiches des Roboters, der auch dem Bereich entspricht, der bei der Aufzeichnung der Trainingsdaten verwendet wurde. Die Darstellungen demonstrieren die Fähigkeit des Ansatzes, verschiedene Arten gelernter Bewegungen auf zuvor nicht beobachtete Situationen zu generalisieren. Gleichzeitig zeigen sie, dass dabei die charakteristische Form der Bewegungen beibehalten wird und dass der Algorithmus zu glatten Trajektorien zwischen den gewählten Endpunkten führt.

2.4 Zusammenfassung

In diesem Kapitel wurde ein neuer, inkrementeller Algorithmus zur probabilistischen Aktionserkennung in aufgenommenen Bewegungssequenzen eines Menschen, zur Generalisierung gelernter Aktionen auf unbekannte Situationen und zur Erzeugung glatter Trajektorien auf Basis von HMM vorgestellt.

Durch eine lineare Transformation der Bewegungssequenzen in eine Repräsentation relativ zu den Endpunkten der Bewegung erhält der Algorithmus eine Merkmalsdarstellung, die von der konkreten Aufgabenstellung abstrahiert und so zu Aktionsklassen mit einer geringen Intra-Klassen-Varianz führt. Zudem wird durch diese Repräsentation die charakteristische Form gerichteter Bewegungen erhalten. Im Gegensatz zu vielen anderen Ansätzen ermöglicht der hier vorgestellte Ansatz die Segmentierung, Klassifizierung, Reproduktion und Generalisierung von Aktionen auf der Basis eines gemeinsamen probabilistischen Aktionsmodells. Da die Ergebnisse der Segmentierung und der Klassifizierung sich gegenseitig beeinflussen, wird ein kombinierter Algorithmus vorgeschlagen, der diese beiden Aufgaben gemeinsam löst. Somit werden bei der Segmentierung Informationen über bekannte Aktionsklassen berücksichtigt und die Genauigkeit der Klassifizierung wird durch die Bestimmung der richtigen Segmentgrenze verbessert. Um zuverlässig gute Segmentierungsergebnisse zu erhalten und systematische Fehler zu vermeiden, wird in dem hier vorgestellten Algorithmus eine iterative Vorausschau über das aktuelle Segment hinaus genutzt. Dies erlaubt es, bei der Evaluation einer möglichen Segmentgrenze die Likelihood-Werte sowohl von vorangehenden Aktionen als auch von darauffolgenden Aktionen zu berücksichtigen. Somit werden durch den in diesem Kapitel präsentierten Algorithmus Segmentgrenzen ermittelt, die gleichzeitig wahrscheinliche Endpunkte von Segmenten und wahrscheinliche Startpunkte ihrer Nachfolger darstellen.

Die durchgeführten Experimente zeigen, dass der vorgestellte Ansatz in der Lage ist, unterschiedlich lange Sequenzen aus verschiedenen Aktionen korrekt zu segmentieren und zu klassifizieren. Dabei ist der Ansatz robust gegenüber Unregelmäßigkeiten wie beispielsweise Lücken in den Eingabedaten. Zudem wird gezeigt, dass der kombinierte Segmentierungs- und Klassifizierungsansatz in Verbindung mit der beidseitigen Optimierung der Segmentgrenzen zu einer deutlichen Verbesserung der Segmentierungsgenauigkeit führt. Auf einem Beispielsatz von 321 Aktionen erreicht der Algorithmus eine Fehlerfreiheit von 96% für die Segmentierung und 100% für die Klassifizierung der korrekt erkannten Aktionen.

Durch Extraktion von Stützstellen aus den HMM der Aktionsklassen und deren Interpolation mittels Gauß-Prozess-Regression ist es mit dem Ansatz möglich, gelernte Aktionen unter Beibehaltung der charakteristischen Merkmale der Originalbewegungen zu reproduzieren und für diese glatte Trajektorien zu erzeugen. Die Wahl der invertierbaren Merkmale ermöglicht es zusätzlich, diese Bewegungen auf neue, unbekannte Situationen zu generalisieren. In den Experimenten wird demonstriert, dass die durch den Ansatz generierten Trajektorien auch auf einem humanoiden Roboter ausgeführt werden können.

3 Kapitel Vorarbeiten zum Lernen von Bewegungsprimitiven

Die in den folgenden Kapiteln vorgestellten Ansätze zum Lernen komplexer Aufgaben bauen auf der eigenen Diplomarbeit mit dem Titel „Lernen von Greifbewegungen aus Imitation und eigener Erfahrung“ [36] und den in den Arbeiten [37] und [38] von uns präsentierten Verfahren zum Lernen von Bewegungsprimitiven auf. Diese integrieren probabilistisches Bestärkendes Lernen in kontinuierlichen Zustands- und Aktionsräumen und das Lernen aus Demonstrationen als alternative Kontrollflüsse in einem kohärenten System. Hierfür wird ein Entscheidungskriterium vorgeschlagen, anhand dessen das System in jeder Situation unter Berücksichtigung des vorhandenen Wissens eines der beiden Lernverfahren wählt, um diese entsprechend ihrer jeweiligen Stärken einzusetzen. Hierzu approximiert das System die Q-Funktion über den kontinuierlichen Raum der Zustände und Aktionen anhand gesammelter Erfahrungen durch Gauß-Prozesse. Erfahrungen können dabei sowohl durch Nachahmung demonstrierter Aktionen als auch durch Ausführung von autonom ausgewählten Aktionen gewonnen werden. Ist das vorhandene Wissen nicht ausreichend, fordert das System selbstständig eine Demonstration der Aufgabe an. Andernfalls generiert es autonom eine Lösung, indem es die vorhandenen Erfahrungen generalisiert und versucht, diese zu optimieren. Dazu wird im Bestärkenden Lernverfahren eine spezielle Strategie zur Auswahl der zu testenden Parameter genutzt. Diese basiert auf dem Kriterium der Erwarteten Veränderung, das nach vielversprechenden Aktionen sucht, allerdings gleichzeitig verhindert, dass das System Aktionen wählt, deren Ausgang nicht vorhersagbar ist. Die wesentlichen Merkmale, durch die sich das System auszeichnet, und die sich dadurch ergebenden Vorteile für das Lernen von Bewegungsprimitiven in kontinuierlichen Räumen lassen sich wie folgt zusammenfassen:

Effizientes Lernen in kontinuierlichen Räumen: Die Approximation der von Bestärkenden Lernverfahren verwendeten Q-Funktion ermöglicht dem Verfahren ein effizientes Lernen von Bewegungsprimitiven in kontinuierlichen Räumen, die für reale Anwendungen benötigt werden. Die Gauß-Prozess-Approximation erlaubt eine probabilistische Vorhersage des Nutzens von beliebigen Paaren aus Zustand und Aktion anhand einer kleinen Menge von beobachteten Daten. Die Verwendung der Erwarteten Veränderung als Optimierungskriterium ermöglicht die Formulierung einer effizienten Explorationsstrategie, aufbauend auf der Gauß-Prozess-Approximation. Durch die gleichwertige Integration von Bestärkendem Lernen und Lernen aus Demonstrationen kann Expertenwissen genutzt werden, um die Suche des Bestärkenden Lernens zu fokussieren, gleichzeitig aber den

Arbeitsaufwand des Experten gering zu halten.

Sicherheit: Ein weiterer wichtiger Aspekt in praktischen Robotikanwendungen ist die Sicherheit des Roboters, seiner Umgebung und insbesondere der Menschen, die sich dort aufhalten oder mit dem Roboter zusammenarbeiten. Die von dem hier präsentierten Verfahren verwendete Explorationsstrategie im Bestärkenden Lernen beinhaltet zu diesem Zweck eine Abwägung bei der Aktionswahl, durch die nicht nur der potenziell erreichbare Q-Wert optimiert, sondern auch die Unsicherheit der Vorhersage berücksichtigt wird. Somit werden Aktionen, die sicher negative Konsequenzen haben und Aktionen, deren Q-Werte nicht sicher vorhersagbar sind und die somit Risiken bergen, vermieden. Für zusätzliche Sicherheit sorgt die Kombination mit dem Wissen eines Experten, die es dem System erlaubt, Demonstrationen anzufordern, falls die Vorhersage aufgrund der vorliegenden positiven Trainingsdaten eine zu große Unsicherheit aufweist und das System den Ausgang einer möglichen Lösung nicht abschätzen kann.

Intuitive Interaktion: Roboter, die für Serviceaufgaben eingesetzt werden sollen, müssen auch von Anwendern ohne technische Ausbildung im Alltag bedienbar sein. Um dies zu erreichen, werden in dem System Lernverfahren eingesetzt, die durch die menschliche Lernpsychologie motiviert und typischen menschlichen Lernverhalten nachempfunden sind. Somit sind die wesentlichen Abläufe beim Training des Systems für Anwender intuitiv verständlich. Durch Verwendung von Demonstrationen eines Experten als Ausgangspunkt für die Explorationsstrategie des Bestärkenden Lernens weisen die vom System generierten Aktionen eine hohe Ähnlichkeit zu den Lösungen eines Menschen auf. Die durch das System generierten Aktionen führen somit nicht zu für Menschen unerwarteten Aktionen sondern entsprechen dem von Menschen erwarteten Verhalten zum Lösen einer Aufgabe. Dies fördert die Akzeptanz des Roboters als Kooperationspartner.

Die Vorteile dieses Verfahrens für das Lernen von Bewegungsprimitiven sollen auch in den Ansätzen der nachfolgenden Kapitel, welche sich im Gegensatz zu den hier vorgestellten Vorarbeiten mit dem Lernen von komplexen Aufgaben auf höheren Abstraktionsebenen beschäftigen, ausgenutzt werden. Zum besseren Verständnis werden die dazu verwendeten Teilverfahren und die dazugehörigen Grundlagen in den folgenden Abschnitten als Vorarbeiten eingeführt.

3.1 Approximation der Q-Funktion durch Gauß-Prozesse

Für das Lernen von Fähigkeiten ist es in den meisten Fällen nicht zielführend, Lösungsbeispiele für eine Menge beobachteter Situationen auswendig zu lernen. Vielmehr ist es notwendig, im Laufe der Zeit gemachte Erfahrungen zu nutzen und sie zu verallgemeinern, um daraus Wissen für ähnliche Aufgaben zu erlangen. Dies ist vor allem dann wichtig, wenn, wie in der hier vorgestellten Arbeit, Aufgaben mit kontinuierlichen Zustands- beziehungsweise Aktionsräumen gelöst werden sollen. In diesen Fällen ist es weder sinnvoll noch möglich, jede Situation und Aktion zu besuchen. Eine wichtige Eigenschaft eines Lernverfahrens ist daher, aus wenigen Beispielen gute Schätzungen über Lösungen im kompletten Zustands- und Aktionsraum generalisieren zu können.

Für die Entwicklung maschineller Lernverfahren wird diese Lernaufgabe als Suche nach einem funktionalen Zusammenhang zwischen zwei Größen formalisiert, von denen angenommen wird, dass sie korreliert sind. Mit Hilfe statistischer Verfahren können anschließend anhand einer vergleichsweise kleinen Menge an Beobachtungen Aussagen über Funktionswerte für beliebige Eingaben getroffen werden.

In den vorangegangenen Arbeiten zum Lernen von Bewegungsprimitiven wird zu diesem Zweck eine Approximation der gesuchten Funktion durch *Gauß-Prozesse* [24] eingesetzt. Bei Gauß-Prozessen handelt es sich um ein stochastisches Modell, das eine Wahrscheinlichkeitsverteilung über Funktionen durch einen Zufallsprozess modelliert. Dies heißt durch eine indizierte Menge von Zufallsvariablen, die die Gesamtheit aller Funktionswerte darstellen. Ein wichtiges Merkmal der Gauß-Prozesse ist dabei, dass jede endliche Teilmenge von Zufallsvariablen gemeinsam normalverteilt ist. Somit lassen sich Gauß-Prozesse als eine Verallgemeinerung der mehrdimensionalen Normalverteilung von endlich-dimensionalen Vektorräumen auf unendlich-dimensionale Räume von Funktionen auffassen. Jede Realisierung der Zufallsvariablen liefert dabei anschaulich eine Folge von Funktionswerten. Ein Gauß-Prozess wird, analog zur endlich-dimensionalen Normalverteilung, vollständig durch eine Mittelwert- und eine Kovarianzfunktion definiert:

$$\mu(x) : \mathcal{X} \to \mathbb{R} \qquad\qquad k(x, x') : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$$

Die Kovarianzfunktion gibt dabei an, wie sehr sich die Funktionswerte an zwei unterschiedlichen Punkten im Eingaberaum $\mathcal{X}$ beeinflussen. Entsprechend lassen sich bei der Modellierung einer Funktion mittels Gauß-Prozessen durch die Wahl der Kovarianzfunktion Annahmen über die Struktur der modellierten Funktion ausdrücken.

Mit Hilfe dieses Modells lässt sich Wissen über den Verlauf einer unbekannten Funktion nutzen, um mittels Bayesscher Inferenz Aussagen über wahrscheinliche Funktionswerte an unbekannten Stellen zu treffen. Wird eine Menge von Trainingspunkten $x \in \mathcal{X}$ mit bekannten Funktionswerten $y \in \mathbb{R}$ betrachtet sowie eine zweite Menge von Punkten $x_* \in \mathcal{X}$, deren Funktionswerte $y_* \in \mathbb{R}$ unbekannt sind, so führt die Annahme, dass sich die gesuchte Funktion durch einen Gauß-Prozess beschreiben lässt zu einer gemeinsamen a priori-Verteilung der bekannten und unbekannten Funktionswerte:

$$p\left(\begin{bmatrix} y \\ y_* \end{bmatrix}\right) = \mathcal{N}\left(0, \begin{bmatrix} K + \sigma^2 I & K_* \\ K_*^T & K_{**} \end{bmatrix}\right) \tag{3.1}$$

Dabei wird vereinfachend ein Wert von null für die Mittelwertfunktion angenommen. Falls entsprechendes Vorwissen verfügbar ist, lassen sich im Gauß-Prozess-Ansatz auch andere Mittelwertfunktionen verwenden. Die Matrizen K, K_* und K_{**} enthalten dabei die Kovarianzen zwischen den vorherzusagenden Punkten x_* beziehungsweise zwischen x_* und den Trainingspunkten x, die anhand der Kovarianzfunktion $k(x, x')$ ermittelt werden:

$$K_{ij} = k(x_i, x_j)$$

Eine häufige Wahl für die Kovarianzfunktion sind dabei radiale Funktionen der folgenden Form

$$k(x, x') = \theta \ \exp\left(-\frac{1}{2}(x - x')^T \Sigma^{-1} (x - x')\right). \tag{3.2}$$

Aufgrund ihrer Form wird diese Art von Kovarianzfunktion auch als Gauß-Kernel[1]

[1] engl.: Squared Exponential (SE) Kernel

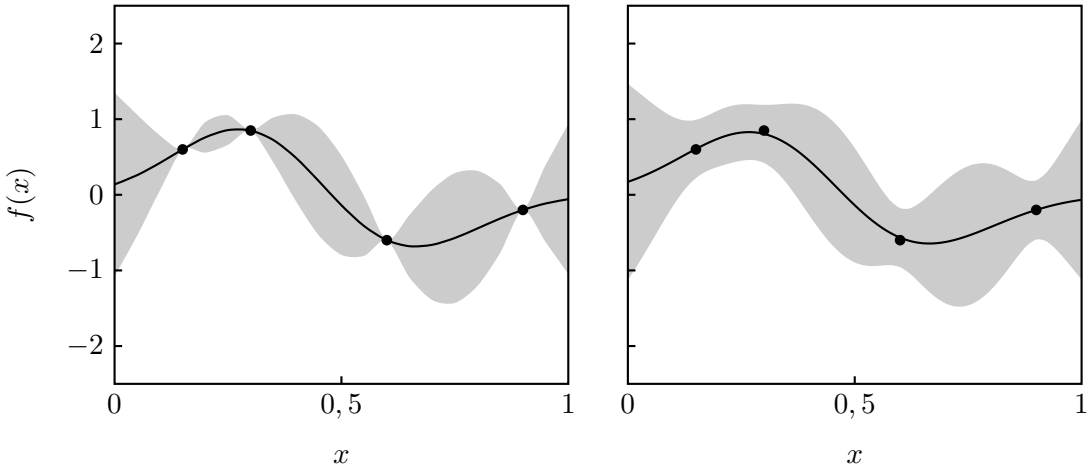


Abbildung 3.1: *Beispiele für a posteriori-Verteilungen der durch einen Gauß-Prozess repräsentierten Funktionswerte. Die schwarzen Punkte entsprechen dabei den beobachteten Trainingsdaten, während die Mittelwerte und Unsicherheiten der vorhergesagten Funktionswerte durch eine schwarze Linie und die sie umgebende graue Fläche dargestellt werden. Allgemein ist zu sehen, dass sich die Unsicherheit der Vorhersagen in der Umgebung der beobachteten Beispiele reduziert. In der linken Grafik wird angenommen, dass diese Beispiele fehlerfrei beobachtet wurden. Entsprechend ist die a posteriori-Unsicherheit an diesen Stellen in dem gewählten Beispiel null. Dagegen wird in der rechten Abbildung davon ausgegangen, dass die Beobachtungen mit Rauschen behaftet sind, so dass die beobachteten Funktionswerte hier lediglich approximiert werden und nach der Inferenz eine Rest-Unsicherheit an diesen Stellen verbleibt.*

bezeichnet. Der Parameter θ bestimmt dabei die Varianz der Funktionswerte an einem einzelnen Punkt, während die Elemente der Matrix Σ die charakteristische Längenskala der Kovarianzfunktion festlegen. Häufig wird für Σ die Form einer Diagonalmatrix gewählt, deren Diagonalelemente beschreiben, wie schnell sich die Werte der gesuchten Funktion in den verschiedenen Eingabedimensionen verändern. Da in den Vorarbeiten zum Lernen von Bewegungsprimitiven mit Eingaberäumen mit einem Euklidischen Distanzmaß gearbeitet wird, wird dort ebenfalls diese Form der Kovarianzfunktion ausgewählt. Durch diese Wahl kann der gewünschte Effekt erzielen werden, dass die Funktionswerte von Punkten, die im Eingaberaum nah beieinanderliegen, durch die Kovarianzfunktion stärker korreliert sind als solche von Punkten, die weit auseinanderliegen.

Aus der gemeinsamen a priori-Verteilung (3.1) können mittels Bayesscher Inferenz Schätzungen der unbekannten Funktionswerte y_*, gegeben die Trainingsdaten (x, y), bestimmt werden. Aufgrund der Eigenschaft von Gauß-Prozessen, dass alle endlichen Teilmengen von Zufallsvariablen gemeinsam normalverteilt sind, lassen sich die bedingten Verteilungen $y_* \mid y$ wie folgt analytisch berechnen:

$$\begin{aligned} \mu(x_*) &= K_*^T C^{-1} y \\ \sigma^2(x_*) &= K_{**} - K_*^T C^{-1} K_* \end{aligned} \tag{3.3}$$

Die Matrix C ist dabei gegeben durch $C = K + \sigma^2 I$. Als Ergebnis des Verfahrens wird

eine vollständige a posteriori-Verteilung der geschätzten Funktionswerte an den Stellen x_* zurückgeliefert, deren Varianz Rückschlüsse auf die Ähnlichkeit der Anfragepunkte x_* zu den Trainingsdaten zulässt. Diese Eigenschaft wird im weiteren Verlauf dieser Arbeit an verschiedenen Stellen ausgenutzt, um die Sicherheit von Lernverfahren zu verbessern. Beispiele für a posteriori-Verteilungen der Funktionswerte sind in Abbildung 3.1 dargestellt.

In Gleichung (3.3) fällt auf, dass die Punkte x und x_* nicht direkt verwendet werden. Stattdessen gehen sie indirekt durch die Einträge der Kovarianzmatrizen K, K_* und K_{**}, deren Einträge durch die Funktion $k(x, x')$ bestimmt werden, in die Berechnung ein. Dies hat den Vorteil, dass je nach Problemstellung verschiedene Arten von Kovarianzfunktionen passend zu der Struktur der jeweiligen Eingabedaten gewählt werden können und die Methode somit auf unterschiedlichste Arten von Eingabedaten anwendbar ist. Diese Eigenschaft wird in Kapitel 4 ausgenutzt, um mit Hilfe eines Gauß-Prozess-Modells ein Lernverfahren für Sequenzen von Bewegungen zu entwickeln. Darüber hinaus werden in Gleichung (3.3) die Vorhersagen für die Punkte x_* unmittelbar aus den Trainingsdaten berechnet. Die Gauß-Prozess-Regression wird daher auch als nicht-parametrisches Lernverfahren bezeichnet, da keine explizite Wahl eines parametrischen Modells für die gesuchte Funktion erforderlich ist. Stattdessen können Annahmen über die Struktur der gesuchten Funktion durch Angabe einer a priori-Verteilung mit entsprechend gewählten Mittelwert- und Kovarianzfunktionen in dem Verfahren berücksichtigt werden.

3.2 Sicheres Bayessches Bestärkendes Lernen

Bei *Bestärkendem Lernen*[2] [39, 40] handelt es sich um ein Teilgebiet des maschinellen Lernens, das auch in der Robotik zur Entwicklung leistungsfähiger Lernalgorithmen eingesetzt wird. Algorithmen für Bestärkendes Lernen stellen einen Mittelweg zwischen überwachten Lernverfahren, die zum Lernen für jedes Beispiel eine vollständige Lösung benötigen, und unüberwachten Lernverfahren, die keinerlei Lösungsinformationen verwenden, dar. Die Verfahren sind inspiriert durch die menschliche Lernpsychologie. Im Besonderen von der Art und Weise, wie Menschen durch Interaktion mit ihrer Umgebung und durch Beobachtung der Auswirkungen der eigenen Aktionen lernen, Aktionen so zu wählen, dass sie einen bestimmten Effekt erreichen. Für Anwendungen des maschinellen Lernens wird diese Lernart mit Hilfe der Theorie der Markov-Entscheidungsprobleme [41] formalisiert, in der ein lernender Agent den Zustand seiner Welt durch eine Folge von Aktionen beeinflussen kann und hierfür nach jeder Aktion eine Rückmeldung über die Auswirkungen dieser Aktionen auf die Umgebung erhält. Diese unmittelbare Rückmeldung wird dabei durch eine skalare Belohnungsfunktion modelliert, die den Agenten in seinen Handlungen bestärkt beziehungsweise ihn von diesen abhält.

Die Lernaufgabe für ein gestelltes Problem besteht beim Bestärkenden Lernen darin, mit der Hilfe von gesammelten Informationen über zu erwartende Belohnungen für verschiedene Situationen und Aktionen eine Aktionsfolge zu finden, die die erwartete Summe der Belohnungswerte über die Zeit maximiert. Gegenüber überwachtem Lernen haben Bestärkende Lernverfahren den Vorteil, dass ein Roboter mit ihrer Hilfe Fähigkeiten

[2] engl.: Reinforcement Learning

erwerben kann, selbst wenn keine algorithmische Lösung für das entsprechende Problem vorliegt. Stattdessen verwendet das Verfahren Belohnungswerte, die abhängig von dem gestellten Problem explizit durch einen Menschen gegeben werden können oder aber auch implizit in Form einer Berechnungsvorschrift, mit deren Hilfe der Agent beispielsweise anhand von Sensorwerten, die die Effekte einer ausgeführten Aktion messen, einen Belohnungswert berechnen kann. Ein Designer eines Robotersystems muss somit selbst nicht wissen, wie eine Aufgabe zu lösen ist, sondern muss lediglich in der Lage sein, die Effekte von Aktionen des Roboters zu bewerten, und kann diesen anschließend autonom Lösungen finden lassen.

Formal wird die Lernaufgabe im Bestärkenden Lernen durch die aktuelle Situation $s \in \mathcal{S}$ charakterisiert, in der sich der Agent zu einem Zeitpunkt t befindet, und von der er durch Ausführen der Aktion $a \in \mathcal{A}$ in die Folgesituation $s' \in \mathcal{S}$ gelangt. Zusätzlich erhält der Agent für das Ausführen der Aktion a in der Situation s mit dem Ergebnis, in Situation s' zu gelangen, eine Belohnung $r_{t+1} \in \mathbb{R}$. Die Aktionen, die der Agent ausführt, werden anhand einer Strategie $\pi : \mathcal{S} \rightarrow \mathcal{A}$ ausgewählt, die jedem Zustand eine Aktion zuordnet. Der kumulierte Wert der Belohnungen $V^\pi(s) : \mathcal{S} \rightarrow \mathbb{R}$ wird, in Zustand s startend und der Strategie π folgend, wie folgt bestimmt:

$$V^\pi(s) = \sum_{i=0}^{\infty} \gamma^i r_{t+i+1} \tag{3.4}$$

Dabei wird in dieser Arbeit davon ausgegangen, dass die Umgebung des Roboters deterministisch ist, was bedeutet, dass sich der Agent nach Ausführung einer Aktion a in einem Zustand s immer in demselben Nachfolgezustand s' befindet. In diesem Fall entspricht die Summe aller Belohnungen ab einem Zustand s dem Nutzen des Zustands s für den Agenten. Entsprechend wird die Funktion $V^\pi(s)$ als Nutzenfunktion[3] der Strategie π bezeichnet. Der Faktor γ stellt in der obigen Gleichung einen Diskontierungsfaktor dar, der bestimmt, wie stark die einzelnen Belohnungswerte, die der Agent für die Ausführung einer Folge von Aktionen erhalten hat, in der Berechnung von $V^\pi(s)$ berücksichtigt werden. Der Wert dieses Faktors wird üblicherweise im Bereich $0 \leq \gamma \leq 1$ gewählt, so dass in den meisten Fällen die Belohnungen für unmittelbar ausgeführte Aktionen stärker gewichtet werden als Belohnungen für Aktionen in der fernen Zukunft. Ziel des Bestärkenden Lernens ist es, die optimale Strategie $\pi^\star$ zu finden, durch die für jede Situation diejenige Aktion gewählt wird, die die Langzeitbelohnung des Agenten maximiert.

Die Grundlagen zur algorithmischen Lösung dieser Art von Problemen wurden von Bellman et al. [39] in den fünfziger Jahren durch Arbeiten im Bereich der Theorie der optimalen Steuerung gelegt. Sie verwenden zur Bestimmung der optimalen Strategie die optimale Nutzenfunktion, die mit Hilfe dynamischer Programmierung bestimmt wird. Die optimale Nutzenfunktion $V^*(s)$ beschreibt dabei diejenige Nutzenfunktion, die alle optimalen Strategien miteinander teilen, und wird dazu für jede Situation als das Maximum über die Kombination des unmittelbaren Gewinnes mit dem Nutzen der Nachfolgesituationen berechnet. Diese erfüllt die rekursive Bellman-Gleichung:

$$V^*(s) = \max_a (r(s,a) + \gamma V^*(s')) \tag{3.5}$$

[3] engl.: value function

Dabei bezeichnet $V^*(s')$ die optimale Nutzenfunktion des Nachfolgezustandes s'. Eine Strategie, die zu jedem Zustand eine Aktion wählt, durch die der optimale Nutzen V^* erreicht wird, stellt eine optimale Strategie π^* dar. Zur Berechnung der Nutzenfunktion nach diesem Schema benötigt der Agent allerdings ein Modell der Zustandsübergänge, zu denen die verschiedenen Aktionen führen. Außerdem ist ein Modell der Belohnungen erforderlich, die der Agent für die verschiedenen Zustandsübergänge erhält. Diese stellen eine Voraussetzung dar, um bei der Auswahl von Aktionen deren Ausgang berücksichtigen zu können. Eine Alternative dazu ist das Lernen der sogenannten Q-Funktion, die den Nutzen beschreibt, Aktion a in einer bestimmten Situation s auszuführen. Für die Bestimmung dieser Werte ist kein Modell der Umgebung nötig. Stattdessen kann anhand der Q-Funktion eine optimale Strategie bestimmt werden, indem der Agent den in der Q-Funktion gespeicherten Nutzen der ihm zur Verfügung stehenden Aktionen vergleicht, ohne die tatsächlichen Ergebnisse der einzelnen Aktionen kennen zu müssen. Diese Methode wird daher auch als modellfrei bezeichnet. Durch das Lernen der Q-Funktion, die als Argument sowohl eine Aktion als auch eine Situation enthält, entfällt unter anderem das Problem, dass Aktionen, die zu schlechten Belohnungen führen, den Nutzen der gesamten Situation entwerten, wie dies beim Lernen mit der Nutzenfunktion aus Gleichung (3.5) der Fall sein kann. Die folgende Q-Funktion erfüllt dabei ebenfalls das Bellmansche Optimalitätsprinzip:

$$Q^*(s,a) = r(s,a) + \gamma \max_{a'} Q^*(s',a') \tag{3.6}$$

Um diese Funktion durch Interaktion mit der Umgebung zu lernen, entwickelte Watkins [42] den Q-Lernalgorithmus. Dabei bezeichnet $Q_t(s,a)$ die Schätzung der optimalen Q-Funktion zum Zeitpunkt t. Der Agent beobachtet in jedem Zeitschritt t in einem Zustand s für eine Aktion a, die anhand einer Explorationsstrategie π_x ausgewählt wurde, einen Nachfolgezustand s' und eine Belohnung $r(s_t, a_t)$. Anhand dieser Werte wird die Q-Funktion wie folgt aktualisiert:

$$Q_{t+1}(s,a) = (1-\alpha) \cdot Q_t(s,a) + \alpha \cdot [r(s,a) + \max_{a'} Q_t(s',a')] \tag{3.7}$$

Dabei stellt α die sogenannte Lernrate dar, die in einem Bereich zwischen $0 < \alpha \leq 1$ liegt.

Da insbesondere in praktischen Anwendungen das Sammeln von Informationen aufgrund einer aufwendigen Auswertung der Zielfunktion oft teuer ist, werden spezielle Algorithmen benötigt, die effizient gute Lösungen finden, indem sie auf geschickte Weise die nächsten zu testenden Aktionen wählen. Menschen entscheiden dies intuitiv, indem sie ihre vorherigen Versuche analysieren und dieses Wissen nutzen, um die Folgen möglicher Aktionen abzuschätzen und daraufhin eine geschickte Wahl für eine neue Aktion zu treffen. Diese Entscheidungsfindung versuchen maschinelle Lernverfahren durch mathematische Verfahren zu realisieren, die die zuvor gesammelten Informationen in die Entscheidung für eine Aktion einfließen lassen. Somit wird ein sinnvolles Gleichgewicht zwischen Exploration und der Berücksichtigung bereits vorhandenen Wissens gebildet, um die Anzahl der Testversuche zu reduzieren.

In der hier vorgestellten Arbeit wird eine Bayessche Strategie [43] verwendet, um eine informierte Entscheidung über vielversprechende, als nächstes zu evaluierende Aktionen

zu treffen. Diese nutzt die im vorangegangenen Kapitel beschriebene Gauß-Prozess-Approximation der Q-Funktion über den Raum der Zustände und Aktionen, um Schätzungen für unbekannte Paare aus Zuständen und Aktionen zu ermitteln:

$$Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R} \sim \mathcal{GP}(\mu(x), k(x, x')) \qquad x, x' \in \mathcal{S} \times \mathcal{A}$$

Dabei stellen $\mu(x)$ und $k(x, x')$ die Mittelwert- beziehungsweise Kovarianzfunktionen dar, durch die der Gauß-Prozess definiert wird. Ein wesentlicher Vorteil der Verwendung von Gauß-Prozessen für die Approximation ist, dass diese zusätzlich zu Informationen über den erwarteten Wert der Q-Funktion eines Paares aus Zustand und Aktion auch eine Unsicherheit liefern, die mit der Schätzung verbunden ist. Um diese Unsicherheit bei der Auswahl von Aktionen zu berücksichtigen, wird die sogenannte *Erwartete Verbesserung*[4] [44, 45] verwendet. Diese beschreibt, welche Verbesserung an einem neuen Testpunkt gegenüber einem Vergleichswert $\overline{f}$ zu erwarten ist. Eine mögliche Verbesserung kann dabei sowohl durch die Differenz des Erwartungswertes der Q-Funktion für ein Paar aus Zustand und Aktion zu diesem Vergleichswert begründet sein als auch durch eine große Unsicherheit der Schätzung des Erwartungswertes, da diese die Wahrscheinlichkeit einer Vergrößerung der Belohnung ebenfalls beeinflusst. Mit Hilfe der Erwarteten Verbesserung kann ein Suchkriterium berechnet werden, das eine Abwägung zwischen der Erkundung von unbekannten Regionen und der Ausnutzung von vielversprechenden bekannten Bereichen ausdrückt.

Je nach Aufgabenstellung und Interpretation des Begriffes Verbesserung lässt sich die Erwartete Verbesserung sowohl für die Suche nach einem Maximum als auch für die Suche nach einem Minimum definieren. In den folgenden Erklärungen steht dabei $\oplus$ für die Berechnung der entsprechenden Variablen für ein Maximierungsproblem, während $\ominus$ diese für ein Minimierungsproblem beschreibt. Sind die Variablen mit $\odot$ gekennzeichnet, so können in den Formeln entweder die mit $\oplus$ oder die mit $\ominus$ gekennzeichneten Werte eingesetzt werden, je nachdem, welche Problemstellung betrachtet wird. Die vorhergesagte Verbesserung gegenüber $\overline{f}$ für ein Maximierungs- beziehungsweise Minimierungsproblem ist gegeben durch:

$$\begin{aligned} D^{\oplus}(x, \overline{f}) &= \max(f(x) - \overline{f}, 0) \text{ und} \\ D^{\ominus}(x, \overline{f}) &= \max(\overline{f} - f(x), 0) \end{aligned} \tag{3.8}$$

Die Erwartete Verbesserung entspricht dem Erwartungswert dieser Größe, der als Integral über die Dichtefunktion definiert ist. Ist der vorhergesagte Funktionswert $f(x)$ um den Mittelwert $\mu(x)$ mit der Varianz $\sigma^2(x)$ normalverteilt, so kann der Wert der Erwarteten Verbesserung $\mathbf{E}D^{\odot}$ an der Stelle x in geschlossener Form berechnet werden [46]:

$$\begin{aligned} \mathbf{E}D^{\odot} &= \int_{-\infty}^{+\infty} D^{\odot} \cdot p(D^{\odot})\, dD^{\odot} \\ &= \int_{-\infty}^{\infty} D^{\odot} \cdot \frac{1}{\sqrt{2\pi} \cdot \sigma(\vec{x})} \exp\left(-\frac{(D^{\odot} - (\overline{f} - \mu(\vec{x})))^2}{2 \cdot \sigma^2(\vec{x})} \right) dD^{\odot} \\ &= \sigma(x) \cdot \left[u^{\odot} \cdot \Phi(u^{\odot}) + \phi(u^{\odot}) \right] \end{aligned} \tag{3.9}$$

[4] engl.: Expected Improvement

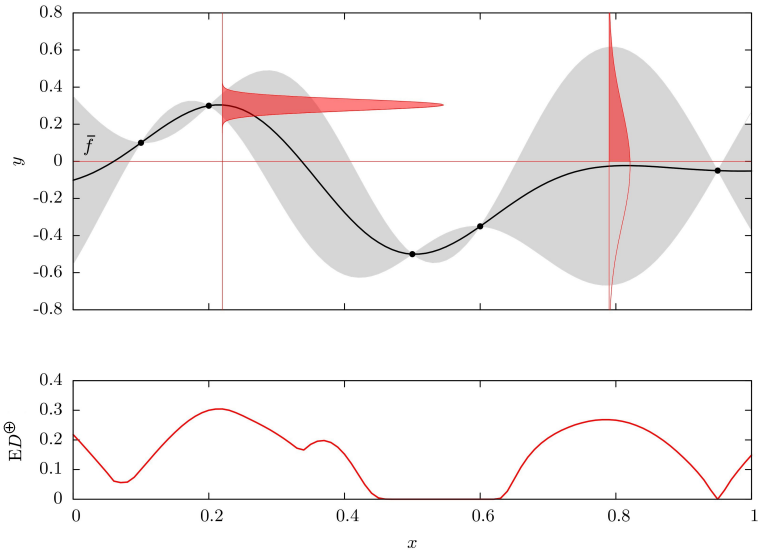


Abbildung 3.2: *Illustration der Erwarteten Verbesserung an einem Beispiel. In der oberen Grafik stellt die schwarze Linie die a posteriori-Mittelwertfunktion eines Gauß-Prozesses dar. Diese wurde anhand der als schwarze Punkte eingezeichneten Trainingsbeispiele bestimmt. Zudem ist als grau schattierte Fläche die Standardabweichung der einzelnen Funktionswerte eingezeichnet. Exemplarisch sind die Dichtefunktionen von zwei Funktionswerten in rot dargestellt. Der rot schattierte Bereich überdeckt den Bereich ihrer Verteilung, in dem der Funktionswert über dem Referenzwert $\overline{f}$ liegt, und endet daher an der roten horizontalen Linie bei $y = 0$, die $\overline{f}$ beschreibt. Die Erwartete Verbesserung entspricht dem Abstand des Schwerpunktes der schattierten Fläche zu $\overline{f}$. In der unteren Grafik sind die Erwarteten Verbesserungen aller Funktionswerte dargestellt. Die Werte der Erwarteten Verbesserung der beiden Beispielpunkte der oberen Grafik sind dabei etwa gleich groß, obwohl der rechte Punkte einen deutlich niedrigeren Mittelwert aufweist. Der Grund hierfür liegt darin, dass die Erwartete Verbesserung die Varianz in die Berechnung mit einbezieht. Diese ist für den rechten Punkt größer.*

Dabei sind $\Phi(u)$ und $\phi(u)$ die Verteilungs- beziehungsweise Dichtefunktion der Standardnormalverteilung:

$$\Phi(u) = \frac{1}{2} \cdot \operatorname{erf}\left(\frac{u}{\sqrt{2}}\right) + \frac{1}{2} \qquad \phi(u) = \frac{1}{\sqrt{2\pi}} \cdot \exp\left(-\frac{u^2}{2}\right) \tag{3.10}$$

Die darin vorkommende Variable u wird je nachdem, ob es sich ein Maximierungs- oder Minimierungsproblem handelt, wie folgt definiert:

$$u^{\oplus} = \frac{\mu(x) - \overline{f}}{\sigma(x)} \qquad u^{\ominus} = \frac{\overline{f} - \mu(x)}{\sigma(x)} \tag{3.11}$$

Der erste Term der Erwarteten Verbesserung $\mathbf{E}D^{\odot}$ beschreibt die vorhergesagte Differenz zwischen $\overline{f}$ und dem Mittelwert der Schätzung an der zu untersuchenden Stelle x, die in den hier verwendeten Anwendungen aus einem Paar aus Zustand und Aktion besteht. Durch

Multiplikation mit dem zweiten Term wird diese Differenz mit der Wahrscheinlichkeit der Verbesserung gewichtet. Die Erwartete Verbesserung hat somit ihr Maximum an der Stelle x, an der der Mittelwert besser ist als der bisherige beste Wert und die Unsicherheit hoch ist. Ebenso nimmt sie große Werte an, wenn die Vorhersage klein ist und die Unsicherheit hoch, da durch die große Unsicherheit der Schätzung die Möglichkeit besteht, dass die Verbesserung an dieser Stelle größer ist als erwartet. Dagegen geht das Ergebnis der Erwarteten Verbesserung bei bekannten Punkten und bei Punkten, an denen keine Verbesserung erwartet wird, gegen null. Insgesamt sind die Werte der Erwarteten Verbesserung nie negativ. Ein Beispiel der Erwarteten Verbesserung ist in Abbildung 3.2 dargestellt.

Da zur Bestimmung einer geeigneten Aktion nach einem Maximum der Q-Funktion gesucht wird, wird für $\mathbf{E}D^{\ominus}$ die Bezeichnung *Erwartete Verschlechterung* eingeführt, da der Wert in diesem Fall beschreibt, welche Verschlechterung gegenüber $\overline{f}$ zu erwarten ist. Den Wert $\mathbf{E}D^{\oplus}$ wird weiterhin als Erwartete Verbesserung bezeichnet. Ist ein Funktionslevel $\overline{f}$ als Vergleichswert gegeben, so wird der gemäß der Erwarteten Verbesserung beziehungsweise Verschlechterung an der Stelle x erreichbare Funktionswert definiert als

$$\begin{aligned} \mu^{\oplus}(x, \overline{f}) &= \overline{f} + \mathbf{E}D^{\oplus}(x, \overline{f}) \\ \mu^{\ominus}(x, \overline{f}) &= \overline{f} - \mathbf{E}D^{\ominus}(x, \overline{f}). \end{aligned} \tag{3.12}$$

Würde zur Auswahl auszuführender Aktionen nur mit Hilfe der Erwarteten Verbesserung nach einem Maximum der Q-Funktion gesucht werden, so wäre dies als Entscheidungskriterium für theoretische Anwendungen durchaus möglich. Grobe Fehlentscheidungen auf einem realen System, wie in der Robotik, können allerdings zu fatalen Folgen für den Roboter oder seine Umgebung führen und müssen somit vermieden werden. Aus diesem Grund wird für die Suche nach Aktionen ein kombiniertes Entscheidungskriterium verwendet. Während in den meisten Ansätzen [47] nur die Erwartete Verbesserung für die Suche nach einem Maximum oder einem Minimum eingesetzt wird, kombiniert der Ansatz die Erwartete Verbesserung mit der Erwarteten Verschlechterung. Durch diese Kombination entsteht eine effiziente Strategie bei der Suche nach vielversprechenden Lösungen, während gleichzeitig verhindert wird, dass Aktionen ausgewählt werden, deren Ausgang aufgrund der zur Verfügung stehenden Informationen nicht sicher vorhersagbar ist. Die durch diese Suchstrategie ausgewählten Aktionen stellen einen Kompromiss zwischen Aktionen mit einem guten vorhergesagten Q-Wert und Aktionen, deren erwartetes Ergebnis mit einer großen Unsicherheit versehen ist, dar. So wird unter anderem der Nutzen von Aktionen abgeschwächt, die zwar auf der einen Seite einen hohen Q-Wert versprechen, aber auf der anderen Seite auch eine hohe Unsicherheit aufweisen. Würde das System nur die Erwartete Verbesserung berücksichtigen, so wären diese Aktionen vielversprechend, da sie eine besonders hohe Verbesserung erwarten lassen. Es würde allerdings nicht beachtet, dass das System an dieser Stelle, ebenfalls durch die große Unsicherheit, auch eine große Erwartete Verschlechterung zu erwarten hat und es dadurch zu gefährlichen Situationen kommen kann. Auf diese Weise wird eine sichere Kombination aus der Exploration neuer Aktionen und der Anwendung bekannter Strategien bei der Suche nach Lösungen für eine Aufgabe erzeugt. Diese Kombination aus Erwarteter Verbesserung und Erwarteter Verschlechterung wird in dieser Arbeit mit dem Begriff der Erwartete Veränderung bezeichnet.

3.3 Gleichberechtigte Integration von Expertenwissen

Bestärkendes Lernen erlaubt es, autonom Strategien zum Lösen einer Aufgabe in verschiedenen Situation zu entwickeln und diese, aufbauend auf gesammelten Erfahrungen, selbstständig zu verbessern. Für praktische Anwendungen ist es dabei nötig, den oftmals sehr großen Suchraum möglicher Aktionen einzuschränken, um potenziell gefährliche Aktionen zu vermeiden und eine effiziente Suche nach vielversprechenden Aktionen zu ermöglichen. Dies kann durch Einbeziehung von Vorwissen über mögliche Lösungen, etwa durch einen Experten, erreicht werden. Zudem werden diese Informationen auch als Hilfestellung für eine informierte Suche nach vielversprechenden Aktionen genutzt, die eine gerichtete Explorationsstrategie im Rahmen des Bestärkenden Lernens ermöglichen. Eine solche Strategie, die aufbauend auf gegebenem Wissen nach vielversprechenden Aktionen sucht und gleichzeitig Aktionen mit unsicherem Ausgang vermeidet, wird in Abschnitt 3.2 vorgestellt.

Eine besonders attraktive Art, aufgabenspezifisches Wissen eines Experten zu integrieren, ist das Lernen aus Demonstrationen. Dabei handelt es sich um den Erwerb von Fähigkeiten durch Beobachtung eines Vorbildes und Nachahmung seiner Aktionen. Für den Experten hat diese Art des Wissenstransfers den Vorteil, dass es häufig einfacher ist, die Lösung einer Aufgabe anhand einer konkreten Situation vorzuführen, als sie in einem Computerprogramm oder auf eine andere maschinenlesbare Art und Weise zu formulieren. Darüber hinaus ist das Lernen durch Demonstration, Beobachtung und Nachahmung eine typische menschliche Art des Lernens. In der menschlichen Lernpsychologie wird diese auch als Lernen am Modell oder Beobachtungslernen [48] bezeichnet und ist eine wesentliche Eigenschaft, die es Menschen erlaubt, sich effizient komplexe Verhaltensweisen anzueignen. Das Demonstrieren von Fähigkeiten ist demnach nicht nur eine einfache, sondern auch eine für Menschen intuitive Art, Wissen zu vermitteln. Die Anlehnung maschineller Lernverfahren an menschliche Verhaltensweisen und die Modellierung der kognitiven Fähigkeiten des Menschen stellt somit aus Sicht der Informatik eine Möglichkeit dar, Mensch-Maschine-Schnittstellen benutzerfreundlich zu gestalten und an die effiziente Informationsverarbeitung des menschlichen Vorbildes anzuknüpfen.

Neben dem Lernen aus Demonstrationen stellt das autonome Lernen durch Ausprobieren von Aktionen und der Beobachtung ihrer Auswirkungen eine weitere von Menschen genutzte Art des Lernens dar, die essentiell zum Erwerb komplexer Fähigkeiten ist. Dieses sogenannte Lernen durch Versuch und Irrtum[5] ist unter verschiedenen Aspekten komplementär zum Lernen aus Demonstration. Da sich nur wenige Aspekte der Welt konkret beobachten lassen, sind Menschen darauf angewiesen, viele Fähigkeiten durch Ausprobieren zu erwerben oder durch Demonstrationen erlernte Fähigkeiten zu optimieren. Im Bereich des maschinellen Lernens wird die Art des Lernens, wie in Abschnitt 3.2 beschrieben, durch das Konzept des Bestärkenden Lernens formalisiert, das das Lernen anhand von Belohnungen der Umgebung, ausgelöst durch die eigenen Aktionen, mittels Algorithmen beschreibt. Auch hier ist es intuitiv für Menschen, ein System mittels Belohnungen zu trainieren.

In Anlehnung an die Art und Weise, mit der Menschen neue Fähigkeiten erwerben, indem sie in unbekannten Situationen Vorbilder beobachten oder, falls sie unsicher sind,

[5] engl.: trial and error

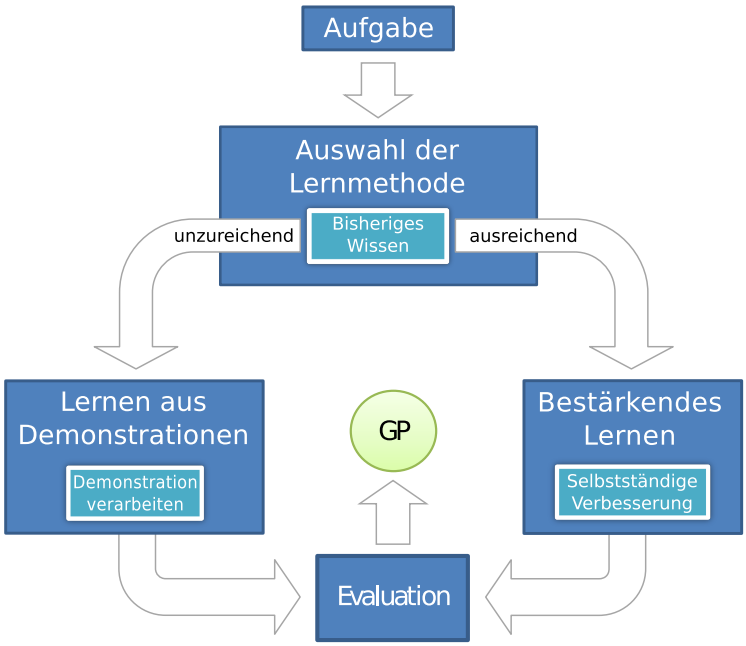


Abbildung 3.3: *Schematische Übersicht des vorgeschlagenen Systems. Zu einer gegebenen Aufgabenstellung entscheidet das System aufgrund des zur Verfügung stehenden Wissens, ob durch Bestärkendes Lernen eine sichere Lösung gefunden werden kann. Ist dies nicht möglich, fordert das System Hilfe in Form einer Demonstration durch einen Experten an. In beiden Fällen wird die Lösung anschließend evaluiert und als neues Beispiel für die Gauß-Prozess-Regression gespeichert.*

um eine Demonstration bitten, und die gemachten Beobachtungen anschließend nachahmen und anhand ihrer eigenen Erfahrungen zu verbessern versuchen, wird in dieser Vorarbeit ebenfalls ein kombinierter Lernansatz aus Lernen aus Demonstrationen und Bestärkendem Lernen verfolgt. Dies bietet den Vorteil, dass beim Bestärkenden Lernen nicht der gesamte Suchraum exploriert werden muss und ausgehend von demonstrierten Lösungen, die, insbesondere für den Nachahmer, nicht immer perfekt sein müssen, nach der optimalen Lösung gesucht werden kann. Während das Lernen aus Demonstrationen erlaubt, detailliertes Aufgabenwissen in geschlossener Form auf den Roboter zu übertragen und dadurch die Suche des hier vorgestellten Systems zu fokussieren, wird Bestärkendes Lernen genutzt, um die erworbenen Fähigkeiten des Systems zu verbessern und Lösungen für ähnliche Situationen zu generieren. Kinematische Unterschiede zwischen dem Experten, der eine Demonstration gibt, und dem Robotersystem, das die demonstrierte Aktion erlernen soll, können dazu führen, dass demonstrierte, für den Experten optimale Lösungen nicht optimal für den Roboter sind. Ausgehend von diesen Demonstrationen kann es Bestärkendes Lernen dem System ermöglichen, solche Unterschiede auszugleichen und die durch den Experten demonstrierten Lösungen anzupassen, um die Qualität der eigenen Lösungen zu steigern. Da die Szenarien, in denen das Lernen aus Demonstrationen und Bestärkendes Lernen aufgrund ihrer individuellen Stärken optimal eingesetzt werden können, sehr gegensätzlich zueinander sind, wurde

ein Entscheidungskriterium entwickelt, um beide Verfahren als alternative Kontrollflüsse in diesen Ansatz zu integrieren. Dieser Ansatz wird schematisch in Abbildung 3.3 dargestellt. Er steht im Gegensatz zu vielen verwandten Ansätzen [49], die ebenfalls mit einer Kombination aus Lernen aus Demonstrationen und Bestärkendem Lernen arbeiten, allerdings das Expertenwissen ausschließlich zur Initialisierung des Verfahrens nutzen und anschließend lediglich autonomes Lernen verwenden. Da das System nach Abschluss der Initialisierungsphase nicht mehr auf Expertenwissen zurückgreifen kann, erfordert diese Art von Ansätzen eine detaillierte Planung der Demonstrationsphase, in der die für den späteren Einsatz relevanten Situationen im Voraus berücksichtigt werden müssen und ein Kriterium für den Abschluss dieser Phase für den gesamten Zustandsraum gefunden werden muss. Der hier präsentierte Ansatz bietet stattdessen den Vorteil, dass in jeder Situation zwischen den beiden Lernverfahren gewählt wird und Expertenwissen nur in den Situationen genutzt wird, in denen tatsächlich Bedarf hierfür besteht. Durch das Lernen aus Demonstrationen steigt die Effizienz des Verfahrens, während gleichzeitig mit Hilfe des Bestärkenden Lernens die Anzahl an benötigten Demonstrationen gering gehalten wird. Dies ist ein wichtiges Kriterium für die Praxistauglichkeit des Verfahrens in einem realen Szenario, da das Demonstrieren von vielen Lösungen mit einem großen Zeitaufwand für den Experten verbunden ist und somit eine Belastung für ihn darstellt. Da beide Verfahren typische menschliche Lernverfahren sind, sind sie intuitiv bedienbar und der Mensch kann sich bei Training ähnlich zum Training mit einem anderen Menschen verhalten [50]. Dies erleichtert die Arbeit mit dem System und ermöglicht auch Anwendern ohne technische Ausbildung, auf eine einfache und für sie natürliche Art und Weise einen Roboter zu trainieren.

Die Auswahl des Lernverfahrens, welche individuell für jede Situation getroffen wird, beruht auf den zuvor gesammelten Erfahrungen über das Lösen der Aufgabe in dieser oder einer ähnlichen Situation. Ist das aus diesen Erfahrungen durch Gauß-Prozess-Regression abgeleitete Wissen ausreichend, so entscheidet sich das System, autonom eine Lösung zu generieren. Erscheint das Wissen als nicht ausreichend, da die potenziellen Aktionen zu hohe Risiken bergen, unvorhersehbare oder schlechte Ergebnisse zu erzeugen, so verlangt das System für die aktuelle Situation nach einer Demonstration des Experten. Dies ist zum Beispiel der Fall, wenn nie eine ähnliche Situation beobachtet worden ist oder alle evaluierten Lösungen zu der gegebenen Situation schlechte Ergebnisse geliefert haben. Der Algorithmus zur Einschätzung des vorhandenen Wissens sucht aus Sicherheitsgründen nur unter den positiven Erfahrungen nach geeigneten Aktionen zur Lösung der Aufgabe. Dies stellt zum einen sicher, dass nur Aktionen ausgewählt werden, die kein Risiko für den Roboter oder seine Umgebung darstellen, und zum anderen, dass diese Aktionen vielversprechend sind, da mit ihnen die Aufgabe zumindest für eine bestimmte Situation schon einmal erfolgreich gelöst wurde. Dabei betrachtet der Algorithmus eine Kombination aus dem mittels Gauß-Prozess-Regression bestimmten, zu erwartenden Q-Wert $\mu(s, a)$ eines solchen Paares aus Zustand und Aktion (s, a) und der Ähnlichkeit der darin enthaltenen Situation s zu der aktuell betrachteten Situation s_{curr}. Die nachfolgende Funktion bestimmt eine gute Kombination aus diesen beiden Werten:

$$\zeta = \min_{(s,a)\in\mathcal{X}} \left(\alpha \left(Q_{max} - \mu(s,a) \right) + (1-\alpha) \left\| s_{\text{curr}} - s \right\|_2 \right) \tag{3.13}$$

Das Trainingsbeispiel, das diese Funktion minimiert, wird mit $\hat{x} = (\hat{s}, \hat{a})$ bezeichnet

und $\mathcal{X} \subseteq \mathcal{S} \times \mathcal{A}$ stellt die Menge der positiven Trainingsbeispiele dar, die bis zu diesem Zeitpunkt schon getestet wurden. Der Wert Q_{max} ist definiert als der größtmögliche Q-Wert, der mit einer Aktion erreicht werden kann, und wird beim Design der Belohnungsfunktion festgelegt. Mit dem Gewichtungsfaktor α kann die Präferenz zwischen einem hohen Q-Wert oder einer großen Ähnlichkeit der verglichenen Situationen festgelegt werden.

Zur Entscheidung für eines der beiden Lernverfahren wird der Wert ζ mit einem Schwellwert θ verglichen. Unterschreitet er den Schwellwert, so liegt ein gutes Trainingsbeispiel mit einer hohen Ähnlichkeit zu der aktuellen Situation vor und die Entscheidung wird zugunsten des Bestärkenden Lernens getroffen. Die Aktion $\hat{a}$ dieses Trainingsbeispieles wird als Ausgangspunkt für die in Abschnitt 3.2 beschriebene Bayessche Explorationsstrategie verwendet. Der Q-Wert dieses Beispieles $\mu(\hat{s}, \hat{a})$ beschreibt den bis zu diesem Zeitpunkt besten gesehenen Q-Wert und stellt daher den Referenzwert $\overline{f}$ bei den weiteren Berechnungen dar. Wird θ überschritten, so ist in dieser Situation der Wissensstand des Systems nicht ausreichend, um selbstständig eine sichere Lösung zu generieren. In diesem Fall ist eine Demonstration durch einen Experten nötig. Liegen gar keine Erfahrungen vor, so fällt die Entscheidung ebenfalls auf das Lernen aus Demonstrationen.

Der Schwellwert θ steuert somit, wie risikoreich sich das System verhält. Umso größer θ, umso seltener benötigt das System Hilfe in Form von Demonstrationen. Dadurch erhöht sich allerdings auch das Risiko, dass durch das Bestärkende Lernen Aktionen ausgewählt werden, deren Ergebnis nur ungenau vorhersehbar ist und die somit auch das Risiko vergrößern, dass die Aufgabe fehlerhaft gelöst wird oder es zu Beschädigungen am Roboter oder seiner Umgebung kommt. Wird der Schwellwert θ gesenkt, so reduziert sich dieses Risiko. Allerdings steigt dadurch auch die Anzahl an Demonstrationen und somit der Arbeitsaufwand für den Experten.

4 Kapitel Lernen sequentieller Aufgaben

Das in Kapitel 3 vorgestellte System erlaubt es, Robotern einfache Fähigkeiten in Form von Bewegungsprimitiven beizubringen. Diese reichen jedoch nicht aus, um alltägliche Aufgabenstellungen zu lösen. Vielmehr müssen für praxisrelevante Aufgaben mehrere einfache Fähigkeiten kombiniert und zu komplexen Bewegungsabläufen zusammengesetzt werden. Selbst einfache Manipulationsaufgaben, wie das Bewegen eines Objektes von einer Ausgangsposition an eine Zielposition, erfordern die Kombination mehrerer Bewegungsprimitive. Somit sind Lernverfahren, die Roboter nur mit einer bestimmten Anzahl an Bewegungsprimitiven ausstatten und diese gegebenenfalls verbessern oder sie an verschiedene Situationen anpassen, nicht ausreichend, um Robotern die für reale Aufgaben benötigten Fähigkeiten beizubringen. Vielmehr müssen Roboter auch lernen, wie sie eine variable Anzahl Bewegungsprimitive kombiniert einsetzen können, um gewünschte Effekte auf verschiedene Objekte einer Umgebung zu erreichen.

In diesem Kapitel wird ein für Menschen intuitives Lernverfahren vorgestellt, das es Robotern ermöglicht, Sequenzen von Bewegungen zu erlernen. Dabei werden, wie in dem in Kapitel 3 beschriebenen Lernverfahren, Lernen aus Demonstrationen und Bestärkendes Lernen kombiniert. Dadurch profitiert auch das hier entwickelte Lernverfahren von den Vorteilen eines integrierten Ansatzes, der auf der einen Seite ein effizientes und sicheres Lernen ermöglicht und auf der anderen Seite den Arbeitsaufwand für den Experten gering hält. Zusätzlich wird das System um eine interaktive Komponente erweitert, bei der das System zunächst einen Lösungsvorschlag macht, ehe es eine Demonstration verlangt. Auf diese Weise wird der Aufwand für den Experten reduziert, da unnötige Demonstrationen vermieden werden. Beide Arten des Lernens, das Lernen aus Demonstrationen und das Bestärkende Lernen, werden in diesem Ansatz als gleichwertige, alternative Kontrollflüsse betrachtet. Das System wählt in jeder Episode eines der beiden Lernverfahren zum Lösen der Aufgabe. Dadurch kann das Lernverfahren die Stärken beider Ansätze optimal ausnutzen und ihre bekannten Schwächen vermeiden.

Beim Lernen von Bewegungssequenzen ergeben sich gegenüber dem in Kapitel 3 betrachteten Problem des Lernens einfacher Bewegungsprimitive zusätzliche Herausforderungen. Dazu gehören unter anderem die verschiedenen Repräsentationsarten des realen, oftmals kontinuierlichen Zustandsraumes und des diskretwertigen Raumes der Aktionssequenzen. Des Weiteren erfordern verschiedene Aufgabenstellungen oder auch nur das Lösen ein und derselben Aufgabe in unterschiedlichen Situationen individuelle Lösungsstrategien, die oft-

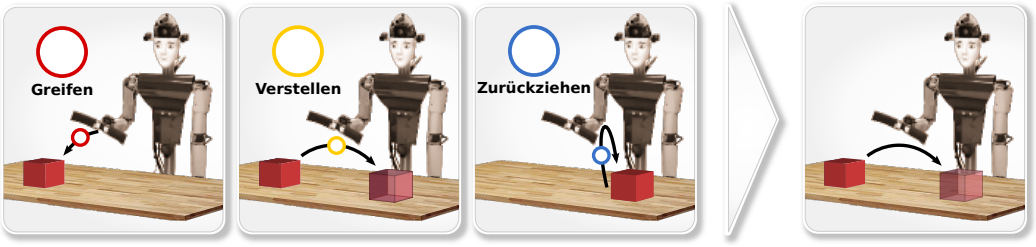


Abbildung 4.1: *Darstellung einer Aktionssequenz, die aus der Kombination von drei einzelnen Bewegungsprimitiven besteht, und des durch die Sequenz bewirkten Effekts, ein Objekt zu einer gewünschten Position zu bewegen. So können durch die Kombination einfacher Bewegungen komplexe Aufgaben gelöst werden.*

mals Aktionssequenzen unterschiedlicher Länge beinhalten. Um diesen Herausforderungen zu begegnen, wird zur Generalisierung der Nutzenfunktion über den kombinierten Raum der Zustände und Aktionssequenzen ein Gauß-Prozess-Modell mit einer speziellen, kombinierten Kernelfunktion verwendet, die in diesen Räumen eine sinnvolle Approximation ermöglicht. Das erlaubt es, den Wert der Erwarteten Veränderung für eine probabilistische Explorationsstrategie zu nutzen. Diese sorgt, einem Bayesschen Optimierungskriterium folgend, dafür, dass das System durch Auswahl vielversprechender Aktionssequenzen effizient gute Lösungen findet und gleichzeitig Sequenzen mit unvorhersagbarem Ausgang vermeidet.

Die Fähigkeit dieses Ansatzes, effizient geeignete Aktionssequenzen für verschiedene Situationen zu lernen, wird an einer Manipulationsaufgabe gezeigt, in der gestapelte Boxen mit verschiedenen Strategien an gewünschte Zielpositionen bewegt werden müssen. Der Ansatz in diesem Kapitel basiert auf der eigenen Arbeit [51], die auf der IROS 2013 vorgestellt wurde.

4.1 Einleitung

Viele Aufgaben, die Roboter Menschen in ihrem Alltag abnehmen könnten, basieren auf der Manipulation von Objekten. Um Effekte auf diese Objekte in der Umgebung des Roboters ausüben zu können, müssen lange Bewegungen in Form von Trajektorien ausgeführt werden. Eine häufige Annahme bei der Analyse dieser Art von Bewegungen ist, dass sich diese komplexen Aktionen in Sequenzen von elementaren und oftmals zielgerichteten Bewegungen zerlegen lassen. Dementsprechend gibt es eine erhebliche Menge an Arbeiten, die das Problem der Darstellung solcher Bewegungsprimitive untersucht haben, um zu erforschen, wie diese effizient auf den Roboter übertragen werden können [49]. Eine Möglichkeit für eine Zerlegung komplexer Aktionen in ihre Einzelbewegungen und zur Generierung angepasster Bewegungen für neue Kontexte wurde in Kapitel 2 vorgestellt. Allerdings ermöglicht alleine die Ausstattung mit einer Menge an Bewegungsprimitiven es dem Roboter nicht, komplexe Aufgaben selbstständig zu lösen. Vielmehr bedarf es zusätzlich der Fähigkeit, einzelne dieser Aktionen in der richtigen Reihenfolge zu kombinieren, um durch geschickte Manipulation von Objekten gewünschte Effekte auf

die Umgebung zu erreichen. Beispielhaft ist eine solche Sequenz mit dem dazugehörigen Effekt in Abbildung 4.1 dargestellt. Abhängig von der Situation, in der eine Aufgabe gelöst werden soll, kann es sein, dass verschiedene Bewegungsprimitive eingesetzt oder Bewegungen in unterschiedlicher Reihenfolge kombiniert werden müssen, um denselben Effekt zu erzielen.

In diesem Kapitel wird ein interaktives Lernverfahren vorgestellt, um Robotern beizubringen, komplexe Aufgaben anhand von Sequenzen aus Bewegungsprimitiven zu lösen. Dazu kombiniert dieses Verfahren Bestärkendes Lernen mit Lernen aus Demonstrationen. Dabei wird davon ausgegangen, dass das Wissen über den Q-Wert von Zuständen und Aktionssequenzen in der ausgewählten Repräsentation durch Gauß-Prozesse generalisiert werden kann. Unter dieser Annahme lässt sich mittels Gauß-Prozess-Regression [24] eine Schätzung, die den erwarteten Q-Wert einer Aktionssequenz in einer bestimmten Situation angibt, und deren Unsicherheit berechnen. Auf diese Weise können die Q-Werte über den Raum der Zustände und Aktionssequenzen, die in vorangegangenen Episoden beobachtet wurden, generalisiert werden. Dieses wiederum wird von dem hier vorgestellten Ansatz genutzt, um geeignete Aktionssequenzen für ähnliche Aufgabenstellungen, unter Bedingungen, die zuvor nicht beobachtet wurden, abzuleiten. Die Formulierung dazu stützt sich auf vorausgegangene Arbeiten, die in Kapitel 3 vorgestellt werden und sich mit dem Erlernen der Parameter einzelner Bewegungsprimitive beschäftigen. Das in diesem Kapitel vorgestellte Verfahren erweitert diese Arbeiten um das Erlernen von Aufgaben mit einem kombinierten kontinuierlichen und diskreten Parameterraum und variabel langen Bewegungsabläufen. Die Bewegungsabläufe bestehen dabei aus verschiedenen Kombinationen aus Bewegungsprimitiven. Die Verwendung von Gauß-Prozess-Regression mit komplexen Aktionssequenzen führt dabei zu einer Reihe neuer Herausforderungen. Dazu gehört die Notwendigkeit, kontinuierliche Zustandsräume mit dem von Natur aus diskreten Raum der Aktionssequenzen zu vereinigen. Eine weitere Herausforderung besteht darin, dass Aktionssequenzen je nach Aufgabe und Situation eine unterschiedliche Anzahl Elemente enthalten können. In diesem Kapitel wird eine Kombination von aufgabenspezifischen Kerneln betrachtet, um die verschiedenen Repräsentationen der Zustände und Aktionssequenzen sowie Sequenzen variabler Länge im Gauß-Prozess-Ansatz zu berücksichtigen.

In dem hier vorgestellten System werden das Lernen aus Demonstrationen und das Lernen durch autonome Verbesserung als zwei gleichwertige Kontrollflüsse integriert. Im Gegensatz zu vielen bestehenden Ansätzen, die Demonstrationen eines Experten nur während einer Initialisierungsphase für das Bestärkende Lernen benutzen [49], entscheidet das Verfahren, basierend auf dem vorhandenen Wissen, in jeder Situation neu, welche Methode angewendet werden soll. Dadurch ergibt sich der Vorteil, dass sich die Stärken beider Methoden gegenseitig ergänzen. Des Weiteren erhöht diese Kombination die Flexibilität des Systems, denn im Fall neuer Aufgaben oder veränderter Rahmenbedingungen erhält das System weiterhin die Möglichkeit, Expertenwissen anzufragen. Dies vereinfacht die Anpassung an diese Arten von Veränderungen. Gleichzeitig bedeutet die Demonstration von komplexen Bewegungssequenzen für den Experten einen Mehraufwand gegenüber der Demonstration einzelner Bewegungsprimitive. Aus diesem Grund wird das System um eine weitere interaktive Komponente ergänzt, in der das System bei unzureichendem Hintergrundwissen nicht direkt eine Expertendemonstration anfordert, sondern dem

Experten zunächst die ihm am besten erscheinende Aktionssequenz vorschlägt. Dieser Vorschlag wird von dem Experten im nächsten Schritt geprüft. Erst wenn der Experte den Vorschlag ablehnt, wird er vom System aufgefordert, die Lösung der Aufgabe zu demonstrieren. Auf diese Weise wird der Experte entlastet. Im umgekehrten Fall, wenn genügend gute Informationen über die aktuelle oder vergleichbare Situationen verfügbar sind, generiert das System autonom eine Lösung für die gegebene Aufgabe und entlastet auf diese Weise ebenfalls den Experten. Für die autonome Suche nach geeigneten Lösungen wird ein auf der Gauß-Prozess-Approximation basierendes Bayessches Optimierungskriterium verwendet, mit dessen Hilfe das System versucht, bekannte Lösungsstrategien zu verbessern oder sie an neue Situationen anzupassen.

Damit die gesuchte Strategie schnell gegen gute Aktionssequenzen für jede gegebene Situation konvergiert, wird eine auf der Optimierung der Erwarteten Veränderung basierende Explorationsstrategie genutzt. Durch eine Gewichtung der Erwarteten Verbesserung mit ihrem Gegenstück, der Erwarteten Verschlechterung, führt dieses Kriterium zu einem Optimierungsprozess, der unbekannte Bereiche des Aktionsraumes vermeidet und potenziell unsichere Aktionssequenzen umgeht. Sicherzustellen, dass Aktionssequenzen weder für den Roboter noch für seine Umgebung ein Risiko bergen, ist von besonderer Bedeutung für praktische Anwendungen, in denen Roboter Aufgaben in Zusammenarbeit mit Menschen lösen. In dieser Art von Anwendungen ist darüber hinaus die Anwendung von Lernprinzipien, die aus der menschlichen Verhaltensforschung bekannt sind, ein vielversprechender Ansatz, um die Zugänglichkeit von technischen Lernsystemen zu verbessern und somit eine größere Akzeptanz bei den potenziellen Nutzern zu schaffen.

4.1.1 Verwandte Arbeiten

In den letzten Jahren wurde viel Arbeit der Frage gewidmet, wie ein Roboter mit einfachen Fähigkeiten, repräsentiert durch Bewegungsprimitive, ausgestattet werden kann [52]. Diese Ergebnisse haben die Forschung motiviert, über die offene Frage nachzudenken, wie darauf aufbauend Robotern beigebracht werden kann, Sequenzen aus solchen Bewegungsprimitiven zusammenzusetzen, um damit komplexe Aufgaben zu lösen.

Eine Forschungsrichtung formuliert dabei das Lernproblem als Planungsaufgabe. Toussaint et al. [53] schlagen einen Ansatz vor, der auf einem Modell der Umgebung und der Effekte von Aktionen auf diese basiert. Dabei übersetzen sie relationale Regeln, die die Effekte der Aktionen erfassen, in ein Bayes-Netzwerk. Dies ermöglicht es, Aussagen über zukünftige Zustände und Belohnungen zu treffen. Basierend auf dieser Repräsentation entwickeln die Autoren einen probabilistischen Planungsalgorithmus, der inkrementell Aktionen basierend auf den vorhergesagten Belohnungssequenzen auswählt. Sobald auf der aufgabenbeschreibenden Ebene eine Aktion ermittelt wurde, werden mittels stochastischer Optimierung die Motorkommandos für die entsprechende Bewegung ermittelt und ausgeführt. In dem Ansatz von Abdo et al. [54] werden Vorbedingungen und Effekte von Aktionen aus demonstrierten Bewegungen eines Experten gelernt. Diese werden als Eingabe für einen Planungsalgorithmus verwendet, mit dessen Hilfe ein Roboter in die Lage versetzt wird, Sequenzen von Manipulationsbewegungen zu generieren. Unter der Annahme, dass eine geeignete Segmentierung der demonstrierten Bewegungen vorliegt, analysieren die Autoren die Varianz in den Zuständen an beiden Enden der demonstrierten

Segmente, um die Vorbedingungen und die Effekte von Aktionen zu identifizieren. Eine Generalisierung wird durch die Analyse von mehreren Demonstrationen derselben Aufgabe erreicht oder interaktiv durch Anfragen an den Trainer, Vorbedingungen aufzulockern, indem er auf irrelevante Aufgabenmerkmale aufmerksam macht.

Eine bei planungsbasierten Ansätzen naturgemäße Herausforderung ist es, eine geeignete symbolische Repräsentation der Umgebung zu finden. Mehrere Ansätze umgehen die Wahl einer symbolischen Repräsentation, indem sie das Problem aus der Perspektive des Bestärkenden Lernens betrachten. Daniel et al. [55] entwickelten einen Ansatz, um simultan sowohl die Parameter von Bewegungsprimitiven zu lernen als auch die Reihenfolge, in der diese angewandt werden müssen, um eine Aufgabe zu erfüllen. Dieser baut auf einem von ihnen zuvor entwickelten Ansatz mit dem Namen Hierarchical Relative Entropy Policy Search [56] auf, mit dem in episodischen Aufgaben sowohl die Wahl einer auszuführenden Aktion als auch deren Parameter gelernt werden können. Um eine sequentielle Wahl mehrerer Bewegungen und gleichzeitig die Parameter der zugehörigen Motorprimitive zu lernen, erweitern die Autoren diesen Ansatz in [55] auf Aufgabenstellungen mit endlichem Horizont, so dass durch den Algorithmus in jedem Schritt einer Episode eine primitive Bewegung gewählt wird. Die einzelnen Bewegungsprimitive werden dabei durch eine spezielle Form dynamischer Systeme parametrisiert. Für eine festgelegte Anzahl an Segmenten kann damit das Problem, Sequenzen zu lernen, als ein Optimierungsproblem mit Nebenbedingungen formuliert werden. Mit diesem Ansatz bringen die Autoren einem Roboterarm innerhalb von 300 Iterationen eine Strategie für ein spezielles Hockeyspiel für Roboter bei.

Um schneller eine gute Strategie zu finden, verwenden Stulp et al. [57] bei der Optimierung der Formparameter der Bewegungsprimitive eine demonstrierte Trajektorie zur Initialisierung. In ihrem Ansatz nutzen sie den PI^2-Algorithmus [58] zum Lernen von Sequenzen von Bewegungsprimitiven. Dabei erweitern sie diesen um, über das Lernen von Formparametern hinaus, gleichzeitig die Ziele der Bewegungen, die die Endpunkte einer Trajektorie beschreiben, zu optimieren. Diesen neu entwickelten Ansatz bezeichnen die Autoren mit dem Namen PI^2SEQ. Beide Ansätze sind auf eine vordefinierte Anzahl an Bewegungsprimitiven limitiert, was den Spielraum des Roboters bei der Suche nach optimalen Aktionssequenzen zum Lösen einer Aufgabe einschränkt.

Neumann et al. [59] schlagen eine Methode vor, die ausschließlich mit Bestärkendem Lernen arbeitet, um die Reihenfolge und die Parameter von Aktionen zu lernen, deren Ausführung sich über mehrere Zeitschritte erstreckt. Dabei modellieren die Autoren diese Aktionen analog zu den sogenannten Options [60], wobei sie annehmen, dass die dazugehörige Strategie eine parametrische Form aufweist. Somit erlaubt der Ansatz, zeitliche Abstraktionen auch in kontinuierlichen Umgebungen einzusetzen. Wird eine Aktion gewählt, wird diese so lange ausgeführt, bis eine spezielle Endbedingung erfüllt ist. Um die kontinuierlichen Parameter dieser Aktionen sowie eine Strategie zur Auswahl von Aktionen zu lernen, nutzen die Autoren einen Q-Lernalgorithmus, der mit einer iterativen Approximation der Q-Funktion arbeitet und eine Suche im kontinuierlichen Aktionsraum durch eine Approximation der Zustands-Nutzenfunktion vermeidet.

Um Sequenzen von Aktionen zu lernen, schlagen verschiedene Autoren auch graphbasierte Repräsentation auf Aufgabenebene in Verbindung mit Bestärkendem Lernen und Imitationslernmethoden vor. Konidaris et al. [61] kodieren generalisierte Aktionssequenzen

in einer Baumstruktur, die sie als Fähigkeitsbäume bezeichnen. Sequenzen können entweder durch das Demonstrieren geeigneter Lösungen oder durch exploratives Bestärkendes Lernen trainiert werden. Mehrere Sequenzen, die das gleiche Ziel von verschiedenen Ausgangssituationen aus erreichen, werden zu einem Baum zusammengefasst. Dies geschieht, indem ihre statistischen Gemeinsamkeiten ausgehend vom Zielzustand in umgekehrter Reihenfolge ausgewertet werden. Die Autoren weisen in ihren Experimenten darauf hin, dass die Verfügbarkeit von Demonstrationen den Lernprozess stark beschleunigen kann. Ähnlich wie in dem hier vorgestellten Ansatz ist durch die gewählte Repräsentation zum Lösen einer Aufgabe keine Planung zur Laufzeit erforderlich. Im Gegensatz zu diesem Ansatz wird in der vorliegenden Arbeit allerdings ein interaktiver Ansatz vorgeschlagen, der Bestärkendes Lernen mit dem Lernen aus Demonstrationen auf der Aufgabenebene mit Hilfe eines Entscheidungskriteriums zwischen den beiden Verfahren kombiniert. Kulić et al. [16] stellen einen Ansatz vor, in dem verschiedene Bewegungssequenzen als alternative Pfade in einem Bewegungsgraphen kodiert werden. Demonstrierte Trajektorien werden dazu segmentiert und hierarchisch zu Bewegungsprimitiven zusammengefasst, welche anschließend in Hidden Markov Modellen kodiert werden, um die Zuordnung und die Generierung neuer Bewegungen zu ermöglichen. Die Knoten des Graphen entsprechen den Aktionen und deren Kanten sind mit den Übergangswahrscheinlichkeiten markiert, die aus den beobachteten Daten gelernt werden. Eine autonome Verbesserung, der aus Demonstrationen gewonnenen Bewegungsprimitiven ist nicht Teil ihres Ansatzes.

Die hier präsentierte Arbeit steht im Einklang mit den modellfreien Ansätzen, die zuvor skizziert wurden und in denen weder eine umfassende Planung zur Laufzeit noch die Verfügbarkeit von dynamischen Modellen des Roboters benötigt werden. Stattdessen wird vorgeschlagen, eine probabilistische Approximation der Q-Funktion für Paare aus Zuständen und Aktionssequenzen zu lernen und Lernen aus Demonstrationen mit Bestärkendem Lernen zu kombinieren, um so den Experten nur einzubeziehen, wenn dies erforderlich ist. Ein Schwerpunkt dieser Arbeit liegt auf der Sicherheit sowohl in der Interaktion zwischen einem Roboter und einem Menschen als auch in der Interaktion eines Roboters mit seiner Umgebung. Zu diesem Zweck wird eine effiziente, aber trotzdem sichere Optimierungsstrategie präsentiert, um während des Bestärkenden Lernens vielversprechende Aktionssequenzen auszuwählen.

4.1.2 Wissenschaftlicher Beitrag

In diesem Kapitel wird ein Lernverfahren vorgestellt, mit dem Sequenzen von Bewegungsprimitiven zum Lösen komplexer Aufgaben gelernt werden können. Aufgebaut ist das Lernverfahren auf dem in Kapitel 3 vorgestellten Ansatz zum Erlernen einzelner Bewegungsprimitiven. Dabei wird auch zum Lernen von Bewegungssequenzen eine Gauß-Prozess-Approximation der Q-Funktion zur Abschätzung der zu erwartenden Langzeitbelohnung für unbekannte Paare aus Zuständen und Aktionssequenzen verwendet. Durch die neue Aufgabenstellung des Lernens von Aktionssequenzen ergeben sich hierbei neue Herausforderungen gegenüber den in Kapitel 3 beschriebenen Anwendungen für Bewegungsprimitive. So beinhaltet der kombinierte Raum der Zustände und Aktionssequenzen sowohl reellwertige Komponenten, die beispielsweise den Zustand der Umgebung beschreiben, als auch kategorische Komponenten, die die Aktionsklassen der

auszuführenden Aktionen identifizieren und deren Parameter beschreiben. Des Weiteren können Aktionssequenzen unterschiedliche Längen aufweisen, da sie unterschiedliche Aufgaben oder unterschiedliche Situationen bearbeiten. Ein zentraler Beitrag dieses Kapitels ist daher eine Technik, die eine Approximation der Q-Funktion über den beschriebenen gemischten Eingaberäumen und über Aktionssequenzen variabler Länge mittels Gauß-Prozessen ermöglicht. Dazu wird ein aufgabenspezifischer kombinierter Kernel eingesetzt.

Basierend auf dieser Approximation wird, analog zu Kapitel 3, ein iteratives Lernverfahren entwickelt, in dem Bestärkendes Lernen und Lernen aus Demonstrationen als alternative Kontrollflüsse gleichberechtigt integriert werden. Im Unterschied zu Kapitel 3 wird zum Lernen von Aktionssequenzen die Entscheidung für ein Lernverfahren dahingehend erweitert, dass das System zunächst die ihm anhand des aktuellen Wissensstandes am besten erscheinende Aktionssequenz vorschlägt, falls eine Hilfestellung durch den Experten benötigt wird. Somit wird in unnötigen Fällen das aufwändige Demonstrieren von Aktionssequenzen vermieden. Lehnt der Experte den Vorschlag des Systems ab, so wird die demonstrierte Bewegung mittels des in Kapitel 2 vorgestellten Ansatzes in eine Reihe von Bewegungssegmenten unterteilt und daraus eine Aktionssequenz generiert. Ist hingegen genügend Wissen in der aktuellen Situation vorhanden, so wählt das System im Bestärkenden Lernen eine vielversprechende und sichere Aktionssequenz anhand des Kriteriums der Erwarteten Veränderung aus, welche eine Kombination aus dem Wert der Erwarteten Verbesserung und der Erwarteten Verschlechterung darstellt.

Das Kapitel ist wie folgt aufgebaut: In Abschnitt 4.2 wird das kombinierte Verfahren zum Lernen von kompletten Bewegungssequenzen zum Lösen komplexer Aufgaben beschrieben. Dazu werden zunächst (i) der kombinierte Kernel zum Umgang mit unterschiedlichen Repräsentationen im Raum der Zustände und Aktionssequenzen wie auch mit unterschiedlich langen Sequenzen und (ii) die um eine interaktive Komponente erweiterte Entscheidungsmethode, die bestimmt, in welcher Situation ein durch das System generierter Vorschlag, Lernen aus Demonstrationen oder Bestärkendes Lernen benutzt werden soll, vorgestellt. Zum Abschluss werden die beiden unterschiedlichen für das Lernen von Bewegungssequenzen verwendeten Lernverfahren, (iii) Lernen aus Demonstrationen und (iv) Bestärkendes Lernen, beschrieben. In Abschnitt 4.3 wird das kombinierte Lernverfahren in einer Manipulationsaufgabe, in der verschiedene Stapel von Boxen versetzt werden sollen, evaluiert. Nach der Beschreibung des Versuchsaufbaus und der zu lösenden Aufgabe mit ihren Herausforderungen wird der Ansatz zunächst auf nur einer Teilaufgabe untersucht, in der nur eine einzelne Box vorhanden ist, um eine bessere Visualisierung der verschiedenen Aspekte des Verfahrens zu ermöglichen. Anschließend werden die Vorteile des Ansatzes an der kompletten Aufgabe mit verschieden hohen Stapeln von bis zu drei Boxen gezeigt. Abschnitt 4.4 fasst abschließend den Ansatz zusammen und diskutiert die Ergebnisse.

4.2 Integriertes System zum Lernen sequentieller Aufgaben

Dieses Kapitel beschäftigt sich mit der Aufgabe, einem Roboter mit einer Kombination aus Lernen aus Demonstrationen und Bestärkendem Lernen beizubringen, wie bekannte

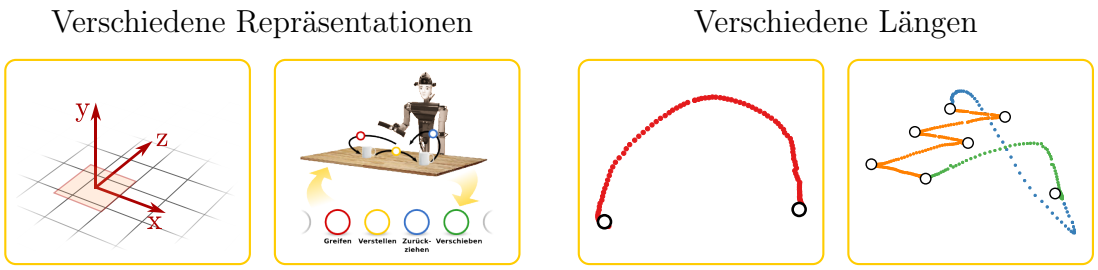


Abbildung 4.2: *Die verschiedenen Herausforderungen, die beim Lernen von Aktionssequenzen zusätzlich auftreten. Zum einen entstehen diese durch die Verwendung von verschiedenen Repräsentationen, die für die Kombination von Parametern aus der realen Welt und Parametern zur Beschreibung diskreter Aktionen benötigt werden. Zum anderen stellt das Lernen von verschieden langen Aktionssequenzen, die zum Lösen der Aufgabe in verschiedenen Situationen nötig sind, eine weitere Herausforderung dar.*

Bewegungsprimitive zu einer Aktionssequenz kombiniert werden können, um komplexe Aufgaben zu lösen. In dem hier präsentierten Ansatz wird die Q-Funktion auf dem kombinierten Raum der Zustände und Aktionssequenzen durch einen Gauß-Prozess approximiert, um die beobachteten Q-Werte auf ähnliche Situationen und Aktionssequenzen generalisieren zu können. Basierend auf dieser Information entscheidet das System in jeder Situation, ob eine sichere und vielversprechende Aktionssequenz autonom generiert werden kann oder ob die Hilfe eines Experten nötig ist. Diese Hilfestellung gibt der Experte, indem er einen vom System erstellten Lösungsvorschlag bewertet und diesem entweder bestätigt, dass der Vorschlag eine verlässliche Lösung darstellt oder, falls der Vorschlag dem Experten nicht geeignet erscheint, eine Demonstration zum Lösen der Aufgabe vorführt. Das System lernt anhand dieser Demonstrationen Aktionssequenzen, die aus demonstrierten Bewegungen segmentiert und klassifiziert werden, um eine symbolische Repräsentation der Aktionssequenz zu erhalten. Zur Generalisierung der Aktionssequenzen wird Gauß-Prozess-Regression eingesetzt. Dabei wird für diese Aufgabe ein speziell an die Herausforderungen, die beim Lernen variabel langer symbolischer Aktionssequenzen in Kombination mit einem reellwertigen Zustandsraum entstehen, kombinierter Kernel entwickelt. Das Bestärkende Lernverfahren basiert in diesem Ansatz ebenfalls auf Gauß-Prozess-Approximation und nutzt ein Bayessches Optimierungskriterium, das den Wert der Erwarteten Veränderung maximiert, um effizient sowohl sichere als auch vielversprechende Aktionssequenzen zu finden.

4.2.1 Kombinierter Kernel

In dem hier vorgestellten Ansatz wird Gauß-Prozess-Regression verwendet, um die Q-Funktion über den kombinierten Raum der Zustände und Aktionen zu approximieren. Dies macht es möglich, probabilistische Schätzungen der Q-Funktion für beliebige Paare aus Zustand und Aktion zu bestimmen und so die Q-Werte zuvor beobachteter Beispiele

auf neue Situationen oder Aktionen zu generalisieren.

Dieser Ansatz erlaubt die Formulierung eines effizienten Lernverfahrens, das die Vorteile des Lernens aus Demonstrationen mit denen des Bestärkenden Lernens verbindet. Diese Art des Lernens wird in dem in Kapitel 3 vorgestellten Verfahren erfolgreich für das Lernen einzelner Bewegungsprimitive eingesetzt. Dort werden sowohl Situationen als auch Aktionen durch Vektoren des $\mathbb{R}^n$ beschrieben. Im Rahmen des Lernens von Aktionssequenzen ergibt sich allerdings die Herausforderung, dass sich Aktionssequenzen und die für sie relevanten Parameter, im Gegensatz zu den in Kapitel 3 betrachteten Parametern von Bewegungsprimitiven, nicht sinnvoll durch Elemente des $\mathbb{R}^n$ beschreiben lassen (siehe Abbildung 4.2). Dies liegt zum einen daran, dass die einzelnen Elemente einer Aktionssequenz aus den diskreten endlichen Mengen der möglichen Aktionen stammen und ihre Parameter, etwa die Zuordnung der Start- und Endpunkte der Aktionen, durch aufgabenspezifische, ebenfalls diskrete Referenzpunkte beschrieben werden. Zum anderen besitzen die Aktionssequenzen, die zum Lösen komplexer Aufgaben nötig sind, eine variable Länge, da die Anzahl der benötigten Schritte und ihre Reihenfolge sowohl von der Komplexität der Aufgabe als auch von der Situation abhängen, in der die Aufgabe gelöst werden soll. Wird beispielsweise die Aufgabe betrachtet, einen Tisch zu decken, so sind dafür erheblich mehr Schritte notwendig als zum Einschenken eines Glases Wasser. Letztere Aufgabe kann sich allerdings verkomplizieren, wenn sowohl das Wasser als auch das Glas zunächst geholt werden müssen oder die Flasche geöffnet werden muss. Dies führt zu Zustandsräumen, die reellwertige Variablen beinhalten können, und zu Aktionsräumen, deren Elemente Sequenzen variabler Länge mit kategorischen Elementen sind. Zur Generalisierung der Q-Funktion für Aktionssequenzen mit Hilfe von Gauß-Prozess-Regression muss eine Möglichkeit gefunden werden, die Q-Werte über diese heterogenen Räume sinnvoll durch einen Gauß-Prozess zu approximieren.

Obwohl Gauß-Prozess-Regression oftmals zur Approximation von Funktionen über Euklidische Räume eingesetzt wird, ist einer der wichtigsten Vorteile dieses Verfahrens und ganz allgemein von Kernelmethoden, dass sie auf eine Vielzahl von Repräsentationen angewendet werden können. Die einzige Voraussetzung dazu ist, dass eine gültige positiv semidefinite Kovarianz- beziehungsweise Kernelfunktion $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ über dem Eingaberaum $\mathcal{X}$ definiert werden kann. Bei der Gauß-Prozess-Regression ist der Wert der Kernelfunktion ausschlaggebend dafür, wie sehr Wissen über den wahren Funktionswert an der Stelle $x \in \mathcal{X}$ die Schätzung des Wertes an einer anderen Stelle $x' \in \mathcal{X}$ beeinflusst. Es ist daher sinnvoll, die Kernelfunktion so zu konstruieren, dass ihr Wert die Ähnlichkeit ihrer Eingaben widerspiegelt. Der Begriff der Ähnlichkeit ist dabei abhängig von der Struktur der Eingaben und der konkreten Anwendung.

In diesem Abschnitt wird davon ausgegangen, dass Aktionssequenzen aus einer Folge von Identifikationsnummern der Klassen der Bewegungsprimitive und den dazugehörigen Parametern in Form von Referenzpunkten bestehen. Dagegen werden Zustände, die die Umgebung des Roboters beschreiben, zumeist durch euklidische Vektoren dargestellt, wie zum Beispiel durch Positionen von Objekten im Raum. Diese unterschiedlichen Bestandteile lassen sich nicht sinnvoll anhand eines gemeinsamen Ähnlichkeitsmaßes vergleichen. In diesem Ansatz wird daher ein neuer Kernel durch die Kombination verschiedener Kernel definiert, die jeweils auf einen Teil des Eingaberaumes zugeschnitten sind.

Aufgrund seiner euklidischen Repräsentation wird der Zustandsraum mit Hilfe eines Gauß-Kernels approximiert. Dieser ist für Aktionssequenzen nicht geeignet, da der euklidische Abstand der Identifikationsnummern, die die Aktionen der Sequenzen und ihre Parameter beschreiben, keine Aussage über deren Ähnlichkeit trifft. Vielmehr weisen die Folgen der Bewegungsprimitive und die ihrer Parameter strukturelle Ähnlichkeit zu Zeichenketten auf. Genau wie Identifikationsnummern von Bewegungsprimitiven und ihrer Parameter entstammen die Symbole, aus denen Zeichenketten bestehen, endlichen Mengen, die mit einer diskreten Metrik ausgestattet sind:

$$d_{\text{diskret}}(x, y) = \begin{cases} 0 & \text{falls } x = y, \\ 1 & \text{sonst.} \end{cases} \tag{4.1}$$

Unter dieser Metrik ist jedes Symbol beziehungsweise jede Aktion nur ähnlich zu sich selbst und gleichermaßen unähnlich zu allen anderen.

Zur Klassifikation von Text schlägt Lodhi [62] einen Kernel über Teilfolgen von Zeichenketten[1] vor. Dabei wird die Ähnlichkeit zweier Zeichenfolgen berechnet, ohne explizit Merkmale zu extrahieren. Stattdessen zeigt das Maß eine umso höhere Ähnlichkeit an, umso mehr Teilketten die beiden zu vergleichenden Zeichenfolgen gemeinsam haben. Unter Berücksichtigung aller geordneten Teilsequenzen einer Zeichenfolge führt dieses Maß zu einer elastischen Distanz, die Reihenfolgen beachtet und Eingaben verschiedener Längen akzeptiert. Die betrachteten Teilsequenzen müssen dabei nicht unbedingt zusammenhängend sein, so dass der Kernel auch robust gegenüber Unähnlichkeiten aufgrund von Lücken in den Eingabedaten ist.

Aufgrund der strukturellen Ähnlichkeit mit dem Raum der Zeichenketten, stellt der Teilfolgenkernel ein geeignetes Ähnlichkeitsmaß für den in dieser Arbeit verwendeten Aktionsraum dar und erlaubt es, mittels Gauß-Prozess-Regression die Q-Funktion über verschiedene Aktionssequenzen zu generalisieren.

Der Wert des Teilfolgenkernels für zwei Aktionssequenzen x und x' über dem Aktionsraum $\mathcal{A}$ ist definiert als die Summe über alle möglichen Teilsequenzen. Bei den Summanden handelt es sich dabei um innere Produkte, die gemeinsame Teilsequenzen anhand ihrer Häufigkeit und der Länge, mit der sie in den Eingaben vorkommen, gewichten. Die Werte des Kernels werden somit wie folgt bestimmt:

$$\begin{aligned} k'_{\text{str}}(x, x') &= \sum_{u \in \mathcal{A}^*} \phi_u(x)\phi_u(x'), \\ \phi_u(x) &= \sum_{\mathbf{i}: u = x[\mathbf{i}]} \lambda^{l(\mathbf{i})} \end{aligned} \tag{4.2}$$

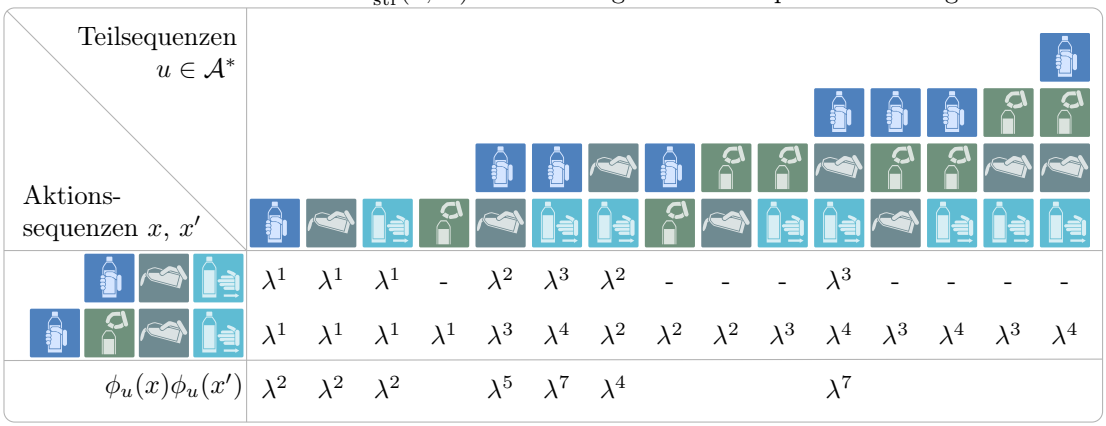
Dabei wird $\mathcal{A}^*$ als die Menge aller Sequenzen von Aktionen aus $\mathcal{A}$ bezeichnet. Die Abbildung ϕ misst die Anzahl der Vorkommen einer Teilsequenz $u \in \mathcal{A}^*$ in der Aktionssequenz x. Jedes Vorkommen wird entsprechend seiner Länge mit einem Abklingfaktor $\lambda \in (0, 1]$ gewichtet. Die Länge beschreibt unter anderem den Grad des Zusammenhangs der Teilsequenzen in der Eingabesequenz. Der Faktor λ bestraft somit Lücken abhängig von ihrer Länge. Wird beispielsweise die Aufgabe betrachtet, einen Gast mit einem Getränk zu bewirten, so unterscheiden sich die dazu notwendigen Aktionssequenzen je nachdem, ob

[1] engl.: String Subsequence Kernel (SSK)

Beispiel

Gegeben seien die folgenden Bewegungssequenzen x und x':

Dann berechnet sich der Wert $k'_{\text{str}}(x, x')$ aus den möglichen Teilsequenzen wie folgt:



$$k'_{\text{str}}(x, x') = 3\lambda^2 + \lambda^4 + \lambda^5 + 2\lambda^7$$

Abbildung 4.3: *Beispiel zur Berechnung des Teilfolgenkernels für die beiden Aktionssequenzen x:* `Greifen` $\rightarrow$ `Einschenken` $\rightarrow$ `Abstellen` *und x':* `Greifen` $\rightarrow$ `Öffnen` $\rightarrow$ `Einschenken` $\rightarrow$ `Abstellen`*. Dabei werden in der hier gezeigten Tabelle nur diejenigen Teilsequenzen $u \in \mathcal{A}^*$ betrachtet, die in mindestens einer der beiden zu vergleichenden Sequenzen vorkommen. Jede vorkommende Teilsequenz wird durch den Abklingfaktor λ entsprechend ihrer Länge in x beziehungsweise x' gewichtet. Die nicht vorhandenen Teilsequenzen haben keinen Einfluss auf das Ergebnis des Teilfolgenkernels.*

sich das Getränk in einer Flasche oder in einem Krug befindet, da eine Flasche zunächst geöffnet werden muss. So kommt die Teilsequenz

`Greifen` $\rightarrow$ `Einschenken`

in den folgenden beiden Sequenzen vor:

`Greifen` $\rightarrow$ `Einschenken` $\rightarrow$ `Abstellen`
`Greifen` $\rightarrow$ `Öffnen` $\rightarrow$ `Einschenken` $\rightarrow$ `Abstellen`

Allerdings unterscheiden sich die Gewichtungen dieser Vorkommen, da die gesuchte Teilsequenz in der zweiten Sequenz mit einer Lücke vorkommt, wohingegen sie einen zusammenhängenden Teil der ersten Sequenz ohne Lücke darstellt.

Die Schreibweise $x[\mathbf{i}]$ bezeichnet die Folge von Elementen von x, welche durch eine Folge $\mathbf{i}$ der Länge $|u|$ monoton steigender Indizes aus $[1, \ldots, |x|]$ gebildet wird. Die

Menge $\mathbf{i} : u = x[\mathbf{i}]$ umfasst somit alle Indexfolgen $\mathbf{i}$ für die gilt, dass die durch sie gebildete Teilfolge von x identisch zu u ist. Die Länge $l(\mathbf{i})$ einer solchen Indexfolge berechnet sich aus der Differenz des ersten und letzten Elements zu $l(\mathbf{i}) = \mathbf{i}_{|u|} - \mathbf{i}_1 + 1$. Um eine systematische Verzerrung der Kernelwerte aufgrund unterschiedlich langer Eingabesequenzen zu vermeiden, wird der Kernel normiert zu

$$k_{\text{str}}(x, x') = \frac{k'_{\text{str}}(x, x')}{\sqrt{k'_{\text{str}}(x, x)\; k'_{\text{str}}(x', x')}}. \tag{4.3}$$

Abbildung 4.3 zeigt beispielhaft die Berechnung des Teilfolgenkernels für zwei Aktionssequenzen anhand der Formeln (4.3).

Um einen gemeinsamen Kernel zur Generalisierung der Q-Funktion über den kombinierten Raum der Zustände und Aktionssequenzen zu erhalten, wird ein kombinierter Kernel aus dem Produkt des für den Zustandsraum verwendeten Gauß-Kernels sowie des im Aktionsraum genutzten Teilfolgenkernels konstruiert. Diese Kombination bildet, wie in [24] gezeigt wird, wieder einen gültigen Kernel und kann daher zur Definition eines Gauß-Prozesses verwendet werden:

$$\begin{aligned} k(x, x') &= k_{\text{rbf}}(s, s') \cdot \hat{k}_{\text{str}}(a, a'), \\ \hat{k}_{\text{str}}(a, a') &= k_{\text{str}}(a_{1:p-1}, a'_{1:q-1}) \cdot k_{\text{str}}(a_{p:n}, a'_{q:m}) \end{aligned} \tag{4.4}$$

Hier bezeichnet $x = (s, a)$ ein Paar aus Zustand und Aktionssequenz, m und n beschreiben die Längen der Aktionssequenzen a und a', welche nicht notwendigerweise gleich lang sein müssen. Die Indizes $p \in [1, n]$ und $q \in [1, m]$ werden benutzt, um die Eingaben a und a' in eine Sequenz von Identifikationsnummern von Bewegungsprimitiven und eine Sequenz von zugeordneten Referenzpunkten zu unterteilen. Durch die Multiplikation liefert der kombinierte Kernel für ein Paar aus Zustand und Aktionssequenz genau dann eine hohe Ähnlichkeit, wenn sowohl die Zustände als auch die Aktionssequenzen gemäß ihrer individuellen Kernel ähnlich zueinander sind.

Allgemein ist diese Art der Kombination von Kerneln nicht auf die oben gewählten Gauß- beziehungsweise Teilfolgenkernel beschränkt, sondern kann auch bei anderen Repräsentationen des Raumes der Zustände beziehungsweise Aktionssequenzen auf passende und valide Kernel angewandt werden.

4.2.2 Erweiterungen zur Wahl der Lernmethode

In diesem Kapitel wird ein Verfahren eingesetzt, das, wie in Abschnitt 3.3 beschrieben, in jeder Episode die Entscheidung trifft, welches der beiden Lernverfahren für die gegebene Situation das geeignetere ist. Wenn zu wenig Informationen für eine Situation vorliegen, entscheidet sich das System, um sowohl für den Roboter selbst als auch für seine Umgebung kein Sicherheitsrisiko einzugehen, für das Lernen aus Demonstrationen. Sind allerdings genügend Informationen vorhanden, so wird der Experte entlastet, indem das System autonom über die bekannten Lösungen generalisiert und dadurch auch für gegebenenfalls neue Situationen eine Lösung bestimmen kann.

Durch das Lernen ganzer Bewegungssequenzen statt nur einzelner Bewegungsprimitive erhöht sich der Aufwand für den Experten beim Demonstrieren einer gesuchten Lösung. Zudem hat das System in einigen Fällen schon Informationen darüber, wie die

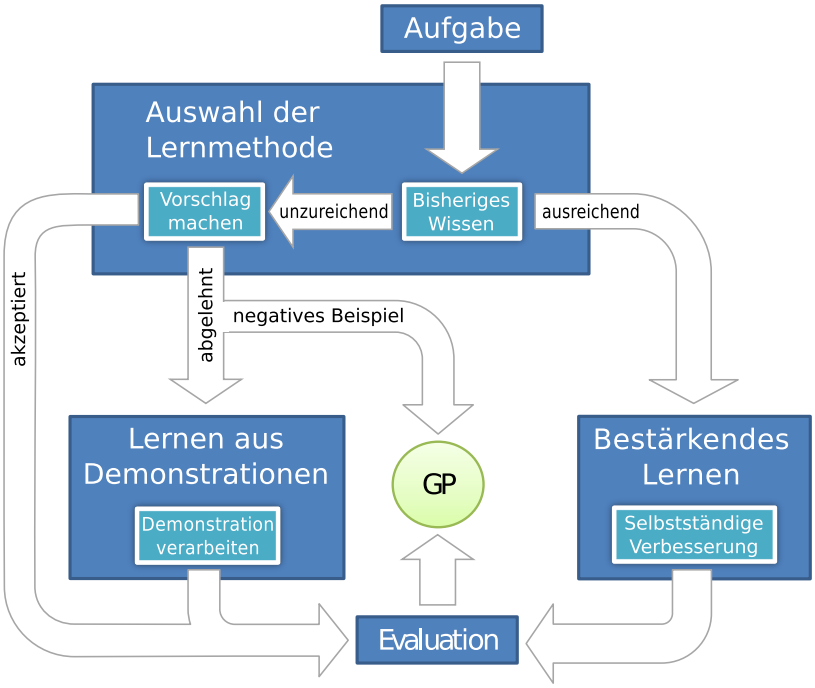


Abbildung 4.4: *Schematische Übersicht des vorgeschlagenen Systems, erweitert um die interaktive Komponente, dass das System einen Vorschlag für eine Aktionssequenz macht, um den Arbeitsaufwand gegenüber der direkten Anfrage nach einer Demonstration für den Experten zu reduzieren.*

gegebene Aufgabe zu lösen wäre, allerdings sind die Schätzungen dieser Informationen mit einer zu großen Unsicherheit behaftet, um diesen zu vertrauen. Der Grund dafür ist, dass die beobachteten Trainingsbeispiele im Zustandsraum zu unähnlich zu der aktuell gegebenen Situation sind und sich daher die Frage stellt, ob sich die aus ihnen gewonnenen Informationen übertragen lassen. Würde zugelassen werden, dass das System an dieser Stelle autonom, ausgehend von der besten bekannten Lösung nach einer Bewegungssequenz sucht, ohne diese zu überprüfen, bestünde das Risiko, dass das System Aktionssequenzen wählt, die dem Roboter oder seiner Umgebung schaden könnten.

Um ein gutes Gleichgewicht zwischen der Sicherheit und dem Aufwand für das Demonstrieren von Aktionssequenzen zu wahren, wird der Ansatz aus Abschnitt 3.3 für den Fall, dass keine sichere geeignete Aktionssequenz gefunden werden kann, um ein zweistufiges interaktives Element erweitert. Dies ist motiviert durch die Beobachtung, dass Aktionssequenzen, deren Q-Werte mit einer hohen Unsicherheit behaftet sind, zwar das Risiko bergen, dass diese dem Roboter oder der Umgebung schaden oder nicht geeignet sind, die Aufgabe zu erfüllen, aber in vielen Fällen doch die richtige Lösung beschreiben. In solchen Situationen schlägt das System zunächst die beste mit dem Verfahren aus Abschnitt 3.3 gefundene Aktionssequenz vor, ehe es nach einer Demonstration fragt. Der Experte wird in diesem Fall gebeten, der Lösungssequenz zuzustimmen oder, falls er diese zur Lösung der Aufgabe als ungeeignet empfindet, sie abzulehnen. Stimmt der Experte

der Aktionssequenz zu, braucht an dieser Stelle keine Demonstration der Lösung in der gegebenen Situation vorgeführt zu werden. Dadurch verringert sich der Arbeitsaufwand des Experten. Lehnt der Experte den Vorschlag des Systems ab, so fordert dieses eine Demonstration an. Zusätzlich wird die vom System vorgeschlagene und vom Experten abgelehnte Sequenz als neues, negatives Trainingsbeispiel zum Gauß-Prozess hinzugefügt. Dadurch bleibt dem System diese Sequenz als ungeeignet für die aktuelle Situation im Gedächtnis, ohne dass diese jemals ausgeführt werden muss, so dass der Roboter diesen negativen Effekt nicht direkt erfahren muss. Auch dieses negative Expertenwissen ist wertvoll für das Lernen von Aktionssequenzen, welches mit dem Verfahren aus Abschnitt 3 jedoch noch nicht genutzt wird. Die demonstrierte Aktionssequenz oder der akzeptierte Vorschlag wird anschließend ausgeführt, wodurch das System eine Belohnung für dieses Paar aus Zustand und Aktionssequenz erhält. Anschließend wird beides zusammen als ein neues Trainingsbeispiel gespeichert. Der beschriebene Verlauf der Wahl einer Aktionssequenz und des Generierens von Trainingsbeispielen ist anschaulich in Abbildung 4.4 dargestellt.

4.2.3 Lernen aus Demonstrationen

In dem hier vorgestellten Ansatz, mit dem komplexe Aufgaben mit Hilfe von Sequenzen von Bewegungsprimitiven gelöst werden, bestehen Demonstrationen jeweils aus einer Trajektorie aus 3D-Punkten, die als eine Sequenz von nahtlos ineinander übergehenden Aktionen aufgefasst wird. Mit dieser Sequenz erzielt der Experte den gewünschten Effekt auf seine Umgebung. Dies kann zum Beispiel eine Kombination aus den Einzelbewegungen `Greifen`, `Verstellen` und `Zurückziehen der Hand` sein, um ein Objekt zu versetzen. Damit der Roboter lernen kann, wie der gewünschte Effekt reproduziert werden kann, bestimmt das System die Reihenfolge und die Aktionsklassen der ausgeführten Aktionen. Es werden somit nicht die einzelnen Bewegungsprimitive und deren Form gelernt, sondern wie diese kombiniert und auf welche Objekte oder Positionen diese angewendet werden müssen, um die gegebene Aufgabe zu lösen. Dazu wird die demonstrierte Trajektorie segmentiert, die einzelnen Komponenten klassifiziert und mit der Identifikationsnummer der jeweiligen Aktionsklasse gekennzeichnet. Dies geschieht mit Hilfe einer Bibliothek von zuvor gelernten Bewegungsprimitiven. Zusätzlich ermittelt der Ansatz für jedes einzelne Segment den Start- und Endpunkt der Bewegung und weist jedem dieser Punkte einen Referenzpunkt aus einer zuvor definierten Menge von Punkten zu. Die Referenzpunkte sind dabei keine festen Koordinaten, sondern beschreiben aufgabenabhängige Positionen in der Umgebung. Zum Beispiel können sie die Position von bestimmten Objekten beschreiben oder stellvertretend für Orte stehen, die in jeder Aufgabeninstanz mit konkreten Positionen belegt werden. Die Zuweisung zu den Referenzpunkten erfolgt über eine Nächste-Nachbarn-Klassifikation [63] der Start- und Endpunkte der einzelnen Segmente. Um ein vollständiges Trainingsbeispiel zu generieren, wird anschließend die durch den Algorithmus bestimmte Sequenz von Bewegungsprimitiven in Bezug auf die dazugehörigen festgestellten Referenzpunkte ausgeführt und die Belohnung anhand des erreichten Effektes bestimmt. Der zeitliche Ablauf des Verfahren ist in Abbildung 4.5 zusammengefasst.

Allgemein kann in dem hier vorgestellten Verfahren jeder beliebige Algorithmus benutzt

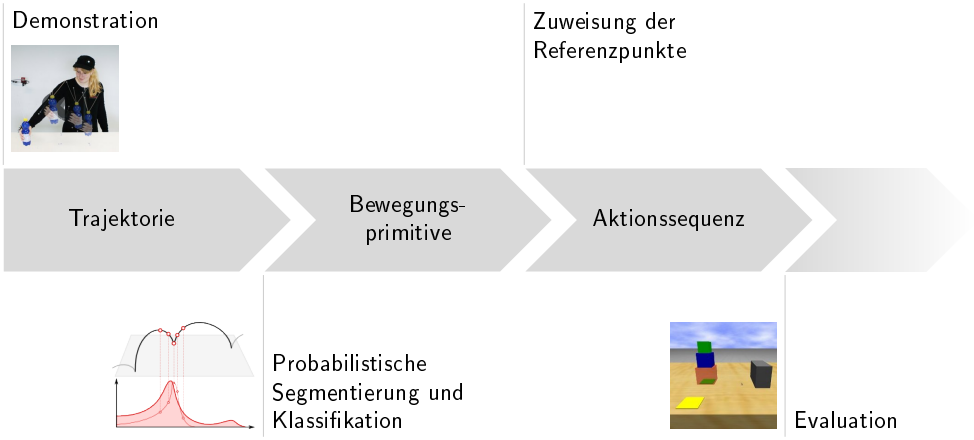


Abbildung 4.5: *Schematischer Ablauf des Lernens von Bewegungssequenzen anhand von Demonstrationen. Dabei wird zunächst eine Bewegungssequenz durch einen Experten demonstriert, die als Trajektorie an den in Kapitel 2 vorgestellten probabilistischen Segmentierungs- und Klassifikationsalgorithmus weitergegeben wird. Als Ergebnis daraus ergeben sich eine Reihe von Aktionsklassen, zugeordnet zu den einzelnen Bewegungsprimitiven. Anschließend werden den Endpunkten der einzelnen Bewegungen Referenzpunkte zugewiesen und ergeben dadurch eine komplette Aktionssequenz. Sowohl die Aktionsklassen als auch die dazugehörigen Referenzpunkte werden durch Identifikationsnummern beschrieben. Im letzten Schritt wird diese Sequenz evaluiert, und es ergibt sich zusammen mit der Rückmeldung der Umgebung ein neues Trainingsbeispiel.*

werden, um aus den Trajektorien, die zum Beispiel durch Beobachtung eines Menschen entstehen oder künstlich generiert werden können, Segmente zu bestimmen und diese den unterschiedlichen Bewegungsklassen zuzuordnen. In dieser Arbeit wird dazu der in Kapitel 2 beschriebene Ansatz benutzt. Für diesen wird in den Experimenten in Abschnitt 2.3 gezeigt, dass er präzise die Grenzen der einzelnen Segmente bestimmt und diese gleichzeitig richtig klassifiziert. Des Weiteren ist die in diesem Ansatz verwendete Merkmalsrepräsentation speziell auf die Beschreibung gerichteter Bewegungen ausgelegt, wie sie in diesem Abschnitt betrachtet werden. Diese Wahl der Merkmale sowie die HMM-basierte Repräsentation der Bewegungsprimitive ermöglichen die Generalisierung bekannter Bewegungen zu beliebigen Zielen und die Generierung entsprechender glatter Trajektorien. In den Experimenten werden die auf diesem Weg generierten Trajektorien genutzt, um die Effekte der Aktionssequenzen auf ihre Umgebung zu ermitteln und eine entsprechende Belohnung zu bestimmen.

4.2.4 Strategie zur Aktionsauswahl im Bestärkenden Lernen

Das Bestärkende Lernverfahren arbeitet mit einem Lernalgorithmus, der auf Q-Lernen basiert. Dazu wird der Raum der Zustände und Aktionssequenzen durch einen Gauß-Prozess approximiert, der mit einem speziell für das Lernen von Aktionssequenzen entwickelten Kernel arbeitet, der in Abschnitt 4.2.1 beschrieben ist. Aufbauend auf den Gauß-Prozesses-Schätzungen der Q-Funktion wird der Wert der sogenannten *Erwarteten Veränderung*[2] $\mathbf{E}D$ berechnet, um Aktionssequenzen auszuwählen, die in der nächsten Episode getestet werden sollen. Die Erwartete Veränderung stellt dabei eine Kombination aus der Erwarteten Verbesserung $\mathbf{E}D^{\oplus}$ und Verschlechterung $\mathbf{E}D^{\ominus}$ dar und sorgt dafür, dass auf der einen Seite vielversprechende Kandidaten im Bezug auf eine gute Performanz ausgewählt werden, aber gleichzeitig auf der anderen Seite darauf geachtet wird, dass keine risikoreichen Aktionssequenzen ausgewählt werden oder solche, die sehr unähnlich zu bisher gesehenen Sequenzen sind. Dadurch werden sowohl der Roboter als auch die Umgebung, in der er später eingesetzt werden soll, und die Menschen in dieser Umgebung geschützt. Diese Eigenschaft des Ansatzes ist besonders in realen Szenarien wichtig, da es die Vorhersehbarkeit der ausgewählten Aktionssequenzen für Menschen in interaktiven Anwendungen erhöht und gleichzeitig das Risiko von Schäden durch Ausführung unvorhersagbarer Sequenzen reduziert. Genauere Details der hier kombinierten Werte und der Vorteile, die durch ihre Kombination entstehen, sind in Abschnitt 3.2 beschrieben.

Durch Nutzung der folgenden Formulierung für die Erwartete Veränderung eines Paares aus Zustand und Aktionssequenz x

$$\begin{aligned} \mathbf{E}D(x) &= \mathbf{E}D^{\oplus}(x, Q_{\text{best}}) - f(\mathbf{E}D^{\ominus}, Q_{\text{best}}), \\ f(\mathbf{E}D^{\ominus}, Q_{\text{best}}) &= \Big(Q_{max} - \underbrace{(Q_{\text{best}} - \mathbf{E}D^{\ominus}(x, Q_{\text{best}}))}_{=\mu^{\ominus}(x, Q_{\text{best}})}\Big)^2 \end{aligned} \tag{4.5}$$

erreicht das Verfahren einen guten Kompromiss zwischen den beiden Werten der Erwarteten Verbesserung und Verschlechterung. Der lineare Einfluss der Erwarteten Verbesserung $\mathbf{E}D^{\oplus}$ favorisiert dabei das Auffinden von Aktionssequenzen, die eine hohe Belohnung versprechen. Der quadratische Einfluss der Differenz zwischen dem höchsten erreichbaren Q-Wert $Q_{\max}$ und dem erreichbaren Wert $\mu^{\ominus}(x, Q_{\text{best}})$ sorgt hingegen dafür, dass Aktionssequenzen abgewertet werden, sollte $\mu^{\ominus}(x, Q_{\text{best}})$ den Wert $Q_{\max}$ stark unterschreiten. Der Wert $Q_{\max}$ wird dabei zur Normalisierung benutzt und ist schon zur Designzeit aufgrund der gewählten Belohnungsfunktion bekannt. Der Funktionswert $\mu^{\ominus}(x, Q_{\text{best}})$ beschreibt den Wert, der, ausgehend von dem aktuell besten bekannten Wert Q_{best} mit Hilfe der Erwarteten Verschlechterung abgeschätzt wird. Der Wert Q_{best} wird während der Entscheidungsphase bestimmt und entspricht $\overline{f}$ aus Abschnitt 3.2.

Die Methode, die genutzt wird, um den Wert der Erwarteten Veränderung aus Gleichung (4.5) zu maximieren, ist durch diesen Ansatz nicht begrenzt und kann frei in Abhängigkeit von der Repräsentation der zu bestimmenden Aktionen gewählt werden. Da in den Experimenten in diesem Kapitel Sequenzen kategorischer Aktionen gelernt werden, wird im Folgenden eine lokale Nachbarschaftssuche zur Optimierung der Erwarteten Veränderung verwendet. Diese evaluiert alle Sequenzen, die sich um höchstens ein

[2] engl.: Expected Deviation

Element von der schon zuvor im Entscheidungsschritt für ein Lernverfahren bestimmten Aktionssequenz unterscheiden. Dabei setzt die Optimierung des Wertes der Erwarteten Veränderung keine beispielhaften Belohnungen durch Interaktion mit der Umwelt voraus. Stattdessen werden die Q-Werte der Aktionssequenzen effizient durch die Nutzung der Gauß-Prozess-Approximation vorhergesagt.

4.3 Experimente und Evaluation

Um den hier vorgestellten Ansatz zu validieren, wird ein Manipulationsszenario auf einem Tisch betrachtet, in dem die Aufgabe darin besteht, verschiedene Stapel von Boxen sicher von einer Position zu einer anderen zu transportieren. Die Instanzen dieses exemplarischen Szenarios sind allgegenwärtig im täglichen Leben zu finden. Beispiele für solche alltäglichen Aufgaben sind das Decken oder Aufräumen eines Tisches. Eine Herausforderung bei der gewählten Aufgabe liegt darin, dass verschiedene Manipulationsstrategien abhängig von der jeweiligen Situation angewendet werden müssen, um die Aufgabe zu lösen. Genau dies ist auch in realen Aufgaben häufig der Fall. Zum Beispiel können Hindernisse den Weg für eine direkte Bewegung zu der gewünschten Zielposition versperren oder physikalische Gesetzmäßigkeiten schränken die Wahl der Strategie zum Lösen der Aufgabe ein, so dass nur eine begrenzte Anzahl von gestapelten Boxen gleichzeitig bewegt werden kann. Im letzteren Fall muss das System lernen, die Aufgabe in mehrere Teilschritte zu zerlegen und nur eine begrenzte Anzahl an Boxen zur selben Zeit zu bewegen. Die hier vorgestellten Ergebnisse wurden in Experimenten erzeugt, die mit Hilfe des physikbasierten Simulators Gazebo [34] durchgeführt wurden. Sie zeigen die Fähigkeit des Ansatzes, die verschiedenen Strategien, die nötig sind, um diese Aufgabe in verschiedenen Situationen zu lösen, sicher zu erlernen.

4.3.1 Versuchsaufbau

In jeder Episode der Experimente wird ein Stapel von Boxen im Arbeitsbereich angeordnet und eine der Boxen speziell markiert. Die Aufgabe des Systems ist es, durch eine geeignete Kombination von Bewegungsprimitiven diese Box mit allen darauf stehenden Boxen zu einer bestimmten Zielposition im Arbeitsbereich zu bewegen. Die Bestandteile dieses Aufbaus, bestehend aus

- der Startposition des Stapels
- der Anzahl an Boxen, aus denen der Stapel besteht
- der Vorgabe, welche Box bewegt werden soll
- der Zielposition,

werden in jedem Durchlauf zufällig ausgewählt. Somit kann die Aufgabe daraus bestehen, einen kompletten Stapel von Boxen zu bewegen oder auch nur einen Teil davon, so dass ein Stumpf an der ursprünglichen Stelle zurückgelassen wird.

Abhängig von den zufällig bestimmten Situationsparametern sind verschiedene Aktionssequenzen nötig, um die gestellte Aufgabe lösen zu können. Diese Aufgabe wird

zusätzlich durch ein in der Mitte des Arbeitsbereiches angeordnetes Hindernis verkompliziert. Des Weiteren modelliert die Simulation die physikalische Interaktion der gestapelten Starrkörper. Für die simulierten Experimente wird der Reibungskoeffizient so gewählt, dass ein Stapel instabil wird und zusammenbricht, wenn mehr als zwei Boxen gleichzeitig auf eine neue Position bewegt werden. Dies ist der realen Welt nachempfunden, bei der ebenfalls das Balancieren eines Stapels immer schwieriger wird, je mehr Boxen dieser enthält. Als weiteres der realen Welt nachempfundenes Kriterium werden kürzere und energieeffizientere Aktionssequenzen vorgezogen, um den Roboter nicht unnötig zu belasten.

Um die gestapelten Boxen zu bewegen, ist das System mit vier einfachen Aktionen ausgestattet: Greifen eines Objektes, verstellen desselben, verschieben des Objektes zu einer Zielposition ohne es anzuheben und zurückziehen der Hand zu einer bequemen Ruheposition. Diese einfachen Aktionen werden mit Hilfe des Ansatzes, der in Kapitel 2 vorgestellt wird, trainiert und können in beliebiger Reihenfolge zusammengefügt werden, um komplexere Bewegungen zu erzeugen.

4.3.2 Aufgabendefinition

Die Leistung, einen Stapel von Boxen zu einer gewünschten Position zu versetzen, wird als die Arbeit W gemessen, die während einer Episode der Länge T aufgewendet werden muss. Diese wird aus der Änderung der kinetischen Energie E_k der bewegten Objekte abgeleitet, welche sich wiederum aus der Masse m der Objekte und deren Geschwindigkeit v zur Zeit t zusammensetzt:

$$W = \sum_{t=1}^{T} |E_k(t) - E_k(t-1)| \qquad E_k(t) = \frac{1}{2} m v^2(t) \tag{4.6}$$

Am Ende einer Episode erhält das System eine Belohnung, die proportional zu diesem Wert ist, wenn es erfolgreich den Stapel von Boxen versetzt hat. Wenn die ausgewählten Bewegungen zu einer Kollision mit dem Hindernis führen, wird die Aktionssequenz durch eine negative Belohnung bestraft, so dass das System davon abgehalten wird, weitere Aktionssequenzen zu wählen, die zu einer Kollision führen. Aktionssequenzen, die von einem Experten abgelehnt werden, bekommen ebenfalls eine negative Belohnung. Dies entspricht der Einschätzung des Experten, dass diese Aktionsfolge für die aktuelle Situation nicht geeignet ist oder gar ein Risiko für den Roboter oder seine Umgebung darstellen könnte. Sollte das System nach der Episode das Ziel, den ausgewählten Teilstapel an eine gewünschte Position zu bewegen, nicht erreicht haben, so wird eine Belohnung von null vergeben. Zusammengefasst ergibt sich die Belohnung für eine Aktionssequenz a in einer gegebenen Situation s aus

$$r(s,a) = \begin{cases} -1 & \text{Kollision oder abgewiesene Vorschläge,} \\ 0 & \text{Fehlschlag,} \\ 1 - W & \text{erfolgreiches Versetzen des Stapels.} \end{cases} \tag{4.7}$$

Beim Bestärkenden Lernen schätzt der Q-Wert die erwartete Langzeitbelohnung für Paare aus Zuständen und Aktionssequenzen. In dem hier vorgestellten Ansatz wird

Tabelle 4.1: *Repräsentation der Zustände und Aktionssequenzen in den Experimenten*

Zustandsvariablen	
Boxposition	$\in \mathbb{R}^2$
Ziel	$\in \mathbb{R}^2$
Stapelhöhe	$\in \{1, \ldots,$ Anzahl der Boxen im Stapel$\}$
Ausgewählte Box	$\in \{0, \ldots,$ Anzahl der Boxen im Stapel $- 1\}$
Aktionsvariablen pro Aktion in der Sequenz	
Aktion	$\in$ {Greifen, Verschieben, Verstellen, Zurückziehen}
Zielreferenzpunkt	$\in$ {Start, Ziel, Ablage} $\times$ Anzahl Boxen $\cup$ {Ruheposition}

eine Episode in einem einzigen Schritt mittels einer einzelnen, komplexen Aktion gelöst, welche eine ganze Sequenz von einfachen Aktionen abstrahiert. Dadurch ist Q in diesem Fall mit r gleichzusetzen. Anstatt die zusammengesetzte Belohnungsfunktion durch nur einen Gauß-Prozess zu approximieren, wird eine Mischung aus jeweils einem eigenen Gauß-Prozess-Modell für die verschiedenen Komponenten der Belohnung aus Gleichung (4.7) genutzt. Dies ermöglicht es, den einzelnen Komponenten unterschiedliche Hyperparameter zuzuordnen und dadurch ihren Einflussbereich zu bestimmen. Während der Experimente wird eine Kernelbreite von 0,25 für die erfolgreichen und 0,055 für die Fälle, die nicht erfolgreich waren, verwendet. Durch die Wahl schmaler Kernelbreiten haben nicht erfolgreiche Beispiele einen lokaleren Einfluss. Der Schwellwert θ, der mit dem Wert aus Gleichung (3.13) verglichen wird, um zu entscheiden, ob das Lernen aus Demonstrationen oder Bestärkendes Lernen das geeignetere Werkzeug in der gegebenen Situation ist, wird auf $\theta = 0,25$ gesetzt. Dieser Wert wurde empirisch bestimmt und stellt einen guten Kompromiss zwischen der Sicherheit des Roboters und seiner Umgebung und der Anzahl der Interaktionen mit dem menschlichen Experten, um dessen Arbeitsaufwand nicht zu groß werden zu lassen, dar.

Der Zustandsraum $\mathcal{S}$ in den Experimenten besteht aus den Koordinaten des Stapels aus Boxen zu Beginn einer Episode und einer Zielposition auf dem Tisch, der Anzahl von Objekten, aus denen der Stapel besteht, und der Nummer der Box im Stapel, die, mit allen Boxen auf ihr, zu der Zielposition bewegt werden soll. Während im Zustandsraum die Start- und Zielposition mittels Koordinaten beschrieben werden, werden die benötigten Positionen im aufgabenspezifischen Aktionssequenzraum nicht direkt durch Koordinaten dargestellt, sondern durch eindeutige Namen, die auf bestimmte Positionen und Objekte verweisen. Dafür werden mehrere Orte, die relevant für die Aufgabe sind, durch Referenzpunkte benannt. Diese Punkte sind unabhängig von ihren genauen Positionen auf dem Tisch, die sich in jeder Episode ändern. Die Menge der Referenzpunkte enthält den Start- und Zielort des Stapels, zwei Positionen, an denen Boxen vorübergehend abgelegt werden können—zum Beispiel, um einen zu großen Stapel zu unterteilen—und eine für den Arm bequeme Ruheposition, die der Roboter zwischen einzelnen Aktionen einnehmen kann. Bis auf die Ruheposition haben alle Referenzpunkte verschiedene Werte in der Höhe innerhalb des Stapels. Da nur die Start- und Zielposition des Stapels während der verschiedenen Episoden variieren, sind deren Koordinaten die

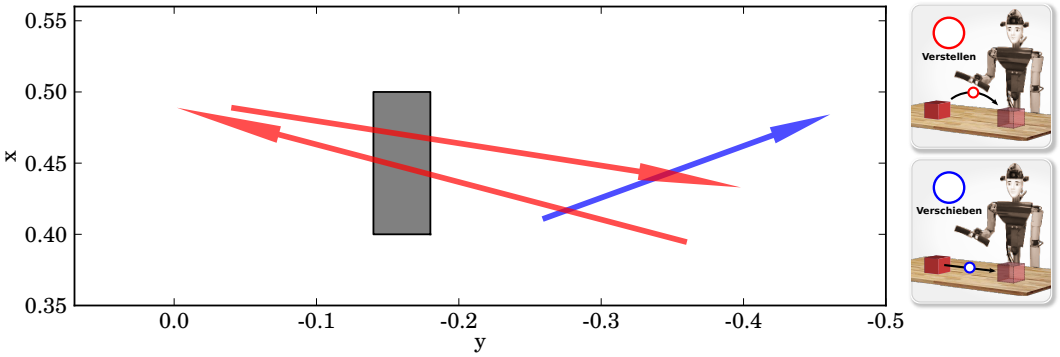


Abbildung 4.6: *Ansicht von oben auf den Arbeitsbereich auf dem Tisch und die zufällig bestimmten Aufgabensituationen, in denen Demonstrationen von einem Experten angefordert werden. Diese Demonstrationen werden als Pfeile dargestellt, die die Start- und Zielposition der Box verbinden. Das graue Rechteck stellt das Hindernis auf dem Tisch dar. Die Farbe der Pfeile gibt an, ob die Box während der Demonstration über das Hindernis gehoben (rot/hell) oder an die Zielposition verschoben wird (blau/dunkel).*

einzigen, die in den zustandsbeschreibenden Vektor einfließen, da sie Einfluss auf die richtige Lösung der Aufgabe haben. Zum Beispiel hängt von diesen Positionen ab, ob es einen direkten Weg ohne eine Kollision zwischen den beiden Positionen gibt oder ob ein Objekt angehoben werden muss, um eine Kollision zu vermeiden.

Die zum Lösen einer Aufgabe benötigten Aktionssequenzen $a \in \mathcal{A}$ bestehen aus einer variabel langen Sequenz von Identifikationsnummern von vorgegebenen Bewegungsprimitiven, die in der angegebenen Reihenfolge ausgeführt werden sollen, und einer Liste mit den zu den Bewegungsprimitiven dazugehörigen Parametern in Form der Identifikationsnummern der Referenzpunkte. Während des Generierens der Aktionssequenzen werden die Identifikationsnummern der Referenzpunkte mit Koordinaten belegt. Bei der Generierung wird zudem davon ausgegangen, dass jede Aktionssequenz an einer festgelegten Ruheposition des Roboterendeffektors beginnt und dass der Zielreferenzpunkt einer Bewegung auch der Startreferenzpunkt der nachfolgenden Bewegung ist. Die Variablen, die den Raum der Zustände und Aktionssequenzen beschreiben, sind in Tabelle 4.1 zusammengefasst.

Für die Experimente werden die Demonstrationen des Experten mit Hilfe eines statistischen Modells, das auf gesammelten Motion Capture-Daten trainiert wird, simuliert. Genauere Informationen dazu sind in Abschnitt 2.2.4 zu finden. Mit den gelernten Modellen der einzelnen Bewegungen werden von diesem System Aktionssequenzen in Trajektorien umgewandelt. Daraus werden die Auswirkungen der Sequenz auf die verschiedenen Objekte berechnet, ausgeführt und anhand dieser Ergebnisse die simulierte Umgebung aktualisiert.

4.3.3 Vorteile des integrierten Ansatzes

Um zu zeigen, wie der Algorithmus aus der Kombination der beiden Lernverfahren, dem Lernen aus Demonstrationen und dem Bestärkenden Lernen, profitiert, um effizient

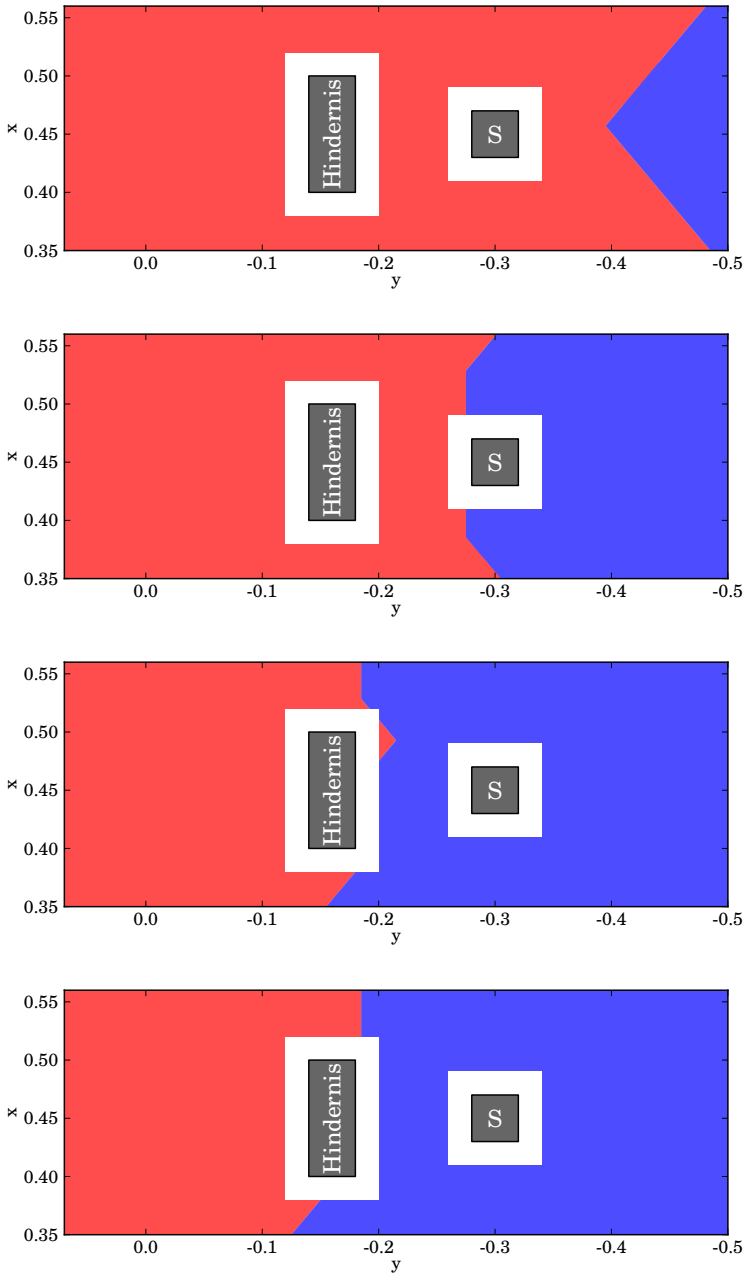


Abbildung 4.7: *Die Ansicht von oben auf den Arbeitsbereich stellt die Entwicklung der Strategie des Bestärkenden Lernansatzes nach 10, 20, 30 und 60 Episoden dar. Die graue Box auf der rechten Seite repräsentiert die Startposition S der Box, welche für dieses Experiment festgehalten wird. Die linke graue Box ist ein Hindernis. Die Bewegungssequenzen, die durch das Bestärkende Lernmodul bestimmt werden, sind farblich gekennzeichnet. Für Zielpositionen in dem rot (hell) markierten Bereich wählt das System eine Sequenz mit einer Greif-, einer Verstell- und einer Zurückziehbewegung aus, während bei blau (dunkel) gekennzeichneten Zielpositionen die Box zum Ziel verschoben wird, ohne das Objekt anzuheben. Ziele, die sich nah an der initialen Position oder dem Hindernis befinden, werden durch einen weißen Bereich dargestellt und nicht beachtet, da sie schon aufgrund der Objektgröße in jedem Fall zu Kollisionen führen würden.*

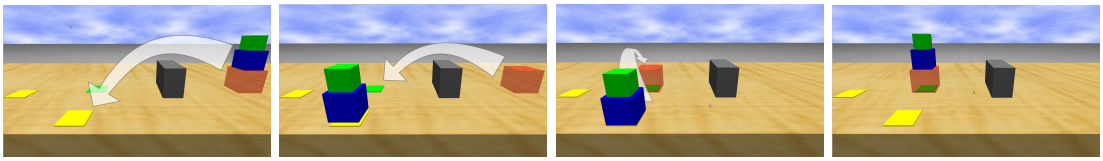


Abbildung 4.8: *Visualisierung einer Strategie, die erforderlich ist, um einen Stapel aus drei Boxen zu versetzen. Die Zielposition ist grün markiert, während die gelben Markierungen mögliche zusätzliche Abstellpositionen für die Boxen repräsentieren. Das Hindernis wird durch eine graue Box gekennzeichnet. Da drei gestapelte Boxen in den Experimenten instabil werden, falls versucht wird, diese alle gleichzeitig zu bewegen, muss der Stapel zunächst aufgeteilt werden. Dafür werden die oberen beiden Boxen zu einer der Abstellpositionen bewegt. Dann wird der übriggebliebene Stapel, in diesem Fall bestehend aus nur einer Box, zur Zielposition transportiert, wo der Stapel in einem dritten Schritt wieder zusammengesetzt wird.*

und sicher Strategien zur Lösung eines Problems zu finden, wird in diesem Experiment eine vereinfachte Version der kompletten Aufgabe mit nur einer einzigen Box betrachtet. Dies macht es möglich, unterschiedliche Aspekte des Lernverfahrens zu visualisieren. Die Start- und Zielpositionen der Box werden dabei weiterhin zufällig gewählt, so dass die Herausforderungen bestehen bleiben, dass Aktionssequenzen mit Hilfe der zuvor untersuchten Beispiele generalisiert werden müssen und die Aufgabe in verschiedenen Situationen nur mit unterschiedlichen Aktionssequenzen optimal gelöst werden kann.

Abbildung 4.6 zeigt eine Ansicht auf den Tisch mit dem Hindernis. Das Experiment wird ohne Vorwissen gestartet, was dazu führt, dass das System für die Episoden eins, drei und sieben nach Demonstrationen fragt. Die dazugehörigen Situationen sind durch Pfeile in der Abbildung dargestellt, mit denen die Start- und Zielpositionen verbunden werden. Bevor das System nach einer Demonstration fragt, erkennt es, dass nicht genügend Informationen vorliegen, um eine sichere Aktionssequenz selbstständig zu generieren. Um den Aufwand für den Menschen gering zu halten, versucht das System trotzdem zunächst aufgrund der vorhandenen Informationen eine möglichst gute Aktionssequenz vorzuschlagen. Diese Sequenz hätte in den oben genannten Fällen allerdings entweder zu einer Kollision geführt oder dem Experten schien eine andere Lösung geeigneter, so dass er diese Vorschläge abgelehnt hat. Nach Episode sieben ist das bis dahin gesammelte Wissen zum Lösen der Aufgabe für das System ausreichend, um autonom eigene Lösungen für alle weiteren Situationen zu finden.

Um zu veranschaulichen, wie das System während des Lernprozesses inkrementell Wissen über die Aufgabe und somit über mögliche Zustände und Aktionssequenzen sammelt, wird zu verschiedenen Zeitpunkten eine Reihe von Anfragen von einem gemeinsamen Ausgangspunkt zu Zielpositionen auf einem regelmäßigen Gitter im gesamten Arbeitsbereich generiert. Abbildung 4.7 zeigt die Verteilung von Aktionssequenzen, die dabei von dem System für verschiedenen Zielsituationen durch das Bestärkende Lernen ausgewählt werden. Die gewünschten Zielpositionen auf dem Tisch sind farbkodiert gemäß der vorgeschlagenen Bewegungssequenz. Für Zielpositionen im roten (hellen) Bereich werden Sequenzen mit einer Greif-, einer Verstell- und einer Zurückziehbewegung gewählt. Für Positionen im blauen (dunklen) Bereich entscheidet sich das System statt für eine

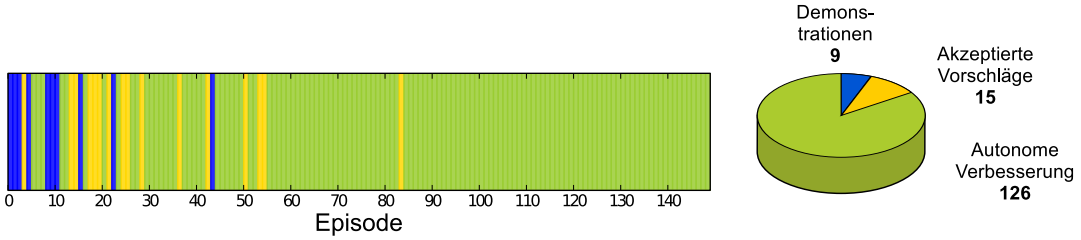


Abbildung 4.9: *Entscheidungen des Systems während des Experimentes mit 150 Episoden, wobei die komplette Aufgabe mit bis zu drei Boxen zu lösen ist. In Episoden, die blau (dunkel) markiert sind, fragt das System nach einer Demonstration, nachdem ein initialer Vorschlag durch den Experten abgelehnt wurde. Episoden, in denen der Vorschlag des Systems angenommen wird, werden in gelb (hell) dargestellt. In der Mehrzahl der Episoden, in grün (mittelhell) hervorgehoben, wählt das System eine Aktionssequenz autonom. Das Diagramm auf der linken Seite gibt die zeitliche Reihenfolge der Entscheidungen an, wohingegen das Tortendiagramm auf der rechten Seite die Anzahl der Entscheidungen für einen bestimmten Lernansatz zusammenfasst.*

Verstell- für eine Verschiebebewegung, um die Box an die gewünschte Zielposition zu transportieren. Da die Belohnungsfunktion aus Gleichung (4.7) die physikalische Arbeit, die benötigt wird, um eine Bewegung auszuführen, bestraft, kann das System bessere Ergebnisse durch das Verschieben eines Objektes statt durch das Verstellen desselben erreichen. Auf der anderen Seite ist eine Verschiebebewegung aber nur möglich, wenn kein Hindernis den direkten Weg zu der gewünschten Zielposition versperrt, da es ansonsten zu Kollisionen kommt. Da das System keine willkürlichen Aktionssequenzen testet, nutzt es für alle Zielpositionen eine der beiden demonstrierten Aktionssequenzen, die in Abbildung 4.6 dargestellt sind, und teilt den Arbeitsbereich in zwei in sich zusammenhängende Bereiche auf, in denen es jeweils eine der beiden Strategien verfolgt. Die Abbildung zeigt, wie das System nach den ersten 10 Episoden für die meisten Zielposition vorschlägt, das Objekt zu verstellen. Zu diesem Zeitpunkt sind alle Demonstrationen bereits vorhanden. Das Verstellen der Box zu Positionen auf der rechten Seite des Arbeitsbereiches ist hinlänglich ähnlich zu vorausgegangenen Beispielen, was zur Folge hat, dass eine Verstellbewegung als sicherer als eine Verschiebebewegung angesehen wird. Andererseits gibt es noch nicht genug Beispiele, um eine verlässliche Schätzung des Q-Wertes abzugeben, so dass noch Raum für Verbesserungen besteht. Nachdem weitere Beispiele gesammelt wurden, sinkt die Unsicherheit der Schätzungen der Q-Werte und das System entscheidet sich richtigerweise für die energieeffizienteren Aktionen, soweit dieses möglich ist und keine Kollision dadurch entsteht. Zu beachten ist, dass durch das Abwägen zwischen der Erwarteten Verbesserung und Verschlechterung und dem Einbeziehen eines Experten das System immer eine geeignete Aktionssequenz produziert, um die gegebene Aufgabe zu lösen und dabei keine einzige Kollision mit dem Hindernis erzeugt wird.

4.3.4 Lernen von Aktionssequenzen variabler Länge

Um die Fähigkeit des Systems zu demonstrieren, effizient eine Strategie von Aktionssequenzen von variabler Länge zu lernen, wird in diesem Abschnitt die komplette Aufgabe

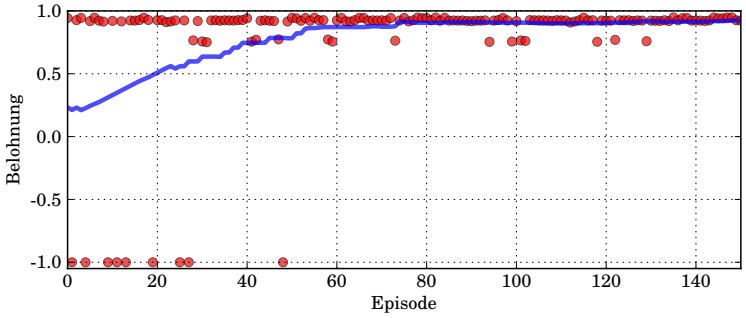


Abbildung 4.10: *Erhaltene Belohnung, dargestellt durch die roten Punkte, und deren Mittelwert über 50 Episoden, visualisiert durch die blaue Linie.*

mit bis zu drei gestapelten Boxen betrachtet. Abbildung 4.8 stellt eine Strategie dar, die das System lernen muss, um einen Stapel von drei Boxen zu versetzen. In der Simulation werden Stapel mit mehr als zwei Boxen instabil, wenn sie gleichzeitig bewegt werden, und kollabieren. Als Konsequenz kann das Versetzen eines Stapels mit mehr als zwei Boxen nur erfolgreich absolviert werden, indem zunächst ein Teil des Stapels abgehoben wird, dann der restliche Teil auf die Zielposition bewegt wird und am Ende die Teile dort wieder zusammengesetzt werden. Des Weiteren können bessere Ergebnisse durch das Verschieben statt durch das Verstellen der Boxen zur Zielposition erlangt werden, da die Belohnungsfunktion die physikalische Arbeit der Bewegungen mit einbezieht. Dies ist allerdings nur möglich, wenn kein Hindernis im direkten Weg zwischen den beiden Positionen liegt und das zu bewegende Objekt zusätzlich direkt auf dem Tisch steht. Die Experimente zeigen, dass das System alle diese Aufgabenaspekte erfasst und zuverlässig sichere Aktionssequenzen generiert, ohne dabei eine einzige Kollision zu erzeugen.

Für dieses Experiment werden 150 Episoden mit zufällig generierten Aufgaben mit bis zu drei Boxen ausgeführt. Die Abbildung 4.9 zeigt die Reihenfolge und eine Zusammenfassung der durch das System getroffenen Entscheidungen. Insgesamt wird in 24 Episoden Hilfe durch den Experten angefordert. In den ersten Episoden ist zu wenig Wissen über die Aufgabe vorhanden und somit können noch keine Aktionssequenzen autonom erzeugt werden. Folglich wird in 20 der ersten 50 Episoden der Experte um Hilfe gebeten und in 45 % dieser Fälle wird der Vorschlag, der durch das System generiert wird, zurückgewiesen. Dies liegt ebenfalls an dem aktuell noch niedrigen Wissensstand des Systems. Somit kommt es zu neun Anfragen einer Demonstration für die jeweiligen Aufgaben. Über diesen Punkt hinaus werden keine weiteren Demonstration mehr angefordert. Da die gesammelten Erfahrungen durch Demonstrationen und erfolgreiche autonome Versuche wachsen, nimmt die Qualität der Vorschläge des Systems zu und dadurch werden sie auch zunehmend vom Experten akzeptiert. Zur selben Zeit nimmt die Anzahl der Instanzen, in denen der Experte um die Zustimmung oder Ablehnung eines Vorschlages gebeten wird, ab. Nach 84 zufälligen Episoden hat das System eine vollständige Strategie für jede der möglichen Aufgabensituationen gelernt und beobachtet keine unbekannten Situationen mehr, in denen die Hilfe eines Experten erforderlich ist. Da die verwendete Explorationsstrategie die Unsicherheit verbunden mit dem vorherge-

sagten Mittelwert des zu erwartenden Q-Wertes miteinbezieht, schlägt das System keine willkürlichen Aktionssequenzen vor und variiert stattdessen die Aktionssequenzen in den Grenzen, die durch die zuvor gesammelten Erfahrungen festgelegt wurden.

Die Abbildung 4.10 zeigt die Belohnungswerte, die während des gesamten Lernprozesses beobachtet werden, und ihren Durchschnitt mit einer Fensterbreite von 50 Episoden. Während abgelehnte Vorschläge zu mehreren negativen Belohnungen in den ersten 50 Episoden führen, erhält das System danach nur noch positive Belohnungen. Es kommt also zu keinerlei Kollisionen. Die Ausreißer mit etwas schlechteren Belohnungen entsprechen Situationen, für die das Zerlegen des Stapels von Nöten ist. Die zusätzlichen Aktionen, die dazu benötigt werden, erfordern zusätzliche physikalische Arbeit und führen somit zu diesen reduzierten Belohnungen.

4.4 Zusammenfassung

In diesem Kapitel wurde ein neuer Ansatz zum Lernen von Aktionssequenzen variabler Länge vorgestellt. Dieser ermöglicht es Robotern, komplexe Aufgaben durch Anwendung verschiedener Strategien in Abhängigkeit von der gegebenen Situation zu lösen. In diesem Verfahren wird eine Kombination von Bestärkendem Lernen und Lernen aus Demonstrationen eingesetzt. Dabei kombiniert der Ansatz die komplementären Stärken beider Lernverfahren so, dass sie sich ergänzen und gleichzeitig ihre individuellen Schwächen reduziert werden. In jeder Episode entscheidet das System dazu autonom, anhand der bis zu diesem Zeitpunkt gesammelten Erfahrungen, ob zur Lösung der Aufgabe eine Aktionssequenz mit Hilfe der durch das Bestärkende Lernen ermittelten Strategie generiert werden kann oder ob diese durch die Hilfe eines Experten gelernt werden muss. Für den letzteren Fall wird eine zweistufige, interaktive Komponente vorgeschlagen, um den Arbeitsaufwand des Experten gering zu halten. Dazu schlägt das System die ihm beste bekannte Lösung zunächst vor und erst, wenn diese durch den Experten abgelehnt wird, wird eine Demonstration der Aufgabenlösung notwendig. Somit wird die benötigte Anzahl an Demonstrationen reduziert.

Ist keine Demonstration erforderlich, so setzt das Verfahren einen Q-Lernalgorithmus zur Bestimmung einer geeigneten Aktionssequenz ein. Dabei wird die Q-Funktion über den kombinierten Raum der möglichen Zustände und Aktionssequenzen mit einem Gauß-Prozess approximiert. Dazu setzt das System eine speziell entwickelte Kernelfunktion ein, die als Kombination des Gauß-Kernels mit einem Teilfolgenkernel über Zeichenketten definiert ist. Dies erlaubt es dem Ansatz, eine Strategie über kontinuierliche Zustandsräume und Aktionssequenzräume, die variabel lange Sequenzen kategorischer Aktionen beinhalten, zu lernen. Durch Generalisierung der Beobachtungen des Systems mittels Gauß-Prozess-Regression werden probabilistische Schätzungen der Q-Funktion für unbekannte Situationen und Aktionssequenzen bestimmt. Die in diesen Schätzungen enthaltene Unsicherheit erlaubt es, mögliche Paare aus Zuständen und Aktionssequenzen mit einer Bayesschen Explorationsstrategie zu bewerten. Diese nutzt eine Kombination aus den Werten der Erwarteten Verbesserung und der Erwarteten Verschlechterung, um vielversprechende Aktionssequenzen zu finden, aber auch gleichzeitig das Risiko für den Roboter und seine Umgebung beim Ausführen einer Sequenz niedrig zu halten.

Der hier vorgestellte Ansatz wird anhand einer typischen Manipulationsaufgabe unter-

sucht, bei der gestapelte Boxen versetzt werden sollen. Die Ergebnisse zeigen die Fähigkeit des Systems, sicher und effizient die Aufgabe aus einer geringen Anzahl an Demonstrationen eines Experten zu lernen. Dabei profitiert das System von der Kombination aus Bestärkendem Lernen und Expertenwissen sowie der auf Optimierung der Erwarteten Veränderung basierenden Explorationsstrategie. Während der Experimente kam es in keinem Fall zu einer Kollision. Ist kein ausreichendes Wissen für einen autonomen Lösungsversuch vorhanden, so dass das System dem Experten einen Lösungsvorschlag zur Evaluation vorschlägt, wird dieser Vorschlag in den Experimenten in über 62% der Fälle von dem Experten akzeptiert, so dass unnötige Demonstrationen vermieden werden.

5 Kapitel Lernen hierarchischer Aufgaben

Im vorangegangenen Kapitel wurde ein Ansatz vorgestellt, um Robotern beizubringen, zusammengesetzte Aufgaben durch geeignete Kombination und Anordnung von Bewegungsprimitiven zu lösen. Dieses Verfahren stellt einen Schritt auf dem Weg zu Robotern dar, die typische Serviceaufgaben im Alltag übernehmen können. Da aber viele Aufgaben des alltäglichen Lebens hochdimensionale Zustandsräume aufweisen und lange Sequenzen von komplexen Aktionen erfordern, werden leistungsfähige Lernalgorithmen benötigt, die mit der kombinatorischen Komplexität der häuslichen Umgebung skalieren können und es ermöglichen, gelernte Aufgabenteile wiederzuverwenden. Inspiriert durch die Art und Weise, wie Menschen komplexe Aufgaben zerlegen, haben hierarchische Methoden für die Entwicklung von Lernverfahren in der Robotik signifikantes Interesse geweckt.

In diesem Kapitel wird ein Lernverfahren vorgestellt, um Robotern das Lösen komplexer, hierarchisch strukturierter Aufgaben zu ermöglichen. Zentraler Bestandteil des Ansatzes ist die MAXQ-Methode für hierarchisches Bestärkendes Lernen, die hier für den Einsatz in kontinuierlichen Zustandsräumen erweitert wird. Dazu werden die einzelnen Komponenten der MAXQ-Zerlegung der Q-Funktion auf den verschiedenen Abstraktionsebenen der Aufgabenhierarchie durch Gauß-Prozesse approximiert. Dies ermöglicht die Berechnung einer rekursiven, probabilistischen Schätzung des Nutzens für beliebige Paare aus Zuständen und Aktionen für verschiedene Teilaufgaben. Aus dieser Schätzung lässt sich für ein solches Paar aus Zustand und Aktion der Wert der Erwarteten Veränderung für eine Teilaufgabe berechnen. Dieser wird in diesem Ansatz als Optimierungskriterium einer Bayesschen Explorationsstrategie eingesetzt. Dadurch wird ein Kompromiss zwischen der Optimierung des Q-Wertes und der Vermeidung potenzieller Risiken durch die Exploration von Aktionen, deren Ausgang nicht sicher abgeschätzt werden kann, dargestellt. Diese Strategie wird in dem hier vorgestellten Ansatz in den MAXQ-Algorithmus für Bestärkendes Lernen integriert, um eine effiziente Auswahl vielversprechender und gleichzeitig sicherer Aktionen zu erreichen.

Eine Voraussetzung für die Anwendung von Gauß-Prozess-Regression und somit für eine effiziente Exploration ist dabei die Verfügbarkeit von Erfahrungen, die als Trainingsdaten dienen können. Aus diesem Grund wird in diesem Ansatz das hierarchische Bestärkende Lernen mit dem Lernen aus Demonstrationen kombiniert. Dazu wird ein Entscheidungsmodul verwendet, das für jede Teilaufgabe in der MAXQ-Hierarchie in jeder Iteration evaluiert, ob genügend Informationen für eine sichere autonome Wahl der

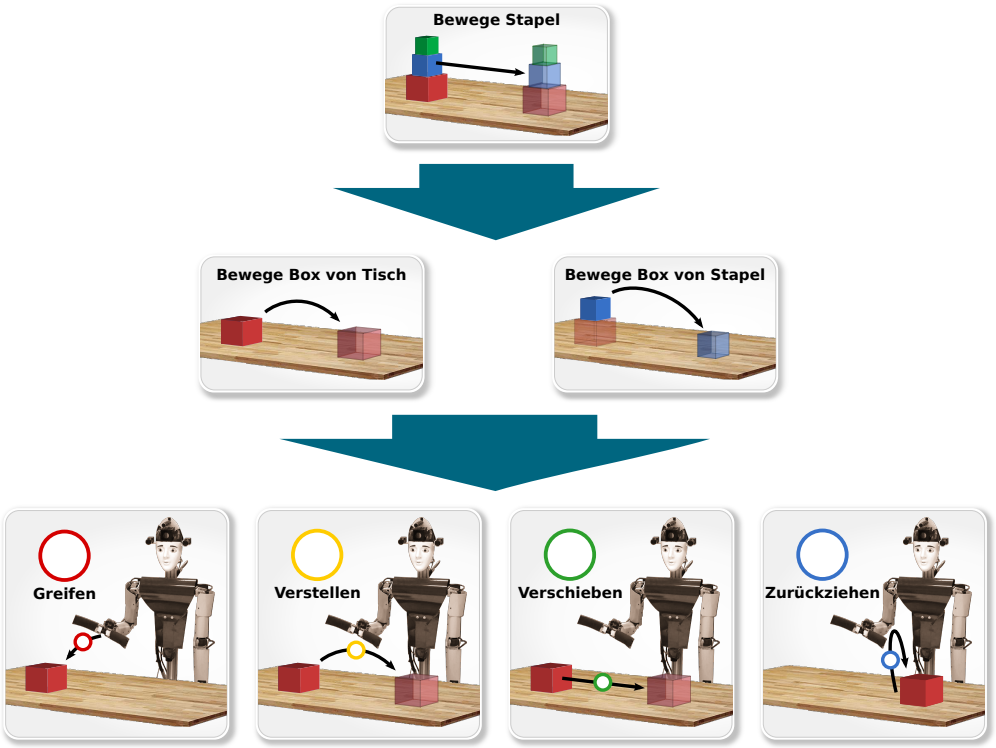


Abbildung 5.1: *Hierarchische Dekomposition einer Beispielaufgabe, bei der ein Stapel von Boxen versetzt werden soll. Dabei sind die komplette Aufgabe auf der obersten Ebene über die Teilaufgaben auf der mittleren Ebene, bei denen verschiedenen Teilstapel bewegt werden, bis hin zu den Bewegungsprimitiven auf der untersten Ebene dargestellt.*

nächsten Aktion zur Verfügung stehen. Auf diese Weise wird Hintergrundwissen eines Experten nur in den Situationen integriert, in denen dieses benötigt wird, und auch nur für die Teile der Aufgabe, für die nicht genügend Informationen vorhanden sind. Für den Experten bedeutet dies, dass er zielgenau Hilfestellung geben kann und unnötige Demonstrationen oder Wiederholungen von Aufgabenteilen vermieden werden. Aufgrund der modularen Architektur, in der dasselbe Lernverfahren für alle Teilaufgaben beliebiger Aktionsebenen eingesetzt werden kann, ist das hier vorgestellte System skalierbar und lässt sich leicht an Aufgaben unterschiedlicher Komplexität anpassen.

Die Fähigkeit des hier vorgeschlagenen Systems, effizient Lösungen für komplexe Aufgaben zu finden, wird anhand von Experimenten in demselben Szenario wie in Abschnitt 4.3 demonstriert. In diesem soll ein Stapel von Boxen auf verschiedene Art und Weise, abhängig von der gegebenen Situation, zu einer Zielposition bewegt werden. Der Ansatz in diesem Kapitel basiert auf der eigenen Arbeit [64], die auf der ICRA 2014 vorgestellt wurde.

5.1 Einleitung

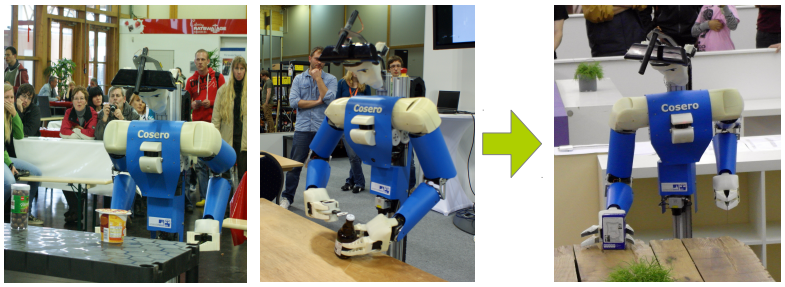
Um für Serviceaufgaben im Alltag eingesetzt werden zu können, müssen Roboter unter anderem in der Lage sein, mehrere einfache Fähigkeiten zum Lösen einer Aufgabe zu kombinieren und dabei verschiedenste Aspekte der Umgebung zu berücksichtigen. Angesichts der Komplexität dieser Aufgaben und der häuslichen Umgebung sind hierzu Lernmethoden erforderlich, die in der Lage sind, effizient komplexe Aufgabenstrategien auch in großen Zustandsräumen zu lernen. Das im vorangegangenen Kapitel beschriebene Verfahren leistet hierzu einen Beitrag, indem es Robotern ermöglicht, geeignete Aktionssequenzen durch Beobachtung eines Experten oder durch autonome Interaktion mit der Umgebung zu erlernen. Für anspruchsvolle praktische Anwendungen stößt dieses Verfahren jedoch mit zunehmender Komplexität der Aufgaben und der Umgebung durch das exponentielle Wachstum der Zustands- und Aktionsräume an seine Grenzen. Dieser Effekt wird auch als Fluch der Dimensionen[1] [65] bezeichnet und ist eines der wesentlichen Hindernisse bei der Entwicklung von Lernverfahren für Serviceroboter.

Erkenntnisse der Kognitionswissenschaften [66] legen nahe, dass Menschen diesen Herausforderungen begegnen, indem sie Strukturen in den Aufgaben und in ihrer Umgebung erkennen und Aufgaben so lange in Teilaufgaben zerlegen, bis nur noch atomare, einfach zu lösende Aufgabenteile vorhanden sind. Inspiriert durch dieses biologische Vorbild stellen hierarchische Methoden einen vielversprechenden Ansatz dar, um Lernverfahren für Roboter für komplexe Anwendungen zu skalieren. Diese können von hierarchischen Strukturen profitieren, die in der Regel in komplexen Aufgaben vorhanden sind. Außerdem können sie verschiedene Arten der Abstraktion verwenden, um Aufgaben in kleinere Einheiten mit reduzierter Komplexität zu unterteilen und somit das Lernen zu beschleunigen. Ein Beispiel für solch eine hierarchische Unterteilung einer Aufgabe ist in Abbildung 5.1 dargestellt. Das Lernen einer langen Folge von Aktionen kann beispielsweise häufig vereinfacht werden, indem repetitive Sequenzen von Aktionen zu Makrooperationen zusammengefasst werden. Komplexe Aufgaben können durch Verwendung dieser abstrakteren Aktionen gelöst werden. Durch Reduzierung der benötigten Anzahl von Schritten und der Entscheidungen, die auf den höheren Ebenen getroffen werden müssen, kann auf diese Weise das Lernen erheblich beschleunigt werden. Diese Art der Abstraktion wird oftmals als *zeitliche Abstraktion*[2] [67] bezeichnet. Eine weitere Art, die Komplexität einer Aufgabe zu reduzieren, ist es, diejenigen Dimensionen des Zustandsraumes zu identifizieren, die unwichtig für bestimmte Teilaufgaben sind. Somit können für die verschiedenen Teilaufgaben verschiedene Zustandsräume definiert werden, die nur die jeweils benötigten Variablen betrachten. Eine solche Strategie wird als *Abstraktion des Zustandsraumes*[3] bezeichnet. Schließlich kann eine Teilaufgabe häufig in verschiedenen Kontexten angewandt werden, so dass diese nicht mehrfach gelernt werden muss, sondern vorhandenes Wissen durch *Wiederverwendung*[4] effizient genutzt werden kann. Dies führt dazu, dass Erfahrungen beim Lernen auf weniger Aufgaben aufgeteilt werden müssen und somit zum Lernen pro Teilaufgabe mehr Informationen zur Verfügung stehen. Die

[1] engl.: curse of dimensionality
[2] engl.: temporal abstraction
[3] engl.: state abstraction
[4] engl.: subtask sharing

(a) Abstraktion in der Zeit

(b) Abstraktion im Zustandsraum



(c) Wiederverwendung von Teilaufgaben

Abbildung 5.2: *Beispielhafte Darstellungen der verschiedenen Abstraktionsarten beim hierarchischen Lernen. In Abbildung (a) wird die zeitliche Abstraktion beschrieben, bei der, statt viele kleine Schritte zu machen, auf höheren Ebenen größere Schritte gemacht werden können und somit weniger Entscheidungen gefällt werden müssen. In Abbildung (b) ist die Abstraktion des Zustandsraumes dargestellt. Diese beschreibt, dass zum Lernen nicht immer der gesamte Zustand benötigt wird, sondern für verschiedene Aufgaben auf verschiedenen Abstraktionsebenen unter Umständen nur ein kleiner Teil davon. Zum Beispiel sind zum Greifen eines Objektes nur die Informationen über dieses und die Oberfläche, auf der sich das Objekt befindet, wichtig. In Abbildung (c) ist die gemeinsame Nutzung von Teilaufgaben dargestellt. So kann der Roboter die gelernte Greifbewegung für verschiedene Arten von Aufgaben einsetzen beziehungsweise diese aus den verschiedenen Aufgaben lernen. Hier wird zum Beispiel die Greifbewegung sowohl für das Einkaufen im Supermarkt als auch zum Öffnen einer Flasche benutzt.*

verschiedenen Arten der Abstraktionen sind beispielhaft in Abbildung 5.2 dargestellt. Um diese Ideen in einem technischen Lernverfahren umzusetzen, schlug Dietterich das hierarchische MAXQ-Verfahren [68] vor. Dabei handelt es sich um einen Ansatz für Bestärkendes Lernen, der auf einer hierarchischen Zerlegung der Q-Funktion beruht und es ermöglicht, die genannten Arten von Abstraktionen in diskreten Räumen auszunutzen.

In diesem Kapitel wird der MAXQ-Ansatz auf kontinuierliche Zustandsräume angewendet, welche ein typisches Merkmal von praktischen Aufgaben sind, und das hierarchische Bestärkende Lernverfahren mit dem Lernen aus Demonstrationen in einem interaktiven hierarchischen Ansatz ergänzt. Um MAXQ für hierarchisch strukturierte Aufgaben mit kontinuierlichen Zustandsräumen anwenden zu können, wird Gauß-Prozess-Regression [24] eingesetzt, um kontinuierliche Approximationen der Q-Funktionen der einzelnen Teilaufgaben auf den unterschiedlichen Abstraktionsebenen zu bestimmen. Als Grundlage der Approximation werden die Erfahrungen des Systems genutzt, die bei der Evaluation von Aktionen gesammelt werden, die dem System demonstriert oder die autonom ausgewählt werden. Daraus werden probabilistische Schätzungen von Nutzenwerten über den Raum aus Zuständen und Aktionen berechnet, indem rekursiv die Schätzungen der zusammengesetzten und atomaren Teilaufgaben der Aufgabenhierarchie folgend aggregiert werden. Diese Schätzungen sind zentraler Bestandteil des Ansatzes, da die damit verbundenen Unsicherheiten die Sicherheit des Systems über das Ergebnis einer Aktion anhand des vorhandenen Wissens ausdrücken. Basierend auf den geschätzten Q-Werten und den dazugehörigen Unsicherheiten, berechnet das System im Bestärkenden Lernverfahren ein Bayessches Explorationskriterium, welches fundierte Entscheidungen für vielversprechende Aktionen trifft, anstatt mit einer Zufallsexploration zu arbeiten. Dadurch wird die benötigte Anzahl an Versuchen, die durchgeführt werden müssen, um Aktionen mit einem hohen Q-Wert zu finden, stark reduziert.

Um auch beim Lernen hierarchischer Aufgaben die Sicherheit des Roboters und seiner Umgebung zu gewährleisten, wird auch für diese Aufgaben der Wert der Erwarteten Veränderung als Entscheidungskriterium zwischen autonomem Lernen und Lernen aus Demonstrationen eingesetzt. Diese wird ebenfalls für das Lernen von Aktionssequenzen in Kapitel 4 eingesetzt und wird hier in den MAXQ-Algorithmus für hierarchisches Bestärkendes Lernen für die verschiedenen Teilaufgaben integriert. Dabei fließen in die Berechnung der Erwarteten Veränderung für eine Teilaufgabe die rekursiv entlang der MAXQ-Hierarchie berechneten Schätzungen der Q-Werte ein.

Um effizient geeignete Aktionen zur Lösung einer Aufgabe zu finden, benötigt das Bestärkende Lernverfahren mit der beschriebenen Bayesschen Explorationsstrategie Hintergrundwissen über die Aufgabe. Anstatt das System mittels einer Initialisierungsphase vorzutrainieren, wird ein interaktiver Ansatz genutzt, der dem System erlaubt, Demonstrationen für beliebige Teilaufgaben von einem Experten zu erfragen, wenn nicht genügend Wissen vorhanden ist. Durch diese Demonstrationen wird die Suche des Bestärkenden Lernens nach geeigneten Aktionen fokussiert und somit beschleunigt. Zudem wird der Arbeitsaufwand für das Demonstrieren von Lösungen gering gehalten, indem die iterative Auswahl der geeignetsten Lernmethode für jede einzelne Teilaufgabe auf jeder Abstraktionsebene der hierarchischen Aufgabe einzeln getroffen wird. Des Weiteren sind die Aktionen, die zur Demonstration der Lösung einer hierarchischen Teilaufgabe vorgeführt werden müssen, auf eine Hierarchieebene beschränkt. Sofern genügend Wissen vorhanden ist, versucht das System die darin enthaltenen Teilaufgaben autonom zu lösen. Schließlich wird durch die beobachteten Demonstrationen eine Präferenz für die autonome Bayessche Suche nach geeigneten Aktionen ausgedrückt, welche diejenigen Aktionen des Roboters bevorzugt, die für einen menschlichen Mitarbeiter vorhersagbar sind. Dies ist besonders vorteilhaft für Anwendungen, in denen Roboter eng mit Menschen zusammenarbeiten.

5.1.1 Verwandte Arbeiten

Im Laufe der Zeit haben Forscher verschiedene Möglichkeiten untersucht, um hierarchische Strukturen und Abstraktionen zum Skalieren von Bestärkendem Lernen in komplexen Anwendungen für Roboter zu verwenden [69, 70].

Eine Reihe von Arbeiten konzentriert sich dabei auf zeitliche Abstraktion, indem mehrere Aktionen zu sogenannten Makro-Aktionen oder Options [60] zusammengefasst werden. Somit werden komplexe Strategien in eine Reihe einfacher Bewegungsabläufe und eine Strategie zur situationsabhängigen Aktionswahl zerlegt. Eine Reihe von Arbeiten betrachtet hierbei parametrische Formen von Makro-Aktionen und untersucht Methoden, die deren Parameter durch die Optimierung der langfristigen Belohnung verbessern. Daniel et al. [71] schlagen eine hierarchische Erweiterung der Relative Entropy Policy Search (REPS) [72] Methode vor, um verschiedene Aktionen zur Lösung einer Aufgabe in unterschiedlichen Situationen zu lernen. Dabei verwenden sie dynamische Bewegungsprimitive (DMP) [18] zur Beschreibung von Aktionen und schlagen einen Optimierungsansatz vor, um sowohl die Parameter als auch eine Strategie zur Auswahl einer geeigneten DMP zu bestimmen. In einer weiteren Arbeit [55] erweitern die Autoren diesen Ansatz, um auch Sequenzen einfacher Bewegungen lernen zu können. Stulp und Schaal [73] schlagen einen Ansatz basierend auf dem PI^2-Algorithmus [58] vor, um gleichzeitig sowohl die Zwischenziele einer Sequenz von Bewegungsprimitiven als auch deren Formparameter zu optimieren. Auch hier werden Bewegungen durch DMP repräsentiert. Die Parametrisierung von Bewegungen durch dynamische Systeme erlaubt es beiden Ansätzen, auch Aufgaben zu lernen, für die kontinuierliche Zustands- und Aktionsräume benötigt werden. Weiterhin nutzen beide Ansätze überwachtes Lernen aus Demonstrationen, um die jeweiligen Parameter zu initialisieren. In beiden Fällen werden jedoch nur zweistufige Strategien und Abstraktionen durch zeitliche Zusammenfassung von Aktionen betrachtet. Darüber hinaus muss die Länge der Aktionssequenzen jeweils im Voraus bekannt sein, was die Flexibilität der Systeme bei der Wahl der optimalen Aktionssequenz im Gegensatz zu dem in diesem Kapitel vorgestellten Verfahren einschränkt.

Konidaris et al. [61] nutzen anstelle parametrisierter Strategien eine Approximation der Nutzenfunktion zur impliziten Repräsentation von Bewegungen. Darüber hinaus ermöglichen sie die Verwendung von Zustandsabstraktionen, indem sie jeder Bewegung einen individuellen Zustandsraum aus einer vorgegebenen Liste von Abstraktionen zuordnen und so die Komplexität der einzelnen Nutzenfunktionen reduzieren. Die einzelnen Bewegungen sowie die ihnen zugeordneten Abstraktionen werden dabei aus demonstrierten Trajektorien gelernt. Die Zuordnung zu verschiedenen Abstraktionen liefert gleichzeitig ein Kriterium zur Segmentierung der Trajektorien. Die so bestimmten Segmente werden unter Berücksichtigung ihrer statistischen Ähnlichkeiten in umgekehrter Reihenfolge zusammengefasst und zu längeren Bewegungsfolgen verknüpft. Mehrere Bewegungsfolgen, die aus unterschiedlichen Situationen zu demselben Zielzustand führen, bilden dabei eine Baumstruktur aus verschiedenen Strategien zur Lösung einer Aufgabe. Um die aus Demonstrationen gelernten Teilbewegungen sowie die Übergänge zwischen ihnen weiter zu verbessern, schlagen die Autoren den Einsatz von Bestärkenden Lernverfahren vor.

Die bisher vorgestellten Ansätze um hierarchische Strukturen in Bestärkenden Lernverfahren zu nutzen folgen dem Options-Formalismus, bei dem Strategien zur Lösung verschiedener Teilaufgaben jeweils unabhängig voneinander repräsentiert werden. Das

MAXQ-Verfahren stellt hierzu einen alternativen Ansatz dar, der es durch eine geschickte Zerlegung der Q-Funktion ermöglicht, weitere Formen von Abstraktionen im Bestärkenden Lernen zu nutzen. Zu der ursprünglichen MAXQ-Methode werden unterschiedliche Erweiterungen vorgeschlagen, um die Skalierbarkeit des Verfahrens und dessen Leistungsfähigkeit durch die Integration von Vorwissen zu verbessern. In verschiedenen Arbeiten werden hierzu modellbasierte Ansätze entwickelt, in denen Vorwissen als a priori-Verteilung über die Modellparameter eingebracht werden kann. Cao et al. [74] erweitern das MAXQ-Verfahren um Modelle der Belohnungen und der Übergangswahrscheinlichkeiten für primitive Aktionen. Dies erlaubt es, Vorwissen über die Dynamik und die Belohnungsstruktur der Umgebung in den Lernprozess einfließen zu lassen. Da diese Informationen jedoch nur auf der Ebene atomarer Aktionen zur Verfügung stehen, schlagen die Autoren eine Erweiterung des MAXQ-Lernverfahrens zu einem hybriden Algorithmus vor, der modellbasierte und modellfreie Elemente vereint. Nach jeder atomaren Aktion wird dabei die a posteriori-Verteilung der Modellparameter anhand des beobachteten Zustandsübergangs und der erhaltenen Belohnung aktualisiert. In festen Intervallen werden gemäß dieser Verteilung neue Parameterwerte bestimmt. Mit diesen werden anschließend die Werte der Komponenten der MAXQ-Zerlegung rekursiv entsprechend den Vorschriften des regulären MAXQ-Lernalgorithmus aktualisiert. Die Autoren zeigen empirisch, dass der Lernprozess durch die Einführung von Modellen beschleunigt wird. Der Vorteil wächst dabei mit der Genauigkeit der zur Verfügung stehenden a priori-Informationen. Der in diesem Kapitel vorgestellte Ansatz unterscheidet sich von dem hybriden, modellbasierten Ansatz von Cao et al. dadurch, dass er die ursprüngliche modellfreie Struktur der MAXQ-Methode beibehält und eine Bayessche Explorationsstrategie vorschlägt. Vorhandenes Expertenwissen wird in dieser Arbeit in Form von demonstrierten Aufgabenlösungen anstatt durch a priori-Verteilungen über Modellparameter integriert.

Jong und Stone [75] schlagen eine hierarchische Zerlegung der Belohnungs- und Übergangsmodelle für alle Teilaufgaben der Hierarchie vor und erweitern den MAXQ-Ansatz um die optimistische Exploration des modellbasierten R-MAX-Algorithmus [76]. Der resultierende Algorithmus mit dem Namen Fitted R-MAXQ nutzt kernelbasierte Funktionsapproximationstechniken zur Generalisierung der Modellparameter auf unbeobachtete Zustands- und Aktionspaare in Kombination mit einer iterativen Approximation der Nutzenfunktion [77], um Strategien für Teilaufgaben in kontinuierlichen Zustands- und Aktionsräumen zu lernen. Während bei der Explorationsstrategie von R-MAX unbekannte Zustands- und Aktionspaare begünstigt werden, wird in dieser Arbeit eine Explorationsstrategie verfolgt, die zwischen der Optimierung der Belohnungen und dem Risiko, unbekannte Aktionen auszuführen, abwägt.

Bai et al. [78] bauen in ihrer Arbeit ebenfalls auf der MAXQ-Methode auf und wenden diese in kontinuierlichen Zustands- und Aktionsräumen an, indem sie eine explizite Repräsentation der Komponenten der MAXQ-Zerlegung vermeiden. Stattdessen nutzen sie einen Online-Algorithmus, der rekursiv den Nutzen von besuchten Zuständen entsprechend der MAXQ-Zerlegung berechnet. Um diese Berechnung effizient durchführen zu können, schlagen die Autoren vor, die erwartete Belohnung für das Beenden einer Teilaufgabe nach Ausführung der nächsten Aktion anhand einer a priori-Verteilung der Zielzustände dieser Aktion zu approximieren. Die Verfügbarkeit von Wissen über die

Aufgabe und die Umgebung zur Bestimmung der a priori-Verteilung der Zielzustände hat dabei maßgeblichen Einfluss auf die Ergebnisse dieses Verfahrens. Die resultierende Rekursion wird anhand einer Tiefenbegrenzung und heuristischer Aktionsauswahltechniken optimiert. Im Gegensatz zu dem Ansatz von Bai et al. nutzt der hier vorgestellte Ansatz Demonstrationen der Aufgabe, um dem System Expertenwissen zu vermitteln. Durch Generalisierung der Informationen von besuchten Zuständen vermeidet der Ansatz in diesem Kapitel ebenfalls die Vorberechnung vollständiger Strategien zur Lösung von Teilaufgaben, ähnlich zu einem Online-Ansatz. Doch im Gegensatz zu den ad-hoc Berechnungen, die in dem Ansatz von Bai et al. durchgeführt werden, verwendet das in diesem Kapitel vorgestellte System beobachtete Instanzen, um zukünftige Entscheidungen zu verbessern.

5.1.2 Wissenschaftlicher Beitrag

In diesem Kapitel wird die MAXQ-Methode für hierarchisches Bestärkendes Lernen für den Einsatz in kontinuierlichen Zustandsräumen erweitert. Zusätzlich wird darauf aufbauend ein Lernalgorithmus vorgestellt, der den MAXQ-Algorithmus für hierarchisches Bestärkendes Lernen in kontinuierlichen Zustandsräumen mit dem Lernen aus Demonstrationen von Teilaufgaben integriert.

Um die Anwendung des MAXQ-Lernalgorithmus auf kontinuierliche Zustandsräume zu ermöglichen, werden die Komponenten der MAXQ-Zerlegung der Q-Funktion durch Gauß-Prozesse approximiert. Die Nutzenfunktionen von Zustands- und Aktionspaaren der einzelnen Teilaufgaben auf den verschiedenen Abstraktionsebenen der MAXQ-Hierarchie werden dabei jeweils durch unabhängige Gauß-Prozesse modelliert. Um ein effizientes Bestärkendes Lernen zu ermöglichen, setzt der Ansatz innerhalb des MAXQ-Lernalgorithmus ein Bayessches Explorationskriterium ein, das auf der Optimierung der Erwarteten Veränderung aus Kapitel 4 basiert.

Zur Berechnung der Erwarteten Veränderung wird eine probabilistische Schätzung der Q-Funktion aus den Gauß-Prozess-Approximationen ihrer Komponenten bestimmt. Dazu werden die Erwartungswerte und Unsicherheiten von durch Gauß-Prozess-Regression bestimmten Schätzungen rekursiv entlang der MAXQ-Zerlegung der Q-Funktion aggregiert. In den vorangegangenen Kapiteln konnte bereits gezeigt werden, dass durch die Berücksichtigung von Unsicherheiten bei der Exploration und die Kombination der Erwarteten Verbesserung und Verschlechterung in einem gemeinsamen Kriterium vielversprechende Aktionen gewählt werden, während gleichzeitig die Sicherheit des Roboters und seiner Umgebung gewährleistet wird. Durch die rekursive Bestimmung probabilistischer Schätzungen der Q-Funktion ermöglicht es der hier vorgestellte Ansatz, auch in einem hierarchischen Bestärkenden Lernverfahren von diesen Vorteilen zu profitieren.

In dem hier vorgestellten Lernverfahren wird das Bestärkende Lernen als eines von zwei komplementären Modulen eingesetzt. Ergänzt wird es durch ein interaktives Modul, das dem System erlaubt, Demonstrationen von einem Experten anzufordern, um anhand dieser die Lösung einer Teilaufgabe zu erlernen. Die Entscheidung zwischen diesen beiden Modulen wird in jeder Iteration des Verfahrens und für alle Teilaufgaben auf den verschiedenen Ebenen der Zerlegungen individuell durch ein Entscheidungsmodul getroffen. Dieses nutzt ebenfalls den rekursiv bestimmten Q-Wert, um abzuwägen, welches

der beiden Lernverfahren das geeignetere ist, um eine sichere und effektive nächste Aktion zur Lösung der aktuellen Teilaufgabe zu bestimmen. Indem das System gezielt Demonstrationen für einzelne Teilaufgaben auf bestimmten Ebenen anfordern kann, für die nicht genug Erfahrungen für eine sichere autonome Lösung vorliegen, werden unnötige Demonstrationen bereits gelernter Aufgabenteile vermieden und der Aufwand für den Experten reduziert. Bei der Demonstration wird der angefragte Aufgabenteil durch die Aktionen der darunter liegenden Ebene vorgeführt, so dass unnötig komplexe Demonstrationen über mehrere Ebenen vermieden werden.

Durch eine modulare Architektur kann für alle Teilaufgaben auf den unterschiedlichen Abstraktionsebenen der MAXQ-Hierarchie das kombinierte Lernverfahren auf die gleiche Weise eingesetzt werden, so dass ein leicht zu skalierendes System entsteht, welches um weitere Abstraktionsebenen zum Lösen komplexerer Aufgaben erweiterbar ist. Hierfür müssen lediglich die Parameter für die neuen Ebenen, wie die Beschreibung der Situationen und Aktionen, sowie die Zuteilung der Belohnungen, angepasst werden.

Das Kapitel hat den folgenden Aufbau: In Abschnitt 5.2 wird das kombinierte Verfahren zum Lernen hierarchischer Aufgaben mit kontinuierlichen Zustandsräumen erläutert. Dazu werden zunächst die Version des MAXQ-Algorithmus mit Gauß-Prozessen und die sich dadurch ergebenden Vorteile vorgestellt, anschließend werden die beiden auf jeder Ebene eingesetzten Lernverfahren, das Lernen aus Demonstrationen und das Bestärkende Lernen, beschrieben. Zuletzt wird erklärt, wie die Entscheidung zwischen den beiden Verfahren in jeder Situation und auf jeder Ebene getroffen wird. Im darauffolgenden Abschnitt 5.3 wird zunächst die Aufgabe, verschieden hohe Stapel von Boxen zu versetzen, wie auch die hierarchische Zerlegung dieser, insbesondere im Zustands- und Aktionsraum und bei den zu vergebenen Belohnungen, erläutert und anschließend das hierarchische System evaluiert. Im Abschnitt 5.4 wird der hierarchische Ansatz zusammengefasst und die Ergebnisse der in diesem Kapitel vorgestellten Arbeiten diskutiert.

5.2 Hierarchisches Lernverfahren für sequentielle Aufgaben

In diesem Kapitel wird ein integrierter Ansatz für das hierarchische Lernen mit Robotern in einer komplexen Domäne vorgeschlagen. Dabei wird auf der MAXQ-Methode für hierarchisches Bestärkendes Lernen aufgebaut. Dieses Verfahren, das ursprünglich für diskrete Umgebungen entwickelt wurde, wird in diesem Kapitel für die Verwendung in kontinuierlichen Zustandsräumen erweitert. Dazu werden Gauß-Prozess-Approximationen der Komponenten der MAXQ-Zerlegung der Q-Funktion für zusammengesetzte und atomare Teilaufgaben verwendet. Mit Hilfe dieser Approximation berechnet das System Schätzungen der Nutzenfunktion über Zustands- und Aktionsräume durch rekursives Aggregieren von Gauß-Prozess-Vorhersagen und deren Unsicherheiten über die Hierarchie der Teilaufgaben. Zur Bestimmung geeigneter Lösungsstrategien für die einzelnen Teilaufgaben im Bestärkenden Lernverfahren wird jeweils ein Bayessches Explorationskriterium basierend auf dem Wert der Erwarteten Veränderung verwendet. Dieses implementiert eine Abwägung zwischen der Optimierung des Q-Wertes und der Vermeidung potenzieller Risiken für den Roboter, wobei Informationen über Aktionen auf tieferen Hierarchieebenen berücksichtigt werden. Diese Strategie führt zu einem effizienten und sicheren

Lernverhalten des hier vorgestellten Systems.

Des Weiteren wird der MAXQ-Algorithmus mit dem Lernen aus Demonstrationen kombiniert, um durch Einbeziehung von Expertenwissen die Suche des Bestärkenden Lernens zu fokussieren. Dazu wird es dem hier vorgestellten System erlaubt, nach Demonstrationen eines Experten zu fragen, wenn die zuvor gesammelten Erfahrungen nicht ausreichend sind, um eine Aktion vorzuschlagen, die mit hoher Wahrscheinlichkeit zu einer guten Belohnung führt. Die Entscheidung für eine der beiden Arten des Lernens wird inkrementell für jede auftretende Situation und für jede Teilaufgabe in der Hierarchie getroffen. Dies führt zu einem kohärenten System, in dem die gleiche Logik in allen Teilaufgaben auf jeder Abstraktionsebene angewendet wird. Das bringt den Vorteil mit sich, dass das System leicht für komplexere Aufgaben um weitere Ebenen erweiterbar ist, da jede Ebene gleich aufgebaut ist und immer die gleichen Algorithmen benutzt werden. Indem der Ansatz die Stärken beider Lernverfahren, des Lernens aus Demonstrationen und des Bestärkenden Lernens, ausnutzt, ist der kombinierte Ansatz in der Lage, Lösungen für Aufgaben mit einem großen Zustandsraum schneller als mit Bestärkendem Lernen alleine und mit angemessenem Aufwand seitens des Experten zu lernen.

5.2.1 Kontinuierliches MAXQ-Lernen mit Unsicherheiten

Die *MAXQ-Methode* wurde von Dietterich [68] vorgeschlagen, um durch Bestärkendes Lernen Strategien für hierarchisch strukturierte Markov-Entscheidungsprobleme [41] zu lernen. Dazu wird durch den MAXQ-Ansatz die Nutzung von Abstraktionen, wie sie in der Einleitung beschrieben wurden, im Rahmen des Q-Lernens [42] ermöglicht. Dies erhöht die Skalierbarkeit dieser Methode, was für reale Aufgaben erforderlich ist. Die Kernidee des Ansatzes ist dabei, dass sich die Q-Funktion einer hierarchischen Strategie zum Lösen einer Teilaufgabe in eine Summe aus zwei Komponenten zerlegen lässt. Die erste Komponente enthält die erwartete Belohnung für das Ausführen der unmittelbaren Aktion. Die zweite Komponente beschreibt die erwartete kumulative Belohnung für das Lösen der Teilaufgabe nach Abschluss dieser unmittelbaren Aktion. Wird die unmittelbare Aktion, die ihrerseits aus mehreren Schritten bestehen kann, als eigenständige Teilaufgabe aufgefasst, entspricht die erste Komponente dem Wert der Nutzenfunktion dieser Teilaufgabe, wobei Aktionen anhand einer gegebenen Strategie ausgewählt werden. Der Notation von [68] folgend, kann dies wie folgt ausgedrückt werden. Sei s der aktuelle Zustand und a eine durch die Strategie $\pi_i(s)$ ausgewählte Aktion innerhalb der Teilaufgabe i. Dann beschreibt die Q-Funktion $Q^\pi(i, s, a)$ einer Strategie π den erwarteten Gewinn für das Ausführen der Aktion a in der Situation s und die anschließende Vollendung der Teilaufgabe i durch Aktionen, die gemäß der hierarchischen Strategie π gewählt werden. Diese Funktion lässt sich mit Hilfe der MAXQ-Zerlegung in die Summe der Nutzenfunktion $V^\pi(a, s)$ der Teilaufgabe a von i und der erwarteten kumulativen Belohnung $C^\pi(i, s, a)$ unterteilen, die den Nutzen für das Abschließen der Teilaufgabe i nach Ausführung von a beschreibt. In diesem Zusammenhang wird $C^\pi(i, s, a)$ als Abschlussnutzenfunktion[5] der Teilaufgabe i in der Situation s für die Aktion a bezeichnet. Auf diese Weise ergibt sich eine rekursive Zerlegung der Q-Funktion einer gelernten Strategie entlang der hierarchischen Struktur

[5] engl.: completion function

der zu lernenden Aufgabe:

$$
\begin{aligned}
Q^{\pi}(i,s,a) &= V^{\pi}(a,s) + C^{\pi}(i,s,a), \\
V^{\pi}(i,s) &= \begin{cases} Q(i,s,\pi_i(s)) & \text{falls } i \text{ zusammengesetzt} \\ \sum_{s'} P(s'|s,i)R(s'|s,i) & \text{falls } i \text{ atomar} \end{cases}
\end{aligned} \tag{5.1}
$$

Dabei beschreibt R die Belohnung eines einzelnen Schrittes, die das System erhält, wenn es eine atomare Aktion i in einem Zustand s ausführt und dieses zu einem Übergang in den Zustand s' führt und P die Wahrscheinlichkeit des Übergangs nach s' ist. Stellt $m_0, m_1, ..., m_k$ den Pfad der ausgewählten Teilaufgaben dar, der mit m_0 auf der obersten Ebene einer Teilaufgabe beginnt und mit m_k als atomarer Aktion auf der untersten Ebene endet, und sei für jede Teilaufgabe m_i auf einer Ebene i die Aktion a_i anhand der hierarchischen Strategie $\pi(s) = a_i$ bestimmt, so kann $Q^{\pi}(m_0, s, \pi(s))$ nach Gleichung (5.1) wie folgt bestimmt werden:

$$
\begin{aligned}
& Q^{\pi}(m_0, s, \pi(s)) \\
&= \overbrace{V^{\pi}(m_1, s) + C^{\pi}(m_0, s, a_0)} \\
&= \overbrace{V^{\pi}(m_2, s) + C^{\pi}(m_1, s, a_1)} + C^{\pi}(m_0, s, a_0) \\
&\vdots \\
&= V^{\pi}(m_k, s) + C^{\pi}(m_{k-1}, s, a_{k-1}) + \ldots + C^{\pi}(m_1, s, a_1) + C^{\pi}(m_0, s, a_0)
\end{aligned} \tag{5.2}
$$

Analog zu vielen anderen Ansätzen für hierarchisches Bestärkendes Lernen [67] wird auch durch das MAXQ-Verfahren die Komplexität der Lernaufgabe verringert, indem diese in Teilaufgaben zerlegt wird, die von Teilaufgaben höherer Abstraktionsebenen in einem einzigen Aktionsschritt ausgeführt werden können. Eines der wichtigsten Merkmale von MAXQ besteht jedoch darin, dass dieses Verfahren nicht nur mit einer Hierarchie aus Teilaufgaben arbeitet, sondern mit der MAXQ-Zerlegung auch eine entsprechende Gliederung der Q-Funktion einführt. Dadurch wird die Nutzenfunktion der Gesamtaufgabe in Nutzenfunktionen für die individuellen Teilaufgaben unterteilt. Diese Repräsentation führt dazu, dass die Nutzenfunktionen der Teilaufgaben kontextfrei sind, was bedeutet, dass der Nutzen einer Aktion in einer Teilaufgabe nicht von dem Kontext abhängt, in dem die Aufgabe ausgeführt wird. Gelernte Nutzenfunktionen können somit für verschiedene Aufgaben wiederverwendet werden. Gleichzeitig können Erfahrungen, die bei der Anwendung einer Aktion in verschiedenen Aufgaben gesammelt werden, zum Lernen dieser Aktion kombiniert werden. Da in der Regel nicht der gesamte Zustandsraum für alle Teilaufgaben relevant ist, führt die hierarchische Struktur gleichzeitig dazu, dass große Teile des Zustandsraumes in den einzelnen Teilaufgaben vernachlässigt werden können. Somit wird sowohl der Speicheraufwand für die Nutzenfunktionen beziehungsweise die Abschlussnutzenfunktionen reduziert, als auch der Aufwand, der nötig ist, um eine gegebene Aufgabe zu erlernen. Diese Reduktion der Dimensionen des Zustandsraumes führt dazu, dass mehrere Situationen des ursprünglichen Zustandsraumes auf denselben

Zustand im reduzierten Zustandsraum einer Teilaufgabe abgebildet werden. Somit wird die Effizienz des Verfahrens erhöht, da die zur Verfügung stehenden Informationen zum Lernen einer geringeren Anzahl an Paaren von Zuständen und Aktionen verwendet werden können.

Zusammen mit der Zerlegung der Nutzenfunktion für Aktionen in eine Summe aus Abschlussnutzenfunktionen zusammengesetzter Aufgaben und Nutzenfunktionen atomarer Aufgaben präsentieren Dietterich et al. eine Variante des Q-Lernalgorithmus, um die Komponenten dieser Zerlegung zu lernen. Dazu führt dieser, beginnend in einem beliebigen Zustand einer beliebigen Teilaufgabe, rekursiv Aktionen gemäß einer vom Benutzer festzulegenden Explorationsstrategie aus. Nach jeder atomaren Aktion wird die Schätzung über den Nutzen dieser Aktion entsprechend der erfahrenen Belohnung aktualisiert. Sobald eine Teilaufgabe abgeschlossen ist, wird die Abschlussnutzenfunktion der entsprechenden Aktion in der übergeordneten Aufgabe mit dem erwarteten Gewinn des neuen Zustandes in der Elternaufgabe aktualisiert. Die entsprechenden Gleichungen hierfür lauten:

$$\begin{aligned} V_{t+1}(i,s) &= (1-\alpha_t(i)) \cdot V_t(i,s) + \alpha_t(i) \cdot r_t \\ C_{t+1}(i,s,a) &= (1-\alpha_t(i)) \cdot C_t(i,s,a) + \alpha_t(i) \cdot \gamma^N V_t(i,s') \end{aligned} \tag{5.3}$$

Dabei bezeichnet $\alpha_t(i)$ die zeitabhängige Lernrate für eine Teilaufgabe i, der Wert γ^N bezeichnet den entsprechend der Länge einer ausgeführten Teilaufgabe potenzierten Abklingfaktor und r_t die Belohnung, die das System für die Ausführung einer primitiven Aktion erhält. Da die Nutzenfunktion $V_t(i,s)$, die in der obigen Gleichung zur Aktualisierung der Werte der Abschlussnutzenfunktion benötigt wird, nur für atomare Aktionen explizit repräsentiert wird, muss ihr Wert für zusammengesetzte Aktionen rekursiv berechnet werden, indem die Gleichungen für die MAXQ-Zerlegung (5.1) rekursiv aufgelöst werden. Dabei wird die Strategie π durch den max-Operator ersetzt, so dass in jedem Schritt die beste Aktion zur Berechnung des erwarteten Gewinns ausgewählt wird:

$$\begin{aligned} V_t(i,s) &= \begin{cases} \max\limits_a Q_t(i,s,a) & \text{falls } i \text{ zusammengesetzt} \\ V_t(i,s) & \text{falls } i \text{ atomar} \end{cases} \\ Q_t(i,s,a) &= V_t(a,s) + C_t(i,s,a) \end{aligned} \tag{5.4}$$

Die Indizes t deuten dabei an, dass sich die Funktionen über die Zeit ändern, da in jeder Iteration des Lernverfahrens neue Informationen einfließen, die dazu beitragen, dass die Funktionen sich iterativ den optimalen Funktionen $V^*(i,s)$ beziehungsweise $Q^*(i,s,a)$ annähern.

In hierarchisch strukturierten Aufgaben werden somit auf jeder Ebene Entscheidungen für Aktionen getroffen, während die Belohnungen typischerweise nur auf der untersten Ebene vergeben werden. Somit kann es dazu kommen, dass der Lernalgorithmus trotz korrekter Entscheidungen in einer atomaren Aufgabe eine schlechte Belohnung bekommt, die durch eine zuvor getroffene Entscheidung auf einer höheren Ebene verursacht wurde und die das System durch atomare Aktionen nicht beeinflussen kann. Um diese hierarchische Form des Zuweisungsproblems von Belohnungen[6] während des Lernens zu vermeiden, kann der Designer der MAXQ-Hierarchie verschiedene Arten der Belohnung verschiedenen Teilaufgaben auf verschiedenen Aufgabenebenen zuordnen.

[6] engl.: hierarchical credit assignment problem

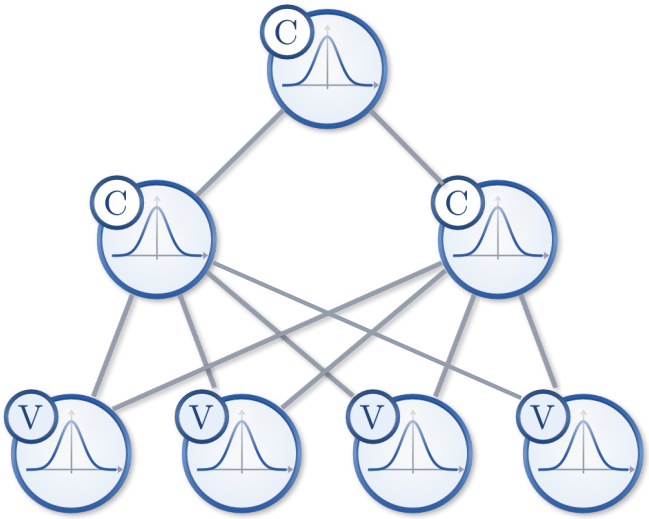


Abbildung 5.3: *Darstellung einer möglichen MAXQ-Zerlegung einer Aufgabe mit der dazugehörigen Zerlegung der Q-Funktion in Nutzen- und Abschlussnutzenfunktionen. Auf der obersten Ebene ist die zu lösende Gesamtaufgabe dargestellt, welche mit Hilfe der auf der mittleren Ebene gegebenen zwei Teilaufgaben gelöst werden kann. Zum Lösen dieser liegen wiederum vier verschiedene Teilaufgaben auf der untersten Ebene vor, die primitive Aktionen darstellen. Die Nutzenfunktionen beziehungsweise die Abschlussnutzenfunktionen jeder der durch Kreise dargestellten Aufgaben oder Teilaufgaben werden durch einen eigenen Gauß-Prozess modelliert. Für Aufgaben, die mit dem Buchstaben V versehen sind, wird die Nutzenfunktion der ausgeführten Aktion approximiert, während für Aufgaben, mit einem C gekennzeichnet sind, die Abschlussnutzenfunktionen beschrieben werden.*

Das Lernen einer hierarchischen Strategie mit dem MAXQ-Ansatz erfordert, dass die Werte der Abschlussnutzenfunktionen aller zusammengesetzten Teilaufgaben und die Werte der Nutzenfunktionen aller atomaren Aufgaben gespeichert werden. In Anwendungen mit kontinuierlichen Zustandsräumen ist es unmöglich, diese Werte für alle Zustände explizit zu repräsentieren. Zusätzlich ist es in praktischen Anwendungen in der Regel der Fall, dass ein Bestärkender Lernagent nur einen kleinen Teil des Zustands- und Aktionsraumes besucht. Zum einen können nicht alle theoretisch möglichen Konfigurationen in der Praxis eingenommen werden, zum anderen ist der Aufwand hierfür zu groß, wodurch der Lernprozess langsam und ineffizient würde. Daher ist es sinnvoller, die vorhandenen Ressourcen auf diejenigen Bereiche des Zustands- und Aktionsraumes zu konzentrieren, die für die zu lösende Aufgabe relevant sind, und diese Informationen für ähnliche Zustände und Aktionen auszunutzen. Eine übliche Technik ist es daher, die Nutzenfunktion mit Hilfe von Funktionsapproximationstechniken zu repräsentieren. Funktionsapproximatoren haben zum einen den Vorteil, dass sie gegenüber einer tabellarischen Darstellung eine Repräsentation der gesuchten Funktion mit vergleichsweise wenigen Parametern ermöglichen. Zum anderen liefern sie anhand weniger Trainingsbeispiele Informationen über den Verlauf der Nutzenfunktion in weiten Teilen des Zustands- und Aktionsraumes. Darüber hinaus werden durch die Wahl des Regressionsmodells

implizit Annahmen über die Struktur der zu approximierenden Funktion getroffen. Diese erlauben es, durch Generalisierung der Trainingsdaten eine gerichtete Exploration im Bestärkenden Lernen zu erreichen. Dieses Kapitel beschäftigt sich mit der Anwendung von Funktionsapproximationen im Kontext des hierarchischen Bestärkenden Lernens. Dazu wird Gauß-Prozess-Regression zum Einsatz der MAXQ-Methode in kontinuierlichen Zustandsräumen verwendet.

Gauß-Prozess-Regression ist eine Technik für nicht-parametrische Bayessche Regression, die Vorhersagen über gesuchte Funktionswerte für beliebige Eingabedaten liefert, zusammen mit einer dazugehörigen Unsicherheit der Schätzung. Gegenüber anderen Regressionsverfahren bietet Gauß-Prozess-Regression durch diese Unsicherheiten den Vorteil, dass Lernverfahren die Verlässlichkeit der Schätzungen des erwarteten Nutzens in ihren Entscheidungen berücksichtigen können. Die Details der Regression mit Gauß-Prozessen werden in Abschnitt 3.1 beschrieben. Da im MAXQ-Ansatz die Werte der Abschlussnutzenfunktionen zusammengesetzter Aufgaben und die Nutzenfunktionen atomarer Aufgaben zur Berechnung der MAXQ-Zerlegung der Nutzenfunktion der Gesamtaufgabe gespeichert werden müssen, approximiert das System diese in dem hier vorgestellten Ansatz durch Gauß-Prozesse. Dabei wird für jede Teilaufgabe a auf jeder Ebene i ein separates Gauß-Prozess-Modell für die jeweils zu speichernde Funktion verwendet. Somit können durch Anwendung von Gauß-Prozess-Regression auf diesen Modellen Schätzungen des Abschlussnutzens beziehungsweise der Nutzenwerte für beliebige Paare aus Zuständen und Aktionen und Teilaufgaben in geschlossener Form berechnet werden. Die Funktionen, die auf den unterschiedlichen Ebenen einer MAXQ-Zerlegung durch Gauß-Prozesse approximiert werden, und deren Repräsentationen in den jeweiligen Teilaufgaben vorgehalten werden müssen, sind in Abbildung 5.3 anhand eines Beispiels einer hierarchischen Aufgabe dargestellt.

Die Mittelwerte der Schätzungen erlauben die rekursive Berechnung der Q-Werte für beliebige Teilaufgaben gemäß Gleichung (5.1). Diese Werte werden im Rahmen des Bestärkenden Lernens zur Bestimmung auszuführender Aktionen durch die Explorationsstrategie und zur Aktualisierung der Trainingsdaten nach Gleichung (5.4) benötigt. Allerdings wird durch diese Art der Berechnung einer der wichtigsten Vorteile der Modellierung durch Gauß-Prozesse ignoriert. Dieser besteht darin, dass durch Gauß-Prozess-Regression nicht nur der wahrscheinlichste Funktionswert für ein Paar aus Zustand und Aktion bestimmt wird, sondern eine vollständige Normalverteilung des Funktionswertes, gegeben durch Mittelwert und Varianz. Um diese zusätzlichen Informationen im Lernverfahren zur Verbreiterung der Entscheidungsgrundlage für zu explorierende Aktionen nutzen zu können, werden auch die Unsicherheiten der Gauß-Prozess-Schätzungen entlang der Hierarchieebenen der MAXQ-Zerlegung propagiert. Auf diesem Weg erhält das System eine probabilistische Schätzung, die eine vollständige Verteilung des Q-Wertes für ein Paar aus Zustand und Aktion beinhaltet. Die Berechnung erfolgt analog zur Gleichung (5.1):

$$Q(i,s,a) \sim \mathcal{N}\begin{pmatrix}\mu_{Q(i,s,a)}\\ \sigma^2_{Q(i,s,a)}\end{pmatrix} = \begin{pmatrix}\mu_{V(a,s)} + \mu_{C(i,s,a)}\\ \sigma^2_{V(a,s)} + \sigma^2_{C(i,s,a)}\end{pmatrix} \tag{5.5}$$

Hier wird $V(a,s)$ rekursiv in die Summe der normalverteilten Gauß-Prozess-Schätzungen für die Werte der Abschlussnutzenfunktionen für zusammengesetzte Aufgaben und für die Nutzenwerte von atomaren Aktionen zerlegt. Aufgrund der Generalisierungsfähig-

keit der Gauß-Prozess-Approximation steigt die Genauigkeit der Schätzungen, je mehr Erfahrungen das System durch die Interaktion mit der Umgebung sammelt, und das System generiert mit der Zeit immer bessere Aktionen zum Lösen einer Aufgabe. Wie die kombinierten Gauß-Prozess-Schätzungen zu einer effizienten, probabilistischen Explorationsstrategie zur Bestimmung von auszuführenden Aktionen beitragen, wird im nächsten Abschnitt beschrieben.

5.2.2 Hierarchisches Bestärkendes Lernen mit Erwarteter Veränderung

Der MAXQ-Algorithmus für kontinuierliche Zustandsräume, wie er in Abschnitt 5.2.1 beschrieben wird, ist eine Adaption des Q-Lernalgorithmus für die hierarchische Zerlegung der Q-Funktion in eine Summe aus Abschlussnutzenfunktionen für zusammengesetzte Aufgaben und einer Nutzenfunktion für atomare Aufgaben. Zum Lernen benötigt dieses Verfahren eine vom Benutzer zu definierende Explorationsstrategie, anhand derer in jeder Iteration eine auszuführende Aktion bestimmt wird. Während für theoretische Untersuchungen, etwa über das Konvergenzverhalten von Lernalgorithmen, Strategien mit einer zumindest teilweise zufälligen Aktionsauswahl interessant sind, werden für praktische Anwendungen Strategien benötigt, die in wenigen Iterationen erfolgreiche Aktionen finden. Dabei sollen sie zusätzliche Kriterien wie die Sicherheit des Roboters und seiner Umgebung oder die Vorhersagbarkeit der Ergebnisses berücksichtigen. Eine Strategie für solche praktischen Anwendungen wird in Abschnitt 4.2.4 im Kontext des Bestärkenden Lernens von Aktionssequenzen vorgestellt. Die Vorteile dieses Ansatzes werden dort am Beispiel einer Manipulationsaufgabe gezeigt, für die in verschiedenen Situationen verschiedene Aktionssequenzen eingesetzt werden müssen.

In diesem Kapitel wird diese Strategie auf das Bestärkende Lernen hierarchischer Aufgaben mit der MAXQ-Methode übertragen, um ihre Vorteile zur Entwicklung einer effizienten hierarchischen Lernstrategie zu nutzen. Die Grundlage hierfür bilden die im vorangegangenen Abschnitt beschriebene Approximation der Komponenten der MAXQ-Zerlegung durch Gauß-Prozesse sowie die hierarchische Berechnung probabilistischer Schätzungen der Q-Werte. Der Kern der Explorationsstrategie besteht in der Optimierung des Wertes der Erwarteten Veränderung über mögliche Aktionen. Diese vereint die sogenannte Erwartete Verbesserung, die für eine ausgewählte Aktion vorhergesagt wird, mit der Erwarteten Verschlechterung. Während die Erwartete Verbesserung die Suche nach gewinnversprechenden Aktionen ermöglicht, dient die Erwartete Verschlechterung zur Vermeidung unvorhersagbarer und damit potenziell für den Roboter und seine Umgebung risikoreicher Aktionen. Die Berechnung dieser beiden Werte wird detailliert in Abschnitt 3.2 beschrieben. Diese basiert auf probabilistischen Schätzungen der Q-Werte von Paaren aus Zuständen und Aktionen und den mit den Schätzungen verknüpften Unsicherheiten. In der im vorherigen Abschnitt vorgestellten kontinuierlichen Erweiterung des MAXQ-Lernverfahrens werden diese Schätzungen mittels Gleichung (5.1) rekursiv aus den durch Gauß-Prozess-Regression gewonnenen Schätzungen der Komponenten der MAXQ-Hierarchie bestimmt. Der Wert der Erwarteten Veränderung, dessen detaillierte Berechnung in Abschnitt 4.2.4 beschrieben wird, vereinigt diese beiden gegensätzlichen Ziele in einem Optimierungskriterium, mit dem gleichzeitig vielversprechende wie auch sichere Aktionen gesucht werden können.

Durch die rekursive Berechnung der geschätzten Q-Werte werden bei der Aktionsauswahl auch Informationen über den erreichbaren Nutzen von Zustands- und Aktionspaaren und mögliche Risiken durch die Ausführung von Teilaufgaben auf tieferen Ebenen der MAXQ-Hierarchie berücksichtigt. Dies führt dazu, dass die Wahrscheinlichkeit für schlechte Lösungen auf niedrigeren Hierarchieebenen oder Anfragen nach Demonstrationen auf diesen Ebenen sinkt.

5.2.3 Hierarchische Integration von Expertenwissen

Eine informierte Exploration, wie sie im vorangegangenen Abschnitt vorgestellt wurde, setzt die Verfügbarkeit von Erfahrungen voraus, anhand derer Entscheidungen für vielversprechende, zu evaluierende Aktionen getroffen werden können. Im Fall der vorgestellten Bayesschen Explorationsstrategie bedeutet dies, dass eine genügende Anzahl an Trainingsbeispielen vorhanden sein muss, um eine zuverlässige Gauß-Prozess-Approximation der Nutzen- beziehungsweise Abschlussnutzenfunktionen der einzelnen Teilaufgaben zu ermöglichen. Diese sind Voraussetzung, um vielversprechende Aktionen während des Bestärkenden Lernens auszuwählen und somit das Lernverfahren zu beschleunigen. In vielen Fällen verfügen Menschen über die hierfür benötigten Erfahrungen, so dass Bestärkende Lernverfahren häufig mit Ansätzen des Lernens aus Demonstrationen kombiniert werden, um dieses Wissen zu nutzen und einem lernenden Robotersystem zugänglich zu machen. Dieser interaktive Ansatz des maschinellen Lernens bringt den Vorteil mit sich, dass er eine bequeme und dem Menschen vertraute Art und Weise ist, einem Gegenüber Wissen zu vermitteln. Er ist intuitiv, so dass es auch für Menschen ohne spezielle Ausbildung vereinfacht wird, die mit dem Roboter zusammenarbeiten sollen, die Rolle eines Experten zu übernehmen und dem Roboter mögliche Lösungen der geforderten Aufgaben zu demonstrieren.

Während bereits in Kapitel 4 die Integration von Expertenwissen durch Lernen aus Demonstrationen in ein flaches Bayessches Lernverfahren zum Lernen von Aktionssequenzen beschrieben wurde, wird in diesem Abschnitt die Anwendung dieser Idee auf das kontinuierliche MAXQ-Verfahren vorgestellt. Dabei wird die Eigenschaft des hier präsentierten Verfahrens genutzt, dass in jeder Teilaufgabe ein eigenes Gauß-Prozess-Modell zur Approximation der Komponenten der MAXQ-Zerlegung verwendet wird, wodurch diese mit Hilfe einer separaten Menge von Trainingsbeispielen gelernt werden können. Dies erlaubt es dem System, Expertenwissen in Form von Demonstrationen gezielt in ausgewählten Teilaufgaben auf einzelnen Hierarchieebenen einzusetzen.

Dabei wird ein interaktiver Ansatz wie auch in Kapitel 4 verfolgt, der es dem hier vorgestellten Lernverfahren ermöglicht, für beliebige Teile einer hierarchischen Aufgabe Unterstützung durch einen Experten anzufragen. Im Gegensatz zu dem flachen Lernansatz für Aktionssequenzen braucht dieser daraufhin nicht die gesamte Teilaufgabe zu demonstrieren, sondern kann für seine Demonstration die durch die Aufgabe vorgegebenen Aktionen der Abstraktionsebene unterhalb der zu demonstrierenden Teilaufgabe nutzen. Der Experte profitiert somit in zweierlei Hinsicht von der hierarchischen Struktur der Aufgabe. Zum einen müssen vorangegangene oder nachfolgende Teilaufgaben, zu deren Lösung das System keine Demonstration benötigt, nicht demonstriert werden. Zum anderen entfällt auch die Demonstration der Teilaufgaben auf niedrigeren Abstrakti-

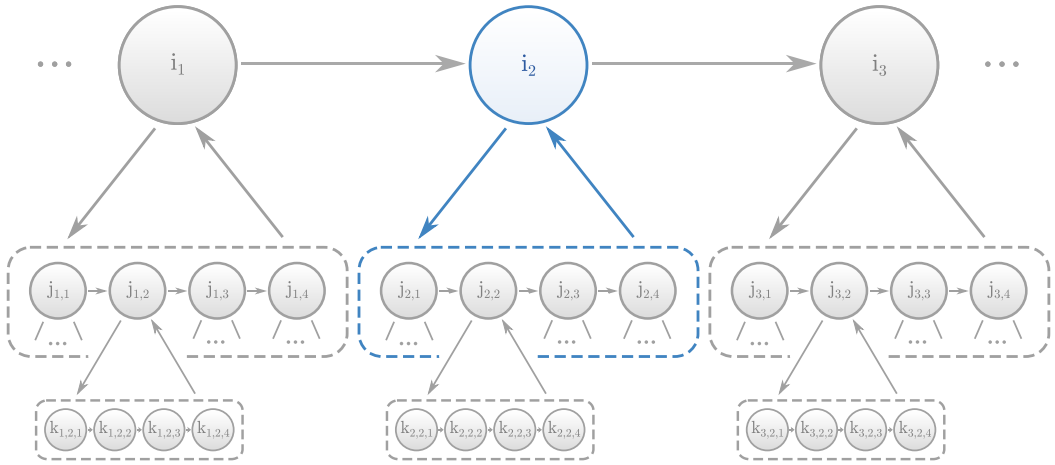


Abbildung 5.4: *Beispielhafte Darstellung der Aktionen, die in dem hier vorgestellten hierarchischen Lernverfahren zur Demonstration einer Teilaufgabe vorgeführt werden müssen. Durch die Entscheidung für Bestärkendes Lernen oder Lernen aus Demonstrationen in jeder Teilaufgabe und in jeder Situation, werden Demonstrationen nur gezielt in einzelnen Teilaufgaben angefordert. In dem dargestellten Beispiel ist die Teilaufgabe, für die vom System eine Demonstration angefordert wird, blau gekennzeichnet. Die Demonstration erfolgt durch eine Sequenz von Aktionen, die Teilaufgaben der darunterliegenden Ebene entsprechen. Auf dieser Ebene wird abermals eine Entscheidung getroffen, ob eine Demonstration benötigt wird. Diese Entscheidung wird unabhängig von der Anfrage nach einer Demonstration auf der darüberliegenden Ebene getroffen. Entsprechend sind die Teilaufgaben dieser Ebene nicht blau hervorgehoben. Selbiges gilt für die vorangehenden und nachfolgenden Teilaufgaben der oberen Ebene.*

onsebenen einer Teilaufgabe und die Demonstration beschränkt sich auf die angefragte Hierarchieebene. Dadurch werden ebenso unnötige Demonstrationen bereits gelernter Aufgabenteile vermieden wie auch unnötig komplexe Demonstrationen, die Lösungen für Aufgaben über mehrere Ebenen beinhalten. Die somit benötigten Demonstrationen und die Aufgabenteile, für die gegebenenfalls keine Demonstrationen erforderlich sind, sind in Abbildung 5.4 dargestellt. Da bei jeder Anfrage nur ein kleiner Teil der Aufgabe gezielt durch den Experten gelöst wird, wird der für Demonstrationen benötigte Arbeitsaufwand erheblich reduziert, insbesondere bei Demonstrationsanfragen auf höheren Ebenen.

Um aus der Demonstration des Experten für eine bestimmte Teilaufgabe ein Trainingsbeispiel für das Gauß-Prozess-Modell zu generieren, wird die vorgeführte Aktionssequenz von dem System ausgeführt und anhand der dadurch erzielten Effekte auf die Umgebung bewertet. Sobald die vorgeführte Teilaufgabe abgeschlossen ist, werden im Fall einer zusammengesetzten Aufgabe die Abschlussnutzenfunktionen beziehungsweise im Fall von atomaren Aktionen die Nutzenfunktionen beginnend von der letzten Aktion aktualisiert und wie im Bestärkenden Lernen der Gleichung (5.3) folgend bestimmt. Somit startet die Aktualisierung der Abschlussnutzenwerte beziehungsweise der Nutzenwerte mit dem Zielzustand und endet in dem Zustand, in dem die Demonstration angefordert wurde. Dieses ist in dem Ansatz zum Lernen aus Demonstrationen im Gegensatz zum Bestärken-

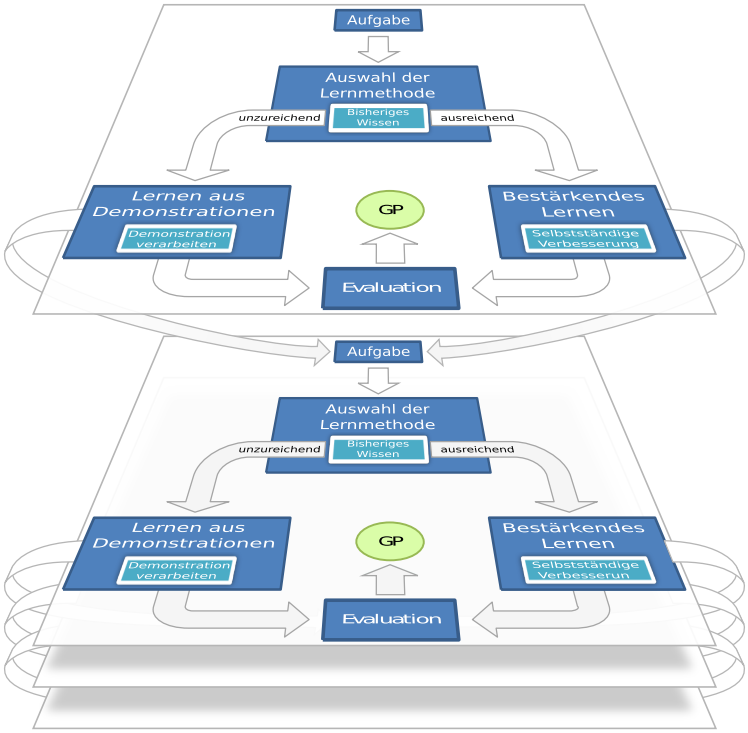


Abbildung 5.5: *Schematische Übersicht des hierarchischen Systems. Dabei wird auf jeder Ebene dieselbe Kombination aus Lernen aus Demonstrationen und Bestärkendem Lernen eingesetzt. In diesem System wird in jeder Teilaufgabe anhand des zur Verfügung stehenden Wissens eine Entscheidung für eines dieser beiden Verfahren getroffen. Unabhängig von der Entscheidung wird die gewählte Lösung evaluiert, indem die ausgewählte Aktion durch das Verfahren auf der darunterliegenden Ebene ausgeführt wird. Dabei werden auf der unteren Ebene die dort benötigten situationsbeschreibenden Parameter anhand der ausgewählten Aktionen der darüberliegenden Ebene bestimmt und erneut eine Entscheidung für eines der beiden Lernverfahren getroffen. Der hierarchischen Zerlegung folgend wird dieser Prozess bis zur Ebene der primitiven Aktionen wiederholt, wo diese ausgeführt werden. Nachdem eine Teilaufgabe abgeschlossen ist und eine Evaluation für diese vorliegt, wird daraus ein neues Beispiel für die Gauß-Prozess-Approximation der MAXQ-Zerlegung der entsprechenden Teilaufgabe generiert.*

den Lernen möglich, da mittels einer Demonstration die gesamte Teilaufgabe auf einmal gelöst wird und nicht, wie im Bestärkenden Lernen, schrittweise. Der Ansatz profitiert von dieser Art der Aktualisierung der Nutzen- und Abschlussnutzenfunktionen, da diese Berechnung auf dem Wert der Nachfolgesituation aufbaut, der zuvor schon mit den neuen Werten aktualisiert wurde und somit dafür sorgt, dass bessere Schätzungen entstehen.

Aufgrund der Generalisierungsfähigkeit der Gauß-Prozess-Approximation, die in Abschnitt 3.1 erläutert wird, können die aus den Demonstrationen berechneten Informationen auf andere Situationen und Aktionen der bearbeiteten Teilaufgabe übertragen werden, wodurch in diesen keine weitere Hilfe durch einen Experten notwendig ist. Durch die kontextfreien Teilaufgaben in der MAXQ-Hierarchie können die für eine Teilaufgabe

gesammelten Informationen zum Lernen verschiedener Teilaufgaben der nächst höheren Ebene verwendet werden. Somit wird zusätzlich die Anzahl der Demonstrationen verringert, die zum Erlernen einer Teilaufgabe gegeben werden müssen.

5.2.4 Wahl der Lernmethode in hierarchisch strukturierten Aufgaben

In den vorangegangenen zwei Abschnitten ist deutlich geworden, dass sowohl das Lernen aus eigener Erfahrung und autonomer Interaktion mit der Umgebung als auch das Lernen aus Demonstrationen effektive Ansätze zum Lernen einer Strategie für eine hierarchische Aufgabe darstellen. Dabei sind die Vorteile beider Verfahren bezogen auf das Lernen der Gesamtaufgabe oftmals komplementär zueinander, wodurch sich ihre Stärken besonders gut kombinieren lassen, wie schon in den Ansätzen zum Lernen von Bewegungsprimitiven in Abschnitt 3.3 und von Aktionssequenzen in Abschnitt 4.2.2 gezeigt werden konnte. Auf der einen Seite stellen Demonstrationen einen effektiven und für den Menschen intuitiven Weg dar, um vorhandenes Expertenwissen und die für eine Aufgabe wesentlichen Aspekte auf ein lernendes System zu übertragen. Somit ist keine explizite Modellierung dieses Wissens erforderlich, was in vielen Fällen schwierig oder gar unmöglich wäre, da die zu bestimmenden Parameter dazu nicht explizit bekannt sind. Das Bestärkende Lernen auf der anderen Seite ermöglicht es dem System, in einer hierarchischen Aufgabe Lösungen für Teilaufgaben in vielen Situationen autonom zu finden oder aus Demonstrationen gelernte Lösungen für neue Situationen oder wechselnde Bedingungen anzupassen oder sie zu optimieren. Bestärkendes Lernen stellt eine Ergänzung zum Lernen aus Demonstrationen dar, da die Anzahl der Demonstrationen durch autonomes Lernen gering gehalten werden kann und somit der durch die Demonstrationen verursachte Aufwand für den Experten reduziert werden kann. Wie im vorangegangenen Abschnitt erläutert, stellt aber auch das Lernen aus Demonstrationen eine wichtige Erweiterung für das Bestärkende Lernen dar, da Letzteres sehr ineffektiv sein kann, wenn kein Vorwissen vorhanden ist, durch das der Suchraum eingeschränkt wird oder das zumindest Informationen über interessante Gebiete im Suchraum liefert.

Aus diesem Grund werden beide Arten des Lernens als gleichwertige, komplementäre Module in einem kohärenten Lernverfahren integriert. Dabei wird der bereits für das Lernen von Aktionssequenzen eingesetzte Ansatz aus Abschnitt 4.2.2 in einer flexiblen Kombination benutzt. Um diesen Ansatz auf das Lernen hierarchisch strukturierter Aufgaben auf verschiedenen Abstraktionsebenen zu übertragen, wird auf allen Ebenen der Hierarchie das gleiche Entscheidungsmodul verwendet, wie in Abbildung 5.5 dargestellt wird. Dieses trifft in jeder Situation, in der eine Aktion ausgewählt werden soll, autonom und unabhängig für jede Teilaufgabe eine Entscheidung für das Lernverfahren, das am besten geeignet ist, um eine passende Aktion zur Lösung der Teilaufgabe zu finden. Bei einer hierarchischen Aufgabe wird somit die Entscheidung für eines der beiden Verfahren nicht für die Gesamtaufgabe getroffen, sondern in jeder Teilaufgabe auf den verschiedenen Ebenen in der Hierarchie unabhängig von allen anderen. Dies ermöglicht es dem System, gezielt Demonstrationen für bestimmte Teilaufgaben auf verschiedenen Abstraktionsebenen anzufragen. Somit kann sich das System auf tieferen Abstraktionsebenen abhängig von dem gegebenen Wissensstand ebenfalls neu entscheiden, ob für diesen Aufgabenteil eine autonome Lösung gefunden werden kann oder ob auch hier eine Demonstration

benötigt wird. So können zum Beispiel aufgrund von Abstraktionen des Zustandsraumes unterschiedliche Zustandsräume auf verschiedenen Ebenen vorliegen. Dies kann dazu führen, dass auf einigen Ebenen schon Informationen in dem dort genutzten Zustandsraum für die gesuchte Situation vorliegen, so dass auf dieser Ebene autonom eine Lösung generiert werden kann, während auf anderen Ebenen diese Informationen aufgrund eines anderen Zustandsraumes fehlen und eine Demonstration benötigt wird. Entsprechend muss nicht die komplette Aufgabe mittels Lernen aus Demonstrationen oder Bestärkendem Lernen gelöst werden. Auf diese Weise wird die Beteiligung des Experten auf die Bereiche fokussiert, in denen selbstständiges Lernen nicht ohne Risiko durchgeführt werden kann und die Hilfe eines Experten somit unbedingt erforderlich ist. Gleichzeitig wird aber auch der Gesamtaufwand für den Experten niedrig gehalten, indem das System, wann immer dies möglich ist, durch die autonome Interaktion mit der Umgebung lernt.

Diese Art des Aufbaus des Lernverfahrens und der Kombination der verschiedenen Lernstrategien erlaubt eine modulare Architektur des hier vorgestellten Systems. Jede Komponente des Verfahrens wird durch ein in sich abgeschlossenes Modul dargestellt, welche zu einem Lernverfahren für die Teilaufgaben einer Abstraktionsebene zusammengesetzt werden können. Auf jeder dieser Ebenen verfügt das Lernverfahren über ein übergeordnetes Entscheidungsmodul, welches in jeder Situation entweder das Modul für das Bestärkende Lernen oder dasjenige für das Lernen aus Demonstrationen wählt. Das System wurde so entwickelt, dass auf den unterschiedlichen Abstraktionsebenen der MAXQ-Hierarchie dasselbe kombinierte Lernverfahren eingesetzt werden kann. Dazu werden Instanzen des kombinierten Lernverfahrens entsprechend der Struktur der zu lösenden Aufgabe miteinander verknüpft. Aufgabenübergreifende Funktionalitäten, wie etwa die Gauß-Prozess-Regression, werden den einzelnen Teilaufgaben durch aufgabenunabhängige Module zur Verfügung gestellt, die auf verschiedenen Abstraktionsebenen ebenfalls wiederverwendet werden können.

Durch diesen zweistufigen modularen Aufbau wird ein System erzeugt, das durch die konsistent gewählte Struktur innerhalb der verschiedenen Abstraktionsebenen leicht skalierbar und um neue Abstraktionsebenen für unterschiedlich komplexe Aufgaben erweiterbar ist. Zur Anpassung an eine konkrete Aufgabe müssen lediglich eine entsprechende MAXQ-Hierarchie definiert und die Variablen, in denen sich die verschiedenen Abstraktionsebenen unterscheiden können, angepasst werden. Dazu gehören die Verknüpfungen zwischen den Ebenen, die zustands- sowie aktionsbeschreibenden Parameter und die auf einer Ebene zu vergebenden Belohnungen. Darüber hinaus können durch die Modularität auf einfachste Weise einzelne Komponenten gegen gleichwertige ausgetauscht werden, falls dieses erwünscht ist.

5.3 Experimente und Evaluation

Um den hier vorgestellten Ansatz zu evaluieren, wird die Aufgabe gewählt, einen Stapel von Boxen sicher auf einem Tisch zu versetzen. Diese ist typischen Manipulationsaufgaben aus dem alltäglichen Leben nachempfunden, bei denen abhängig von der gegebenen Situation verschiedene Manipulationsstrategien angewandt werden müssen. Erschwert wird dieses Experiment durch ein Hindernis, das oftmals den direkten Weg zu einer Zielposition versperrt, und durch die Instabilität des Stapels, so dass nicht beliebig viele

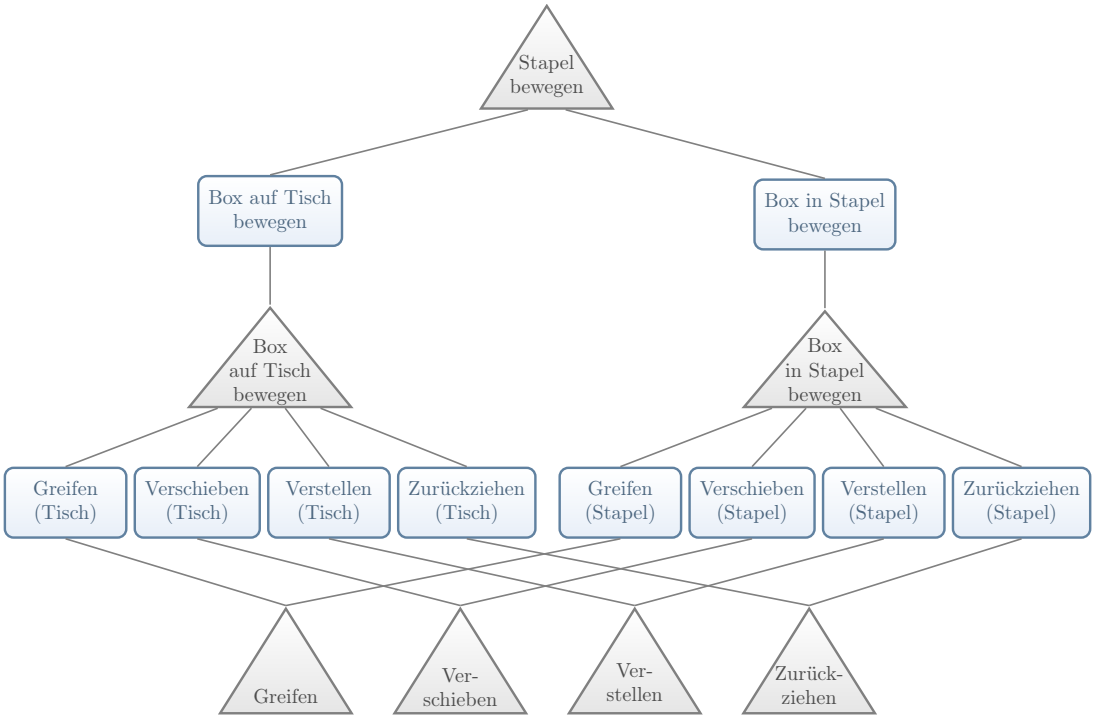


Abbildung 5.6: *Darstellung der Beispielaufgabe in den Experimenten als MAXQ-Graph. MAX-Knoten repräsentieren die Teilaufgaben beziehungsweise atomare Aktionen und sind als Dreiecke dargestellt. Entsprechend wird in jedem MAX-Knoten die zu erwartende Belohnung für das Lösen der jeweiligen Teilaufgabe berechnet, unabhängig davon, innerhalb welcher übergeordneten Aufgabe diese ausgeführt wird. Somit können MAX-Knoten von mehreren übergeordneten Aufgaben verwendet werden. Q-Knoten verbinden eine übergeordnete Aufgabe mit den Aktionen, die für diese verfügbar sind, und werden durch Rechtecke repräsentiert. Sie berechnen den Nutzen für die Ausführung der jeweiligen Aktion und das anschließende Beenden der übergeordneten Aufgabe.*

Objekte auf einmal bewegt werden können, ohne dass der Stapel zusammenbricht. In jeder Episode werden zufällig die kartesischen Start- und Zielpositionen des Stapels ausgewählt. Außerdem werden die Anzahl an Boxen im Stapel sowie die einschließlich aller auf ihr stehenden Boxen zu bewegende Box ebenfalls zufällig bestimmt. In jeder Iteration einer Episode wählt das System, der gelernten hierarchischen Strategie folgend, eine passende Aktion für die gegebene Situation aus. Ausgeführt werden die Experimente in der physikbasierten Simulation Gazebo [34] und auf dem Roboter Baxter. Genauere Informationen zu dem Versuchsaufbau und der zu lösenden Aufgabe sind in Abschnitt 4.3.1 nachzulesen, wo diese Aufgabenstellung im Kontext von Aktionssequenzen untersucht wird. In den Experimenten wird gezeigt, dass das in diesem Kapitel vorgestellte System sicher und erfolgreich verschiedene Strategien für unterschiedliche Situationen lernt und von dem hierarchischen Lernen mit der Kombination aus Bestärkendem Lernen und Lernen aus Demonstrationen profitiert.

5.3.1 Aufgabendefinition

Um die Aufgabe mit Hilfe des vorgeschlagenen Algorithmus zu lösen, wird diese in eine Hierarchie von Teilaufgaben auf drei unterschiedlichen Abstraktionsebenen gegliedert, über die anschließend eine MAXQ-Hierarchie definiert wird. Die unterschiedlichen Abstraktionsebenen und ihre Teilaufgaben sind in Abbildung 5.6 in einem MAXQ-Graphen [68] dargestellt. Die gewählte Struktur erlaubt die Verwendung von reduzierten Zustands- und Aktionsräumen sowie Belohnungsmodellen, die passend auf die jeweilige Abstraktionsebene zugeschnitten sind und die durch die verringerte Komplexität ein effizientes Lernen ermöglichen. Details zu den gewählten Aktions- und Zustandsräumen, den Belohnungen sowie der Modularität des Ansatzes und der Wahl der Parameter des Lernverfahrens werden in den folgenden Abschnitten beschrieben, bevor auf die Ergebnisse der Experimente eingegangen wird.

Aktionsräume

Auf der obersten Ebene der Abstraktionshierarchie gibt es nur eine einzige Aktion, die die komplette Aufgabe löst, einen Stapel von Boxen zu einer Zielposition zu bewegen. Um dieses Ziel zu erreichen, muss das System lernen, sich für eine Manipulationsstrategie, bestehend aus Aktionen der mittleren Ebene, zu entscheiden. Auf der mittleren Ebene gibt es zwei verschiedene Teilaufgaben, aus denen die oberste Ebene wählen kann. Bei dieser Entscheidung muss das System die zu bewegende Box im Stapel berücksichtigen. Entweder muss eine Box bewegt werden, die sich direkt auf dem Tisch befindet, oder eine Box, die eine höhere Position im Stapel einnimmt. Jede Iteration auf der mittleren Abstraktionsebene umfasst das Bewegen einer einzelnen Box, einschließlich aller darauf gestapelten Boxen, zu einer bestimmten Zielposition. In manchen Fällen kann die komplette Aufgabe auf der obersten Ebene mit einen einzigen Schritt aus der mittleren Ebene gelöst werden indem der gesamte gewünschte Stapel auf einmal versetzt wird. In anderen, komplizierteren Situationen wird eine Kombination mehrerer Aktionen der mittleren Ebene, mit denen verschiedene Teile des Stapels bewegt werden, benötigt, um die gesamte Aufgabe erfolgreich auszuführen.

Zusätzlich zu der Entscheidung für die richtige Aktion, abhängig von der gegebenen Situation, muss das System lernen, jeder Aktion die geeigneten Start- und Zielpositionen zuzuordnen. Diese Positionen werden aus einer vordefinierten Menge von Referenzpunkten bestimmt. Sie repräsentieren aufgabenspezifische Positionen unabhängig von ihren kartesischen Koordinaten in einer bestimmten Instanz der Aufgabe und werden durch den Designer der MAXQ-Hierarchie vorgegeben. Für die Experimente werden auf der obersten Ebene die Startposition des Stapels, die gewünschte Zielposition und eine Position, an der ein Stapel oder Teilstapel vorübergehend zwischengelagert werden kann, als relevant identifiziert. Für jeden von diesen Punkten gibt es Referenzpunkte in verschiedenen Höhen, die der Position einer Box innerhalb eines Stapels entsprechen.

Die beiden Teilaufgaben auf der mittleren Ebene teilen sich dieselben vier Bewegungsprimitive der untersten Ebene, die das System kombinieren kann, um diese Teilaufgabe zu lösen. Bei diesen handelt es sich um das Greifen eines Objektes, das Verstellen eines Objektes an eine andere Position, das Verschieben des Objektes zu einer Zielposition ohne es dabei anzuheben und das Zurückziehen der Hand zu einer Ruheposition. Für

Tabelle 5.1: *Aktionsrepräsentation in den Experimenten*

Ebene	**Aktionsvariablen**	
Oben	Aktion	$\in$ {Bewege von Stapel, Bewege von Tisch}
	Start	$\in$ {Start, Ziel, Ablage} $\times$ Anzahl Boxen
	Ziel	$\in$ {Start, Ziel, Ablage} $\times$ Anzahl Boxen
Mitte	Aktion	$\in$ {Greifen, Verschieben, Verstellen, Zurückziehen}
	Start	$\in$ {Start, Ziel, Ruheposition}
	Ziel	$\in$ {Start, Ziel, Ruheposition}
Unten	–	

jede Situation muss das System die korrekte Aktion aus diesen vier Möglichkeiten und die Identifikationsnummern für zwei Referenzpunkte, aus denen sich die Start- und Zielposition der gewünschten Bewegung berechnen lassen, wählen. Auf der mittleren Ebene beinhalten die verfügbaren Referenzpunkte die Start- und Zielpositionen, die durch die Aktionsparameter der obersten Ebene bestimmt werden, und eine vordefinierte Ruheposition, die der Manipulator zwischen einzelnen Aktionen oder Aktionssequenzen einnehmen kann.

Die Aktionen auf der untersten Ebene der MAXQ-Hierarchie entsprechen den Bewegungsprimitiven, die als atomare Aktionen in den Experimenten angenommen werden. Für die Zwecke der Experimente wurden die Parameter der Bewegungsprimitive mit dem in Kapitel 2 beschriebenen Ansatz im Voraus trainiert. Werden die Identifikationsnummern einer primitiven Bewegung und die der benötigten Referenzpunkte für die Start- und Zielposition angegeben, so kann das System daraus Trajektorien generieren, aus denen die Effekte der Bewegungen auf die Umwelt berechnet oder diese auf dem Roboter ausgeführt werden können.

Tabelle 5.1 fasst die Variablen zur Beschreibung der Aktionsräume auf den verschiedenen Ebenen der Aufgabenhierarchie zusammen. Während der Experimente wird davon ausgegangen, dass die Effekte der Aktionen deterministisch sind, was bedeutet, dass mehrere Ausführungen der gleichen Aktion für den gleichen Zustand zu einem Übergang in immer den gleichen Nachfolgezustand führen.

Zustandsräume

Die Verwendung der MAXQ-Zerlegung der hier verwendeten Aufgabe und ihrer Q-Funktion ermöglicht es dem hier vorgestellten Lernverfahren mit Abstraktionen des Zustandsraumes zu arbeiten und von diesen zu profitieren. Sie bietet den Vorteil, dass so in kleineren Teilräumen auf den verschiedenen Ebenen der Aufgabenhierarchie gearbeitet werden kann. Dies reduziert die Komplexität der einzelnen Teilaufgaben und beschleunigt somit das Lernen. Auf der obersten Ebene umfasst ein Zustand die Identifikationsnummern, die für jede Box angeben, ob diese Box an der zufälligen Startposition des Stapels, an der zufälligen Zielposition oder der Zwischenablageposition positioniert ist oder ob diese nicht Teil der aktuellen Aufgabe ist. Eine weitere Identifikationsnummer bezeichnet die Position der Box innerhalb des Stapels, die bewegt werden soll, da Boxen

Tabelle 5.2: *Zustandsrepräsentation in den Experimenten*

Ebene	Zustandsvariablen	
Oben	Position der Box	$\in$ {Start, Ziel, Zwischenablage, nicht vorhanden} (pro Box)
	Ausgewählte Box	$\in \{0, \ldots,$ Anzahl der Boxen - 1$\}$
Mitte	Position der Box	$\in \mathbb{R}^3$
	Ziel	$\in \mathbb{R}^3$
	Objekt haltend	$\in \{0, 1\}$
	aktuelle Handposition	$\in$ {Start, Ziel} $\times$ Anzahl Boxen $\cup$ {Ruheposition}
Unten	Anfang der Bewegung	$\in \mathbb{R}^3$
	Ende der Bewegung	$\in \mathbb{R}^3$

an verschiedenen Positionen im Stapel unterschiedlich behandelt werden müssen.

Auf der mittleren Ebene wird der Zustandsraum reduziert, indem in diesem nur die Informationen enthalten sind, die für die Teilaufgaben auf dieser Ebene relevant sind. Da sich die mittlere Ebene im Gegensatz zur höchsten Ebene, auf der die komplette Aufgabe gelöst werden soll, nur noch mit dem Versetzen einer bestimmten Box mit allen darauf gestapelten Boxen beschäftigt, müssen hier auch nur die Parameter betrachtet werden, die nötig sind, um eine bestimmte Box zu bewegen. Dies sind die aktuellen Koordinaten der zu bewegenden Box, die Koordinaten der gewünschten Zielposition, ob der Roboter derzeit ein Objekt in der Hand hält und eine Identifikationsnummer des Referenzpunktes, der die aktuelle Position des Endeffektors beschreibt.

Obwohl das Lernen der Parameter der einfachen Bewegungsprimitive der untersten Ebene nicht Bestandteil dieser Arbeit ist, werden die erhaltenen Belohnungen auf dieser Ebene ebenfalls durch ein einzelnes Gauß-Prozess-Modell pro Bewegungsprimitive repräsentiert. Dies ist erforderlich, um für die höheren Ebenen Informationen für eine geeignete Aktionsauswahl zur Verfügung zu stellen, da manche Aktionen aufgrund der für sie benötigten Arbeit effizienter für das System sind als andere. Der Zustandsraum auf dieser Ebene ist definiert durch die Koordinaten einer Start- und Zielposition der ausgeführten Bewegung, da diese direkten Einfluss auf die benötigte Anstrengung haben und somit auch Einfluss auf die Belohnung, die auf dieser Ebene erreicht wird. Die Zustandsvariablen aller Ebenen sind in Tabelle 5.2 zusammengefasst.

Belohnungen

In dem hier vorgeschlagenen Ansatz werden die Belohnungen individuell den Teilaufgaben auf den verschiedenen Ebenen der hierarchischen Aufgabe zugewiesen. Dies ermöglicht es, spezifische Teilaufgaben ohne Beeinflussung ihrer Elternknoten, die nicht zu dem Ergebnis der Teilaufgabe beigetragen haben, zu belohnen oder zu bestrafen. Dies wird auch als hierarchische Form des Zuweisungsproblems von Belohnungen für Aktionen [68] bezeichnet.

Um den Erfolg einer Aktion auf der niedrigsten Abstraktionsebene zu bewerten, wird, wie in Kapitel 4, die Arbeit W für die Ausführung der einzelnen Bewegungsprimitiven

der Läge T für einen gegebenen Start- und Zielpunkt gemessen. Eine genauere Definition ist in Gleichung (4.6) zu finden.

Auf der mittleren Ebene wird ermittelt, ob es eine Kollision zwischen dem zu manipulierenden Stapel und anderen Elementen in der Umgebung, wie zum Beispiel einem Hindernis, gegeben hat. Die Aktionssequenzen, die zu einer Kollision geführt haben, werden mit einer negativen Belohnung bestraft. Wenn alle unbeteiligten Objekte nach Ausführung einer Teilaufgabe noch an ihren ursprünglichen Positionen sind und keine Kollisionen erkannt wurden, wird eine Belohnung von null gegeben.

Auf der obersten Ebene betrachten wir, wie weit die Aufgabe fertiggestellt ist, um eine Belohnung zu vergeben. Wenn die Aufgabe vollständig abgeschlossen ist, ohne die Umgebung auf unbeabsichtigte Art und Weise verändert zu haben, wie zum Beispiel durch Umwerfen oder zusätzliches Versetzen von Boxen unter der zu bewegenden Box, die an ihrer ursprünglichen Position bleiben sollten, wird dem System eine positive Belohnung zugewiesen. Wenn die Umgebung während der Ausführung der Aufgabe verändert wurde, dann werden die ausgewählten Aktionen mit einer negativen Belohnung bestraft. In allen anderen Fällen wird dem System eine neutrale Belohnung von null zugewiesen und eine weitere Iteration gestartet. Dies wird so lange wiederholt, bis die gegebene Aufgabe vollständig gelöst ist. Zusammenfassend sind die Belohnungen für alle drei Ebenen folgendermaßen verteilt:

$$R(a,s) = \begin{cases} & \underline{\text{Oberste Ebene}} \\ -1 & \text{beendet mit Nebeneffekten,} \\ 1 & \text{erfolgreich beendet,} \\ 0 & \text{sonstiges,} \\ & \underline{\text{Mittlere Ebene}} \\ -1 & \text{beendet mit Kollision oder Nebeneffekten,} \\ 0 & \text{sonstiges,} \\ & \underline{\text{Untere Ebene}} \\ -W & \text{jede Aktion} \end{cases}$$

Modulare Architektur

Das hier vorgestellte System verfügt über einen modularen Aufbau. Dieser drückt sich zum einen darin aus, dass die einzelnen Komponenten des Lernverfahrens als in sich abgeschlossene Module aufgebaut sind, die sich zur Lösung von Teilaufgaben auf unterschiedlichen Abstraktionsebenen flexibel kombinieren lassen. Zum anderen lässt sich innerhalb jeder Abstraktionsebene dieselbe Kombination von Modulen für alle Teilaufgaben dieser Ebene einsetzen. Beispiele für solche Komponenten, die sowohl innerhalb einer Abstraktionsebene als auch über verschiedene Abstraktionsebenen hinweg zur Lösung von Teilaufgaben wiederverwendet werden können, sind das Lernen aus Demonstrationen, das Bestärkende Lernen, die Strategie der Aktionsauswahl, die Entscheidung für eines der beiden Verfahren oder die Approximation mittels Gauß-Prozessen. Der Vorteil dieser Architektur besteht darin, dass das System leicht skalierbar ist und zusätzliche Ebenen mit nur geringem Aufwand hinzugefügt werden können.

Für die Implementierung des Systems und der Experimente wird die Middleware ROS [29] verwendet. Der modulare Aufbau des Systems wird dabei erreicht, indem

die einzelnen Module als sogenannte ROS-Knoten implementiert werden, die jeweils in einem eigenen, isolierten Prozess laufen und durch den Austausch von Nachrichten über definierte Schnittstellen miteinander kommunizieren. Der Ablauf des Lernverfahrens wird mit Hilfe eines endlichen Automaten beschrieben, der die verschiedenen Module zum Lernen einer Teilaufgabe koordiniert.

Die Lernverfahren auf unterschiedlichen Ebenen sind ebenfalls lose durch nachrichtenbasierte Kommunikation miteinander gekoppelt. Durch diesen flexiblen Aufbau lassen sich einfach verschiedene Komponenten durch gleichartige Verfahren ersetzen, solange die Schnittstelle unverändert bleibt. Ebenso lassen sich einfach neue Abstraktionsebenen erstellen und untereinander verknüpfen, um das in diesem Fall über drei Ebenen verlaufende Lernverfahren mit den in den vorherigen Abschnitten beschriebenen Situations-, Aktions- und Belohnungsparametern aufzubauen.

Wahl der Parameter

Anstatt die Q-Funktion jeder Teilaufgabe mit einem einzigen Gauß-Prozess mit einer festen Kernelbreite zu approximieren, wird, wie in Kapitel 4 beschrieben, jede der möglichen Belohnungsarten (positive, neutrale und negative Belohnung) mit einem separaten Gauß-Prozess modelliert und deren Vorhersagen in einer Mischverteilung kombiniert. Dies ermöglicht es, den verschiedenen Komponenten unterschiedlichen Einfluss zuzuordnen. Während der in diesem Abschnitt beschriebenen Experimente wird eine Kernelbreite von $0,2$ auf der obersten Ebene und $0,35$ auf der mittleren und unteren Ebene jeweils für die erfolgreichen Fälle und $0,055$ auf allen Ebenen für die nicht erfolgreichen Fälle gewählt. Somit haben erfolglose Fälle einen lokaleren Einfluss als die erfolgreichen. Zudem wird, wie in Abschnitt 4.2.1 vorgestellt, auch bei diesen Experimenten ein kombinierter Kernel eingesetzt, der die diskrete Metrik für kategorische Parameter mit der Euklidischen Metrik für reellwertige Parameter kombiniert. Der Schwellwert θ, der bei der Entscheidung zwischen dem Lernen aus Demonstration und dem Bestärkenden Lernen genutzt wird, wird auf $\theta = 0,3$ auf der obersten Ebene und $0,25$ auf sowohl der mittleren als auch der unteren Ebene gesetzt. Dieser Wert wurde empirisch bestimmt und stellt einen guten Kompromiss zwischen der Sicherheit des Roboters und seiner Umgebung und der Anzahl der angeforderten menschlichen Demonstrationen dar. Dass auf der unteren und mittleren Ebene gleiche Werte, auf der oberen allerdings andere Werte gewählt werden, hängt damit zusammen, dass die mittlere und untere Ebene ähnliche Zustands- und Aktionsräume aufweisen, wohingegen die Räume der oberen Ebene verschieden dazu sind.

5.3.2 Hierarchisches Lernen mit dem integrierten Ansatz

Um zu zeigen, wie der vorgeschlagene Ansatz sowohl von der hierarchischen Struktur der Aufgabe als auch von der Kombination aus Bestärkendem Lernen und Lernen aus Demonstrationen profitiert, wird ein Experiment mit 300 Episoden der Aufgabe mit zufällig initialisierten Parametern ausgeführt. Zusätzlich wird das Lernverfahren, nachdem die hierarchische Aufgabe erfolgreich in einer simulierten Umgebung gelernt wurde, auf den humanoiden Roboter Baxter angewendet. Auf der untersten Ebene werden dabei die Bewegungen genutzt, die durch den in Abschnitt 2.2.4 vorgestellten Ansatz generiert

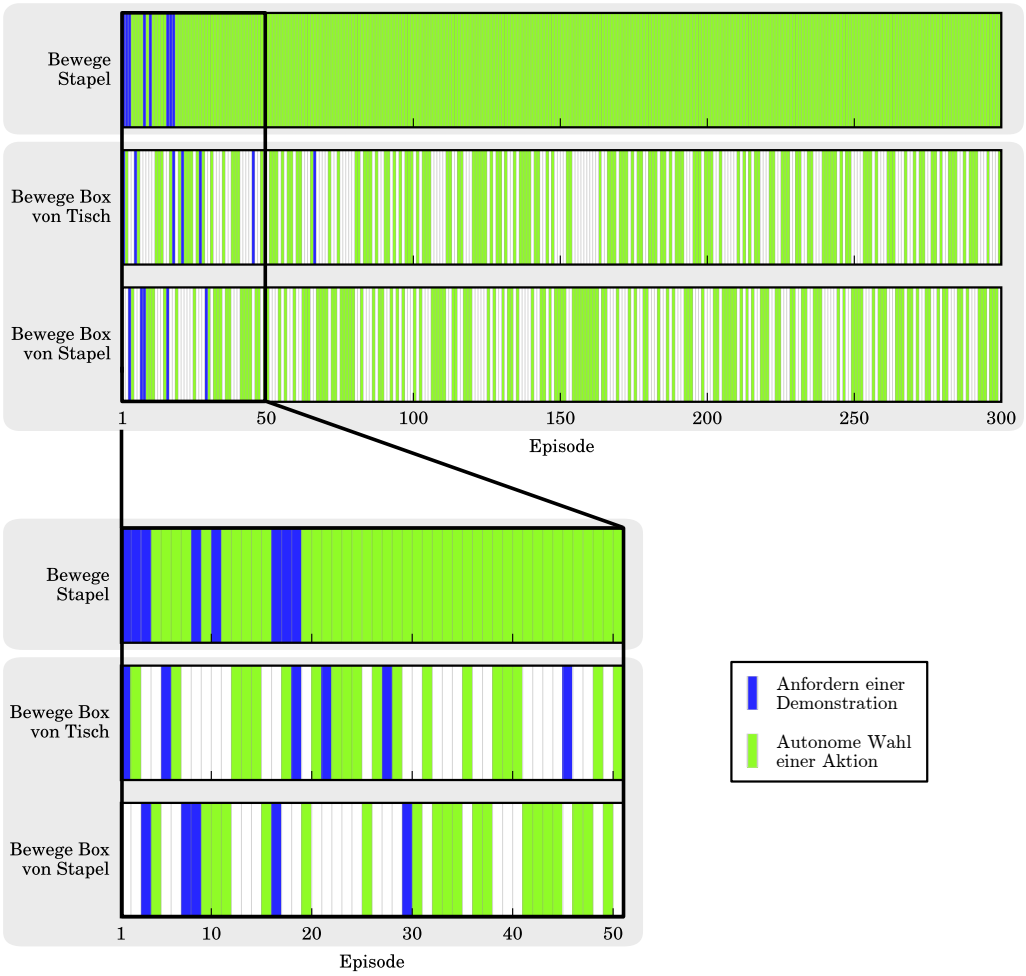


Abbildung 5.7: *Entscheidungen des Systems während des Experimentes mit 300 Episoden der kompletten Aufgabe mit bis zu drei Boxen. Jeder Balken zeigt die Entscheidungen, die für eine bestimmte Teilaufgabe getroffen wurden, und jede Episode wird durch ein Segment repräsentiert. Im oberen Teil der Grafik werden die Entscheidungen aller 300 Episoden dargestellt, während der untere Teil der Grafik die Entscheidungen der ersten 50 Episoden vergrößert darstellt. Beide Teile der Grafik sind in drei Balken unterteilt, entsprechend der drei nicht-primitiven Komponenten der betrachteten Aufgabe. Die jeweils oberen Balken stellen die Entscheidungen dar, die auf der obersten Ebene der Aufgabe getroffen werden. Die beiden unteren Balken, die gemeinsam hinterlegt sind, zeigen die alternativen Teilaufgaben der mittleren Ebene. Der mittlere Balken stellt die Aufgabe dar, eine Box, die sich direkt auf der Tischoberfläche befindet, zu einer Zielposition zu bewegen. Bei der Teilaufgabe des unteren Balkens befindet sich die zu bewegende Box an einer höheren Position im Stapel. In beiden Fällen müssen alle darüber gestapelten Boxen mitbewegt werden. Da es sich um zwei komplementäre Teilaufgaben handelt, ist folglich jedes Segment nur in einem der beiden unteren Balken markiert. Blaue (dunkle) Segmente bezeichnen Episoden, in denen das System nach einer Demonstration fragte. Episoden, in denen das System eine Aktion autonom auswählte, sind in grün (hell) gekennzeichnet. Bei weißen Segmenten wird diese Teilaufgabe nicht ausgewählt.*

werden. Details zu der Generierung der Bewegungen, zu dem Roboter Baxter und zu der Ausführung der Bewegungen auf diesem sind in Abschnitt 2.3.5 beschrieben.

Abbildung 5.7 stellt die zeitliche Abfolge der Entscheidungen der Lernmethode für eines der beiden Lernverfahren dar, die von dem System über den Verlauf der einzelnen Episoden mit zufälligen Aufgabestellungen getroffen wurden. Es wird deutlich, dass das System erfolgreich aus nur 18 Demonstrationen über alle Ebenen die komplette Aufgabe lernt. Auf der obersten Ebene wird die Hilfe eines Experten in insgesamt sechs Fällen angefordert, die genau einmal jeder der möglichen Konfigurationen des Stapels entsprechen. Alle Anfragen nach einer Demonstration erfolgen während der ersten 16 Episoden. Drei der sechs Anfragen werden dabei während der ersten drei Episoden des Experimentes gestellt. Die letzte Anfrage führt zu einer Demonstration, die im Gegensatz zu den vorherigen aus drei Aktionen der darunterliegenden Ebene besteht, da in der gegebenen Situation der Stapel zerlegt und wieder zusammengesetzt werden muss. Dies steht in Einklang mit der Erwartung, dass auf der einen Seite das System keine zufälligen Aktionen wählt, wenn keine Vorkenntnisse vorhanden sind, und auf der anderen Seite keine Anfragen nach Demonstrationen mehr verlangt, wenn ausreichend Erfahrungen gesammelt wurden. Im Vergleich zu Ansätzen, die einen Experten benötigen, um den Lernprozess mit einer gewissen Anzahl an gut ausgewählten Trainingsbeispielen zu initialisieren, fordert das vorgeschlagene System Demonstrationen nur dann an, wenn diese aufgrund unbekannter Situationen nötig sind. Eine Initialisierung ist daher in dem hier vorgestellten Ansatz nicht notwendig, obwohl sie angewendet werden kann, um verfügbares Wissen über die Aufgabe zu integrieren. Die Anfragen nach Demonstrationen in späteren Episoden entstehen durch das zufällige Generieren der Aufgabenstellung und der Parameter, so dass teilweise neue Aufgabenstellungen erst spät im Lernprozess das erste Mal entstehen. Je nach Ähnlichkeit zu den bereits gelösten Aufgaben kann dies zu einer Demonstrationsanfrage führen. Zum Beispiel wird in Episode 16 die Aufgabe, den kompletten Stapel mit drei Boxen zu versetzen, zum ersten Mal angefragt und führt daher in dem System zu einer Anfrage nach einer Demonstration, um die zuvor noch nie gesehene Aufgabenstellung zu lösen. Ein ähnlicher Effekt kann auf der mittleren Ebene beobachtet werden, wo fünf von sieben Demonstrationen um eine Box, die direkt auf dem Tisch positioniert ist, zu bewegen, während der ersten 27 Episoden angefragt werden. Die letzte Demonstration wird allerdings erst in Episode 66 angefordert, in der die Box auf einer anderen Box platziert werden soll. Abermals ist wieder keine ähnliche Konfiguration in diesem Teil des Arbeitsbereichs zuvor beobachtet worden. Die Fähigkeit, Demonstrationen auch spät im Verlauf in das Verfahren zu integrieren, ist besonders von Vorteil, wenn sich die Aufgabenstellung ändert oder zusätzliches Wissen zum Lösen der Aufgabe benötigt wird. In dieser Hinsicht sind Ansätze, die Demonstrationen lediglich zum Lernen einer initialen Strategie für das Bestärkende Lernen nutzen, limitiert gegenüber dem hier vorgestellten Ansatz. Auf der mittleren Ebene ist der Aufgabenteil, eine Box, die auf einer anderen Box steht, zu einer Zielposition zu bewegen, einfacher zu lösen als der Aufgabenteil, eine Box, die sich direkt auf dem Tisch befindet, zu versetzen. Dies hat den Grund, dass das System für die letztere Aufgabe das Hindernis auf dem Tisch für die Entscheidung der Lernmethode mit berücksichtigen muss. Folglich sind nur fünf Demonstrationen in den ersten 29 Episoden nötig, um den Aufgabenteil, eine nicht direkt auf dem Tisch positionierte Box zur gewünschten Zielpositionen zu versetzen, zu erlernen.

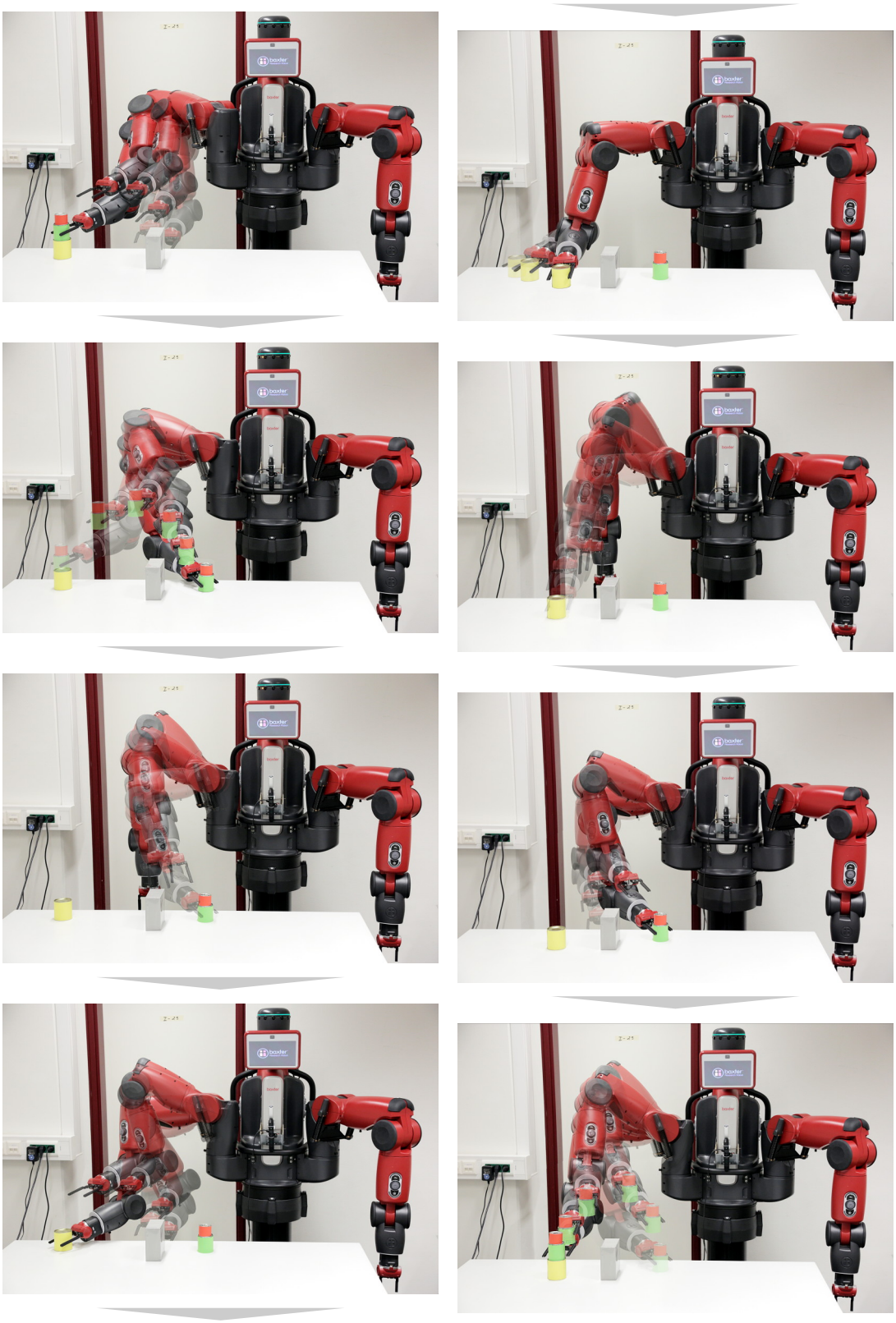


Abbildung 5.8: *Von dem System gelernte Strategie zum Versetzen eines Stapels mit drei Boxen. Der Roboter Baxter unterteilt den Stapel hierfür, bewegt die einzelnen Komponenten getrennt voneinander und setzt diese am Ziel wieder zusammen.*

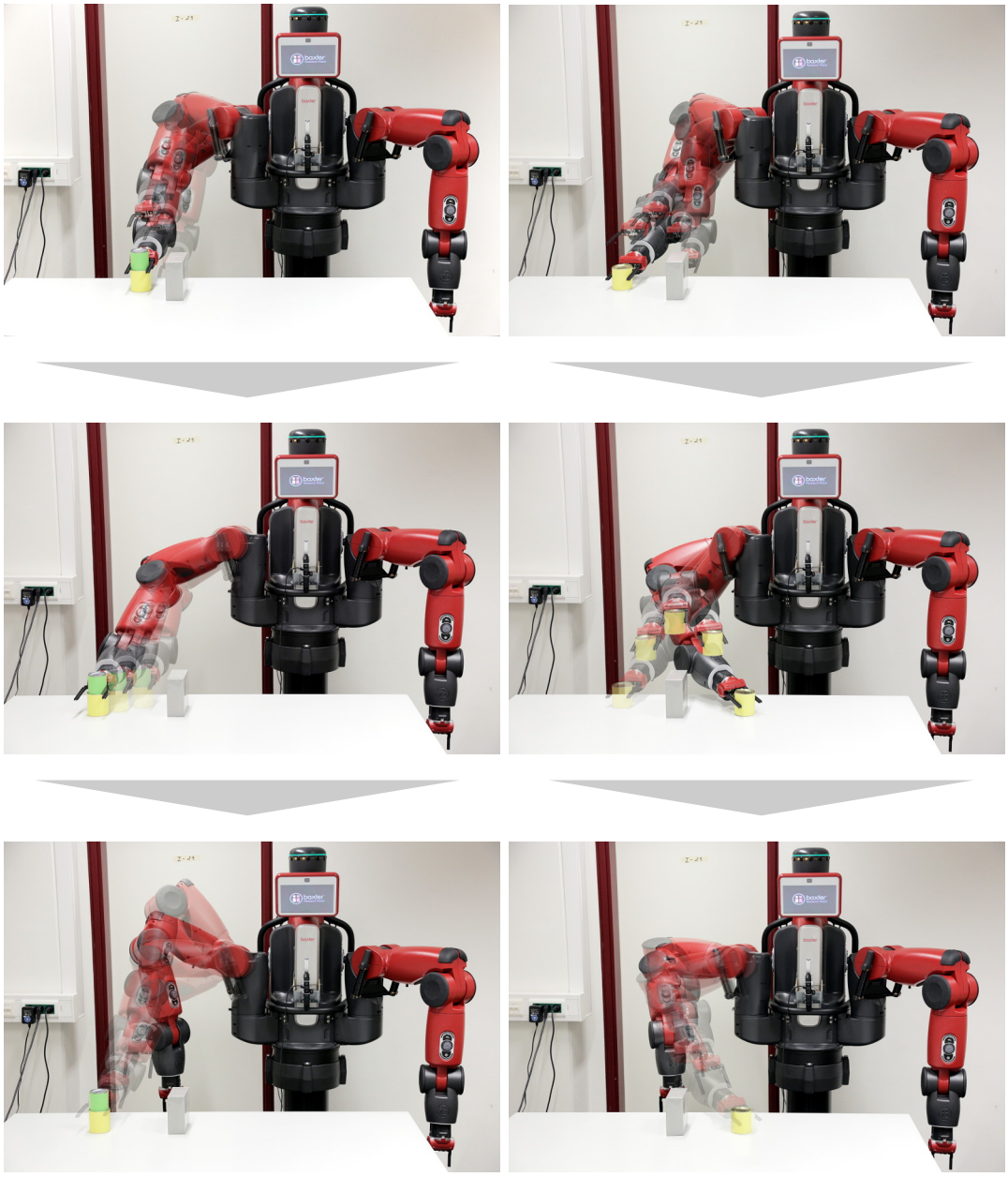


Abbildung 5.9: *Verschiedene Strategien zum Bewegen von Boxen, die von dem System in Abhängigkeit von der gegebenen Situation gewählt werden. Die Abbildung zeigt zwei erlernte Aktionssequenzen für unterschiedliche Situationen, die jeweils aus drei aufeinanderfolgenden Aktionen bestehen, die untereinander dargestellt werden. In Situationen, in denen sich kein Hindernis zwischen den Endpositionen befindet, greift das System das Objekt, verschiebt es und zieht die Hand zur Ruheposition zurück. Dies ist anhand einer Beispielkonfiguration in der linken Spalte dargestellt. Dagegen wird in Fällen, in denen ein Hindernis den direkten Weg zur Zielposition versperrt, eine Verstellbewegung anstatt einer Verschiebebewegung verwendet. Ein Beispiel dazu ist in der rechten Spalte dargestellt. Zudem ist zu sehen, dass das System gelernt hat, dass zwei gestapelte Objekte gemeinsam versetzt werden können, ohne dass der Stapel instabil wird. In jedem Bild werden mehrere Posen überlagert, um den zeitlichen Ablauf darzustellen, wobei die Zielpose nicht-transparent angezeigt wird.*

Werden alle drei Grafiken zusammen betrachtet, wird deutlich, dass die Demonstrationen oft in verschiedenen Episoden auf den unterschiedlichen Ebenen angefordert werden. Dies zeigt, dass ein Informationsmangel, um eine sichere Lösung auf einer bestimmten Ebene zu generieren, nicht notwendigerweise bedeutet, dass Teilaufgaben auf den tieferen Ebenen ebenfalls nicht ohne eine Demonstration gelöst werden können und anders herum, dass, wenn auf tieferen Ebenen Hilfe benötigt wird, diese nicht zwingend für die höheren Ebenen erforderlich ist. Durch die individuelle Entscheidung für eine der beiden Lernmethoden auf jeder Ebene wird Arbeitsaufwand für den Experten eingespart und unnötig komplexe Demonstration werden verhindert. Zum Beispiel wird in Episode 9 eine Demonstration zum Bewegen der grünen Box angefordert, die sich als oberste Box in einem Stapel mit insgesamt drei Boxen befindet. Die Demonstration in diesem Fall beinhaltet die Auswahl der geeigneten Teilaufgabe und deren Parameter. In diesem Fall ist dies die Teilaufgabe „Bewege von Stapel“. Das System ist dann in der Lage, die benötigten primitiven Aktionen, die in dieser Teilaufgabe involviert sind, autonom auszuwählen und anzuwenden, basierend auf Erfahrungen aus vergangenen Episoden und verschiedenen Aufgabenkonfigurationen. Das Gleiche gilt für die Strategie, die benötigt wird, um einen kompletten Stapel aus drei Boxen zu versetzen, wie in Abbildung 5.8 mit dem Roboter Baxter gezeigt wird. Auch hier muss die Strategie auf der obersten Ebene dem System vorgemacht werden, auf der mittleren Ebene können die Teilaufgaben allerdings autonom ausgeführt werden. Dieses wird durch die Möglichkeit, Teilaufgaben in unterschiedlichen Aufgaben gemeinsam zu nutzen, und die Generalisierungsfähigkeiten des Systems gefördert, die die Wiederverwendung von erworbenem Wissen in verschiedenen Kontexten, unabhängig von der übergeordneten Elternaufgabe, erlaubt. Auf die gleiche Weise nutzen die beiden Teilaufgaben der mittleren Ebene dieselben vier primitiven Aktionen der tiefsten Ebene. Es gibt auf der anderen Seite auch Fälle, in denen die Aufgabe auf der obersten Ebene ohne Hilfe abgeschlossen werden kann, jedoch eine Demonstration benötigt wird, um eine Teilaufgabe auf einer niedrigeren Ebene zu lösen. Beispielsweise ist dies der Fall in Episode 5, wo eine Demonstration auf der mittleren Ebene angefragt wird, da nur wenige Informationen für diesen gegebenen Bereich des kontinuierlichen Zustandsraumes dieser Teilaufgabe vorhanden sind, wogegen die Elternaufgabe autonom gelöst werden kann.

Zusammenfassend bestätigen die Experimente, dass das System effizient lernt, verschiedene Strategien abhängig von der Anzahl der Boxen im Stapel anzuwenden. Zudem lernt das System, hohe Stapel zu zerlegen und an der Zielposition wieder zusammenzusetzen, um zu verhindern, dass der Stapel zusammenbricht. Ebenso schnell erwirbt das System die Fähigkeit, Objekte anzuheben, wenn ein Hindernis den direkten Weg zwischen der Start- und Zielposition versperrt, und wenn dies nicht der Fall ist und das Objekt direkt auf dem Tisch liegt, dieses zu verschieben, da dies die energieeffizientere Lösung darstellt. Beispiele für diese erlernten Strategien, ausgeführt auf dem Roboter Baxter, sind in Abbildung 5.9 dargestellt. Während das System zunächst bei der Aktionswahl nah an den demonstrierten Beispielen bleibt, beginnt es, sobald mehr Beispiele vorhanden sind und die Sicherheit über die erwarteten Schätzungen steigt, Lösungen aus anderen, aber ähnlichen Kontexten anzuwenden. Es versucht allerdings trotzdem nicht, beliebige Aktionen mit unbekanntem Ausgang zu testen, sondern kombiniert zuvor gesehene Aktionen neu. Zum Beispiel werden Objekte in den Episoden 22, 24 und 51 unnötigerweise angehoben. In den

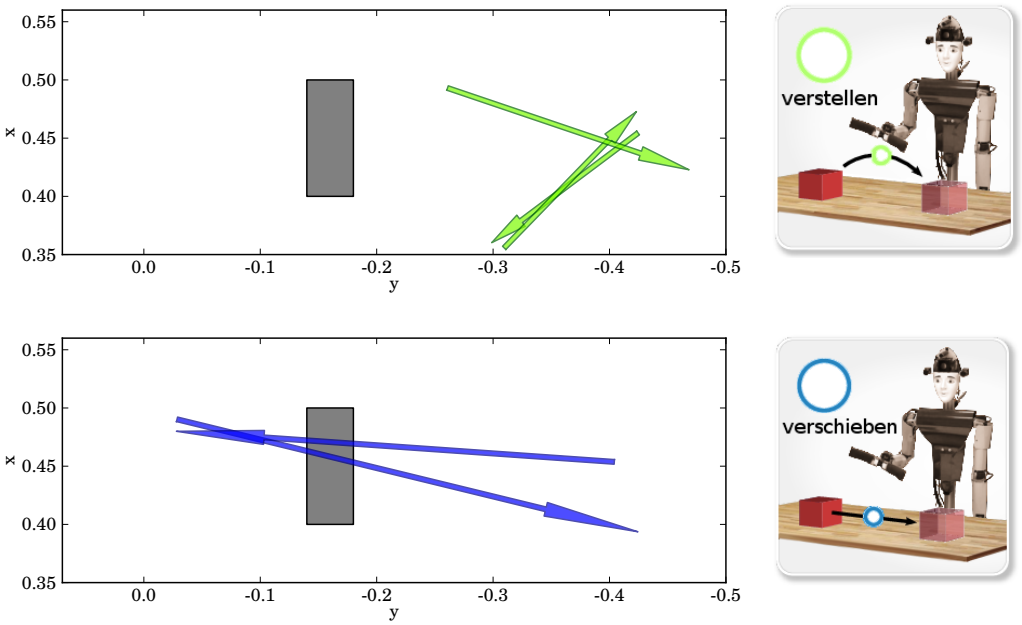


Abbildung 5.10: *Zufällig generierte Aufgaben, in denen das System eine suboptimale, autonome Wahl für die Aktionen der Teilaufgabe „Bewege Box von Tisch“ getroffen hat. Die obere Abbildung zeigt die Aufgabenstellungen, in denen die Box zur der Zielposition verstellt wird, anstatt die Box zu verschieben, was die effizientere Lösung darstellt. Die Pfeile in der unteren Grafik stellen die Aufgabenstellungen dar, in denen das System eine Kollision mit dem Hindernis verursacht, indem es versucht, das Objekt trotz Hindernis auf der direkten Linie zwischen der Start- und Zielposition zu verschieben. Der Start eines Pfeils beschreibt die Startposition der Aufgabe und seine Spitze deren Zielposition.*

Episoden 200 und 221 versucht das System die Box über den Tisch zu schieben, obwohl ein Hindernis den Weg zur Zielposition versperrt. Diese Episoden sind in Abbildung 5.10 visualisiert. Kollisionen führen zu einer bestrafenden Belohnung und die entsprechenden Aktionen werden zugunsten von korrekten Lösungen schnell aufgegeben. Daher versucht das System nur einmal in jede Richtung Lösungen, die zu Kollisionen mit dem Hindernis führen.

Im Laufe des Experimentes mit 300 Episoden sind die fünf oben erwähnten Instanzen die einzigen, in denen das System eine Teilaufgabe nicht perfekt löst. Auf der oberen Ebene gibt es keine fehlerhaften Instanzen, da das System auf die Sicherheit in der Explorationsstrategie achtet und somit verhindert, dass Aktionen ausgewählt werden, die zuvor weder in dieser noch ähnlicher Form beobachtet worden sind.

5.4 Zusammenfassung

In diesem Kapitel wird ein neues System zum hierarchischen Lernen für Roboter vorgestellt, das Bestärkendes Lernen mit Lernen aus Demonstrationen kombiniert, um Robotern komplexe Aufgaben beizubringen, wie sie beispielsweise für Serviceaufgaben

im Alltag benötigt werden. Als Bestärkendes Lernverfahren wird dazu eine Erweiterung der MAXQ-Methode vorgeschlagen, die den Einsatz dieser Methode in kontinuierlichen Zustandsräumen ermöglicht, welche ein typisches Merkmal realer Aufgaben sind. Durch Integration eines Entscheidungsmoduls, das für jede Teilaufgabe die zur Verfügung stehenden Informationen beurteilt und daraufhin eines der integrierten Lernverfahren wählt, wird eine zielgenaue Integration von Expertenwissen erreicht und die komplementären Eigenschaften des Lernens aus Demonstrationen und des Bestärkenden Lernens vereint, so dass sie sich gegenseitig ergänzen und die Vorteile beider Verfahren ausgenutzt werden.

Um die MAXQ-Methode in diesem Ansatz zu integrieren und sie zum Bestärkenden Lernen einer Strategie für eine hierarchisch strukturierte Aufgabe einsetzen zu können, werden die Komponenten der MAXQ-Zerlegung auf den verschiedenen Abstraktionsebenen mit Gauß-Prozess-Modellen approximiert. Dabei wird jede Teilaufgabe der MAXQ-Hierarchie durch einen eigenen, von den anderen unabhängigen Gauß-Prozess dargestellt. Durch diese spezielle Form der Strukturierung der Aufgabe und die dazugehörige Zerlegung der Q-Funktion können verschiedene Arten der Abstraktion genutzt werden, die ein effizientes Lernen erlauben und die Wiederverwendung einzelner Teilaufgaben möglich machen. Um eine vielversprechende, aber auch sichere Explorationsstrategie auf allen Hierarchieebenen des Bestärkenden Lernens zu generieren, wird eine Kombination der Erwarteten Verbesserung und Verschlechterung genutzt, die als Erwartete Veränderung bezeichnet wird. Diese wird berechnet, indem rekursiv Schätzungen der Gauß-Prozess-Modelle der Komponenten entlang der MAXQ-Zerlegung zu einer probabilistischen Schätzung der Q-Funktion aggregiert werden. Neben dem erwarteten Q-Wert für ein Paar aus Zustand und Aktion weist diese zusätzlich eine Unsicherheit der Schätzung auf und ermöglicht somit probabilistische Strategien zur Aktionsauswahl in Kombination mit der MAXQ-Zerlegung der Q-Funktion. In dem hier vorgestellten Verfahren werden zur Bestimmung der Lösung einer Teilaufgabe auch Informationen der tieferliegenden Ebenen in der Hierarchie berücksichtigt, indem die einzelnen Schätzungen verschiedener Teilaufgaben und ihrer Unsicherheiten kombiniert werden. Auf diese Weise entsteht ein Explorationskriterium für hierarchische Aufgaben, welches sowohl vielversprechende Aktionen auswählt als auch die Sicherheit des Roboters und seiner Umgebung gewährleistet.

Zudem wird das hierarchische Bestärkende Lernen mit dem Lernen aus Demonstrationen kombiniert, wodurch der Suchraum des Bestärkenden Lernens anhand von Expertenwissen fokussiert und somit Risiken, die durch das Ausführen von unbekannten Aktionen entstehen, gesenkt werden. Die Demonstrationen werden dazu nicht auf eine Initialisierungsphase des Systems beschränkt. Stattdessen werden beide Arten des Lernens als alternative Kontrollflüsse in das hierarchische System integriert. In jeder Situation und in jeder Teilaufgabe auf jeder Ebene entscheidet sich das System aufgrund der zuvor gesammelten Erfahrungen autonom für eines der beiden Verfahren, so dass Demonstrationen gezielt für die Situationen und Teilaufgaben eingesetzt werden, in denen sie benötigt werden, wohingegen in den anderen Fällen selbstständiges Lernen favorisiert wird. Somit wird der Aufwand für den Experten reduziert, indem unnötige Demonstrationen für bereits gelernte Aufgabenteile vermieden werden. Zudem werden Demonstration von angefragten Aufgabenteilen unter der Verwendung von Aktionen der darunter liegenden Ebene vorgeführt, so dass keine unnötig komplexen Demonstrationen über mehrere Ebenen vorgeführt werden müssen. Durch die Kombination der beiden

Lernverfahren in einem hierarchischen System wird ein effizientes Verfahren entwickelt, das zum einen von der hierarchischen Struktur an sich profitiert, zum anderen aber auch von den gegensätzlichen Stärken der beiden Ansätze, die auf jeder Hierarchieebene eingesetzt werden. Durch eine modulare Architektur kann das hier entwickelte Lernverfahren auf jeder Ebene auf die gleiche Weise eingesetzt werden, wodurch das System leicht skalierbar und um neue Ebenen erweiterbar ist.

Der Ansatz wird anhand einer Aufgabe evaluiert, in der verschieden hohe Stapel von Boxen versetzt werden sollen. Die Aufgabe ist in eine Hierarchie von Teilaufgaben gegliedert und erfordert das Lernen verschiedener Manipulationsstrategien, aus denen je nach Situation die geeignete ausgewählt werden muss. Die Ergebnisse demonstrieren, dass das vorgeschlagene System in der Lage ist, die Aufgabe erfolgreich zu lernen und Vorteile aus der hierarchischen Struktur und der unabhängigen Entscheidung für eine Lernmethode auf jeder einzelnen Ebene zu nutzen, um effizient und sicher verschiedene Arten von Stapeln aus Boxen zu versetzen. Zur Effizienz des Verfahrens tragen somit sowohl die Abstraktionen, die durch die Hierarchie möglich sind, wie auch die Fokussierung der Suche für das Bestärkende Lernen durch den Einsatz gezielter Demonstrationen bei. Bei der Untersuchung der gewählten Lernverfahren für verschiedene Situationen und Teilaufgaben ist deutlich zu sehen, dass in nur sehr wenigen Fällen in der gleichen Iteration Demonstrationen für mehrere Teilaufgaben auf unterschiedlichen Hierarchieebenen angefragt werden. Vielmehr ist dem System in vielen Fällen ein Teil der Lösung schon bekannt und kann selbstständig bearbeitet werden. Dadurch erreicht das Lernverfahren einen effizienten, aber auch sicheren Verlauf, während gleichzeitig der Arbeitsaufwand eines Experten für das Vorführen von Lösungen minimiert wird.

6 Kapitel Zusammenfassung und zukünftige Arbeiten

In der hier vorgestellten Dissertation werden verschiedene Ansätze diskutiert, mit deren Hilfe Roboter komplexe Aufgaben erlernen können. Diese Entwicklungen stellen einen Schritt auf dem Weg zu Robotern dar, die für Aufgaben im Alltag eingesetzt werden und somit ein Teil des alltäglichen Lebens werden können. Im Hinblick auf die sich dabei ergebenden Herausforderungen wird in dieser Arbeit zunächst ein Algorithmus zur Segmentierung, Klassifizierung und Generalisierung von Bewegungssequenzen vorgestellt, der seine Anwendung unter anderem beim Lernen komplexer Aufgaben anhand von Demonstrationen findet. Darauf aufbauend werden in dieser Arbeit Algorithmen entwickelt, die das Erlernen komplexer Aufgaben mittels Lernen aus Demonstrationen und Bestärkendem Lernen ermöglichen. Zunächst werden dabei sequentielle Aufgaben betrachtet und die vorgeschlagene Kombination von Lernverfahren zum Lernen der hierfür benötigten Aktionssequenzen eingesetzt. Dabei ist das hier entwickelte Lernverfahren in der Lage, mit Aktionssequenzen variabler Länge und verschiedenen Repräsentationen der Zustands- und Aktionsräume umzugehen. Dieser Ansatz wird zu einem hierarchischen Lernverfahren erweitert, das mit den komplexen Zustands- und Aktionsräumen realer Aufgaben skaliert und so ein effizientes Lernen ermöglicht.

In den folgenden Abschnitten wird zunächst eine Übersicht über die wichtigsten Beiträge dieser Arbeit gegeben. Anschließend werden die verschiedenen Richtungen diskutiert, in welche die hier vorgestellten Ansätze ausgebaut oder erweitert werden können. Dabei orientiert sich die Aufteilung an der thematischen Gliederung der Arbeit und unterteilt die beiden folgenden Abschnitte entsprechend der Kapitel zu den jeweiligen Ansätzen.

6.1 Zusammenfassung

In diesem Abschnitt werden die in dieser Arbeit vorgestellten Ansätze zusammengefasst, ihre Arbeitsweise und die Ergebnisse der Experimente erläutert sowie die Vorteile dieser Ansätze diskutiert. Dabei werden Verfahren zur Aktionserkennung und Generalisierung sowie zum Lernen von Aktionssequenzen und zum Lernen hierarchischer Aufgaben beschrieben.

Aktionserkennung und Generalisierung

Um das Wissen eines Experten nutzen und aus dessen Demonstrationen lernen zu können, muss ein Roboter in der Lage sein, Aktionen in beobachteten Bewegungssequenzen zu erkennen und sie in verschiedenen Situationen anzuwenden. Die Aufgaben des Segmentierens und Klassifizierens, die Teil des Erkennungsproblems sind, und die des Generierens stellen dabei verschiedene, eng miteinander verknüpfte Aspekte der Verarbeitung von Bewegungssequenzen dar. Viele Verfahren betrachten nur einzelne dieser Aspekte oder sie bearbeiten die Gesamtaufgabe durch eine Kombination verschiedener Ansätze, in denen jeweils eigene Modelle verwendet werden. In dieser Arbeit wird im Gegensatz dazu ein Ansatz vorgeschlagen, der ein gemeinsames probabilistisches Modell zum Lösen aller drei genannten Teilaufgaben nutzt. Zu diesem Zweck wird eine Merkmalsrepräsentation vorgeschlagen, die die Charakteristika gerichteter Bewegungen beschreibt und gleichzeitig invariant gegenüber Veränderungen ist, die sich durch die Ausführung solcher Bewegungen in verschiedenen Situationen ergeben. Auf der Grundlage dieser Repräsentation werden Hidden Markov Modelle trainiert, um eine kompakte probabilistische Repräsentation von Aktionsklassen zu erhalten, die gleichzeitig zur Segmentierung, Klassifizierung und Generierung von Bewegungen genutzt werden kann. Zur Segmentierung und Klassifizierung wird auf diesem Modell aufbauend ein kombinierter Algorithmus vorgeschlagen. Er unterteilt Trajektorien von Bewegungssequenzen inkrementell in eine Folge einfacher Aktionen, indem er Wissen über bekannte Aktionsklassen nutzt. Gleichzeitig werden systematische Fehler, die durch eine gierige Segmentierung auf Basis der zuvor beobachteten Daten entstehen, durch eine Vorausschau über das aktuelle Segment hinaus vermieden. Dies ermöglicht eine zweiseitige Segmentierungsstrategie, die bei der Bestimmung einer Segmentgrenze sowohl die Wahrscheinlichkeit berücksichtigt, dass es sich um den Endpunkt des vorausgehenden Segments handelt, als auch die Wahrscheinlichkeit, dass an diesem Punkt der Startpunkt der darauffolgenden Bewegung liegt. Neben der kompakten Repräsentation von Aktionen ermöglicht die in dieser Arbeit entwickelte Merkmalsrepräsentation durch ihre Invertierbarkeit und die Abstraktion von konkreten Situationen auch eine Generalisierung gelernter Aktionen auf unbekannte Situationen. Diese wird in dieser Arbeit genutzt, um anhand der Modelle der Aktionsklassen neue Bewegungen zwischen beliebigen Endpunkten zu generieren.

In verschiedenen Experimenten kann gezeigt werden, dass der hier vorgeschlagene Ansatz Kombinationen aus verschieden langen Sequenzen, die aus unterschiedlichen Aktionen bestehen können, mit großer Zuverlässigkeit segmentieren und klassifizieren kann. Der Algorithmus zur Bestimmung der Segmentgrenzen profitiert dabei von dem hier vorgeschlagenen, vorausschauenden Ansatz. Sogar Daten mit größeren Lücken werden durch den Ansatz korrekt segmentiert und klassifiziert. Bei Versuchen mit Sequenzen, bestehend aus jeweils drei unterschiedlichen Aktionen, können von 321 Aktionen 96% korrekt segmentiert und 100% der korrekt segmentierten Abschnitte korrekt klassifiziert werden. Zudem werden in den Experimenten glatte Bewegungen zwischen verschiedenen Endpunkten generiert, die nicht Teil der Trainingsdaten waren, und erfolgreich durch einen realen humanoiden Roboter ausgeführt.

Lernen von Bewegungssequenzen

Im zweiten Teil dieser Arbeit werden Methoden zum Erlernen von komplexen Aufgaben betrachtet, deren Lösung die Kombination mehrerer einfacher Bewegungen zu Aktionssequenzen verlangt. Um einen Roboter in die Lage zu versetzen, geeignete Aktionssequenzen zur Lösung einer solchen Aufgabe in verschiedenen Ausgangssituationen zu bestimmen, wird eine Kombination aus Bestärkendem Lernen und Lernen aus Demonstrationen vorgeschlagen. Demonstrationen stellen einen effizienten Weg dar, um dem Roboter vorhandenes Expertenwissen zugänglich zu machen. Bestärkendes Lernen ermöglicht es, autonom Lösungen für unbekannte Situationen zu bestimmen, indem vorhandenes Wissen in ähnlichen Situationen generalisiert wird. Um diese komplementären Vorteile zu nutzen, werden beide Arten des Lernens in einem kohärenten Lernverfahren integriert. Dabei ist es insbesondere in praktischen Anwendungen wichtig, dass durch das Bestärkende Lernen keine Lösungen generiert werden, bei denen das Risiko besteht, dass sie zu einer Gefährdung des Roboters oder seiner Umgebung führen. In dem hier vorgestellten Ansatz wird daher eine aktive Strategie zur Risikovermeidung verfolgt, bei der in jeder Situation evaluiert wird, ob eine sichere autonome Bestimmung einer Lösung durch Bestärkendes Lernen möglich ist. Ist dies nicht der Fall, bittet das System den Benutzer zunächst um eine Überprüfung der generierten Lösung und, falls nötig, um eine Demonstration. In diesem Punkt unterscheidet sich der Ansatz von vielen existierenden Ansätzen, die Demonstrationen lediglich als Initialisierung des Bestärkenden Lernens verwenden. Durch die Möglichkeit, Expertenwissen zu beliebigen Zeitpunkten in den Lernprozess zu integrieren, ermöglicht der Ansatz stattdessen eine flexible Anpassung oder Erweiterung der Fähigkeiten des Roboters an veränderte Rahmenbedingungen oder Aufgabenstellungen.

Beim Bestärkenden Lernen ergibt sich eine besondere Herausforderung dadurch, dass Aktionssequenzen und zustandsbeschreibende Parameter auf verschiedene Arten repräsentiert werden. Dazu kommt, dass verschiedene Strategien zum Lösen einer Aufgabe häufig eine unterschiedliche Anzahl von Aktionen beinhalten. Um diesen Problemen zu begegnen, wird die Q-Funktion über den kombinierten Raum der Zustände und Aktionssequenzen durch einen Gauß-Prozess approximiert und dabei der hier entwickelte zusammengesetzte Kernel eingesetzt, der für diese heterogenen Eingaberäume ausgelegt ist. Diese Approximation stellt gleichzeitig die Grundlage für ein Bayessches Explorationskriterium dar, das durch Optimierung der Erwarteten Veränderung Aktionssequenzen auswählt, die sowohl vielversprechend als auch sicher sind.

Die Experimente zeigen, dass mit dem Ansatz eine komplexe Aufgabe effizient gelernt werden kann und dass dabei durch die geschickte, interaktive Kombination von Bestärkendem Lernen und Lernen aus Demonstrationen der Aufwand eines Experten für das Demonstrieren von Lösungen reduziert werden kann. Zum Lösen einer komplexen Manipulationsaufgabe, in der unterschiedlich hohe Stapel von Boxen versetzt werden sollen, wird der Experte nur in 24 von 150 Episoden um Hilfe gebeten. Von diesen Anfragen müssen in nur neun Fällen Demonstrationen vorgeführt werden, während in den anderen 62% der Anfragen der Vorschlag des Systems angenommen werden kann. Zudem kann gezeigt werden, dass das System durch Optimierung der Erwarteten Veränderung sichere Aktionen bevorzugt. In den durchgeführten Experimenten kommt es so zu keiner Kollision und trotzdem wird in nur wenigen Iterationen die gewünschte Strategie gefunden.

Lernen hierarchischer Aufgaben

Im dritten Teil dieser Arbeit wird das Lernen von strukturierten Aufgaben betrachtet und ein skalierbares Lernverfahren entwickelt, das die zuvor erfolgreich beim Lernen von Aktionssequenzen angewandten Ansätze in ein hierarchisches System integriert.

Wie zum Lernen von Aktionssequenzen wird eine Kombination aus Bestärkendem Lernen und Lernen aus Demonstrationen eingesetzt, um auf diese Weise von den komplementären Vorteilen beider Verfahren zu profitieren. Die Entscheidung für eines der beiden Lernverfahren wird in jeder Iteration des Verfahrens individuell für jede Teilaufgabe auf jeder Ebene anhand des verfügbaren Wissens zur Lösung der Teilaufgabe getroffen. Sind nicht genügend Informationen über eine Situation oder eine Teilaufgabe gegeben, so wird an dieser Stelle, in der ansonsten keine sichere Auswahl der Aktion erfolgen könnte, gezielt Lernen aus Demonstrationen eingesetzt. Die entsprechende Teilaufgabe wird anhand der Aktionen der darunterliegenden Ebene vorgeführt. Dabei wird für diese Aktionen abermals eigenständig entschieden, welches Lernverfahren in diesen eingesetzt wird. Dies ermöglicht es, Expertenwissen in das Verfahren genau in den Situationen und Teilaufgaben zu integrieren, in denen dies erforderlich ist, und so den Aufwand für das Demonstrieren von Lösungen gering zu halten. Ist genügend Wissen vorhanden, so ermittelt das System eine geeignete Lösung autonom mit Hilfe des Bestärkenden Lernverfahrens. In diesem Ansatz wird dazu die MAXQ-Methode als Bestärkendes Lernverfahren integriert, welche auf hierarchischen Strukturen arbeitet. Diese wird für den Einsatz in kontinuierlichen Zustandsräumen realer Aufgaben erweitert, indem die verschiedenen Komponenten der MAXQ-Zerlegung mittels eigenständiger, unabhängiger Gauß-Prozess-Modelle approximiert werden. Aufgrund der Zerlegung sowohl der Aufgabe als auch der Q-Funktion lassen sich in dem hier vorgestellten Ansatz verschiedene Arten von Abstraktionen nutzen, so dass das Lernverfahren mit den komplexen Aufgaben des alltäglichen Lebens skalieren kann. Durch rekursive Aggregation von Gauß-Prozess-Vorhersagen und deren Unsicherheiten entlang der MAXQ-Hierarchie werden probabilistische Schätzungen der Q-Werte für beliebige Teilaufgaben berechnet. Dies ermöglicht gleichzeitig den Einsatz der Bayesschen Explorationsstrategie, so dass in diesem Ansatz auch im hierarchischen Bestärkenden Lernen vielversprechende Aktionen ausgewählt und gleichzeitig Risiken durch den unvorhersehbaren Ausgang von Aktionen vermieden werden. Durch die rekursive Berechnung der Schätzungen der Q-Werte fließen in die Wahl von Aktionen nicht nur Informationen der aktuellen Abstraktionsebene ein, sondern auch die der darunterliegenden Ebenen. Ein weiterer Vorteil des Systems liegt in seiner modularen Struktur, in der auf jeder Ebene ein einheitliches Lernverfahren verwendet wird. Somit ist das System leicht für neue Aufgaben um weitere Ebenen erweiterbar.

In den Experimenten wird gezeigt, dass das hierarchische Lernverfahren in der Lage ist, Strategien für komplexe Aufgaben zu lernen. Als Beispiel wird die bereits beim Lernen sequentieller Aufgaben verwendete Manipulationsaufgabe betrachtet und demonstriert, dass durch die hierarchische Strukturierung der Aufgabe, die Nutzung verschiedener Abstraktionen und die Kombination der beiden Lernverfahren Vorteile beim Lernen erzielt werden. Das System lernt dadurch effizient und sicher, während gleichzeitig die Anzahl der erforderlichen Demonstrationen reduziert wird. Des Weiteren zeigen die Experimente, dass Demonstrationen in nur wenigen Fällen auf mehr als nur einer Ebene gleichzeitig benötigt werden, da auf den anderen Ebenen bereits Informationen über diesen Teil der

Aufgabe vorliegen. Somit werden durch den gezielten Einsatz von Demonstrationen in dem System nur insgesamt 18 Demonstrationen einzelner Teilaufgaben benötigt, um die unterschiedlichen zum Lösen der Gesamtaufgabe benötigten Strategien während des Experimentes mit 300 Episoden zu erlernen.

6.2 Ausblick

In dieser Arbeit werden Beiträge zum Stand der Forschung in den Bereichen des maschinellen Lernens und der Robotik vorgestellt, die sich mit der Entwicklung neuer Algorithmen zum Lernen komplexer Aufgaben befassen und einen Schritt zu Robotern darstellen, die für Aufgaben im Alltag eingesetzt werden können. In diesem Abschnitt werden mögliche Erweiterungen der hier aufgezeigten Ansätze beschrieben und diskutiert.

Aktionserkennung und Generalisierung

Ein Beitrag dieser Arbeit ist ein kombinierter Ansatz zum Segmentieren, Klassifizieren und Generieren von Bewegungssequenzen. Dieser wurde in Kapitel 2 vorgestellt und im weiteren Verlauf der Arbeit zur Verarbeitung von Bewegungen beim Lernen aus Demonstrationen eingesetzt. Im Folgenden werden verschiedene Möglichkeiten zur Erweiterung dieses Ansatzes im Hinblick auf die zusätzliche Erkennung von Objekten, die durch Bewegungen manipuliert werden, eine effizientere Nutzung von Informationen und die Verwendung verschiedener Sensoren zur Aufnahme von Bewegungen skizziert.

- **Vorhersage von Objekten:** Für viele Manipulationsaufgaben ist es neben der Erkennung demonstrierter Aktionen auch relevant zu erfahren, welche Objekte durch die Bewegungen manipuliert werden. Da der bisherige Ansatz sich auf die Erkennung von Aktionen konzentriert, kann ein Ziel zukünftiger Arbeiten sein, den Ansatz entsprechend um die Erkennung relevanter Objekte zu erweitern. Dies erfordert die Entwicklung geeigneter Merkmale, die die charakteristische Beziehung zwischen den manipulierten Objekten und dem Endeffektor beschreiben. Durch Einbeziehung dieser Merkmale beim Training von Aktionsklassen kann das vorgeschlagene System neben der Klassifizierung der Bewegung auch zur Bestimmung des Objektes, mit dem diese Aktion interagiert, genutzt werden. Dabei ist weiterhin nur ein Modell pro Aktionsklasse erforderlich, das für beliebige Objekte ausgewertet werden kann.

- **Inkrementelle Verbesserung der Aktionsmodelle:** In der momentanen Implementierung des Ansatzes werden die Modelle der Aktionsklassen in einem separaten Schritt gelernt. Eine sinnvolle Weiterentwicklung besteht darin, Informationen erfolgreich segmentierter und klassifizierter Segmente zu nutzen, um die Modelle von Aktionsklassen kontinuierlich zu verbessern. Dies erlaubt es beispielsweise auch, die Anzahl der Modelle zur Laufzeit an die Daten anzupassen und so neue Modelle hinzuzufügen, falls eine Bewegung beobachtet wird, die keiner der bekannten Klassen entspricht. Zusätzlich müssen Mechanismen zum inkrementellen Lernen der Parameter der Hidden Markov Modelle gefunden werden. Eine weitere Herausforderung besteht in der Entwicklung von Ansätzen zur Vermeidung einer Verschlechterung der Modelle durch falsch zugeordnete oder falsch segmentierte Bewegungen.

- **Verwendung eines Kamerasystems:** In den Experimenten zu dem hier vorgestellten Ansatz wird zur Aufzeichnung der Bewegungsdaten eine Motion Capture-Anlage eingesetzt. Diese Art der Aufzeichnung bringt jedoch das Problem mit sich, dass sie vergleichsweise aufwändig, teuer und unflexibel ist. So müssen beispielsweise in diesem Fall die Objekte und Menschen mit speziellen Markern ausgestattet werden, damit ihre Bewegungen verfolgt werden können. Gerade für die Entwicklung von Robotern, die im Alltag eingesetzt werden sollen, ist es stattdessen wünschenswert, demonstrierte Bewegungen auf einfache Weise und an unterschiedlichen Orten aufzeichnen zu können. Eine mögliche Alternative stellen daher RGBD-Kameras dar. Diese sind kostengünstig verfügbar und portabel, so dass sie fest an einem Roboter montiert werden können und das Lernverfahren auf diese Weise mobil mit dem Roboter an beliebigen Orten genutzt werden kann. Durch die Aufnahme von Bewegungen aus der Perspektive des Roboters ergeben sich allerdings auch einige Herausforderungen. So liefern RGBD-Kameras typischerweise Daten mit geringerer Genauigkeit, und durch die einseitige Betrachtung besteht eine größere Gefahr von Verdeckungen. Dazu kommt, dass ein System entwickelt werden muss, durch das die Aufmerksamkeit des Roboters auf den Bereich gerichtet wird, in dem die Aktionen vorgeführt werden.

Lernen komplexer Aufgaben

In den in den Kapiteln 4 und 5 vorgestellten Arbeiten werden verschiedene Lernverfahren zum Lernen komplexer Aufgaben durch eine Kombination aus Bestärkendem Lernen und Lernen aus Demonstrationen vorgeschlagen. In diesen Systemen werden Gauß-Prozesse zur Approximation der Q-Funktion eingesetzt und interaktive Elemente integriert, um die benötigte Anzahl an Demonstrationen zu verringern. In den folgenden Abschnitten werden mögliche Erweiterungen dieser Ansätze vorgestellt, durch die eine automatische Wahl der Hyperparameter der Gauß-Prozess-Approximation und ein intuitiveres Lernen anhand einer zusätzlichen interaktiven Komponente erreicht wird.

- **Lernen der Kernelparameter:** Zur Approximation der Q-Funktion durch Gauß-Prozesse müssen die Hyperparameter der verwendeten Kovarianzfunktionen bestimmt werden. Diese Parameter haben einen Einfluss darauf, wie schnell das System lernt, wie viele Demonstrationen notwendig sind und wie sicher die vom System gewählten Aktionen für den Roboter und seine Umgebung sind. Für verschiedene Aspekte des Zustands- und Aktionsraumes ist es dabei möglich, unterschiedliche Parameterwerte zu wählen. Im aktuellen Lernprozess werden diese Parameter empirisch bestimmt. Eine mögliche Erweiterung des Systems besteht entsprechend in der automatischen Bestimmung dieser Parameter, so dass das System leichter an unterschiedliche Aufgaben und Umgebungen angepasst werden kann. Kernelparameter für Gauß-Prozesse können mittels statistischer Techniken anhand einer geeignet großen Menge von Trainingsdaten bestimmt werden. Da aber das Generieren von Trainingsdaten in realen Anwendungen mit einem großen Aufwand verbunden ist, ist es wünschenswert, diese Parameter iterativ während des Lernprozesses zu bestimmen und zu verbessern. Dazu können diese in regelmäßigen Abständen mit der wachsenden Anzahl an Trainingsbeispielen neu berechnet werden, um so mit der immer größer werdenden Anzahl an Beispielen die Qualität der Parameterwerte zu verbessern. Um am Anfang kein Risiko für den

Roboter und die Umgebung einzugehen, kann der Schwellwert, der im Entscheidungsmodul zur Wahl eines der beiden Lernverfahren eingesetzt wird, mit angepasst werden, so dass das System am Anfang recht vorsichtig agiert und mit der Zeit mutiger wird.

- **Erweiterungen um interaktive Elemente:** In dem hier präsentierten Ansatz wird in jeder Iteration zwischen dem Lernen aus Demonstrationen und dem Bestärkenden Lernen gewählt. Fällt die Wahl dabei auf das Lernen anhand von Expertenwissen, wird vom System ein eigener Lösungsvorschlag generiert und dem Experten vorgeschlagen. Nur wenn dieser den Vorschlag ablehnt, wird eine Demonstration erforderlich. In zukünftigen Arbeiten kann dieser interaktive Ansatz weiter ausgebaut werden, um den Menschen noch stärker in das Lernverfahren zu integrieren und sein spezifisches Wissen einem lernenden Roboter besser zugänglich zu machen. Ein Vorschlag dazu ist beispielsweise, einem Beobachter die Möglichkeit zu geben, den Roboter in seinen Aktionen jederzeit zu unterbrechen und ihn durch Sprache, Gesten oder eine Demonstration gezielt zu verbessern oder zu korrigieren, anstatt zu warten bis diese Situation ein weiteres Mal auftritt und von dem Entscheidungsmodul identifiziert wird.

Lernen von Bewegungssequenzen

In Kapitel 4 wird ein Ansatz vorgestellt, mit dem Bewegungssequenzen zum Lösen komplexer Aufgaben gelernt werden können. Dieser Ansatz kann um weitere Aktionsparameter ergänzt werden, die die Formen der Bewegungssequenzen genauer beschreiben.

- **Lernen weiterer, aktionsbeschreibender Parameter:** In dem aktuellen Lernverfahren setzen sich Aktionssequenzen aus Parametern zusammen, die die Reihenfolge von Aktionen beschreiben, und den dazugehörigen Parametern, die die Start- und Endpunkte der Bewegungen kennzeichnen. Doch dieses sind nicht die einzigen Parameter, die Bewegungen charakterisieren. Menschen passen ihre Bewegungen zusätzlich je nach der Situation auch in der Form der Bewegung an, wie zum Beispiel deren Höhe. Zudem wählen sie die Bewegungsform beziehungsweise deren Übergänge unter anderem aufgrund der vorangegangenen oder nachfolgenden Bewegung. Solche beschreibenden Parameter könnten von dem System als weitere Aktionsparameter aufgenommen werden. Dieses würde sowohl einen veränderten kombinierten Kernel benötigen wie auch ein verändertes Verfahren zur Bestimmung der Aktionssequenzen anhand der Optimierung des Wertes der Erwarteten Veränderung.

Lernen hierarchischer Aufgaben

In Kapitel 5 wird ein hierarchischer Lernansatz vorgestellt, in dem die MAXQ-Methode für hierarchisches Bestärkendes Lernen in Kombination mit einem Verfahren zum Lernen aus Demonstrationen eingesetzt wird. Die MAXQ-Methode wird dabei für den Einsatz in kontinuierlichen Zustandsräumen erweitert. In zukünftigen Arbeiten kann das Verfahren ausgebaut werden, so dass es auch in kontinuierlichen Aktionsräumen eingesetzt werden kann. Darüber hinaus ist es zur Untersuchung der Skalierbarkeit des Verfahrens interessant, dieses auf komplexere Aufgaben in größeren Anwendungsbereichen anzuwenden. Einen weiteren zu betrachtenden Aspekt stellt die kooperative Zusammenarbeit von Menschen und Robotern dar.

- **Lernen kontinuierlicher Aktionen mittels MAXQ:** In dieser Arbeit wird das MAXQ-Verfahren durch Anwendung von Gauß-Prozess-Approximationen und den Einsatz eines Bayesschen Explorationskriteriums für den Einsatz in kontinuierlichen Zustandsräumen erweitert. In vielen Anwendungen spielen jedoch auch kontinuierliche Aktionsräume eine wichtige Rolle, beispielsweise wenn Parameter zur Motorsteuerung oder Formparameter von Bewegungen gelernt werden sollen. Aus diesem Grund kann durch eine Erweiterung des Ansatzes auf kontinuierliche oder gemischte Aktionsräume das Einsatzgebiet des Verfahrens ausgebaut werden. Dies erfordert eine Weiterentwicklung der hierarchischen MAXQ-Methode in dem Bestärkenden Lernansatz sowie die Entwicklung eines Ansatzes zur Optimierung des Kriteriums der Erwarteten Veränderung in kontinuierlichen Aktionsräumen. Hierzu bieten sich gradientenbasierte Verfahren an, um effizient gute Lösungen zu finden. Dabei ergibt sich aufgrund der hierarchischen Aufgabenstruktur und der MAXQ-Zerlegung der Nutzenfunktion die Herausforderung, dass ein Gradient in diesem Verfahren ebenfalls entlang der hierarchischen Zerlegung über mehrere Ebenen bestimmt werden muss.

- **Erweiterung des Aufgabenbereiches:** In dieser Arbeit wird die Leistungsfähigkeit des hierarchischen Lernverfahrens anhand der Aufgabe untersucht, verschieden hohe Stapel aus Boxen von einer Start- zu einer Zielposition zu bewegen. Hierfür müssen verschiedene, situationsabhängige Strategien gelernt werden. In zukünftigen Arbeiten kann das System um weitere Schichten auf höheren Abstraktionsebenen erweitert werden, um so komplexere Aufgaben zu lösen und die Skalierbarkeit des Ansatzes zu untersuchen. Dabei können komplexere Aktionsräume betrachtet werden, so zum Beispiel die Zielpositionen mehrerer Objekte oder ihre Ausrichtungen untereinander, um einem Roboter komplexe Aufgaben, wie beispielsweise das Decken eines Tisches oder das Einräumen einer Spülmaschine, beizubringen. Genauso könnten Aktionen, die derzeit als primitive Bestandteile der Aufgabenhierarchie modelliert werden, weiter unterteilt werden, um bewegungsbeschreibende Parameter oder Steuerbefehle der Motoren zu erlernen. Schließlich stellt eine Ausdehnung des Aufgabenbereichs über reine Manipulationsaufgaben hinaus ein interessantes Gebiet für zukünftige Arbeiten dar. Für reale Aufgaben sind beispielsweise Navigation, verbale Kommunikation oder visuelle Wahrnehmung wichtige Fähigkeiten, die zum Lösen von Aufgaben benötigt werden. Je nach Aufgabe weisen diese individuelle Zustands- und Aktionsräume auf, die sich stark von den hier betrachteten unterscheiden. Eine interessante Fragestellung ist in diesem Zusammenhang, wie sich Ansätze zum Lernen aus Demonstrationen und zur effizienten Nutzung von Expertenwissen auf diese Aufgaben übertragen lassen.

- **Kooperatives System:** Das vorgestellte hierarchische System ist in der Lage, aktiv Demonstrationen von einem Experten anzufordern und aus diesen zu lernen. Das kooperative Lösen von Aufgaben mit Menschen wird jedoch bisher nicht betrachtet. Eine Erweiterung des Ansatzes in diese Richtung ist interessant für eine Reihe von alltäglichen Aufgaben, bei denen Menschen sich gegenseitig zur Hand gehen und zusammen arbeiten. Um Menschen bei dieser Art von Aufgaben unterstützen zu können, ist die Fähigkeit zur Zusammenarbeit auch bei Robotern wünschenswert. Dabei könnten beobachtete Demonstrationen und eigene Erfahrungen genutzt werden, um Aktionen eines Menschen vorherzusehen und ihm entsprechend zuzuarbeiten.

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[1] *Service Robot Statistics (IFR International Federation of Robotics)*. `http://www.ifr.org/service-robots/statistics/`. Online, zuletzt besucht am 24.11.2014.

[2] Kathrin Gräve und Sven Behnke. „Incremental Action Recognition and Generalizing Motion Generation based on Goal-Directed Features“. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vilamoura, Portugal*. 2012, S. 751–757.

[3] Lawrence R. Rabiner. „A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition“. In: *Proceedings of the IEEE*. 1989, S. 257–286.

[4] Faisal I. Bashir, Ashfaq A. Khokhar und Dan Schonfeld. „Object Trajectory-based Activity Classification and Recognition Using Hidden Markov Models“. In: *IEEE Transactions on Image Processing* 16.7 (2007), S. 1912–1919.

[5] Ian T. Jolliffe. *Principal Component Analysis*. Second. Springer Series in Statistics. New York: Springer, 2002.

[6] Al Mansur, Yasushi Makihara und Yashuhi Yagi. „Action Recognition using Dynamics Features“. In: *IEEE International Conference on Robotics and Automation (ICRA), Shanghai, China*. 2011, S. 4020–4025.

[7] Joris De Schutter, Enrico Di Letto, Jochem F. M. De Schutter, Roel Matthysen, Tuur Benoit und Tinne De Laet. „Recognition of 6 DOF Rigid Body Motion Trajectories using a Coordinate-Free Representation“. In: *IEEE International Conference on Robotics and Automation (ICRA), Shanghai, China*. 2011, S. 2071–2078.

[8] Mahmoud Elmezain, Ayoub Al-Hamadi und Bernd Michaelis. „Hand Trajectory-based Gesture Spotting and Recognition using HMM“. In: *International Conference on Image Processing (ICIP), Cairo, Ägypten*. 2009, S. 3577–3580.

[9] Hee-Deok Yang, A-Yeon Park und Seong-Whan Lee. „Gesture Spotting and Recognition for Human-Robot Interaction“. In: *IEEE Transactions on Robotics* 23.2 (2007), S. 256–270.

[10] Jens Kohlmorgen und Steven Lemm. „A Dynamic HMM for On-line Segmentation of Sequential Data“. In: *Advances in Neural Information Processing Systems 14*. MIT Press, 2002, S. 793–800.

[11] Zhe Li, Sven Wachsmuth, Jannik Fritsch und Gerhard Sagerer. „Manipulative Action Recognition for Human-Robot Interaction“. In: *Vision Systems: Segmentation and Pattern Recognition*. I-Tech Education und Publishing, 2007. Kap. 8.

[12] Komei Sugiura, Naoto Iwahashi, Hideki Kashioka und Satoshi Nakamura. „Learning, Generation and Recognition of Motions by Reference-Point-Dependent Probabilistic Models“. In: *Advanced Robotics* 25.6-7 (2011), S. 825–848.

[13] Aude Billard, Sylvain Calinon und Florent Guenter. „Discriminative and Adaptive Imitation in Uni-Manual and Bi-Manual Tasks“. In: *Robotics and Autonomous Systems* 54.5 (2006), S. 370–384.

[14] Sylvain Calinon, Florent Dhalluin, Darwin G. Caldwell und Aude Billard. „Handling of Multiple Constraints and Motion Alternatives in a Robot Programming by Demonstration Framework“. In: *IEEE International Conference on Humanoid Robots (Humanoids), Paris, Frankreich*. 2009, S. 582–588.

[15] Hsi G. Sung. „Gaussian Mixture Regression and Classification“. Diss. Rice University, 2004.

[16] Dana Kulić, Christian Ott, Dongheui Lee, Junichi Ishikawa und Yoshihiko Nakamura. „Incremental Learning of Full Body Motion Primitives and their Sequencing through Human Motion Observation“. In: *The International Journal of Robotics Research* 31.3 (2012), S. 330–345.

[17] Franziska Meier, Evangelos Theodorou, Freek Stulp und Stefan Schaal. „Movement Segmentation using a Primitive Library“. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), San Francisco, USA*. 2011, S. 3407–3412.

[18] Auke J. Ijspeert, Jun Nakanishi und Stefan Schaal. „Movement Imitation with Nonlinear Dynamical Systems in Humanoid Robots“. In: *Proceedings of the 2002 IEEE International Conference on Robotics and Automation (ICRA), Washington, DC, USA*. 2002, S. 1398–1403.

[19] Gernot A. Fink. *Markov Models for Pattern Recognition, From Theory to Applications*. Berlin, Heidelberg: Springer, 2008.

[20] Pierre Baldi, Yves Chauvin, Tim Hunkapliier und Marcella A. McClure. „Hidden Markov Models of Biological Primary Sequence Information“. In: *Proceedings of the National Academy of Sciences* 91.3 (1994), S. 1059–1063.

[21] Krishna Kumar Tripathi und Mahesh A. Pavaskar. „Survey on Credit Card Fraud Detection Methods“. In: *International Journal of Emerging Technology and Advanced Engineering* 2 (2012).

[22] Arthur P. Dempster, Nan M. Laird und Donald B. Rubin. „Maximum Likelihood from Incomplete Data via the EM Algorithm“. In: *Journal of the Royal Statistical Society: Series B* 39 (1977), S. 1–38.

[23] Sylvain Calinon, Florent Guenter und Aude Billard. „Goal-Directed Imitation in a Humanoid Robot“. In: *IEEE International Conference on Robotics and Automation (ICRA), Barcelona, Spanien*. 2005, S. 299–304.

[24] Carl E. Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. Adaptive Computation And Machine Learning. MIT Press, 2006.

[25] *OptiTrack Optical Motion Capture Solutions*. `http://www.naturalpoint.com/optitrack/products/motion-capture/`. Online, zuletzt besucht am 15.11.2009.

[26] *ARENA Motion Capture Software*. `http://www.naturalpoint.com/optitrack/products/full-body-mocap.html`. Online, zuletzt besucht am 15.11.2009.

[27] *NaturalPoint NatNet SDK*. `http://media.naturalpoint.com/software/OptiTrack_files/NatNetSDK1.4.zip`. Version 1.4 vom 23.04.2008.

[28] *ROS-Package mocap_optitrack*. `http://wiki.ros.org/mocap_optitrack`. Online, zuletzt besucht am 22.12.2014.

[29] Morgan Quigley, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler und Andrew Y. Ng. „ROS: An Open-Source Robot Operating System“. In: *ICRA Workshop on Open Source Software, Sendai, Japan*. 2009.

[30] A. Schliep's Group for Bioinformatics, Dep. of Computer Science, Rutgers University, NJ, USA. *The General Hidden Markov Model library (GHMM)*. Website. Available online at `http://www.ghmm.org`. 2011.

[31] *Baxter*. `http://www.rethinkrobotics.com/baxter/`. Online, zuletzt besucht am 22.10.2014.

[32] *Rethink Robotics*. `http://www.rethinkrobotics.com/`. Online, zuletzt besucht am 22.10.2014.

[33] *Baxter SDK*. `https://github.com/RethinkRobotics/baxter`. Online, zuletzt besucht am 22.10.2014.

[34] Nathan Koenig und Andrew Howard. „Design and Use Paradigms for Gazebo, An Open-Source Multi-Robot Simulator“. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Sendai, Japan*. 2004, S. 2149–2154.

[35] Sachin Chitta, Ioan A. Şucan und Steve Cousins. „MoveIt! [ROS Topics]“. In: *IEEE Robotics and Automation Magazine* 19.1 (2012), S. 18–19.

[36] Kathrin Gräve. „Lernen von Greifbewegungen aus Imitation und eigener Erfahrung“. Diplomarbeit. Universität Bonn, Nov. 2009.

[37] Kathrin Gräve, Jörg Stückler und Sven Behnke. „Learning Motion Skills from Expert Demonstrations and Own Experience using Gaussian Process Regression“. In: *International Symposium on Robotics (ISR) and German Conference on Robotics (ROBOTIK), München, Deutschland*. 2010.

[38] Kathrin Gräve, Jörg Stückler und Sven Behnke. „Improving Imitated Grasping Motions through Interactive Expected Deviation Learning“. In: *IEEE International Conference on Humanoid Robots (Humanoids), Nashville, USA*. 2010, S. 397–404.

[39] Richard S. Sutton und Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[40] Stuart J. Russell und Peter Norvig. *Artificial Intelligence: A Modern Approach*. 2. Aufl. Pearson Education, 2003.

[41] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. 1st. John Wiley & Sons, Inc., 1994.

[42] Christopher J. C. H. Watkins und Peter Dayan. „Q-Learning“. In: *Machine learning* 8.3-4 (1992), S. 279–292.

[43] Jasper Snoek, Hugo Larochelle und Ryan P. Adams. „Practical Bayesian Optimization of Machine Learning Algorithms“. In: *Advances in Neural Information Processing Systems 25*. 2012.

[44] Donald R. Jones. „A Taxonomy of Global Optimization Methods Based on Response Surfaces“. In: *Journal of Global Optimization* 21.4 (2001), S. 345–383.

[45] Hansjörg Schmidt. „Parallelisierung Ersatzmodell-gestützter Optimierungsverfahren“. Diplomarbeit. Technische Universität Chemnitz, Feb. 2009.

[46] Phillip Boyle. „Gaussian Processes for Regression and Optimisation“. Diss. Victoria University of Wellington, 2007.

[47] Adam D Bull. „Convergence rates of efficient global optimization algorithms“. In: *The Journal of Machine Learning Research* 12 (2011), S. 2879–2904.

[48] Albert Bandura und Richard H. Walters. *Social learning and personality development*. Holt Rinehart und Winston: New York, 1963.

[49] Brenna D. Argall, Sonia Chernova, Manuela Veloso und Brett Browning. „A Survey of Robot Learning from Demonstration“. In: *Robotics and Autonomous Systems* 57.5 (2009), S. 469–483.

[50] Aude Billard, Sylvain Calinon, Rüdiger Dillmann und Stefan Schaal. „Robot Programming by Demonstration“. In: *Handbook of Robotics*. Hrsg. von Bruno Siciliano und Oussama Khatib. Springer, 2008, S. 1371–1394.

[51] Kathrin Gräve und Sven Behnke. „Learning Sequential Tasks Interactively from Demonstrations and Own Experience“. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Tokyo, Japan*. 2013, S. 3237–3243.

[52] Dana Kulic, Danica Kragic und Volker Krüger. „Learning Action Primitives“. In: *Visual Analysis of Humans*. 2011, S. 333–353.

[53] Marc Toussaint, Nils Plath, Tobias Lang und Nikolay Jetchev. „Integrated Motor Control, Planning, Grasping and High-Level Reasoning in a Blocks World using Probabilistic Inference“. In: *IEEE International Conference on Robotics and Automation (ICRA), Anchorage, USA*. 2010.

[54] Nichola Abdo, Henrik Kretzschmar, Luciano Spinello und Cyrill Stachniss. „Learning Manipulation Actions from a Few Demonstrations“. In: *IEEE International Conference on Robotics and Automation (ICRA), Karlsruhe, Deutschland*. 2013.

[55] Christian Daniel, Gerhard Neumann, Oliver Kroemer und Jan Peters. „Learning Sequential Motor Tasks“. In: *IEEE International Conference on Robotics and Automation (ICRA), Karlsruhe, Deutschland*. 2013.

[56] Christian Daniel, Gerhard Neumann und Jan Peters. „Hierarchical Relative Entropy Policy Search“. In: *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS 2012), La Palma, Spanien*. 2012.

[57] Freek Stulp, Evangelos Theodorou und Stefan Schaal. „Reinforcement Learning with Sequences of Motion Primitives for Robust Manipulation“. In: *IEEE Transactions on Robotics* 28.6 (2012), S. 1360–1370.

[58] Evangelos Theodorou, Jonas Buchli und Stefan Schaal. „A Generalized Path Integral Control Approach to Reinforcement Learning“. In: *Journal of Machine Learning Research* 11 (Dez. 2010), S. 3137–3181.

[59] Gerhard Neumann, Wolfgang Maass und Jan Peters. „Learning Complex Motions by Sequencing Simpler Motion Templates“. In: *International Conference on Machine Learning (ICML)*. ICML ’09. Montreal, Quebec, Kanada, 2009, S. 753–760.

[60] Richard S. Sutton, Doina Precup und Satinder Singh. „Between MDPs and Semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning“. In: *Artificial intelligence* 112.1 (1999), S. 181–211.

[61] George D. Konidaris, Scott R. Kuindersma, Roderic A. Grupen und Andrew G. Barto. „Robot Learning from Demonstration by Constructing Skill Trees“. In: *International Journal of Robotics Research* 31.3 (2012), S. 360–375.

[62] Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini und Chris Watkins. „Text Classification using String Kernels“. In: *The Journal of Machine Learning Research* 2 (2002), S. 419–444.

[63] Thomas A. Runkler. *Data Analytics: Models and Algorithms for Intelligent Data Analysis*. Springer Vieweg, 2012.

[64] Kathrin Gräve und Sven Behnke. „Bayesian Exploration and Interactive Demonstration in Continuous State MAXQ-Learning“. In: *Proceedings of 2014 IEEE International Conference on Robotics and Automation (ICRA), Hongkong, China*. 2014, S. 3323–3330.

[65] Richard Bellman. *Dynamic Programming*. 1. Aufl. Princeton University Press, 1957.

[66] Matthew M. Botvinick. „Hierarchical Models of Behavior and Prefrontal Function“. In: *Trends in Cognitive Sciences* 12 (5 2008), S. 201–208.

[67] Bernhard Hengst. „Hierarchical Approaches“. In: *Reinforcement Learning: State of the Art*. Hrsg. von Marco Wiering und Martijn Otterlo. Bd. 12. Adaptation, Learning, and Optimization. Springer Berlin Heidelberg, 2012. Kap. 9, S. 293–323.

[68] Thomas G. Dietterich. „Hierarchical Reinforcement Learning with the MAXQ Value Function Decomposition“. In: *Journal of Artificial Intelligence Research* 13 (2000), S. 227–303.

[69] Andrew G. Barto und Sridhar Mahadevan. „Recent Advances in Hierarchical Reinforcement Learning“. In: *Discrete Event Dynamic Systems* 13.4 (2003), S. 341–379.

[70] Jens Kober, James A. Bagnell und Jan Peters. „Reinforcement Learning in Robotics: A Survey“. In: *International Journal of Robotics Research* (2013).

[71] Christian Daniel, Gerhard Neumann und Jan Peters. „Learning Concurrent Motor Skills in Versatile Solution Spaces“. In: *Proceedings of the International Conference on Robot Systems (IROS), Vilamoura, Portugal*. 2012.

[72] Jan Peters, Katharina Muelling und Yasemin Altun. „Relative Entropy Policy Search“. In: *Proceedings of the Twenty-Fourth National Conference on Artificial Intelligence (AAAI), Atlanta, USA*. 2010.

[73] Freek Stulp und Stefan Schaal. „Hierarchical reinforcement learning with movement primitives“. In: *Proceedings of the 11th IEEE-RAS International Conference on Humanoid Robots (Humanoids), Bled, Slowenien*. 2011, S. 231–238.

[74] Feng Cao und Soumya Ray. „Bayesian Hierarchical Reinforcement Learning“. In: *Advances in Neural Information Processing Systems 25*. 2012, S. 73–81.

[75] Nicholas K. Jong und Peter Stone. „Compositional Models for Reinforcement Learning“. In: *Machine Learning and Knowledge Discovery in Databases*. Springer, 2009, S. 644–659.

[76] Ronen I. Brafman und Moshe Tennenholtz. „R-Max - A General Polynomial Time Algorithm for Near-Optimal Reinforcement Learning“. In: *The Journal of Machine Learning Research* 3 (2003), S. 213–231.

[77] Geoffrey J. Gordon. *Stable Function Approximation in Dynamic Programming*. Techn. Ber. Carnegie-Mellon University, Pittsburgh, PA, 1995.

[78] Aijun Bai, Feng Wu und Xiaoping Chen. „Online Planning for Large MDPs with MAXQ Decomposition“. In: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS), Valencia, Spanien*. 2012, S. 1215–1216.