Combining Model-Based with Learning-Based Approaches for Autonomous Manipulation

zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

von der KIT-Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)

genehmigte

Dissertation

von

Dipl.-Inform. Daniel Kappler

aus Karlsruhe

Tag der mündlichen Prüfung:	19.11.2018
Erster Gutachter:	Prof. Dr-Ing. Tamim Asfour
Zweiter Gutachter:	Prof. Dr. Stefan Schaal

Zusammenfassung

Kollaboration zwischen Menschen und Robotern gewinnt zunehmend an Bedeutung in der Industrie und Forschung. Manipulation ist eine Grundvoraussetzung für eine erfolgreiche Kollaboration und deshalb eine grundlegende Forschungsfrage in der Robotik. Bei der Manipulation von Objekten, zum Beispiel beim Greifen eines Bohrers, müssen Roboter mit einer dynamischen Umgebungen, partieller Wahrnehmung, Model- und Ausführungsunsicherheit zurechtkommen. In dieser Arbeit identifizieren wir Einschränkungen von modellbasierten Ansätzen des gegenwärtigen Standes der Technik für Manipulationsaufgaben und untersuchen wie man diese mit Lernverfahren kombinieren und verbessern kann, um autonome Manipulation zu ermöglichen. Maschinelle Lernverfahren wie *neuronale Netze*, die mithilfe von großen Datenmengen ein gutes Modell lernen, sind sehr geeignet für die Robotik, da Roboter ihre Umgebung mithilfe von einer Vielzahl an Sensoren wahrnehmen und dadurch eine Fülle von Daten erzeugen. Im Gegensatz zu anderen Forschungsgebieten, wie zum Beispiel Sprach- und Bildverarbeitung, interagieren Roboter mit ihrer Umgebung, sodass Vorhersagen einen physikalischen Einfluss auf die Umgebung haben. Aufgrund der Interaktion mit der Umgebung und der kontinuierlichen Wahrnehmung ergibt sich eine Rückkopplungsschleife die neue Herangehensweisen erfordert um Sicherheitsbedenken und Geschwindigkeitsanforderungen zu erfüllen.

Das Ziel dieser Dissertation ist es zu untersuchen, wie man bestehende *modellbasierte* Robotersysteme mithilfe von *Lernverfahren* verbessern kann. Dabei ist es wichtig das vorhandene domänenspezifische Wissen nicht zu vernachlässigen, sondern in die *Lernverfahren* zu integrieren. Die Ergebnisse dieser Arbeit zeigen, dass *lernbasierte* Ansätze *modellbasierte* Methoden sehr gut ergänzen und es ermöglichen Probleme, die ansonsten unlösbar wären, zu lösen. Wir zeigen, wie man bestehende Modelle zum Trainieren von Lernverfahren verwenden kann. Dadurch wird problemspezifisches Expertenwissen in den Datengenerierungsprozess integriert und somit an das gelernte Modell weitergegeben. Wir entwickeln außerdem ein neues Optimierungsverfahren, das während der Optimierung etwas über den Vorgang an sich lernt. Ein solches Verfahren ist sehr relevant für eine Vielzahl von Problemen in der Robotik, da *autonome* Manipulationssysteme kontinuierlich neue Aufgaben lösen müssen.

Im Folgenden stellen wir die Hauptbeiträge dieser Dissertation vor, eingebettet in den Kontext von Manipulationsaufgaben.

Visuelle Wahrnehmung in Echtzeit trifft auf reaktive Bewegungsplanung Der Hauptbeitrag dieser Arbeit ist ein voll integriertes Manipulationssystem das erste einheitliche Experimente und dadurch empirische Ergebnisse ermöglicht. Diese zeigen eindeutig, dass kontinuierliche, zeitnahe Wahrnehmung und die Integration mit schnellen Verfahren zur Erzeugung von reaktiven Bewegungen essenziell für erfolgreiche Manipulation in dynamischen Szenarien ist. Wir vergleichen drei verschiedene Systeme, welche die gängigsten Architekturen im Bereich Robotik für Manipulation repräsentieren: (i) Ein traditioneller *Sense-Plan-Act* Ansatz (aktuell am weitesten verbreitet), (ii) einen myopischen Regelungsansatz, der nur auf lokale Veränderungen reagiert und (iii) ein reaktives Planungsverfahren, das auf Änderungen der Umgebung reagiert diese in die Bewegungsplanung einbezieht und den aktuellen Plan transparent an einen schnelleres lokales Regelungsverfahren übergibt. Unser Gesamtsystem ist rein *modellbasiert* und umfangreich auf einer realen Roboterplattform in vier Szenarien empirisch evaluiert worden. Unsere experimentellen Szenarien beinhalten anspruchsvolle Geometrien im Arbeitsraum des Roboters, dynamische Umgebungen und Objekte mit denen der Roboter interagieren muss. Diese Arbeit zeigt den aktuellen Stand der Forschung, der mit einem *modellbasierten* Manipulationssystem im Bereich der Robotik unter Verwendung von schnellen Rückkopplungen und langsamerer reaktiver Planung möglich ist. Angesichts des Interesses in der Robotikforschung *modellbasierte* Systeme mit *Ende-zu-Ende Lernansätzen* ganzheitlich zu ersetzen, ist es wichtig ein performantes *modellbasiertes* Referenzsystem zu haben um neue Methoden qualitativ in Hinblick auf ihre Fähigkeiten und ihre Generalisierbarkeit zu vergleichen. Weiterhin erlaubt ein solches System Probleme mit *modellbasierten* Ansätzen zu identifizieren und diese mithilfe von *learnbasierten* Methoden zu verbessern.

Online Entscheidungsfindung für Manipulation Die meisten Robotermanipulationssysteme verfügen über viele Sensoren mit unterschiedlichen Modalitäten und Rauschverhalten. Die Entwicklung von *Modellen* für alle Sensoren ist nicht trivial und die resultierende Modelle zu komplex für Echtzeitverarbeitung in *modellbasierten* Manipulationssystem. Planen mit vielen Sensormodalitäten ist besonders komplex aufgrund der vielen Modellunsicherheiten. Dies ist besonders ausgeprägt für Manipulationsaufgaben bei denen Kontakte zwischen Roboter und Objekten von Bedeutung sind. Eine der Hauptherausforderung für autonome Manipulation ist daher die Erzeugung geeigneter multimodaler Referenztrajektorien, die es ermöglichen Steuerbefehle für Regelungssysteme zu berechnen die nicht modellierte Störungen kompensieren und damit die Erfüllung der gestellten Manipulationsaufgabe ermöglichen. In dieser Arbeit stellen wir einen *lernbasierten* Ansatz zur inkrementellen Erfassung von Referenzsignalen vor, der in Echtzeit entscheidet *wann* ein Verhalten abgebrochen und zu *welchem* Verhalten gewechselt werden sollte, um eine erfolgreiche Ausführung zu gewährleisten. Wir formulieren dieses Online-Entscheidungsproblem als zwei miteinander verbundene Klassifikationsprobleme. Beide verarbeiten die aktuellen Sensormesswerte, zusammengesetzt aus mehreren Sensormodalitäten, in Echtzeit (in 30 Hz). Dieser Ansatz basiert auf unserem domänenspezifischen Problemverständnis, dass stereotypische Bewegungsgenerierung ähnliche Sensordaten erzeugt. Unsere Experimente zeigen, dass dieser Ansatz es ermöglicht schwierige kontextbasierte Aufgaben zu erlernen, die präzise Manipulation von relativ kleinen Objekten voraussetzen. Um eine solche Aufgabe zu erlernen, benötigt ein Benutzer unseres Systems kein Expertenwissen. Das System benötigt nur kinästhetische Demonstrationen und Unterbrechungen in Fehlersituationen. Die gelernte Aufgabenausführung ist robust gegen Störeinflüsse und Sensorrauschen, da unsere Methode online entscheidet, ob sie aufgrund von unerwarteter sensorischer Signale zu einer anderen Ausführung wechseln sollte oder nicht.

Big-Data Greifen Greifen ist ein wichtiges Forschungsproblem in der Robotik, da es eine Grundvoraussetzung für Manipulation darstellt. In dieser Arbeit konzentrieren wir uns auf das Problem der Vorhersage von Position und Orientierung bevor ein Kontakt zwischen Objekt und Endeffektor eintritt. Für diesen grundlegenden Schritt um "erfolgreich zu greifen" stehen nur visuelle Sensordaten wie 2D-Bilder und/oder 3D-Punktwolken zur Verfügung. Die Verwendung von *modellbasierten* Greifplanern ist in solchen Situationen nicht optimal, da präzise Simulationen zu rechenintensiv sind und alle Objekte bekannt, erkannt und visuell verfolgt werden müssen. *Lernbasierte* Verfahren die direkt von visuellen Sensordaten stabile Griffe vorhersagen sind sehr effizient in der Auswertung jedoch benötigen die aktuell vielversprechendsten Verfahren, neuronale Netze, eine Vielzahl von annotierten Beispielen um diese Abbildung zu lernen. Im Rahmen dieser Arbeit stellen wir eine umfangreichen Datenbank mit einer Vielzahl von Objekten aus sehr unterschiedlichen Kategorien vor. Auf Basis dieser Datenbank analysieren wir drei Aspekte: (i) Eine Crowdsourcing Studie zeigt, dass unsere neu vorgestellte Metrik auf Basis einer physikalischen Simulation ein besserer Indikator für Greiferfolg im Vergleich zu der bestehenden Standard ε-Metrik ist. Darüber hinaus deutet unsere Studie darauf hin, dass unsere Datengenerierung keine manuelle Datenannotation benötigt. (ii) Die daraus resultierende Datenbank ermöglicht die Optimierung von parametrischen Lernverfahren wie neuronale Netze. Dadurch, dass wir eine Abbildung von Sensordaten zu möglichen Griffen lernen, muss das Objekt, seine Position und Orientierung nicht bekannt sein. Darüber hinaus zeigen wir, dass einfachere Methoden wie logistische Regression nicht die Kapazität haben um die Komplexität unserer Daten zu erfassen. (iii) Roboter nehmen ein Szenario typischerweise aus einem Blickwinkel wahr und versuchen ein Objekt mit dem ersten Versuch zu greifen. Klassifikationsverfahren sind nicht speziell für diese Verwendung optimiert, weshalb wir eine neue Formulierung erarbeiten, welche die beste, *top-1* Hypothese aus den jeweiligen Teilmengen auswählt. Diese neuartige Optimierungszielsetzung ermöglicht dies selbst auf unserem binären Datensatz, da das Lernverfahren selbst die Daten ordnet und somit einfach zu erkennende Griffe selbst auswählen kann.

Lernen von inversen Dynamikmodellen für Manipulationsaufgaben Sichere Bewegungsausführung auf Basis von Regelungskreisen sind entscheidend für Roboter die mit Menschen kollaborativ Manipulationsaufgaben lösen. Daher werden neue Methoden benötigt, die es ermöglichen inversen Dynamikmodelle zu lernen und bestehende Modelle zu verbessern, um Verstärkungsgrößen in Regelungskreisen zu minimieren. Dies ist besonders wichtig, wenn Objekte manipuliert werden, da sich das bekannte inverse Dynamikmodell dadurch verändert. Aktuelle Verfahren, welche Fehlermodelle zu bestehenden *modellbasierten* Regler für die inverse Dynamik zu lernen, werden auf Basis der erzielten Beschleunigungen und Drehmomenten optimiert. Da die tatsächlich realisierten Beschleunigungen, eine indirekte Datenquelle, jedoch nicht die gewünschten Beschleunigungen darstellen, werden hohe Verstärkungen im Regelkreis benötigt, um relevantere Daten zu erhalten die es erlauben ein gutes Modell zu lernen. Hohe Verstärkung im Regelkreis ist wiederum schlecht für die Sicherheit. In dieser Arbeit leiten wir ein zusätzliches Trainingssignal her, das auf der gewünschten Beschleunigungen basiert und von dem Rückkopplungssignal abgeleitet werden kann. Wir analysieren die Nutzung beider Datenquellen in Simulation und demonstrieren ihre Wirksamkeit auf einer realen Roboterplattform. Wir zeigen, dass das System das gelernte inverse Dynamikmodell inkrementell verbessert. Durch die Kombination beider Datenquellen kann ein neues Modell konsistenter und schneller gelernt werden und zusätzlich werden keine hohen Verstärkungen im Regelungskreis benötigt.

Lernen wie man lernt, während man lernt Menschen sind bemerkenswert gut darin, neue oder angepasste Fähigkeiten schnell zu erlernen. Dies ist darauf zurückzuführen, dass wir nicht jede neue Fähigkeit von Grund auf neu erlernen, sondern stattdessen auf den bereits gewonnenen Fertigkeiten aufbauen. Die meisten robotergestützten Lernaufgaben würden davon profitieren, wenn sie ein solches abstraktes Meta-Lernverfahren zur Verfügung hätten. Ein solcher Ansatz ist von großer Bedeutung für die Robotik, da autonomes Lernen ein inhärent inkrementelles Problem ist. In dieser Arbeit stellen wir einen neuen *Meta-Lernansatz* vor, der es erstmals ermöglicht die Roboterdynamik online zu erlernen und auf neue Probleme zu übertragen. Während der Optimierung lernt unser Verfahren die Struktur der Optimierungsprobleme, welche für neue Aufgaben verwendet werden kann, was zu einer schnelleren Konvergenz führt. Das vorgeschlagene *Meta-Lernverfahren* kann zudem mit jedem beliebigen gradientenbasierten Optimierungsansatz verwendet werden. Wir zeigen, dass unser Ansatz die Dateneffizienz für inkrementelles Lernen erhöht. Weiterhin ist unser Verfahren für das *online Lernen* mit korrelierten Daten geeignet, zum Beispiel für inverse Dynamikmodelle. Der vorgestellte Ansatz eröffnet zusätzlich völlig neue Wege um in Simulation gewonnene Erfahrungen in die reale Welt zu transferieren. Dadurch kann möglicherweise bestehendes Domänenwissen in Form von *modellbasierter* Simulation auf völlig neue Weise verwendet werden.

Contents

Chapter 1

Introduction

1.1 Motivation

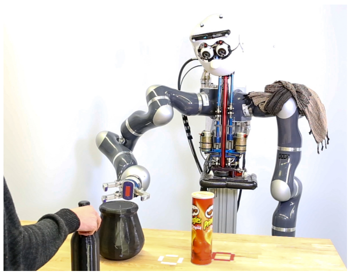
Simple, task-specific robotic manipulators were first successfully deployed for car assembly in the 1960s (Nof, Shimon Y. 1999) and have changed industrial manufacturing ever since. Nowadays, robots perform many repetitive, high precision tasks such as welding, part assembly, labeling, packaging, palletizing and quality control (Hägele et al. 2008). Such robotic manipulation systems operate on various scales, ranging from large robots moving cars, to small ones performing high precision assembly of electrical sockets. Despite the success of robotic manipulation in industry, commercial robots still have to operate in well-structured environments with limited uncertainty and known dynamics to ensure safety and task success. Thus, current robots are designed to be highly specialized for their industrial task and the manufacturing process. The environment must also be adapted to the robot capabilities to ensure high throughput. Setting up such specialized robot-based industrial pipelines requires domain knowledge, problem-specific programming, and design. As a result of the complex setup and limited autonomy, robotic systems are only applicable to industrial high-volume, highly repetitive, easy to isolate and fully observable tasks.

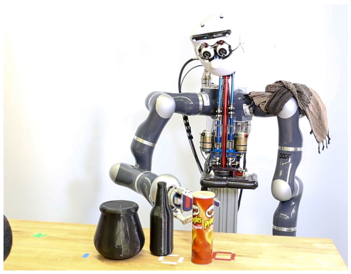
In recent years, the industrial fabrication landscape has drastically changed. There is a new demand for personalized products, resulting in more complicated, often adaptive, manufacturing steps. Unfortunately, traditional robot-based manufacturing processes do not fulfill such requirements. Highly specialized robots, often designed to perform exactly one manipulation step, are not practical for short, low-volume production cycles given the initial cost of buying, commissioning and the required task-specific programming time. Thus, there is an increasing demand for general purpose robots, which are efficiently adaptable to new manipulation tasks and human-in-the-loop setups. Leveraging the strengths of precise and repeatable robotic manipulation systems while being safe in collaborative settings poses new challenges. To address these challenges, we need to enable robots to perform *autonomous manipulation* in uncertain and dynamic environments.

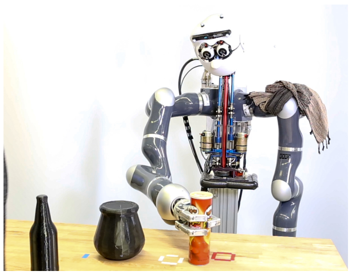
Aside from industrial applications, progress towards *autonomous manipulation* will also be very advantageous for society. For the first time in human history, the world demographic is going through a remarkable shift thanks to advances in healthcare, improvements in the standard of living and political policy changes (WHO 2011). Further, the study (WHO 2011) predicts that in the coming years the number of people aged 65 and older will for the first time outnumber children under the age of 5 years. With a lack of personnel for geriatric care (Bardach et al. 2012) in these countries, elderly care will become an even more pressing problem in the centuries to come. Autonomous robots could help to alleviate this problem, allowing the elders to stay autonomous in their own home and maintain their independence (Yamazaki et al. 2012). Safe and autonomous manipulation is necessary for robots to assist humans in everyday environments. Such highly unstructured environments pose many challenges. For example, it is impossible to know all the situations a robot manipulation platform will encounter before deployment. Further, a successful assistant will require safe human-robot physical interaction and collaboration. Therefore, robotic manipulation systems have to be adaptive and agnostic to uncertainties in their dynamically changing environments. Importantly, they also require the ability to learn novel tasks.

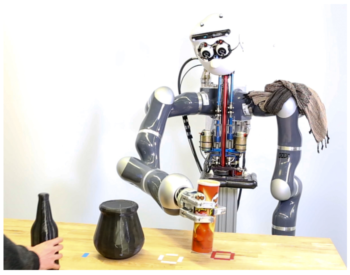
Similar to *autonomous manipulation*, autonomous driving has been a continuous research effort since the first prototypical autonomous cars in the 1980s (Wallace et al. 1985; Kanade et al. 1986; Dickmanns et al. 1992). With recent advances in sensor and actuator technology, computational hardware and data storage, both fields have gone through a dramatic shift and have shown tremendous improvements. Previously intractable approaches are now tractable and mature enough to solve the low dimensional control problems such as autonomous driving (Levinson et al. 2011). Fully integrated control system architectures have been developed, allowing to integrate additional perceptual feedback such as obstacle tracking and obstacle detection (Urmson et al. 2008). Computer vision problems, such as perceptual feedback for autonomous driving (Angelova et al. 2015), have been disrupted by *learning-based* approaches, better leveraging large datasets and increased computational resources.

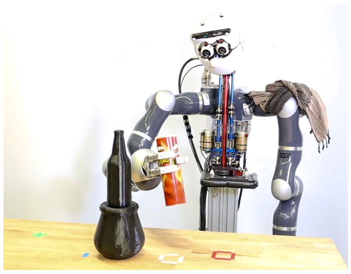
Despite similar challenges, autonomous driving is on the verge of commercialization while *autonomous manipulation* is still in its infancy. Currently, there exists no high-dimensional general purpose manipulation system capable of handling the inevitable uncertainties in unstructured human environments. Hence, naturally, the question arises: "How can we enable robotic manipulations systems to operate autonomously and safely in dynamic, uncertain environments?".


(a) Perceive the environment and the target object (Pringles box PB).


(b) Attempt to grasp the PB while avoiding the introduced obstacle.


(c) Successfully grasp the PB after the obstacle was removed.


(d) Lift the PB and plan towards the placing position.


(e) React to the introduced obstacle in real-time and alter the trajectory.

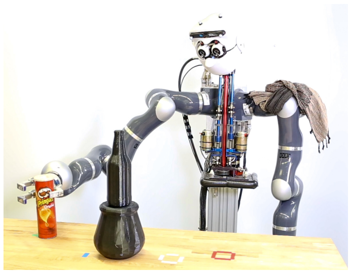

(f) Place the PB at the desired target.

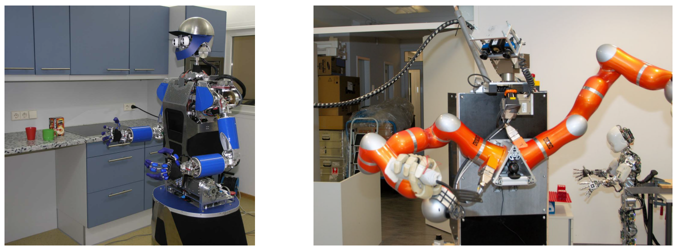
Figure 1.1: Example pick and place scenario.

1.2 Challenges in Autonomous Manipulation

Here, we discuss important challenges towards *autonomous manipulation* which hinder similar progress as observed in autonomous driving. We identify open research challenges, to which this thesis contributes novel *model-based* and *learning-based* approaches. The *simple* pick and place task in a dynamic environment depicted in Figure 1.1 demonstrates the open research challenges. In order to pick up an object, the robot must first identify, locate, and estimate the pose of the object from partial scene observations (Figure 1.1a). Upon knowing the object pose, possible grasps have to be identified from an infinite amount of possible solutions. Then, the robot has to keep track of the target object and its surroundings while attempting to approach the grasp pose (Figure 1.1b). During the approaching motion, the robot wants to avoid contacts with the environment (Figure 1.1b-1.1c), humans in the scene, in order to make deliberate contact with the object, resulting in a robust and stable grasp (Figure 1.1c). Hence, continuous re-planning and feedback integration is a necessity to achieve robust behaviors in such dynamic and uncertain scenarios. Assuming that we can generate collision-free policies, we need a controller that achieves good tracking performance. When operating in dynamic environments (Figure 1.1a-1.1f), compliant controller design is essential to ensure that the system is inherently safe. With a sufficiently good controller, tracking of the collision-free and continuously updated trajectory, the robot still has to achieve the desired contact interactions, resulting in a stable grasp such that the object does not slip (Figure 1.1c-1.1f). After the robot successfully grasps the target object, its system dynamics change, posing new challenges for the underlying control system to track the desired trajectories (Figure 1.1d-1.1e). Finally, grasping objects effectively alters the end-effector of the robot, changing the kinematics and often introducing new dynamic constraints (Figure 1.1c-1.1f). For example, a glass filled with water should be kept upright and moved such no contents are spilled.

This example illustrates the key research challenges towards *autonomous manipulation*, controlling high degree of freedom (DoF) robotic manipulation systems: successful object interaction while safely operating in highly dynamic, uncertain, and unstructured environments. A key advantage of autonomous driving is that the task is more structured. For instance, there are traffic laws, unknown objects are to be avoided, and no complicated object interactions are required. Data collection for data-intensive *learning-based* approaches, allowing for many of the recent breakthroughs in computer vision, natural language processing, and autonomous driving, are another key differentiator for *autonomous manipulation*. Static, but large labeled and unlabeled datasets are sufficient for computer vision tasks, e.g., image classification and object detection, as well as natural language processing tasks, e.g., translation. The system dynamics for autonomous driving and manipulation tasks render static dataset collection approaches impractical. Yet, autonomous driving benefits from the ability of easily obtaining a large data corpus of demonstrations by collecting sensory data and control actions during normal human driving. For *autonomous manipulation*, data collection imposes a much more significant overhead since we currently have to solve tasks by remote controlling robots, this is not only less efficient but also requires humans to learn how to operate these systems. Further, measuring human data for manipulation is incredibly difficult, e.g., human grasping data is very complex due to the complex actuation and sensing, the need to measure/account for dynamics at hand-object interactions.

In summary, *autonomous manipulation* has to adapt to new tasks, such as establishing contacts with new objects and coping with novel environment properties. The control and planning problems in *autonomous manipulation* have a much higher complexity compared to autonomous driving. Data collection requires more effort and has to incorporate system feedback for generalization. Finally, the set of possible manipulation tasks cannot be known a-priori for *autonomous manipulation* systems; thus, autonomously acquiring new skills is a fundamental requirement.



(a) ARMAR-III (Asfour et al. 2006) **(b)** TUM-Rosie (Beetz et al. 2011) **(c)** ARM-S robot, photo by Luke Fisher Photography.

Figure 1.2: Robotic manipulation systems.

1.2.1 Model-Based and Learning-Based Manipulation

Despite these significant challenges, there has been tremendous progress towards *autonomous manipulation* using *model-based* approaches. Robotic manipulation systems are complex and consist of many abstractions which make use of models, e.g., dynamics, contact, kinematic, object and cost/reward models for planning, control, and perception. Models are further used on very different time scales ranging from milliseconds to minutes. Robots such as ARMAR-III (Asfour et al. 2006) (see Figure 1.2a) have been shown to successfully operate in kitchen environments, picking up objects and placing them in a dishwasher. (Beetz et al. 2011) (see Figure 1.2b) presented a system capable of dexterous manipulation tasks such as cooking, performing dynamic pancake flipping. The Darpa Autonomous Robotic Manipulation Software (ARM-S) challenge (see Figure 1.2c) resulted in fully integrated systems (Hudson et al. 2014; Righetti et al. 2014) able to perform difficult manipulation tasks such as opening and closing doors, locating, picking up and placing objects in cluttered environments. Successful demonstrations of dual arm manipulation tasks was another outcome of the ARM-S project. For example, cutting cables with a pruning shear and unbolting, unmounting and bolting of a car wheel using standard work equipment. Nevertheless, all mentioned systems and methods require carefully designed models of the environment and the system itself, task-specific problem programming and manual robustification of environment interactions. Although the presented methods exhibit excellent generalization performance for single tasks, allow introspection and rigorous analysis, there is still a significant gap towards true autonomy. For example, it is not clear how to scale up any of the existing approaches/systems to new tasks without overcoming the manual task description engineering, and error condition tuning that is currently required.

Similar to research in computer vision, natural language processing, and autonomous driving, exploring the use of *learning-based* approaches towards autonomous manipulation is currently a highly active research field. Data-driven methods are particularly appealing for a multitude of robotics applications including manipulation (Bohg et al. 2014a). *Learning-based* methods allow solving problems leveraging sensor experience. This is especially useful in cases when no mathematical tools exist, or traditional *model-based* approaches would be computationally intractable. Further, it is unrealistic that we can pre-design *all sufficient* models required to achieve autonomous manipulation, given the need to solve novel tasks with unknown novel objects and unexpected events. Similar to *model-based* approaches, *learning-based* methods have been shown to learn single tasks successfully such as inserting a block into a shape sorting cube (Levine et al. 2016a), pick and place and opening doors (Chebotar et al. 2017). These methods often specify tasks in terms of higher level cost functions or demonstrations, often easier to define since they directly describe what we want to achieve, e.g., a binary reward for successfully grasping an object. However, the optimization itself is still an open research problem for such sparse reward settings (Riedmiller et al. 2018).

Hence, the problem of learning to solve tasks has shifted from designing the sufficient underlying models and algorithms, solvable within the system time constraints, to finding the right data representation and generation, learning and optimization method, as well as cost description. Different from autonomous driving (Urmson et al. 2008), state-of-the-art *learning-based* approaches in autonomous manipulation focus on *end-to-end* task learning (Levine et al. 2016a). Therefore, these methods neglect almost all existing model knowledge, meaning they have to learn how to control a robot from sensory observations alone for every task from scratch. Thus, very basic, fundamental and well-understood components, such as point to point movements, have to be re-learned for each task. Dynamic environments pose further challenges for end-to-end *learning-based* methods. The relevant information has to be identified automatically, often from weakly supervised data, and the right decision has to be inferred from raw sensory measurements. For example, approaches learning a mapping from pixel to torque/positions have to learn to identify the target object and the manipulator itself (Levine et al. 2016a). The field of computer vision is actively working on efficient and high-performance object detection methods to address this fundamental problem. By formulating robotic tasks as an end-to-end *learning-based* problems (Levine et al. 2016a), the underlying function approximation problem needs to solve object detection from smaller robotic datasets in dynamic environments with less variation. As a result of the problem complexity, either a significant amount of system executions or problem specific demonstrations are usually required. Finally, state-of-the-art *learning-based* methods in robotics often require an even larger number of system executions to achieve an acceptable task generalization. The resulting *learning-based* systems function as black boxes, which do not allow introspection of the learned model. Thus, safety guarantees and system insights can only be obtained by observing the system operate.

We argue that both *model-based* and *learning-based* approaches are required towards *autonomous manipulation*. We investigate the challenges of current state-of-the-art *model-based* approaches face and how to extend them with complementary *learning-based* methods. We address a diverse set of the different learning problems posed by robotics systems operating on different time scales and generating feedback loops. For example, a controller should appropriately react to changes in the system dynamics like when picking up an object. Rigid body dynamics for fixed-based manipulation robots are very well understood. A rich set of *model-based* tools that generalize well across different free space motion tasks already exist for this problem type. Hence, we do not and should not re-learn inverse dynamics models from scratch for different manipulation problems. Since robots act in the real world, creating feedback loops between our predictions and next observation, *learning-based* approaches in robotics pose additional challenges differing from successful approaches in computer vision and natural language processing. High-frequency inference and learning is a further important distinction from other more established learning problem domains. Besides posing hard constraints on the computational complexity of the learning method, high-frequency continuous data streams result in highly correlated data samples. This property violates the most common assumption of traditional learning-based methods, namely that data is distributed independently and identically. Therefore, we will propose efficient *model-based* and *learning-based* methods using domain knowledge which cope better with the inherent robotic manipulation constraints. Further, we aim to maintain the generalizability of existing *model-based* approaches, successfully used for locomotion (Herzog et al. 2016) and autonomous cars (Urmson et al. 2008), while being able to incorporate and adapt to new objects and tasks. Ideally, we aspire to build a system which requires minimal model knowledge and allows for automatic bootstrapping of all further enhancements in an autonomous manner using *learning-based* methods. For example, simulating grasps based on knowledge of the geometric object model, the robot kinematics and contact interactions to optimize a *learning-based* method which can from then on further improve with every additionally obtained data sample, e.g. on the real system. As we will show in this thesis, simulation can be a great tool to leverage existing model knowledge and explore new solution spaces which can be incorporated using *learning-based* methods.

In this thesis, we identify current limitations of previous (integrated) manipulation approaches and present novel *learning-based* extensions, overcoming the unique robotic challenges. We test our hypotheses on real robotic manipulation systems as well as within simulated environments, providing empirical evidence of the effectiveness of our key contributions.

1.2.2 Limitations of System Architectures

In addition to the challenges as mentioned earlier regarding *autonomous manipulation*, the system architecture itself poses many challenging research problems. How to best design the system architecture itself is an open problem. In order to cope with the overall problem complexity, system architectures follow a modularization, divide and conquer principle. The main idea of *separation of concerns* is to split the overall problem into separate *independent* problems. The most common example of manipulation systems is composed of these three major components, *Sense-Plan-Act* (SPA) (Kortenkamp et al. 2016). *Perception (Sense)* reasons about the system and world state from sensor measurements, observations and is also addressed by the field of computer vision. *Planning (Plan)* is responsible for generating high-level action sequences based on, e.g. state machines and collision-free trajectories. *Control (Act)* attempts to generate the right motor commands to follow and track the desired trajectories and apply the desired forces. The limited bandwidth of data exchange between the modules is a major limitation of this particular system architecture. In its most basic form SPA, does not allow for any online adaptation, while this limitation is well known in the field (R. A. Brooks 1990), it is still the most common system architecture used in robotic manipulation research. One argument for *Sense-Plan-Act* is its simplicity. It makes software development very easy and allows research to be focused on well-defined sub-problems without having to deal with the overall system complexity. To allow for some uncertainty and dynamics in the environment, *Sequential-Sense-Plan-Act* (SeSPA) is used, repeating the SPA steps at a low frequency (in the order of 0.1 Hz). This architecture proved to be very successful for complex manipulation tasks such as those in the Amazon picking challenge and the Darpa ARM-S challenge. Further, the most prominent frameworks for robotic manipulation, such as MoveIT (Chitta et al. 2012), follow this paradigm. Nevertheless, the low bandwidth and resulting limited adaptability to unforeseen environmental changes have recently been identified as a key issue by the teams of the Amazon picking challenge (Correll et al. 2016). Although additional efforts such as special hardware design can increase robustness, closing the perceptual feedback loop is a key component to achieve good performance and robustness.

Without closing the perceptual feedback loop, even slight changes in the initial condition, e.g., uncertainty in the object state estimate, can render the initially planned solution, e.g., a trajectory, infeasible. Only when a system continuously updates the world state and re-plans can it react to environmental changes, either rendering current solutions infeasible or allowing for new solutions. Closing feedback loops is by no means a new idea. It has been a long time effort in robotic manipulation, starting with the work on visual servoing by (Shirai et al. 1973) and (Hill et al. 1979). Nevertheless, fully integrated systems with closed loop perception integration are not yet common within the community.

The decomposition of the overall problem into planning and control poses even more fundamental problems besides low bandwidth communication. Since the communication between the controller and planner has a low bandwidth, it is often not clear *what* information should be communicated, and to *which* level of abstraction in the controller should it be provided. This becomes apparent for contact-based tasks. We will illustrate key issues with a simple object grasping problem, for which the robot has to grasp and lift an object. The grasp controller is responsible for tracking the desired interaction forces and pose trajectories. However, there is an inherent simulation reality gap for simulating interaction forces. Hence, the information a planner provides to the underlying controller can easily diverge very quickly from the planned trajectory. Whereas, a *simple* but well-tuned force controller often results in good grasp performance if the initial end-effector pose is *good*. Designing and tuning feedback controller for every task does not scale. Thus, the planner should be aware and choose the right controller parameterization and optimization. This raises the question of what constitutes *good* controller parameterizations such that a planner can exploit them.

1.2.3 Limitations of Model-Based Approaches

Robotics has a strong *model-based* history. Models have been extensively used on various time scales, ranging from milliseconds to minutes as well as for perception, planning, and control.

Humans design robots, therefore, kinematic models and their geometry is readily available for planning algorithms, computing collision-free trajectories. Nevertheless, planning has been shown to be an algorithmically challenging, an NP-hard problem by (Reif et al. 1985). Geometric simplifications are often required in order to find a feasible path in a reasonable time horizon. The robot's dynamics model allows to control the system with model predictive control (Allgöwer et al. 2004) or more straightforward inverse dynamics control approaches (Herzog et al. 2016). In order to achieve precise control, the robot's dynamics model has to be identified. The dynamics models might change over time due to motor heat affecting friction or object interactions such as grasping, requiring online re-identification or a detection mechanism which identifies if the underlying model has changed. This poses further challenges for designing *model-based* approaches since observed errors cannot be easily attributed to either imprecise models or algorithmic issues. For many tasks, precise models are not crucial. Rather crude approximations which facilitate fast optimal control approaches are sufficient (Herzog et al. 2016). Yet, determining sufficient partial or approximate models is a difficult manual engineering task, which is not practical for characterizing all required tasks towards *autonomous manipulation*.

Geometric environment modeling, as well as interaction planning based on geometric shapes, is another common *model-based* domain in robotic manipulation. Building an environment model consisting of all objects of interest is necessary to plan for safe trajectories. Therefore objects have to be identified and their pose estimated from noisy 2D/3D observations. This computer vision problem can be further complicated from partial occlusion or varying light conditions, or example. Since new objects are created on any given day, it is not possible to provide a robotic manipulation system with all required object models prior to deployment in a household environment. Fortunately, for many tasks only approximate local models are required. For example, stable grasps can be determined from local shape primitives (Herzog et al. 2014) which can even generalize to novel objects. Nevertheless, finding the right abstraction is again a manual engineering task, and the resulting solution is not guaranteed to solve novel other tasks since important information has been abstracted away. Additionally, object features such as inertia, the center of mass and friction coefficients, relevant quantities for manipulation tasks, are unknown, unidentifiable, uncertain or dynamically changing for many objects. Interaction is required to estimate such properties. Yet, *model-based* approaches require such features to be known to determine meaningful object interactions.

Computational complexity is one of the most fundamental problems for *model-based* methods since they are generally designed to perform the main computation at runtime. For example, contact planning is especially challenging mainly since there exists no computationally tractable contact model to date which captures the required real-world interaction dynamics. Strong assumptions are necessary to reduce the problem significantly, otherwise resulting in intractable contact planning. Hence, only a relatively small body of work, attempting to plan contacts for manipulation, exists. (Dogar et al. 2010) plan push-grasps, establishing contacts with an object on a planar surface while pushing it to increase the probability of grasp success.

In summary, successful *model-based* approaches provide new problem insights, achieve great generalization within a task, and allow for rigorous analysis. But, *model-based* methods face fundamental issues towards *autonomous manipulation* since it is simply not possible to know all models and possible tasks a priori. Therefore, we hypothesize that *learning-based* methods, improving with new data, are a necessary complement towards *autonomous manipulation*.

1.2.4 Limitations of Learning-Based Approaches

Robotic systems create large amounts of interesting data during operation. Therefore, *learning-based* methods have become a very common alternative to *model-based* approaches, especially for robotic manipulation tasks. Traditionally, different modules within the SPA system architecture have been replaced with learning approaches by collecting static datasets for learning, e.g., learning how to grasp (Pinto et al. 2016).

However, since the computer vision *end-to-end* breakthrough results on the large-scale image classification challenge ImageNet in 2012 (Deng et al. 2009; Krizhevsky et al. 2012), the robotic manipulation research community started to increasingly focus on *end-to-end* deep learning approaches to learn robotic skills. Approaches such as *guided policy search* (Levine et al. 2013) have lead to impressive results, learning control policies directly from pixel-based image inputs (Levine et al. 2016a). Yet, to achieve these impressive results the user must provide either a basic engineered control policy or a sufficiently close demonstration. Further, the learned control policies only perform well in areas around the explored state space and thus, do not generalize well (Levine et al. 2016a). Conceptually, *end-to-end* learning of a robust skill that generalizes to dynamic environments requires solving continuous object detection/identification/tracking and learning dynamics model of the robot manipulator, as well as visual servoing. Thus, it is not surprising that *end-to-end* learning frameworks for robot manipulation, using a general purpose function approximators with almost no explicit prior information and model regularization, cannot learn a general skill from a few examples on a single system. Further, all previously mentioned sub-problems are open research areas on their own.

Although very different learning methods have been used for robot manipulation tasks, deep neural networks represent the most used approach in recent literature. This framework scales well to large datasets and does not require feature engineering since it optimizes a hierarchical function approximator in an end-to-end fashion. Nevertheless, if a manipulation skill has been learned for a particular platform and environment, current state-of-the-art tools and theory do not allow to introspect the learned policies (Lipton 2016) and make no further statements about their region of validity. This issue is typically less severe for computer vision problems since deep neural networks are used for single image classification. In robotics, however, different systems have different kinematics and appearances, making it challenging to transfer learned skills from one system to another. Very different to domains in which deep learning approaches have reached industry grade maturity, robotic systems act according to the predictions, changing the learning problem significantly due to the introduced feedback. Predictions within this feedback loop can drive the robotic system into catastrophic states without the possibility of recovery or result in irreversible environmental changes. Additionally, for problems such as inverse dynamics, there are strict time requirements for inference and online learning which need to be orders of magnitude faster than in traditional supervised learning tasks.

(Pinto et al. 2016) and (Levine et al. 2017) have shown that it is possible to entirely bootstrap grasping tasks autonomously in very structured environments without explicit prior model knowledge. Still, in order to successfully bootstrap such tasks, there are fundamental implicit underlying requirements such as the initial potentially *random* actions of the robotic system have to yield a significant amount of successful task outcomes, and the task itself has to be automatically or easily resettable. The reward formulation and data distribution are further crucial aspects to successfully achieve autonomous bootstrapping and require careful tuning. Hence, manipulation tasks, such as stacking blocks in a particular order or opening doors, could not be solved in this manner on real systems without providing initial prior knowledge.

Learning-based approaches in robotics require a large amount of diverse data to achieve excellent performance, even in the case of fully supervised tasks (Levine et al. 2017). While robots typically generate lots of data, this data is highly correlated and often not diverse. Further, without strong assumptions on the structure of the system dynamics, a significant fraction of state space would have to be explored. Due to recent breakthrough results on playing Atari games, (deep) (model-free) reinforcement learning has received renewed attention. Albeit these methods do not need models and task specification is straightforward, e.g., binary rewards for the final task success, millions sometimes even billions of execution (A. Nair et al. 2017) have to be executed to learn elementary concepts. During this process, already solved robotic problems such as trajectory tracking are relearned from scratch as a necessary intermediate solution. Most research in this sub-field of *learning-based* robotics is based on simulation-based experiments due to the large number of trials required for learning. The simulation reality gap is a key issue for policy learning for real systems. Although the learning approach is model-free, high fidelity models and computationally expensive contact simulation are required to generate the data to optimize these methods.

In summary, *learning-based* methods have been shown to acquire new manipulation skills with self-supervision and allow them to adapt to changing environments. Yet, end-to-end learning is not data efficient enough to be applicable to learn a fully autonomous manipulation system. For new problems, these methods relearn previously solved problems from scratch. Hence, enhancing *learning-based* with *model-based* feedback systems should alleviate these issues.

1.3 Problem Statement

The objective of this thesis is to study how *learning-based* methods can improve the system performance by exploiting domain knowledge usually applied in *model-based* approaches. Therefore, we identify current limitations in existing state-of-the-art *model-based* system architectures for robotic manipulation tasks and focus on necessary extensions towards autonomous manipulation. Our results demonstrate that *learning-based* approaches efficiently complement *model-based* methods, allowing us to solve problems which are often otherwise intractable. We show how to use existing models to bootstrap *learning-based* methods, encoding expert problem information in the data generation process. We further introduce a new *learning-based* optimization technique relevant to robotic problems such as continual/online task learning and model predictive control problems.

1.4 Thesis Structure & Contribution

The core contributions of this thesis, embedded in the context of robotic manipulation tasks, are presented here, and their interaction visualized in Figure 1.3. First, we describe the foundations for our contributions, followed by a high-level systems perspective. Next, we focus on three

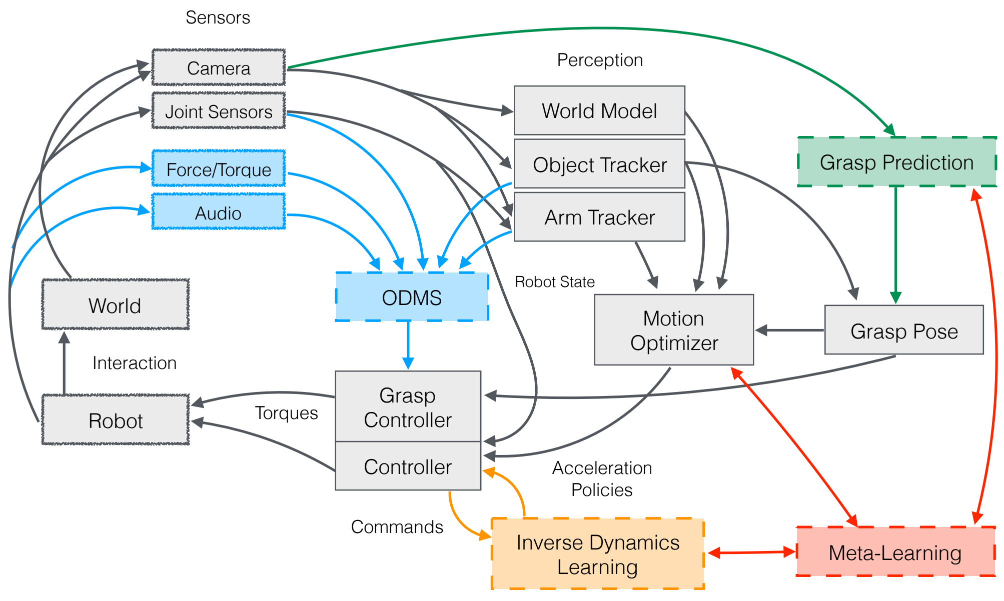


Figure 1.3: Here we embed all thesis contributions into our reactive *model-based* system (Chapter 3). All *model-based* modules are represented by boxes with solid borders. Our *learning-based* extensions have dashed borders and their additional interactions with the baseline *model-based* system are added in their respective colors (Chapter 4, Chapter 5, Chapter 6, and Chapter 7).

progressively more low-level aspects which arise in robotic manipulation tasks. After that, we present an abstract learning approach relevant to applications in robotics, as well as the broader fields of machine learning and optimization. Finally, we discuss the thesis contributions and their impact on the field of *autonomous manipulation*.

Chapter 3, Real-Time Perception meets Reactive Motion Generation: The main contribution of this work is a fully integrated system, presenting the first coherent empirical results, demonstrating the importance of continuous, real-time perception and its tight integration with reactive motion generation methods in dynamic manipulation scenarios. Three different systems that are instantiations of the most common architectures in the field are compared: (i) a traditional sense-plan-act approach (still most widely used), (ii) a myopic controller that only reacts to local environment dynamics and (iii) a reactive planner that integrates feedback control and motion optimization. The overall system is purely *model-based* and extensively evaluated on a real robotic platform in four scenarios that exhibit challenging workspace geometries and/or dynamic environments. This work shows the performance which can be achieved with a state-of-the-art *model-based* manipulation system using fast feedback. Given the rise of recent *end-to-end* *learning* approaches in robotic manipulation, neglecting almost all existing domain knowledge (e.g., in the form of models), it is especially important to have a *model-based* reference system. Hence, this work functions as a performance baseline for learning approaches, not only in terms of capabilities but also with respect to generalization. Within the context of this thesis, these results allow identifying problems which arise in purely *model-based* approaches and are suitable for *learning-based* extensions. For example, automatic online failure detection, inverse dynamics model learning, data association as well as grasp planning. This chapter is based on (**Kappler** et al. 2018).

Chapter 4, Online Decision Making for Manipulation: Most robotic manipulation systems have a very diverse set of sensor modalities with different noise characteristics. Often it is not trivial to simulate (*model-based*) all sensors; thus, it is very hard to plan with such modalities. This is especially pronounced in cases when a robotic system makes physical contact with the world, where bidirectional forces are simultaneously applied and felt by the robot. Therefore, one of the main challenges in autonomous manipulation is to generate appropriate multi-modal reference trajectories that enable feedback controllers to compute control commands that compensate for unmodeled perturbations and therefore to achieve the task at hand. In this thesis, we propose a data-driven (*learning-based*) approach to incrementally acquire reference signals from experience and decide online *when* and to *which* successive behavior to switch, ensuring successful task execution. The online decision-making problem is reformulated as a pair of related classification problems. Both process the current sensor readings, composed of multiple sensor modalities, in real-time (at 30 Hz). This approach exploits our domain specific problem understanding that movement generation can dictate sensor feedback. Thus, enforcing stereotypical behavior will yield stereotypical sensory events which can be accumulated and stored along with the movement plan. Such movement primitives (DMPs), augmented with sensor experience, are called Associative Skill Memories (ASMs) (Pastor et al. 2013b). Sensor experience consists of (real) sensors, including haptic, auditory information and visual information, as well as additional (virtual) features. Experiments show that this approach can be used to teach dexterous tasks, e.g., a bimanual manipulation task on a real platform that requires precise manipulation of relatively small objects. Task execution is robust against perturbation and sensor noise because our method decides online whether or not to switch to alternative ASMs due to unexpected sensory signals. In summary, this work demonstrated that we could *learn simple models* for various sensor modalities, which can be evaluated in real-time, to make decisions about the performance of our currently executed task. This chapter is based on (**Kappler** et al. 2015c).

Chapter 5, Big-Data Grasping: Grasping is a prerequisite for object manipulation, making it an important research problem in the robotics community. In this thesis, we focus on the problem of predicting/planning grasps prior to contact, where only visual information (e.g., 2D images and/or 3d point clouds) is available to infer a grasp pose and configuration. In such settings *model-based* approaches typically suffer from high computational complexity, require model identification from partial noisy sensor observation, and do not scale well due to a large number of available objects in the world. One promising approach is the use of deep neural networks which have been very successful in learning similar mappings, given a large set of examples. We contribute a new large-scale database that contains grasps applied to a broad set of objects from numerous categories. These grasps are generated in simulation, using our *model and domain knowledge*, and they are automatically annotated with different grasp stability metrics. Three different aspects are further analyzed within the scope of this thesis: (i) Crowdsourcing to investigate the correlation of the metrics with grasp success as predicted by humans. The results show that our proposed metric based on physics simulation is a more consistent predictor for grasp success than the standard analytical ε-metric. Further, our study suggests that human labels are not required for good ground truth grasp data. (ii) Instead, we use our domain knowledge to design a fully automatic data generation process with high-quality labels using a physics-metric that may be used to bootstrap learning in the real world. The resulting rich dataset allows us to optimize highly parametric function approximators, such as deep neural networks, to learn both the partial identification and pose/configuration regression problem. Further, these parametric models allow for very fast and computationally efficient inference. Additionally, we verify that simpler methods, such as logistic regression, do not have the capacity to leverage the large-scale database. (iii) Robots typically observe a scene from one viewpoint when attempting to grasp an object. In general, we strive for a grasp predictor which enables robots to grasp an object on the first attempt successfully. This domain specific observation enables us to reformulate the typical grasp prediction-/classification- into a top-1 prediction-problem, to obtain the best hypothesis from a binary labeled dataset. Our novel loss formulation significantly outperforms previous, more traditional classification and regression formulations by a large margin while still using the same underlying models. This chapter is based on (**Kappler** et al. 2015b, 2016).

Chapter 6, Inverse Dynamics Learning: Inherently safe robot behavior is crucial for collaborative humanoid robotic manipulation tasks. Therefore, we need methods to track precisely desired policies while still maintaining system compliance. Our previously presented approaches generate reference policies for typical robot manipulation tasks such as picking up objects. As part of this thesis, we present a novel approach which extends existing *model-based* inverse dynamics controllers with *learned error models*, translating desired accelerations into torques. Existing *model-based* approaches typically suffer from poor performance in cases of significant system payload changes. This is often the case when grasping and lifting an object. Traditional learning techniques attempt to overcome this limitation by training a full or error model on the actually realized accelerations, an indirect data source, instead of the policy's desired accelerations. We show how an additional training signal — measured at the desired accelerations — can be derived from a feedback control signal. This effectively creates a second data source for learning inverse dynamics models. We analyze the use of both data sources in simulation and demonstrate its effectiveness on a real-world robotic platform. We show that our system incrementally improves the learned inverse dynamics model. By combining both data sources, we further show that *error model learning* converges more consistently and faster. This performance improvement is likely due to our domain knowledge and understanding of the robotic task, resulting in the derivation of the proposed additional data source. This chapter is based on (**Kappler*** et al. 2017).

Chapter 7, Learning to Learn While Learning: Humans are remarkably skilled at quickly acquiring new skills, or adapting existing ones. This ability can be attributed to the fact that we do not learn each new skill from scratch, but instead build upon prior knowledge. Most robotic learning tasks would benefit from being guided by a "meta-learner", especially given that multiple skill learning is an inherently incremental problem for truly autonomous systems. In this thesis, we present a novel meta-learning approach which learns the robot dynamics online and affords transfer of the learned structure to new tasks, resulting in faster convergence. The proposed meta-learning approach is advantageous because (i) it can be combined with any gradient-based optimizer, (ii) allows learning on the fly, and (iii) can be transferred to new optimization tasks. Model-based control, often used in robotic manipulation, is particularly well suited for our approach since very similar optimization problems have to be solved in a recurrent manner. Further, our approach not only significantly reduces the amount of required observed data samples, a challenge for real robot experiments, but opens up an entirely new avenue to achieve simulation to real-world transfer. Hence, it allows us to exploit existing domain knowledge in the form of *model-based* simulation in entirely novel ways. This chapter is based on (**Kappler*** et al. 2018).

Chapter 2

Foundations and Related Work

In this thesis, we present novel *learning-based* methods to complement *model-based* approaches toward *autonomous manipulation*. Here we introduce the foundations relevant to the contributions of this thesis.

We start out providing an overview of robotic manipulation and present the main building blocks used in current state-of-the-art robotic systems towards autonomous manipulation. After that, we introduce the central machine learning concepts used throughout the thesis. After the general machine learning introduction, we focus on the core methods essential for the thesis contributions.

2.1 Robotic Manipulation

The objective of robotic manipulation is to interact with an object to achieve a desired task. For example, picking up a drill, drilling a hole and afterward placing the tool in a specific pose. Successful *autonomous manipulation* requires the robotic system to understand its environment to the degree that actions can be executed safely and efficiently. Pick and place manipulation tasks have been studied extensively throughout the last decades of robotic manipulation research. Since almost any manipulation task first requires to grasp an object to either use it as a tool or place it in a different location, pick and place is one of the most fundamental manipulation skills also addressed throughout this thesis. Manipulation tasks, in general, pose a very rich and diverse set of challenges ranging from hardware component design, fusing different sensor modalities, precisely actuating the system, and higher level task reasoning. This breadth of research components makes robotic manipulation difficult to tackle at once, even for large teams. Therefore, we first discuss different system architectures, typically used for robotic manipulation, to cope with the aforementioned complexity. The subfields, *perception*, *planning* and *control*, emerging from the inherent robotic system design are presented in more detail hereafter.

2.1.1 System Architecture

Any robotic system is composed of three components: (i) sensors allowing to estimate the system state and perceive their surrounding environment, (ii) computation to interpret and fuse the sensor information and reason about future actions toward desired goals, and (iii) actuators which physically change the system state based on the desired computed control inputs.

Modularization, a common and important abstraction for software-based systems to facilitate code and functionality reuse, component testing and *separation of concerns*, is essential for robotic manipulation systems. Manipulation tasks require complex novel research on various abstraction levels. Since robot manipulation systems are composed of distributed sensors and actuators and robotic systems operate on different operation timescales, distributed computation is often necessary to cope with the high-performance demand. In order to fulfill the inherent distributed computation requirements, most robotic system architectures are based on inter-process communication middleware such as ROS (Quigley et al. 2009), YARP (Metta et al. 2006), ARMARX (Wächter et al. 2016; Vahrenkamp et al. 2015), CORBA (Orfali et al. 1998).

To simplify the overall system complexity and facilitate reusability, three conceptual system architectures have been developed, as further discussed in (Coste-Maniere et al. 2000) and (Kortenkamp et al. 2008): (i) a top-down hierarchical approach composed of functional modules, (ii) a bottom-up behavioral approach and (iii) a hybrid of behavioral and functional modules.

2.1.1.1 Sense-Plan-Act

The most traditional approach for robotic system architectures (Moravec 1990; Crowley 1985) and more specifically robotic manipulation is Sense-Plan-Act (SPA) (R. Brooks 1986), a hierarchical top-down functional approach (Kortenkamp et al. 2016). In this paradigm, *perception* provides a model of the environment, including the robot state, using the latest sensor information. A *planner* uses the resulting world model to generate desired trajectories to solve the manipulation task. Trajectory planning is mostly concerned with avoiding collisions in free space. Typically, the goal of the planning phase is to find feasible solutions that are optimal with respect to a user-defined cost. In a final step, the systems *acts* according to the desired trajectories. In order to guarantee precision, stiff but accurate control approaches generate the underlying controls. The main advantage of this paradigm lies in its simplicity, assuming a static world state during planning and control. The subdivision of the complex problem of robotic manipulation into well defined and independent sub-fields allows for module reuse. Yet, in the case of system manipulation failures, it is difficult to attribute errors to a single module, since there exist implicit dependencies which raise questions about the right representations communicated between modules. Systems that are built according to *sense-plan-act* (SPA) are perfectly suited for environments that are well-defined, structured, controlled, and not dynamic.

However, SPA struggles to achieve robust manipulation performance in the presence of uncertainty and changing environments (Correll et al. 2016). Due to the well-known limitations of *sense-plan-act* (R. A. Brooks 1990), robotics researchers have proposed extension, e.g. *sequential sense-plan-act* (seqSPA), acknowledging the importance of environmental feedback during motion execution. Here, the robot does not only request feedback once at the very beginning but also at deliberately chosen moments during task execution. This approach is more robust against uncertainties in both sensing and actuation than SPA. Some teams competing in the Amazon Picking Challenge and the DARPA ARM Challenge followed seqSPA (Correll et al. 2016; Righetti et al. 2014). However, it is still not able to cope with dynamic environments or target objects.

2.1.1.2 Behavior-Based System Architecture

In the 1980's the limits of sense-plan-act became apparent (R. A. Brooks 1990) because of dynamic environments, slow reaction times, and system failures due to inaccurate modeling. Therefore, (R. Brooks 1986) proposed the *subsumption architecture*, a system composed of hierarchical finite state machines which all operate at the same time, considering the latest sensory feedback and inhibit each other to create complex behaviors. This paradigm, known as the *behavior-based* system architecture, has shown tremendous success for simple tasks such as fetching and delivering mail. Typically systems controlled with this paradigm exhibit very reactive behavior and can cope with rapidly changing environments. However, designing behaviors, which solve complex tasks such as manipulating objects requiring higher-level reasoning, has proven to be very difficult (Kortenkamp et al. 2008). Hybrid systems have been proposed (Gat 1992; Isla 2005; Colledanchise et al. 2014) to use the best of both worlds, behaviors to react to dynamic environmental changes and planning modules shaping the long-term behavior.

2.1.2 Perception

Perceiving the world is essential for robotic manipulation to build an internal representation of the world, facilitating planning and control. Even Shakey (Nilsson 1984), one of the first mobile robotic platforms, used multiple sensor modalities such as a camera, rangefinder, and haptic bumper sensors to perceive its surrounding. Different sensor modalities with applications in robotic manipulation are depth cameras, laser- and radio-based rangefinder, proximity, audio, and force/torque sensors, joint encoders, and accelerometers.

A common problem for manipulation tasks is to reliably identify objects in the environment and track their poses and geometries with respect to the manipulator. Ever since the low-cost Kinect (Zhang 2012) became available, RGB-D sensors have become the standard sensor modality to tackle object detection, classification, and tracking. Therefore, we focus on this sensor modality after that. With the recent computation advances and new large-scale image datasets, such as ImageNet (Deng et al. 2009), Pascal VOC (Everingham et al. 2010), and MSCOCO (Lin et al. 2014), deep learning approaches have changed the landscape of computer vision research. All state-of-the-art approaches for 2D image-based bounding box object detection (Redmon et al. 2017; Ren et al. 2017; Liu et al. 2016) and object classification (Krizhevsky et al. 2012; Simonyan et al. 2014; Szegedy et al. 2017) are now based on deep learning methods (LeCun et al. 2015). In robotics, 2D object bounding box detection and tracking (Kragic et al. 2001; Harris et al. 1990; Choi et al. 2010) is used for planning within robotic manipulation applications. Different to the computer vision community, robotics perception typically deals with small amounts of labeled data and limited datasets. Additionally, 2D image-based information is often not sufficient to infer precise object 6D poses or 3D object surfaces which are important for *model-based* control approaches. 3D perception alleviates many issues arising from the 2D ambiguities and allows robots to estimate the 6D pose, position and orientation, of the 3D objects. Depth sensing, readily available on most manipulation platforms, allows to collect 3D object models (Newcombe et al. 2011) and extend existing large-scale 3D object model datasets such as ShapeNet (Chang et al. 2015). For instance, using appearance-based and *model-based* information, (Azad et al. 2006) and (Hinterstoisser et al. 2012) propose a real-time object localization and recognition approach for solid colored known objects. (Brachmann et al. 2014) present a learning-based approach estimating the initial 6D pose of an object from a single RGB-D image. However, manipulation tasks in dynamic environments not only require the initial 6D pose of an object but continuous tracking in order to react and adapt to environmental changes. Manipulation tasks create unique challenges for 6D object tracking such as partial occlusion and rapid movements due to grasping and object interactions. While manipulating an object, the robot end-effector has to be in contact with the object which results in further difficulties for tracking since the resulting observations might be difficult to associate with either the manipulator or the object. Hence, *model-based* object tracking remains an active research topic in robotics with very different approaches such as probabilistic formulations (Wüthrich et al. 2013; Wüthrich et al. 2015) and optimization-based methods (Schmidt et al. 2014).

In addition to requirements for real-time 6D object tracking, robotic manipulation needs precise hand-eye coordination such that the desired plan can result in the desired interactions. Conceptually, there are two main approaches to ensure good hand-eye coordination. First, calibrating the kinematic parameters of the robot consisting of an offline stage at which a well-known calibration object is moved in the target workspace under the expected system conditions (Pradeep et al. 2014; Pastor et al. 2013a). Yet, this paradigm often causes issues during execution due to system changes. For example, the motor characteristics might change or sensor measurements, required for the system state estimate, might drift. Both issues result in deteriorating system calibration. Hence, tracking both, the manipulator/arm and objects in the environment is another promising paradigm. Using this methodology the same reference frame, namely, the perception sensor frame can be used to represent both tracked objects and the manipulator. Such a system automatically calibrates online with respect to the tracked manipulator and therefore does not suffer from sensor drift, cable stretch or other changes to the system state estimates. Approaches for arm tracking such as (Vahrenkamp et al. 2009b) track fiducial markers which are attached to the robot's arm or manipulator at precisely known locations, allowing the reconstruct the exact 6D-pose of the manipulator. In contrast, *model-based* methods, not requiring any visual altering of the system have been proposed by (Klingensmith et al. 2013), using the iterative closest point (ICP) algorithm (Besl et al. 1992). (Schmidt et al. 2015) present another *model-based* optimization approach using signed distance fields (SDFs) and recent computational advances in general-purpose computing on graphics processing units (GPGPU) to achieve real-time articulated tracking. A probabilistic formulation of *model-based* articulated real-time tracking for robot manipulation is discussed in (Garcia Cifuentes et al. 2017), using the high-frequency joint angle measurements in addition to low-frequency camera images, combining prior work (Wüthrich et al. 2013; Wüthrich et al. 2015).

In summary, the main perceptual methods, required for robotic manipulation research, use RGB images for appearance-based object detection, classification and identification and depth information to estimate 6D object poses and track them over time. There exists a large body of *learning-based* approaches for 2D object detection using appearance and object classification and *model-based* 3D object pose estimation and tracking work which state-of-the-art manipulation systems exploit for planning and control.

2.1.3 Planning

There are two main conceptually different planning problems for robot manipulation, discrete symbolic high-level reasoning and continuous trajectory planning. Higher-level reasoning, planning on a symbolic level, creates an abstract sequence of sub-tasks such as, first locating a particular object, then approaching the object, picking it up, using it for a task and placing it again. Motion generation, trajectory planning, is concerned with finding collision-free paths. Object interactions, albeit discrete when contact is being made, might further be considered a continuous planning problem in time. Hereafter, we focus on planning for motion generation and therefore assume that higher-level reasoning is provided to the system either in the form of task-specific programming or learning.

Motion planning has been of interest to the robotics community since the 1960s and as a result of that, free space motion generation is well understood problem (Siciliano et al. 2010). The two predominant frameworks to tackle motion planning are: (i) sampling-based approaches and (ii) optimization-based methods.

Sampling-based motion planning has intriguing properties such as completeness, meaning with enough samples random sampling-based approaches are guaranteed to find a solution if one exists. Further, sampling-based methods such as rapidly-exploring random trees (RRTs) (LaValle 1998) are global methods, meaning they do not get stuck in local minima. Although sampling-based algorithms for robot manipulation are an active field of research (Diankov et al. 2008), they generally suffer from the curse of dimensionality in high dimensional spaces. Therefore, such methods are commonly used for low-dimensional planning problems such as for autonomous driving (Kuwata et al. 2009), high-dimensional robotic manipulation tasks in static environments (Vahrenkamp et al. 2009a), but not in complex dynamic environments. Due to the random nature of most sampling procedures, solutions for similar problems, for example with similar initial and final configurations, often result in vastly different trajectory solutions. This unpredictability poses issues for human-robot interaction, a typical application for manipulation tasks (Mainprice et al. 2011). Additionally, as discussed by (Hauser et al. 2010), the resulting trajectory of most sampling-based motion planning approaches requires an additional post-processing step to generate smooth, actually executable trajectories. (Karaman et al. 2011) present anytime executable sampling-based planning methods which optimize additional cost functions while planning such as RRT* (Jeon et al. 2011) to alleviate this problem. Yet, due to their high computational requirements, such methods are impractical for high-DoF manipulation platforms.

Optimization-based planning treats trajectory generation as an optimization objective solved by iteratively estimating the Jacobian and Hessian (in case of second order methods) of the kinematic model of the robot to improve the objective. Since motion planning is an inherently non-convex problem, often with complex non-convex constraints, optimization-based approaches aim to find local rather than global minima (Schulman et al. 2014). Obstacle avoidance results in very complex optimization landscapes which can lead to poor solutions due to local minima even if other much better solution exists. Yet, different from sampling-based methods optimization-based planning does not suffer from the curse of dimensionality at the cost of sacrificing the completeness property. There are three main distinctions between different optimization-based planning methods: (i) how updates (gradients) are determined, (ii) the representation of the trajectory and how it is initialized, and (iii) the representation of the workspace geometry.

CHOMP (N. Ratliff et al. 2009) combines two objective terms, one for generating smooth trajectories and one for collision avoidance. The objective is further invariant to the trajectory parameterization. Gradients are required for both terms to perform gradient descent updates. The trajectory itself is represented by a uniformly discretized set of configurations with equal time steps. STOMP (Kalakrishnan et al. 2011a) based on a very similar trajectory representation uses a derivative-free stochastic optimization method, PI^2 (Theodorou et al. 2010). Similar to CHOMP, STOMP uses signed distance fields (SDFs) to avoid collisions with obstacles. This approach is especially efficient if the workspace is composed of known *models* since SDFs can easily be composed and allow for fast collision checks. Both are initialized with a straight line trajectory, connecting the start configuration to the goal configuration, which can be in collision with obstacles. The SDF workspace representation allows to resolve these collisions and bend the trajectory around obstacles. (Schulman et al. 2013) propose to formulate the optimization problem as a sequential quadratic program and use an iterative solver as well as convex-convex collision resolution which provides more information compared to the single point gradient of the collision. More recent work on Riemannian Motion Optimization (RieMO) by (N. Ratliff et al. 2015) changes the optimization space using a Riemannian geometry formulation which bends the space around obstacles. Different from prior work, this approach starts with a zero velocity trajectory at the starting configuration; thus, the robot is never in a collision.

The resulting trajectories of optimization-based motion planning strongly depend on the user-defined cost function formulation which is often challenging to design (Ng et al. 1999). Usually, there are infinitely many paths for a defined motion planning problem. However, humans expect robots to behave collaboratively when performing manipulation tasks in close proximity to them. This implicit knowledge can be inferred from human demonstrations in a learned objective function (Kalakrishnan et al. 2013) using inverse reinforcement learning (IRL) or inverse optimal control (IOC) formulations (Ng et al. 2000; Aghasadeghi et al. 2011; Kalakrishnan et al. 2013; Finn et al. 2016b).

So far all approaches heavily rely on the kinematic robot *model* and manually designed or *learned* cost functions. An alternative to these approaches is to build a sufficiently large corpus of *learning-based* motion primitives, e.g., Dynamic Movement Primitives (DMPs) (Ijspeert et al. 2013). Here, nominal trajectories have to be demonstrated or generated by a planner to cover a robot's workspace adequately. *Planning* in this formulation boils down to identifying the right primitive and following and adapting its nominal trajectory. Notice, primitives such as DMPs allow generalizing to new goals, significantly reducing necessary library size, and automatically adapt to new obstacles, even dynamically using feedback terms (Rai et al. 2014; Rai et al. 2016).

2.1.4 Control

Control for robotic manipulation is concerned with finding actions, e.g., torques or accelerations, for the robot which achieve a desired motion or interaction behavior. Control theory consists of two major divisions, classical time domain and state-space theory (Friedland 2012). Within the field of robotic manipulation research, control approaches are typically based on state-space formulations which we focus on hereafter.

State space approaches try to compute a policy, mapping the current state to actions, such that the underlying dynamical system fulfills a well-defined criterion. State space methods are categorized by their underlying assumptions (i) *model-based*, most notably optimal control formulations and (ii) *model-free* approaches, operating without any knowledge of the underlying dynamical system.

(Pontryagin 1962) and (Bellman 1952) contributed most foundational work in the 1950s, leading up to the extensive body of literature we have today about optimal control theory. The core idea of this framework is to compute the optimal controls according to a cost function, a task-specific design choice. In order to compute the cost of the controls, required for the optimization, the underlying *model* of the system has to be known. Optimal control methods are especially suited for the tracking of Cartesian end-effector or joint space trajectories, the bread and butter for control in robot manipulation. Simple methods such as Jacobian transpose control (Balestrino et al. 1984) and more sophisticated operational space control formulations (Khatib et al. 1986) have a long history and have matured to become an integral component of industrial robotics. (Peters et al. 2005) showed that these methods could be thought of as special cases within the more general framework of optimal control. Based on strong assumptions on the system and cost, (Kalman 1960) was able to unify this framework for state estimation and control using the "Principle of Optimality" (Bellman 1952) resulting in the *Kalman filter* and *Linear Quadratic Regulators (LQRs)*. However, assuming an unconstrained linear quadratic system or linear dynamics with quadratic cost is too restrictive for complex manipulation systems. However, promising results for high-dimensional whole-body tasks have recently been shown by (Mason et al. 2014) based on system linearization and fast feedback LQR control. Hardware improvements in computation enabled more general quadratic programming (QP) methods to achieve superior control performance on whole-body control tasks (Ames et al. 2013; Herzog et al. 2016). Despite the computational advances, the overall optimization problem is still too complex to solve for very complex models and sensitive to modeling uncertainties. Achieving high-frequency control loops is essential, especially for contact interactions. Therefore, complex QP control formulations consider only the next timestep to compute the next optimal action. Alternatively, model predictive control (MPC) (Allgöwer et al. 2004) represents an optimal control framework which optimizes over a receding horizon, meaning the optimal control problem is solved for a fixed horizon, but only the first controls are executed, and then the optimization problem is solved again. Hence, the control policy is a result of continuously solving this receding time horizon optimization problem (Erez et al. 2013; Herzog et al. 2015). Fortunately, in a high-frequency control system, the state and thus the optimization problem changes gracefully, allowing to warm start the next optimization iteration from the previous solution.

Model-free control approaches have a very long history dating back to the 17th century. Often, they are simpler to design, and they do not require to determine and identify complicated dynamics models, have less computational requirements. Further, model-free control completely sidesteps the issues that "All models are wrong" (Box 1976). PID control, the most commonly used model-free direct feedback control approach, was in simpler forms already used for control of the original steam engine systems in the 18th century. The first theoretical analysis dates back to (Minorsky. 1922). (Ogata 1990) present a more recent in-depth discussion of PID control application. The acronym for this approach stems from the control computation itself, which consists of gains scaling three different error terms, proportional (P), integral (I) and derivative (D). The ease of use of this approach, the absence of any requirements to identify complicated dynamics models, and it is well-understood behavior makes it the most fundamental control method in industrial robotics. While increasing gains can sometimes improve trajectory tracking, due to the limited bandwidth of most real systems, this eventually leads to the system going unstable. Further, high gain feedback control results in very stiff systems, not well suited for unexpected contact interactions in uncertain manipulation environments.

Reinforcement learning is another framework to determine controls. Different from optimal control approaches for which knowledge of the underlying *dynamics model* is a fundamental requirement, *model-free* reinforcement learning approaches learn how to take actions (controls) without such models by observing the system behavior. More precisely, model-free reinforcement learning is formulated as a Markov decision process (MDP) with an externally provided reward function and no transition/dynamics model. The main objective is to infer a control policy which optimizes the expected future reward for a given task. (Sutton et al. 1998b) introduce the reinforcement learning problem in more detail and discusses standard techniques and formulations such as policy gradient, q-learning, temporal difference learning, and actor-critic methods. Different methods for model-free reinforcement learning can be distinguished based on (i) how they evaluate the policy, (ii) how they update the policy and (iii) how they explore the space, with active research in robotic manipulation for all categories. A more detailed discussion of relevant approaches can be found in the recent survey by (Deisenroth et al. 2013). In light of recent advances in optimizing highly parameterized nonlinear function approximators, most notably deep neural networks (Chapter 2.2.2.4), and advances in computational hardware allowing for highly distributed simulation and learning frameworks, first results for learning policies for manipulation tasks with model-free reinforcement learning have been presented (Lillicrap et al. 2015; Heess et al. 2015; Andrychowicz et al. 2017).

Learning dynamical systems is another interesting framework for model-free control, used for dexterous robotic manipulation tasks. The key idea of these approaches is to observe the system behavior and learn a direct mapping from a measurable system state to the controls. Hence, none of the following approaches depends on any knowledge of the underlying system dynamics. (Khansari-Zadeh et al. 2012) present real-time obstacle avoidance capable policies learned from user demonstrations. Dynamic movement primitives (DMPs) (Ijspeert et al. 2013), another *learning-based* method to infer policies from demonstrations, have been extended to use feedback or coupling terms (Pastor et al. 2011b) to react to environmental changes such as moving obstacles (Rai et al. 2014; Rai et al. 2016). (Gams et al. 2010) further show how DMPs can be used to learn force-profiles in conjunction with periodic movements from visual imitation.

2.2 Learning

Machine learning is concerned with computational methods which update predictive models from experience (data), to improve on a performance measure (Mitchell et al. 1997). The abstract concept of learning is of interest to the robotic community since it allows to optimize and improve a functional representation, for instance, where to grasp an object, from data. For many complex manipulation tasks, it is difficult to develop a generic algorithm to characterize desirable states and how to achieve them for all possible situations a robot might be in. However, it is easy to provide labels in the form of successes or failures as task outcomes in hindsight; thus, acquire new data. In this context, data can be seen as the central design mechanism constraining the functional representation to learn predictive models.

The broad field of machine learning can be divided into three main directions, supervised, unsupervised, and reinforcement learning. The main distinction between supervised and unsupervised learning is the availability of labels/targets for the corresponding features and the objective. Supervised learning has a well-defined objective, learning a mapping from features to labels/targets. Unsupervised learning is generally concerned with learning the underlying structure of the data (features) and often results in ill-posed problem formulations, for instance, clustering (Kleinberg 2003). Reinforcement learning is a framework to learn policies, mapping states to actions while optimizing for the highest possible expected reward. It is distinctly different from super and unsupervised learning since it creates a feedback loop between the learning method and data generation.

In the following, we discuss supervised learning, an essential framework for this thesis. We introduce the main methods and concepts, not strictly limited to supervised learning, building the foundation of our thesis contributions. We refer the interested reader to standard textbooks (C. Bishop 2007) and (Murphy 2012) for a more detailed introduction into supervised and unsupervised learning and (Sutton et al. 1998a) for reinforcement learning.

2.2.1 Supervised Learning

The name supervised learning stems from the objective to find a mapping from features x to targets y, the supervision signal. Hereafter, we discuss differentiating aspects and concepts for supervised learning. Within the following sections, we refer to machine learning *models* (Chapter 2.2.1.2), which are models that have been trained to capture the structure of the mapping we attempt to optimize. Such learned predictive models can be used within the context of *model-based* robotic manipulation, but are not to confused with more traditional analytical models, typically used in robotics.

2.2.1.1 Dataset

Supervised learning methods use datasets

$$\mathcal{D} = \{(x,y) \in \mathcal{X} \times \mathcal{Y}\}, \tag{2.1}$$

also referred to as *data design matrix*, composed of pairs of features x and targets y. Without loss of generality both x and y might represent multi dimensional vector representations. In order to assess the performance of a learning method it is common practice to have three disjoint datasets, referred to as *train*, *validation*, and *test* sets. The *training* set is used to optimize the model (Chapter 2.2.1.2), the *validation* set to optimize the methods hyper-parameters and the *test* set to evaluate the methods generalization capabilities. In order to use this optimization methodology, the data samples composing the different datasets should be drawn from the same underlying data distribution. The most common assumption, relevant for theoretical and performance analysis, is that the data is independently, identically distributed (i.i.d.).

Relaxing requirements of the dataset $\mathcal{D}$, composition, and changing the dataset generation process, characterize distinctly different learning problems. We refer to learning problems on a dataset $\mathcal{D}$ composed of labeled pairs (x,y) and additional samples x without a corresponding label y as *semi-supervised learning*. This is a common scenario, given that manual data annotation is a time-consuming and labor-intensive process. For example, grasp hypotheses would require either manual annotation of real robot data, a self-supervision mechanism, or simulation robot experiments to obtain meaningful labels y. *Weakly labeled learning* problems operate on datasets for which targets y encode a property relevant for the mapping at hand instead of the true labels. For instance, (Oquab et al. 2015) present a method that locates objects in 2D images solely optimized from labeled data pairs containing information if an object is present or not, but not the location of the object itself. The data generation process allows categorizing learning methods into *offline* and *online* learning. Most supervised problems are tackled with *offline* learning approaches, greatly simplifying the learning problem. *Offline* learning consists of data acquisition, e.g., robot movement data (Rasmussen et al. 2006), separating the full set into the aforementioned *train*, *validation* and *test* set, followed by optimizing the model and reporting the model performance on the *test* set. However, many real-world problems result in a continuous data stream, meaning the dataset would grow indefinite and the underlying data distribution often cannot be captured with a static dataset since it is time-varying. *Online* learning methods can process such data streams and adapt to changing data distributions. Robots create an abundance of data, stemming from their high-frequency sensor measurements and control actions. Hence, efficient *online* learning is of interest to the field of robot manipulation.

2.2.1.2 Models and Methods

In order to make any predictions from previously seen data, every model and method has to make prior assumptions. These *structural* choices encode some intrinsic assumptions, such as how the observed data points are correlated. There cannot exist a single machine learning method which outperforms or is on par with all other methods on all possible datasets, domains, as stated by the no free lunch theorem (D. H. Wolpert et al. 1997). However, it is fair to assume that there exist subsets of problems, relevant to robotic manipulation or to humans in general, for which a learning algorithm can outperform all other methods since it is based on the right assumptions.

Models and methods for machine learning can be broadly categorized into three different categories: (i) *lazy* learning methods such as *nearest neighbor* (C. M. Bishop 2006) store the training data and perform inference by computing statistics on the set of k-nearest neighbors. Locally weighted regression, another example of *lazy* learning does not simply store all training examples but builds local models, lowering the inference complexity. (ii) *Eager* learning based on *parametric* models such as logistic regression (Chapter 2.2.2.3) and deep neural networks (Chapter 2.2.2.4) do not require any data samples at inference time. Such methods update a parametric function representation to explain all observed data samples. *Hyper-parameters* for *parametric* models might encode assumptions such as the functional form, or how data affects the *parametric* model. (iii) Models which grow with new data samples from the last category, typically referred to as *non-parametric* models, e.g. random forests (Ho 1995), decision trees (Quinlan 1986) and Bayesian non-parametric (Murphy 2012).

The broad family of *parametric models* is most relevant for this thesis and most commonly used in the robotics manipulation literature. In the following, we represent parametric models as

$$f(\cdot;\boldsymbol{\theta}) : \mathcal{X} \rightarrow \mathcal{Y} \tag{2.2}$$

where $\boldsymbol{\theta}$ encapsulates all parameters optimized from data. The number of the open parameters $\boldsymbol{\theta}$ in conjunction with the functional structure defines the model capacity. Generally, one can assume the more parameters, the higher the capacity of the model, typically required to express complex mappings from data. Conversely, it is often beneficial to use the least amount of parameters to explain the data, also referred to as Occam's razor (c.1287–1347).

Within the framework of probability theory, supervised learning can be formalized as follows:

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)} = \frac{p(x,y)}{p(x)} \tag{2.3}$$

$p(y|x)$ is the feature conditional, $p(x|y)$ the target conditional, $p(y)$ the target prior and $p(x)$ the marginal distribution. Hence, we can categorize our parametric model $f(\cdot;\boldsymbol{\theta})$ depending on the term it attempts to fit, either directly the left hand side $p(y|x)$ (*discriminative*) or the right hand side (*generative*).

Generative Model: *Generative* models allow synthesizing data, reason about missing features x and transparently encode prior knowledge about the targets y and model knowledge of the data generating process. Hence, if we have a good model of the underlying data generation process, less data is required to achieve good regression and classification results. However, a severely wrong prior model introduces a strong bias. Thus, the resulting fitted model might not achieve good performance. Generative models often lend themselves for better analysis since the assumptions about the data generation process are explicitly expressed in the model. Additionally, we can draw samples from the model and therefore get insights about the shortcomings of the model. Further, generative models provide an uncertainty estimate, therefore, allow to better reason about their predictions. Since robotic manipulation systems act in their environment, can actively generate and acquire new data, uncertainty estimates can be exploited to make more informed decisions about future actions. However, capturing uncertainty as well as modeling the data generation process is a much more complex learning problem compared to simple discriminative learning discussed hereafter. For example, in order to classify objects, we might not be interested in rendering natural images, the inverse graphics problem. More related to manipulation, if we are interested in inferring if a grasp hypothesis results in a stable or unstable grasp from RGB-D sensor observations, we do not require to understand how realistic partially occluded depth maps are generated and used to inform a grasp.

Discriminative Model: The second model class attempts to directly learn a mapping from features x to targets y, therefore learning a *discriminative* function $p(y|x)$. For classification (Chapter 2.2.1.6) this results in learning a decision boundary whereas for regression (Chapter 2.2.1.5) in predicting the values of y. Models in this category are often computationally less expensive and scale better with large-scale datasets and high dimensional feature spaces compared to generative models. Further, since this model class learns a direct mapping, the inference is straightforward, requires simple model evaluation. Hence, discriminative models result in fast inference. Different from generative models, information about the underlying data distribution has to be learned implicitly during the optimization. Therefore, large datasets are essential for good performance, usually resulting in gradient-based optimization methods which have been studied extensively in the optimization literature (Bottou et al. 2018; Nocedal et al. 1999). Although discriminative models are very flexible and allow to encode the assumptions implicitly, it is often not clear how to encode domain knowledge. For instance, how to use existent knowledge about a robots kinematics, useful for robot manipulation learning tasks, within a discriminative learning approach such as deep neural networks (Chapter 2.2.2.4). Introspection of the resulting models is another distinction to generative models. Understanding the biases of the learned model and analyzing error cases is inherently difficult. Additionally, many state-of-the-art discriminative models do not provide well-calibrated uncertainty estimates. Thus, it is unclear if a robotic system should rely on, e.g. torque predictions from such methods since there is no assurance that the resulting predictions are safe to execute on.

Global Model: Supervised learning approaches attempt to capture the important structure in the feature space to infer targets y in an efficient manner. Global models such as Support Vector Machines (SVMs) (Chapter 2.2.2.1) and deep neural networks (DNNs) (Chapter 2.2.2.4) consist of one model $f(\cdot;\boldsymbol{\theta})$ for all data samples in $\mathcal{D}$. Therefore, different data samples $(x,y) \in \mathcal{D}$ affect the prediction for all other data samples $(x',y') \in \mathcal{D}\backslash(x,y)$. In the case of discriminative approaches, this property facilitates extrapolation since the parametric model does not encode a notion of proximity to the training data. Gaussian Process (Rasmussen et al. 2006), a generative Bayesian approach, does encode proximity to the training data, essential to obtain a measure of uncertainty. Still, the open parameters are optimized based on all data points which can result in worse performance if the underlying data distribution has very different noise characteristics in different parts of the feature space. Continuous *online* learning is another challenge for global models. Since updates have a global effect, *old data* has to be revisited to not *forget* about its mapping. Generally speaking, a representative dataset has to be maintained; otherwise, the model performance degenerates concerning the previously learned mapping.

Local Model: In order to better cope with shifting data distributions and streaming data, local learning methods allocate new local models over time such as locally weighted regression (LWR) (C. G. Atkeson et al. 1997) and incremental local Gaussian regression (iLGR) (Meier et al. 2014b). Learning the "optimal" similarity metric is a challenging task for this learning regime since a globally accessible proximity evaluation is vital to scale such methods to high dimensional features spaces. Further, changing the similarity metric over time introduces further complexity since efficient data structures such as k-d trees (Bentley 1975) required for fast proximity computation would have to be updated over time as well. Other challenges are adding models over time and altering local models while maintaining the best possible performance. Extrapolation with local models is posing further difficulties since global effects and structures are not captured; thus, it cannot be used to extrapolate to unseen parts of the feature space.

2.2.1.3 Objective

For the following objectives, we always assume that the dataset $\mathcal{D}$, used for optimization, is i.i.d. Hereafter, we present three different objectives, common in machine learning and robotic learning applications.

Frequentist: Maximum likelihood estimation (MLE) is a common method to obtain a point estimate of the model parameters $\boldsymbol{\theta}$ given a dataset $\mathcal{D}$. This *frequentist* method assumes no prior information on the model parameter distribution. Since machine learning methods are often formulated as minimization problems and the logarithm is a monotonic increasing function, we formulate the two main MLE objectives as minimizing the negative log likelihood. Discriminative models directly optimize

$$\underset{\boldsymbol{\theta}}{\operatorname{argmin}} - \sum_{(x,y)\in\mathcal{D}} \log p(y|x;\boldsymbol{\theta}), \tag{2.4}$$

the conditional negative log likelihood estimate, a point estimate of $\boldsymbol{\theta}$, maximizing the conditional probability of predicting $p(y|x;\boldsymbol{\theta})$. Generative models optimize the model likelihood

$$\underset{\boldsymbol{\theta}}{\operatorname{argmin}} - \sum_{(x,y)\in\mathcal{D}} \log p(y|x;\boldsymbol{\theta}) = - \sum_{(x,y)\in\mathcal{D}} \log p(x|y;\boldsymbol{\theta}) + \log p(y) - \log p(x). \tag{2.5}$$

MLE allows for efficient gradient descent based optimization algorithms, suitable for complex problems with large sample sizes. It further is guaranteed to converge to the true parameter in the limit of examples if the true data distribution is within the parametric family of $p(\cdot;\boldsymbol{\theta})$ and is generated by at least one value $\boldsymbol{\theta}$ (C. M. Bishop 2006). Hence, MLE optimization approaches tend to require regularization, see optimization based perspective, to limit the effect of overfitting to the data. Overfitting basically results in remembering the observed data points without allowing for inter- or extrapolation.

Bayesian: Different from frequentist methods, Bayesian approaches treat parameters as random variables. Prior subjective information about the parameters $\boldsymbol{\theta}$ is provided in $p(\boldsymbol{\theta})$ encoding all problem-specific information. In order to use Bayesian inference, the posterior distribution over the parameters has to be computed given the dataset

$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{\int p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta}}. \tag{2.6}$$

Using the posterior distribution $p(\boldsymbol{\theta}|\mathcal{D})$ the parameters $\boldsymbol{\theta}$ are integrated out to infer the conditional probability

$$p(y^*|x^*,\mathcal{D}) = \int p(y^*|x^*,\boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{D})d\boldsymbol{\theta}. \tag{2.7}$$

For most interesting distributions integrating out the parameters is computationally intractable. But since the parameters are treated probabilistically this problem fromulation is less prone to overfitting. Optimization is performed on the parameters of the prior distribution, resulting in a hyper-parameter optimization problem.

The maximum a posteriori (MAP) objective

$$\operatorname*{argmin}_{\boldsymbol{\theta}} - \sum_{(x,y)\in\mathcal{D}} \log p(y|x,\boldsymbol{\theta}) + \log p(\boldsymbol{\theta}), \tag{2.8}$$

alleviates the computational burden of integrating out the parameters $\boldsymbol{\theta}$. The main difference compared to MLE is the additional prior term $\log p(\boldsymbol{\theta})$ which can be interpreted as a regularizer, informed by prior problem specific knowledge. In case of a uniform prior, the MAP objective recovers the MLE objective. Similar to MLE, the MAP objective is prone to overfitting, weakening the proper probabilistic interpretation.

Optimization Perspective: Many methods without a probabilistic interpretation such as max-margin classification, deep neural network architectures, and decision trees are typically formulated as a minimization problem. The canonical objective for supervised learning consists of the sum of all prediction errors on the training dataset:

$$\operatorname*{argmin}_{\boldsymbol{\theta}} \sum_{(x,y)\in\mathcal{D}} l(y, f(x;\boldsymbol{\theta})) + r(\boldsymbol{\theta}) \tag{2.9}$$

This formulation is a generalization of maximum likelihood estimation for $p(y|x)$ assuming $l(y, f(x;\boldsymbol{\theta})) = -\log(p(y|x))$ and $\boldsymbol{\theta}$ and $r(\boldsymbol{\theta}) = 0$ or $r(\boldsymbol{\theta}) = -logp(\boldsymbol{\theta})$ for maximum a posteriori estimation. Here, $r(\boldsymbol{\theta})$ acts as a regularizer, biasing the model to avoid singularities, disambiguating multiple solutions. A regularizer is also commonly used to constrain, bias the solutions of the resulting model to be sparse, e.g. using $||\boldsymbol{\theta}||_1$ to encourage many parameters to be zero. This popular approach allows learning a rich feature representation in a higher dimensional space, such that linear separation or linear combination is encouraged. Data overfitting, common for discriminative methods, is another reason for regularization. An overfitted model might have resulted in a better, more compressed dataset representation, yet, if the training dataset does capture all test examples, an overfitted model results in bad inference performance. Underfitting is a common problem for generative models. Here, a regularizer might be required to encourage a model to generate more diverse predictions. For instance, not capturing perceptual details in images and missing modes of the data distribution is a common result of model underfitting. Recent game theoretic inspired optimization approaches such as generative adversarial networks (GANs) (I. Goodfellow et al. 2014) have shown first results to alleviate these issues.

The loss in the optimization objective (Equation 2.9) ultimately encodes the desired outcome of the learning problem. In the case of binary classification (Chapter 2.2.1.6), for which $y \in \{0, 1\}$, the optimal performance of a learning-based method would be if we fulfill the following element-wise loss

$$l_{\text{classification}}(y, f(\cdot;\boldsymbol{\theta})) := \mathbb{1}(y, f(\cdot;\boldsymbol{\theta})) = \begin{cases} 0, & \text{if } y = f(\cdot;\boldsymbol{\theta}) \\ 1, & \text{otherwise} \end{cases} \tag{2.10}$$

Yet, this loss is not differentiable and therefore difficult to optimize. The *hinge loss*

$$l_{\text{hinge}}(y, f(\cdot;\boldsymbol{\theta})) := \max(0, 1 - yf(\cdot;\boldsymbol{\theta})), \tag{2.11}$$

a partial differentiable relaxation to equation 2.10, is a common proxy objective, used for SVMs (Scholkopf et al. 2001) and for deep neural networks (Tang 2013a). If the dataset $\mathcal{D}$ is separable and the underlying model parameterization results in zero *hinge loss*, the original classification objective (Equation 2.10) will also be fulfilled. Based on the same assumptions, the same holds true for the cross entropy objective (C. Bishop 2007)

$$l_{\text{crossentropy}}(y, f(\cdot;\boldsymbol{\theta})) := y\log(f(\cdot;\boldsymbol{\theta})) + (1-y)\log(1 - f(\cdot;\boldsymbol{\theta})). \tag{2.12}$$

In the case of a regression problem (Chapter 2.2.1.5), the mean squared error

$$l_{\text{mse}}(y, f(\cdot;\boldsymbol{\theta})) := ||y - f(\cdot;\boldsymbol{\theta})||^2 \tag{2.13}$$

represents the gold standard objective, albeit not suitable for datasets with outliers. The Huber loss

$$l_{\text{huber}}(y, f(\cdot;\boldsymbol{\theta})) := \begin{cases} \frac{1}{2}(y - f(\cdot;\boldsymbol{\theta}))^2, & \text{for } |y - f(\cdot;\boldsymbol{\theta})| < \delta \\ \delta|y - f(\cdot;\boldsymbol{\theta})| - \frac{1}{2}\delta^2, & \text{otherwise} \end{cases} \tag{2.14}$$

is an extension to the mean squared error, specially designed to be robust against outliers.

Designing the right objective or loss is crucial for learning methods to converge to good and desired solutions. In robotic manipulation, the learning objective can be used to provide additional domain-specific information, implicitly guiding the learning algorithms parameter updates.

2.2.1.4 Optimizer

Optimizing the open model parameters $\boldsymbol{\theta}$ is a key component for the success of learning methods. Optimization methods can be categorized into gradient-free and gradient-based. Sampling is a common gradient-free approach whereas Newton-Raphson (Newton 1664; Raphson 1690) allows for fast convergence in the proximity of a minimum and availability of second-order information. Most state-of-the-art parametric approaches relevant to this thesis, such as deep neural networks, are highly-parametric and have well-defined gradients. Sampling-based optimization is very costly in this regime, and therefore gradient-based approaches represent the predominant optimization approach for large-scale machine learning methods (Bottou et al. 2018). Recent efforts in large-scale data collections render computing the gradient for equation 2.9 intractable. Fortunately, we can obtain a sufficient stochastic gradient approximation

$$\hat{\nabla}_{\boldsymbol{\theta}^t} = \frac{1}{\#\mathcal{D}^{\text{mb}}} \sum_{(x,y)\in\mathcal{D}^{\text{mb}}\subseteq\mathcal{D}} \nabla_{\boldsymbol{\theta}^t} l(y, f(x;\boldsymbol{\theta}^t)) \tag{2.15}$$

by sampling subsets of the dataset $\mathcal{D}$ uniformly at random. We iteratively optimize our objective by updating the parameters $\boldsymbol{\theta}$ using a first order optimizer h and the latest $\hat{\nabla}_{\boldsymbol{\theta}^t}$ or some additional statistics thereof.

$$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t + h_1(\hat{\nabla}_{\boldsymbol{\theta}^t};\psi) \tag{2.16}$$

Stochastic gradient descent (SGD)

$$h_{\text{SGD}}(\hat{\nabla}_{\boldsymbol{\theta}^t};\psi) := \psi\hat{\nabla}_{\boldsymbol{\theta}^t}, \tag{2.17}$$

the most basic algorithm to optimize parametric models for large datasets, scales the noisy gradient estimate $\hat{\nabla}_{\boldsymbol{\theta}^t}$ with a learning rate ψ. A more detailed discussion of theoretical properties and use cases for SGD can be found in (Bottou 2010). In the recent literature many more first order stochastic gradient descent based methods have been proposed (Kingma et al. 2014; Tieleman et al. 2012; Duchi et al. 2011; Zeiler 2012), all designed to better cope with the noisy approximation and differently scaled gradients.

Newton's method allows to speed up convergence significantly in the vicinity of a local minimum by capturing additional information about the loss landscape based on the Hessian, the second derivative with respect to the loss. The cubic computational and quadratic space complexity renders second-order approaches not applicable to large-scale state-of-the-art models. Noisy estimates of the Hessian pose additional challenges for second-order methods when applied in highly stochastic settings. Even for large-scale convex optimization problems, with a factorizing objective (Equation 2.2.1.3), stochastic first-order methods achieve faster convergence concerning the overall computation time (Bottou 2010).

2.2.1.5 Regression

Supervised learning tasks which map from a continuous input $\mathcal{X} := \mathbb{R}^n : n \in \mathbb{N}^+$ to a continuous target space $\mathcal{Y} := \mathbb{R}^n : n \in \mathbb{N}^+$ are commonly referred to as *regression* problems. Many robotic manipulation problems, for instance, predicting the expected sensory reading (Pastor et al. 2012), learning inverse dynamics models (C. G. Atkeson et al. 1997; Meier et al. 2014a), forward models (C. G. Atkeson et al. 1997) and grasp pose prediction from 2D images (Redmon et al. 2015) have been phrased as *regression* problems. These approaches generally assume that there exists a functional mapping (many-to-one or one-to-one) which learning methods such as Gaussian processes (Rasmussen et al. 2006), linear models or deep neural networks (I. Goodfellow et al. 2016) can infer from data. The mean squared error (MSE) (Equation 2.13) is the most commonly used objective for the latter to methods to optimize its parameters.

However, if the dataset is ambiguous, for example, two grasp poses are equally suitable for a given image of an object, we cannot learn a proper functional mapping from input features x to targets y. Hence, using MSE for regression inevitably results in significant errors, since an intermediate value and neither of the modes will be optimal. Therefore, such problems require to learn a multi-modal distribution which allows resolving the ambiguity by sampling. Fortunately, learning multi-modal distributions can be rephrased as a multi-class classification problem, discussed hereafter, by approximating the output space $\mathcal{Y}$ with a sufficiently large number of classes.

2.2.1.6 Classification

Supervised learning problems with categorical a output space $\mathcal{Y} := \{0,1\}^n : n \in \mathbb{N}^+$ are referred to as *binary classification* for $n = 1$ and *multi-class classification* for $n > 1$. This general class of learning problems is a well understood and has a well established theoretical analysis (Vapnik 1999). There exist several standard benchmarks of increasing complexity and dataset size allowing to compare the performance and develop new classification approaches. For example, UCI (Dheeru et al. 2017), MNIST (LeCun 1998) and the ImageNet Classification Challenge (Russakovsky et al. 2015) are well-established benchmarks, the latter two commonly used for computer vision.

Many robotic manipulation problems have been cast into this learning framework. Failure prediction due to unexpected sensor readings during real-world robotics experiments has been phrased as a binary classification problem-based on statistical tests (Pastor et al. 2011a). Grasp planning is another very active research field within the manipulation community. (Saxena et al. 2008a) proposed to formulate grasp planning as a probabilistic classification problem, using vision-based features. (Lenz et al. 2015) use an exhaustive sliding window classification approach to grasp *detection* to solve the structured prediction problem, discussed in more detail hereafter, in a brute force manner.

2.2.1.7 Structured Prediction

Structured prediction differs significantly to classification and regression due to a richer and more complex output space $\mathcal{Y}$. A typical example of more complex output spaces might be graph structures like parsing trees in natural language processing or pairwise or higher order terms between different elements of the output space. Such problems require different loss formulations

$$l(y, f(x,y;\boldsymbol{\theta})) := \max(0, \max_{y'}(\Delta(y,y') + f(x,y';\boldsymbol{\theta})) - f(x,y;\boldsymbol{\theta})) \tag{2.18}$$

and often different model formulations $f(x,y;\boldsymbol{\theta})$, mapping $\mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, to capture the additional structure in the output space. Inference in such structured model formulations is not a mere mapping but requires to solve an optimization objective as well (Tsochantaridis et al. 2005):

$$\hat{y} = \operatorname*{argmax}_{y} f(x,y;\boldsymbol{\theta})) \tag{2.19}$$

Hence, this is formulation is only applicable if Equation 2.19 can be optimized efficiently or the search space is sufficiently small to be enumerated.

Other structured problems, for instance, ranking bounding boxes, can be formulated in a different manner

$$l_{\text{ranking}}(y, f(x;\boldsymbol{\theta})) := \max(0, \max_{(x',y')\in\mathcal{D}}(\Delta(y,y') + f(x';\boldsymbol{\theta})) - f(x;\boldsymbol{\theta})) \tag{2.20}$$

introducing a ranking between different data samples, without requiring the samples from the output space as input to the model. In order to infer the best hypothesis from a set of hypotheses, the model only needs to be evaluated for every hypothesis once. (Lehmann et al. 2011a) proposed to use such a ranking formulation for efficient object detection within an image, significantly reducing the search space compared to exhaustive classification formulations.

Semantic and instance segmentation are common computer vision problems with direct applications in robotic manipulation. Here, the output space $\mathcal{Y}$ is represented by labels for all pixels. Hence, the assumption that all labels/pixels are independent given the input data does not hold. Conditional random fields, an extension of logistic regression (Chapter 2.2.2.3) to structured outputs, modeling pixel-wise relationships (Lafferty et al. 2001), better capture the output space structure, resulting in easier learning problems and better predictive performance. (N. D. Ratliff et al. 2007) have formulated learning from demonstration, a common approach to teach manipulation skills for robotic manipulation, as a max-margin structured prediction problem.

In summary, if there exists a known structure within the output space, it is often beneficial to include additional objective terms or constraints such that the underlying learning method is aware and can exploit this structure. Nevertheless, complex output space structures result in more complex learning objectives, e.g., rendering otherwise convex problems indefinite. Thus, global optimality cannot be ensured anymore.

2.2.2 Learning Methods

There exist a large body of work on different learning methods, exploiting different problem structures, assumptions, scale to large datasets, as well as providing computationally efficient uncertainty estimates. In the following, we discuss machine learning approaches which have been used throughout this thesis to extend *model-based* approaches.

2.2.2.1 Support Vector Machine

Linear classification boils down to finding a separating hyperplane. In the case of linearly separable data, with high probability, an infinite number of separating hyper-planes exist. For example, the perceptron algorithm (Rosenblatt 1958) is one particular method to determine a particular separating hyperplane. Max-margin formulations attempt to find the hyperplane with the maximal possible distance between the different classes. The primal optimization objective can be written as a convex Quadratic program (QP) (Cortes et al. 1995; Scholkopf et al. 2001):

$$\underset{w}{\operatorname{argmin}} ||w|| + C\sum_{i=1}^{n} \xi_i \tag{2.21}$$

$$s.t. \tag{2.22}$$

$$y(w^T x_i + b) \geq 1 - \xi_i, \forall i \tag{2.23}$$

$$\xi_i \geq 0, \forall i \tag{2.24}$$

with $\theta = [w, b]$ and $||w||$ regularization to resolve the redundancy by re-scaling b and w. Slack variables ξ_i are required if the dataset is not linearly separable. Linear separation in the original feature space $\mathcal{X}$ is often too restrictive. This issue is commonly addressed by introduciing a feature transformation function $\Phi(\cdot)$. Using a hinge-loss objective (Equation 2.11) we obtain the following max-margin classification objective

$$\underset{w}{\operatorname{argmin}} ||w|| + C\sum_{i=1}^{n} \max(0, 1 - y_i(w^T\Phi(x_i) + b)). \tag{2.25}$$

In its primal form, (stochastic) gradient descent can be used to solve this optimization problem, albeit second order methods are more efficient for feasible dataset sizes. The primal objective allows to further optimize the feature function $\Phi(\cdot)$ using backpropagation (Rumelhart et al. 1986), for example in form of deep neural networks. The dual problem formulation

$$\underset{\alpha}{\operatorname{argmin}} \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\alpha_j y_i y_j \Phi(x_i)^T\Phi(x_j) \tag{2.26}$$

$$s.t. \tag{2.27}$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \tag{2.28}$$

$$C \geq \alpha_i \geq 0 \forall i \tag{2.29}$$

provides an alternative perspective and extension for learning a linear separating hyperplane by applying a non-linear feature transform. The dual problem is again a convex Quadratic program with a unique global optimal solution for which efficient optimization approaches exist (Scholkopf et al. 2001). The separating hyper-plane is recovered by computing

$$w = \sum_{i=1}^{n} \alpha_i y_i \Phi(x_i) \tag{2.30}$$

$$b = y_i - w^T \Phi(x_i) : C > \alpha_i > 0. \tag{2.31}$$

The *support vectors* (all non zero α) allow to compute the hyperplane. Analog to Gaussian processes (Rasmussen et al. 2006), the input data is only used in an inner product $\Phi(x_i)^T \Phi(x_j)$ computation, lending itself to the kernel trick, replacing $\Phi(x_i)^T \Phi(x_j)$ with a positive semi-definite kernel function $k(x_i, x_j)$ (Scholkopf et al. 2001) e.g. the RBF kernel

$$k_{\text{rbf}}(x_1, x_2) = \exp\left(-\frac{(x_1 - x_2)^2}{l}\right). \tag{2.32}$$

Thus, a kernel-based SVM determines *support vectors*, representative example templates in the training dataset, based on the similarity metric defined by the kernel. In order to infer the class of a new data sample the distance to all representative example templates, weighted by their corresponding α_i has to be evaluated. The number of *support vectors* is negligible compared to the original dataset size resulting in a computationally efficient classification method. However, the quadratic scaling of the kernel matrix fundamentally limits optimizing the kernel-based SVM objective for large-scale datasets. In recent years large-scale datasets such as ImageNet (Deng et al. 2009), Pascal VOC (Everingham et al. 2010) and MSCOCO (Lin et al. 2014) render kernel-based methods computationally infeasible. Therefore, the primal objective with parametric and optimized feature transform, resulting in a non-convex optimization problem, has been revisited (Tang 2013b).

2.2.2.2 Naive Bayes

This generative naive classifier (Ng et al. 2002) is based on the assumption that all features are independent given the class:

$$p(y_k|x) = \frac{p(y_k) \prod_{i=1}^{n} p(x_i|y_k)}{p(x)} \tag{2.33}$$

Hence, the estimation of the likelihood model factorizes across features. The maximum likelihood solution can be found by a closed form solution, depending on the likelihood model, allowing for efficient and incremental learning by updating the sufficient statistics of the feature wise likelihood models. This simple model has been used with great success for complex tasks such as spam classification and decision making with medical data (Kazmierska et al. 2008) and sampling-based motion planning (Burns et al. 2007).

2.2.2.3 Logistic Regression

Despite the name “logistic regression” this method is in fact a discriminative binary classification approach.

$$p(y_k|x) = o(x)^{y_k} \cdot (1 - o(x))^{1-y_k} \tag{2.34}$$

$$o(x) = \frac{1}{1 + \exp(\boldsymbol{\theta}^T x)} \tag{2.35}$$

In order to optimize Equation 2.34, we minimize the sum of the negative log-likelihood over all samples, thus, compute the maximum likelihood estimate. Different from ordinary linear regression, the additional non-linear transformation (Equation 2.35) hinders a closed form solution. As a generalized linear model for classification, this objective is straightforward and fast to minimize using iterative gradient-based optimization. The resulting solution allows introspecting the decision boundary by analyzing the feature weighting. Further, the computational requirements for inference are minimal, only a single inner product between the features and the data sample and an additional exponential are required. Therefore, this method functions as a great baseline for many robotics tasks, such as affordance predictions from 3D point clouds (Kim et al. 2014) and grasp prediction (Saxena et al. 2008b). However, good manual feature engineering is essential for many tasks to achieve good performance with such a simple, learned mapping.

2.2.2.4 Deep Neural Networks

Neural networks have a long history in the field of machine learning (Schmidhuber 2015). On an abstract level deep learning and thus deep neural networks represent a learning framework composed of a directed computation graph for which most computational nodes can be derived with respect to their inputs and open parameters. The hierarchical composition of the computational graph structure lends itself to create complex parametric function approximators while maintaining computationally efficient inference and optimization. Due to computational advances, allowing to solve convex Quadratic programs efficiently, neural networks, known to be difficult to optimize, became *unfashionable* in the late 1990s, and early 2000s and the field of machine learning focused on SVMs and Gaussian processes. The breakthrough result by (Krizhevsky et al. 2012) has shown that DNNs make better use of the new available large-scale datasets (Deng et al. 2009), several orders of magnitude larger than before (LeCun 1998), and hardware advances supporting efficient parallel matrix operations. This promising performance improvement on a single classification task has spurred new interest within the research community in DNNs. In the following, we describe the main concepts required to develop deep learning models. We refer to (I. Goodfellow et al. 2016) for a more comprehensive literature review and discussion.

Computational Node: The core idea of deep learning is to create a graph of computational nodes to achieve minimal loss on an objective (Equation 2.9). A general node in this graph is a functional mapping

$$n(\cdot;\boldsymbol{\theta}) : \mathcal{X} \rightarrow \mathcal{Y} \tag{2.36}$$

where $\mathcal{X}$ represents the input feature space and $\mathcal{Y}$ the output space of this particular node. Most computational nodes allow to obtain gradients for their parameters $\boldsymbol{\theta}$ and for their input features in $\mathcal{X}$. Several computational nodes are commonly combined, building a *composed node* also referred to as *layer* to communicate ideas and allow for easier model design. The *fully connected* (FC) layer

$$n_{\text{fullyConnected}}(x; w, b) := wx + b \tag{2.37}$$

is the most fundamental building block for deep neural networks. It represents a linear projection with open parameters $\boldsymbol{\theta} = [w, b]$, using an efficient matrix multiplication and vector addition. Since composing linear functions results in a linear model, the overall model complexity, expressiveness does not increase. In fact, every composed, stacked hierarchical linear function can be expressed by a single linear function. Hence, linear FC layers are typically followed by a nonlinear node, also referred to *nonlinearity*:

$$n_{\text{sigmoid}}(x) := \frac{1}{1 + \exp(-x)} \tag{2.38}$$

$$n_{\text{relu}}(x) := \max(0, x) \tag{2.39}$$

$$n_{\text{prelu}}(x) := \begin{cases} x, & x \geq 0 \\ -\alpha x, & x < 0 \end{cases} \tag{2.40}$$

$$n_{\text{elu}}(x) := \begin{cases} x, & x \geq 0 \\ \alpha * (\exp(x) - 1.), & x < 0 \end{cases} \tag{2.41}$$

Nonlinearities are characterized by their computational complexity, does hardware acceleration for particular operations exist, and most importantly if they saturate or not. For instance, the *sigmoid* nonlinearity (Equation 2.38) (Rumelhart et al. 1986) saturates, since its value approaches 0 for $x \ll 0$ and 1 for $x \gg 0$. In both cases its gradient vanishes. This creates complications while optimizing very deep, thus hierarchical, neural networks, since gradients can easily vanish if the network is not initialized properly. Hence, in recent years several novel non saturating activation functions have been proposed. Most notably *relu* (Equation 2.39) (V. Nair et al. 2010), used in the landmark ImageNet result by (Krizhevsky et al. 2012). This nonlinearty is only subdifferentiable and can further result optimization challenges if many parameters project on $\mathbb{R}^-$, thus, not receiving a zero gradient. *Prelu* (Equation 2.40) (He et al. 2015) and *elu* (Equation 2.41) (Clevert et al. 2015)) mitigate this issue by modeling $x < 0$ differently while maintaining the efficient gradient computation.

For many feature spaces $\mathcal{X}$, more structural information exists, for example, spatial or time locality. FC layers do not use this information, even worse, if applied they destroy this data property. Therefore, *convolution* (Conv) layers represent another major deep learning building block. This layer is relevant for robotics since it is ideal for RGB-D images and time series data. A naive 1D discrete convolution consists of a *convolutional kernel* w of size k, being convolved over the input data x

$$n_{\text{convolution}}(x, w, b) := \sum_i b + \sum_{-k}^{k} w_k x_{i+k}. \tag{2.42}$$

Depending on the kernel size k, it is more efficient to compute *convolution* layers in frequency domain (Vasilache et al. 2014). Since the majority of research within the computer vision and robotics community focuses on models with small kernels, the time domain is more common. Convolutions extend to higher dimensional input data, relevant to robotics such as point clouds, but become increasingly more computationally expensive (Pavel et al. 2013). *Conv* layers greatly reduce the number of open model parameters, another important property, since the number of open parameters w depends on the kernel size k and not on the input feature dimension. Further, *Conv* layers are inherently translation invariant, therefore, ideal for learning hierarchical representations from spatially or temporally structured data.

A large amount of diverse, representative, i.i.d. data, required for optimizing deep neural network models is a common limitation for the direct application in robotic tasks. Further, it is not common to directly integrate existing model knowledge into the computational graph. However, it is possible to directly embed known models into the network structure (Kloss et al. 2017) for as long as the gradients concerning the inputs and possibly its parameters exist, can be derived or approximated. An embedded model is merely another *computational node*, which the overall function approximator can exploit during training and inference. Note, with a sufficient amount of data the neural network might ultimately replace the initial model. Still bootstrapping a complex deep learning model by integrating simpler domain-specific models can be highly beneficial since the overall network only has to learn an error model.

In recent years, deep learning has become a common *learning-based* method for robotics. In part, this can be attributed to the availability of more mature frameworks such as TensorFlow (Abadi et al. 2016), PyTorch (Paszke et al. 2017) and MXNet (Chen et al. 2015). Recently first results on integrating non-differentiable models (Choromanski et al. 2017) into computational graphs with robot applications have been presented. Additionally, state-of-the-art performance in fields for which large-scale datasets can be easily obtained such as computer vision and natural language processing is only achieved by deep learning based approaches. Based on these empirical observations the robotics community has started to investigate large-scale data collections to enable such learning approaches (Chapter 5).

Structure: Deep learning methods can be categorized by the computational problems they address, their objective formulation, and their model design. The best-understood problem is supervised learning (Section 2.2.1), optimizing a fixed mapping from a feature space $\mathcal{X}$ to an output space $\mathcal{Y}$. Deep learning approaches tackling this problem class are commonly referred to as *feed forward* models. (Hornik et al. 1989) proved that this family of models is a universal function approximator, given a sufficient amount of open parameters, it is possible to approximate any function sufficiently well with enough data samples. The high-level requirements are at the neural network consists of least two FC layers with a sufficiently large number of parameters, the first one followed by a saturating nonlinearity, e.g., sigmoid (Equation 2.38). We can further categorize *feed forward* models by their most common computational nodes and graphs, the network structure. *Convolutional neural networks* (CNNs), composed out of Conv layers followed by FC layers, are the current best model structure for image and video-based problems. (Shelhamer et al. 2017) proposed to further relax the requirement for a fixed size feature space by using a *fully convolutional* approach. Here, a dense output map is generated, typically in a much lower resolution as the input data. *Fully convolutional* classification tasks use the maximum value of the last computational layer instead of the dense output. *Auto-encoder* networks (Hinton et al. 1994) are another interesting family of networks, used to learn lower dimensional embeddings. This class of structures has exciting new robotics applications since no labeled data is required, allowing to use data from all available robot sensor streams. However, learning low dimensional embeddings results often in an ill-posed regression problem due to multi-modal output distributions (Chapter 2.2.1.5). *Variational auto-encoder* (Kingma et al. 2013) alleviate this issue by transforming samples from a known latent distribution into the empirical data distribution.

Variable length feature and output spaces build another broad and very general problem class. For instance, translation models have to cope with varying sentence length as input and potentially different varying output length. Time series data such as sensor streams are ubiquitous in robotics and also a representative of this problem class. *Recurrent neural networks* (RNNs) are the predominant network topology for these problem instances. RNNs represent a universal Turing complete computation architecture (Siegelmann et al. 1995). This computational expressiveness results in a challenging learning problem. The computational graph is rolled out for every given input sequence. Thus, the resulting network structure can be interpreted as a dynamically created feedforward network per example. Since the dynamically created feedforward network repeatedly applies the same computational node, gradients are very likely to either vanish (converge towards zero) or explode (converge towards infinity). (Hochreiter et al. 1997) identified this problem and proposed a novel structure, long short-term memory (LSTMs) cells, to improve the gradient flow through time. (Chung et al. 2014) presented a more straightforward but effective Gated RNN unit (GRU) which has found great adaptation in the recent literature. Despite these improvements, RNNs are still notoriously difficult to optimize. Therefore, attention-based models build a promising new direction for limited size time series data as shown by (Vaswani et al. 2017). In addition to the better-understood optimization problem formulation, attention-based models are more suitable for the recent advances in parallel computation, resulting in faster training and inference time.

Generative adversarial networks (GANs) (I. J. Goodfellow et al. 2014) are a novel concept, deep neural network class, with broad applications in robotics, most commonly used for unsupervised learning problems. The main idea of this framework is to use two networks, the *generator* which attempts to synthesize data to fool the second network, the *discriminator* which tries to differentiate synthesized from real data. Hence, this formulation does not require to specify the desired target objective explicitly. For instance, in the case of learning how to generate realistic robotic sensor measurements, it is not trivial to define the required noise characteristics. GANs mitigate this problem by learning a *discriminator* which penalizes examples which are seemingly different from the empirical data distribution. There exist further relevant relationships between GANs and robotic learning problems, for example, inverse reinforcement learning (Finn et al. 2016a) and learning from demonstration. Unfortunately, GANs are very difficult to optimize and parameterize (Lucic et al. 2017), hence, new optimization methods and tools have to be developed to make this framework easier accessible to robotic manipulation research.

Chapter 3

Real-time Perception meets Reactive Motion Generation

Robotic manipulation is a challenging multidisciplinary field with a rich history of *model-based approaches* (Chapter 2.1). To better understand current limitations of fully integrated *model-based* manipulation systems we analyze a state-of-the-art fast feedback *model-based* system in this chapter. Understanding how all *model-based* components interact enables us to recognize possibilities for *learning-based* extensions.

Already in the 80's (R. A. Brooks 1990), it has been postulated that tightly integrating real-time perception and reactive motion generation is beneficial if not even required for robotic systems that physically interact with uncertain and dynamic environments. Yet, Sense-Plan-Act (Chapter 2.1.1.1) is still the predominant system architecture to date within the robotics manipulation community, used for the recent challenges (*DARPA Robotics Challenge* (DRC) (C. Atkeson et al. 2015; Johnson et al. 2015) or *Amazon Picking Challenge* (APC) (Eppner et al. 2016; Correll et al. 2016)). In this chapter, we present an instantiation of a robotic system which tightly integrates *model-based* real-time[1] perception with reactive motion generation for autonomous manipulation. The resulting *model-based* system architecture provides insights into the achievable performance for dynamic manipulation tasks. It further allows us to identify key assumptions required to achieve such high performance using purely *model-based* methods. Different from *end-to-end learning-based* systems, the presented work enables introspection. It enables us to reason about possible *learning-based* extensions to its components.

Therefore, we aim at drawing conclusions on *model-based* system integration based on empirical evidence from an extensive number of experiments. We quantify the benefit of integrating real-time perceptual feedback and reactive motion generation in dynamic manipulation scenarios for high DoF systems to achieve high performance. We compare to baseline systems that rely on

[1] As common in the Computer Vision area, we use the term *real-time* to indicate that the computation time required by the perception methods are below the frame-rate of the depth camera, i.e., below 30Hz.

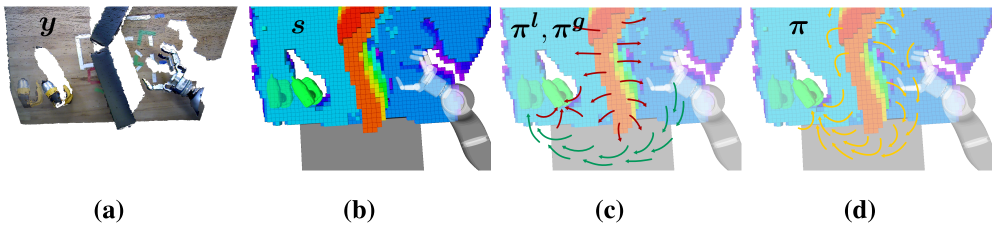


Figure 3.1: One time step to illustrate the different *model-based* components of our reactive system. Left to right: 3.1a sensory input y (we overlay the position of target object [drill] at an earlier time step), 3.1b the perceived state s of the robot and the environment (target object, obstacles) inferred from perception modules, 3.1c local and global policies π^l, π^g, and 3.1d fused policy π.

the same components but process sensory information at different rates or optimize the motion over different time horizons. We propose requirements for components rather than prescribing specific perceptual, planning or control modules. This quantitative evaluation of the level of integration, possible due to the well-defined *model-based* components, is one aspect that makes this work unique in relation to related work on robotic manipulation systems.

Our system is based on assumptions about the manipulation task and prior *model* knowledge. We use *model-based* visual perception to simultaneously track a known target object and the robot arm, and to obtain a geometrical representation of the workspace obstacles (Figure 3.1b). A local controller at a high rate (1 KHz) and a continuous motion optimizer (Figure 3.1c) at a lower rate (5-10 Hz), consume both the object pose, workspace geometry, and robot configuration, resulting in a joined kinematic policy[2] (Figure 3.1d).

Another unique aspect is the complexity of our experimental scenarios: they contain dynamic target objects and a dynamic environment - conditions that are common in human-robot collaboration, household, or disaster relief scenarios. Such scenarios pose difficult challenges for *end-to-end learning-based* systems. Acquiring a sufficiently large corpus of successful task executions is rather challenging for dynamic scenarios without bootstrapping from a *model-based* system. Further, specifying the task, automatically identifying, tracking and extracting key objects in the scene, exploring the "right" action space while being safe, and guaranteeing predictable and generalizable behavior, are just a few of the many additional challenges pure *learning-based* approaches face in such scenarios. Compared to the robot challenges mentioned above (APC and DRC), our experimental scenarios consider a much smaller variety of manipulation tasks and are tested on a fixed-base platform. However, concerning dynamicity, our scenarios go beyond those considered in the challenges where the environment is static and only the robot interacts with it. Furthermore, we do not use any teleoperation as was the case in the

[2]Throughout this chapter we will use *policy* and *controller* interchangeably.

DRC. We extensively evaluate the adaptivity, accuracy, and robustness of the alternative systems in four different scenarios with varying degrees of geometric complexity and dynamicity. We draw the following conclusions: (i) Incorporating real-time feedback on different time-scales is crucial to achieving safe and successful task execution in uncertain, dynamically changing environments, (ii) the availability of reactive motion generation relaxes the requirement for perception systems to achieve maximum, one-shot accuracy since new, updated information can be consumed immediately, (iii) fast feedback *model-based* systems in combination with slower *model-based* feedforward methods, enabling look ahead planning, exhibit great manipulation performance based on the strong assumption about the composition of the world state and limited autonomy, and (iv) data association, inferring grasps and changes to the system dynamics while grasping, lend themselves to *learning-based* extensions.

3.1 System Architectures

We evaluate and compare three alternative *model-based* system architectures along the spectrum from motion planning (Eppner et al. 2016) to pure feedback control: (i) Sense-Plan-Act, (ii) Locally Reactive Control and (iii) Reactive Planning. All discussed architectures belong to the general class of top-down hierarchical approaches composed of functional modules as discussed in Chapter 2.1.1. We are aware that our architecture notations still contain some ambiguity. In the absence of existing terminology, we define what we mean by them in each subsection and use them coherently after that. Figure 3.1 and Figure 3.2 present an overview of the information flow between the perception and the motion generation modules in the different architectures. We discuss them here in relation to existing work on robotic systems. Chapter 2.1 presents a more extensive literature review on robotic manipulation. A review of the vast body of work on visual tracking is the scope of this thesis and we refer to the respective sections in (Wüthrich et al. 2013; Garcia Cifuentes et al. 2017).

3.1.1 Sense-Plan-Act

Building systems through strong modularization into *sensing*, *planning* and *acting* components remains the predominant paradigm for high DoF robotic system building (Kortenkamp et al. 2016). In this paradigm, perception provides a model of the environment, in which a motion planner finds an optimal, collision-free path that is then tracked by a stiff and accurate controller. In Figure 3.2, this architecture corresponds to perception and motion optimization components, with visual feedback being considered only at the beginning of the motion planning task.

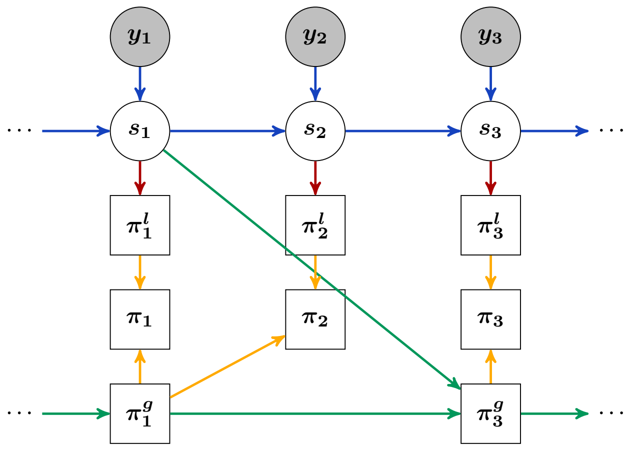


Figure 3.2: Flow of information across three time steps: The perception modules continuously infer the *latent* state of the robot and the world $\boldsymbol{s}$ from *observed* sensory input $\boldsymbol{y}$. The locally reactive control immediately translates this world state into a local policy $\boldsymbol{\pi}^l$. The continuous motion optimization computes a plan $\boldsymbol{\pi}^g$ for some time-horizon at a slightly lower rate. Reactive planning combines these two policies into one policy $\boldsymbol{\pi}$, which enables it to immediately react to local changes, and to look ahead in time to react to larger changes. Finally, this policy $\boldsymbol{\pi}$ produces a control output $\boldsymbol{u}$ (omitted for readability) which is sent to the robot.

The advantage of this paradigm lies in the subdivision of the complex problem of robotic manipulation into intuitive subproblems that are easier to solve. Systems that are built according to *sense-plan-act* (SPA) are perfectly suited for environments that are well-defined, structured and controlled. Chapter 2.1.1.1 provides a more in-depth discussion of this architecture, its shortcomings and common extensions.

3.1.2 Locally Reactive Control

On the other end of the spectrum from feedback control to motion planning, we consider system architectures that rely entirely on visual feedback control. They do not have the global motion optimization modules (Figure 3.2), but purely rely on local policies (Figure 3.2) to generate reactive motion behavior. With *local*, we indicate that they only take the local geometry of the environment around the current manipulator pose into account to compute the optimal, immediately next control command. System architectures in this category react to changes immediately and are very robust to uncertainties in sensing and actuation. However, they may get stuck in local minima, for example when the environment has a complex workspace geometry (Brock et al. 2008).

Pure feedback control systems have a long tradition. For example, (Khatib 1986) propose a well-defined interface for perceptual feedback in the form of potential fields constructed from closest points. The resulting feedback control law can be computed efficiently using the superposition property of potential fields in combination with additive control laws based on the desired motion and constraints such as joint limits.

Visual servoing (Chaumette et al. 2016) broadly refers to the class of locally reactive control that closes the loop around visual data. More recently, there has been much work on *end-to-end learning-based* approaches, neglecting almost all existent *model-based* information, directly inferring motion policies from perceptual feedback in the form of raw camera images and the system joint state, e.g., (Levine et al. 2016a). Another example comes from the team who won the first APC (Eppner et al. 2016). Although they do not close the loop around visual data, they demonstrate the robustness of standard joint space and operational space controllers. (Eppner et al. 2016) admit that locally reactive control approach may have limitations in more complex manipulation tasks that require planning.

3.1.3 Reactive Planning

Summarizing the above, locally reactive control gives robotic systems the ability to consume new information immediately, instantly react to changes and compensate for inaccuracies. However, it is susceptible to local minima. Motion planning as typically used in sense-plan-act finds solutions even in complex situations where feedback controllers may get stuck. Such methods use *kinematic models* and geometric environment models to plan for a longer horizon. However, this comes at a significant computational cost that may break real-time requirements.

A hybrid system that combines both reactive motion planning and locally reactive control would be ideal. Such a system can simultaneously adapt locally but also re-plan in case of larger changes. Compared to SPA, these systems are much more reactive and faster in completing the manipulation tasks. They rely on two motion generation modules, as depicted in Figure 3.2: the global motion optimization and local policies. The motion representations of both modules need to be fused to generate one policy for motion generation.

Combining local control with motion planning is quite common in the area of mobile robots, e.g., (Ferguson et al. 2006; Kuwata et al. 2009; Leonard et al. 2008; Richter et al. 2016). However, fewer approaches exist to date that scale up such a hybrid system to robots with many degrees of freedom which are manipulating the world due to the increasing planning complexity.

One example is the elastic-strip framework (Brock et al. 2002) which combines local control with motion planning. The use of onboard vision sensing is suggested and demonstrated on a real platform but with a simulated, potentially changing world model. Controller funneling (Burridge et al. 1999) is another hybrid approach which takes perceptual information into account. This approach requires a-priori knowledge to design the state space partitioning controllers and their switching conditions. So far we only talked about *model-based* approaches for reactive planning, Dynamic Movement Primitives (DMPs) (Ijspeert et al. 2013) a *learning-based* method can be interpreted as a combination of local control and a locally generalizing trajectory generator. Feedback terms can be learned and incorporated (Rai et al. 2016) for an instant reaction. (Lehner et al. 2015) present a mobile manipulation system that locally adapts and augments global motion plans in response to changes in the environment as perceived by onboard sensors. (Yang et al. 2016) present a system to find a valid stance and collision-free reaching configurations in complex, dynamic environments for a full humanoid robot.

3.2 Real-Time Feedback Modules

We consider scenarios that require manipulation of dynamic objects in dynamic environments. Continuous feedback on the location of target objects and the workspace structure is of utmost importance for systems acting in these scenarios. An important requirement for the feedback components is, therefore, to deliver information as fast as possible. A low-dimensional representation of this information is beneficial to keep bandwidth requirements low.

In the following, we briefly describe the *model-based* methods integrated into our system. However, we want to emphasize that any other approach which fulfills the requirements could be used instead.

3.2.1 Visual Tracking of Target Objects

We use visual tracking to estimate the pose of every object the robot may want to manipulate (Chapter 2.1.2). We choose the probabilistic *model-based* method from previous work by (Wüthrich et al. 2013). This method assumes knowledge of the 3-dimensional shape of the objects of interest, represented as triangle meshes. It takes as input depth images and compresses it into *6 DoF object poses* at the frame rate of the onboard camera. The model, domain-specific knowledge, and its formulation make it very robust to occlusions of object parts which are common in the context of manipulation tasks. Within this work, we assume that a good initialization for the tracker can be provided. However, a *learning-based* object detector and pose estimator would alleviate this requirement, complement our system.

3.2.2 Visual Robot Tracking

Precise positioning of a robot arm with respect to the sensed environment and the target object is a crucial ability for manipulation systems. It is not always possible to use merely forward kinematics to determine the 6D end-effector pose as elucidated in Chapter 2.1.2. On real robotic platforms, kinematic models and measured joint angles are commonly inaccurate and therefore lead to erroneous predictions of end-effector pose relative to the camera.

To mitigate this problem, we continuously estimate the true robot arm configuration relative to the camera mounted on the robot's head. We choose the probabilistic, real-time method from previous work by (Garcia Cifuentes et al. 2017). It fuses depth images and measured joint angles to produce precise estimates of the *robot configuration* at 1kHz which is the rate of the joint encoders. Even under massive occlusion of the arm and swift motion, this method can correct errors due to biases in the joint sensors and too imprecise kinematics of the camera relative to the rest of the kinematic chain. Furthermore, it models the delay between the measurements from the joint sensors and the camera.

3.2.3 Modeling Unstructured Workspace Obstacles

The robot needs to be aware of the workspace geometry and the obstacles therein to generate collision-free motion. Using *model-based* methods in such an unstructured environment is not only an open research field but also not necessary for many manipulation tasks. For the scenarios addressed in this chapter, we assume that only the target object, the table, and the robot has to be modeled since every other obstacle observation has to be avoided by the robot. Commonly, occupancy grids are a common data structure to probabilistically encode unstructured environments at multiple scales by integrating depth measurements over time, e.g., using OctoMaps (Hornung et al. 2013) (see Figure 3.3).

These approaches come at a significant computational cost that does not allow for reactive behaviors. Hence, we represent obstacle regions in a discrete binary occupancy grid which is newly generated frame by frame. We update our occupancy grid by transforming each point cloud to world coordinates while removing the points outside the robot workspace. This transformed point cloud is then converted to an occupancy grid (with 2 cm resolution), which is filtered for statistical outliers using PCL (Rusu et al. 2011). Finally, the voxels corresponding to the robot arms and the tracked object are set to empty (see Figure 3.1) and the occluded regions are set occupied using ray casting (Hornung et al. 2013). These filtering and transformation steps can be performed at approximately 15Hz. We found this rate to be sufficient in practice.

3.3 From Visual Feedback to Continuous Signed Distance Fields

We convert the processed perceptual data as described in the previous sections into a set of *Signed Distance Fields* (SDFs) describing the target object, table, unstructured workspace obstacles, and the robot. SDFs provide a way to perform fast collision checking, used by many motion optimization approaches (Zucker et al. 2013; N. Ratliff et al. 2015; Kalakrishnan et al. 2011b). SDFs also allow defining proper Riemannian metrics to measure path length in workspaces populated by obstacles (N. Ratliff et al. 2015), which we make use of in sense-plan-act and reactive planning.

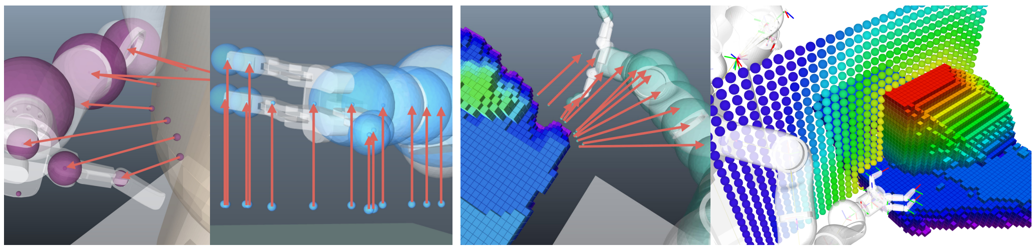
To compute the SDFs, we rely on a combination of analytic formulations and distance propagation algorithms. The manipulated objects, the table, and the robot links are *modeled* by simple geometrical shapes such as spheres, boxes and capsules (see Figure 3.3a and 3.3b). These shape primitives allow for analytic SDF formulations. For the occupancy grid, the SDF is computed using a voxel grid distance propagation algorithm and we use local interpolation techniques (Mainprice et al. 2016) to define gradients over this field (see Figure 3.3a).

To obtain a consistent representation, all SDFs are transformed into world coordinates. The target object SDF is positioned according to the object's visually estimated 6DoF pose. The SDFs of the robot links are positioned according to the forward kinematics and joint angles as estimated by the sensor fusion approach as mentioned earlier. For the table, we assume a known pose.

The combined SDFs are used by sense-plan-act, reactive planning and locally reactive control to produce collision-free movement as will be described in the next section. However, the SDF of the occupancy grid would require a prohibitively high bandwidth if sent directly to locally reactive control. Hence, we further approximate the workspace geometry with the *closest point* to each sphere that approximates a robot link (see Figure 3.3c). The positions of these points are obtained by computing the gradient of the SDF at the center of the sphere (see Figure 3.3c).

3.4 Motion Generation

The *model-based* motion generation module proposed in (N. D. Ratliff et al. 2018) is designed for continuous feedback integration. The architecture enables blending local reactive control and higher-level continuous motion optimization. More specifically, we use reactive control policies of the form $\pi_A = (f, \boldsymbol{A})$ that are defined as a non-linear second-order differential equation of the form $\ddot{\boldsymbol{x}} = f(\boldsymbol{x}, \dot{\boldsymbol{x}})$ with $\boldsymbol{x}, \dot{\boldsymbol{x}}, \ddot{\boldsymbol{x}} \in \mathbb{R}^k$ and a positive semi-definite weighting matrix that may vary smoothly across the space $\boldsymbol{A} : \mathbb{R}^k \times \mathbb{R}^k \to \mathbb{R}^{k \times k}$. Example weighting matrices are discussed in



(a) **(b)** **(c)** **(d)**

Figure 3.3: Local avoidance controllers to robot (3.3a), table (3.3b) and ambient world (3.3c). (3.3d) the occupancy grid (cubes: colors represent height), along with a vertical slice of the corresponding signed distance field (spheres: color represent the distance to the closest occupied cell).

the following sections and further properties and derivations are presented in (N. D. Ratliff et al. 2018).

Suppose we have a number of differentiable maps $\boldsymbol{x}_i = \phi_i(\boldsymbol{q})$, mapping from configuration space, assuming knowledge about the *kinematics model*, to task spaces $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n$ (Toussaint et al. 2010). Then given a collection of local controllers $(f_i, \boldsymbol{A}_i)$ defined on those task spaces, we define their pointwise evaluation at joint angles $\boldsymbol{q}$ as

$$\ddot{\boldsymbol{q}}^{\text{desired}} = \arg\min_{\ddot{\boldsymbol{q}}} \frac{1}{2}\sum_{i=1}^{n} ||\ddot{\boldsymbol{x}}_i^{\text{desired}} - \boldsymbol{J}_i\ddot{\boldsymbol{q}}||^2_{\boldsymbol{A}_i} \tag{3.1}$$

where $\ddot{\boldsymbol{x}}_i^d = f_i(\boldsymbol{q}, \dot{\boldsymbol{q}})$ and $\boldsymbol{J}_i$ is the Jacobian of ϕ_i. Example differentiable maps are forward kinematics maps to points on the robot's body and maps to relative distances between the robot and workspace objects. The *kinematics model* of the robot is easy to obtain i.e. from computer aided design (CAD) models used to construct the robot, allows for efficient differentiation and evaluation, and generalizes to the full state space of the robot. To simplify construction of the quadratic program (QP) in Equation 3.1, we use a common Gauss-Newton-like approximation, removing the term $\dot{\boldsymbol{J}}\dot{\boldsymbol{q}}$ from $\ddot{\boldsymbol{x}} = \boldsymbol{J}\ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}$. This term captures the information about the curvature of the kinematics function which has no perceptible effect when optimizing in 1 kHz.

This formulation allows for expressing both attractor/collision avoidance controllers, based on proximity to the goal/obstacles (Chapter 3.4.1), and complex anticipatory ones, obtained from motion policies optimized over a longer time horizon (Chapter 3.4.2). Providing rich *model-based* goals and obstacle representations enable more anticipatory and precise planning. Yet, with fast optimization and feedback loops as presented in this chapter, primitives are sufficient. Modularity is the key enabling us to experiment with the full spectrum from locally reactive control to higher level motion optimization for longer horizon reasoning.

3.4.1 Locally Reactive Control

Locally Reactive Control combines multiple controllers through Equation 3.1, including collision controllers to instantaneously react to the local workspace geometry, and target controllers for goal convergence. The target controller pulls the system toward position and orientation targets in a purely local Cartesian control fashion. This portion of the system can, therefore, be used by itself, without any higher level planning. It is visualized in Figure 3.1 with red arrows. In addition to the target controller, we use a collection of obstacle avoidance controllers that take effect when parts of the body get close to an obstacle. They create workspace accelerations away from the obstacle with increasing priority as a function of proximity. They are visualized in Figure 3.3. We also use a default posture potential that pulls the arms slightly toward a default posture to resolve redundancy along with simple damping controllers in the c-space and at the end-effector to regulate the velocity of the system. Designing and changing this default posture potential, e.g., for different robots is only possible since our approach is *model-based*. Local control runs at 1kHz for effective integration of the underlying highly nonlinear differential equation. However, it sends joint positions, velocities, and accelerations for low-level execution only at 100Hz. The lowest level of control handles the generation of the torques needed to track the desired joint states through interpolation and an inverse dynamics controller.

3.4.2 Reactive Planning and Continuous Motion Optimization

We use a motion optimizer based on Riemannian Motion Optimization (RieMO) (N. Ratliff et al. 2015) as the basis for reactive planning. It runs continuously, tracking the local minimum based on feedback while obstacles and the target change over time. This motion optimizer integrates information over a time horizon of three seconds (the "approximate" average time length of a reaching motion) based on the underlying kinematics *model*, enabling anticipatory behaviors and efficient coordination of multiple controllers handling collision avoidance, position targets, orientation targets, etc. As is sometimes done in optimal control and MPC (Chapter 2.1.4), it summarizes its policies as Linear Quadratic Regulators (LQRs) built on a local quadratic approximation around the local optimum. However, this is done only kinematically (with accelerations as control variables) since the planning module addresses movement only. This is visualized in Figure 3.1 with green arrows. These are sent to locally reactive control for integration with the other controllers through Equation 3.1. The continuous optimizer operates at a slower time-scale than locally reactive control, updating its optimization at 5-10Hz. To mitigate potential delays, it sends full LQR policies that represent the optimal policy within a region of the locally optimal trajectory. Rather than using a simple attractive potential pulling the end-effector toward a desired pose (which can be expressed as a differential equation in the configuration space) the motion optimizer creates a more expressive attractor differential equation that simultaneously pulls the system toward the desired target while also integrating anticipatory actions (e.g. rotating the wrist to avoid future obstacle) that enable smoother, more efficient, and well-coordinated behavior. Therefore, the local controllers are able to operate cohesively with the planned policies between planning updates.

3.4.3 Grasping

We manually decompose the grasping problem into multiple sequential task states—approach, establish grasp, move object, release, and retract—each governed by either local control, continuous optimization, or some combination thereof. Grasping is controlled independently of the other motion generation modules through force feedback in the fingers. The rest of the system observes the resulting movement of the hand and reacts accordingly to simultaneously adjust the arm to avoid obstacles and stabilize the hand posture to the extent possible under the constraints of the environment. We manually defined a set of grasp poses for each object that we use in our experiments. Annotating all objects with a fixed set of possible grasp hypotheses and a heuristic to decide which one to pick at runtime is not sufficient for true autonomy. *Model-based* grasping has very high computational demands and strongly depends on the perceptional system components. Hence, we identify grasping as a key challenge towards *autonomous manipulation* for which *learning-based* approaches can complement the presented system (see Chapter 5). Nevertheless, restricting the experiments to manually defined grasps enables us to perform consistent experiments across different scenarios and system instantiations for conducting empirical studies discussed hereafter.

3.5 Experiments and Demonstrations

We compare the performance of the three different system architectures that were explained in more detail in Chapter 3.1: (i) sense-plan-act, (ii) locally reactive control and (iii) reactive planning, in different scenarios ranging from static to fully dynamic. As an experimental platform, we use a fixed-base, manipulation platform equipped with two 7-DoF Kuka LWR IV arms, three-fingered Barrett Hands and an RGB-D camera (Asus Xtion) mounted on an active humanoid head by Sarcos. All components are torque controlled using *model-based* inverse dynamics controllers and PID feedback controllers manually tuned to track the desired joint states even if the underlying system dynamics change. It runs at determinable worst-case execution times of 1ms and is executed on a PC running Xenomai, a real-time framework for Linux. We placed system components that frequently interact on the same computer (tracking and SDFs, locally reactive control and motion optimization) to meet the computational requirements and network bandwidth of the different system components.

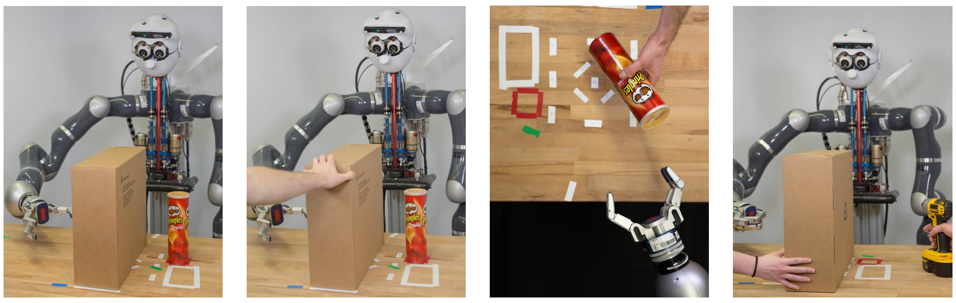
3.5.1 System Architecture Realizations

The visual perception modules (Chapter 3.2) consist of a tracker for the right robot arm and a tracker for the target object. We assume a known table pose. Everything else in the environment is considered to be unstructured workspace obstacles modeled by an occupancy grid map. Since our system is purely *model-based* it was essential but also straightforward to provide this information, whereas *learning-based* approaches would have to infer this information from observations. The algorithms used in all our implementations of the different architectures are identical. We vary the frequency at which information is passed to motion generation and whether we consider policies that are optimized over a longer time horizon.

Sense-Plan-Act Here, we acquire just one depth image at the very beginning of the experiment. Based on this image, the poses of the objects of interest are estimated, and a model of the workspace geometry is created. Then a one-shot motion optimizer, a simple variant of Chapter 3.4.2, generates a plan which will be executed without any further visual feedback. The overall planning time of sense-plan-act is limited to 2s for all experiments. This threshold has been chosen empirically, trading off convergences success and potential restarting of the planner in case no solution can be found.

Locally Reactive Control In this architecture, depth images are processed continuously to estimate the object pose and robot arm configuration. Additionally, the world model is updated online. This information is consumed by locally reactive control (see Chapter 3.4.1) that immediately adapts to the observed changes in the next control cycle.

Reactive Planning As in the previous architecture, the object and robot arm tracker continuously estimate the object pose and robot arm configuration. Also, the world model is updated online. However, here the information from the perception modules is also used to continuously replan in addition to the locally reactive control (see Chapter 3.4.2).



(a) Static pick and place **(b)** Dynamic pick and place **(c)** Dynamic grasping **(d)** Dynamic pointing

Figure 3.4: Experimental scenarios: The human hands indicate which objects are being moved during execution.

3.5.2 Scenarios

We present four different scenarios (see Figure 3.4) and experimental results which illustrate the importance of tightly integrating real-time perception and reactive motion generation. Each experiment parameterization is performed at least three times.

3.5.2.1 Pick and Place in Static Environments of Increasing Difficulty

In this experiment we consider the static pick and place scenario shown in Figure 3.4a. The task is to pick up the Pringles box and place it on the other side of the brown box, without any collisions. Figure 3.5 uses gray lines to visualize the varying positions of the box obstacle in this scenario. The box is always placed before starting each experiment. The closer the box to the robot base, the higher the difficulty to successfully pick and place the Pringles. For each system architecture and complexity level, we run three trials. At position 15, we reached the point where each system failed at least once. Table 3.1 shows the success rate of picking up the object and placing it at the target location. Even though the planning problem itself becomes very challenging, locally reactive control alone already performs very well. Not surprisingly, sense-plan-act performs very well in such a static environment. However, in the most challenging setting, it does fail more often compared to reactive planning. One reason for this is the limited planning time allocated, during which no successful plan may be found. Reactive planning can find a path more often since it can re-plan continuously, thus, has more time to find a feasible path during execution. In Table 3.2 we report the average execution time for successful trials in seconds. The time required for the initial object detection is not part of the execution. Locally reactive control and reactive planning are on par for the simple settings, whereas sense-plan-act is significantly slower. The difference in execution time is because sense-plan-act can only start

Difficulty	l. react. c.	react. pl.	s-p-a
static -10	100% (3)	100% (3)	100% (3)
static 0	100% (3)	100% (3)	100% (3)
static 5	67% (3)	100% (3)	100% (3)
static 10	33% (3)	100% (3)	67% (3)
static 15	17% (6)	50% (6)	17% (6)
dynamic	100% (3)	100% (3)	0% (3)

Table 3.1: Success rates (total runs) of pick-and-place experiment. Difficulty refers to obstacle position (cm) relative to the robot (see Figure 3.5).

Difficulty	l. react. c.	react. pl.	s-p-a
static -10	13.64 s	13.41 s	21.05 s
static 0	14.29 s	13.44 s	21.76 s
static 5	15.83 s	14.26 s	20.16 s
static 10	20.95 s	18.32 s	21.51 s
static 15	28.75 s	21.06 s	18.27 s
dynamic	17.95 s	15.44 s	-

Table 3.2: Average execution time (seconds) of successful pick-and-place. Difficulty refers to obstacle position (cm) relative to the robot (see Figure 3.5).

planning to the next pose after it achieved the old pose. It then has to wait until a solution has been found. The execution time increases with difficulty due to the more confined workspace which results in slower convergence especially for locally reactive control and in general for longer trajectories. This scenario illustrates that even in static environments, the two tightly integrated system architectures can have benefits over sense-plan-act.

3.5.2.2 Pick and Place in Dynamic Environments

In this scenario, the target object is placed in the same location as in the previous scenario, but initially, there is no obstacle. During execution, we move an unmodeled obstacle between the pickup and goal position (see Figure 3.4b). The obstacle is captured by the octomap and thus is considered by both locally reactive control and reactive planning.

For the three different system architectures, we observe very different outcomes. Success rates and execution times are reported in the bottom row (dynamic) in Table 3.1 and 3.2. While locally reactive control and reactive planning are on par regarding these measures, sense-plan-act fails completely - as expected. Results in terms of clearance between the robot and the dynamic obstacle are summarized in Figure 3.6 for the nine experimental runs (three per architecture). In the case of sense-plan-act the motion to perform the pick keeps high clearance with the obstacle.

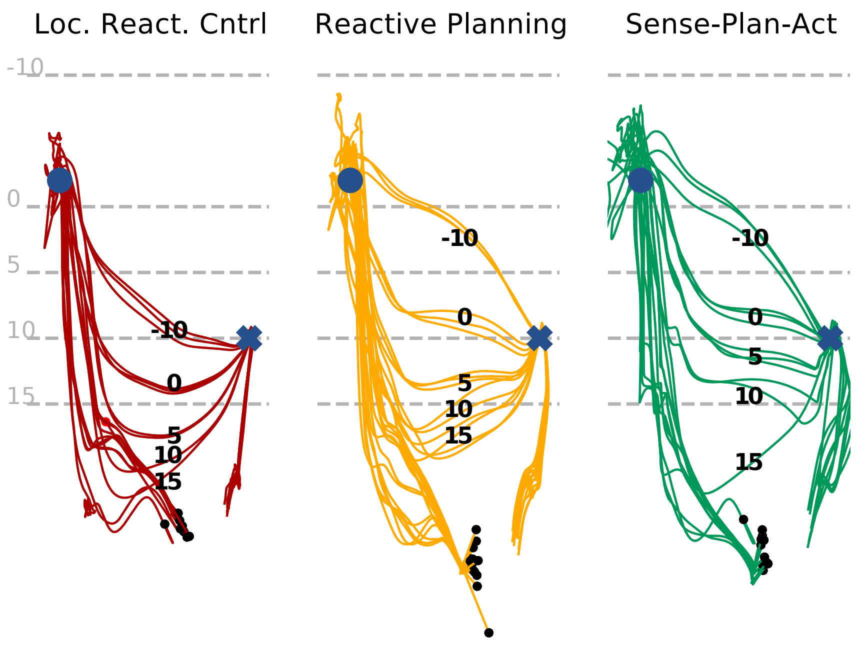
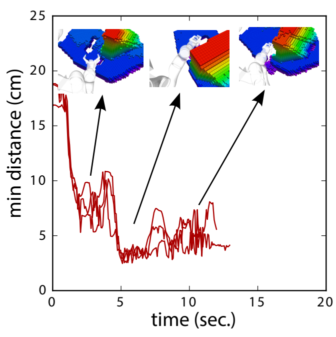
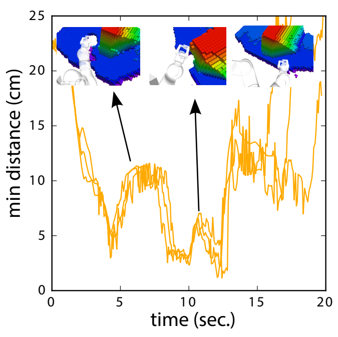


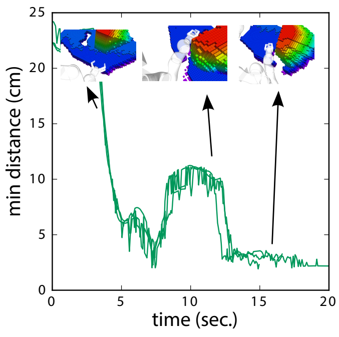
Figure 3.5: Visualization of the successful end-effector trajectories in a pick-and-place task in the presence of a box obstacle (Cf. Table 3.1). Per experiment, the box varies its pose from -10, 0, 5, 10 or 15cm distance to the far edge of the table (as indicated by the dashed lines). The black labels indicate which trajectory belongs to which box position. The black dots indicate the start of each trajectory. The blue dot indicates the picking object position and the cross its placing position. We compare locally reactive control (Left), reactive planning (Middle) and sense-plan-act (Right).



(a) Locally Reactive Control



(b) Reactive Planning



(c) Sense-Plan-Act

Figure 3.6: Minimum distances (i.e., clearance) with the introduced obstacle during the dynamic pick-and-place experiment. Locally reactive control gets stuck after successfully grasping the object. Reactive planning while slower than the locally reactive control to realize the grasping motion is able to perform the place successfully. Sense-plan-act despite higher initial clearance than locally reactive control collides with the obstacle when its position changes.

However, since sense-plan-act does not react to changes in the environment during motion, the arm collides with the unmodeled obstacle when its position changes. Locally reactive control can move faster towards the pick configuration, but this results in lower clearance to the obstacle. Locally reactive control is then unable to perform the place motion and gets stuck in a local minimum. Only when removing the obstacle, it nicely converges to the goal position. Hence, in this case, locally reactive control results in much safer behaviors than sense-plan-act. Finally, reactive planning is able to avoid the introduced obstacle. Additionally, it successfully finds a path to circumvent the obstacle and to place the object at the goal position.

In conclusion, these experiments show the importance of real-time perception to avoid collisions with unmodeled obstacles. In addition, this experiment illustrates that having reactive planning is essential to find collision-free paths in complex dynamically changing environments whereas locally reactive control is safe but gets stuck more easily.

3.5.2.3 Grasping with Dynamic Targets

Not only the unmodeled environment is subject to constant change, but very often also target objects may move when reaching for them. In this scenario, we systematically analyze the importance of perceptual feedback integration into motion generation, when the target is repositioned after motion onset.

This scenario has two levels of difficulty. In both, we place the target object in the center of the grid marked with white tape in Figure 3.4c. Level 1: As soon as the gripper starts moving, we move the target object to another grid cell. Level 2: We move and flip the target object 180 degrees. For each level, this repositioning is done very fast. For each level, we report execution time and grasp success.

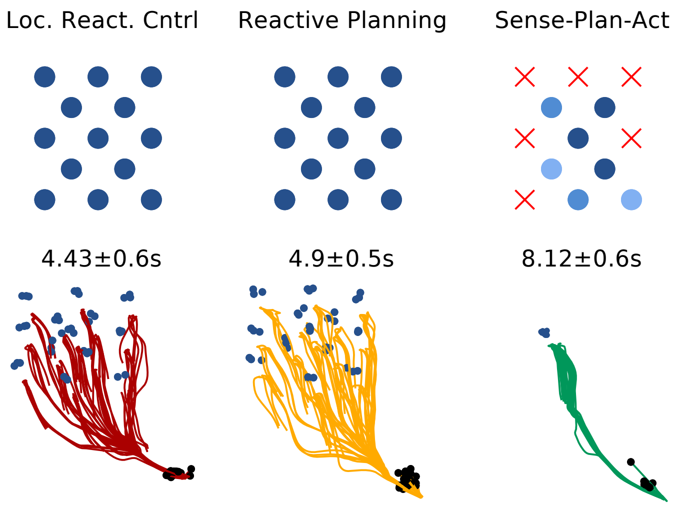
For Level 1, Figure 3.7a illustrates that both locally reactive control and reactive planning are capable of adapting to the fast changing conditions and successfully grasp the object for all possible changes. Notice, locally reactive control is slightly faster in execution which can be explained by the easy path planning problem required to solve this task. Sense-plan-act however, is only capable of grasping the object if it is close to the initial position. For the more complex Level 2 shown in Figure 3.7b, we observe very similar performance in terms of grasp success. Note, sense-plan-act did not adapt to the change of orientation; the grasp just happened to work due to the symmetry of the target object. Concerning execution speed, we observe a big difference between reactive planning and locally reactive control. The main reason for this difference is that reactive planning continuously adapts to the rotated approach whereas locally reactive control has an inherent tradeoff between convergence in position and orientation. For instance, in the case of higher priority on the position, locally reactive control will attempt to reduce the position error faster compared to the orientation error. This can result in overall slower convergence since orientation changes might be more difficult after reaching the target position due to environmental constraints. Arguably this tradeoff can be tuned to fit this scenario, but it always will be task specific whereas reactive planning automatically solves this problem due to the planning horizon. Instead of manually tuning, *learning-based* methods could identify situations and infer the proper tuning in order to resolve this issue.

This experiment emphasizes the importance of continuous feedback of target object pose to adapt the grasping motion in case the object moves during reaching. It also demonstrates the benefit of a longer planning horizon.

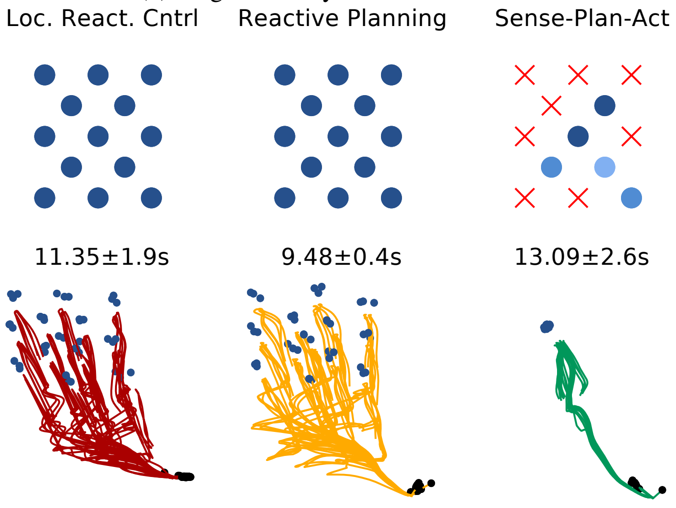
3.5.2.4 Pointing in Dynamic Environments with Dynamic Target

In our most complex scenario, we want to analyze the accuracy and reactivity of simultaneous changes in the environment and target pose. The task of the robot is to align its fingertip with the tip of a drill (see Figure 3.4d).

We have four different levels of complexity for this task. *Level 1*: As a baseline, we start with a static environment without obstacles while the drill is stationary. *Level 2*: we introduce a blocking obstacle (box) during execution, which can be avoided by going around it. *Level 3*: the obstacle is moved into the way such that the arm has to move over the obstacle or take a big detour. *Level 4*: We start with a blocking obstacle. After the system starts moving, we remove



(a) Target with Dynamic Position



(b) Target with Dynamic Pose

Figure 3.7: Visualization of successful grasps of in case of a (3.7a) repositioned target and (3.7b) repositioned and flipped target after motion onset. We compare locally reactive control (Left), reactive planning (Middle) and sense-plan-act (Right). (Top of 3.7a and 3.7b) The grid of possible target object poses after motion onset. The grid is $24 \times 24cm$ large. The central dot marks the initial target object position. Dark blue dots indicate that all three grasp attempts were successful. Red crosses indicate positions at which no grasp trial succeeded. (Bottom of 3.7a and 3.7b) Visualization of the end-effector trajectories in the corresponding color per approach for successful grasps. Black dots indicate the starting position. Blue dots indicate the target pose. For (3.7a) both, locally reactive control and reactive planning grasp successfully in the entire region of variation and successfully adapt the trajectory given the new feedback on target pose. Locally reactive control is on average slightly faster than reactive planning˙Sense-plan-act manages to successfully grasp the target when the new position is in/close to the path to the original target location. For (3.7b) both, locally reactive control and reactive planning grasp successfully in the entire region of variation and successfully adapt the trajectory given the new feedback on target pose. Reactive planning is on average a bit faster (time in seconds ± one standard deviation). Sense-plan-act manages to successfully grasp the target when the new position is in/close to the path to the original target location.

Difficulty	**l. react. c.**	**react. pl.**	**s-p-a**
static	100% (3)	100% (3)	100% (3)
straight	100% (3)	100% (3)	0% (3)
diagonal	100% (3)	100% (3)	0% (3)
turning	100% (3)	100% (3)	0% (3)

Table 3.3: Success rates (total runs) of pointing experiment

the obstacle while also changing the orientation of the drill by 90 degrees. The reorientation of the drill means that the pointing approach has to be adapted.

We report the results for this scenario in Table. 3.3. We define success as reaching the tip of the drill up to a distance of 3 cm without a collision with any environmental obstacle. Neither locally reactive control nor reactive planning collide with any obstacle in this experiment. Sense-plan-act however, collides with the blocking obstacle for both the *Level 2* and *Level 3* experiment. In the case of the *Level 4* experiment, sense-plan-act was able to reach the initial position of the drill tip. Since no perceptual feedback is considered it was not aware of the rotation whereas both locally reactive control and reactive planning could even shorten the path towards the drill tip by taking into account the removed obstacle. For optimal performance, *Level 4* requires both continuous tracking of the target (similar to *Grasping with Dynamic Targets*) and updates of the workspace obstacles (similar to *Pick and Place in Dynamic Environments*).

This experiment supports our hypothesis that integrating real-time perception with motion generation is key to task success and safe behavior in highly dynamic scenarios.

3.6 Summary and Review

In this chapter we presented and analyzed different fully integrated *model-based* system architectures. We showed that already locally reactive control which integrates perceptual feedback at the highest possible rate could be very efficient for simple known tasks while being safe by avoiding collisions with the environment. Reactive Planning achieves a better performance in more complex environments due to its ability to look ahead. Knowledge about the kinematic *models* is essential for the presented system to reason about the future system behavior and achieve its safe and reactive behavior. Sense-plan-act performs well in static scenarios as expected, but even there locally reactive control and reactive planning have advantages as they can start moving earlier while continuing to consume feedback.

We also observed that on the trade-off curve between perceptual accuracy and computational speed, it is more beneficial to have fast feedback than accurate world representations. This is especially the case for dynamic and uncertain manipulation scenarios where a fast reaction to sudden changes or new incoming information is critical. As communication bandwidth is limited, this also places constraints and how much information can be transferred between components. Therefore, we opted for *model-based* visual tracking and querying SDFs only for a small subset of points on the robot. Data association, identifying if observed points are part of the robot or the environment, was significant and we therefore carefully synchronized the different sensory and information streams across the three different computers. Despite synchronization, heuristics for data association are a current limitation. *Learning-based* approaches are an ideal candidate to replace these heuristics and infer whether or not to ignore voxels in the occupancy grid.

The presented system still solves relatively simple pick and place tasks, albeit going beyond state-of-the-art operating in dynamic environments. Constraining ourselves to simpler tasks allows to formulate robust observation models and well-defined cost functions capturing collision avoidance tasks. By choice, the presented system does not rely on any learning to tackle the different scenarios. However, based on the experience with this *model-based* system we can identify several avenues for *learning-based* extensions. The current architecture operates on visual and joint encoder feedback. However, manipulation tasks are heavily concerned with contact interaction. We use the finger strain gauges as feedback in the grasp controller. Our system would further benefit from also taking haptic feedback from tactile sensor arrays or force/torque sensors into account. *Modelling* these sensor modalities is challenging. Fusing multiple measurements from very diverse sensors and using them for execution feedback is a challenging problem. To the best of our knowledge, no complex fully *model-based* robotic manipulation system exists, capable of handling a diverse set of sensor modalities for contact interactions. In Chapter 4 we therefore present a novel *learning-based* approach to tackle this problem.

To empirically quantify the importance of fast feedback, we had to make strong assumptions about the manipulation task. For instance, only known objects with manually predefined grasps have been considered to allow for a fair comparison between the different methods. Albeit robots often have to interact with a small set of known objects, automatically inferring good grasp poses for unknown objects would be essential for the presented system to achieve a higher level of autonomy. *Model-based* methods for grasping are known to not translate well to the real world due to model inaccuracies and computation requirements (Chapter 5). Therefore, we propose *learning-based* methods capable of extending the discussed system by replacing the manually defined grasps for known objects with a deep neural network (Chapter 5), capable of inferring grasps poses from partial sensory observation even for unknown objects.

Another key assumption for the success of our reactive manipulation was precise and fast trajectory tracking. Although our system achieves good tracking using *model-based* inverse dynamics and PID control (Chapter 2.1.4), it requires high feedback gains to allow for unmodeled changes to the robot's dynamics when grasping and lifting objects. Because of the real-time perceptual feedback our presented manipulation system proactively avoids obstacles, thus, is very safe. However, due to high feedback gains required to compensate for *model* inaccuracies while grasping, it is not as compliant as possible, therefore not inherently safe. Chapter 6 discusses a *learning-based* method which is agnostic to the underlying *model-based* inverse dynamics method and improves its performance from data. This approach can further improve the fully integrated system, discussed in this chapter, without sacrificing domain specific robotics knowledge for *learning*.

Chapter 4

Online Decision-Making for Manipulation

Complex and dynamic environments require to close feedback control loops as demonstrated in the previous Chapter 3. Within this thesis, we have already shown that closing perceptual feedback loops allow *model-based* system architectures to perform challenging pick and place manipulation tasks while being agnostic to continuous environmental changes. However, here we argue that robust task execution for complex interaction tasks, beyond pick and place, requires to close feedback control loops in novel ways around many more sensory signals than done traditionally. Without considering different sensor modalities, the robotic system cannot cope with noise and uncertainty in the sensory-motor system of the robot and the environment during contact rich tasks because important events cannot be observed. For instance, when unscrewing the cap of a bottle force/torque measurements allow detecting if the cap becomes lose which is very difficult to infer from perceptual data. Low-level control systems increase the execution robustness by integrating feedback of high-bandwidth sensors, e.g., force/torque. Still, task execution often fails due to external perturbations that are hard to observe and *model* in the low-level control system, using high-bandwidth sensors. Hereafter, we propose to close a high-level feedback loop, leveraging additional low- and high-bandwidth sensor modalities increasing the task robustness using *learning*.

One challenge towards *autonomous manipulation* is to provide feedback controllers with appropriate reference signals and decide online *when* to consider alternative behaviors to counteract perturbations. *Model-based* reactive systems (see Chapter 3) have shown promising results for known manipulation tasks as long as their models are correct and no complex contact interactions are required. When such complex contact interactions are necessary, for instance, complex grasping interactions, modeling errors are inevitable and will significantly degrade the performance of the *model-based* planner (Weisz et al. 2012), e.g., for dexterous object manipulation. Uncertainty about the state of the manipulated object increases exponentially fast and therefore planning becomes intractable, unless smart heuristics/biases (Brock 2011) are applied.

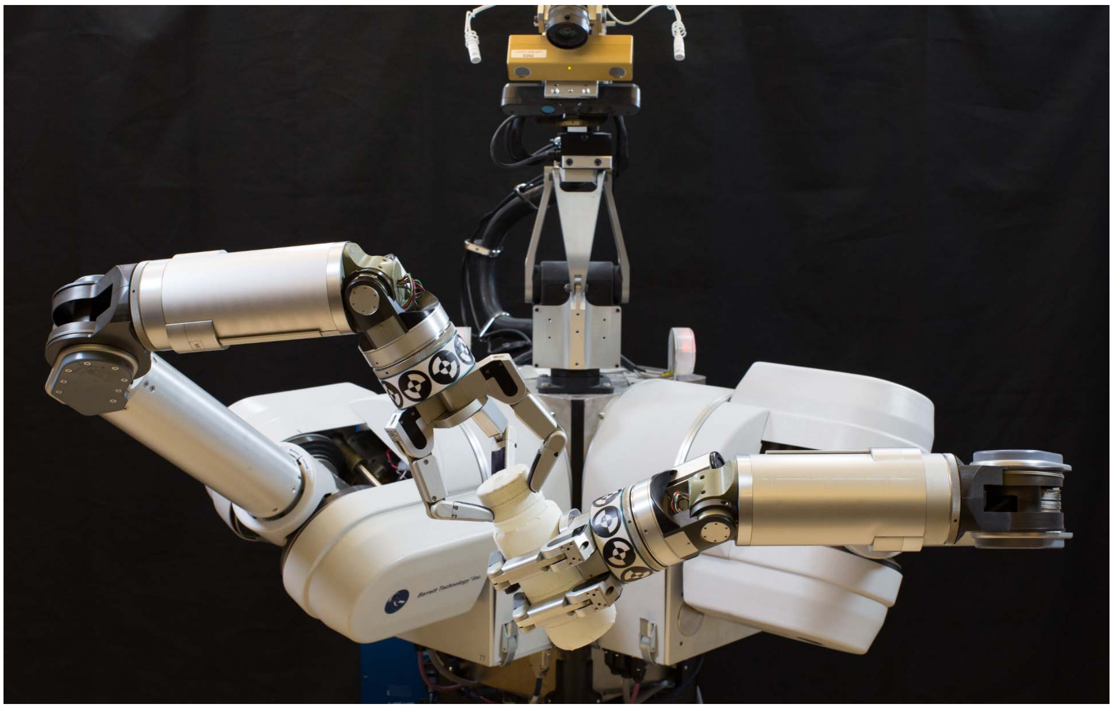


Figure 4.1: The ARM-S robot manipulating a bottle.

Additionally, *model-based* methods (Chapter 3) require proper task-specific objectives and programming. Here, we present a novel *learning-based* approach, treating manipulation tasks as a data-driven sequential online decision-making problem, bootstrapped from demonstrations. We do not neglect all *model-based* information, for instance, the perceptual system uses the same *model-based* probabilistic object tracker as before. To effectively use *learning from demonstration*, we leverage the fact that most manipulation tasks, e.g., unscrewing a bottle (see Figure 4.1), locally decompose into a sequence of skills. A sequence can be encoded in a state machine, termed *manipulation graph*, see Figure 4.3. Conceptually this graph is similar to the task decomposition used in the previous Chapter 3 with the main difference that we merely have to demonstrate skills and associate their high-level interconnection rather than determining objectives for *model-based* optimization-based planning and locally reactive control. This graph representation imposes constraints onto the possible sequence of manipulation skills, similar to how a grammar constraints the possible sequences of words to form sentences. Such a representation can be either inferred from data (Kroemer et al. 2014; Niekum et al. 2013) or provided by human operators, who usually have good intuition about the necessary high-level information and skill decomposition to accomplish a certain task. However, manually designing rules, based on current sensory information to determine whether or not the current skill execution is valid and where to start an alternative skill, is very hard. This is mainly due to differences in sensor modalities, noise characteristics, and the high dimensionality of the signal space.

In this chapter, we develop a method which addresses this complex sensor-based online decision making using two *learning-based* methods. We propose to decompose the underlying online decision-making problem into two related classification problems. Both problems have to operate in real-time to close the high-level feedback loop, learning *when* to switch skills and *which* skill to execute next. Sensory experiences, stored in the *manipulation graph* structure, contain all required information and should allow autonomous data extraction, only requiring user interventions at failure conditions.

Task-relevant perceived sensor signals are strongly correlated with the executed manipulation action (Pastor et al. 2011b) is the critical insight enabling this *learning-based* system. Thus, given similar task execution, the robot will perceive similar sensory feedback (Pastor et al. 2011a). Therefore, skills used throughout this chapter are encoded as Associative Skill Memories (ASMs), sensor information encoded in sync with movement primitives, as introduced in (Pastor et al. 2012). *Model-based* planning approaches (Chapter 3) address a more general global solution space, making it challenging to associate sensory traces with their execution. However, these associated signals are essential to training our classification-based online decision-making system (ODMS). To use stereotypical movements, we focus on solving a particular manipulation task instance in this chapter, unscrewing a bottle cap and removing it and screwing it back on. Note, the presented approach could also be used to demonstrated robust grasping behavior, bridging the gap between perceptual grasp pose prediction (Chapter 5) and manual grasp controller design (Chapter 3).

The development and discussion of the *learning-based* ODMS is the main contribution of this chapter. It decides in real-time, at any moment of the task execution, if the currently executed skill should be replaced with another one (*when*) and which skill to choose as a replacement (*which*). We only require human demonstration to learn stereotypical movements and intervention, stop task execution in case the system is about to fail. These real robot skill demonstrations and execution interventions result in a dataset, stored in the *manipulation graph* structure. ODMS automatically extracts supervised datasets from the real-robot experiments allowing for training our classification based decision-making system. Our formulation enables integrating sensors even if they contain no valuable information about the task. The importance of different sensor modalities at different execution stages is automatically inferred. Thus, manual task-specific feature selection is less crucial. We further report qualitative results of our method applied to a dexterous manipulation task performed by a bimanual robotic system, see Figure 4.1.

4.1 Related Work

Motion planning approaches such as (N. Ratliff et al. 2009; Kalakrishnan et al. 2011a) are mostly applied to problems neglecting object interactions, see Chapter 2.1.3, 3.1, and 3.4 for a more thorough discussion. One recent approach that creates plans for contact manipulation, proposed in (Dogar et al. 2010), still uses a quasi-static assumption to obtain a feasible search space. This work builds upon results reported in (Pastor et al. 2011a,b, 2012), a skill-based formulation towards autonomous manipulation. In contrast to (Pastor et al. 2012), we propose to close the high-level feedback loop by running a decision-making process online at all time and not only selecting the next ASM at the terminal condition of the current ASM. Different from (Pastor et al. 2011a), we propose to *learn* failure case prediction from a supervised signal, obtained from the ASMs organized in the *manipulation graph* structure. Additionally, we reduce the number of open hyper-parameters, compared to the previous work by (Pastor et al. 2011a). The *manipulation graph* is capable of generating a large number of skill sequences. This structure is assumed to be manually provided by a user similar to the manually designed task structure in Chapter 3. A manipulation graph could be constructed automatically, as proposed in (Kroemer et al. 2014; Niekum et al. 2013). However, the construction process is an open research problem by itself. (Kroemer et al. 2014) proposed to infer the hidden phases of the manipulation task from several trial executions, using an autoregressive Hidden Markov model. They train a logistic classifier (see Chapter 2.2.2.3), on a small, manually selected set of discriminative features to model the transition probability deciding when to switch between states in their inferred Hidden Markov model. (Niekum et al. 2013) present another approach to automatically infer a state machine representation which can be incrementally refined. Similar to (Pastor et al. 2012), this approach is limited to skill switching at the end of each skill execution.

4.2 Problem Formulation

Similar to existing *learning-based* approaches to manipulation, we assume that tasks can be decomposed into re-usable skills. Skills can be interpreted as low-level control systems (see Chapter 2.1.4). Figure 4.2 illustrates such a control loop (blue). Although low-level control systems allow counteracting local perturbations, they are not robust enough for manipulation in unstructured environments. This is mainly due to strong unmodeled perturbations (outside of the stability regions), simplistic model assumptions, or missing of high-level sensory feedback. Hereafter, we focus on closing a high-level feedback loop (Figure 4.2, green) using both low- and high-bandwidth sensors (Chapter 4.2.2), while being agnostic to the low-level control loop. This problem can be interpreted as a real-time online decision-making problem, capable of

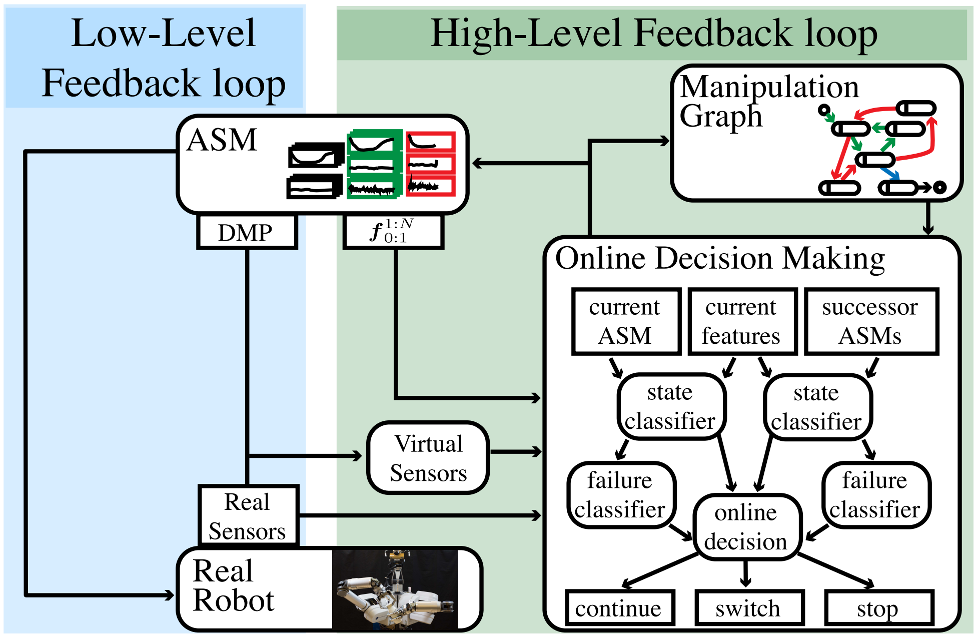


Figure 4.2: This figure illustrates our control concept towards autonomous manipulation. The left side shows the high-bandwidth low-level control loop. The DMP of the currently active ASM generates the reference trajectory (black) locally adapting due to (real) sensory feedback. The right side illustrates the main building blocks and their interaction, discussed in this chapter. Our system leverages previous sensor experience of successful (green) and erroneous (red) trials to train two sets of classifiers (state and failure). At runtime, the manipulation graph restricts the possible successor ASMs, considered for state classification. The box labeled 'Online Decision Making' shows the general data flow and the individual steps to close the high-level feedback loop.

orchestrating skills due to sensor feedback while improving from past experiences. In different from the system described in the previous Chapter 3, here we argue that the training of such a system should only require *common sense* user feedback and no low-level system knowledge.

4.2.1 Manipulation Graph

Inspired by language models, we believe that complex behaviors can be decomposed into skills (words). Similar to grammar models, a task-specific directed graph models constraints on possible skill sequences. Each graph node represents a skill, and an atomic unit represented as an ASM, see Figure 4.3. This graphical representation and the sensory experience contain all the required information for closing the high-level feedback loop.

Despite recent progress in structure learning, we believe that modeling such a directed graph is still a domain where humans outperform state-of-the-art inference methods. Therefore, we propose that a task-specific manipulation graph is constructed by a user, using skills from an existing ASM library or additionally taught task-specific skills. Initially, the manipulation graph consists of a linear sequence (Figure 4.3, green) which encodes the expected outcome of each skill. During task execution, the *manipulation graph* builds the connection between the executed ASM and the ASM library, allowing to add annotations such as success, failure, and the path through the graph. This information is initially based on user intervention for new ASMs, stopping as soon as the execution fails. As soon as both successful and erroneous trails are available, interventions can also be due to a trained ODMS. This initially linear graph is extended to achieve the target task by intervention if failures occur. For example, while executing *unscrew*, the cap of a bottle becomes loose and thus can be removed. Therefore, a failure case is triggered since the expected skill (*retreat*), opening the gripper, cannot be applied. Then either a new behavior (*remove*) is added as a possible successor to the current graph node, or a terminal stopping node is added, representing an unrecoverable system condition. This procedure guarantees an efficient teaching process, only considering human labeled failure cases, occurring on the real system[1]. Hence, it becomes much simpler to teach a new contact interaction task using this novel *learning-based* formulation compared to the purely *model-based* system presented in Chapter 3. Structuring available ASMs in a graphical representation reduces computational complexity and avoids perceptual aliasing, i.e., when the currently processed features cannot distinguish between two or more ASMs.

4.2.2 Features

For our demonstration-based system, we strive for requiring only a small amount of user demonstration to bootstrap the system. State-of-the-art deep neural network based approaches (see Chapter 2.2.2.4) are therefore not applicable; such methods require large datasets. In our small-scale data regimes, *learning-based* methods strongly depend on the information provided by features, and the representation itself.

Robotic systems usually provide a lot of different sensor modalities which allow observing both the state of the robot and of the environment. For some tasks, it might be intuitively clear which sensors to select, to acquire all task-relevant information manually. However, for many tasks, especially those that involve contact interactions and external perturbations from a nominal state, this may not be as clear. We argue that it is beneficial to include many sensors in the ASM representation and automatically infer their relevance from data. This allows using the same methodology for a broader range of tasks and perturbations.

[1] Previous related work (Pastor et al. 2012) does not support failure driven incremental teaching, since switching is assumed to take place exclusively during the last 10 percent of the ASM execution.

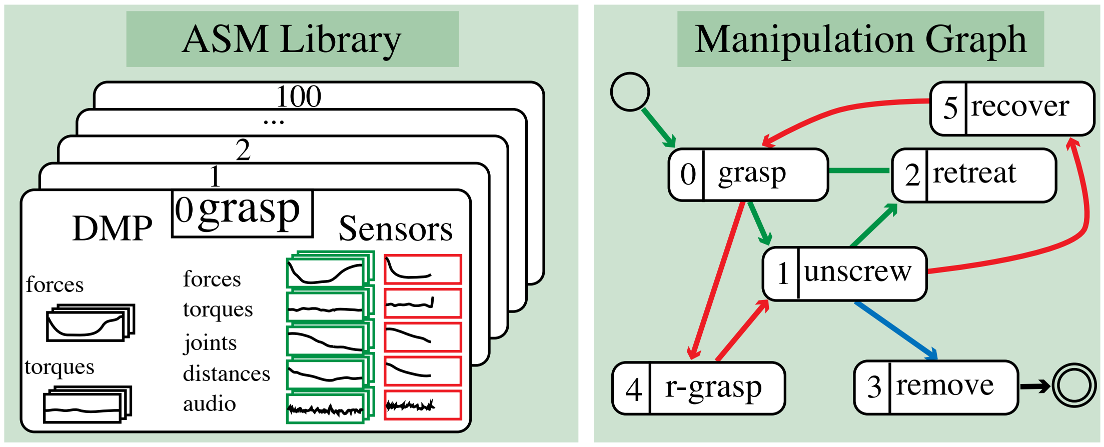
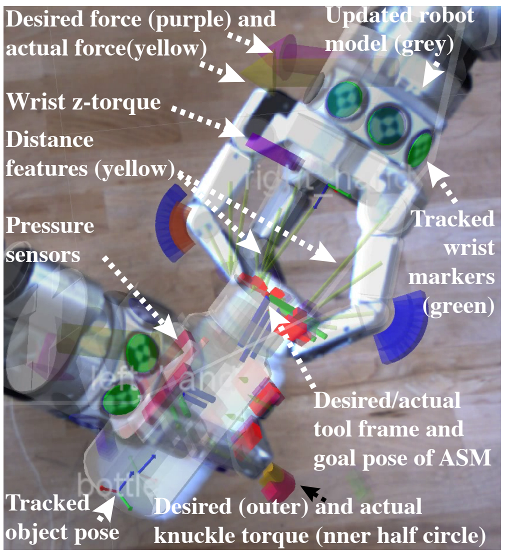


Figure 4.3: ASMs (left) contain a DMP which updates its internal representation based on several executions (black). Sensor experience is stored in sync with the DMP progress and split into successful (green) and unsuccessful (red) trials. Sensor signals for unsuccessful trials stop as soon as a failure is indicated. The right-hand side shows an exemplary manipulation graph for unscrewing a bottle initially (green) containing only a sequence of three ASMs, connected with their expected successor ASM. As soon as the cap is loose on the bottle we stop the execution and demonstrate a new ASM *remove* (blue), added to *unscrew*. Finally, we add additional recovery ASMs (red) which result in a more robust overall performance.

The inherent noise of real sensor readings increases the learning complexity of classification based systems (see Chapter 2.2.1.6). Therefore, we propose to enrich the ASM representation, compared to (Pastor et al. 2012), with additional (real) as well as (virtual) sensors, adding possible redundancy to the feature representation. Note, virtual sensors are possible since we use *model-based* object tracking, incorporating additional domain-specific information. Hereafter, we use the term features and sensors for both real as well as virtual sensor information interchangeably. Virtual sensors are features, computed online from (real) sensor signals. Such (manually) uncovered virtual features exploit task knowledge, e.g., the proximity of the robot's end-effector the target object, increasing robustness against sensor noise. These features provide an alternative channel to obtain similar information, e.g., proximity for force/torque measurements.

Auditory information is rarely being considered during manipulation, yet we believe that valuable information about the progress of a task can be inferred from this signal type. Especially failure conditions are frequently accompanied with sound feedback, as shown for object grasping by (Sinapov et al. 2014). Thus, we enrich our ASM representation with audio features. Figure 4.4 shows an exemplary feature representation, used for the experimental results. Chapter 4.4.1 provides further details of the feature acquisitions process.


Desired force (purple) and actual force(yellow)
Updated robot model (grey)
Wrist z-torque
Distance features (yellow)
Pressure sensors
Tracked wrist markers (green)
Desired/actual tool frame and goal pose of ASM
Tracked object pose
Desired (outer) and actual knuckle torque (nner half circle)
right hand
left hand
bottle

(a)

Feature count and descriptions	
4	Joint angles of the right hand
12	Force/torque measurements at both wrists
3	Torques at the knuckles of the right hand
10	Distance between locations on the visually tracked right hand and the corresponding closest point on the tracked bottle
13	Position and orientation (in quaternion notation) of the right palm and tool frame to the tracked pose of the bottle
11	Relative linear and angular velocities of the right palm, tool frame, and the tracked bottle
8	Mel-frequency cepstral coefficients (MFCC) computed using the audio recordings of the two microphones located in the sensor head

(b)

Figure 4.4: The left picture shows an image recorded from the left Bumblebee camera of the ARM-S robot (see Figure 4.1) overlaid with sensor and feature information. The green markers at the wrist are visually tracked and used to estimate the pose of the arm, overlaid in gray. The pose of the bottle is tracked using 3D information from the depth camera as introduced in (Wüthrich et al. 2013). Predicted and actual forces are also illustrated. The table on the right lists the feature count and a corresponding description of each feature used during the experiments.

4.3 Data-Driven Online Decision Making

The principal goal of the presented *learning-based* approach in this chapter is to close high-level feedback loops (Figure 4.2 (green)), integrating low-level control systems while improving from limited supervised data, using a transparent user process. High-level feedback loops can leverage information from low-bandwidth *model-based* object tracker and high-bandwidth sensors (force/torque), still operating in real-time (30 Hz), adding robustness to autonomous manipulation systems. The change of feature importance throughout the execution of skills is a common issue for the design of *learning-based* online decision-making systems. For example, force/torque sensor readings might have no relevance before object grasping and completely different signal characteristics after making contact. Partial skill execution, required to achieve robust recovery behaviors, adds further complexity to the ODMS design.

We address these issues by formulating the online decision-making problem as two related state-based classification problems (Figure4.2). A **state** (see Chapter 4.3.1) in the context of our method represents a distinct execution stage of an ASM, with a corresponding data set of feature readings, which is automatically associated from successful and erroneous trials. The first classification process predicts the most likely state of the currently executed skill and successor skills, discretizing the time series into independent **states**. Only possible successor skills, determined by the *manipulation graph*, are considered, reducing the computational complexity. Chapter 4.3.2 describes the optimization procedure for this classification problem and the automatic supervised dataset construction from previous task executions. The second classification task determines whether or not the sensory information is valid or not, meaning if the currently executed skill should be switched or the system should stop (Chapter 4.3.3). Finally, the high-level feedback loop is closed by fusing the outcome of all predictions, as described in Chapter 4.3.4.

4.3.1 Representation

We use the following data representation to optimize our proposed ODMS, obtained and updated using the information provided by the manipulation graph, sensory feedback, and user intervention. Each ASM, stored in the skill-library, can be identified by a unique id and each skill execution during real robot experiment has a corresponding trial id. We store the predecessor trial id and the successor trial id respectively for every execution, allowing us to draw connections between these executions. This information is automatically acquired during experiments, only requiring the user to stop executions at failure cases and select or teach appropriate recovery behaviors. While running an experiment, a pre-defined set of features is recorded at the highest possible frequency. The current feature readings are denoted as the feature vector $\boldsymbol{x}$. We assume that unsuccessful trials are interrupted before the skill is fully executed. Skills are represented by ASMs. Thus skill execution is based on DMPs (Ijspeert et al. 2013). Therefore, unsuccessful trials can be identified by the phase, meaning before the phase variable ρ of the DMP has converged $\rho = 0$. Thus, we store every trial in the ASM library and add success ($\rho = 0$) or failure ($\rho > 0$) labels. All (N) successful trials, associated with a particular ASM are aligned using ρ and discretized in the target frequency (e.g., 30 Hz), resulting in the time series data with T states $\boldsymbol{f}_{1:T}^{1:N}$, where T is determined by the skill execution time and target frequency. Every state $\boldsymbol{s} = \boldsymbol{f}_t^{1:N} \in \boldsymbol{f}_{1:T}^{1:N}$ is treated as a different class with corresponding data samples from the N successful trials. Unsuccessful trials are stored but not aligned.

4.3.2 State Classification

State classification predicts the most likely state given the current sensor signals $\boldsymbol{x}$ (Figure 4.5). The most likely state $\boldsymbol{s}_c$ of the currently executed ASM is computed independently of the most likely state of all possible successor ASMs $\boldsymbol{s}_s$ (Figure 4.2). Using the ASM progress ρ to estimate the current state $\boldsymbol{s}_c$ directly is sub-optimal due to perturbation and noise, e.g., object contact while grasping almost always happens at different stages but has a large effect on the sensor readings. Our formulation decomposes state classification into T isolated problems. All classifiers optimize their feature importance model completely independent of each other. This rather strong assumption enables our system to predict at which state a new skill should start since independent classifiers for each state $\boldsymbol{s}$ do not require any partial information up to the current state.

Our approach is based on two main assumptions. First, we assume that sensor readings $\boldsymbol{x}$ contain enough information to recover the corresponding state $\boldsymbol{s}$. Second, we assume that feature variations during successful executions are solely due to sensor noise or counteracting external perturbation, using low-level feedback control.

Skills have typically a duration in the order of seconds, thus, thousands of classes have to be considered to evaluate new sensor readings $\boldsymbol{x}$ (Figure 4.5). To ensure real-time classification, the computational complexity of each individual classifier must be rather low. Fortunately each classifier is only concerned with a short period of time, allowing us to use simple models. We use a probabilistic naive Bayes formulation (see Chapter 2.2.2.2), which assumes feature independence resulting in a diagonal matrix $\boldsymbol{\Sigma}$ for our Gaussian sensor likelihood models:

$$\hat{p}(\boldsymbol{s}|\boldsymbol{x}) \propto p(\boldsymbol{s}|\rho)\mathcal{N}(\boldsymbol{x};\boldsymbol{\mu}_{\boldsymbol{s}},\boldsymbol{\Sigma}_{\boldsymbol{s}}) \tag{4.1}$$

The ASM state is denoted by $\boldsymbol{s}$, $\{\boldsymbol{\mu_s}, \boldsymbol{\Sigma_s}\}$ represent the statistics of a particular ASM state, and $p(\boldsymbol{s}|\rho)$ a progress ρ dependent prior. The naive Bayes independence assumption is crucial for fast training and inference time. Training is reduced to computing the mean $\boldsymbol{\mu_s}$ and variance $\boldsymbol{\Sigma_s}$ of the state-specific dataset $\boldsymbol{s} = \boldsymbol{f}_t^{1:N}$, obtained automatically from the successful trials of the corresponding ASM (Chapter 4.3.1). Furthermore, incremental learning is possible by simply re-estimating the feature mean and variance of the likelihood model. This likelihood model automatically trades off the importance of different sensor modalities for each state, allowing to handle completely different sensor modalities.

The presented state classification method factorizes both over time and features. Therefore training and prediction can be parallelized for all states in the current and possible successor ASMs, crucial to achieving real-time performance. Due to recent advances in hardware, especially parallel computation accelerators such as GPUs, this can be done efficiently, scaling up to hundreds of possible successor ASMs. Since a progress based prior $p(\boldsymbol{s}|\rho)$ for our naive Bayes classifier would only be meaningful for the currently executed ASM, e.g., imposing a Gaussian prior around the current skill progress, we assume a uniform prior for both the current and all successor skills resulting in a maximum likelihood classification problem:

$$\hat{p}(\boldsymbol{s}|\boldsymbol{x}) \propto \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu_s}, \boldsymbol{\Sigma_s}) \tag{4.2}$$

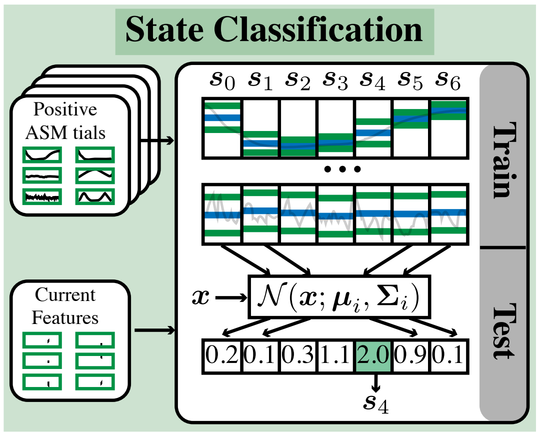
4.3.3 Failure Classification

Due to the complexity of manipulation tasks, failure cases are inevitable during task execution. However, we assume that the majority of the task executions succeed. Therefore, less data is available for failure case classification. Failure cases rarely happen several times at the exact same execution stage, adding further complexity to the prediction problem.

Since our aforementioned state classifiers already determine the most likely state $\boldsymbol{s}$ for both the current and possible successor ASMs, we address failure classification independently for each state. Hence, by associating sensor readings $\boldsymbol{x}$ from successful and unsuccessful trials of a particular ASM with their most likely state, our system automatically generates state dependent supervised datasets, enabling discriminative learning:

$$D' = \{(\boldsymbol{s}_k, d_k, y_k) : \boldsymbol{s}_k = \underset{\boldsymbol{x} \in ASM}{\arg\max}\, \hat{p}(\boldsymbol{s}|\boldsymbol{x}), d_k = \max_{\boldsymbol{x} \in ASM} \hat{p}(\boldsymbol{s}|\boldsymbol{x})\} \tag{4.3}$$

Success and failure label y are determined by the label of the trial (see Chapter 4.3.1). To associate feature vectors $\boldsymbol{x}$ from successful trials $\boldsymbol{f}_{1:T}^{1:N}$ with their corresponding state, leave-one-out cross



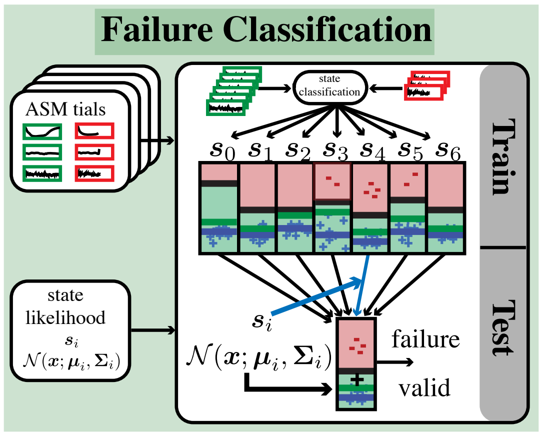

Failure Classification
ASM tials
state classification
state likelihood s_i $\mathcal{N}(x;\mu_i,\Sigma_i)$
s_0 s_1 s_2 s_3 s_4 s_5 s_6
s_i
$\mathcal{N}(x;\mu_i,\Sigma_i)$
failure
valid
Train
Test

Figure 4.5: State classification first discretizes all successful trials of a particular ASM for all sensors, shown in the top. The feature mean (blue) and standard deviation (green) is updated for each feature independently, based on the time aligned signals. At test time the likelihood of the current signal $\boldsymbol{x}$ is computed for all states $\boldsymbol{s}_i$ and the most likely state is returned.

Figure 4.6: Failure classification uses all ASM trials and assigns the likelihood $\mathcal{N}(\boldsymbol{x};\boldsymbol{\mu}_{\boldsymbol{s}_i},\boldsymbol{\Sigma}_{\boldsymbol{s}_i})$ of $\boldsymbol{x}$ to the most likely corresponding state $\boldsymbol{s}_i$. For each state we train a linear SVM (see Chapter 2.2.2.1) using the generated dataset. The decision boundary is propagated to neighboring states in terms of standard deviations of the samples from successful trials. Test time evaluation selects the most likely state and checks if the state likelihood is below the error threshold.

validation (Kohavi et al. 1995) has to be used, otherwise the resulting state classification would be overconfident. This means that state classification is trained on all successful trials except for the one which is currently added to the supervised dataset. Further, the state classifier, trained on all successful trials, assigns the last m feature vectors for unsuccessful trials of the corresponding ASM to the most likely states (Equation 4.3). Thus, for generating the supervised datasets the user only has to stop an execution in an error case. Since each state classifier returns the predicted likelihood, resembling the similarity of the current feature readings with the state, we can further simplify the given problem into a one-dimensional binary classification problem. Hence, our failure classifiers automatically benefit from model improvements of our state classifiers, further reducing the computational complexity at training and test time.

Although the particular classifier is not decisive for the success of the proposed method, using a discriminative classifier allows to easily integrate data from the state classifier without requiring further processing or modeling. We use the automatically generated dataset (Equation 4.3) to learn a max-margin linear support vector machine (SVM) (see Chapter 2.2.2.1) In the case of our one-dimensional problem, this optimization will result in a one-dimensional state dependent threshold $\varepsilon_{\boldsymbol{s}}$, which can be used to classify the predicted state likelihood into success or failure. One reason to choose a discriminative over a probabilistic classifier is the dynamical nature of the dataset generation. It is to be expected that the data distribution will change throughout experiments, resulting in multimodal distributions, adding additional model complexity.

Successful and unsuccessful trials have different effects on the decision boundary ε_s. All data samples are used to determine the decision boundary. Data samples from successful trials have an additional use case. They update the state classifiers such that similar feature readings become more likely and therefore affect the dataset generation (Equation 4.3). Given the system assumptions, our proposed ODMS does not require any threshold tuning, feature scaling, or additional data annotation. The single user intervention, stopping the system in failure conditions, is sufficient.

The decision boundary for failure case detection only exists for a fraction of an ASM due to the state independence assumption. This is theoretically sufficient to learn to predict failure cases. Practically this would require to demonstrate unwanted behavior multiple times to cover a generous space of an ASM. Therefore, we assume that the dataset (Equation 4.3) can be enhanced, by assigning failure examples to neighboring states. Based on the same assumption the learned failure detection decision boundary can be transferred to close by parts of an ASM, with vanishing impact based on the difference in ASM progress, e.g., exponential decay. Since the likelihood statistics of successful trials are different for different states of an ASM, we propose to propagate the decision boundary as scaling factor regarding one standard deviation of a Gaussian fitted to the likelihood predictions for all successful trials, see Figure 4.6. This decision boundary propagation enables failure classification for close by states when observing an equally unlikely sensory event.

4.3.4 Decision Making

The box *Online Decision Making* in Figure 4.2 visualizes the decision making process. First, the state of the currently executed ASM and independently for all possible successor ASMs is estimated using our probabilistic classifier for every state (Figure 4.5). If no failure is classified for the most likely state of the currently executed ASM, the results for the most likely successor are ignored, and the execution of the current ASM continues. Otherwise, the most likely successor state has to be analyzed by applying the corresponding failure classifier. In case of failure classification for both the most likely current and successor state our system halts, otherwise, the ASM is switched to the best successor ASM and state. Although the most likely successor state is only required if failure for the current ASM is classified, continuous classification of the most likely successor enables further post-processing, e.g., verifying the consistency of the successor, which we want to investigate in future work.

Switching could also be done solely based on the likelihood of the current and the best successor state. Perceptual aliasing and system stalling are two reasons to avoid this methodology. For example, a manipulation graph for robust grasping might require several alternative ASMs, but for all ASMs the final configuration might be very similar.

For each decision, our state classification complexity scales linearly with the number of considered states. Our proposed failure classification is constant with respect to states and features because it leverages the information provided by the state classifiers.

4.4 Experiments

The experimental setup (Figure 4.1) consists of dual arm manipulation setup similar to the one presented in Chapter 3. Here, we use the ARM-S robot, a dual-armed robot of two Barrett WAMs and a sensor head. For details about the platform we refer to `http://bit.ly/2xq1OtT`. The considered manipulation task consists of unscrewing a cap of a bottle as well as to screw the cap back on. This task requires very dexterous manipulation capabilities given the hardware constraints of the robot, the three-fingered Barrett Hands with four degrees of freedoms, and a tiny object compared to the size of the hand. Different from the pick and place tasks evaluated in Chapter 3.5 unscrewing a bottle requires exact finger placement. Any imprecision can lead to contact slippage resulting in a failure case. Further, this task requires additional sensor information to determine whether or not the cap is loose and can be removed. This would not be possible solely based on perceptual information as used for the system described in Chapter 3. All of our experiments were conducted with the same parameters (see Chapter 4.4.1). The presented results only reflect a subset of our experiments to illustrate the different properties of our proposed method. High-resolution videos of more experiments are available at `http://bit.ly/DDOMA` and `http://bit.ly/DDOMF`, also showing how to screw the cap back on the bottle.

4.4.1 Task Demonstration and System Parameters

For the exemplary bimanual manipulation task, we teach the required ASMs in the sequence, *grasp*, *unscrew*, and *retreat*. Every ASM is executed at least five times to gather the required statistics. As soon as the cap of the bottle comes off the first time during the execution of *unscrew*, we stop the execution and teach a new *remove* ASM. Therefore, we obtain a manipulation graph similar to Figure 4.3. For each trial execution, the ASM contains the previous and next manipulation graph node. We teach the DMPs for the corresponding ASMs similar to the methodology described in (Pastor et al. 2011b, 2012). First, movements are learned from

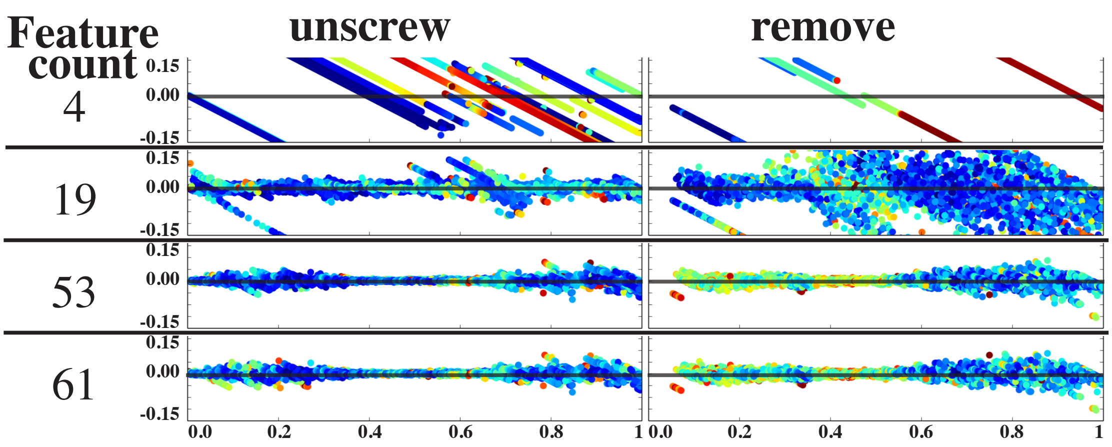


Figure 4.7: With the help of two skills (*unscrew*,*remove*), this plot illustrates the benefit of a rich set of features for state classification. The position of each data point indicates the estimated DMP progress (y-axis) vs. the deviation from the ground truth DMP progress (x-axis). Therefore, the closer the points are to the zero-line, the better the prediction. The colors indicate the negative log-likelihood, where blue means small therefore high confidence and red vice versa. With just four features, (right finger joints) it is impossible to classify the right state. Using 19 features, most similar to (Pastor et al. 2012), the state classification is possible for *unscrew* but still impossible for *remove*. Integrating virtual vision based features (53), we achieve a good correlation. Adding audio features (61), not providing any further task information, does not degrade our system performance.

kinesthetic demonstrations and encoded into DMPs. Second, learned DMPs are executed multiple times on the robot, and experienced sensor information is accumulated. Third, the mean position and force trajectories are computed and re-encoded into DMPs. During execution, the end-effector and finger positions, as well as forces, are controlled and adapted as described in (Pastor et al. 2011b), i.e., only positional and force trajectories are encoded into DMPs. Given the re-encoded DMPs, the experienced 61 features (see Figure 4.4) from at least five trials are linearly re-sampled (30 Hz) and stored time aligned with each DMP to form ASMs.

In contrast to (Pastor et al. 2012), we encode end-effector trajectories into the start frame of the movement. This encoding scheme generates movements that are locally similar to the demonstration. It thus enforces a strong correlation among subsequent movements despite changing absolute coordinates. End-effector force/torque trajectories are encoded in the goal frame of the movement (see (Pastor et al. 2011b)). This ensures that the desired forces/torques are independent of the (global) end-effector pose. We track the target object in real-time using the *model-based* algorithm proposed in (Wüthrich et al. 2013), matching 3D information from the depth camera against the geometric model of the bottle. Based on the object pose we enrich our ASM representation with ten additional distance-based features between the manipulator and the surface of the tracked target object, e.g., the fingertip to bottle, palm to bottle, finger

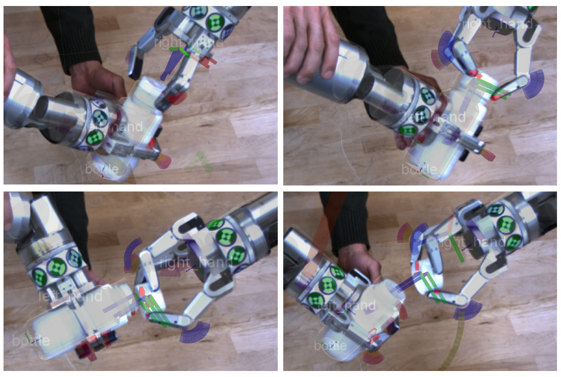


Figure 4.8: Snapshots of the robot performing the bottle opening task despite perturbation introduced by moving the left hand of the robot. The compliance of our control framework including DMP adaptation as well as closed-loop visual tracking of the bottle and both hands enable the robot to give in without losing the necessary accuracy to perform the task. The corresponding video is available at `http://bit.ly/2MzRiVt`.

segment to bottle distance (Figure 4.4 distances). Parallel to using vision-based virtual features, our system integrates vision sensors to improve the estimation of the robot end-effector pose, directly improving low-level feedback control. Therefore, we use fiducials, round markers, to visually track the hands during manipulation, arranged around the wrist (Figure 4.1). The fiducial detector [2] is used in combination with triangulation to estimate the 3D position of each marker to obtain a least-squares fit of the hand pose. Together with the known control input, these measurements are used in a Kalman filter to smooth the position estimates over time. Besides, we use the lower 8 Mel-frequency cepstral coefficients (MFCC) (Davis et al. 1980) as audio features. They sufficiently cover the frequency spectrum of interest. To obtain the state classifiers for each ASM, the mean and variance are computed as described in Chapter 4.3.2, using only successful trials. We obtain the failure classification thresholds $\varepsilon_{\boldsymbol{s}}$ for all ASMs for which erroneous trials exist, using the automatically generated datasets (Equation 4.3) with the last $m = 5$ samples of erroneous executions and all corresponding positive trials (see Chapter 4.3.3). Thresholds $\varepsilon_{\boldsymbol{s}}$ are propagated to neighboring states using exponential growth ($\varepsilon_{\boldsymbol{s}_i} = \min(\varepsilon_{\boldsymbol{s}_i}, 1.05^{|i-k|}\varepsilon_{\boldsymbol{s}_k})$), since we are optimizing the negative log likelihood. Thresholds are always encoded concerning the state variance of the negative log-likelihood based on the generated datasets (Figure 4.6).

4.4.2 State Classification and Feature Importance

Our proposed method is based on the assumption that we can *learn* a model for every state individually and identify the overall system state given the current sensor readings. Figure 4.7 illustrates that our method is capable of state classification, despite the strong independence assumption. Results are obtained by leave-one-out cross-validation, training the state classifiers for *unscrew* and *remove* on at least five successful trials. The correlation between the classified

[2] The fiducial detector has kindly been provided by Paul Hebert, Jeremy Ma, and Nicolas Hudson from the Jet Propulsion Laboratory, Pasadena.

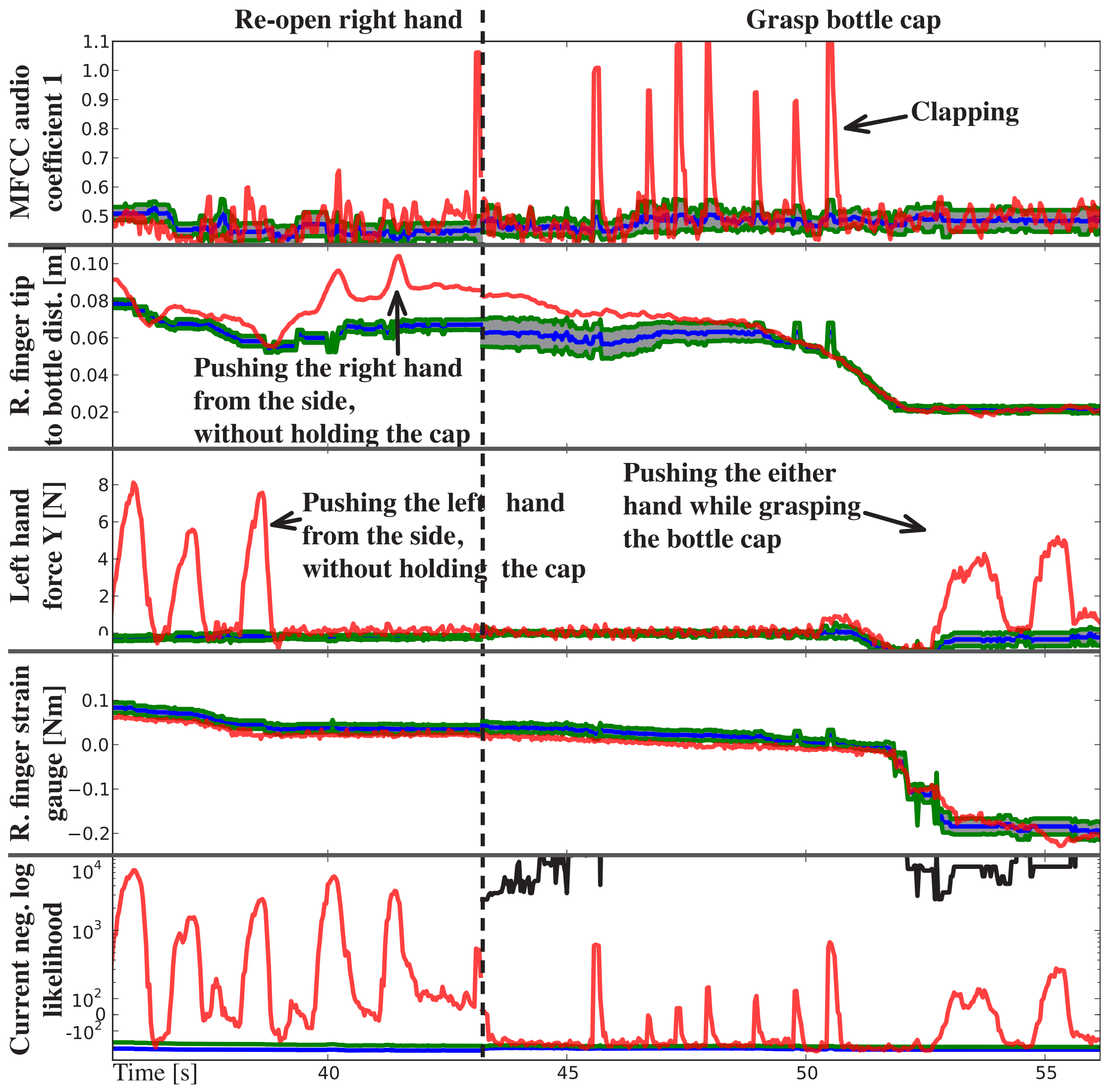


Figure 4.9: The top four plots show the expected mean feature value (blue), standard deviation (green), and the currently processed feature value (red) of a subset of all 61 processed features. Horizontal dashed lines indicate transitions between subsequent ASMs. The selected features are the first MFCC coefficient, the distance of the right fingertip of the right hand to the visually tracked bottle, the Y component of the experienced force at the wrist of the left hand, and the torque experienced at the right finger knuckle of the right hand (top to bottom). Note, due to space constraints the plot only shows 4 out of 61 features. Nevertheless, all features (auditory, visual, and haptic) are used for classification. The negative log-likelihood (bottom) shows that each individual perturbation of each sensor modality (top three plots) has been scaled appropriately since each perturbation became noticeable. The corresponding video is available at `http://bit.ly/2OrfV8L`.

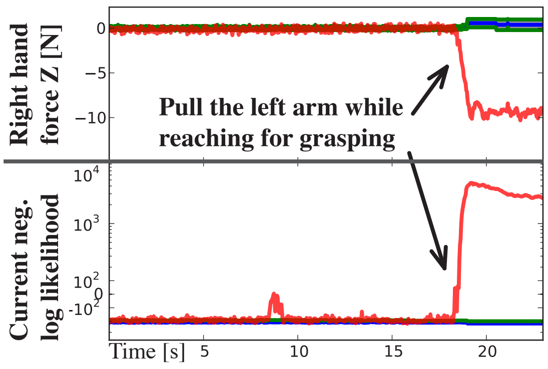
state and the DMP progress is less good at the beginning, the end of an ASM execution, and when object contact happens. Such events rarely happen at exactly the same time which is why the DMP progress is not a good estimate for the current state. One key difference to other approaches (Pastor et al. 2012; Niekum et al. 2013; Kroemer et al. 2014) is that we propose always to use a high dimensional feature representation and automatically infer the feature importance. Figure 4.7 supports the hypothesis that more features improve the system performance, although not required for all skills. The lower two plots show that our method does not degrade in performance even if we add features which contain no further information for the task. This supports the hypothesis that our method is capable of learning the changing feature importance for an ASM and thus, manual task-specific feature selection is less crucial.

4.4.3 Robustness through Perception-Action Loop

Robustness of our system is achieved by closing two control loops. The low-level loop implements a compliant control framework, using the DMP adaptation mechanism as proposed in (Pastor et al. 2011b) and integrating information from visual based tracking of both hands and the bottle. To demonstrate the low level-adaptability, we introduce a large perturbation by moving the left arm during the manipulation task as depicted in Figure 4.8. Despite this perturbation, the right hand can maintain contact and continue the DMP execution due to vision in the loop, adapting the goal of the DMP online. This approach significantly differs from Chapter 3 since we require stereotypical motion generation to associated sensor traces for learning rather than fast and reactive planning. The high-level control loop is capable of detecting perturbations of different sensor modalities as shown in Figure 4.9. Our method can automatically determine the importance of all available 61 features and fuse the information into a one-dimensional signal, for which failure detection is easily achieved.

4.4.4 Failure Demonstration and Detection

Adding recovery behaviors incrementally in the case of failure, e.g., due to external perturbation, is a key ingredient to add robustness to task execution. To demonstrate a typical use case, we start our manipulation graph and pull the left arm during the first *grasping* attempt. Although our system classifies the event as highly unlikely, it will not detect a failure condition initially, see Figure 4.10. This design choice, to assume that even unlikely events will not result in failure conditions if not known otherwise, allows to bootstrap our system easily but is not required for our system. After retraining the classifiers of the corresponding ASM (Chapter 4.3.2 and 4.3.3), taking this erroneous execution into account, our system detects perturbations of similar magnitude and will stop (Figure 4.11). The only user information required to teach this



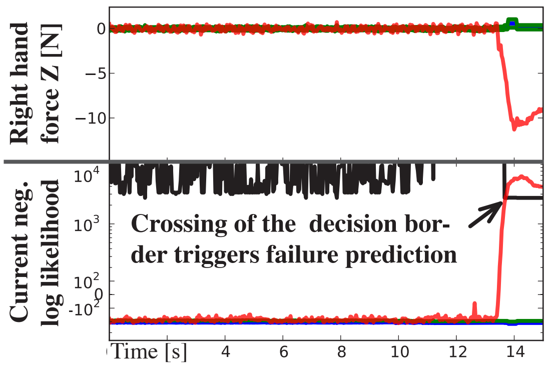


Figure 4.10: The top plot shows the measured right wrist force in the Z direction (red) as well as the expected mean (blue) and standard deviation (green). The intentional perturbation is immediately detected as shown in the lower plot indicated by the rise of the negative log-likelihood (red). Without demonstrated failure cases our system assumes that the introduced perturbation, although highly unlikely, does not constitute a problem.

Figure 4.11: The plots show the same experiment as in Figure 4.10 *after* the data obtained in Figure 4.10 has been processed as an unwanted (erroneous) execution. Given this failure case example data, our approach updated the decision border (black) which now enables our system to detect the unwanted failure condition indicated by the arrow correctly. Now, our system immediately stops the execution, as shown in the video available at `http://bit.ly/2NgBLzf`, if a similar perturbation is detected.

failure detection is stopping the current ASM execution before it has finished. If a possible successor/recovery already exists, the user can add the connection in the manipulation graph to enable automatic recovery. It is not clear how such a strategy could be achieved within the *model-based* reactive motion generation framework (Chapter 3).

4.4.5 Failure Detection and Recovery

Figure 4.12 shows a sequence of automatically chosen ASMs, successfully unscrewing the cap of a bottle without external perturbation. This experiment was part of a longer experiment to unscrew the bottle. During which we repeatedly screwed the cap back on the bottle (during ASM *retreat* execution). This demonstrates that our method automatically detected the subtle event, when the cap becomes completely loose from the bottle, prior to finishing the ASM execution and successfully switched to ASM *remove*. Figure 4.13 shows the prediction of *unscrew* at different stages of a successful attempt to unscrew a cap of a bottle with manually introduced unknown perturbations. For this experiment, an additional ASM has been taught named *recover*. This ASM has been added to the manipulation graph as an additional outcome of *unscrew* after stopping the execution of *unscrew* at the beginning of an external perturbation similar to the one in this experiment. Our method successfully determines if the sensor information

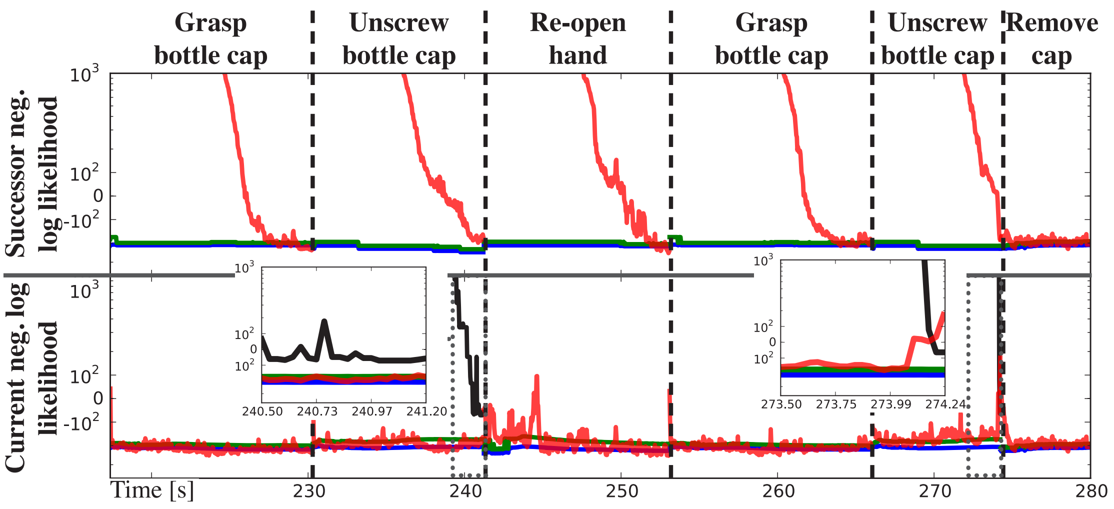


Figure 4.12: Both plots show the predicted negative log-likelihood (red) in log scale of the most likely state within the currently active ASMs. The bottom plot shows that the system is certain about the task progress since the predicted negative log-likelihood remains close to the expected mean. The top plot shows that the successive ASM becomes more likely towards the end right before the transition to this ASM. Note that the duration of the second unscrew movement is shorter than the initial unscrew movement since our approach correctly detected that the cap came off the bottle. The two overlaid plots show a closeup of these two events. The subtle event of lifting the cap has been detected as the negative log-likelihood (red) passed the decision border (black).

indicates a failure condition and selects the right successor ASM, either *recover*, *retrieve*, or *remove*. *Model-based* approaches (Chapter 3) would potentially not require such fine grain control over recovery behaviors. However, there exists no straightforward strategy to provide similar high-level information to *model-based* systems allowing them to solve the task in future attempts automatically. Again, the skill execution time in different runs varies due to the online detection of failure conditions. Some failure conditions are detected while not even reaching half of the expected skill execution time (Figure 4.13, left), not possible without closing the high-level feedback loop (Pastor et al. 2012; Niekum et al. 2013).

4.5 Summary and Review

In this chapter, we focused on how to *learn* a complex contact interaction task by using online failure detection and recovery. Determining whether or not a skill or planned trajectory execution exhibits expected sensory measurements is crucial for contact rich manipulation. The *model-based* system presented in the previous Chapter 3 proactively avoids obstacles and therefore achieves robust pick and place performance. However, detecting *when* the reactive manipulation

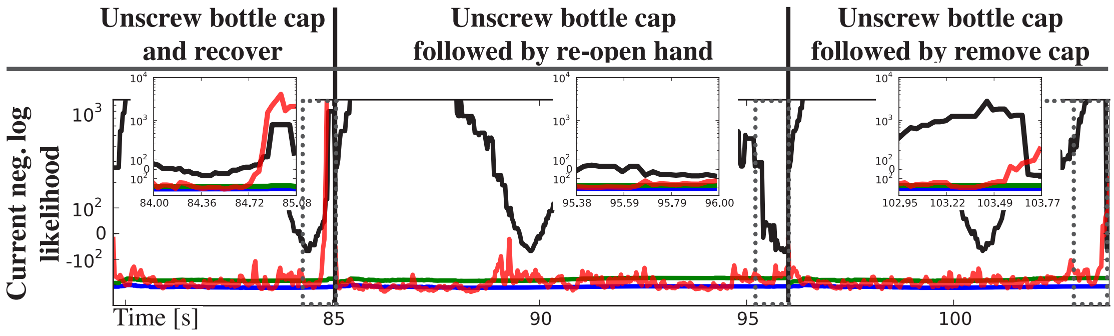


Figure 4.13: The plots show the three considered outcomes of one real experimental attempt of unscrewing the bottle, triggering the learned recovery behavior after pulling the left arm (left), triggering the retreat behavior because the cap of the bottle is not yet detached (middle), and triggering the remove behavior immediately after cap has correctly been detected to have detached from the bottle (right). Three small plots at the bottom show a close up of the negative log-likelihood at the point of switching to the corresponding successor ASM (dotted line). The solid horizontal lines indicate transitions. Note, the intermediate ASMs have been left out due to space constraints. The complete experiment along with corresponding plots is available at `http://bit.ly/2pnbhOb`.

system should abort grasping and *which* alternative objective it should optimize next based on the latest sensor measurements is unclear. Systems like the one described in Chapter 3 are very general and therefore require more abstract target parameterization. It is challenging to infer these parameterizations from low-level sensory data. Hence, in this chapter we presented a *learning-based* approach, exploiting the assumption that stereotypical movements result in similar sensory traces during execution. By leveraging stereotypical movements and an object-relative encoding, we can learn a classification-based real-time online decision-making process. Our experiments demonstrate qualitatively that the proposed method is capable of detecting otherwise catastrophic perturbations and failure conditions in real-time, using 61 features from different sensor modalities. Furthermore, our system automatically triggers recovery behaviors if needed, closing a high-level feedback loop. These results were achieved without the need for any task-specific programming. Therefore, we believe that this method can complement reactive *model-based* systems (Chapter 3). The presented approach is by no means a replacement for the reactive free space obstacle avoidance capabilities achieved by *model-based* systems (Chapter 3). ODMS is ideal for splitting up higher-level actions, such as establishing contacts for grasping, into robust low-level skills using multimodal sensory feedback.

Non-expert users can tackle this problematic robotic manipulation challenge by demonstrating robust strategies. Nevertheless, before manipulating objects, like opening a cap from a bottle, we often want to grasp the object which so far was assumed to be known and either demonstrated or hard-coded by a user. Therefore, we are interested in inferring possible grasps from partial perceptual sensory observations, discussed in the following chapter, to, for instance, automatically select an ODMS for robust grasping.

Chapter 5

Big Data Grasping

In this chapter we seek to answer the question: "How can *learning-based* grasping, learning where to position a robot gripper, improve grasp success towards autonomous manipulation?". Let us first take a step back and analyze why grasping is even necessary and why the previously presented *model-based* and *learning-based* methods are not sufficient.

Grasping is a fundamental requirement towards *autonomous manipulation* systems (see Chapter 3). Almost any robotic manipulation task requires tools to be picked up for usage or repositioning objects, classical pick and place tasks. So far the presented approaches in this thesis tackled this problem using human supervision, either in the form of manually defining grasp poses (Chapter 3) or demonstrations (Chapter 4). The former approach requires object identification and precise object tracking to associate and successfully execute grasps. This methodology cannot scale towards true autonomy. Such systems would have to know all objects, have to reliably identify objects from partial observations and additionally for every such object robust grasps would have to be manually engineered before deployment. Manually engineering robust grasps requires feedback policy design due to complex contact interactions, a challenging problem in its own right. The demonstration-based approaches (Chapter 4) alleviate this design step by *learning* reference and expected sensory signals to identify and automatically recover from grasping failures, allowing to learn robust grasp controllers. Nevertheless, demonstrating grasps and associating them with objects still requires to identify and track the corresponding object *model*. Hereafter, we argue that we can complement the previously discussed approaches by *learning* grasp poses rather than using *model-based* grasping. Such a *learned* grasp pose prediction method can be extended to enable reusing taught demonstration-based grasp controllers (Chapter 4) and naturally fits into our reactive *model-based* system (Chapter 3) to plan to the desired grasp pose.

Classically, grasps are synthesized using *analytic* approaches that rely on a known set of contact points between the object and gripper. Given this, the predominant metric for grasp stability is the ε-metric (Ferrari et al. 1992) that quantifies the quality of a force-closure grasp regarding the smallest external wrench that breaks a grasp when unit force is applied at the contact points. As already mentioned, analytic approaches require knowledge of the object and robot hand model and the relative pose between them. When executing such a synthesized grasp in the real world, achieving these contacts is very difficult because of noisy sensor readings and imprecise actuation. Several papers have analyzed the ε-metric in terms of robustness and concluded that it is not a good predictor for a stable grasp in the real world (Balasubramanian et al. 2012; Diankov 2010; Weisz et al. 2012). *Learning-based* methods towards grasping (Bohg et al. 2014a) address the challenges of grasp synthesis arising in the real world such as noisy sensors and incomplete information about the object and environment. They focus on finding a suitable representation of the perceptual data that allows predicting whether a grasp will succeed or most likely fail. These grasps are typically defined only in terms of a preshape, an approach vector and a wrist rotation. Hence, they still depend on robust grasp controllers, either engineered or *learned* with methods such as discussed in Chapter 4. In that way, they form a natural precursor to reactive grasping approaches for robustly acquiring a grasp under uncertainty (Felip et al. 2009; Hsiao et al. 2009, 2010; Pastor et al. 2011b; Romano et al. 2011).

Many *learning-based* methods try to infer a grasp for an object without knowing its exact shape. Typically, they employ supervised learning techniques that predict the stability of a grasp when applied to a specific part of the environment (Detry et al. 2010, 2012; Goldfeder et al. 2009; Herzog et al. 2014; Lenz et al. 2014; Morales et al. 2004; Popović et al. 2011; Saxena et al. 2008b; Song et al. 2011). One problem with these kinds of methods is to get a sufficient set of coherently labeled data points. In (Detry et al. 2010, 2012; Herzog et al. 2014; Morales et al. 2004), the authors collected examples from a real robot performing grasps. Although the resulting data is instrumental, it is extremely time-consuming to acquire. In other approaches (Goldfeder et al. 2009; Lenz et al. 2014; Popović et al. 2011; Saxena et al. 2008b; Song et al. 2011), this problem is circumvented by either creating a synthesized or human-annotated dataset with discrete or continuous labels. Although some of these approaches are demonstrated on a robot, it remains unclear how well these grasps transfer to the real world. Furthermore, only (Goldfeder et al. 2009; Lenz et al. 2014; Popović et al. 2011; Saxena et al. 2008b) provide publicly available datasets.

Therefore, this chapter presents three significant contributions to improve *learning-based* grasping. First, having big datasets is critical for state-of-the-art *learning* methods such as deep learning (Chapter 2.2.2.4) due to a large number of open parameters that need to be optimized. Hence, we propose a new database containing a large set of grasps generated for a large set of objects in the OpenRAVE simulator (Diankov 2010). We use physics-based simulation because running real-world large-scale robotic grasping experiments involves controlling complex hardware, rendering generating hundred thousands of examples challenging outside of industrial settings. Our proposed algorithm labels grasps with a set of metrics that we analyze in terms of their correlation with grasp stability as predicted by humans. Assuming that humans are optimal in judging whether a grasp will succeed or not, we use crowdsourcing to verify both our proposed physics-metric and the ε-metric.

Second, with the help of the proposed database, we investigate the application of classical supervised machine learning techniques (see Chapter 2.2.1) that require large amounts of training data to learn the mapping from a grasp to grasp stability. Grasp stability is influenced by many aspects that include local and global object shape, object pose relative to the hand or the context in which an object is placed. Our methods use modified features, presented in (Herzog et al. 2014) and do not require knowledge about object identity, category, pose or accurate segmentation. This aspect is crucial to overcome existing limitations in *model-based* and *learning-based* approaches discussed in the previous chapters. We show that large-scale learning methods like deep neural networks indeed outperform simpler logistic regression, making better use of the vast amounts of training data. Furthermore, we provide evidence that our new physics-metric based grasp success annotation is easier to learn for both methods compared to the standard ε-metric. This result suggests that the ground truth labels derived from the ε-metric are much noisier.

Finally, we introduce a new perspective on grasp success prediction, changing the focus from traditional classification-based learning objectives to structured ones (see Chapter 2.2.1.7). Since robots only observe a partial view of a scene and crucially only execute one grasp, hopefully a successful one, we hypothesize that our underlying *learning-based* method should be aware of and exploit this information. We derive a novel predictor from a structured prediction objective and show empirical evidence on our large-scale dataset that without changing the underlying function approximator, this objective indeed results in better *top-1* grasping performance.

5.1 Related Work

Prior to 2000, research in grasping has been focused on analytical *model-based* approaches discussed in several surveys and reviews (Shimoga 1996; Bicchi et al. 2000). More recently several studies have shown that *model-based* approaches using the well established ε-metric in simulation do not transfer to real robot grasping (Balasubramanian et al. 2012; Diankov 2010; Weisz et al. 2012). Therefore the grasping community started to focus on data-driven, *learning-based* approaches. As previously mentioned there exists a large body of *learning-based* methods that use differently acquired datasets to learn grasp mappings from vision-based representations to grasp success using classifiers trained in a supervised manner (Detry et al. 2010, 2012; Goldfeder et al. 2009; Herzog et al. 2014; Lenz et al. 2014; Morales et al. 2004; Popović et al. 2011; Saxena et al. 2008b; Song et al. 2011). As labels, they either use the ε-metric, human annotations or the outcome of real robot trials. For these approaches, different machine learning techniques have been applied to learn a grasp success model, e.g., linear or non-linear discriminative classifiers or non-parametric methods. We refer to (Bohg et al. 2014a) for a more comprehensive survey recent *learning-based* grasping methods. Up until 2015, there were only a few approaches for robotic grasping and robotics in general that used deep learning techniques. This can be mainly attributed to the lack of large publicly available datasets at the time and system challenges towards large-scale real robot data collection efforts. Our work on creating a large-scale publicly available database, presented hereafter, was among the first to address this issue.

However, before 2015, there are two examples that we would like to mention here. (Lenz et al. 2014) frame the problem of finding a stable grasp as a detection problem. A grasp is represented by a rectangle which is positioned on an RGB-D image. This rectangle has 5 DoFs (2D position, orientation, and dimensions) and multiple hypotheses need to be evaluated. Color, depth, and normals serve as features of the potentially graspable part that is enclosed in a rectangle. The authors train a deep network to predict the stability of a grasp given this representation. The resulting classifier is used to demonstrate the approach on two robots with parallel-yaw grippers. The annotated training database is relatively small and consists of 1035 real RGB-D images each with several hand-labeled positive and negative rectangles. These images show a total of 280 different objects.

Another example of applying deep learning to predict the stability of a grasp is presented by (Madry et al. 2014). Here the goal is to predict the stability of a grasp during execution. Given the reading from tactile sensors on the robot fingers, the authors apply a learned classifier to distinguish between stable and unstable grasps. Excellent results are achieved on different datasets recorded with different robots. Also, in this case, the databases are relatively small with, e.g., 926 data points (Bekiroglu et al. 2010). Although this number is impressive when considering that it has been collected and verified on a real robot, it is small when compared to the ImageNet dataset with a total of one million images with bounding box annotations (Deng et al. 2009). Furthermore, maximally ten different objects are included in these different databases.

(Levine et al. 2016b) presented the first truly large-scale real-world grasping data collection effort. Different from methods discussed in this thesis, their approach is purely *learning-based*, and the resulting data has no ground truth object annotation. Hence, it only allows learning mappings directly from 2D image space to poses (e.g., the 3D position and roll around the gripper). (Pinto et al. 2016) also conducted a self-supervised data collection effort, again resulting in a dataset which only allows regressing poses from 2D images without any object-specific ground truth data annotations. Instead, our work uses a large number of object models, problem-specific grasping knowledge, and synthesizes and annotates grasp with a new success metric and ground truth sensory information. Since the inception of our large-scale database, new simulation-based strategies to create vast datasets for *learning* how to grasp have been proposed (Mahler et al. 2016, 2017).

All previously mentioned *learning-based* methods for grasp prediction either directly try to infer grasp poses or classify grasp hypotheses into successes or failures. However, by doing so, any such approach neglects the crucial information that we are typically interested in executing a single successful grasp on our robotic system. Hence, we do not have to find all possible successful grasps. Ranking is a structured prediction problem (see Chapter 2.2.1.7) which is ideal for such problems. Therefore, various applications, including computer vision, robotic locomotion, and grasping, make use of this framework. (Lehmann et al. 2011b) proposed to speed up object search by reducing the number of expensive classifier evaluations by learning how to rank sets of hypotheses. (Kalakrishnan et al. 2009) use ranking to learn to select footholds based on terrain templates, enabling robust walking over rough terrain. (Herzog et al. 2014) implicitly create a ranking over different grasp hypotheses by continuously extending a library of labeled local shape templates and their matching costs. Finding the best fingertip placement on objects for object grasping has been identified as a ranking problem by (Le et al. 2010). The authors manually label the training data with three different classes (Bad, Good and Very Good) instead of two. As a learning method, they employ a ranking SVM that optimizes a measure that prefers better scores for the top grasp candidates. (Jiang et al. 2011) present an extension to this work with a different representation of the grasp but otherwise the same SVM-based ranking method. Our work presented in this chapter differs from this line of work since it directly operates on binary labeled sets of hypotheses rather than requiring additional annotations. This methodology is very closely related to multiple instance learning (MIL) (Dietterich et al. 1997; Andrews et al. 2002). By deriving our formulation from a ranking perspective, we can exploit additional problem-specific insights and use state-of-the-art non-linear function approximators that scale to our large-scale grasping database.

5.2 Model-based Large-scale Grasp Data Generation

Discriminative *learning-based* methods such as deep neural networks (Chapter 2.2.2.4) can uncover complex relationships from large datasets. However, incorporating prior problem-specific knowledge, for instance, what constitutes a stable grasp, is rather challenging. Here, we present a simulation-based approach which leverages our *model-based* grasping knowledge and improvements in distributed computation to generate a large-scale database of ground truth grasps. Different from applying *model-based* grasping for real robot experiments, we can run more expensive physics-based simulations since we merely generate high quality labeled grasp success and failure pairs. Our database contains more than 700 object instance from more than

80 categories. For each object, we simulate approximately 500 grasp hypotheses, each of which consists of more than 30 simulations to get high-quality stability metric estimates. We run a crowdsourcing Mechanical Turk user study verifying the quality of our proposed physics-metric. Our large-scale simulation effort yields approximately 300k different grasps. For each grasp, point clouds are generated from several different viewpoints. For every viewpoint, we generate a template representation and associate metrics obtained from the simulation with it, resulting in about 500k labeled template examples. Hereafter, we discuss the data collection and verification effort in more detail.

5.2.1 Database

Figure 5.1 (left) shows an overview of the dataset generation. A labeled data point in the database consists of a template (local shape representation) and a set of metrics. Each point is associated with a grasp preshape relative to the local coordinate frame of the feature. We store the database in a single HDF5 file (*HDF group - HDF5* n.d.) which allows fast and efficient querying of single grasps. Furthermore, this file format allows to easily extend the grasp database with more grasp metrics, robotic hands, simulation environments, and feature representations. The database is publicly available for download (**Kappler** et al. 2015a). For easy data access, we provide a python and C++ interface to the library, also available on the same website.

5.2.1.1 Objects

Our database contains more than 700 object mesh models from more than 80 categories. These categories contain household objects, tools, toys, music instruments, groceries or drugstore products. They span a wide range of shapes and sizes. Most of the mesh models are from the 3DNet database (Wohlkinger et al. 2012) with further augmentations from (*Archive 3D: Free 35 000+ 3D models* n.d.; *TurboSquid: 3D Models for Professionals* n.d.). All models have been preprocessed such that they are suitable for the OpenRAVE simulator. Specifically, we scale every object mesh such that the mean size of the objects in a category is realistic for this category. Furthermore, each model is re-meshed using Poisson surface reconstruction (Kazhdan et al. 2006) and subsequently simplified to reduce the number of polygons. Although the objects lose some details, this procedure ensures that the meshes are not degenerate and that all surface normals are pointing outwards. This preprocessing procedure is important for the correctness of the collision checker and the computation of the contact normals. Some example models are shown in Figure 5.2.

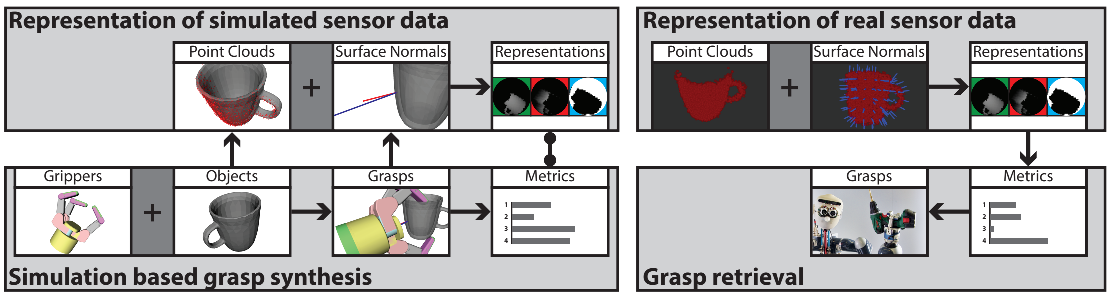


Figure 5.1: (Left) To generate supervised training data samples, we propose to store the following quantities in a database: Models of different grippers and objects to leverage simulation environments that use sampling techniques to synthesize possible grasp configurations and poses, e.g., as described in (Diankov 2010). Different grasp metrics can be obtained for each of the sampled grasp hypotheses. For the generation of local shape representations, point clouds are generated from the object model given a certain viewpoint. To establish the relationship between the point cloud and the computed grasp score, the local shape is represented relative to the surface normal at the intersection between the approach vector of the grasp and object model. The local shape representation is a heightmap of different point types: object surface (green), occluded areas (red) and free space (blue). (Right) In the case of real robot experiments, the local coordinate frames have to be obtained from the raw point cloud (cf. (Herzog et al. 2014)). Based on those and the real point cloud, the local shape representation is extracted and the corresponding metric inferred. Therefore allowing to determine possible grasp hypothesis.

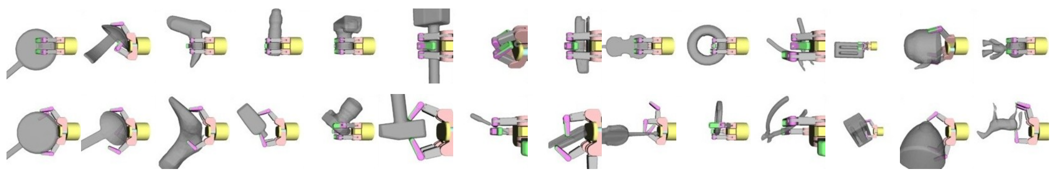


Figure 5.2: Example grasp for instances of a subset of the object categories in the database. The top row shows stable and the bottom row unstable grasps. The categories from left to right: banjo, ladle, boot, bottle, camera, hammer, screwdriver, stapler, violin, tape, pliers, toaster, kettle, and toy.

5.2.1.2 Grasp Configurations

For sampling reference grasps for each object, we use the existing strategy in OpenRAVE (Diankov 2010). Starting from an enclosing bounding box object approximation, rays get cast towards the object from all sides. The intersection points are used as *approach points* and the surface normals as *approach vectors* (see Figure 5.3a). For each approach point, we apply 16 different grasps each with the same preshape but with one of 8 different rotations around the approach vector and one of two possible translation offsets along the approach vector. The first contact between the object and the hand determines the first offset. The second offset backs up the hand 25mm from its position at first contact. In the following, we refer to these offsets as standoffs.

5.2.1.3 Grasp Metrics

As discussed above, defining a suitable grasp quality metric is an open problem in the community. Despite its excellent theoretical properties (Pokorny et al. 2013), several studies (Balasubramanian et al. 2012; Diankov 2010; Weisz et al. 2012) have concluded that the popular ε-metric is not a good grasp stability proxy for real robotic experiments. This metric essentially quantifies grasps that are in force closure which indicates a static equilibrium. However, closure is often wrongly equated with stability that can only be analyzed in a dynamical system (Bicchi et al. 2000). Furthermore, force closure assumes that the manipulator can exert arbitrary forces at the contact points which is never the case in practice. Given this, the results of the above studies are not surprising as the aforementioned assumptions are violated in their experiments. For example, the synthesized grasps are acquired with a real robot and are tested for stability either by lifting (Weisz et al. 2012) or even by vigorous shaking (Balasubramanian et al. 2012). Since all popular grasp simulators implement the ε-metric, it is still very commonly used despite these studies.

Therefore, our database contains this metric as one of a set of grasp quality metrics. Additionally, we evaluate a metric based on the outcome of a physics simulation of a grasp. A common way to more robustly determine whether a grasp is stable or not is to compute a metric averaged over many object pose perturbations (Diankov 2010; Weisz et al. 2012). Summarizing, we include the following metrics in the database: (i) classic ε-metric, (ii) percentage of force-closure grasps for additional object pose perturbed samples (*ε-ratio*), (iii) physics-metric, and (iv) percentage of stable grasps according to the physics-metric of the perturbed samples (*physics-ratio*).

Computation of Raw Metrics

ε-metric Given the robot hand in one of the poses as generated according to Chapter 5.2.1.2, all fingers close simultaneously. As soon as a finger gets into contact with the object surface or with itself, it stops moving. The objects remain static during this process. Then, we compute the ε value with the efficient method proposed in (Pokorny et al. 2013). Given the rubber coating on the fingertips and palm of the real BarrettHand, we assume a conservative friction coefficient of $\mu = 0.4$ for all objects (c.f. $\mu = 1.0$ for rubber on glass). This friction assumption means that most of the objects will be more slippery than in reality and the decision of whether a grasp is in force closure or not is more restrictive.

physics-metric For this metric, we simulate the grasps in OpenRave using the ODE physics engine (*Open Physics Engine* n.d.). Different from the computation of the ε-metric, here the objects move upon contact. We assume the same friction coefficient as for the ε-metric. As a simplification, the objects are assumed to be in free space without gravity acting on it.

Per grasp, the simulation process runs in a loop of maximally M iterations. Each iteration executes the following steps. First, we command all fingers to close simultaneously. Each finger individually stops closing upon collision detection. Then the physics simulation tries to resolve all collisions, i.e., the object moves upon contact with the fingers. If the physics simulation can resolve all collisions, all fingers continue to close simultaneously. Otherwise, the object is randomly perturbed at the collision center, followed by a simulation step to resolve the current collisions. If all attempts to resolve the perturbed instances fail we consider the grasp to be stable according to the physics-metric and unstable otherwise. The magnitude of these random perturbations is an open tuning parameter, as well as the number of perturbations. The latter trades off the required simulation time till a grasp is considered stable and the certainty of the grasp stability.

The physics-metric is more time consuming to compute than the ε-metric. Since we are interested in acquiring high-quality simulation data for *learning*, additional computation requirements only resemble a minor drawback since the offline process does not require any intervention. The physics-metric requires to simulate the time course of a real grasp execution. We, therefore, expect it to be more reliable in predicting grasp success than a static measure. (Popović et al. 2011) and (Rytz et al. 2013) also use the assumption that physics simulation can better predict grasp success. We provide support for this hypothesis by verifying it through crowdsourcing (see Chapter 5.2.2). Another benefit of this metric is that it is much less dependent on the standoff of the grasp relative to the object (see Chapter 5.2.1.4). Furthermore, additional information such as object interactions with, e.g. the simulation process can easily incorporate a table allowing for different grasp executions.

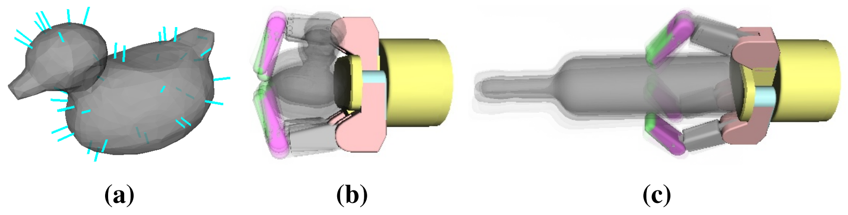


Figure 5.3: Visualization of the grasp generations. (a) All considered approach vectors for a sample object. (b,c) Overlay of 30 noisy samples of object pose relative to one reference grasp for two different objects.

Computation of Ratio Metrics A popular way to enhance simulation-based metrics is to simulate uncertainty in object pose (Diankov 2010; Weisz et al. 2012). The same grasp is applied multiple times on the slightly perturbed object. By taking all resulting grasp quality measure into account, we arrive at a more robust final score.

Generation of Noisy Object Poses In the following, we will refer to each grasp that is generated as described in Chapter 5.2.1.2 as a *reference grasp*. We believe that it is crucial to simulate the perceptual noise which will affect the template extraction as closely as possible. At inference time, for instance, real robot experiments, we only have access to point clouds and therefore need to estimate surface positions and normals from this noisy data source. Therefore, we propose the following strategy to sample 30 perturbations of the objects pose for each reference grasp. We generate a random rotation axis that is placed at the approach point on the object surface. The object is then rotated with a random angle around this axis and with the surface point at the center of the rotation. This process captures estimation errors of the surface normal. Subsequently, a random position is added to the surface pose, reflecting the noisy surface reconstruction. Examples results are visualized in Figure 5.3b and Figure 5.3c.

Computing the Percentage of Stable Grasps Given all the sampled grasps, the following *ratio metrics* indicate the percentage of stable grasps in this set. They differ in the stability criterion, whether one sample is stable or not.

For computing the ε-ratio per reference grasp, we apply a threshold to the *raw* ε value of all the perturbed grasp samples. All grasps above this threshold are considered stable, and all grasps below unstable. For example, using the threshold 0 means that ε-ratio is equal to the percentage of force-closure grasps in the sample set. For computing the physics-ratio, no thresholding on single grasp samples is necessary as the labeling is already binary (see Chapter 5.2.1.3).

Naturally, we prefer grasps that are stable even under some perturbation of the object pose. A reference grasp is therefore considered positive (stable) if 90% of the noisy grasp samples

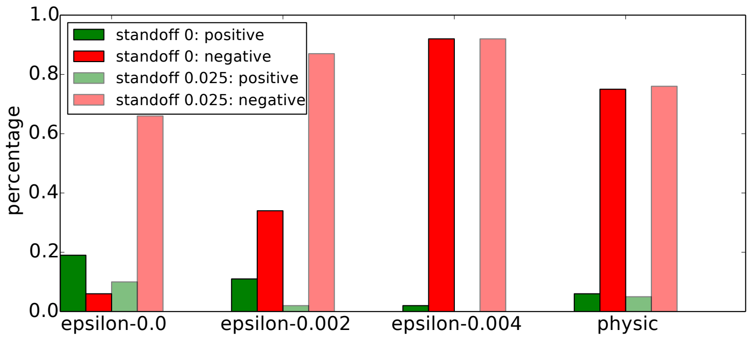


Figure 5.4: Comparison of the distribution of positive (green) and negative (red) reference grasps in the entire dataset when labeled with the two different ratio-metrics: ε-ratio and physics-ratio. A reference grasp is positive if at least 90% of grasp samples are stable. It is negative if less than 10% of the samples are stable. Reference grasps that have intermediate percentages are considered undecided and are not shown here. For ε-ratio we applied three different thresholds on the raw ε-metric (0, 0.002, 0.004) to determine the percentage of stable grasp samples per reference grasp. Additionally, we visualize the influence of the standoff of the hand from the object.

were stable. If the number of stable grasps among the noisy sample set is below 10%, we label this reference grasp as negative (unstable). Note, that this is different from its raw ε-metric or physics-metric.

5.2.1.4 Statistics of the Database

The database contains approximately 300k different grasps. Figure 5.4 shows the distribution of stable and unstable grasps according to the different ratio-metrics. As a first observation, we can see that the dataset is extremely biased towards negative data. This bias mirrors the situation in the real world. Furthermore, for the ε-ratio, there are more positive reference grasps for standoff 0 than for a standoff of 25mm available. Because the object remains static upon contact, the relative pose of robot hand to grasp has much more influence on the ε-metric. The physics-metric is more indifferent to changes of the gripper standoff.

Although a positive ε-metric indicates a force closure grasp, the magnitude of ε quantifies the magnitude of the external wrench it can resist when applying unit force at the contact points. Therefore, we analyzed the impact of increasing the threshold on the raw ε-metric for each sample grasp before computing the ε-ratio per reference grasps. Figure 5.5 shows the histograms over the raw ε-values for all the positive reference grasps. By thresholding the ε-metric of the associated grasp samples at 0 or 0.002, respectively, we obtain these grasp stability histograms. Hereafter, we partition our large-scale simulation-based database into four different datasets

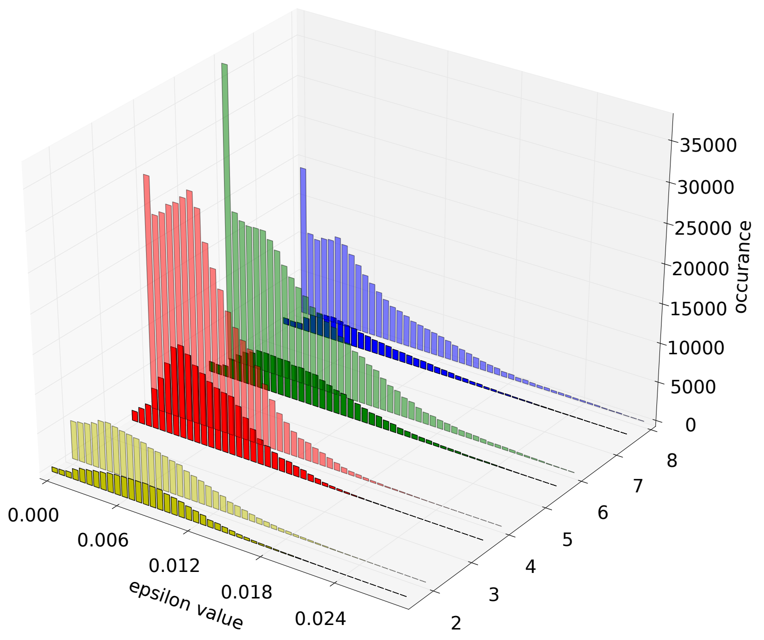


Figure 5.5: Histogram of ε-metric for all positive reference grasps for bottles (yellow), small (red), medium (green), and large objects (blue). ε-metric from reference and sample grasps are included in the plot. The solid colors histograms are generated after thresholding the ε-metric of sample grasps at 0.002 and 0.0 otherwise.

according to object size, namely small, medium and large objects. A toy dataset, containing 63 bottles which is analyzed more extensively. For the majority of sample grasps the force-closure value is close to zero or smaller. Therefore, even thresholding them with a minimal value leads to a huge reduction of positive grasps that also confirmed by Figure 5.4. Some qualitative results on the effect of different epsilon thresholds on the sample grasps and differences between the ε-ratio and physics-ratio are shown in Figure 5.6. It is certainly beneficial to filter sample grasps with a very small ε-metric. However, we can also observe that it is hard to use the ε-metric for separating positive and negative grasps. Moreover, it is even harder to do that for two different objects simultaneously even when they are of the same category. Another interesting result is that only a small fraction of the stable grasps according to physics-ratio are in force closure. Our website at (**Kappler** et al. 2015a) lists additional information about the database.

5.2.2 Using Crowdsourcing to Verify Grasp Metrics

We analyze the different simulation-based metrics in a crowdsourcing setup. The website (**Kappler** et al. 2015a) shows an example page of this setup and additional information. Assuming that humans are optimal predictors of grasp success in the real world, we aim to verify that the

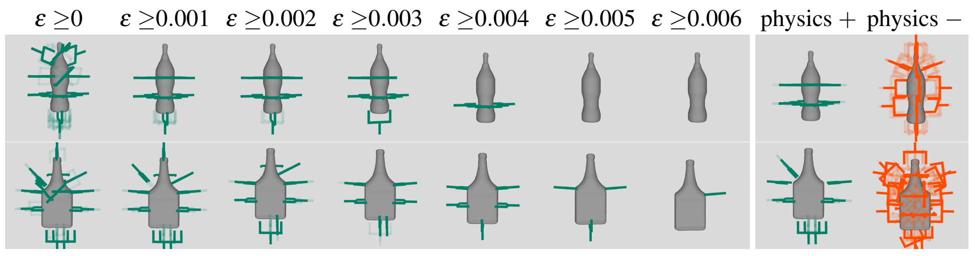


Figure 5.6: Grasp examples for two bottles and the two different metrics. Green Grippers: Positive Examples, i.e., 90% of the associated grasp samples are stable. Red Grippers: Negative Examples, i.e., less than 10% of the associated grasp samples were stable. Left: Visualization of the influence of a varying ε-threshold on the percentage of force closure grasps from the samples. $\varepsilon \geq 0$ means that all force-closure grasps are considered stable while $\varepsilon \geq 0.006$ means that they have to reach this minimum quality. We can observe that a threshold of ε does not easily separate stable from unstable grasps. Even for objects in the same category, it is hard to find one threshold that fits all object instances.

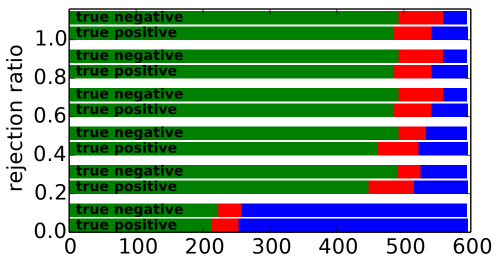
grasp success labels assigned by these metrics match real-world expectations. If this is the case, we can reduce the number of expensive real robot experiments for gathering high-quality labeled data for grasping. Furthermore, we would also not require any additional human labels as the metrics would be deemed reliable.

5.2.2.1 Setup

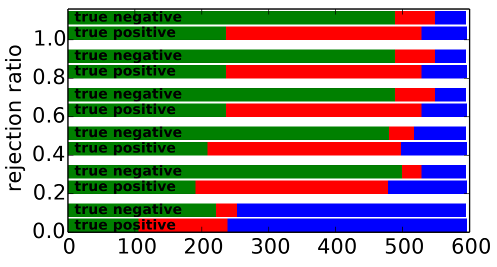
For all crowdsourcing experiments, the force closure threshold of the ε-metric is assumed to be 0.0. Thus any grasp in force closure is considered to be stable for the computation of the ε-ratio.

For the toy problem (bottles), **all** pictures of the positive ($r \geq 0.9$) and negative ($r \leq 0.1$) reference grasps according to both ε-metric and physics-metric are rendered in simulation, where r denotes the corresponding ratio-metric. For the small, medium, and large dataset, we subsample the data, rendering at most 200 positive ($r \geq 0.9$) and negative ($r = 0$) reference grasps, again according to both ratio-metrics. Each image shows one grasp from four different viewpoints and is similar to those in Fig 5.2.

Each Mechanical Turk hit contains 20 such images to label them as either stable, undecided or unstable. Although we assume humans to be optimal predictors of grasp success, we still need to filter noise in the labels due to, e.g. ambiguous pictures, accidental wrong clicks or spamming. We, therefore, labeled 3 of these 20 images ourselves before publishing them on Mechanical Turk and used them for post-processing. The remaining 17 images are drawn from both ratio-based



(a)



(b)

Figure 5.8: The human ground truth labels for bottles assigned to grasps selected by the (a) physics-metric and (b) ε-metric. A rejection ratio of 0.0 means that all labels of a worker with a single mistake are ignored and 1.0 means that any mistake according to the pre-labeled data is ignored. The x-axis denotes the absolute number of grasps with labels. In green we show consistent labels from both humans and metrics, red means human have selected the opposite outcome, and blue reflects undecided.

metrics such that on average one hit contains an equal amount of reference grasps labeled as stable and unstable according to the metrics.

One of the three pre-labeled examples is an obviously unstable grasp which is used to block workers from further experiments if they mislabel it, thus prevent spamming. Furthermore, if a worker makes a mistake while labeling this particular grasp image, the hit is regenerated such that five valid results are obtained per hit. The other two labeled examples are always visible during the labeling task and are part of the 20 samples for human labeling.

5.2.2.2 Analysis

First, worker performance is evaluated based on the aforementioned pre-labeled validation dataset. Figure 5.8 shows the label correlation for varying rejection ratios for the toy dataset. The rejection ratio states how many mistakes on the pre-labeled data are allowed per worker until their labels are discarded from the analysis. Green signifies an agreement between human labels and the labels according to the metric. Red means disagreement and blue undecided. This result shows that many workers make only a small number of mistakes. Already a rejection ratio of 0.3 results in inconsistent data.

Furthermore, we can observe that the correlation between the physics-ratio and human labels is always similar for positive and negative grasps. However, for the ε-ratio, there is a significantly lower correlation to human labels for positively labeled grasps. Negative grasps show a similar correlation as for the physics-metric. Having large amounts of false positives in the dataset is of course not desirable. A model that is learned from this data would require more real-world experimental data, which is hard to acquire in large quantities, to correct for the high false positive rate. A similar outcome can also be observed for the small, medium and large object dataset in Figure 5.12, although less pronounced. The human labeled data supports the hypothesis that the proposed physics-metric is indeed a better proxy to determine whether or not a grasp is stable.

5.3 Large-scale Grasp Prediction as Classification

Now that we have a large-scale database with ground truth object grasp success and sensor data annotations we want to analyze if a state-of-the-art classification based method benefits from the large variety of objects. Therefore, we compare a deep neural network (see Chapter 2.2.2.4) with logistic regression (see Chapter 2.2.2.3) functioning as a baseline. This comparison requires to choose a representation of the input data that relates a grasp to an object. We are particularly interested in a representation that can be easily extracted from real-world sensory data, requires no prior semantic knowledge (object identity or category) and is local, i.e., independent of accurate segmentation. In the following, we discuss our modifications to the feature representation proposed in (Herzog et al. 2014). We want to stress that this feature representation is particularly well suited since it is very task-specific and takes both training and inference requirements into account. However, other feature representations might as well be extracted and added to our database. Figure 5.1 (right) shows an overview of the system for the prediction of grasp success.

5.3.1 Local Shape Representations

For a given grasp, our feature extraction algorithm extracts a tangent plane at the intersection point between the approach vector of the hand and the object. The local shape representation (or template) is a grid of a certain resolution that is aligned with this plane. If the intersection point with this plane was in the view of the camera and thus the point cloud sufficiently covers the area around the intersection point, we project the point cloud onto the aforementioned grid. Each grid cell onto which a point gets projected will carry the distance to that point as a value. All the other grid cells can either be labeled as occlusion (with a height value that depends on the viewing angle) or as free space (with a constant height value). These three different channels are visualized in Figure 5.1 (representation box).

In contrast to the original version in (Herzog et al. 2014), the template coordinate frame is always aligned with the surface normal and not with the potentially different approach vector of the robot hand. Thereby, template extraction becomes more efficient because there is only one template per surface point and point cloud. This reduces the overall processing time to enable responsive

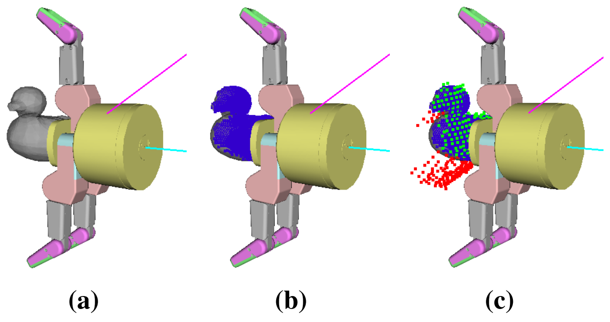


Figure 5.9: Local shape representations for a rubber duck, a specific grasp pose and camera viewpoint. (a) Cyan line indicates the surface normal of a point on the object model. It forms the approach direction of the grasp. The pink line indicates the viewing direction of the camera. (b) Blue dots indicate the point cloud for the object rendered from the given viewpoint. (c) Green (surface) and red dots (occlusion) form the local representation which is a height map relative to the plane that defined by the point on the surface and its normal (cyan line). Green dots are generated by measuring the height of the cloud relative to the plane. Red dots indicate occluded whose height value depends on the viewing angle (pink line).

grasping on a real robotic system. Depending on the *roll* of the robot hand around the surface normal, we rotate the template to allow the classifier to disambiguate different rotations. The lack of label requirements for a supporting plane or background is another simplification. We assume that every solid point is assumed to be part of the object surface. The burden of separating the object surface from the background is now on the classifier. A visualization of a template with the corresponding grasp is shown in Figure 5.9.

For each grasp, we compute several templates from different viewpoints. These are based on a set of point clouds that are generated per object from 64 different regularly-placed viewpoints. Given one grasp and the associated surface normal, we compute templates only based on the subset of point clouds with an angle between viewpoint and surface normal that is less than 20 degrees. This is visualized in Figure 5.10. For generating these templates for the objects in the database, we consider two different kinds of simulated sensors. One is computing the ground truth noise-free depth map using OpenGL. The other is simulating measurement from an RGB-D camera including occlusion boundaries, quantization artifacts and Perlin noise (Bohg et al. 2014b). When extracting this representation from real-world data, the surface normal of a point in the point cloud has to be computed based on its neighboring points. Once this is achieved, the template computation proceeds as above.

5.3.2 Learning Methods

A key observation about the proposed feature representation is that it can be interpreted as a 2D image representation for which different channels have different semantic meanings. Deep neural networks are the state-of-the-art learning approach on large datasets with image like data. A key aspect for these methods is that they do not require data pre-processing but rather extract the feature transformations in a data-driven manner. Inspired by the successful applications of proxy objectives such as unsupervised (Erhan et al. 2010) and supervised pre-training (Collobert et al. 2008), we analyze the possibility to exploit the synthetically generated labeled datasets as initialization and regularizer for real-world data for grasping. Therefore we train a simple logistic classifier (Chapter 2.2.2.3) as a baseline learning method. This approach provides further insights into the overall data complexity since it only optimizes a linear mapping from the feature representation to a scalar output. (Herzog et al. 2014) showed promising results in previous work using this classifier on smaller datasets, both in terms of objects and data samples. The logistic classifier in (Herzog et al. 2014) was manually optimized as part of a non-parametric classifier which does not scale to our dataset size. We compare this method to a deep neural network, LeNet like architecture (Lecun et al. 1998), a highly parametric non-linear function approximator. We chose this architecture since it exploits 2D image like structures while being very efficient to evaluate.

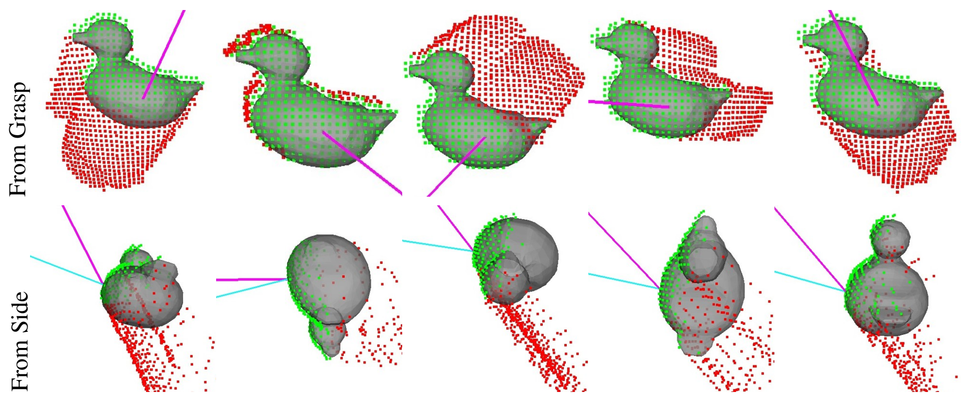


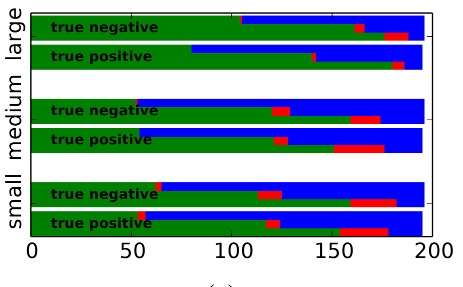
Figure 5.10: Influence of the viewpoint on the local shape representation. All images show these representations for the same grasp (cyan line = approach direction) but with a different viewpoint (pink line). The top row shows the heightmap from the grasp direction while the bottom row shows the same heightmap from the side. While the surface points stay largely the same, the occlusion area changes drastically.

5.3.3 Experiments

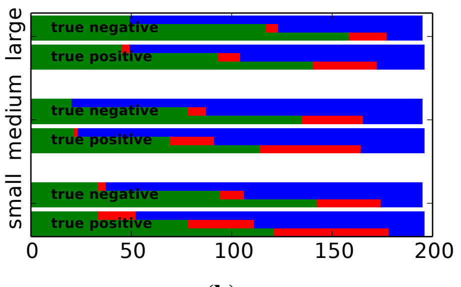
Our experiments, presented hereafter, demonstrate that a simple convolutional neural network (CNN), representing the state-of-the-art deep learning architecture for computer vision, can be used to learn a mapping from our feature representation to grasp success, better leveraging the dataset size. The CNN model applied in our experiments can be seen as a logistic classifier which learns an additional non-linear feature mapping from data. The output of each method can be interpreted as a certainty that the given grasp is stable (close to 1) or unstable (close to 0). A threshold on this certainty can be used to achieve different rates of true and false positives (Figure 5.13).

The dataset for this experiment contains eight different rolls of the hand around every approach direction. This structure could be used to split the problem into eight different learning tasks. However, these eight different rolls can be directly represented in the template by simple rotation. Thus, a single learning method of appropriate complexity should be able to distinguish these rolls. Our feature representation cannot disambiguate different grasp standoffs. Thus we learn a different classifier for these cases.

The classification performance is reported on each dataset, using a binary threshold on the ratio-metric of choice at 0.9. For the ε-ratio, we threshold the raw ε at 0.002 to obtain a comparable data distribution for both ratio metrics. Each of our datasets contains randomly sampled objects within the data group (bottle, small, medium, large, all) if more than one object per category is available. Otherwise, it is only added to the training dataset. We exclude all objects that result in point clouds with less than 30 points, yielding more than 600 remaining object instances. Data from one object is exclusive to either the training, validation or testing dataset. Two thirds of the data are used for training and validation data and one third for reporting the results in Figure 5.13, 5.14, and 5.15. The validation dataset is used to select the best-learned predictor and tune hyperparameters such that the network convergences. We always report results on the test dataset, composed out of unique object not seen during training and validation. Although the majority of possible grasps, both in simulation and reality are unstable, resulting in a very biased dataset, we conducted experiments with balanced datasets to allow comparison with prior work, e.g., (Lenz et al. 2014). However, learning on balanced datasets improves performance but cannot be applied to the real unbalanced data distribution. We believe that in robotic grasping, it is important to achieve a high true positive rate at a low false positive rate. This tradeoff lowers the risk of applying unsuccessful grasps. Therefore, we report the ROC-curve up to 20% false positive rate. We want to stress that this analysis is not about the absolute performance of the deep learning method. We rather want to demonstrate that the data is too complex for linear methods (Figure 5.15b) and that the data sample size is within a domain which is reasonable for frameworks such as deep learning. One insight from the learning performance is that the physics-



(a)



(b)

Figure 5.12: The ground truth labels for the three object groups (small,medium,large) assigned to grasps selected by (a) physics-metric and (b) ε-metric, assuming a rejection ratio of 0.3. For every result we report the correlation with the human labels for which at least > 2 (bottom), > 3 (middle), and > 4 (top) labeled the grasp accordingly. In all cases the proposed physics-metric outperforms the ε-metric. The x axis denotes the absolute number of grasps with labels. In green we show consistent labels from both humans and metrics, red means human have selected the opposite outcome and blue reflects undecided.

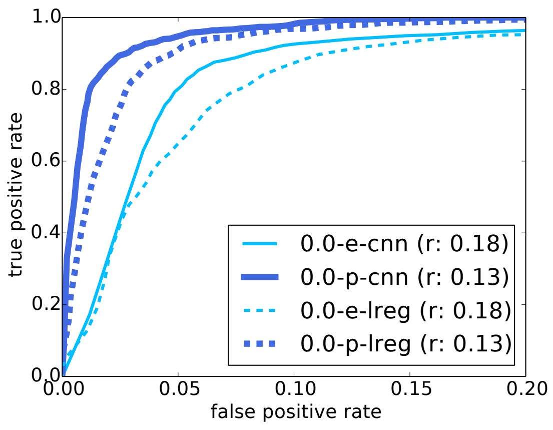


Figure 5.13: ROC curve for the bottle dataset (standoffs 0; results for standoff 25mm are reported online) Labels based on ε-ratio (e) and physics-ratio (p) for CNN and logistic classification (lreg). CNN based prediction always outperforms lreg and physics-ratio outperforms ε-ratio based datasets. The data ratio (r) for each dataset suggests that despite the stronger negative bias of the physics-ratio based dataset it is still more consistent and thus easier to optimize for.

metric-based results are almost always better than the one based on the ε-metric (Figure 5.15b), even though the physics-metric-based datasets are more biased and thus more challenging for classification. Since the presented data is strongly biased towards predicting a negative outcome, we believe that this interesting dataset can also result in new algorithms within the deep learning framework.

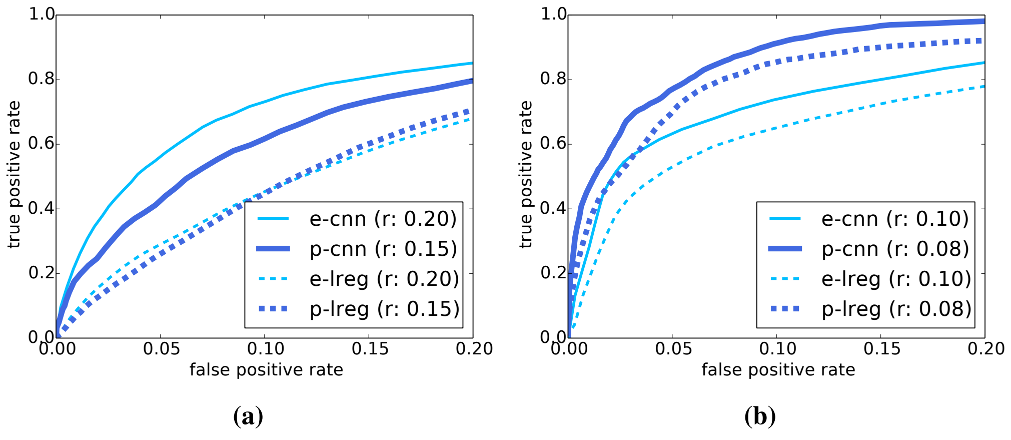


Figure 5.14: Results for small objects (a) and medium objects (b) illustrate that a linear logistic classifier cannot cope with the data complexity.

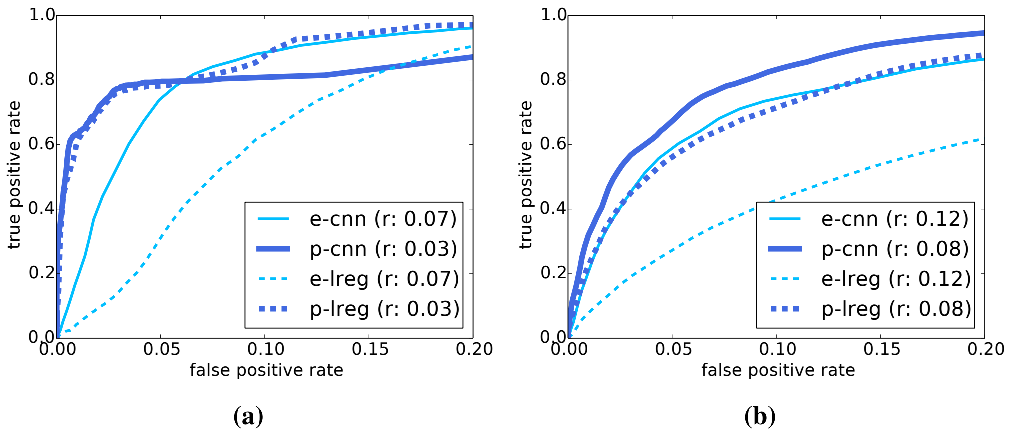


Figure 5.15: Results for large objects (a) and all objects (b) illustrate that a linear logistic classifier cannot cope with the data complexity. In addition to that, the presented result supports the hypothesis that the physics-ratio generates more consistent data which is in term easier to learn.

5.4 Large-scale Grasp Prediction as Top-1 Prediction

So far we have discussed traditional and state-of-the-art large-scale *learning-based* approaches for grasp prediction. Supervised classification formulations such as logistic regression and support vector machines (SVMs) are common methods for small grasping datasets and deep neural networks show promising results for large-scale datasets. Given that robot grasping is actually concerned with executing a single grasp successfully, but supervised classification methods are optimized to create binary labels, typically a score, e.g., distance to the decision boundary, has to be used as a proxy to select the most promising grasp for execution. This is problematic since the underlying classification model is optimized for accurately predicting the binary labels of the entire training dataset. While for some subsets of data points separating positives from negatives may be easy to achieve, it generally can be tough to achieve this separation for all data points. Attempting this separation is mainly a problem when training on datasets with noisy labels or when the employed feature representation cannot disambiguate different hypotheses in order to make a decision.

Here, we argue that for grasping, we should use our *domain-specific knowledge* and train models on subsets of data, where one subset may, for instance, represent all possible grasp hypotheses obtained from one viewpoint of the object. Furthermore, we should optimize an objective that encourages that the highest scoring training data point of such a set represents a positive example. For example, when considering a partial, segmented point cloud of an object, there exists a large set of potential grasps, most of which are not stable. The best scoring hypothesis within this set should correspond to a stable grasp. Such an objective is called a *ranking loss*. Thus far, only a few grasp learning models in the literature consider this kind of objective.

Although similar to multiple instance learning (MIL), we derive a novel ranking formulation for grasp stability prediction for binary labeled data hereafter. There are three main differences between our problem formulation to typical ranking problems. First, our hypothesis set consists only of binary data. Hence there is no inherent ranking between different examples other than the distinction between positive and negative hypotheses. Second, we want to optimize solely for the top-1 ranked hypothesis in a set of hypotheses, and we are not interested in the remaining order of hypotheses. Third, our resulting ranking score can also be interpreted as a score for classification, deciding whether or not the top-1 ranked hypothesis is a positive or negative one.

5.4.1 Classification Versus Ranking for Top-1 Prediction

To the best of our knowledge, data-driven learning methods for grasp planning are almost exclusively formulated as classification problems, often from an optimization perspective (see

Chapter 2.2.1.3).

We argue that for grasp planning, classification is a sub-optimal objective. The canonical problem for grasp planning is to predict a successful grasp for a target object given an entire set of hypotheses. Such a hypothesis set can, for example, contain all possible grasp rectangles for a view of an object as provided by the Cornell dataset (Lenz et al. 2014) or all possible templates extracted from a 3D point-cloud as in our previously discussed database (Chapter 5.2.1). Obtaining all possible successful grasps is not necessary for robot grasping. Ideally, the robot succeeds on the first grasp attempt. Therefore, the grasp stability prediction should be reformulated as a ranking problem, trying to robustly identify **one** successful grasp within each hypothesis set. Additionally, it should provide a calibrated score to decide whether the best hypothesis is stable or not.

Similar to max-margin multiple instance learning (Andrews et al. 2002) we can formulate our problem as a structure prediction (Chapter 2.2.1.7) ranking objective. Figure 5.16b illustrates the canonical pair-wise ranking problem for different hypothesis sets (x,y) and $(x,y)'$, formulated as a standard max-margin classification problem, defined as

$$\begin{aligned}\min_{\boldsymbol{\theta}} \quad & \mathcal{R}(\boldsymbol{\theta}) + \\ & \sum_{(x,y)} \sum_{(x,y)'} L(F(x;\boldsymbol{\theta}) - F(x';\boldsymbol{\theta}), \Delta(y,y')). \end{aligned} \tag{5.1}$$

The main difference to the standard classification problem is the pair-wise classification and the pair-wise loss Δ. Notice this optimization problem can be specialized to the ranking SVM formulation proposed in (Joachims 2002) by applying the appropriate max-margin loss L. As shown in Figure 5.16b, the ranking problem as described in Equation 5.1 is concerned with ordering examples from different hypothesis sets according to the Δ loss. For typical grasp datasets, consisting of binary labeled grasp hypotheses, this ranking formulation would result in a solution similar to the binary classification problem (Equation 2.24) up to a hypothesis set dependent scaling and offset. The scaling and offset are necessary since the ranking formulation results in a relative ordering problem. Hereafter we derive a ranking formulation for binary hypothesis sets that allows top-1 prediction within the given hypothesis set as well as classification of that top-1 choice. We further propose a method to optimize this formulation within the standard stochastic gradient descent optimization framework.

5.4.2 Top-1 Ranking

Top-1 ranking addresses the problem of optimizing a function that predicts **one** possible successful grasp within any given hypothesis set, if the hypothesis set contains **at least one** *stable*

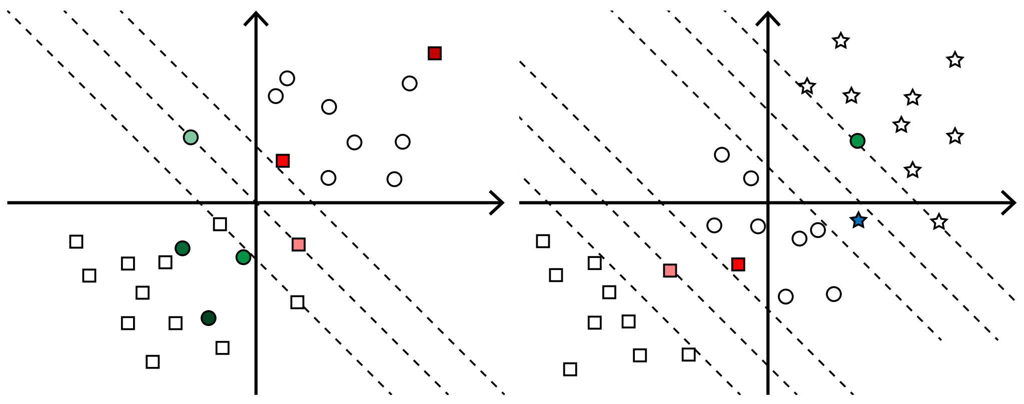


Figure 5.16: We illustrate the difference between the standard (a) max-margin classification problem and (b) pair-wise max-margin ranking problem. All symbols of the same shape are within the same hypothesis set. (a) Binary classification aims at separating these two sets. The magnitude of the error is indicated by the color saturation of the data samples where white means no error. Each set has its own distinct color. The (b) ranking problem attempts to not only separate the three sets, but also maintains an order such that stars are always further to the top right than circles, and circles are further top right than squares. The resulting pair-wise classification problems illustrate the similarity of the ranking problem to the standard classification problem in (a).

grasp. In addition to that, the resulting score has to be discriminative to classify whether or not the best predicted hypothesis is *positive* ($y = 1$) or *negative* ($y = -1$). In our work, hypothesis sets only contain binary labeled grasps, meaning a grasp is either considered *positive* (stable) or *negative* (unstable). We assume no additional label information for the data which would allow to further discriminate between different examples in a set, e.g., if one *positive* grasp is better than another *positive* grasp. Our large-scale grasp database is one concrete example of a collection of hypothesis sets for which this formulation is ideal. In this particular example, every partially observed object is associated with a point cloud and several labeled grasp templates, where the grasp template takes the role of the feature representation x and the binary labels y indicate a stable or unstable grasp. Thus in this setting, a hypothesis set contains all pairs (x, y) available for a particular object view.

Every hypothesis set can either contain only positive examples, or only negative examples or both. To simplify notation we introduce three different index sets:

- $\mathcal{I}^+$ refers to all sets with **only** *positive* examples,
- $\mathcal{I}^-$ refers to all sets with **at least one** *negative* example,
- $\mathcal{I}^{+-}$ refers to all sets with **at least one** *positive* **and** *negative* example.

Every hypothesis set is assigned to at least one index set. Hypothesis sets with positive and negative examples are assigned to both $\mathcal{I}^{+-}$ and $\mathcal{I}^{-}$.

In the following we re-formulate and adapt the general ranking problem (Equation 5.1) to the top-1 grasp prediction problem. We use a max-margin formulation with a margin ($t = 1$)

$$l(t - yk), \tag{5.2}$$

both for classification ($k = F(x;\boldsymbol{\theta})$) and ranking ($k = F(x;\boldsymbol{\theta}) - F(x';\boldsymbol{\theta})$). Here, we use the squared hinge loss $l(v) = \frac{1}{2}\max(0,v)^2$. By introducing the additional square, the loss becomes differentiable everywhere, a property that has been proven useful for stochastic gradient descent based neural network optimization (Tang 2013b).

Our proposed loss function is comprised of three parts – $L^{+-}(\boldsymbol{\theta})$, $L^{+}(\boldsymbol{\theta})$ and $L^{-}(\boldsymbol{\theta})$ operating on the previously introduced index sets $\mathcal{I}^{+-}$, $\mathcal{I}^{+}$ and $\mathcal{I}^{-}$, respectively. The goal of the first part of our loss, $L^{+-}(\boldsymbol{\theta})$, is to rank positive and negative hypotheses using a max-margin formulation, and is given as:

$$\begin{aligned} L^{+-}(\boldsymbol{\theta}) = \sum_{i \in \mathcal{I}^{+-}} \sum_{x_i^+} \Big[& l(1 - F(x_i^+;\boldsymbol{\theta})) + \\ & \sum_{x_i^-} l(1 - (F(x_i^+;\boldsymbol{\theta}) - F(x_i^-;\boldsymbol{\theta}))\Big] \end{aligned}$$

where x_i^- represents all negative and x_i^+ all positive hypotheses in the corresponding hypothesis sets in $\mathcal{I}^{+-}$. Notice, we obtain $l(1 - (F(x_i^+;\boldsymbol{\theta}) - F(x_i^-;\boldsymbol{\theta}))$ from Equation 5.1, using equation 5.2 with $t = \Delta(y,y') = 1$ if y is a positive and y' a negative hypothesis. Furthermore, we ensure that positive examples get a calibrated score by adding the max-margin formulation $l(1 - F(x_i^+;\boldsymbol{\theta}))$ for positive examples.

In the case of separable data, we can rewrite L^{+-} to

$$\begin{aligned} L^{+-} = \sum_{i \in \mathcal{I}^{+-}} \sum_{x_i^+} \Big[& l(1 - (F(x_i^+;\boldsymbol{\theta}) - \max_{x_i^-} F(x_i^-;\boldsymbol{\theta})) + \\ & l(1 - F(x_i^+;\boldsymbol{\theta}))\Big] \end{aligned} \tag{5.3}$$

If the data is separable, summing over all negative examples (as done in initial $L^{+-}(\boldsymbol{\theta})$) will result in the same loss value, as this max formulation. The second part of our loss, $L^{+}(\boldsymbol{\theta})$, operating on index set $\mathcal{I}^{+}(\boldsymbol{\theta})$, ensures that the prediction scores are calibrated in the same manner as positive examples in $\mathcal{I}^{+-}$, again by using the max-margin formulation:

$$L^{+}(\boldsymbol{\theta}) = \sum_{i \in \mathcal{I}^{+}} \sum_{x_i^{+}} l(1 - F(x_i^{+};\boldsymbol{\theta}))$$

The third component $L^{-}(\boldsymbol{\theta})$, establishes that negative examples in the index set $\mathcal{I}^{-}$ are separated from positive ones to ensure the overall calibration of the score such that the final ranking score for a hypothesis set can be used for classification:

$$L^{-}(\boldsymbol{\theta}) = \sum_{i \in \mathcal{I}^{-}} \sum_{x_i^{-}} l(1 + F(x_i^{-};\boldsymbol{\theta}))$$

Finally, we obtain the joint ranking and classification loss formulation

$$\min_{\boldsymbol{\theta}} \left[L^{+-}(\boldsymbol{\theta}) + L^{+}(\boldsymbol{\theta}) + L^{-}(\boldsymbol{\theta}) \right] \tag{5.4}$$

If our binary labeled training data, organized in hypothesis sets, is perfectly separable, this formulation results in the same solution as the standard max-margin classification problem (Equation 2.24). The pair-wise terms in equation 5.3 will vanish as soon as the two classes are perfectly separated. If the dataset is not separable, the pair-wise term will function as an additional loss on all positive examples within hypothesis sets for which the ranking loss cannot be fulfilled. This can be interpreted as a difference in the importance of positive and negative misclassifications. However, this does not resolve the issue that the top-1 prediction might be a negative example. Figure 5.16a illustrates such a case. The reason for misclassification might be the similarity to a positive example within a different hypothesis set. Hence, the perfect order/separation is still not achievable.

More concretely, let us assume that there exists a negative grasp which has an indistinguishable feature representation from several positive grasps in multiple sets. In this case, multiple failure cases can occur. If this particular (negative) hypothesis is in the same hypothesis set as the indistinguishable positive hypotheses, the negative hypothesis can be picked at random. The reason for this is that the negative hypothesis achieves the same score as the positive hypotheses and one hypothesis has to be selected based on this score. Another possibility is that this negative hypothesis is in a different hypothesis set than the indistinguishable positive hypothesis and no *easy* positive example exists for the function approximator $F(x;\boldsymbol{\theta})$ in the hypothesis set containing the negative hypothesis. Thus, this negative hypothesis will achieve the highest score.

In the following, we present our approach to obtaining a top-1 ranking problem despite the binary nature of the hypothesis sets. Since there are no label differences within the set of positive or negative hypotheses, we propose to use the induced difference by the function approximator itself. Thus, while optimizing the function approximator, the currently best positive and negative example, given the current function approximator prediction, is used for the pair-wise loss, resulting in:

$$\begin{aligned}\min_{\boldsymbol{\theta}} \sum_{i\in\mathcal{I}^{+-}} \big[& l(1-(\max_{x_i^+} F(x_i^+;\boldsymbol{\theta}) - \max_{x_i^-} F(x_i^-;\boldsymbol{\theta}))+ \\ & l(1-\max_{x_i^+} F(x_i^+;\boldsymbol{\theta}))\big]+ \\ \sum_{i\in\mathcal{I}^{+}} & l(1-\max_{x_i^+} F(x_i^+;\boldsymbol{\theta}))+ \\ \sum_{i\in\mathcal{I}^{-}} & \sum_{x_i^-} l(1+(F(x_i^-;\boldsymbol{\theta})) \end{aligned} \tag{5.5}$$

Figure 5.17 shows an example why this simple change to the optimization objective does achieve the top-1 ranking property for binary datasets. Intuitively, our formulation does not penalize any prediction for positive examples except for the current best positive and negative one in each hypothesis set. The best examples are determined by the current ranking of the latest function approximator parameterization. This ranking is not optimized by an explicit supervised quantity, but it rather reflects the difficulty for the function approximator to distinguish positive from negative hypotheses. Hence, the function approximator can select one positive example in each hypothesis set, which contains at least one positive example, which is easy to separate from all negative examples. This change enables our formulation to ignore negative examples which are indistinguishable from positive hypotheses, as long as there exists at least one other positive hypothesis which is distinguishable. Notice, that we do not select these positive examples, but the optimization itself will determine these examples. Different learning methods for $F(x;\boldsymbol{\theta})$ might result in different top-1 candidates.

This problem formulation enables automatic selection of positive top-1 examples which are easy to separate from negative examples. Indistinguishable examples under the implicit function approximator similarity measure, existing, e.g. in different hypothesis sets, are not enforced to obtain a positive score anymore.

To be more concrete, this behavior is useful for positive hypotheses for which relevant information, e.g., the surface points of an object, is not available, due to, e.g. partial occlusion. In this scenario, the feature representation for the hypothesis might not contain enough information to distinguish this example from other negative examples. Using our ranking formulation (Equation 5.5), the function approximator is not penalized if it assigns a low score to such examples, as long as there is another positive hypothesis in the set, for which the feature representation contains enough information to separate this example from all negative ones.

The pair-wise loss, solely applied to the two currently maximum examples of different classes can be interpreted as a virtual target for the positive example. Alternatively, the pair-wise loss can be seen as a ranking problem on precisely two hypotheses (highest scoring positive and negative), selected by the score of the function approximator. The optimization tries to increase the score of that particular positive example to outperform the best negative one by a fixed margin (Equation 5.2 and 5.5).

For each hypothesis set, we have to solve at most two different problems. For hypothesis sets in $\mathcal{I}^{+-}$, the pair-wise loss and negative calibration are optimized. For hypothesis sets in $\mathcal{I}^{+}$, the best positive example is calibrated and for hypothesis sets in $\mathcal{I}^{-}$ all negative examples are calibrated. Despite the simple nature of these problems, obtaining an efficient optimization of Equation 5.5 is not straightforward as discussed in the following section.

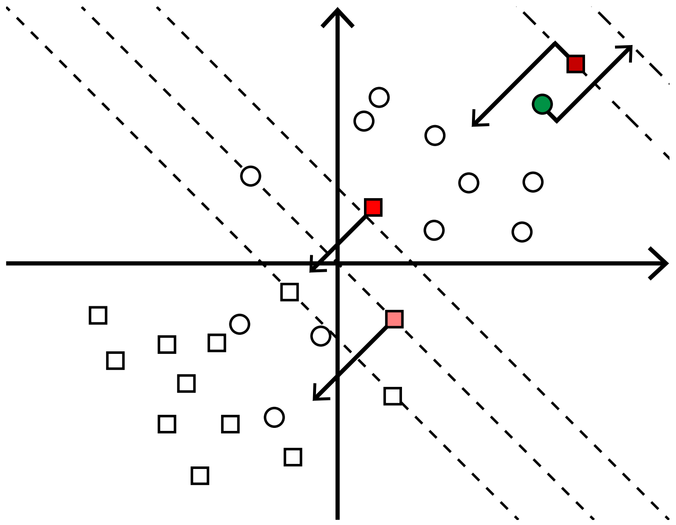


Figure 5.17: This figure illustrates the proposed ranking objective applied to a single binary set of hypotheses. Squares represent negative examples and circles positive ones. The saturation of the color filling the shapes represents the error magnitude for each sample. The three dashed lines through zero represent the standard hinge loss. Notice that positive examples (circles) are not enforced to be separated but negative (squares) are. Since the current best hypothesis is a negative example, an additional classification problem for the best positive hypothesis is created, creating a *virtual* target higher than the current best negative example plus a margin. Arrows indicate the direction in which the optimization objective attempts to change the prediction scores.

5.4.3 Efficient First Order Optimization

The naive problem formulation as proposed in Equation 5.5 could be optimized with first order batch gradient descent. Different from MIL (Andrews et al. 2002) our formulation is not convex since we are using a deep neural network based function approximator. The standard approach to optimizing a loss in the form of equation 5.5 for large datasets is to use mini-batch stochastic gradient descent. This makes each optimization step independent of the total number of available data points. Current state-of-the-art approaches such as CNN's, which can exploit large datasets due to a large number of open parameters, also follow this optimization scheme. Usually, n data points (x, y) are sampled uniform at random from the training dataset, constructing one mini-batch.

For our proposed loss, every mini-batch has to contain all positive examples of a hypothesis set due to the max operation. Notice this is only restricted to the positive examples. Using any subset of the negative examples which is already fulfilled would merely result in zero loss for the pair-wise terms. Thus the naïve approach for our loss would be to sample a hypothesis set uniform at random. All positive hypotheses of this set have to be in the mini-batch together with any subset of negative hypotheses. This process is continued until the mini-batch is filled with samples.

This naïve approach to constructing the mini-batches for stochastic gradient descent has two main drawbacks. First, the number of positive examples would put a lower bound on the mini-batch size. Second, the majority of the computation would result in no improvement, since only the largest positive and negative example will be affected. In the following, we present our approach to overcome the limitations of the naïve approach.

5.4.3.1 Pair-Wise Loss Relaxation

As pointed out before, the max operation in the pair-wise term of our ranking loss equation 5.5, is the limiting factor to draw individual samples from each hypothesis set. Thus, next, we show how to address this issue such that we can use stochastic gradient descent effectively.

Typical state-of-the-art methods for classification and regression such as *(Convolutional) Neural Networks* are global function approximators. Hence, every update of $F(x;\boldsymbol{\theta})$ can affect the prediction of any other data sample. We assume that $F(x;\boldsymbol{\theta})$ changes slowly for not affected values and more so for values for which gradients are applied. This is not a very restrictive assumption since we use stochastic gradient descent which requires to take small steps to converge. Using this assumption, we can exploit that the $\max_{x_i} F(x_i;\boldsymbol{\theta})$ within a hypothesis set is unlikely to change very frequently. Thus, we propose to rewrite the pair-wise term as two max-margin classification problems with a hypothesis set dependent margin t_i:

$$\begin{aligned}\min_{\boldsymbol{\theta}} \sum_{i\in\mathcal{I}^{+-}} \big[&l(t_i^+ - \max_{x_i^+} F(x_i^+;\boldsymbol{\theta}))+\\ &l(t_i^- + \max_{x_i^-} F(x_i^-;\boldsymbol{\theta}))+\\ &l(1 - \max_{x_i^+} F(x_i^+;\boldsymbol{\theta}))\big]+\\ \sum_{i\in\mathcal{I}^{+}} &l(1 - \max_{x_i^+} F(x_i^+;\boldsymbol{\theta}))+\\ \sum_{i\in\mathcal{I}^{-}} &\sum_{x_i^-} l(1 + F(x_i^-;\boldsymbol{\theta}))\end{aligned} \tag{5.6}$$

where $t_i^+ = 1 + \max_{x_i^-} F(x_i^-;\boldsymbol{\theta})$ is computed for each hypothesis set, as well as $t_i^- = 1 - \max_{x_i^+} F(x_i^+;\boldsymbol{\theta})$. The basic idea is to fix the maximum positive hypothesis for one hypothesis set to compute the corresponding margin for the negative hypothesis and vice versa. Instead of always evaluating the function approximator to obtain the true t_i, the last known prediction for every sample is used to update the estimates. This optimization problem will result in the same minimum as equation 5.5, if our assumption, that the maximum hypothesis for a particular hypothesis set does not change frequently, holds. Now, it is possible to draw individual samples from each hypothesis set.

Note, the most informative examples are the best positive and negative examples. Other positive examples of a hypothesis set in $\mathcal{I}^{+-}$ do not contribute to the loss Equation 5.6. Thus, to improve the loss the sample distribution over the hypothesis and hypothesis sets is not uniform but dependent on the loss and an additional term described in the following section.

5.4.3.2 Loss Optimization using Sampling

Random data sample selection is crucial for stochastic gradient descent based optimization. Selecting data which most likely results in zero loss, thus zero gradients, slow down the optimization convergence. Using the previously introduced ranking loss Equation 5.6, the problem with drawing sample hypotheses is to trade of the impact on the loss and the accuracy of the t_i estimation. The latter will ensure that the actual maximum of each hypothesis set is used to compute the loss and not an out of date estimate. Thus, we propose a heuristic to update the distribution for hypothesis sampling, which trades of the following two quantities (i) the error given the current loss (Equation 5.6) and (ii) the iterations since the last update of the function evaluation of each data sample.

More concretely, after every function approximator evaluation, we will update the prediction for the corresponding hypothesis and the iteration when the prediction was performed. For all hypothesis sets for which a hypothesis prediction was updated, the estimates for the corresponding t_i^+ and t_i^- are updated and the loss based error for the hypothesis is updated. Notice, almost all hypothesis in a set have zero loss since only negative and the maximum positive hypothesis are strictly enforced. If we normalize the error per hypothesis with the total error for all hypothesis, we obtain a distribution. Sampling hypotheses from this distribution will solely focus on improving the loss under the current t_i^+ and t_i^- estimates. Due to the assumed global nature of the function approximator, we have to ensure that these estimates are still true. Therefore, we augment this error with an *artificial* error term that captures the number of iterations since the last update of a data point. It is of the following form:

$$e(c,u;o,b) = \exp(-t + (c-u)/b) \tag{5.7}$$

where c is the current iteration, u is the last update iteration of the example, o a trade-off parameter to determine the base influence of not evaluating, and b determines how fast the influence grows.

Finally, after each optimization iteration, the hypothesis predictions and loss errors are updated as previously described. In addition to that, we add the artificial iteration dependent error term Equation 5.7 to the hypothesis error. The overall error for all hypothesis is normalized to get the discrete distribution from which we draw n samples (without replacement) to fill the new mini-batch. This distribution results in low sample frequency for hypotheses with small loss values, basically not until Equation 5.7 increases to a similar error magnitude as the maximum loss violating hypotheses. The maximum positive and negative hypothesis per hypothesis set are sampled more frequently if they do not fulfill the ranking loss. Figure 5.18 illustrates the optimization loop for our proposed loss and mini-batch sampling.

5.4.4 Experiments

As motivated before we are interested in evaluating the top-1 accuracy for grasp prediction using our large-scale grasp database. We use the same underlying deep neural network architecture and compare to an additional classifier evaluated in (Bohg et al. 2016). Table 5.19 reports our empirical results on the test dataset following the same training procedure as for the classification experiments (Chapter 5.3.3). Different from the classification accuracy we report top-1 accuracy as follows. In this case, a true positive is a prediction for a hypothesis set from an object point cloud for which the highest scored hypothesis is classified positive, and the ground truth label is positive. A true negative in this experiment is a prediction for a hypothesis set for which the

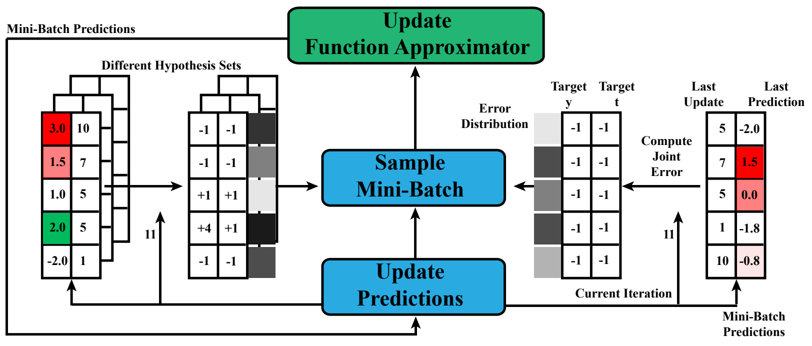


Figure 5.18: This figure illustrates the general optimization loop, sampling a mini-batch, performing one function approximator update step, feeding back the latest prediction values and updating the error distribution. We show two exemplary sets of hypotheses, the one on the left contains positive and negative examples and the one on the right only negative ones. The gray value of the computed error distribution signals the importance of this sample for the mini-batch sampling. Notice how the error due to the loss, indicated in red and green and the time since the last update affects the error distribution. The error distribution is normalized across all hypothesis sets and samples are drawn without replacement from the joint distribution.

highest ranked hypothesis is classified negative, and there is no positive labeled hypothesis in this set. The scalar threshold for the classification prediction, based on the ranking score, is obtained by cross-validation.

The results show that the proposed model trained on a ranking objective outperforms the two baselines by a large margin. This further indicates the importance of using *domain specific* knowledge in order to leverage large-scale machine learning approaches more efficiently within robotics applications such as grasping. For the dataset containing large objects, the performance improves by more than a factor of two. For the toy dataset of bottles, the improvement is moderate. This result is probably due to the simplicity of this subset of data where positive samples can be easily separated from negative ones. Notice that the datasets are highly unbalanced, meaning that the majority of the grasp hypotheses across all hypothesis sets are negative. The results on the other datasets suggest that it is much harder to perfectly separate positive from negative data while it is easy to ensure that the top-ranking one refers to a stable grasp. This can be due to remaining label noise in the dataset for which similarly looking templates can be either positive or negative.

Figure 5.20 illustrates how our proposed sampling procedure (Chapter 5.4.3.2) affects the samples used for optimization, focusing on the problematic examples the most. This graph supports our hypothesis that during the optimization of our proposed loss, the majority of the hypotheses are easy to separate based on the appearance-based features, resulting in low errors early on in the optimization procedure. However, every example is revisited due to the suggested heuristic,

	Bottles	**Small**	**Medium**	**Large**
data ratio	0.13	0.15	0.08	0.03
Forest	0.83	0.43	0.45	0.31
CNN	0.83	0.39	0.51	0.41
OURS	**0.85**	**0.59**	**0.70**	**0.84**

Figure 5.19: We report the data ratio (all positive grasps divided by all grasps) for each test dataset and the top-1 score on the test dataset obtained by three different methods. The top-1 accuracy indicates the ratio of point clouds in the test data set for which the best scoring template was classified positive and also had a positive ground truth label or the best scoring template was classified negative, and there was no positive ground truth example in the set. Results are reported per object group (bottles, small, medium, and large) and for gripper stand-off 0 from the object surface before closing the fingers. The proposed model that is trained on a ranking objective outperforms the baselines by a large margin. For large objects, the performance has more than doubled.

despite changes to the parameters of the learning method. The error for these examples maintains to be low meaning our underlying assumption that the global function approximator is not drastically changing its behavior is valid, even more so it does not *forget* about the *easy* examples over time.

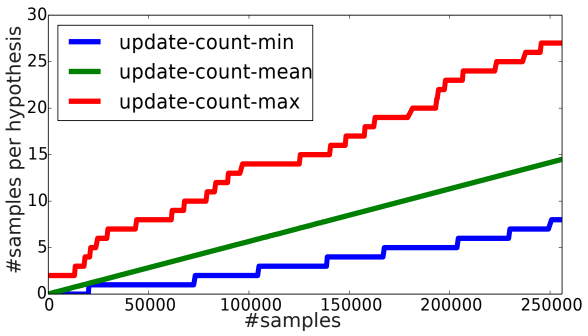


Figure 5.20: This figure shows the influence of the error distribution based sampling for the optimization. The minimal update count (blue) illustrates that due to the error component based on the iterations, all data samples are revisited over time. However, the maximum update count (red) shows that the optimization is mostly focusing on the difficult hypotheses.

5.5 Summary and Review

Motivated by our prior work on reactive *model-based* fast feedback systems for manipulation (Chapter 3) and robust task *learning* from demonstration (Chapter 4), we focused on *learning-based* grasp pose prediction in this chapter. Integrating domain-specific *model-based* approaches into state-of-the-art discriminate learning methods is a complex ongoing research effort. Instead, we propose to use our domain knowledge to produce high-quality ground truth data. Different from *model-based* grasping used during real robot executions, simulation-based data collection is not constraint by time; it is an offline process. Hence, we can leverage computationally expensive simulation, simulating complex sensor noise and expected uncertainties to facilitate a better real-world transfer. The learned mapping from sensor observation to grasp prediction is computationally efficient and different from *model-based* approaches does not require object identification and pose estimation. Since our proposed, physics-based grasp success method, to overcome the known limitations of the ε-metric has no theoretical guarantees, we conducted Mechanical Turk studies to verify that our large-scale and automatically labeled database with our physics-metric is indeed more likely to predict better grasp success. Our classification-based analysis provided further support that our dataset supervised by physics simulation is more consistent and therefore better suited for *learning*. Further, the large object variety and therefore complicated mapping in our database requires high capacity non-linear classification methods such as deep neural networks to achieve good performance. Finally, we derived a new problem specific top-1 ranking objective using our domain knowledge about robotic grasping for manipulation. This method directly optimizes to predict one successful hypothesis per sensor observation, if such a hypothesis exists. Therefore, our top-1 prediction approach automatically determines *easy*, less ambiguous hypotheses given their feature representation, resulting in better performance. The resulting problem is easier to optimize, the learned deep neural network converges quickly, despite its more complex loss objective. Hence, we demonstrated two different approaches to incorporate our *model-based* knowledge into the design of a domain-specific *learning* objective and realistic training data generation for grasping.

In conclusion, this work represents another building block towards *autonomous manipulation*. It complements our overall *model-based* system (Chapter 3) by enabling grasp hypothesis prediction from partially observed sensory data. This contactless pre-grasp pose is an ideal starting point for our *learning from demonstration* based robust task learning approach (Chapter 4) to learn fast feedback grasp controller which can recover from failures. However, after having successfully grasped the object, we still require high gain control to track the desired trajectories due to the additional payload which altered the system dynamics and therefore our wrong inverse dynamics model. In the following chapter, we discuss a novel method which uses the same underlying *model-based* inverse dynamics model but efficiently learns an additional non-linear model from data.

Chapter 6

Inverse Dynamics Learning

Achieving reactive and compliant behavior is a cornerstone of robotic applications involving safe interaction with humans. In previous chapters, we have already presented two reactive systems, a *model-based* fast feedback system enabling safe manipulation by proactive obstacle avoidance (Chapter 3) and a *learning-based* method to teach new tasks using multiple sensor modalities (Chapter 4). Both systems are based on *model-based* inverse dynamics control and PID feedback controller to compensate for *model* inaccuracies. As famously said in (Box 1976): "All models are wrong.". More specifically in the context of inverse dynamics models, *model* errors are inevitable due to system payload changes, e.g., when picking up an object (Chapter 3), and complex highly non-linear system dynamics. Therefore, high feedback gains (PID) are required to achieve good trajectory tracking for these dynamic manipulation tasks. However, inherently safe systems should be compliant, have low feedback gains. In this chapter we tackle the problem of task-specific inverse dynamics learning in low feedback gain settings, agnostic to already existing *model-based* inverse dynamics methods, complementing our work on reactive systems (Chapter 3) and learning from demonstration (Chapter 4).

Considerable effort has been put into developing machine learning methods that can learn or improve inverse dynamics models (Vijayakumar et al. 2000; Nguyen-Tuong et al. 2008; Gijsberts et al. 2013; Meier et al. 2014b). These approaches attempt to identify a global inverse dynamics model. Collecting data that covers the full state-space is typically not considered a viable approach for high-dimensional systems. Furthermore, when considering motions with object interaction (Chapter 3 and 4) learning one global model becomes even more involved if not impossible since the model has to additionally take contact and payload signals into account. Thus online learning (Chapter 2.2.1.1) has been a focus in these settings but remains a challenge. The key difficulties are computationally efficient learning of models that are flexible enough to capture the non-stationary data of inverse dynamics mapping and doing so on streaming data that is highly correlated. To circumvent the issues of the aforementioned methods we follow the path of learning *task-specific* (error) models (Jamone et al. 2014; Toussaint et al. 2005; Petkos et al. 2006; D. M. Wolpert et al. 1998). Error models do not replace the underlying *rigid body dynamics model* but rather complement it. This model is very well understood, quickly to obtain, and achieves excellent performance if it is correctly identified. Thus, we only have to learn error terms correcting the mistakes of the existing *model*, already taking care of many system non-linearities. Further, in the absence of task-specific data, our system is still fully functional due to the underlying *model-based* inverse dynamics approach that generalizes very well to the whole state-space.

Assuming that we want to learn task-specific inverse dynamics for low gain, very compliant systems, we can iterate and improve on the task, collect data and optimize our *learning-based* approach incrementally. While this approach is iterative by nature, we aim at making it as data-efficient as possible, such that only a few iterations are required, while achieving consistent convergence in the error model learning process. Task-specific models do not mitigate the problem of using payload signals or contextual information when trying to build a global model. However, it simplifies the overall global problem into two subproblems: (i) finding a task-specific inverse dynamics model and (ii) detecting which task model to use. In the case of learning from demonstration (Chapter 4), task identification is trivial. Thus, this approach is straightforward to incorporate in such a framework.

In the case of an existing imprecise inverse dynamics model (e.g., *model-based* rigid body dynamics) and low gain feedback controller, traditional inverse dynamics learning approaches learn the (error) model slightly off the desired trajectory. This comes from the fact that the data points used to learn the model are based on the actually achieved accelerations instead of the desired commanded accelerations. Thus if tracking is bad, the collected data points are slightly off the desired trajectory for which we wanted to identify the inverse dynamics model. For instance, one extreme case of bad tracking is encountered when the system cannot overcome static friction. In that case, no useful data for inverse dynamics model learning is generated. Traditionally, this is circumvented by increasing the gains of the feedback control term, at the cost of compliance. Alternatively, this issue is addressed by our recent work (N. Ratliff et al. 2016), which employs a *direct* loss function, minimizing the error between desired and actual accelerations, to learn feedback terms online.

In this work, we explore the intuition that feedback control can be viewed as an online technique to compensate for errors in a given a priori inverse dynamics model, as discussed in (N. Ratliff et al. 2016). The feedback terms, compensate for errors between what a given model predicts and what we actually need. Therefore they naturally act as a convenient source of training data. We show how the *direct loss* on accelerations, as presented in (N. Ratliff et al. 2016), can be transformed into a loss on inverse dynamics torques, measured at desired accelerations. As a result, we now have two training data sources: (i) the traditional inverse dynamics training data points measured at actual accelerations, and (ii) this new training signal measured at commanded accelerations. We show how this additional data source leads to more consistent convergence of the task-specific inverse dynamics learning process.

6.1 Background

Tracking desired accelerations with low feedback controller gains requires an accurate inverse dynamics model. The dynamics of a classical dynamical system can be expressed as

$$\tau = M(q)\ddot{q} + h(q,\dot{q}) \tag{6.1}$$

where $q, \dot{q}, \ddot{q}$ denote the joint positions, velocities, and accelerations, M is the inertia matrix, and h collects all the modeled forces such as gravitational, Coriolis, centrifugal forces, viscous and Coulomb friction. When possible, inverse dynamics approaches (Craig et al. 1987) model the system dynamics via the rigid body dynamics (RBD) equation of motions. Then, given sufficiently rich data, the RBD parameters can be identified using linear regression techniques (An et al. 1985), resulting in the approximate RBD dynamics model

$$\hat{\tau}_{\text{rbd}} = \hat{M}(q)\ddot{q}_d + \hat{h}(q,\dot{q}). \tag{6.2}$$

with approximate $\hat{M}$ and $\hat{h}$. This has been extended in (C. G. Atkeson et al. 1985), to additionally estimate payloads.

Unfortunately, the RBD model typically is not flexible enough to capture all non-linearities of the actual system's dynamics. As a result, the estimated RBD model is generally only a rough approximation. Thus, when attempting to track desired accelerations $\ddot{q}_d$ with $\hat{\tau}_{\text{rbd}}$ we achieve actual accelerations $\ddot{q}_a$ differing from $\ddot{q}_d$. Specifically, when $\hat{\tau}_{\text{rbd}}$ is applied on the real system, with the true unknown dynamics model M, h, we can express the actual accelerations $\ddot{q}_a$ as

$$\begin{aligned} \ddot{q}_a &= M(q)^{-1}[\hat{\tau}_{\text{rbd}} - h(q,\dot{q})] && (6.3)\\ &= M(q)^{-1}[\big(\hat{M}(q)\ddot{q}_d + \hat{h}(q,\dot{q})\big) - h(q,\dot{q})]. && (6.4) \end{aligned}$$

Note, if our estimated model $\hat{M}, \hat{h}$ were accurate, this expression would evaluate to $\ddot{q}_a = \ddot{q}_d$. However, this is typically not the case on real systems and because of this a feedback term τ_{fb} is required. This feedback term measures the error made and adds a corrective term to ensure accurate tracking. Traditionally, this feedback term is realized through PID control. The higher the gains, the better the tacking, at the cost of system compliance.

6.1.1 Learning Inverse Dynamics Models

To this end, various approaches to learning either the full inverse dynamics or an error model have been proposed. When learning an error model the total torque command is then a combination of any existing approximate *model* ($\hat{\tau}_{\text{rbd}}$), an error (torque) model (f_{iderr}) and a feedback term (τ_{fb}),

$$\tau_{\text{total}} = \hat{\tau}_{\text{rbd}} + f_{\text{iderr}} + \tau_{\text{fb}}. \tag{6.5}$$

One of the critical challenges of inverse dynamics learning is computational efficiency. Predicting with learned models needs to be feasible within the real-time constraints of the systems consuming torque commands. Furthermore, it is typically assumed that the inverse dynamics mapping is non-stationary and can change over time. Thus, approaches that can incrementally learn and are computationally efficient enough for real-time deployment (Vijayakumar et al. 2000; Nguyen-Tuong et al. 2008; Gijsberts et al. 2013; Meier et al. 2014b; Jamone et al. 2014; Meier et al. 2016) form one of the main research directions within the topic of inverse dynamics learning. However, learning globally valid models robustly on highly correlated data streams remains a challenge. More in-depth discussion of the challenges and existing approaches can be found in (Sigaud et al. 2011; Nguyen-Tuong et al. 2011).

Some robustness can be achieved by using analytical (parametric) models such as RBD models as priors, and learning an error model on top of that (Nguyen-tuong et al. 2010; de la Cruz et al. 2012; Camoriano et al. 2016). Such approaches can revert to this prior knowledge when the algorithm determines that the error model fit is uncertain.

Because of the difficulties of learning a globally valid model, some research has moved towards learning task/context specific inverse dynamics models (D. M. Wolpert et al. 1998; Toussaint et al. 2005; Petkos et al. 2006; Petrič et al. 2014; Jamone et al. 2014; Calandra et al. 2015; Christiano et al. 2016). In this setting it is feasible to collect task-relevant data for offline learning, therefore simplifying the learning process. However, this comes at the cost of having to detect the correct context at runtime such that the correct task model can be chosen. Our work, fits into this category, with the focus on efficiently learning one task model.

Finally, learning inverse dynamics models is typically motivated by being able to use low-gain feedback control. However, traditional inverse dynamics learning approaches initially need high enough gains to achieve good tracking, such that relevant data is being generated. Little work has been done towards automatically lowering the feedback gains once a suitable model has been learned. An exception is work presented in (Alberto et al. 2014) which allows for variable gains, using high gains, when the model is uncertain about its predictions and low gains when it is certain.

All of these methods learn inverse dynamics models on only the indirect data source measured at actual accelerations. In this chapter, we make use of two different data sources stemming from indirect and direct learning approaches to train inverse dynamics models.

Thus, before going into the details of our proposed approach, we discuss the notion of *indirect* and *direct* learning and present related work within that context.

6.1.2 Indirect vs. Direct Learning of Inverse Dynamics

As mentioned above, most of the recent work on inverse dynamics learning can be classified as *indirect* learning methods. The objective function that these methods optimize is given as

$$\mathcal{L}_{\text{indirect}}(\boldsymbol{\theta}) = \sum_{(x_a,\tau_{\text{total}})\in\mathcal{D}} ||\tau_{\text{total}} - f(x_a;\boldsymbol{\theta})|| \tag{6.6}$$

where the input data point $x_a = (q, \dot{q}, \ddot{q}_a)$ is a combination of the state $(q, \dot{q})$ and the actual accelerations $\ddot{q}_a$. The output value τ_{total} is the corresponding applied torque that achieved the accelerations $\ddot{q}_a$. Here, the function that encodes the mapping from x to torques is defined as $f(x;\boldsymbol{\theta})$, where $\boldsymbol{\theta}$ are the open parameters of the chosen model.

Online, the robot attempts $\ddot{q}_d$ from state $(q, \dot{q})$. It calculates torque τ_{total} and observes true accelerations $\ddot{q}_a$. Rather than training at input point $x_d = (q, \dot{q}, \ddot{q}_d)$, the data point $x_a = (q, q, \ddot{q}_a)$ is used with target τ_{total}. The drawback to indirect learning is that convergence might be slow since the data is off the trajectory/distribution we want to optimize for. This is especially difficult when $\ddot{q}_a = 0$ due to static friction since many torques map to this value (i.e., the inverse function is not one-to-one at this point).

Alternatively, it was shown in (N. Ratliff et al. 2016) that it is possible to directly measure the gradient of the acceleration error of a system, which enables a new class of direct online learning algorithms. This approach aims to minimize the error between the desired and actual accelerations directly, by optimizing the following *direct* loss function, for every pair of desired and actual accelerations,

$$\begin{aligned}\mathcal{L}_{\text{direct}}(\boldsymbol{\theta}) &= ||\ddot{q}_d - \ddot{q}_a(\boldsymbol{\theta})||_M^2 \\ &= ||\ddot{q}_d - M^{-1}[\hat{\tau}_{\text{rbd}} + f_{\text{iderr}} - h]||_M^2\end{aligned} \tag{6.7}$$

where the actual accelerations $\ddot{q}_a$ are a function of the error model f_{iderr}. Note, traditional direct adaptive control methods (Astrom et al. 2008; Nakanishi et al. 2004) have similar motivations – they use Lyapunov techniques to derive controllers that adjust the dynamics offset model with respect to some reference signal. (N. Ratliff et al. 2016) show how to effectively perform online learning on this objective leveraging well established online gradient descent techniques. However, this simple feedback term does not capture any structure of the error model and requires a relatively high learning rate to account for payload changes. (Meier et al. 2016) use the resulting error presented in (N. Ratliff et al. 2016) as a feedback term on acceleration errors and use the indirect loss function to learn a drifting Gaussian process to model larger structured inverse dynamics modeling errors. Note, (Meier et al. 2016) combine the direct and indirect learning as two separate learning processes with two different purposes: direct learning of an online adaptive feedback term and indirect learning of a locally valid inverse dynamics error model to capture larger errors. Also, the approach by (Meier et al. 2016) is a task-independent online learning approach and theoretically applicable anywhere throughout the state space. However, it retains no memory of previously learned error models and thus is not able to improve over time.

Our work presented here is orthogonal to this otherwise very related work: Here, we show that we can use direct and indirect learning within the same learning process of task-specific (feedforward) inverse dynamics (error) models, which can improve over time.

6.2 Combining Indirect and Direct Learning

To be able to learn the state-space dependent structure of the inverse dynamics modeling errors, we assume that the error model is identifiable in the space of $x = (q, \dot{q}, \ddot{q})$. Furthermore, since every task execution may slightly vary, we also follow an incremental learning process, meaning that each task execution generates a new training data set that can be used to update and improve our error model. Thus, our error models are indexed by k, indicating the k^{th} learning iteration. For $k = 0$, meaning that no error model exists for the task at hand, we simply assume $f^0_{\text{iderr}}(x; \boldsymbol{\theta}^0) = 0$. Given this, the total torque applied to the system is the approximate rigid body dynamics model $\hat{\tau}_{\text{rbd}}(x_d)$ (if available) plus an offline learned error model f^k_{iderr} and a feedback term τ_{fb}:

$$\tau_{\text{total}} = \hat{\tau}_{\text{rbd}}(x_d) + f^k_{\text{iderr}}(x_d; \boldsymbol{\theta}^k) + \tau_{\text{fb}}. \tag{6.8}$$

Here we show how the direct and indirect loss functions can be combined into one loss function that uses two different data sources. In order to do so we 1) discuss the loss functions in the context of offline error model learning, 2) show that the two loss functions create two different training signals for the error model, and 3) use this result to combine direct and indirect learning.

6.2.1 Indirect Loss Function

We start by discussing the details of learning an error model with an indirect loss. We compute the torque command based on the current state $(q, \dot{q})$ and desired accelerations $\ddot{q}_d$, apply this torque, and then measure actual accelerations $\ddot{q}_a$. Now we know what torque command achieves these measured accelerations and can use this data point to learn an inverse dynamics model. We collect all of these data points for one task execution, for $t = 1 \dots T$, such that we have T data points to learn parameters $\boldsymbol{\theta}^k$, initialized with the parameters $\boldsymbol{\theta}^{k-1}$.

In the indirect formulation, we try to optimize the parameters $\boldsymbol{\theta}^k$ such that the difference between the applied torque τ_{total} and the inverse dynamics model f_{id} at x_a is minimized:

$$\mathcal{L}_{\text{indirect}}(\boldsymbol{\theta}^k) = \sum_{t=1}^{T} ||\tau^t_{\text{total}} - f_{\text{id}}(x^t_a; \boldsymbol{\theta}^k)||^2 \tag{6.9}$$

Here we would like to utilize an approximate rigid body dynamics model (if available) and learn an error model f_{iderr} in order to optimize the f_{id} model. Notice, our approach does not require a rigid body dynamics model, all derivations hold when assuming a constant model $\hat{\tau}_{\text{rbd}} := 0$ as well. In this case we would learn the full inverse dynamics model, not using any domain specific knowledge. However, if possible we can and should leverage the existing rigitd body dynamics model. To compute what the RBD error is at input x^t_a, we have to evaluate $\hat{\tau}_{\text{rbd}}$ at x^t_a and subtract it from the total torque applied τ_{total}, such that, in the k^{th} learning iteration, we optimize

$$\mathcal{L}_{\text{indirect}}(\boldsymbol{\theta}^k) = \sum_{t=1}^{T} ||\tau^t_{\text{total}} - \hat{\tau}_{\text{rbd}}(x^t_a) - f^k_{\text{iderr}}(x^t_a; \boldsymbol{\theta}^k)||^2$$

Thus, using the indirect learning approach we optimize $f^k_{\text{iderr}}(x; \boldsymbol{\theta}^k)$ on the following data set

$$\mathcal{D}^k_{\text{indirect}} = \{x^t \leftarrow x^t_a, y^t \leftarrow \tau^t_{\text{total}} - \hat{\tau}_{\text{rbd}}(x^t_a)\}^T_{t=1}. \tag{6.10}$$

The quality of this training data set depends on how well we have tracked the task policy or trajectory. With accurate tracking behavior, one learning run should already give us a good approximation of the modeling errors. However, if tracking is bad, it very well may be that we require several learning iterations to estimate a good error model.

6.2.2 Direct Loss Function

To overcome the limitations of the indirect learning process, (N. Ratliff et al. 2016) proposes to use a direct loss (Equation 6.7) to learn modeling errors. Here we use this loss in acceleration space (N. Ratliff et al. 2016), to derive an additional data source for inverse dynamics learning.

We start with Equation 6.7, drop the weighting of the acceleration error by the inertia matrix M and instead multiply the accelerations with M

$$\begin{aligned}\mathcal{L}_{\text{direct}}(\boldsymbol{\theta}) &= \textstyle\sum_{t=1}^{T} \|M\ddot{q}_d^t - M\ddot{q}_a^t(\boldsymbol{\theta}^k)\|^2 \\ &= \textstyle\sum_{t=1}^{T} \|(M\ddot{q}_d^t + h) - M\ddot{q}_a^t(\boldsymbol{\theta}^k) - h\|^2\end{aligned} \tag{6.11}$$

where we have also added and subtracted h. The true dynamics model M, h is never evaluated in our loss formulation, it is merely used to derive the direct loss formulation as shown in the following. We can now summarize the first term as the true rigid body dynamics model $\tau_{\text{rbd}}(\ddot{q}_d^t)$, evaluated at the desired accelerations, and we expand $\ddot{q}_a^t(\boldsymbol{\theta}^k)$ as follows

$$\begin{aligned}\mathcal{L}_{\text{direct}}(\boldsymbol{\theta}) &= \textstyle\sum_{t=1}^{T} \|\tau_{\text{rbd}}(\ddot{q}_d^t) - M\ddot{q}_a^t(\boldsymbol{\theta}^k) - h\|^2 \\ &= \textstyle\sum_{t=1}^{T} \|\tau_{\text{rbd}}(\ddot{q}_d^t) - MM^{-1}[f_{id}(\ddot{q}_d^t;\boldsymbol{\theta}^k) - h] - h\|^2 \\ &= \textstyle\sum_{t=1}^{T} \|\tau_{\text{rbd}}(\ddot{q}_d^t) - (\hat{\tau}_{\text{rbd}}(\ddot{q}_d^t) + f_{\text{iderr}}^k(x_d^t;\boldsymbol{\theta}^k))\|^2 \\ &= \textstyle\sum_{t=1}^{T} \|(\tau_{\text{rbd}}(\ddot{q}_d^t) - \hat{\tau}_{\text{rbd}}(\ddot{q}_d^t)) - f_{\text{iderr}}^k(x_d^t;\boldsymbol{\theta}^k)\|^2\end{aligned} \tag{6.12}$$

where $f_{id}(\ddot{q}_d^t)$ represents the state based rigid body dynamics and error model without a feedback term which should ideally be zero. We now have transformed the loss on accelerations to a loss on torque commands at the input point x_d^t. Note that this transformed loss intuitively means that we want to minimize the difference between our error model $f_{\text{iderr}}^k(x_d^t;\boldsymbol{\theta}^k)$ and the true modeling error $\tau_{\text{error}}^t = (\tau_{\text{rbd}}(\ddot{q}_d^t) - \hat{\tau}_{\text{rbd}}(\ddot{q}_d^t))$ at input $x_d^t = (q^t, \dot{q}^t, \ddot{q}_d^t)$. While intuitively pleasing, we unfortunately do not have access to the true modeling error τ_{error}^t. However, we can get an estimate of the modeling error $\hat{\tau}_{\text{error}}^t = \tau_{\text{fb}}^t + f_{\text{iderr}}^{k-1}(x_d^t;\boldsymbol{\theta}^{k-1})$ by combining the feedback term τ_{fb}^t and the error model f_{iderr}^{k-1} from the previous task execution (similar to feedback error learning (Nakanishi et al. 2004)), which results in the following loss

$$\mathcal{L}_{\text{direct}}(\boldsymbol{\theta}) = \textstyle\sum_{t=1}^{T} \|\hat{\tau}_{\text{error}}^t - f_{\text{iderr}}^k(x_d^t;\boldsymbol{\theta}^k)\|^2 \tag{6.13}$$

Similar to the indirect learning we can now construct a dataset

$$\mathcal{D}_{\text{direct}}^k = \{x^t \leftarrow x_d^t, y^t \leftarrow \tau_{\text{fb}}^t + f_{\text{iderr}}^{k-1}(x_d^t;\boldsymbol{\theta}^{k-1})\}_{t=1}^{T} \tag{6.14}$$

which can be used to learn or update the new error model f^k_{iderr}. We now receive data points directly on the desired accelerations. However, also this data set's quality depends on tracking accuracy. With low feedback gains, the initial τ_{fb} may not really capture the errors very well, such that the first learning iteration may only capture part of the modeling errors.

Thus, with low feedback gains, we may require multiple learning iterations to learn an accurate error model. Whereas increasing the feedback gains g would lead to improved tracking, increasing the fidelity of the data, at the cost of compliance. Here we simply propose to do both: use g_{low} to compute the feedback terms $\tau_{\text{fb}}(g_{\text{low}})$ which are sent to the system, and use g_{high} to compute feedback terms $\tau_{\text{fb}}(g_{\text{high}})$ which are sent to the learner. Notice, the feedback term using $\tau_{\text{fb}}(g_{\text{high}})$ is never applied on the system. Thus, we maintain a very compliant system while obtaining better error data for the portion of the state space reached with the g_{low}. This can be helpful to break stiction or counteract high friction with f_{iderr} after fewer iterations which otherwise would not be possible with traditional inverse dynamics learning approaches and low gains.

6.2.3 Joint Inverse Dynamics Learning

The key insight for our approach is that we can use the feedback term as an error estimate for the desired accelerations. Thereby the error model learning problem has two data sources *indirect* and *direct*. Both exhibit the same structure to optimize the error model. Hence, we can formulate a joint function approximation problem of the form:

$$\mathcal{L}_{\text{joint}}(\boldsymbol{\theta}^k) = \sum_{(x,y)\in\mathcal{D}^k_{\text{joint}}} ||y - f^k_{\text{iderr}}(x;\boldsymbol{\theta}^k)|| \tag{6.15}$$

where data points for the actual accelerations $\ddot{q}^t_a$ for every timestep t can be used as well as data points for the desired accelerations $\ddot{q}^t_d$ as described by

$$\mathcal{D}^k_{\text{joint}} = \mathcal{D}^k_{\text{direct}} \cup \mathcal{D}^k_{\text{indirect}}. \tag{6.16}$$

6.3 Task Specific Inverse Dynamics Learning

Implementing our approach required several design decisions on the levels of motion generation, control, and learning. Here we will give a short overview of our design choices, and give an algorithmic overview of our iterative approach to learning task-specific inverse dynamics models.

Algorithm 1 Execute Task and Collect Data

```
Require: $f^{k-1}_{\text{iderr}}$, $\pi(q^t,\dot{q}^t)$, system($\tau^t$), $q^0,\dot{q}^0$
 1: $\mathcal{D}^k_{\text{joint}} = \emptyset; t = 0$
 2: while not converged do
 3:     $\ddot{q}^t_d = \pi(q^t,\dot{q}^t)$
 4:     $x^t = (q^t,\dot{q}^t,\ddot{q}^t_d)$
 5:     $\tau^t = \hat{\tau}_{\text{rbd}}(x_d) + \tau^t_{\text{fb}} + f^{k-1}_{\text{iderr}}(x^t_d;\boldsymbol{\theta}^{k-1})$
 6:     $t = t+1$
 7:     $q^t = \text{system}(\tau^{t-1})$
 8:     $\dot{q}^t,\ddot{q}^{t-1}_a = \text{finiteDiff}(q^t,q^{t-1})$
 9:     update feedback term $\tau^t_{\text{fb}}$
10:     $\mathcal{D}^k_{\text{joint}} = \mathcal{D}^k_{\text{joint}} \cup (x^{t-1}_a, \tau^{t-1} - \hat{\tau}_{\text{rbd}}(x^{t-1}_a))$
11:     $\mathcal{D}^k_{\text{joint}} = \mathcal{D}^k_{\text{joint}} \cup (x^{t-1}_d, \tau^{t-1}_{\text{fb}} + f^{k-1}_{\text{iderr}}(x^{t-1}_d;\boldsymbol{\theta}^{k-1}))$
12:
13: end while
14: return optimize($f^k_{\text{iderr}}$, $\mathcal{D}^k_{\text{joint}}$)
```

On the motion generation level, we assume that a kinematic policy for our task is provided, similar to Chapter 3, meaning that we can obtain desired accelerations $\ddot{q}^t_d$ for every state $q^t,\dot{q}^t$ relevant to our task. In particular, we use kinematic Linear Quadratic Regulators (LQRs) (Stengel 1994) (see Chapter 2.1.4) to provide us with the acceleration policy that is being executed. On the control level, we use two types of feedback controllers: traditional PID control and the recently introduced adaptive feedback learning DOOMED by (N. Ratliff et al. 2016). When tuned sufficiently well, both approaches can result in good tracking performance. We also have to choose a function approximator for the error model. Because we are learning the error model offline, learning speed is not our main concern. However, prediction speed is important since the models need to be evaluated at 1000Hz (hard real-time) on our robotic platform. We choose a simple feedforward neural network which is capable of learning nonlinear mappings and predictions require only a simple feedforward pass.

Finally, our iterative learning approach (Algorithm 1) can be summarized as follows:

1. task execution: Run task with approximate RBD model, feedback control, and feedback error model if existent (line 5). Construct data set's with two different sources of information (Equation (6.10) (line 10) and (6.14)) (line 11) during the task execution, as shown in Algorithm 1.

2. learning phase: Update the error model function approximators based on new data points, or construct and initialize the error models if none exist (line 14).

3. repeat the task.

As we empirically show in the Chapter 6.4, exploiting both sources of information allows us to obtain a better fit of the error model with fewer task iterations, compared to using a single data source.

6.4 Experiments

We evaluate our approach in two different settings. First, we analyze the proposed usage of indirect and direct data sources in order to learn a task-specific inverse dynamics error model on a 2D simulation. This allows us to extensively test characteristics of the learning problems based on the different data sources, using the same function approximator, under various simulated noise levels, stictions, frictions, and a wrong RBD model. This evaluation is focused on investigating the importance and influence of the different data sources rather than the function approximator itself. Second, we report results on task-specific inverse dynamics learning using both data sources on the KUKA lightweight arm of our robotic platform shown, the same robot as in Chapter 3. We start out describing the evaluation of our 2D simulation setting.

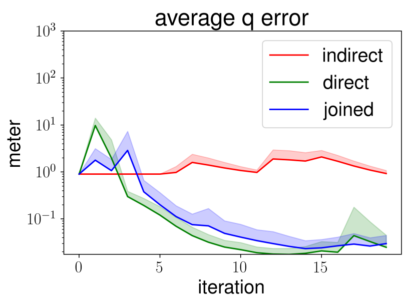
6.4.1 Simulation Experiments

6.4.1.1 System description

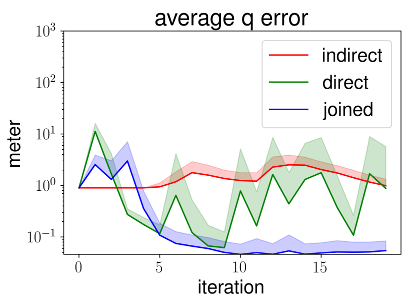
This evaluation is based on a simple 2D example (**Kappler** et al. 2017a) of a system for which the approximate RBD dynamics model differs drastically from the true dynamics. In this simulated system, the true mass is set to $M = 5I$, while the approximate mass is assumed to be $M = 0.5I$. The system attempts to realize a simple acceleration policy, defined as a PD-controller with desired state at $q_{\text{des}} = (1,1)$ and initial state $q_{\text{init}} = (0,0)$ and $\dot{q}_{\text{init}} = (0,0)$. Furthermore, we simulate the true system to experience friction and stiction which is not modeled by the approximate RBD model. Finally, we also add sensing noise to the position trajectories to mimic noisy sensor measurements. The source code of our example simulation with all parameters, friction, and stiction models used for the experiments can be found online (**Kappler** et al. 2017a).

6.4.1.2 Details of error model learning

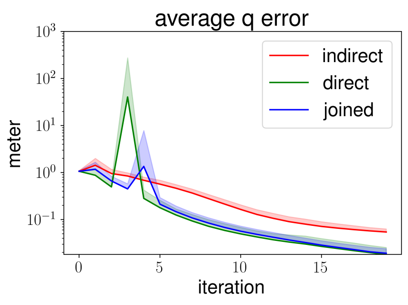
Every experiment involves running the iterative learning process for 20 iterations. After each cycle, a neural network is trained on the collected data and is then used to predict the modeling error in the next cycle. The error model f_{iderr} is learned via a neural network structure which



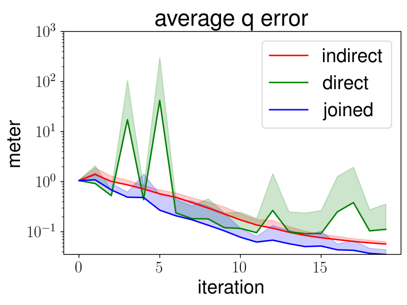
(a) PID, low system noise



(b) PID, very high system noise



(c) doomed, low system noise



(d) doomed, very high system noise

Figure 6.1: Hand-picked simulation runs to illustrate (dis)-advantages of the *direct* or *indirect* data sources. On the top row, we show position tracking error as a function of learning iterations, for both low-gain PID and low-gain DOOMED feedback control on systems with low and very high noise. In low noise settings, for both PID and DOOMED, the direct data source leads to improved position tracking with an increasing number of learning iterations. However, when using the indirect data source the NN cannot capture the error model, and thus position tracking does not improve at all (with PID) or more slowly (with doomed) over time. In the very high noise setting, the learning convergence with the indirect data source is comparable to the low noise setting, but when using the direct data source alone, we see erratic tracking convergence. The direct data is affected more by the noise. For both PID and DOOMED, the position tracking performance converges more consistently when using the joint data set.

consists of fully connected layers (200, 100, 50, 20, 1) with non-linearities (PReLU (He et al. 2015)) after every layer except the last (see Chapter 2.2.2.4 for more information on neural networks). We optimize one network per simulated joint of our system. The neural network is trained on the indirect dataset (Equation (6.10)), the direct dataset (Equation (6.10)), and as proposed in this chapter the joint data set (Equation (6.16)).

6.4.1.3 Experimental Setup

To evaluate the use of the data source variants (direct, indirect or joint) in a principled manner, we simulate various system conditions:

- 4 maximum sensing noise levels in meters are reported: low (0.0001), medium (0.0005), high (0.007), very high (0.008)
- 2 friction levels: medium and high
- 2 stiction levels: medium and high

across different hyper-parameter settings:

- 2 different number of training epochs of the neural network training per task iteration are evaluated (20, 50).
- 2 different gain settings (applied) for DOOMED and PID: low and high. The PID gains have been tuned such that even without error model we achieve convergence to the goal. Low gains are one order of magnitude lower.

The friction model is discontinuous and changes throughout the state space. The feedback term for learning is exponentially filtered with the same value (0.1) for all experiments. This is possible since the learning feedback term is not applied to the system.

For the detailed noise, friction and stiction models as well as the exact parameters to replicate the experiments, please see our reference implementation (**Kappler** et al. 2017a). For each data source variant (direct, indirect, joint) a total of 16 system combinations were executed, 10 times each, with different random seeds. A total of 1280 experiments were performed to cover the different hyper-parameter settings. To make results comparable, the random seed was kept consistent across the data source variants. For each of these runs, we record the average magnitude of the applied feedback torque, the desired and actual acceleration, as well as the desired and actual position.

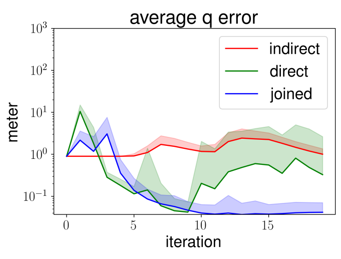
6.4.1.4 Illustration of Indirect vs. Direct

We start by illustrating some hand-picked scenarios that showcase the differences in learning on indirect vs. direct data in Figure 6.1. We choose examples with the same parameter settings and illustrate the difference when going from low to high noise. For the chosen examples, the learning rates/feedback gains were set to the low value, such that friction and stiction were not so easily overcome with feedback terms alone. This has the effect that for low-gain PID control we see no improvement in position tracking over time when using the indirect data only. With DOOMED, some improvement can be observed (with indirect data) - but at a slower rate compared to using direct data. The noise level does not affect the indirect learning process much. However, it affects the learning on direct data. In the low noise setting, we observe how position tracking improves over time, but in the high noise setting, this is not true. However, when combining both data sources, we get consistently good convergence of position tracking performance, even with low-gain feedback control.

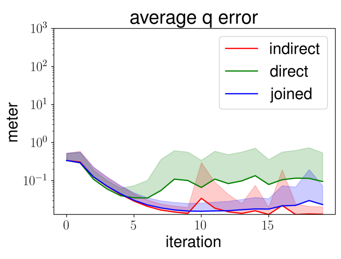
6.4.1.5 Extensive Evaluation

To provide a more extensive evaluation we now present results obtained when averaging across all system and parameter settings, see Figure 6.2. These results show that, on average, using both data sources results in a more consistent and faster convergence of the position tracking error, when compared to using the traditional approach of using indirect data alone. Specifically, in the case of low-gain PID control, the error model trained with the indirect data alone does not improve the position tracking error, whereas the joint learning process does.

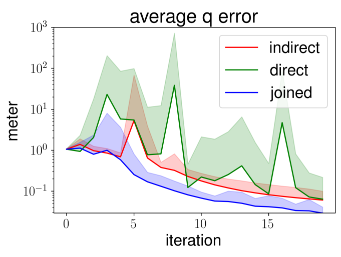
In the high-gain setting, the improvements are less pronounced. However, we want to stress that one important goal of this work is to learn an accurate task-specific inverse dynamics model while being as compliant as possible. Thus, the high gain setting shows that our proposed approach does not deteriorate in case of a less compliant system configuration, but there is not much to gain by using the additional data source. Intuitively, this makes sense, since, in a high gain setting, the feedback control term is expected to provide good tracking performance in the very first task execution already. Thus the indirect data collected during that first run already provides excellent data to learn a model for that particular task. We want to emphasize that there exist parameterizations and system settings that can lead to better performance by a single data source. However, on average, the direct data source seems to be most sensitive to the system/parameter combination, and the indirect data source requires higher feedback gains to be useful. The joint data source, however, can achieve superior and more consistent model learning performance, in low-gain settings.



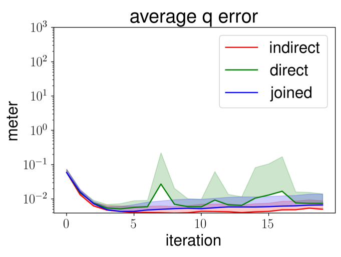
(a) PID: low-gain



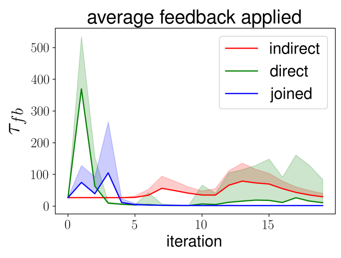
(b) PID: high-gain



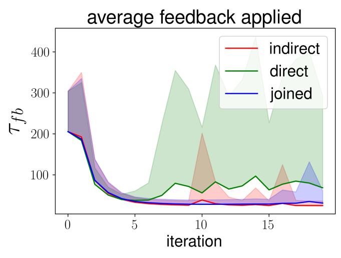
(c) PID: low-gain



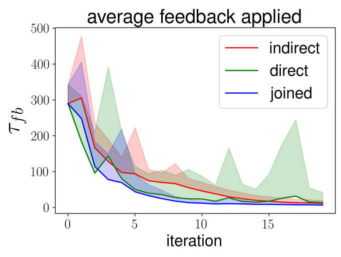
(d) PID: high-gain



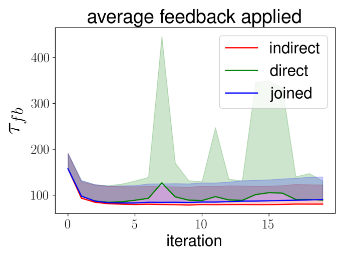
(e) PID: low-gain



(f) PID: high-gain



(g) doomed: low-gain



(h) doomed: high-gain

Figure 6.2: Results for low-gain and high-gain feedback control averaged across all system and parameter settings. (top two rows) shows the position tracking error convergence as a function of the number of learning iterations. (bottom two rows) shows the average feedback term applied. We plot the mean and the mean plus one standard deviation of the results. *low-gain:* In this setting, even when averaging across all system settings and parameters, we observe similar position tracking behavior for the data source variants as in our hand-picked illustrations. Indirect data alone is not able to capture the error model, which also explains why higher feedback torques are required in this setting. Using direct data alone, results have a higher variance. On average the mean tracking behavior degrades again after a few learning iterations (PID) or is somewhat erratic (DOOMED). Using the joint data set results in the most consistent tracking error convergence for both feedback controllers. *high-gain* When using high feedback gains, the joint dataset method does not gain as much in convergence performance (over using indirect data alone). However, using both data sources also does not degrade performance.

6.4.2 Real robot experiments

We have evaluated our method with two real-world robot experiments, a quantitative and a qualitative one. Our platform consists of two KUKA lightweight arms, each of which has 7 degrees of freedom. All experiments were performed on one arm resulting in a 21-dimensional input for the error model learning problem. The control system operates on a hard real-time loop of 1 kHz. Thus, all predictions are performed in less than 1 ms. All our experiments presented here, use DOOMED as the feedback controller. We optimize one neural network per joint with the following structure: 4 fully connected layers (200, 100, 50, 1) with non-linearities (PReLU (He et al. 2015)) after every layer except the last. Furthermore, we bound the predicted torques to $\pm 20Nm$. For both real robot experiments, we analyzed the task-specific inverse dynamics learning based on the joint dataset since this has shown to provide the most consistent performance.

For our quantitative experiments, we used a pre-planned sequence of LQR policies to generate desired accelerations. We execute the task 10 times, always starting in the same position, up to the precision of the arm. After each run, the collected data was used to re-optimize the neural networks. Figure 6.3 shows how the sum of squared feedback terms (τ_{fb}), averaged across all joints, changes with each run. Notice, the first run uses no error model, thus reflecting how much the feedback term has to compensate for the modeling errors. Within one iteration we are already able to capture most of the error with the learned error model. Hence, our learned model in combination with the rigid body dynamics model is now a reliable inverse dynamics model for this task. We want to stress that after every trial the newly obtained data is used to refine our task-specific error model further, and despite the data correlation between trials our updated model does not degrade. In Figure 6.4 we show the feedback term trajectory for the first and last task execution, per joint. Again it can be seen how our learned model compensates for the errors such that the feedback term only has to adjust for system noise.

In our qualitative example, we show the data efficiency of our approach for a real-world manipulation task. The robot has to pick up a heavy drill from a table and place it in a different location on the same table. Since we are interested in collaborative settings, we chose the learning rate of DOOMED as low as possible such that the robot itself is as compliant as possible while still being able to lift the drill at least. As shown in the video at (**Kappler** et al. 2017b), the usage of the joint dataset enables our system to significantly improve the performance of the pick and place task after a single iteration.

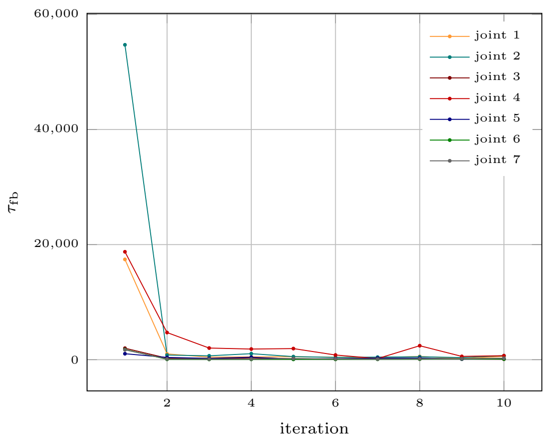


Figure 6.3: Mean squared feedback terms (τ_{fb}) as a function of learning iterations. Run 1 corresponds to Figure 6.4 (top) and run 10 to Figure 6.4 (bottom).

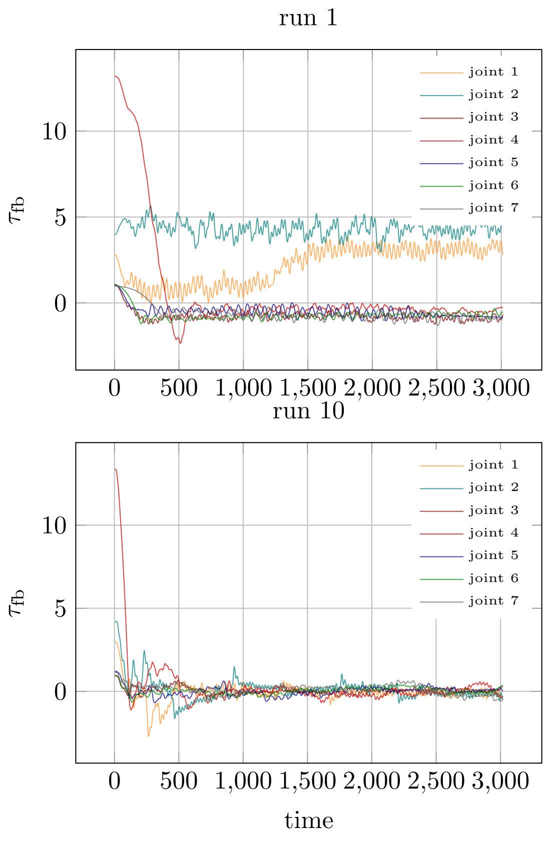


Figure 6.4: (top) The initial feedback terms (τ_{fb}), per joint, without a learned error model. (bottom) τ_{fb} after 10 learning iterations.

6.5 Summary and Review

In this chapter, we presented a novel approach for task-specific inverse dynamics learning agnostic to an underlying rigid body dynamics *model*. To the best of our knowledge, the usage of the two data sources, *indirect*, measured at the actual accelerations, and, *direct*, minimizing the error between desired and actual accelerations, is a novel approach to this problem. We have empirically shown, that using both data sources enables learning error models more consistently and faster. Additionally, our approach allows data collection with low-gain feedback controllers. This ability to use low gains is essential to have inherently safe systems not only after an error model has been learned but already during data collection.

Good inverse dynamics control is essential for the previously presented system (Chapter 3). Our proposed approach to inverse dynamics error model learning is well suited and important for *learning from demonstration* settings (Chapter 4) since both methods are currently assumed to be task-specific. Reactive motion generation systems (Chapter 3), reacting to perceptual feedback in real-time, might drastically alter the desired trajectory towards a task-specific goal due to, e.g. obstacle avoidance. Such strong perturbations can result in undesirable predictions since the error model has not been trained on any data for that region of the state space. Fortunately, we only add error terms in addition to the globally valid rigid body dynamics model mitigating the effect of wrong predictions of the *learned* error model. Still, the overall trajectory tracking performance might degrade due to the learned model in such scenarios. There are interesting future avenues to determine the confidence of the learned error model using feedback terms to alleviate such issues.

Another interesting observation is that robot manipulation system often repeatedly optimize similar objectives, learn similar function approximators. *Learning* task-specific inverse dynamics models is a prime example of this problem category. We believe that there exist many opportunities to speed up robotic optimization problems exploiting this insight, one particular approach towards faster converges using *meta-learning* is presented in the next Chapter 7.

Chapter 7

Learning to Learn While Learning

Within this thesis, we have so far discussed *model-based* reactive manipulation systems (Chapter 3), *learning from demonstration* based task controller using sensory feedback (Chapter 4), *learning-based* grasp pose prediction (Chapter 5), and task-specific inverse dynamics *learning* (Chapter 6). All of these contributions are designed towards increasing the capabilities of our robotic manipulation system towards *autonomy*. All proposed methods have one property in common, they either repeatedly solve *model-based* optimization problems, require incremental *learning* towards true *autonomoy*, or attempt to reduce real-world data samples for *learning*. To be more concrete, our reactive fast feedback system (Chapter 3) continuously solves an optimization-based planning objective. Fast feedback and therefore fast iteration is crucial for the reactiveness of this system. Hence, it would be highly beneficial if we could learn how to achieve better convergence faster. In Chapter 6 we presented a task-specific inverse dynamics learning approach which already improves sample efficiency. However, when learning multiple task-specific models, no information besides initializing the deep neural network is shared. Further, the robots' sensors produce hundreds of data points per second. This information could be used to adapt an existing task-specific model online if the optimizer could handle highly correlated data samples. Thus, both problems would benefit from an approach which learns something about the optimization process itself, better updating the learned models and achieving faster convergence.

In fact, humans have the remarkable ability to quickly learn new skills, tasks and solve repetitive problems more efficiently. This stems from a hierarchical learning process. Instead of learning each new skill from scratch, a higher-level – more abstract – meta-learner acquires information about the learning process itself which is used to guide and speed up the learning of new skills. In the machine learning community, this concept of learning-to-learn has been explored in a variety of contexts (Hochreiter et al. 2001; Schmidhuber 1987; Schweighofer et al. 2003; Vilalta et al. 2002) and recently received renewed attention (Andrychowicz et al. 2016; Li et al. 2016; Daniel et al. 2016; Hansen 2016; Fu et al. 2016).

In this chapter, we take inspiration from human motor control task learning to tackle the problem of *learning to learn*. (Herzfeld et al. 2014) observed in human studies that when learning from motor control errors, we do not necessarily adapt our current behavior based on the magnitude of the error. Instead, a current hypothesis is that when we learn to perform specific motor skills, we also learn a *memory of errors*. This memory of errors guides new motor skill learning tasks to achieve faster learning. In this work we take inspiration from (Herzfeld et al. 2014), and propose a meta-learning algorithm to learn a *memory of learning rates*. This *memory of learning rates* allows us to predict adaptive learning rates and update parameters more effectively.

Recent work on learning how to optimize (Andrychowicz et al. 2016; Li et al. 2016; Daniel et al. 2016; Hansen 2016; Fu et al. 2016) employs a two-phase approach: (i) A meta-learner is optimized to perform well on a priori chosen tasks, and (ii) once this meta-learner has been trained, it can be utilized in similar optimization tasks. In this work, we propose an online meta-learning algorithm, that learns to predict learning rates given currently observed gradients while optimizing task-specific problems. We show how we can train such a *memory of learning rates* online and in a computationally efficient manner. Our proposed approach has several advantages: Because we perform the meta-learning online, our method utilizes each observed data point for both the task learning as well as the meta-learning. Not only does this mean that we utilize observed data more effectively, but also that the effect is immediate. Furthermore, performing the meta-learning online alleviates the need for having to collect data on which to perform the learning-to-learn optimization process. Thus, our meta-learner can improve when necessary, while recent work is constrained to perform well on the task-distributions it was trained on. Finally, the resulting memory of learning rates can be transferred to similar optimization problems, to guide the learning of the new task.

To evaluate our meta-learning approach, we first illustrate its behavior on the well-known Rosenbrock optimization problem. After that we show empirical evaluations of two supervised learning tasks: Sequential binary classification tasks on the MNIST (LeCun 1998) dataset. This task allows us to analyze the effect of incrementally introducing new object classes, relevant to robotics since autonomous systems have to learn about new objects from few samples quickly. Learning (task-specific) inverse dynamics models (see Chapter 6) represents the second experiment type. This problem is ideal for meta-learning given the high frequency of the data and the challenges due to the highly correlated data samples. Our experiments show that when combining an optimizer with our meta-learner, we generally increase convergence speed, indicating that we utilize observed data points more effectively to reduce errors in the learning task. Furthermore, we show that when transferring our meta-learners internal state to new learning tasks, learning progress is faster.

7.1 Background

As mentioned above, most learning control problems use computationally efficient variants of gradient descent

$$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^{t} - h(\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\text{task}}(f(\boldsymbol{\theta}^{t}))) \tag{7.1}$$

where $\mathcal{L}_{\text{task}}$ typically corresponds to some loss function, f the model with parameters $\boldsymbol{\theta}$ to be optimized, and h transforms the observed gradient according to some rule. Note, in the remainder of this chapter we sometimes suppress the dependency of $\mathcal{L}_{\text{task}}$ on f, such that $\mathcal{L}_{\text{task}}(f(\boldsymbol{\theta})) = \mathcal{L}_{\text{task}}(\boldsymbol{\theta})$. State-of-the-art methods for large-scale learning problems (Chapter 5) also use gradient descent variants. Since gradient computation for such problems is bounded by the dataset size, stochastic variants (see Equation 2.17) are often the only viable approach (Chapter 2.2.1.4).

In this gradient-based setting, meta-learning can happen at several levels. For instance, recent work by (Finn et al. 2017) proposes a meta-learner that biases the learning process towards feature representations that supports few-shot learning of new tasks. On the other hand, recent learning to learn approaches (Andrychowicz et al. 2016; Li et al. 2016; Daniel et al. 2016; Hansen 2016; Fu et al. 2016) have been focused on learning optimizers that can be re-used for similar optimization tasks. The focus of these approaches, however, is on mitigating the issue of hyperparameter tuning instead of learning representations that can be transferred between (robotic) learning tasks.

Here, we present a novel meta-learning algorithm that is aligned with this second type of meta-learning but is focused on maintaining an internal memory that can guide the learning of new tasks. In the following, we review related work concerned with adaptive and learned optimizers.

7.1.1 Adaptive First-Order Methods

In the machine learning community adaptive optimizers such as *Adam* (Kingma et al. 2014), *RMSprop* (Tieleman et al. 2012), *AdaGrad* (Duchi et al. 2011), AdaDelta (Zeiler 2012) have been developed. In essence, these optimizers extend the gradient transform mapping h to be a function of sufficient statistics such as the mean and variance of the observed gradients

$$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^{t} - h(\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\text{task}}(\boldsymbol{\theta}^{t}), \mathbb{E}[\nabla\mathcal{L}_{\text{task}}(\boldsymbol{\theta})], \mathbb{E}[(\nabla\mathcal{L}_{\text{task}}(\boldsymbol{\theta}))^{2}]) \tag{7.2}$$

The specific gradient transform h then depends on the algorithm, as has been nicely summarized in (Daniel et al. 2016), Table 1. These optimizers are of linear space and time complexity with respect to the model parameters, and thus are suitable for highly parameterized models such as deep neural networks. Yet, choosing the initial learning rate for each of these methods, or which base learner to choose remains a complex manual tuning task (Loshchilov et al. 2016; Goyal et al. 2017). More importantly, though, such adaptive methods are not meant to extract information about the learning process itself. Thus, while they have proven successful at increasing convergence speed, they are not designed to transfer knowledge to new optimization tasks.

7.1.2 Learning to Learn

More recently, the idea of meta-learning – learning to learn – has re-gained momentum (Andrychowicz et al. 2016; Li et al. 2016; Daniel et al. 2016; Hansen 2016; Fu et al. 2016). These approaches parametrize h to be a function of parameters ψ, which determine how observed gradients are to be transformed

$$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t - h(\nabla_{\boldsymbol{\theta}} \mathcal{L}_{\text{task}}(\boldsymbol{\theta}^t); \psi). \tag{7.3}$$

The goal is to learn ψ to create a well-performing optimizer. In the most simple case, the parameter ψ simply equals the learning rate η, which can be adapted online, as has been shown in (Gunes Baydin et al. 2017). Specifically, the authors propose to compute the gradient $\nabla_\eta h$ online and then perform gradient descent on the learning rate itself. While this approach is simple and general as well as computationally and memory efficient, it does not retain a state – everything is forgotten. Thus subsequent optimization tasks start from scratch.

When going from adaptive optimizers to learned optimizers we hope to have learned something that we can reuse later on; that we performed meta-learning on some level. Recent work such as (Daniel et al. 2016; Andrychowicz et al. 2016; Li et al. 2016) address this to some extent. An optimizer is trained to perform well on some pre-defined set of optimization tasks and is then used to optimize similar learning problems. While some approaches learn to transform the gradient (Andrychowicz et al. 2016; Li et al. 2016), others assume h to be a scaling of the gradient and learn to predict the learning rates (Daniel et al. 2016; Hansen 2016; Fu et al. 2016).

Our work falls into this second category of trying to learn a coordinate-wise scaling of the gradient. To the best of our knowledge, recent work either performs learning on the step size online, but retains no memory (Gunes Baydin et al. 2017), or the learning of the optimizer is performed once at the beginning and never updated again. In the latter setting, two learning phases exist: In phase one h is trained, either via reinforcement learning (Daniel et al. 2016; Li et al. 2016; Hansen 2016; Fu et al. 2016) or in a supervised manner (Andrychowicz et al. 2016); In phase 2, the trained gradient transform h is used to optimize similar learning problems.

These recent learning to learn approaches are mostly focused on mitigating the issue of hyper-parameter tuning by learning an optimizer. The goal of our approach is to learn a representation of optimal gradient transforms that can be transferred to new learning tasks. Furthermore, as opposed to previous work, we learn this representation in an online fashion while using this representation to optimize task-specific optimization problems.

7.2 Learning a Memory for Learning Rates

In this work, we investigate how we can learn a *memory of learning rates* online, in a computationally efficient manner. Ideally, this *memory* can be transferred to subsequent similar learning tasks, and speeds up the convergence of that optimization problem. Furthermore, this memory should be continuously updated to be able to adapt and compress new learning rate landscapes as well.

We envision our meta-learning algorithm to be used as follows: While our system attempts to learn a model $f(\boldsymbol{\theta})$ for a specific task, such as a task-specific inverse dynamics model, by minimizing the loss $\mathcal{L}_{\text{task}}(f(\boldsymbol{\theta}))$ it also aims at building a model h that can predict how much to correct for observed errors. Thus for each optimization step, we perform two updates: a gradient descent step on the task-specific model parameters and update our meta-learners memory h. An overview in the form of a pseudo algorithm is given in 2.

In order to develop such a meta-learning approach, several challenges need to be met. One of the core challenges is the choice of training signal to learn such a memory on. We take inspiration from (Riedmiller et al. 1993) and start by showing how to derive a simple learning algorithm to train a learning rate memory h for one-dimensional optimization problems. Then, we discuss a representation for h that supports computationally efficient, incremental learning. Finally, we show how to generalize our meta-learning approach to optimization problems involving complex models such as neural networks.

7.2.1 Training Signal for the Learning Rate Memory

Let us assume that our main objective is a learning task that requires us to minimize the objective $\mathcal{L}_{\text{task}}(\theta)$ with respect to parameter θ. While optimizing parameter θ we also aim to learn a

Algorithm 2 Online Meta Learning

```
Require: initial task model $f(\theta^0)$, initial meta-memory $h_{\psi^0}$
1: $z^0 = 0$
2: for $t \in 1,...,T$ do
3:     $z^t = \nabla_\theta \mathcal{L}_{\text{task}}(f(\theta^{t-1}))$
4:     $h_{\psi^t}$ = MemoryUpdate($z^t, z^{t-1}, h_{\psi^{t-1}}$)          // eq. 7.10
5:     $\theta^t = \theta^{t-1} - h_{\psi^t}(z^t)$          // eq. 7.4 and eq. 7.11
6: end for
7: return $f(\theta^T), h_{\psi^T}$
```

function h that given gradient information of the main objective $\mathcal{L}_{\text{task}}(\theta)$, can predict a learning rate $\hat{\eta}$

$$h(z;\psi) = \hat{\eta}(z;\psi)z \tag{7.4}$$

where ψ are the parameters of the *learning rate memory h* and $z = \nabla_\theta \mathcal{L}_{\text{task}}(\theta)$. Ideally, we would like to optimize the parameters ψ with respect to the loss $\mathcal{L}_{\text{lr}}$

$$\mathcal{L}_{\text{lr}}(\eta, \hat{\eta}(z;\psi)) = \frac{1}{2}(\eta - \hat{\eta}(z;\psi))^2 \tag{7.5}$$

where η is the true optimal learning rate, and $\hat{\eta}(z;\psi)$ the predicted learning rate for the current gradient. To optimize parameters ψ via gradient descent, we would need access to the true learning rate η

$$\frac{\partial \mathcal{L}_{\text{lr}}}{\partial \psi} = -(\eta - \hat{\eta}(z;\psi))\frac{\partial \hat{\eta}(z;\psi)}{\partial \psi} \tag{7.6}$$

which is unknown to us. However, by comparing the gradient of the current time step with the gradient of the previous time step, we can determine whether we over or underestimated η. If the gradient has flipped between two consecutive optimization steps, the learning rate was too large, meaning $\eta - \hat{\eta}(z;\psi) > 0$. On the other hand, if the signs of the gradients are the same, then we can most likely increase the learning rate.

With this knowledge at time step t, the memory parameter updates can be approximated as

$$\psi^{t+1} = \psi^t - \xi\ (-(\eta^t - \hat{\eta}(z^t;\psi^t)))\frac{\partial \hat{\eta}(z^t;\psi)}{\partial \psi^t} \tag{7.7}$$

$$\approx \psi^t + \xi\ \text{sign}\left(z^{t+1}z^t\right)\frac{\partial \hat{\eta}(z^t;\psi^t)}{\partial \psi} \tag{7.8}$$

where ξ is a step size parameter for the gradient descent on ψ. The exact form of this gradient update depends on what parametric form $\hat{\eta}(z;\psi)$ takes.

7.2.2 Learning Rate Memory Representation

As mentioned previously, we aim at developing an algorithm that can continuously update the learning rate memory h. Designing the function approximator h, such that forgetting of previously learned parameters is minimized, is one of the challenges of this approach. Here we choose to use locally weighted regression (Schaal et al. 1998), which is known for computational efficiency and which can increase model complexity when necessary. With locally weighted regression we decompose the memory h into M local models h_m, each parametrized by their own parameters ψ_m.

Furthermore, in locally weighted regression, the loss function $\mathcal{L}_{lr}$ also decomposes into M separately weighted losses, each dependent on only their respective parameters ψ_m:

$$\mathcal{L}_{lr}(\psi) = \sum_{m=1}^{M} \mathcal{L}_m(\psi) = \sum_{m=1}^{M} \zeta_m(z)(\eta - \hat{\eta}_m(z;\psi_m))^2 \tag{7.9}$$

where $\zeta_m(z)$ is the weighting function that defines the active neighborhood for each local model. A standard selection of this weight function is the squared exponential kernel $\zeta_m(z) = \exp\left(-0.5\frac{(z-c_m)^2}{\lambda_m^2}\right)$.

Using this memory representation leads to the following update rule per local model

$$\psi_m^{t+1} = \psi_m^t + \xi \ \mathrm{sign}\left(z^{t+1}z^t\right) \zeta_m(z^t) \frac{\partial \hat{\eta}_m(z^t;\psi_m^t)}{\partial \psi_m^t}. \tag{7.10}$$

Note, how only local models that are sufficiently activated require updating. Finally, at prediction time the predicted learning rate is a weighted average over all local models' predictions:

$$\hat{\eta} = \frac{\sum_{m=1}^{M} \zeta_m(z)\hat{\eta}_m(z;\psi_m)}{\sum_{m=1}^{M} \zeta_m(z)} \tag{7.11}$$

We now have an algorithm that can train a model $\hat{\eta}$ to predict a learning rate for one-dimensional optimization problems. An illustration of what kind of learning rate landscapes can be trained with this algorithm is given in the experimental Section in Figure 7.1.

As a final step, we show how this approach can be generalized to models with multiple high-dimensional parameter groups, as commonly encountered in deep learning models.

7.2.3 Multi-Dimensional Learning Problems

Above we have assumed $z = \nabla_\theta \mathcal{L}_{\text{task}}(\theta)$ to be one-dimensional. A key question is how to generalize to optimization tasks that not only have high-dimensional gradients $\boldsymbol{z} = \nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta})$, but also multiple layers of parameters $\mathcal{L}_{\text{task}}(\boldsymbol{\theta}^1,\dots,\boldsymbol{\theta}^K)$, where $\boldsymbol{\theta}^k$ stands for the k^{th} parameter group, as is typical for deep neural networks for instance.

The straightforward extension would be to compute $\text{sign}(z^{t+1}z^t)$ as the inner product $\text{sign}(\boldsymbol{z}^{t+1^T}\boldsymbol{z}^t)$ of two consecutive gradients, and place local models in the D_k-dimensional gradient space. However, this has the following consequences: We would only learn to predict one learning rate per time step, and we lose much information through the inner product of the two gradients. Furthermore, the number of local models needed to cover the gradient space evenly would grow exponentially with the gradient dimension D_k and would significantly increase memory requirements while decreasing computational efficiency.

On the other end of the spectrum we can choose $z = \nabla_{\boldsymbol{\theta}_i}\mathcal{L}_{\text{task}}(\boldsymbol{\theta})$ to be the partial derivative with respect to the i^{th} coordinate, and create a learning rate memory per coordinate of the parameter vector $\boldsymbol{\theta}$. Assuming $\boldsymbol{\theta} \in \mathbb{R}^D$, this would require D memory models, each with M local models. This choice, would create a learning rate prediction per gradient coordinate, and thus offers maximum flexibility. However, it also has a high memory (storage) requirements.

Here we choose a different route. First, we identify natural parameter groups, such as parameters $\boldsymbol{\theta}^k$ of each hidden layer in deep neural networks. For each of these K parameter groups, a memory h^k is created. Then, for the k^{th} parameter group, we pool the updates of all coordinate-wise updates, by computing the average update per local model, across all coordinates

$$\boldsymbol{a}^{t,k}_{m,d} = \text{sign}\left([\boldsymbol{z}]^{t+1}_{d,k}[\boldsymbol{z}]^t_{d,k}\right)\zeta^k_m([\boldsymbol{z}]^t_{d,k}) \tag{7.12}$$

$$\psi^{t+1,k}_m = \psi^{t,k}_m + \xi\,\frac{1}{D_k}\sum_d^{D_k}\big(a^{t,k}_{m,d}\,\frac{\partial\hat{\eta}^k_m([\boldsymbol{z}]^t_{d,k};\psi^{t,k}_m)}{\partial\psi^k_m}\big). \tag{7.13}$$

where $[\boldsymbol{z}]_{d,k}$ means we take the d^{th} coordinate of $\boldsymbol{z} = \nabla_{\boldsymbol{\theta}^k}\mathcal{L}_{\text{task}}(\boldsymbol{\theta}^k)$, and D_k denotes the dimensionality of the k^{th} parameter group $\boldsymbol{\theta}^k$. At prediction time, we similarly make predictions for each parameter group, per gradient coordinate $[\boldsymbol{z}]_{d,k}$ to obtain a learning rate per parameter dimension.

Intuitively, this choice means that parameters within the same parameter group share the same learning rate memory, expecting that they would benefit from the same scaling behavior. With this representation, our meta-learner then requires extra memory resources in the order of $O(KM)$. Furthermore, the computational complexity of the memory update as well as learning rate prediction is in the order of $O(DM)$.

7.2.4 Implementation Details

Finally, to implement this approach, a few design choices have to be made: Each memory is pre-allocated with a fixed number M of local models. Since the localization happens in the space of one-dimensional partial gradients - we linearly space the local models' centers within the range of the minimum and maximum gradient value allowed[1]. The size of each local model is determined by parameters λ_m, which are chosen to create a reasonable amount of overlap between neighboring local models. Thus the more local models we allow, the smaller they become.

Furthermore, we choose local constant models, such that $\hat{\eta}(z;\psi_m) = \psi_m$ in our experiments. Intuitively, this means that each ψ_m corresponds to a learning rate value, localized in gradient space. We further use $z^{t+1}z^t$ and clip the resulting (absolute) value at 1 instead of the $\text{sign}(z^{t+1}z^t)$. We have empirically found that this significantly improves the convergence since memory updates are less pronounced in regions with small gradients. Finally, at the beginning of a learning problem, when no previous memory of learning rates exist, we initialize the memories to predict an initial learning rate η_{init}, thus at the very beginning all $\psi_m = \eta_{\text{init}}$. A reference implementation for our meta-learning approach can be found at (Meier* et al. 2017) in PyTorch (Paszke et al. 2017) and Tensorflow (Abadi et al. 2016).

At runtime, when performing a task-specific learning problem, we then immediately start optimizing all h^k as well. At each time step t, we update both, the parameters of the task-specific problem $\boldsymbol{\theta}$ and the learning rate memory parameters ψ.

On a final note, we want to point out that our presented approach here can be used on top of any base-learner. While not explicitly noted, it is easily possible to first apply some fixed-rule based learner, such as Adam (Kingma et al. 2014) to transform the gradient, and then apply our learned memory evaluation on that transformed gradient. In fact, in our experimental evaluation, we include results for both basic gradient descent with learning rate memory and Adam with learning rate memory.

7.3 Experiments

We evaluate the key properties of our proposed meta-learning algorithm in three different settings. On the Rosenbrock function, a well-known non-convex optimization problem, we illustrate the progression of the internal *memory of learning rates* and show that reoptimizing the function indeed speeds up convergence. Then we show extensive results on two learning tasks: First,

[1]this corresponds to the choice of gradient clipping as is common in Tensorflow implementations

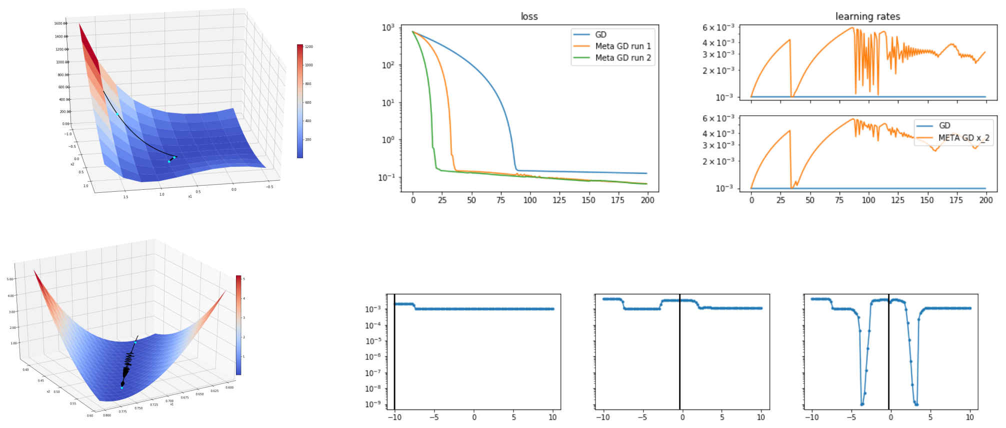


Figure 7.1: top row: (left) The Rosenbrock function with the meta-learners path. (middle) Convergence to the minimum for gradient descent with $\eta = 0.001$, and two consecutive runs of MetaGD with initial $\eta = 0.001$. The first run with a learning rate memory starts with all local models predicting the initial learning rate $\eta = 0.001$, but quickly builds a model of learning rates - thus this already converges faster than without memory. The second run of gradient descent with learning rate memory re-uses and updates the memory from run 1 and converges even faster. (right) the predicted learning rates for both dimensions, as a function of optimization steps. **bottom row**: (left) zoomed in Rosenbrock function to visualize path in the valley. The blue dots (in both top and bottom Rosenbrock plots) indicate where we took a snapshot of the memories. (right) 3 snapshots of the memories, with the x-axis representing values of the partial derivatives z, and the y-axis the predicted learning rate values. The black horizontal lines indicate the current partial gradient - at the time of the snapshot.

we investigate learning binary classifiers on the MNIST dataset, both illustrating the effect of transferring the learning rate memory between similar tasks and showcasing robustness to parameter choices. Finally, we extensively evaluate our meta-learning approach on online inverse dynamics learning tasks for manipulators (see Chapter 6). (**Kappler** et al. 2017c) presents additional preliminary results on challenging deep learning models such as recurrent neural networks (RNNs) and generative adversarial networks (GANs) beyond the scope of this thesis.

7.3.1 Baselines and Experimental Setup

As mentioned in Chapter 7.2, our proposed meta-learning can be combined with different base-optimizers. We need to choose an optimizer on both hierarchies, the task-specific optimizer with base learning rate η, and the meta optimizer with base learning rate ξ. We present results for three variants: basic gradient descent without meta-learning (`GD`), gradient descent with meta-learning with either gradient descent to update the memory (`MetaGD`) or with Adam (Kingma et al. 2014) to update the memory (`MetaGDMemAdam`). For results involving neural networks, a memory of learning rates per hidden layer was learned.

We also compare to another meta-learner: L2LBGDBGD (Andrychowicz et al. 2016) is a very recent approach which treats the optimizer itself as a function approximator, typically represented by a two-layer recurrent LSTM network which transforms the gradients directly. L2LBGDBGD does require to learn the optimizer before task-specific learning. For our binary MNIST problem we sample initial neural network configurations (without structural changes) and optimize on the same data as used for evaluation. In order to avoid overfitting, we run a validation epoch with a random network instance after every second optimization. After 16 rounds of optimizing a L2LBGDBGD learner, we take the best L2LBGDBGD network according to the validation evaluation. All of the evaluated approaches, including our own, are based on TensorFlow (Abadi et al. 2016) implementations. We provide a reference implementation of our algorithm on github (Meier* et al. 2017).

7.3.2 Rosenbrock Problem

We start by illustrating our approach on the Rosenbrock problem, as visualized in Figure 7.1(left). Rosenbrock objective is a 2D optimization problem, and each dimension has its own memory of learning rates (as shown in Figure 7.1). We compare the convergence of gradient descent with $\eta = 0.001$ and two consecutive runs of gradient descent with a memory of learning rates, starting from the same initial position. The memories are initialized to predict $\eta = 0.001$ for the first optimization run. For the second run, we carry over the memories from the previous run.

The middle plot (top row) of Figure 7.1 shows the convergence of each of these optimization runs. Notice how already the first optimization run benefits from the online meta-learning. The second run converges even faster. The right plot (top row), shows the corresponding learning rates of both dimensions applied at each iteration of the second optimization run.

The bottom row shows a zoomed in Rosenbrock function, centered around the path taken by the meta-learner. The blue dots indicate where we have taken a snapshot of the learning rate memories. In the bottom right - we show the memory of the second dimension, at those three different time steps during the first run. We see how the memories initially learn to predict increased learning rates[2]. Once the optimization hits the valley, it learns to predict larger learning rates as long as the optimization stays within a certain gradient range, at the borders of that gradient range the learning rate is drastically reduced to prevent jumping out of the valley.

[2]Note, we clip the gradients at ± 10, and thus it learns to increase the learning rates even in high curvature areas.

7.3.3 Binary MNIST Problems

For both evaluations, sequential batch binary MNIST and sequential online binary MNIST we use a neural network with the following structure: An input layer that takes the image as input; two convolutional layers followed by a max pooling operation; two densely connected layers with dropout follow this; finally, the output layer is another dense layer. All hidden units activation functions are rectified linear units. The loss function is the softmax cross entropy loss. Please, see Chapter 2.2.2.4 for more information on the neural network layers. For all meta-learning variants, the memories (one per hidden layer) allocate a total of $M = 100$ local models.

7.3.4 Sequential Batch Binary MNIST Problem

This experiments analyzes the effect of *meta-learning*, transfer of our *memory of learning rates*, to speed up subsequent binary classification tasks. Problems like grasp prediction (Chapter 5) are related to this experimental setup. For instance, in case new object models become available and extend the grasp database, existing *learned* grasp predictors should be updated faster using our *meta-learning* approach.

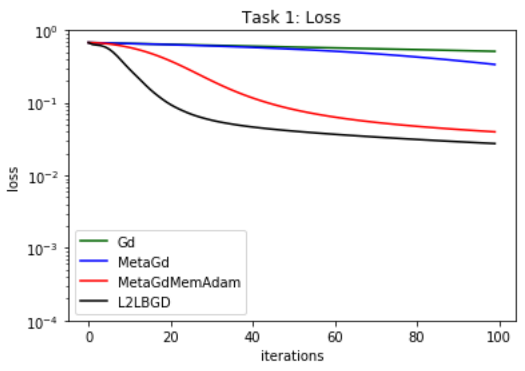
The left three plots of Figure 7.2 illustrates the loss evolution when optimizing the learning tasks in *batch-mode*. In green, we see vanilla gradient descent without any memory of learning rates. The convergence of `MetaGD` and `MetaGDMemAdam` are shown in blue and red respectively. For Task 1, no previous memory exists. Thus all optimizers start with the same learning rate (except `L2LBGDBGD` which does not have an initial learning rate parameter). The meta variants converge faster than vanilla gradient descent. In task 2 we deploy each meta-variant with a new memory and with the memory trained in task 1. Meta-variants with the new memory again convergence faster than vanilla gradient descent; meta-variants initialized with the previously learned memory converge even faster. Notice, that we continue to update the memory of learning rates during task 2 optimization. For task 3 we see the same convergence behavior as for task 2. We also compare against `L2LBGDBGD`. This learned optimizer does not have a base learning rate. It requires costly training on similar network instantiations before usage. However, it requires a (meta) learning rate to train the optimizer. Here we chose to depict the best `L2LBGDBGD` optimizer we could train, achieved with a (meta) learning rate of 0.01. Furthermore, the `L2LBGDBGD` optimizer was explicitly trained for each task. Notice, on task 1 `L2LBGDBGD` achieves faster convergence. However, any subsequent learning tasks achieve similar convergence to the `MetaGDMemAdam`-variant.

Note - we have achieved similar results when using Adam as the base optimizer. Adam itself, without any meta-learner, performs very well on this task and can outperform all of the above meta-learning variants (including `L2LBGDBGD`). However, when combining Adam with our meta-learner, we achieve even better results. This confirms that our online meta-learning approach is versatile and can speed up convergence even when combined with an adaptive optimizer such as Adam. However, Adam does not perform well in sequential (online) learning problems such as the inverse dynamics task discussed in Chapter 7.3.6.2.

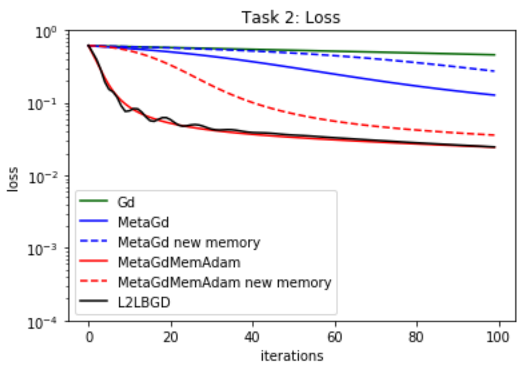
In the rightmost plot of Figure 7.2, we depict how many iterations each optimizer variant requires to achieve an error below 0.1, as a function of initial learning rate values. These results are averaged across three random seeds. Note `L2LBGDBGD` is constant because after having trained no initial learning rate parameter is required. Here we include Adam, with and without meta-learning. In green, we see gradient descent `GD` and `MetaGDMemAdam`, while in blue we see `Adam` and `MetaGDMemAdam`. On average our meta-learning variants achieve a low error faster, irrespective of the initial learning rate, while gradient descent does not converge within the first 100 iterations, and Adam performs well if the learning rate is chosen high enough. This result confirms that at each iteration, our meta-learning approach utilizes the observed data more efficiently and as a result leads to faster learning progress.

7.3.5 Sequential Online Binary MNIST Problem

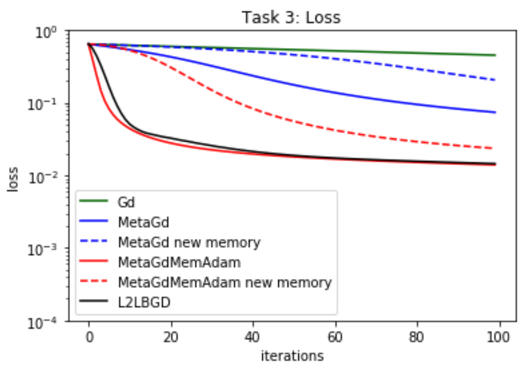
Another essential variant for robotics is online learning - training on one data point at the time. Here, each binary classification task is trained in this fashion. With each new data sample, we update the neural network and the meta-learner and evaluate the test classification rate. For the second (1 vs. 3) and third (1 vs. 4) learning task we can choose to initialize a new meta-learner or reuse the previously trained one and continue to update it. Figure 7.3 shows the convergence of the test classification results. We compare to stochastic gradient descent and Adam with a base learning rate $\eta = 0.001$. For all three tasks, stochastic gradient descent with meta-learning results in high test accuracy with significantly less observed data points. Furthermore, when reloading the meta-learner for task 2 and 3, sample efficiency is improved even further (see three leftmost plots). Results were obtained by averaging over five randomly seeded runs - randomizing both the training data sequence and initialization of the network. On the right-hand side of Figure 7.3 we summarize how many neural network updates (e.g., observed data samples) it took to reach 98% test accuracy, averaged across five randomly seeded runs. The meta-learning with re-loading by far outperforms non-meta-learning variants. This experiments presented promising results for robotic manipulation applications. For instance, it might enable to learn new object classifier from a few observations or demonstrations, or adapt *learned* inverse dynamics models online (Chapter 6).



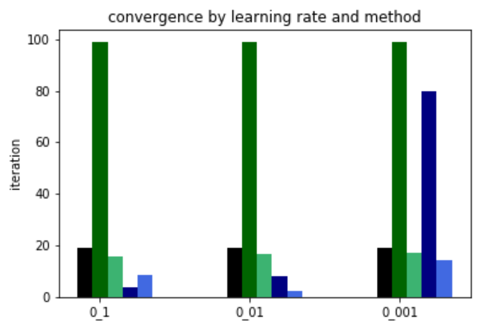
(a)



(b)

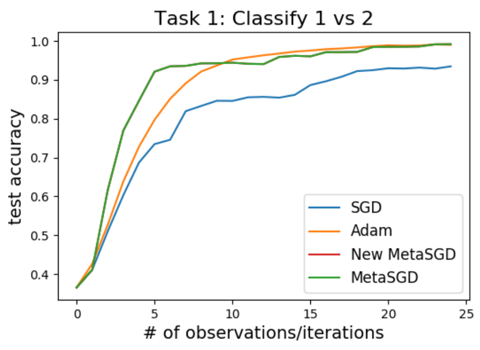


(c)

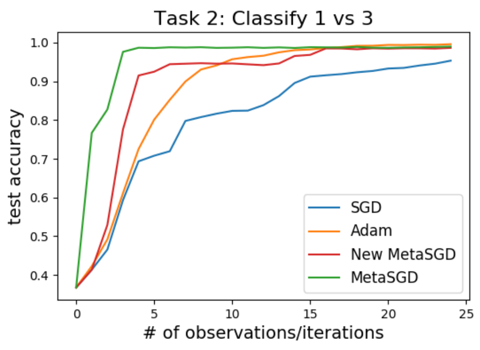


(d)

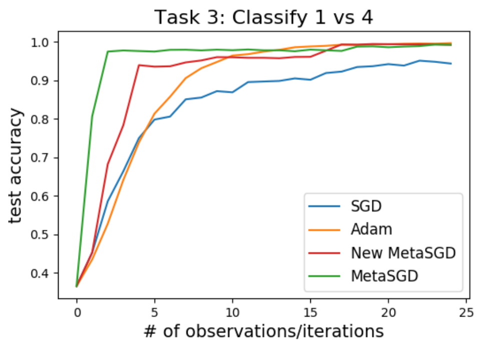
Figure 7.2: Results on the three binary MNIST learning tasks (a-c) convergence behavior of one randomly seeded (same for all optimizers) network on the three subsequent learning tasks with gradient descent based optimizers. (d): the average number of iterations it took for the loss to drop below 0.1 on task 1 for different initial learning rate choices ($[0.1, 0.01, 0.001]$). Results for meta-variants (from left to right: `L2LBGDBGD`, `GD`, `MetaGD`, `Adam`, `MetaGDMemAdam`) are obtained with new memories that have been initialized with the respective learning rate choice.



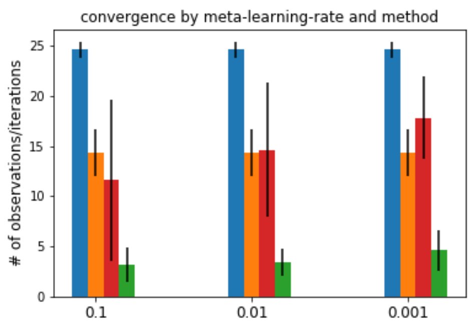
(a)



(b)



(c)



(d)

Figure 7.3: Results on the 3 binary MNIST learning tasks (a-c) Test accuracy as a function of observed data points with gradient descent based optimizers. (d): average number of iterations it took for the test accuracy to reach 98% accuracy on the final task for different meta-learning-rate choices ($[0.1, 0.01, 0.001]$). Learning rate of SGD and Adam is $\eta = 0.001$. Meta-learning with reloading (green) is the most sample-efficient - leading to a good predictive model faster.

7.3.6 Inverse dynamics learning

Inspired by our results on learning task-specific inverse dynamics models (Chapter 6), in our final set of experiments, we explore the use of our meta-learning variants to improve learning convergence. Learning inverse dynamics is a function approximation problem that maps the current joint position, velocities and accelerations $(q, \dot{q}, \ddot{q})$ to torques (τ). Please see Chapter 6 for a more detailed introduction to *learning* inverse dynamics models. Here we consider two possibilities of processing such a high-frequency data stream: we either collect task-specific data and update a task-specific inverse dynamics model in a batch setting (Chapter 7.3.6.1); or we optimize the model online, directly on the streaming data (Chapter 7.3.6.2). We use a similar experimental setup as described in Chapter 6, learn one model per joint for the right arm.

7.3.6.1 Batch Learning

For this experiment we collect data of two motion tasks (in simulation): Task 1 corresponds to acceleration policies (Ijspeert et al. 2013) trained to move along a rectangular path in the horizontal plane, while task 2 corresponds to the same rectangular path in the vertical plane. These movements are performed under strong perturbations of the assumed *model-based* rigid body dynamics inverse dynamics model. Data collection consists of recording joint position, velocities, accelerations, and torques at each time step (1ms). Each task is trained on 10000 collected data points. Per task, we train a neural network per joint in *batch-mode*, meaning that the full dataset is used for each optimization step. The neural network structure has 3 densely connected hidden layers with $[100, 50, 10]$ hidden units, respectively. All activation functions are rectified linear units and dropout before the output layer. The loss function is the mean squared error (MSE) on the predicted torque values. Different from Chapter 6, we learn a full inverse dynamics model, meaning we do not use the underlying rigid body dynamics model for prediction. We report the error on the collected dataset and not apply the *learned* model as presented in Chapter 6.

We perform the two learning tasks in sequence: First, we train a dynamics model on task 1, then we learn a new (uninitialized) dynamics model for task 2. The convergence of each optimization task is shown in Figure 7.4. Notice how using and optimizing our meta-learner while optimizing the inverse dynamics model for task 1 already leads to faster convergence. When transferring the meta-learner for task 2 learning convergence to a low error is even faster. Thus, improving the convergence by re-using the meta-optimizer while adapting to new tasks allows performing faster incremental learning of task-specific inverse dynamics models.

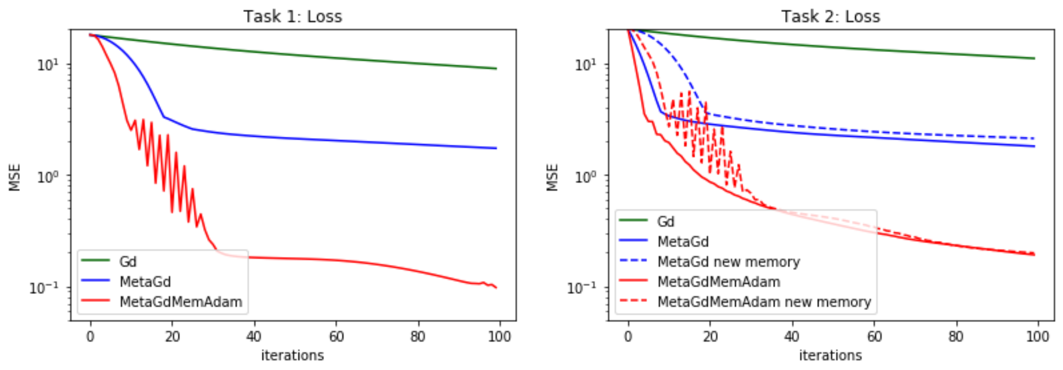


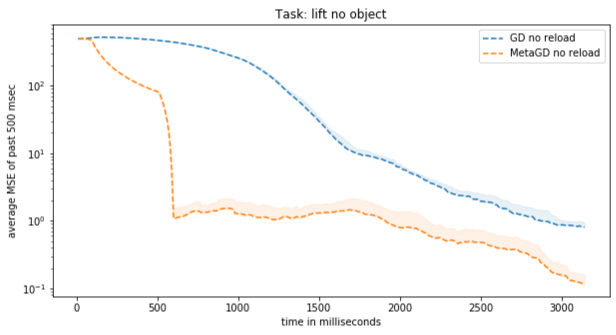
Figure 7.4: Learning task-specific inverse dynamics models in batch mode, on the 1st joint of our 7-DOF manipulator. Meta-Learning helps to convergence faster within task 1 already, when transferring the learned memory of learning rates to task optimization, convergence speed is increased.

7.3.6.2 Sequential Online Learning

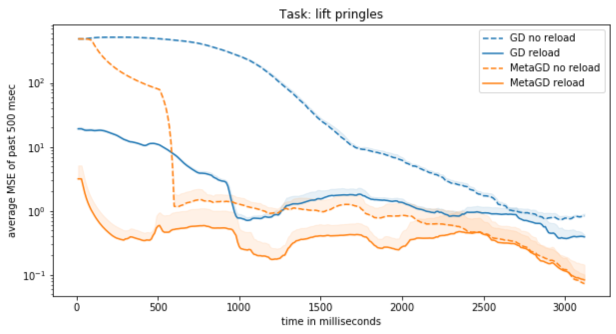
In this experiment, we consider the task of lifting an object with different configurations (no, light, heavy object) on the real manipulation platform. Again, we collect position, velocities, accelerations, and torques at each time step (1ms), resulting in three data sets with more than 3000 samples each. The data was obtained using the *model-based* manipulation system presented in Chapter 3. Further, we execute each task variation 10 times to assess the variance of the learning process. All presented results are averaged across these 10 trials.

This experiment consists of 3 learning phases. First, we train an uninitialized network and meta-learner on the *lifting no object task*. Then this network is further trained on the data corresponding to task-variant 2: *light object*. Finally, the same network is re-used and further adapted on task-variant 3: *heavy lifting*. This scenario tests, how quickly we can adapt previously trained networks, with and without the meta-learner.

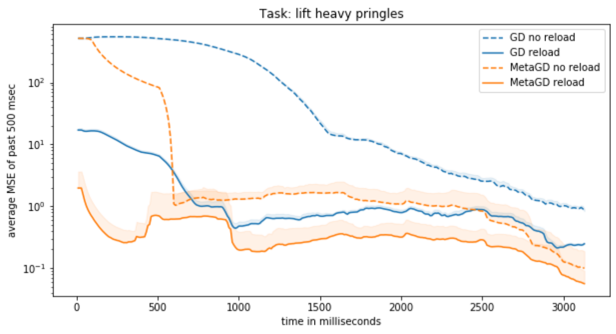
Again, we train one network per joint. We use the same neural network structure as for the *batch learning* experiment, just without the dropout layers. Dropout acts as a regularizer, in online settings such as this experiment each data point is evaluated precisely ones. Thus introducing additional regularization through the noise (dropout) hinders learning and prediction performance of the network. For all experiments, we use $M = 200$ local models for each memory (per layer), a memory learning rate of 0.005, and gradient clipping of 1.0. These values have been determined empirically. We present results for joint 1 since it exhibits large torque variations given payload changes. Notice, the initial high loss values are due to the torque range which is between 30 and 35 N/m hence the model optimized with mean squared error loss has to first adjust for this offset. Different from the previous experiment each learning phase is updating the network online using sequential data batches of size 10, meaning we take data samples collected within the last 10



(a)



(b)



(c)

Figure 7.5: Loss convergence when online learning an inverse dynamics model on the 1st joint of our 7-DOF manipulator. (a) an uninitialized network is updated online with `GD`and `MetaGD`with an uninitialized memory. (b) online learning convergence on task-variant 2 with and without reloading the network and meta-learner from task-variant 1. (c) convergence on task-variant 3 with and without reloading.

milliseconds and perform one optimization step. Hence, the neural network observes every data sample exactly once. This online learning formulation is very challenging since gradients are highly correlated and the overall optimization step should be faster than 10 ms. From our empirical evaluation, we found that `Adam` does not perform well in these very correlated settings which is why we omit the results. Computation times for both `MetaGD` and basic `GD` are less than 3 ms on standard hardware, hence, fast enough to consume a continuous data stream of inverse dynamics data.

In Figure 7.5 we illustrate the convergence of the loss as we are moving along the trajectory of each lifting task when using $\eta = 0.0001$. In the leftmost plot, we see convergence on task 1 of `GD` and `MetaGD`. The middle plot shows convergence on task 2 (light object), comparing learning from scratch (no reloading of network or meta-learner) with continual learning of the previously learned network (reload), for both basic `GD` and `MetaGD`. As we can see, when reloading, we immediately start with a lower error for both `GD` and `MetaGD`. With `MetaGD` we reduce the experienced error faster. Note, even without reloading `GD` and `MetaGD` (net and memory uninitialized) manage to learn a good model by the end of the movement. However, they produce larger prediction errors at the beginning of the movement. Similar behavior can be observed in the final task of lifting the heavy object. Again, similar to the previous experiment we report the prediction error on the offline dataset and do not apply the model on the real robot platform as done in Chapter 6.

We performed this experiment for $\eta = [0.01, 0.001, 0.0001]$ and summarize the mean loss for the first 500 ms of the task execution of joint 1 in Table 7.1. We are mostly interested in assessing fast convergence, therefore the focus on the beginning of the task. The data clearly illustrates the benefit of using our proposed `MetaGD`, that converges faster, meaning using less data, in almost all settings. Interesting to note is that with a really high learning rate all methods achieve excellent results, e.g. 0.01 heavy, yet, due to the high learning rate there is a high chance that some gradient in the sequence will drive the model into a “bad” parameter space, resulting in poor convergence (0.01 no object).

7.4 Summary and Review

Inspired by the *learning-based* extensions to *model-based* robotic manipulation and their learning requirements, we presented a novel meta-learning algorithm in this chapter. Our approach incrementally learns a memory of learning rates as a function of current gradient observations, improving both learning convergence and the number of iterations, thus data efficiency.

η	lift	GD noreload	MetaGD noreload	GD reload	MetaGD reload
0.01	no object	**180.062**	228.421	**180.062**	228.421
	light	234.276	228.704	55.407	**18.926**
	heavy	220.249	167.623	0.034	**0.027**
0.001	no object	119.317	**65.221**	119.317	**65.221**
	light	110.396	63.952	2.233	**0.692**
	heavy	113.62	66.958	0.618	**0.454**
0.0001	no object	468.849	**80.219**	468.849	**80.219**
	light	489.441	78.799	10.712	**0.593**
	heavy	513.898	82.804	6.444	**0.694**
average	no object	256.076	**124.62**	256.076	**124.62**
	light	278.038	123.818	22.784	**6.737**
	heavy	282.589	105.795	2.365	**0.392**

Table 7.1: MSE of the first 500 ms of each task execution.

We have empirically shown that memories of learning rates can be transferred between similar learning tasks, and speed up the convergence of the new – previously unseen – learning problems. However, thus far this effect has been constrained to base optimizers that do not transform the gradient before updating the learning rate memory. Thus, we are interested in investigating whether we can extend the state with which the memory is indexed beyond simple gradient information. The hope would be that this would allow the memory to capture even more complex learning rate landscapes. The challenge here is to do so while maintaining computational efficiency.

Within the context of this thesis, the presented approach complements several contributions. The optimization-based reactive motion generation system (Chapter 3) would benefit from faster planning, allowing to reason about longer time horizons. Grasp prediction (Chapter 5) could be extended to incorporate real-world data to adapt bootstrapped models trained with simulated grasps quickly. Task-specific inverse dynamics discussed in Chapter 6 directly benefits from our formulation as presented in our experimental evaluation.

Another exciting avenue for the presented approach within the context of this thesis would be to use meta-learning for transfer learning. This idea is especially interesting in robotics since real robot experiments are very costly and in general, do not scale in comparison to simulation. Results from simple *model-based* simulations typically do not translate directly to the real world, require fine tuning on additional real-world data. Our meta-learner, however, captures the structure of the optimization process itself. Due to its higher abstraction level, this information hopefully transfers to the real world. Therefore, using our meta-learning, used to train models from *model-based* simulation, should result in less real robot experiments, required to achieve good performance.

Chapter 8

Conclusion

The goal of this thesis was to improve and complement existing *model-based* approaches with *learning-based* methods towards *autonomous manipulation*. Therefore, we developed a state-of-the-art *model-based* manipulation system (Chapter 3) to identify and discuss key problems towards *autonomy*. Truly *autonomous manipulation* systems inevitably have to cope with new, previously unknown tasks. Therefore, we presented a *learning-based* approach (Chapter 4), allowing to teach new capabilities, such as contact interaction tasks, to an existing system by merely demonstrating solutions and stopping execution in failure cases. In Chapter 4 we developed a large-scale simulation-based database to enable state-of-the-art *learning-based* approaches for grasp prediction. We further derived a domain-specific objective to improve grasp prediction for the specific robotic manipulation requirements. This *learning-based* extension complements our *model-based* manipulation system (Chapter 3) and facilitates better use for our *learning* from demonstration approach (Chapter 4). Since manipulation fundamentally requires grasping objects, thus, changing the robots dynamics, we investigated a *learning-based* extension to *model-based* inverse dynamics control (Chapter 6), improving data effeciency and safety. We achieve this safety improvement by enabling learning from data obtained with low-gain feedback control trajectory tracking, compliant systems. Finally, we contributed a new *learning-based* optimizer (Chapter 7), showing promising results to speed up our reactive *model-based* system (see Chapter 3) and our *learning-based* extensions (Chapter 5 and 6).

8.1 Scientific Contributions of the Thesis

Chapter 3, Real-Time Perception meets Reactive Motion Generation: We discussed the design, requirements, and implementation of a fully integrated manipulation system, presenting the first coherent empirical results, demonstrating the importance of continuous, real-time perception and its tight integration with reactive motion generation methods in dynamic manipulation scenarios. We extensively evaluate the purely *model-based* system on a real robotic platform.

Albeit our system is still limited to a rather simple pick and place tasks, our four experimental scenarios are more challenging compared to previous work due to complex workspace geometries and dynamic environments and targets. This work shows the performance which can be achieved with a state-of-the-art *model-based* manipulation system using fast feedback in combination with slower reactive planning and therefore functions as a qualitative performance baseline for *end-to-end learning* approaches towards *autonomous manipulation*.

Chapter 4: Online Decision Making for Manipulation: We contributed a novel approach to *learn* new complex contact rich manipulation tasks from demonstrations. Different from prior work, our method makes use of many available sensor streams during task execution, deciding online *when* and to *which* successive behavior to switch, to ensure successful task execution. The approach exploits our domain specific problem understanding that movement generation can dictate sensor feedback. Our experiments show that this approach can be used to teach dexterous, contact rich tasks, such as unscrewing and removing the cap of a bottle, a bimanual manipulation task performed on a real platform that requires precise manipulation of relatively small objects. Our learned system is robust against perturbation and sensor noise because our method decides *online* whether or not to switch to alternative ASMs due to unexpected sensory signals. In summary, this work demonstrated that we could *learn* simple *models* for various sensor modalities, which can be evaluated in real-time, to make decisions about the performance of the currently executed task. To *learn* a new task we only need in the order of five demonstration and simple user interventions in case of failures to automatically improve the task robustness.

Chapter 5, Big-Data Grasping: Good grasping metrics and realistic simulation are essential for *model-based* grasping, rendering their online usage on real systems impracticable. State-of-the-art *learning-based* methods further alleviate the requirement that the target object has to be known a priory. However, such methods rely on large datasets to achieve good predictive performance. We contribute the first large-scale database for grasping and evaluate our new physics-based quality metric using a human user study. This study showed that our physics simulation based metric is a better predictor for grasp success compared to the standard ε-metric, enabling our automatic high-quality, large-scale data collection effort. This database enabled the usage of state-of-the-art *learning-based* approaches. We empirically verified that high capacity *learning* methods indeed benefit from our rich dataset with high perceptual variance. We further presented a domain-specific objective directly optimizing for the top-1 grasp, better representing the requirements for robotic grasping. An efficient optimization algorithm was proposed to optimize this non-convex objective, and our empirical results demonstrated that this new objective indeed outperforms traditional classification objectives for learning to predict grasp poses.

Chapter 6, Inverse Dynamics Learning: Inherently safe robot control is crucial for collaborative humanoid robotic manipulation tasks. Therefore, we need methods to precisely track desired trajectories while maintaining system compliance. We presented a new method that extends existing *model-based* inverse dynamics controllers with *learned error models*, translating desired accelerations into torques. Different from existing methods our approach allows to quickly improve an existing inverse dynamics model with a *learned* error model from off trajectory data. Leveraging off trajectory data is essential for inverse dynamics learning on compliant, inherently safe systems since they cannot track trajectories well if the system payload changes or they encounter *model* inaccuracies. We achieve this by deriving an additional training signal — measured at the desired accelerations — complementing the existing data source, using our domain-specific knowledge. We extensively analyze the use of both data sources in simulation and demonstrate its effectiveness on a real-world robotic platform. We show that the system incrementally improves the learned inverse dynamics model. By combining both data sources, we further show that *error model learning* converges more consistently and faster.

Chapter 7, Learning to Learn While Learning: The ability to quickly learn new tasks while leveraging prior experiences is essential towards *autonomous manipulation*. We contribute a new online *meta-learning* approach, tackling this problem, inspired by the remarkable human ability to quickly acquiring new, or, adapting existing skills. We show that our method allows for the first time to learn the robot dynamics online and transfer learned structure to new tasks, resulting in faster convergence. The proposed meta-learning approach can be combined with any gradient-based optimizer, learns on the fly and can be transferred to new optimization tasks. Thus, any repetitive gradient-based optimization problem can benefit from our novel *learning-based* optimizer. Additionally, our approach significantly reduces the amount of required observed data samples. This improved data efficiency in combination with the improved convergence speed opens up a promising new avenue for simulation to real transfer by learning how to improve convergence and sample efficiency in simulation and therefore reducing the amount of real-world data acquisition.

8.2 Limitations and Extensions

Within this thesis we have investigated how to improve *model-based* manipulation towards *autonomous manipulation* using *learning-based* extensions. Our purely *model-based* reactive system (Chapter 3) exhibits state-of-the-art performance for dynamic manipulation tasks in uncertain and dynamic environments. In order to achieve good performance, this system relies on several key assumptions: (i) All objects the robot should interact with have to be known, can be identified and a good initialization has to be provided to the object tracker. (ii) We assume that during task execution the object is always visible and the object tracking algorithm does not lose track. (iii) The overall task is simple enough that we can provide a cost function for reactive planning, an observation model for tracking and grasp poses for a simple grasp controller with a heuristic grasp success detection. (iv) We require good trajectory tracking to achieve good pro-active obstacle avoidance. In Chapter 5 we present a *learning-based* grasp pose predictor which overcomes the requirement for knowing the object a priory. Chapter 4 provides a framework which allows to further relax the requirement for a robust hand designed grasp controller by using *learning from demonstration*. However, it is still unclear how to generically define tasks for a *model-based* framework in a scalable manner, allowing to learn new objectives and fully integrate all components into one system. Credit assignment in case of failures is another open research topic which poses significant challenges for fully integrated systems towards *autonomous manipulation*. To be more concrete, in the case of a failed pick and place attempt, it is not yet possible to autonomously infer if either object tracking, or robot trajectory tracking or the grasp controller or all components have to be improved.

In Chapter 4, we carefully analyzed how to use *learning from demonstration* to teach new contact rich manipulation tasks. This approach relies on the assumption that a task is executed in a stereotypical manner and sensory measurements can be modeled with a uni-modal distribution. However, using multi-modal distributions for both the expected sensory events and failure cases would further improve the robustness of the presented approach and alleviate the dependency of the failure propagation based on the skill progress. Recent results on learning distributions using normalizing flows (Rezende et al. 2015) seem promising towards learning a rich multi-modal sensor representation from unsupervised data which our online decision-making process could leverage.

Our work on *learning-based* grasping, discussed in Chapter 5, predicts grasp success for grasp poses extracted from partial sensory observations. However, for true *autonomy*, grasp prediction has to be task specific. For instance, if the task is to drill a hole, only grasps which afford the necessary tool usage to fulfill this task should be considered. Therefore, additional research is required to condition *learned* state-of-the-art deep learning based grasp pose predictors for task-specific grasping. Further, although the presented work has been analyzed in an empirical human study, we still have to evaluate the real world performance of the learned grasp predictor, trained with data from our simulation-based database.

Our work on task-specific dynamics learning (Chapter 6) assumes that we learn a local inverse dynamics model and all information required for the task can be extracted from the state space representation. Therefore, incorporating additional sensor information, for example, force/torque measurements opens up new research directions to generalize learning inverse dynamics for complex interaction tasks. Since inverse dynamics is used to control the robot, exerts forces on the world and achieve desired accelerations, the inherent safety of the robot is in question if *learned* inverse dynamics model harms the system performance. Hence, it is essential to extend existing inverse dynamics learning approaches with confidence estimates, allowing to ignore dire predictions. We believe that using fast-feedback as a signal to obtain a confidence estimate is a promising research direction to use existing non-probabilistic methods, such as deep neural networks used in Chapter 6, in a safe manner.

As shown in this thesis, leveraging fast feedback loops and slower planning over longer time horizons and *learning* are promising approaches towards *autonomous manipulation*. However, for real autonomy, we need to strive towards a system which acts holistically, its components complement and benefit each other and errors are automatically detected, attributed to its components, and automatic improvements are made. Further, additional research towards continual learning is necessary to allow for updating predictive models, within different system components, with new data without forgetting, e.g., previously learned skills. Finally, current state-of-the-art large-scale learning approaches in robotics require too much data in order to be sufficient for the breadth of real-world robotics manipulation tasks. Data acquisition for tasks which do not lend themselves for self-supervision, e.g., desk cleaning or tool usage, and therefore require supervision, labeling or task-specific programming, fundamentally limit the usability of such learning methods. However, simulation allows automating all of these aspects. Unfortunately, results from imprecise simulation often do not transfer the real world. We believe that meta-learning (Chapter 7) is a promising research direction to bridge this gap, allowing to explore learning manipulation systems in simulation and better transfer their performance to the real-world with fewer data samples.

List of Figures

List of Tables

List of Algorithms

Own Publications

J. Bohg, D. **Kappler**, and S. Schaal (May 2016), “Exemplar-based prediction of global object shape from local shape similarity”, *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3398–3405.

R. Gadde, V. Jampani, M. Kiefel, D. **Kappler**, and P. V. Gehler (Oct. 2016), “Superpixel Convolutional Networks Using Bilateral Inceptions”, en, *European Conference on Computer Vision (ECCV)*, Springer, Cham, pp. 597–613.

D. **Kappler**, J. Bohg, and S. Schaal (May 2015b), “Leveraging Big Data for Grasp Planning”, *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4304–4311.

D. **Kappler**, L. Chang, M. Przybylski, N. Pollard, T. Asfour, and R. Dillmann (Dec. 2010), “Representation of pre-grasp strategies for object manipulation”, *IEEE-RAS International Conference on Humanoid Robots*, pp. 617–624.

D. **Kappler**, L. Y. Chang, N. S. Pollard, T. Asfour, and R. Dillmann (Mar. 2012), “Templates for pre-grasp sliding interactions”, *Robotics and Autonomous Systems*, vol. 60, no. 3, pp. 411–423.

D. **Kappler**, F. Meier, J. Issac, J. Mainprice, C. G. Cifuentes, M. Wüthrich, V. Berenz, S. Schaal, N. Ratliff, and J. Bohg (2018), “Real-time Perception meets Reactive Motion Generation”, *IEEE Robotics and Automation Letters (RA-L)*.

D. **Kappler**, F. Meier, and S. Schaal (2017c), “Learning to Learn while Learning”, *NIPS Workshop on Meta-Learning*.

D. **Kappler**, P. Pastor, M. Kalakrishnan, M. Wüthrich, and S. Schaal (2015c), “Data-Driven Online Decision Making for Autonomous Manipulation”, *Robotics: Science and Systems (R:SS)*.

D. **Kappler**, S. Schaal, and J. Bohg (May 2016), “Optimizing for what matters: The top grasp hypothesis”, *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2167–2174.

D. **Kappler***, F. Meier*, N. Ratliff, and S. Schaal (Oct. 2017a), "A New Data Source for Inverse Dynamics Learning", *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, (* both authors contributed equally).

D. **Kappler***, F. Meier*, and S. Schaal (2018), "Online Learning of a Memory for Learning Rates", *IEEE International Conference on Robotics and Automation (ICRA)*, (* both authors contributed equally).

D. **Kappler***, S. Prokudin*, S. Nowozin, and P. Gehler (Sept. 2017b), "Learning to Filter Object Detections", *German Conference on Pattern Recognition (GCPR)*, (* both authors contributed equally), pp. 52–62.

A. Kloss, D. **Kappler**, H. P. A. Lensch, M. V. Butz, S. Schaal, and J. Bohg (Oct. 2016), "Learning where to search using visual attention", *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5238–5245.

C. Lassner, D. **Kappler**, M. Kiefel, and P. Gehler (2016), "Barrista: Caffe Well-Served", *Proceedings of the 2016 ACM on Multimedia Conference*, ACM, pp. 1210–1213.

F. Meier, D. **Kappler**, N. Ratliff, and S. Schaal (Oct. 2016), "Towards robust online inverse dynamics learning", *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4034–4039.

N. D. Ratliff, J. Issac, and D. **Kappler** (Jan. 2018), "Riemannian Motion Policies", *ArXiv e-prints: 1801.02854*, arXiv: `1801.02854 [cs.RO]`.

N. Ratliff, F. Meier, D. **Kappler**, and S. Schaal (Dec. 2016), "DOOMED: Direct Online Optimization of Modeling Errors in Dynamics", *Big Data*, vol. 4, no. 4, pp. 253–268.

C. Rubert, D. **Kappler**, A. Morales, S. Schaal, and J. Bohg (Sept. 2017), "On the relevance of grasp metrics for predicting grasp success", *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

S. Ulbrich, D. **Kappler**, T. Asfour, N. Vahrenkamp, A. Bierbaum, M. Przybylski, and R. Dillmann (2011), "The OpenGRASP benchmarking suite: An environment for the comparative analysis of grasping and dexterous manipulation", *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1761–1767.

F. Widmaier, D. **Kappler**, S. Schaal, and J. Bohg (May 2016), "Robot arm pose estimation by pixel-wise regression of joint angles", *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 616–623.

M. Wüthrich, J. Bohg, D. **Kappler**, C. Pfreundt, and S. Schaal (May 2015), “The Coordinate Particle Filter - a novel Particle Filter for high dimensional systems”, *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2454–2461.

M. Wüthrich, S. Trimpe, C. Garcia Cifuentes, D. **Kappler**, and S. Schaal (Dec. 2016b), “A New Perspective and Extension of the Gaussian Filter”, *The International Journal of Robotics Research (IJRR)*, vol. 35, no. 14, pp. 1731–1749.

Bibliography

M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng (2016), "Tensorflow: Large-scale machine learning on heterogeneous distributed systems", *arXiv e-prints:1603.04467*.

N. Aghasadeghi and T. Bretl (2011), "Maximum entropy inverse reinforcement learning in continuous state spaces with path integrals", *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1561–1566.

N. T. Alberto, M. Mistry, and F. Stulp (2014), "Computed torque control with variable gains through Gaussian process regression", *IEEE-RAS. International Conference on Humanoid Robots (Humanoids)*, IEEE, pp. 212–217.

F. Allgöwer, R. Findeisen, and Z. K. Nagy (2004), "Nonlinear model predictive control: From theory to application", *J. Chin. Inst. Chem. Engrs*, vol. 35, no. 3, pp. 299–315.

A. D. Ames and M. Powell (2013), "Towards the Unification of Locomotion and Manipulation through Control Lyapunov Functions and Quadratic Programs", *Control of Cyber-Physical Systems: Workshop held at Johns Hopkins University, March 2013*, ed. by D. C. Tarraf, Heidelberg: Springer International Publishing, pp. 219–240.

C. H. An, C. G. Atkeson, and J. M. Hollerbach (1985), "Estimation of inertial parameters of rigid body links of manipulators", *IEEE Conference on Decision and Control (CDC)*, pp. 990–995.

S. Andrews, I. Tsochantaridis, and T. Hofmann (2002), "Support Vector Machines for Multiple-instance Learning", *Proceedings of the 15th International Conference on Neural Information Processing Systems*, NIPS'02, Cambridge, MA, USA: MIT Press, pp. 577–584.

M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, and N. de Freitas (2016), "Learning to learn by gradient descent by gradient descent", *Advances In Neural Information Processing Systems*, pp. 3981–3989.

M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. Pieter Abbeel, and W. Zaremba (2017), "Hindsight Experience Replay", *Advances in Neural Information Processing Systems (NIPS)*, ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Curran Associates, Inc., pp. 5048–5058.

A. Angelova, A. Krizhevsky, V. Vanhoucke, A. S. Ogale, and D. Ferguson (2015), "Real-Time Pedestrian Detection with Deep Network Cascades", *BMVC*, vol. 2, p. 4.

Archive 3D: Free 35 000+ 3D models (n.d.), `archive3d.net`.

T. Asfour, K. Regenstein, P. Azad, J. Schroder, A. Bierbaum, N. Vahrenkamp, and R. Dillmann (2006), "ARMAR-III: An Integrated Humanoid Platform for Sensory-Motor Control", *IEEE-RAS. International Conference on Humanoid Robots (Humanoids)*, pp. 169–175.

K. J. Astrom and D. B. Wittenmark (2008), *Adaptive Control*, 2nd, Dover.

C. Atkeson, B. Babu, N. Banerjee, D. Berenson, C. Bove, X. Cui, M. DeDonato, R. Du, S. Feng, P. Franklin, M. Gennert, J. Graff, P. He, A. Jaeger, J. Kim, K. Knoedler, L. Li, C. Liu, X. Long, T. Padir, F. Polido, G. Tighe, and X. Xinjilefu (2015), "No Falls, No Resets: Reliable Humanoid Behaviour in the Darpa Robotics Challenge", *IEEE-RAS. International Conference on Humanoid Robots (Humanoids)*, pp. 623–630.

C. G. Atkeson, C. H. An, and J. M. Hollerbach (1985), "Rigid body load identification for manipulators", *IEEE Conference on Decision and Control (CDC)*, vol. 24, pp. 996–1002.

C. G. Atkeson, A. W. Moore, and S. Schaal (1997), "Locally weighted learning for control", *Artificial Intelligence Review*, no. 1-5, pp. 75–113.

P. Azad, T. Asfour, and R. Dillmann (2006), "Combining appearance-based and model-based methods for real-time object recognition and 6d localization", *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

R. Balasubramanian, L. Xu, P. Brook, J. Smith, and Y. Matsuoka (2012), "Physical Human Interactive Guidance: Identifying Grasping Principles From Human-Planned Grasps", *IEEE Transactions on Robotics*, vol. 28, no. 4, pp. 899–910.

A. Balestrino, G. De Maria, and L. Sciavicco (July 1984), "Robust Control of Robotic Manipulators", *IFAC Proceedings Volumes*, vol. 17, no. 2, pp. 2435–2440.

S. H. Bardach and G. D. Rowles (Oct. 2012), "Geriatric education in the health professions: are we making progress?", en, *The Gerontologist*, vol. 52, no. 5, pp. 607–618.

M. Beetz, U. Klank, I. Kresse, A. Maldonado, L. Mösenlechner, D. Pangercic, T. Rühr, and M. Tenorth (Oct. 2011), “Robotic roommates making pancakes”, *IEEE-RAS. International Conference on Humanoid Robots (Humanoids)*, pp. 529–536.

Y. Bekiroglu, D. Kragic, and V. Kyrki (2010), “Learning grasp stability based on tactile data and HMMs”, *IEEE Int. Conf. on Robot and Human Interactive Communication (RO-MAN)*, Viareggio, Italia, pp. 132–137.

R. Bellman (Aug. 1952), “On the Theory of Dynamic Programming”, en, *Proceedings of the National Academy of Sciences of the United States of America*, vol. 38, no. 8, pp. 716–719.

J. L. Bentley (Sept. 1975), “Multidimensional Binary Search Trees Used for Associative Searching”, *Communications of the ACM*, vol. 18, no. 9, pp. 509–517.

P. J. Besl and N. D. McKay (Feb. 1992), “A method for registration of 3-D shapes”, *IEEE transactions on pattern analysis and machine intelligence*, vol. 14, no. 2, pp. 239–256.

A. Bicchi and V. Kumar (Apr. 2000), “Robotic grasping and contact: a review”, *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 1, 348–353 vol.1.

C. Bishop (2007), “Pattern Recognition and Machine Learning (Information Science and Statistics), 1st edn. 2006. corr. 2nd printing edn”, *Springer, New York*.

C. M. Bishop (2006), *Pattern Recognition and Machine Learning*, New York: Springer.

J. Bohg, A. Morales, T. Asfour, and D. Kragic (2014a), “Data-Driven Grasp Synthesis - A Survey”, *IEEE Transactions on Robotics*.

J. Bohg, J. Romero, A. Herzog, and S. Schaal (2014b), “Robot Arm Pose Estimation through Pixel-Wise Part Classification”, *IEEE International Conference on Robotics and Automation (ICRA)*, `github.com/jbohg/render_kinect`, Hongkong, China.

L. Bottou (2010), “Large-Scale Machine Learning with Stochastic Gradient Descent”, *Proceedings of COMPSTAT’2010*, Physica-Verlag HD, pp. 177–186.

L. Bottou, F. E. Curtis, and J. Nocedal (2018), “Optimization Methods for Large-Scale Machine Learning”, *Siam Reviews*, vol. 60, no. 2, pp. 223–311.

G. E. P. Box (Dec. 1976), “Science and Statistics”, *Journal of the American Statistical Association*, vol. 71, no. 356, pp. 791–799.

E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother (2014), “Learning 6D Object Pose Estimation Using 3D Object Coordinates”, *European Conference on Computer Vision (ECCV)*, Springer International Publishing, pp. 536–551.

O. Brock (Dec. 2011), “Berlin Summit on Robotics”, *Conference Report*, Berlin Summit on Robotics.

O. Brock and O. Khatib (2002), “Elastic strips: A framework for motion generation in human environments”, *The International Journal of Robotics Research (IJRR)*, vol. 21, no. 12, pp. 1031–1052.

O. Brock, J. Kuffner, and J. Xiao (2008), “Motion for Manipulation Tasks”, *Springer Handbook of Robotics*, ed. by B. Siciliano and O. Khatib, Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 615–645.

R. Brooks (Mar. 1986), “A robust layered control system for a mobile robot”, *IEEE Journal on Robotics and Automation*, vol. 2, no. 1, pp. 14–23.

R. A. Brooks (1990), “Elephants Don’T Play Chess”, *Robot. Auton. Syst.* Vol. 6, no. 1-2.

B. Burns and O. Brock (Apr. 2007), “Sampling-Based Motion Planning With Sensing Uncertainty”, *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3313–3318.

R. R. Burridge, A. A. Rizzi, and D. E. Koditschek (1999), “Sequential composition of dynamically dexterous robot behaviors”, *The International Journal of Robotics Research (IJRR)*, vol. 18, no. 6, pp. 534–555.

R. Calandra, S. Ivaldi, M. P. Deisenroth, E. Rueckert, and J. Peters (2015), “Learning inverse dynamics models with contacts”, *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3186–3191.

R. Camoriano, S. Traversaro, L. Rosasco, G. Metta, and F. Nori (2016), “Incremental semi-parametric inverse dynamics learning”, *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 544–550.

A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu (2015), *ShapeNet: An Information-Rich 3D Model Repository*, tech. rep. arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago.

F. Chaumette, S. Hutchinson, and P. Corke (2016), “Visual Servoing”, *Springer Handbook of Robotics*, pp. 841–866.

Y. Chebotar, M. Kalakrishnan, A. Yahya, A. Li, S. Schaal, and S. Levine (May 2017), “Path integral guided policy search”, *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3381–3388.

T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang (Dec. 2015), "MXNet: A Flexible and Efficient Machine Learning Library for Heterogeneous Distributed Systems", arXiv: `1512.01274 [cs.DC]`.

S. Chitta, I. Sucan, and S. Cousins (2012), "Moveit! [ROS topics]", *IEEE robotics & automation magazine / IEEE Robotics & Automation Society*, vol. 19, no. 1, pp. 18–19.

C. Choi and H. I. Christensen (May 2010), "Real-time 3D model-based tracking using edge and keypoint features for robotic manipulation", *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4048–4055.

K. M. Choromanski and V. Sindhwani (2017), "On Blackbox Backpropagation and Jacobian Sensing", *Advances in Neural Information Processing Systems 30*, ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Curran Associates, Inc., pp. 6521–6529.

P. Christiano, Z. Shah, I. Mordatch, J. Schneider, T. Blackwell, J. Tobin, P. Abbeel, and W. Zaremba (2016), "Transfer from Simulation to Real World through Learning Deep Inverse Dynamics Model", *arXiv preprint arXiv:1610.03518*.

J. Chung, C. Gulcehre, K. Cho, and Y. Bengio (Dec. 2014), "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling", arXiv: `1412.3555 [cs.NE]`.

D.-A. Clevert, T. Unterthiner, and S. Hochreiter (Nov. 2015), "Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)", arXiv: `1511.07289 [cs.LG]`.

M. Colledanchise, A. Marzinotto, and P. Ögren (May 2014), "Performance analysis of stochastic behavior trees", *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3265–3272.

R. Collobert and J. Weston (2008), "A unified architecture for natural language processing: Deep neural networks with multitask learning", *Int. Conf. on Machine learning*.

N. Correll, K. E. Bekris, D. Berenson, O. Brock, A. Causo, K. Hauser, K. Okada, A. Rodriguez, J. M. Romano, and P. R. Wurman (2016), "Analysis and Observations From the First Amazon Picking Challenge", *IEEE Transactions on Automation Science and Engineering*, no. 99.

C. Cortes and V. Vapnik (Sept. 1995), "Support-vector networks", *Machine learning*, vol. 20, no. 3, pp. 273–297.

E. Coste-Maniere and R. Simmons (2000), "Architecture, the backbone of robotic systems", *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 1, pp. 67–72.

J. J. Craig, P. Hsu, and S. S. Sastry (1987), “Adaptive Control of Mechanical Manipulators”, *The International Journal of Robotics Research (IJRR)*.

J. Crowley (Mar. 1985), “Navigation for an intelligent mobile robot”, *IEEE Journal on Robotics and Automation*, vol. 1, no. 1, pp. 31–41.

C. Daniel, J. Taylor, and S. Nowozin (2016), “Learning Step Size Controllers for Robust Neural Network Training.”, *AAAI*, pp. 1519–1525.

S. Davis and P. Mermelstein (1980), “Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences”, *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 28.

J. S. de la Cruz, E. Calisgan, D. Kulić, W. Owen, and E. A. Croft (2012), “On-line dynamic model learning for manipulator control”, *IFAC Proceedings Volumes*, vol. 45, no. 22, pp. 869–874.

M. Deisenroth, D. Fox, and C. Rasmussen (2013), “Gaussian Processes for Data-Efficient Learning in Robotics and Control”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

J. Deng, W. Dong, R. Socher, L. Li, K. Li, and F. Li (2009), “ImageNet: A large-scale hierarchical image database”, *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Miami, Florida, USA, pp. 248–255.

R. Detry, C. H. Ek, M. Madry, J. Piater, and D. Kragic (2012), “Generalizing Grasps Across Partly Similar Objects”, *IEEE International Conference on Robotics and Automation (ICRA)*.

R. Detry, D. Kraft, A. G. Buch, N. Krüger, and J. Piater (2010), “Refining Grasp Affordance Models by Experience”, *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2287–2293.

D. Dheeru and E. Karra Taniskidou (2017), *UCI Machine Learning Repository*, `http://archive.ics.uci.edu/ml`.

R. Diankov (Aug. 2010), “Automated Construction of Robotic Manipulation Programs”, PhD thesis, CMU, Robotics Institute.

R. Diankov, N. Ratliff, D. Ferguson, S. Srinivasa, and J. Kuffner (2008), “Bispace planning: Concurrent multi-space exploration”, *Robotics: Science and Systems (R:SS)*, vol. 63.

E. D. Dickmanns and B. D. Mysliwetz (Feb. 1992), “Recursive 3-D road and relative ego-state recognition”, *IEEE transactions on pattern analysis and machine intelligence*, vol. 14, no. 2, pp. 199–213.

T. G. Dietterich, R. H. Lathrop, and T. Lozano-Pérez (Jan. 1997), “Solving the multiple instance problem with axis-parallel rectangles”, *Artificial intelligence*, vol. 89, no. 1, pp. 31–71.

M. Dogar and S. Srinivasa (2010), “Push-Grasping with Dexterous Hands: Mechanics and a Method”, *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

J. Duchi, E. Hazan, and Y. Singer (2011), “Adaptive subgradient methods for online learning and stochastic optimization”, *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2121–2159.

C. Eppner, S. Höfer, R. Jonschkowski, R. Martín-Martín, A. Sieverling, V. Wall, and O. Brock (2016), “Lessons from the Amazon Picking Challenge: Four Aspects of Robotic Systems Building”, *Robotics: Science and Systems (R:SS)*.

T. Erez, K. Lowrey, Y. Tassa, V. Kumar, S. Kolev, and E. Todorov (Oct. 2013), “An integrated system for real-time model predictive control of humanoid robots”, *IEEE-RAS. International Conference on Humanoid Robots (Humanoids)*, pp. 292–299.

D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio (2010), “Why does unsupervised pre-training help deep learning?”, *Jour. of Machine Learning Research*.

M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman (June 2010), “The Pascal Visual Object Classes (VOC) Challenge”, *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338.

J. Felip and A. Morales (2009), “Robust sensor-based grasp primitive for a three-finger robot hand”, *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, St. Louis, MO, USA, pp. 1811–1816.

D. Ferguson and A. Stentz (2006), “Using interpolation to improve path planning: The Field D* algorithm”, *Journal Field Robotics*, vol. 23, no. 2, pp. 79–101.

C. Ferrari and J. Canny (1992), “Planning optimal grasps”, *IEEE International Conference on Robotics and Automation (ICRA)*.

C. Finn, P. Abbeel, and S. Levine (2017), “Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks”, *arXiv e-prints:1703.03400*.

C. Finn, P. Christiano, P. Abbeel, and S. Levine (Nov. 2016a), “A Connection between Generative Adversarial Networks, Inverse Reinforcement Learning, and Energy-Based Models”, arXiv: `1611.03852 [cs.LG]`.

C. Finn, S. Levine, and P. Abbeel (Mar. 2016b), “Guided Cost Learning: Deep Inverse Optimal Control via Policy Optimization”, arXiv: `1603.00448 [cs.LG]`.

B. Friedland (2012), *Control system design: an introduction to state-space methods*, Courier Corporation.

J. Fu, Z. Lin, M. Liu, N. Leonard, J. Feng, and T.-S. Chua (2016), "Deep Q-Networks for Accelerating the Training of Deep Neural Networks", *arXiv preprint arXiv:1606.01467*.

A. Gams, M. Do, A. Ude, T. Asfour, and R. Dillmann (Dec. 2010), "On-line periodic movement and force-profile learning for adaptation to new surfaces", *IEEE-RAS. International Conference on Humanoid Robots (Humanoids)*, pp. 560–565.

C. Garcia Cifuentes, J. Issac, M. Wüthrich, S. Schaal, and J. Bohg (2017), "Probabilistic Articulated Real-Time Tracking for Robot Manipulation", *IEEE Robotics and Automation Letters (RA-L)*, vol. 2 (2).

E. Gat (1992), "Integrating Planning and Reacting in a Heterogeneous Asynchronous Architecture for Controlling Real-world Mobile Robots", *National Conference on Artificial Intelligence (AAAI)*, AAAI'92, AAAI Press, pp. 809–815.

A. Gijsberts and G. Metta (2013), "Real-time model learning using Incremental Sparse Spectrum Gaussian Process Regression", *Neural Networks*, vol. 41, pp. 59–69.

C. Goldfeder, M. Ciocarlie, H. Dang, and P. K. Allen (2009), "The Columbia grasp database", *IEEE International Conference on Robotics and Automation (ICRA)*, Kobe, Japan.

I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio (2014), "Generative Adversarial Networks", *ArXiv e-prints*, arXiv: `1406.2661 [stat.ML]`.

I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio (2016), *Deep learning*, vol. 1, MIT press Cambridge.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio (2014), "Generative Adversarial Nets", *Advances in Neural Information Processing Systems (NIPS)*, ed. by Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Curran Associates, Inc., pp. 2672–2680.

P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He (June 2017), "Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour", *ArXiv e-prints*, arXiv: `1706.02677 [cs.CV]`.

A. Gunes Baydin, R. Cornish, D. Martinez Rubio, M. Schmidt, and F. Wood (Mar. 2017), "Online Learning Rate Adaptation with Hypergradient Descent", *ArXiv e-prints*, arXiv: `1703.04782`.

M. Hägele, K. Nilsson, and J. N. Pires (2008), “Industrial Robotics”, *Springer Handbook of Robotics*, ed. by B. Siciliano and O. Khatib, Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 963–986.

S. Hansen (2016), “Using Deep Q-Learning to Control Optimization Hyperparameters”, *arXiv preprint arXiv:1602.04062*.

C. Harris and C. Stennett (Sept. 1990), “RAPID - A Video Rate Object Tracker”, *British Machine Vision Conference*.

K. Hauser and V. Ng-Thow-Hing (May 2010), “Fast smoothing of manipulator trajectories using optimal bounded-acceleration shortcuts”, *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2493–2498.

HDF group - HDF5 (n.d.), `www.hdfgroup.org/HDF5`.

K. He, X. Zhang, S. Ren, and J. Sun (2015), “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification”, *IEEE International Conference on Computer Vision (ICCV)*.

N. Heess, G. Wayne, D. Silver, T. Lillicrap, T. Erez, and Y. Tassa (2015), “Learning Continuous Control Policies by Stochastic Value Gradients”, *Advances in Neural Information Processing Systems (NIPS)*, ed. by C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Curran Associates, Inc., pp. 2944–2952.

D. J. Herzfeld, P. A. Vaswani, M. K. Marko, and R. Shadmehr (2014), “A memory of errors in sensorimotor learning.”, *Science*.

A. Herzog, P. Pastor, M. Kalakrishnan, L. Righetti, J. Bohg, T. Asfour, and S. Schaal (2014), “Learning of Grasp Selection based on Shape-Templates”, *Autonomous Robots*.

A. Herzog, N. Rotella, S. Schaal, and L. Righetti (Nov. 2015), “Trajectory generation for multi-contact momentum control”, *IEEE-RAS. International Conference on Humanoid Robots (Humanoids)*, pp. 874–880.

A. Herzog, N. Rotella, S. Mason, F. Grimminger, S. Schaal, and L. Righetti (Mar. 2016), “Momentum Control with Hierarchical Inverse Dynamics on a Torque-Controlled Humanoid”, *Autonomous Robots*, vol. 40, no. 3, pp. 473–491.

J. Hill and W. T. Park (Mar. 1979), “Real Time Control of a Robot with a Mobile Camera”, *In Proc. 9th ISIR*, pp. 233–246.

S. Hinterstoisser, C. Cagniart, S. Ilic, P. Sturm, N. Navab, P. Fua, and V. Lepetit (May 2012), "Gradient response maps for real-time detection of textureless objects", en, *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 5, pp. 876–888.

G. E. Hinton and R. S. Zemel (1994), "Autoencoders, Minimum Description Length and Helmholtz Free Energy", *Advances in Neural Information Processing Systems 6*, ed. by J. D. Cowan, G. Tesauro, and J. Alspector, Morgan-Kaufmann, pp. 3–10.

T. K. Ho (Aug. 1995), "Random decision forests", *Proceedings of 3rd International Conference on Document Analysis and Recognition*, vol. 1, ieeexplore.ieee.org, 278–282 vol.1.

S. Hochreiter and J. Schmidhuber (Nov. 1997), "Long Short-Term Memory", *Neural computation*, vol. 9, no. 8, pp. 1735–1780.

S. Hochreiter, A. S. Younger, and P. R. Conwell (2001), "Learning To Learn Using Gradient Descent", *Proc. Intl. Conf. On Arti. Neural Networks*.

K. Hornik, M. Stinchcombe, and H. White (Jan. 1989), "Multilayer feedforward networks are universal approximators", *Neural networks: the official journal of the International Neural Network Society*, vol. 2, no. 5, pp. 359–366.

A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard (2013), "OctoMap: An efficient probabilistic 3D mapping framework based on octrees", *Autonomous Robots*, vol. 34, pp. 189–206.

K. Hsiao, S. Chitta, M. Ciocarlie, and E. G. Jones (Oct. 2010), "Contact-Reactive Grasping of Objects with Partial Shape Information", *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Taipei, Taiwan, pp. 1228–1235.

K. Hsiao, P. Nangeroni, M. Huber, A. Saxena, and A. Y. Ng (2009), "Reactive grasping using optical proximity sensors", *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2098–2105.

N. Hudson, J. Ma, P. Hebert, A. Jain, M. Bajracharya, T. Allen, R. Sharan, M. Horowitz, C. Kuo, T. Howard, L. Matthies, P. Backes, and J. Burdick (2014), "Model-based autonomous system for performing dexterous, human-level manipulation tasks", *Autonomous Robots Journal Special Issue : Autonomous Grasping and Manipulation*, vol. 36, no. 1-2, pp. 31–49.

A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal (2013), "Dynamical movement primitives: learning attractor models for motor behaviors", *Neural computation*, vol. 25, no. 2, pp. 328–373.

D. Isla (2005), "Handling Complexity in the Halo 2 AI", *Game Developers Conference (GDC)*.

L. Jamone, B. Damas, and J. Santos-Victor (2014), “Incremental learning of context-dependent dynamic internal models for robot control”, *Proc. of the IEEE International Symposium on Intelligent Control (ISIC)*.

J. Jeon, S. Karaman, and E. Frazzoli (Dec. 2011), “Anytime computation of time-optimal off-road vehicle maneuvers using the RRT*”, *IEEE Conference on Decision and Control and European Control Conference*, pp. 3276–3282.

Y. Jiang, S. Moseson, and A. Saxena (2011), “Efficient Grasping from RGBD images: Learning using a new Rectangle Representation”, *IEEE International Conference on Robotics and Automation (ICRA)*.

T. Joachims (2002), “Optimizing search engines using clickthrough data”, *Int. Conf. on Knowledge Discovery and Data Mining (ACM)*.

M. Johnson, B. Shrewsbury, S. Bertrand, T. Wu, D. Duran, M. Floyd, P. Abeles, D. Stephen, N. Mertins, A. Lesman, J. Carff, W. Rifenburgh, P. Kaveti, W. Straatman, J. Smith, M. Griffioen, B. Layton, T. de Boer, T. Koolen, P. D. Neuhaus, and J. E. Pratt (2015), “Team IHMC’s Lessons Learned from the DARPA Robotics Challenge Trials”, *Journal Field Robotics*, vol. 32, no. 2, pp. 192–208.

M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal (2011a), “STOMP: Stochastic Trajectory Optimization for Motion Planning”, *IEEE International Conference on Robotics and Automation (ICRA)*.

M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal (2011b), “STOMP: Stochastic Trajectory Optimization for Motion Planning”, *IEEE International Conference on Robotics and Automation (ICRA)*.

M. Kalakrishnan, P. Pastor, L. Righetti, and S. Schaal (May 2013), “Learning objective functions for manipulation”, *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1331–1336.

M. Kalakrishnan, J. Buchli, P. Pastor, and S. Schaal (2009), “Learning locomotion over rough terrain using terrain templates”, *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*.

R. E. Kalman (1960), “Contributions to the Theory of Optimal Control”, *Conference on Ordinary Differential Equations*.

T. Kanade, C. Thorpe, and W. Whittaker (1986), “Autonomous Land Vehicle Project at CMU”, *Proceedings of the 1986 ACM Fourteenth Annual Conference on Computer Science*, CSC ’86, New York, NY, USA: ACM, pp. 71–80.

D. **Kappler**, J. Bohg, and S. Schaal (2015a), *Grasp database library and data*, `https://grasp-database.dkappler.de`.

D. **Kappler** and F. Meier (2017a), *The code used to obtain the simulation results.* `https://github.com/dkappler/idsim.git`.

D. **Kappler** and F. Meier (2017b), *Video showing the qualitative experimental results.* `https://vimeo.com/182948172`.

S. Karaman and E. Frazzoli (June 2011), "Sampling-based algorithms for optimal motion planning", *The International journal of robotics research*, vol. 30, no. 7, pp. 846–894.

M. Kazhdan, M. Bolitho, and H. Hoppe (2006), "Poisson surface reconstruction", *Eurographics symposium on Geometry processing (SGP)*, Cagliari, Sardinia, Italy: Eurographics Association, pp. 61–70.

J. Kazmierska and J. Malicki (Feb. 2008), "Application of the Naïve Bayesian Classifier to optimize treatment decisions", *Radiotherapy and oncology: journal of the European Society for Therapeutic Radiology and Oncology*, vol. 86, no. 2, pp. 211–216.

S. M. Khansari-Zadeh and A. Billard (May 2012), "A dynamical system approach to realtime obstacle avoidance", *Autonomous robots*, vol. 32, no. 4, pp. 433–454.

O. Khatib and J. Burdick (Apr. 1986), "Motion and force control of robot manipulators", *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 3, pp. 1381–1386.

O. Khatib (1986), "Real-time obstacle avoidance for manipulators and mobile robots", *The International Journal of Robotics Research (IJRR)*, vol. 5, no. 1.

D. I. Kim and G. S. Sukhatme (May 2014), "Semantic labeling of 3D point clouds with object affordance for robot manipulation", *IEEE International Conference on Robotics and Automation (ICRA)*, ieeexplore.ieee.org, pp. 5578–5584.

D. P. Kingma and J. Ba (Dec. 2014), "Adam: A Method for Stochastic Optimization", *ArXiv e-prints*, arXiv: `1412.6980 [cs.LG]`.

D. P. Kingma and M. Welling (Dec. 2013), "Auto-Encoding Variational Bayes", arXiv: `1312.6114v10 [stat.ML]`.

J. M. Kleinberg (2003), "An Impossibility Theorem for Clustering", *Advances in Neural Information Processing Systems (NIPS)*, ed. by S. Becker, S. Thrun, and K. Obermayer, MIT Press, pp. 463–470.

M. Klingensmith, T. Galluzzo, C. Dellin, M. Kazemi, J. A. (Bagnell, and N. Pollard (May 2013), “Closed-loop Servoing using Real-time Markerless Arm Tracking”, *International Conference on Robotics And Automation (Humanoids Workshop).*

A. Kloss, S. Schaal, and J. Bohg (Oct. 2017), “Combining learned and analytical models for predicting action effects”, arXiv: `1710.04102 [cs.RO]`.

R. Kohavi et al. (1995), “A study of cross-validation and bootstrap for accuracy estimation and model selection”, *International Joint Conference on Artificial Intelligence (IJCAI)*, vol. 14, 2, pp. 1137–1145.

D. Kortenkamp and R. Simmons (2008), “Robotic Systems Architectures and Programming”, *Springer Handbook of Robotics*, ed. by B. Siciliano and O. Khatib, Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 187–206.

D. Kortenkamp, R. G. Simmons, and D. Brugali (2016), “Robotic Systems Architectures and Programming”, *Springer Handbook of Robotics*, pp. 283–306.

D. Kragic, A. T. Miller, and P. K. Allen (2001), “Real-time Tracking Meets Online Grasp Planning”, *IEEE International Conference on Robotics and Automation (ICRA).*

A. Krizhevsky, I. Sutskever, and G. E. Hinton (2012), “ImageNet Classification with Deep Convolutional Neural Networks”, *Advances in Neural Information Processing Systems (NIPS)*, ed. by F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Curran Associates, Inc., pp. 1097–1105.

O. Kroemer, H. van Hoof, G. Neumann, and J. Peters (2014), “Learning to Predict Phases of Manipulation Tasks as Hidden States”, *IEEE International Conference on Robotics and Automation (ICRA).*

Y. Kuwata, J. Teo, G. Fiore, S. Karaman, E. Frazzoli, and J. P. How (Sept. 2009), “Real-Time Motion Planning With Applications to Autonomous Urban Driving”, *IEEE Transactions on Control Systems Technology*, vol. 17, no. 5, pp. 1105–1118.

J. D. Lafferty, A. McCallum, and F. C. N. Pereira (June 2001), “Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data”, *International Conference on Machine Learning (ICML)*, Morgan Kaufmann Publishers Inc., pp. 282–289.

S. M. LaValle (1998), “Rapidly-exploring random trees: A new tool for path planning”.

Q. V. Le, D. Kamm, A. F. Kara, and A. Y. Ng (2010), “Learning to grasp objects with multiple contact points”, *IEEE International Conference on Robotics and Automation (ICRA).*

Y. LeCun (1998), “The MNIST database of handwritten digits.”

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner (Nov. 1998), "Gradient-based learning applied to document recognition", *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324.

Y. LeCun, Y. Bengio, and G. Hinton (May 2015), "Deep learning", en, *Nature*, vol. 521, no. 7553, pp. 436–444.

A. D. Lehmann, P. V. Gehler, and L. J. Van Gool (2011a), "Branch&Rank: Non-Linear Object Detection", *BMVC*, vol. 2, p. 1.

A. D. Lehmann, P. V. Gehler, and L. J. Van Gool (2011b), "Branch&Rank: Non-Linear Object Detection.", *BMVC*.

P. Lehner, A. Sieverling, and O. Brock (May 2015), "Incremental, Sensor-Based Motion Generation for Mobile Manipulators in Unknown, Dynamic Environments", *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4761–4767.

I. Lenz, H. Lee, and A. Saxena (2015), "Deep learning for detecting robotic grasps", *The International Journal of Robotics Research (IJRR)*.

I. Lenz, H. Lee, and A. Saxena (2014), "Deep Learning for Detecting Robotic Grasps", *The International Journal of Robotics Research (IJRR)*.

J. Leonard, J. How, S. Teller, M. Berger, S. Campbell, G. Fiore, L. Fletcher, E. Frazzoli, A. Huang, S. Karaman, et al. (2008), "A perception-driven autonomous urban vehicle", *Journal Field Robotics*, vol. 25, no. 10.

S. Levine, C. Finn, T. Darrell, and P. Abbeel (2016a), "End-to-end Training of Deep Visuomotor Policies", *J. Mach. Learn. Res.* Vol. 17, no. 1.

S. Levine and V. Koltun (Feb. 2013), "Guided Policy Search", *International Conference on Machine Learning (ICML)*, jmlr.org, pp. 1–9.

S. Levine, P. Pastor, A. Krizhevsky, and D. Quillen (2016b), "Learning Hand-Eye Coordination for Robotic Grasping with Deep Learning and Large-Scale Data Collection",

S. Levine, P. Pastor, A. Krizhevsky, and D. Quillen (2017), "Learning Hand-Eye Coordination for Robotic Grasping with Large-Scale Data Collection", *International Symposium on Experimental Robotics*, Springer International Publishing, pp. 173–184.

J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, J. Z. Kolter, D. Langer, O. Pink, V. Pratt, M. Sokolsky, G. Stanek, D. Stavens, A. Teichman, M. Werling, and S. Thrun (June 2011), "Towards fully autonomous driving: Systems and algorithms", *IEEE Intelligent Vehicles Symposium (IV)*, ieeexplore.ieee.org, pp. 163–168.

K. Li and J. Malik (2016), “Learning to optimize”, *arXiv preprint arXiv:1606.01885*.

T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra (Sept. 2015), “Continuous control with deep reinforcement learning”, arXiv: `1509.02971 [cs.LG]`.

T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick (2014), “Microsoft COCO: Common Objects in Context”, *European Conference on Computer Vision (ECCV)*, Springer International Publishing, pp. 740–755.

Z. C. Lipton (June 2016), “The Mythos of Model Interpretability”, arXiv: `1606.03490 [cs.LG]`.

W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg (2016), “SSD: Single Shot MultiBox Detector”, *Computer Vision – ECCV 2016*, Springer International Publishing, pp. 21–37.

I. Loshchilov and F. Hutter (Aug. 2016), “SGDR: Stochastic Gradient Descent with Warm Restarts”, *ArXiv e-prints*, arXiv: `1608.03983 [cs.LG]`.

M. Lucic, K. Kurach, M. Michalski, S. Gelly, and O. Bousquet (Nov. 2017), “Are GANs Created Equal? A Large-Scale Study”, arXiv: `1711.10337 [stat.ML]`.

M. Madry, L. Bo, D. Kragic, and D. Fox (May 2014), “ST-HMP: Unsupervised Spatio-Temporal Feature Learning for Tactile Data”, *IEEE International Conference on Robotics and Automation (ICRA)*.

J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg (2017), “Dex-Net 2.0: Deep Learning to Plan Robust Grasps with Synthetic Point Clouds and Analytic Grasp Metrics”.

J. Mahler, F. T. Pokorny, B. Hou, M. Roderick, M. Laskey, M. Aubry, K. Kohlhoff, T. Kröger, J. Kuffner, and K. Goldberg (2016), “Dex-net 1.0: A cloud-based network of 3d objects for robust grasp planning using a multi-armed bandit model with correlated rewards”, *IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, pp. 1957–1964.

J. Mainprice, E. A. Sisbot, L. Jaillet, J. Cortés, R. Alami, and T. Siméon (May 2011), “Planning human-aware motions using a sampling-based costmap planner”, *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5012–5017.

J. Mainprice, N. Ratliff, and S. Schaal (2016), “Warping the workspace geometry with electric potentials for motion optimization of manipulation tasks”, *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

S. Mason, L. Righetti, and S. Schaal (2014), “Full dynamics LQR control of a humanoid robot: An experimental study on balancing and squatting”, *IEEE-RAS. International Conference on Humanoid Robots (Humanoids)*.

F. Meier, P. Hennig, and S. Schaal (2014a), “Efficient Bayesian local model learning for control”, *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2244–2249.

F. Meier, P. Hennig, and S. Schaal (2014b), “Incremental Local Gaussian Regression”, *Advances in Neural Information Processing Systems (NIPS)*.

F. Meier*,
bibinitperiod **Kappler***, and S. Schaal (2017), *Online Meta-Learning reference implementation*, `https://github.com/fmeier/online-meta-learning`.

G. Metta, P. Fitzpatrick, and L. Natale (2006), “YARP: Yet Another Robot Platform”, *International Journal on Advanced Robotics Systems (IJARS)*, pp. 43–48.

N. Minorsky. (May 1922), “DIRECTIONAL STABILITY OF AUTOMATICALLY STEERED BODIES”, *Journal of the American Society for Naval Engineers*, vol. 34, no. 2, pp. 280–309.

T. M. Mitchell et al. (1997), *Machine learning.* McGraw-Hill Boston, MA.

A. Morales, E. Chinellato, A. Fagg, and A. del Pobil (2004), “Using Experience for Assessing Grasp Reliability”, *Int. Jour. of Humanoid Robotics*.

H. P. Moravec (1990), “The Stanford Cart and the CMU Rover”, *Autonomous Robot Vehicles*, ed. by I. J. Cox and G. T. Wilfong, New York, NY: Springer New York, pp. 407–419.

K. P. Murphy (2012), *Machine learning: A Probabilistic Perspective*, Cambridge, MA, US: The MIT Press.

A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel (Sept. 2017), “Overcoming Exploration in Reinforcement Learning with Demonstrations”, arXiv: `1709.10089 [cs.LG]`.

V. Nair and G. E. Hinton (2010), “Rectified Linear Units Improve Restricted Boltzmann Machines”, *International Conference on Machine Learning (ICML)*, USA: Omnipress, pp. 807–814.

J. Nakanishi and S. Schaal (Dec. 2004), “Feedback error learning and nonlinear adaptive control”, *Neural networks: the official journal of the International Neural Network Society*, vol. 17, no. 10, pp. 1453–1465.

R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon (2011), “KinectFusion: Real-Time Dense Surface Mapping and Tracking”, *IEEE Int. Symposium on Mixed and Augmented Reality*.

I. Newton (1664), *Methodus fluxionum et serierum infinitarum*.

A. Y. Ng, D. Harada, and S. J. Russell (1999), “Policy Invariance Under Reward Transformations: Theory and Application to Reward Shaping”, *Proceedings of the Sixteenth International Conference on Machine Learning*, ICML ’99, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., pp. 278–287.

A. Y. Ng and M. I. Jordan (2002), “On Discriminative vs. Generative Classifiers: A comparison of logistic regression and naive Bayes”, *Advances in Neural Information Processing Systems (NIPS)*, ed. by T. G. Dietterich, S. Becker, and Z. Ghahramani, MIT Press, pp. 841–848.

A. Y. Ng and S. J. Russell (2000), “Algorithms for Inverse Reinforcement Learning”, *International Conference on Machine Learning (ICML)*, ICML ’00, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., pp. 663–670.

D. Nguyen-Tuong and J. Peters (2011), “Model learning for robot control: a survey”, *Cognitive Processing*, vol. 12, no. 4, pp. 319–340.

D. Nguyen-tuong and J. Peters (2010), “Using Model Knowledge for Learning Inverse Dynamics”, *IEEE International Conference on Robotics and Automation (ICRA)*.

D. Nguyen-Tuong, J. R. Peters, and M. Seeger (2008), “Local Gaussian process regression for real time online model learning”, *Advances in Neural Information Processing Systems (NIPS)*, pp. 1193–1200.

S. Niekum, S. Chitta, B. Marthi, S. Osentoski, and A. G. Barto (2013), “Incremental Semantically Grounded Learning from Demonstration”, *Robotics: Science and Systems (R:SS)*.

N. J. Nilsson (1984), *Shakey the robot*, tech. rep., SRI INTERNATIONAL MENLO PARK CA.

J. Nocedal and S. J. Wright (1999), *Nonlinear Optimization*, Springer.

Nof, Shimon Y. (1999), *Handbook of industrial robotics*, 2. ed., New York: Wiley.

K. Ogata (1990), *Modern Control Engineering*, 2nd, Upper Saddle River, NJ, USA: Prentice Hall PTR.

Open Physics Engine (n.d.), `www.ode.org`.

M. Oquab, L. Bottou, I. Laptev, and J. Sivic (June 2015), “Is object localization for free? - Weakly-supervised learning with convolutional neural networks”, *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 685–694.

R. Orfali and D. Harkey (1998), *Client/Server Programming with Java and CORBA (2Nd Ed.)* New York, NY, USA: John Wiley & Sons, Inc.

P. Pastor, M. Kalakrishnan, J. Binney, J. Kelly, L. Righetti, G. Sukhatme, and S. Schaal (2013a), “Learning Task Error Models for Manipulation”, *IEEE International Conference on Robotics and Automation (ICRA)*.

P. Pastor, M. Kalakrishnan, S. Chitta, E. Theodorou, and S. Schaal (2011a), “Skill Learning and Task Outcome Prediction for Manipulation”, *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3828–3834.

P. Pastor, M. Kalakrishnan, F. Meier, F. Stulp, J. Buchli, E. Theodorou, and S. Schaal (2013b), “From Dynamic Movement Primitives to Associative Skill Memories”, *Robotics and Autonomous Systems*, vol. 61.

P. Pastor, M. Kalakrishnan, L. Righetti, and S. Schaal (2012), “Towards Associative Skill Memories”, *IEEE-RAS. International Conference on Humanoid Robots (Humanoids)*.

P. Pastor, L. Righetti, M. Kalakrishnan, and S. Schaal (2011b), “Online movement adaptation based on previous sensor experiences”, *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer (2017), “Automatic differentiation in PyTorch”.

K. Pavel and S. Davi (Jan. 2013), “Algorithms for Efficient Computation of Convolution”, *Design and Architectures for Digital Signal Processing*, ed. by G. Ruiz, InTech.

J. Peters, M. Mistry, F. Udwadia, R. Cory, J. Nakanishi, and S. Schaa (Aug. 2005), “A unifying methodology for the control of robotic systems”, *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1824–1831.

G. Petkos, M. Toussaint, and S. Vijayakumar (2006), “Learning multiple models of non-linear dynamics for control under varying contexts”, *International Conference on Artificial Neural Networks*, Springer, pp. 898–907.

T. Petrič, A. Gams, L. Žlajpah, and A. Ude (2014), “Online learning of task-specific dynamics for periodic tasks”, *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1790–1795.

L. Pinto and A. Gupta (May 2016), "Supersizing self-supervision: Learning to grasp from 50K tries and 700 robot hours", *IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, pp. 3406–3413.

F. T. Pokorny and D. Kragic (2013), "Classical Grasp Quality Evaluation: New Theory and Algorithms", *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Tokyo, Japan.

L. S. Pontryagin (1962), *Mathematical Theory of Optimal Processes*, en, vol. 4, CRC Press.

M. Popović, G. Kootstra, J. A. Jørgensen, D. Kragic, and N. Krüger (2011), "Grasping Unknown Objects using an Early Cognitive Vision System for General Scene Understanding", *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, San Francisco, USA.

V. Pradeep, K. Konolige, and E. Berger (2014), "Calibrating a multi-arm multi-sensor robot: A Bundle Adjustment Approach", *Experimental Robotics*, vol. 79, Springer Tracts in Advanced Robotics, Springer Berlin Heidelberg, pp. 211–225.

M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng (2009), "ROS: an open-source Robot Operating System", *ICRA workshop on open source software*, vol. 3, p. 5.

J. R. Quinlan (Mar. 1986), "Induction of Decision Trees", *Machine learning*, vol. 1, no. 1, pp. 81–106.

A. Rai, F. Meier, A. Ijspeert, and S. Schaal (Nov. 2014), "Learning coupling terms for obstacle avoidance", *IEEE-RAS. International Conference on Humanoid Robots (Humanoids)*, pp. 512–518.

A. Rai, G. Sutanto, F. Meier, and S. Schaal (2016), "Learning Feedback Terms for Reactive Planning and Control", *IEEE International Conference on Robotics and Automation (ICRA)*.

J. Raphson (1690), *Analysis aequationum universalis.* London.

C. E. Rasmussen and C. Williams (2006), *Gaussian Processes for Machine Learning*, MIT Press.

N. D. Ratliff, D. M. Bradley, J. A. Bagnell, and J. Chestnutt (2007), "Boosting Structured Prediction for Imitation Learning", *Advances in Neural Information Processing Systems (NIPS)*, ed. by B. Schölkopf, J. C. Platt, and T. Hoffman, MIT Press, pp. 1153–1160.

N. Ratliff, M. Toussaint, and S. Schaal (2015), "Understanding the geometry of workspace obstacles in Motion Optimization", *IEEE International Conference on Robotics and Automation (ICRA)*, IEEE.

N. Ratliff, M. Zucker, J. A. (Bagnell, and S. Srinivasa (2009), "CHOMP: Gradient Optimization Techniques for Efficient Motion Planning", *IEEE International Conference on Robotics and Automation (ICRA)*.

J. Redmon and A. Angelova (May 2015), "Real-time grasp detection using convolutional neural networks", *IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, pp. 1316–1322.

J. Redmon and A. Farhadi (July 2017), "YOLO9000: Better, Faster, Stronger", *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, ieeexplore.ieee.org, pp. 6517–6525.

J. Reif and M. Sharir (1985), "Motion Planning in the Presence of Moving Obstacles", *Proceedings of the 26th Annual Symposium on Foundations of Computer Science*, SFCS '85, Washington, DC, USA: IEEE Computer Society, pp. 144–154.

S. Ren, K. He, R. Girshick, and J. Sun (June 2017), "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", en, *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 6, pp. 1137–1149.

D. J. Rezende and S. Mohamed (May 2015), "Variational Inference with Normalizing Flows", arXiv: `1505.05770 [stat.ML]`.

C. Richter, A. Bry, and N. Roy (2016), "Polynomial Trajectory Planning for Aggressive Quadrotor Flight in Dense Indoor Environments", *Int. Symposium on Robotics Research (ISRR)*, ed. by M. Inaba and P. Corke, Cham: Springer International Publishing, pp. 649–666.

M. Riedmiller and H. Braun (1993), "A direct adaptive method for faster backpropagation learning: the RPROP algorithm", *IEEE International Conference on Neural Networks*.

M. Riedmiller, R. Hafner, T. Lampe, M. Neunert, J. Degrave, T. Van de Wiele, V. Mnih, N. Heess, and J. T. Springenberg (Feb. 2018), "Learning by Playing - Solving Sparse Reward Tasks from Scratch", arXiv: `1802.10567 [cs.LG]`.

L. Righetti, M. Kalakrishnan, P. Pastor, J. Binney, J. Kelly, R. Voorhies, G. Sukhatme, and S. Schaal (2014), "An autonomous manipulation system based on force control and optimization", *Autonomous Robots*, vol. 36, no. 1-2, pp. 11–30.

J. Romano, K. Hsiao, G. Niemeyer, S. Chitta, and K. Kuchenbecker (Dec. 2011), "Human-Inspired Robotic Grasp Control With Tactile Sensing", *IEEE Trans. on Robotics*, vol. 27, no. 6, pp. 1067–1079.

F. Rosenblatt (Nov. 1958), "The perceptron: a probabilistic model for information storage and organization in the brain", en, *Psychological review*, vol. 65, no. 6, pp. 386–408.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams (Oct. 1986), “Learning representations by back-propagating errors”, *Nature*, vol. 323, p. 533.

O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei (2015), “ImageNet Large Scale Visual Recognition Challenge”, *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252.

R. B. Rusu and S. Cousins (2011), “3d is here: Point cloud library (pcl)”, *IEEE International Conference on Robotics and Automation (ICRA)*.

J. A. Rytz, L.-P. Ellekilde, D. Kraft, H. G. Petersen, and N. Krüger (2013), “On Transferability of grasp-affordances in data-driven grasping”, *22nd Int. Workshop on Robotics in Alpe-Adria-Danube Region (RAAD)*.

A. Saxena, J. Driemeyer, and A. Y. Ng (Feb. 2008a), “Robotic Grasping of Novel Objects using Vision”, *The International journal of robotics research*, vol. 27, no. 2, pp. 157–173.

A. Saxena, J. Driemeyer, and A. Y. Ng (2008b), “Robotic Grasping of Novel Objects using Vision”, *The International Journal of Robotics Research (IJRR)*.

S. Schaal and C. G. Atkeson (1998), “Constructive incremental learning from only local information”, *Neural computation*, vol. 10, no. 8.

J. Schmidhuber (2015), “Deep Learning in Neural Networks: An Overview”, *Neural Networks*, vol. 61, pp. 85–117.

J. Schmidhuber (1987), “Evolutionary principles in self-referential learning”, *On learning how to learn: The meta-meta-... hook.) Diploma thesis, Institut f. Informatik, Tech. Univ. Munich*.

T. Schmidt, R. Newcombe, and D. Fox (July 2014), “DART: Dense Articulated Real-Time Tracking”, *Proceedings of Robotics: Science and Systems*, Berkeley, USA.

T. Schmidt, R. Newcombe, and D. Fox (Oct. 2015), “DART: dense articulated real-time tracking with consumer depth cameras”, *Autonomous robots*, vol. 39, no. 3, pp. 239–258.

B. Scholkopf and A. J. Smola (2001), *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, Cambridge, MA, USA: MIT Press.

J. Schulman, J. Ho, A. X. Lee, I. Awwal, H. Bradlow, and P. Abbeel (2013), “Finding Locally Optimal, Collision-Free Trajectories with Sequential Convex Optimization”, *Robotics: Science and Systems (R:SS)*.

J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel (Aug. 2014), “Motion planning with sequential convex optimization and convex collision checking”, *The International Journal of Robotics Research (IJRR)*, vol. 33, no. 9, pp. 1251–1270.

N. Schweighofer and K. Doya (2003), “Meta-learning in reinforcement learning”, *Neural Networks*, vol. 16, no. 1, pp. 5–9.

E. Shelhamer, J. Long, and T. Darrell (Apr. 2017), “Fully Convolutional Networks for Semantic Segmentation”, en, *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 4, pp. 640–651.

K. B. Shimoga (1996), “Robot grasp synthesis algorithms: A survey”, *The International Journal of Robotics Research (IJRR)*, vol. 15, no. 3, pp. 230–266.

Y. Shirai and H. Inoue (June 1973), “Guiding a robot by visual feedback in assembling tasks”, *Pattern recognition*, vol. 5, no. 2, pp. 99–108.

B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo (2010), *Robotics: Modelling, Planning and Control*, Second, Springer.

H. T. Siegelmann and E. D. Sontag (Feb. 1995), “On the Computational Power of Neural Nets”, *Journal of Computer and System Sciences*, vol. 50, no. 1, pp. 132–150.

O. Sigaud, C. Salaün, and V. Padois (2011), “On-line regression algorithms for learning mechanical models of robots: a survey”, *Robotics and Autonomous Systems*, vol. 59, no. 12, pp. 1115–1129.

K. Simonyan and A. Zisserman (2014), “Very Deep Convolutional Networks for Large-Scale Image Recognition”, *CoRR*, vol. abs/1409.1556.

J. Sinapov, C. Schenck, K. Staley, V. Sukhoy, and A. Stoytchev (May 2014), “Grounding semantic categories in behavioral interactions: Experiments with 100 objects”, *Robotics: Science and Systems (R:SS)*, vol. 62, no. 5, pp. 632–645.

D. Song, C. H. Ek, K. Hübner, and D. Kragic (2011), “Multivariate discretization for Bayesian Network structure learning in robot grasping”, *IEEE International Conference on Robotics and Automation (ICRA)*.

R. Stengel (1994), *Optimal Control and Estimation*, Dover, New York.

R. S. Sutton and A. G. Barto (1998a), *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning)*, The MIT Press.

R. S. Sutton and A. G. Barto (1998b), *Reinforcement learning: An introduction*, vol. 1, MIT press Cambridge.

C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi (2017), "Inception-v4, inception-resnet and the impact of residual connections on learning", *AAAI*.

Y. Tang (June 2013a), "Deep Learning using Linear Support Vector Machines", arXiv: `1306.0239 [cs.LG]`.

Y. Tang (2013b), "Deep learning using linear support vector machines", *arXiv preprint arXiv:1306.0239*.

E. Theodorou, J. Buchli, and S. Schaal (2010), "Reinforcement Learning of Motor Skills in High Dimensions: a Path Integral Approach", *IEEE International Conference on Robotics and Automation (ICRA)*.

T. Tieleman and G. Hinton (Apr. 2012), "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude.", *COURSERA: Neural Networks for Machine Learning*.

M. Toussaint and C. Goerick (2010), "A bayesian view on motor control and planning", *From Motor Learning to Interaction Learning in Robots*, Springer, pp. 227–252.

M. Toussaint and S. Vijayakumar (2005), "Learning discontinuities with products-of-sigmoids for switching between local models", *International Conference on Machine Learning (ICML)*.

I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun (2005), "Large Margin Methods for Structured and Interdependent Output Variables", *Journal of machine learning research: JMLR*, vol. 6, no. Sep, pp. 1453–1484.

TurboSquid: 3D Models for Professionals (n.d.), `www.turbosquid.com`.

C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. N. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, M. Gittleman, S. Harbaugh, M. Hebert, T. M. Howard, S. Kolski, A. Kelly, M. Likhachev, M. McNaughton, N. Miller, K. Peterson, B. Pilnick, R. Rajkumar, P. Rybski, B. Salesky, Y.-W. Seo, S. Singh, J. Snider, A. Stentz, W. " Whittaker, Z. Wolkowicki, J. Ziglar, H. Bae, T. Brown, D. Demitrish, B. Litkouhi, J. Nickolaou, V. Sadekar, W. Zhang, J. Struble, M. Taylor, M. Darms, and D. Ferguson (Aug. 2008), "Autonomous driving in urban environments: Boss and the Urban Challenge", *Journal of Field Robotics*, vol. 25, no. 8, pp. 425–466.

N. Vahrenkamp, D. Berenson, T. Asfour, J. Kuffner, and R. Dillmann (Oct. 2009a), "Humanoid motion planning for dual-arm manipulation and re-grasping tasks", *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2464–2470.

N. Vahrenkamp, C. Böge, K. Welke, T. Asfour, J. Walter, and R. Dillmann (Dec. 2009b), "Visual servoing for dual arm motions on a humanoid robot", *IEEE-RAS. International Conference on Humanoid Robots (Humanoids)*, pp. 208–214.

N. Vahrenkamp, M. Wächter, M. Kröhnert, K. Welke, and T. Asfour (2015), "The robot software framework armarx", *it-Information Technology*, vol. 57, no. 2, pp. 99–111.

V. N. Vapnik (1999), "An overview of statistical learning theory", *IEEE transactions on neural networks a publication of the IEEE Neural Networks Council.*

N. Vasilache, J. Johnson, M. Mathieu, S. Chintala, S. Piantino, and Y. LeCun (Dec. 2014), "Fast Convolutional Nets With fbfft: A GPU Performance Evaluation", arXiv: `1412.7580 [cs.LG]`.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. U. Kaiser, and I. Polosukhin (2017), "Attention is All you Need", *Advances in Neural Information Processing Systems 30*, ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Curran Associates, Inc., pp. 5998–6008.

S. Vijayakumar and S. Schaal (2000), "Locally weighted projection regression: Incremental real time learning in high dimensional space", *International Conference on Machine Learning (ICML)*, pp. 1079–1086.

R. Vilalta and Y. Drissi (2002), "A perspective view and survey of meta-learning", *Artificial Intelligence Review*, vol. 18, no. 2.

M. Wächter, S. Ottenhaus, M. Kröhnert, N. Vahrenkamp, and T. Asfour (2016), "The ArmarX Statechart Concept: Graphical Programming of Robot Behaviour", *Frontiers in Robotics and AI*, vol. 3, p. 33.

R. S. Wallace, A. Stentz, C. E. Thorpe, H. P. Moravec, W. Whittaker, and T. Kanade (1985), "First Results in Robot Road-Following", *International Joint Conference on Artificial Intelligence (IJCAI)*, Citeseer, pp. 1089–1095.

J. Weisz and P. Allen (2012), "Pose Error Robust Grasping from Contact Wrench Space Metrics", *IEEE International Conference on Robotics and Automation (ICRA).*

WHO (2011), *Global Health and Aging*, `http://www.who.int/ageing/publications/global_health/en/`.

W. Wohlkinger, A. Aldoma, R. B. Rusu, and M. Vincze (2012), "3DNet: Large-scale object class recognition from CAD models", *IEEE Int. Conf. on Robotics and Automation, ICRA*, St. Paul, Minnesota, USA, pp. 5384–5391.

D. H. Wolpert and W. G. Macready (Apr. 1997), "No free lunch theorems for optimization", *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82.

D. M. Wolpert and M. Kawato (1998), “Multiple paired forward and inverse models for motor control”, *Neural networks*, vol. 11, no. 7.

M. Wüthrich, P. Pastor, M. Kalakrishnan, J. Bohg, and S. Schaal (2013), “Probabilistic object tracking using a range camera”, *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

K. Yamazaki, R. Ueda, S. Nozawa, M. Kojima, K. Okada, K. Matsumoto, M. Ishikawa, I. Shimoyama, and M. Inaba (Aug. 2012), “Home-Assistant Robot for an Aging Society”, *Proceedings of the IEEE*, vol. 100, no. 8, pp. 2429–2441.

Y. Yang, V. Ivan, Z. Li, M. Fallon, and S. Vijayakumar (2016), “iDRM: Humanoid motion planning with realtime end-pose selection in complex environments”, *IEEE-RAS. International Conference on Humanoid Robots (Humanoids)*.

M. D. Zeiler (Dec. 2012), “ADADELTA: An Adaptive Learning Rate Method”, *ArXiv e-prints*, arXiv: `1212.5701 [cs.LG]`.

Z. Zhang (Feb. 2012), “Microsoft Kinect Sensor and Its Effect”, *IEEE Multimedia*, vol. 19, no. 2, pp. 4–10.

M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa (2013), “CHOMP: Covariant hamiltonian optimization for motion planning”, *The International Journal of Robotics Research (IJRR)*, vol. 32.